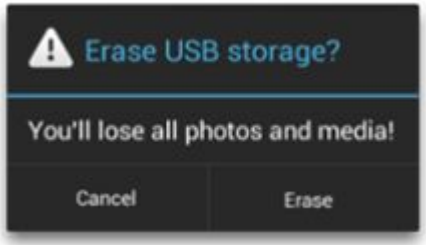
CUADROS DE DIÁLOGO Y VENTANAS MODALES

Programación multimedia
y dispositivos móviles
2º DAM

ÍNDICE DE CONTENIDOS

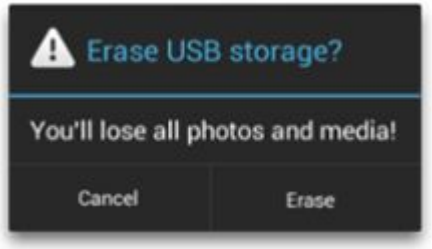
1. INTRODUCCIÓN

- Un diálogo es una ventana pequeña que le indica al usuario que debe tomar una decisión o ingresar información adicional.
- Un diálogo no ocupa toda la pantalla y generalmente se usa para eventos modales que requieren que los usuarios realicen alguna acción para poder continuar.

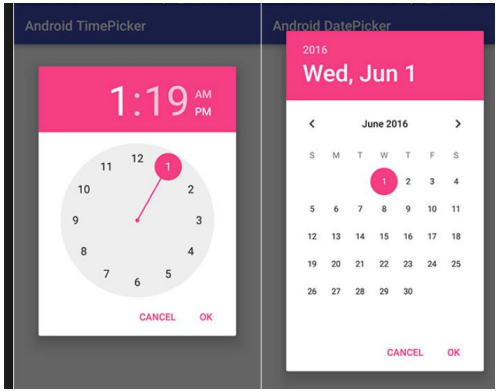


1. INTRODUCCIÓN

- Existen diferentes tipos de cuadros de diálogo básicos en Android:
  - **AlertDialog** → Es el clásico cuadro de diálogo en el que se ofrece un texto informativo y varias opciones como OK/Cancel. En android, este tipo de diálogos soportan contener diferentes elementos como podría ser una lista o cualquier otro view modificable.



  - **DatePicker/TimePicker** → Del mismo modo, podemos usar algunos diálogos que el sistema nos proporciona por defecto, en este caso, son dos cuadros de diálogo que solicitan al usuario la fecha o la hora de forma que garantizamos una entrada correcta (Sin errores) en ambos casos.



2. ALERTDIALOG

Para crear un nuevo diálogo de alerta crearemos una nueva clase extendiendo de la clase DialogFragment y, en el método onCreateDialog definiremos nuestro cuadro de diálogo.

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

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

        builder.setTitle(getResources().getString(R.string.titleDialog));
        builder.setMessage(getResources().getString(R.string.mensajeDialog));
        builder.setPositiveButton(getResources().getString(R.string.positiveButton), listener: this);
        builder.setNeutralButton(getResources().getString(R.string.neutralButton), listener: this);
        builder.setNegativeButton(getResources().getString(R.string.negativeButton), listener: this);

        return builder.create();
    }
```

2. ALERTDIALOG

- Como se observa en la imagen, debemos crear el objeto de la clase AlertDialog.Builder, en la que estableceremos el contexto de la Actividad en la cual se va a mostrar.
- Después se establecerán sus atributos (titulo, mensaje) y se indicarán los botones que va a tener.
    - Hay tres botones disponibles, pero no es necesario mostrarlos todos.
- Por último, se devolverá la creación del diálogo.

2. ALERTDIALOG

Este Dialog tiene un evento para realizar las opciones dependiendo del botón que se ha pulsado.

El parámetro which indica el botón que se ha pulsado.

```
@Override
public void onClick(DialogInterface dialog, int which) {
    //positiveButton -> which -1
    //negativeButton -> which -2
    //neutralButton -> which -3
    switch (which) {
        case -1:
            Toast.makeText(getContext(),  text: "ACEPTAR", Toast.LENGTH_SHORT).show();
            break;
        case -2:
            Toast.makeText(getContext(),  text: "CANCELAR", Toast.LENGTH_SHORT).show();
            break;
        case -3:
            Toast.makeText(getContext(),  text: "NO HACER NADA", Toast.LENGTH_SHORT).show();
            break;
    }
}
```

2. ALERTDIALOG

Para mostrar el Dialog en una actividad, es necesario:

- Crear el objeto de la clase.
- Llamar a su método onCreateDialog, mediante el método show().
- El método show necesita 2 parámetros:
  - contexto del fragmento en el que se va a mostrar: getSupportFragmentManager.
  - tag un nombre para identificarlo.

2. ALERTDIALOG

Cuando se cierra un cuadro de diálogo, normalmente se necesita que se realicen algunas acciones en otras partes de nuestra app, para ello, el método onDismiss(...) se lanza cuando el usuario pulsa cualquiera de los botones del diálogo cerrandolo.

En nuestro ejemplo, la acción será que MainActivity recargue la vista. Esto se realizará de la misma forma que el callback de los fragment.

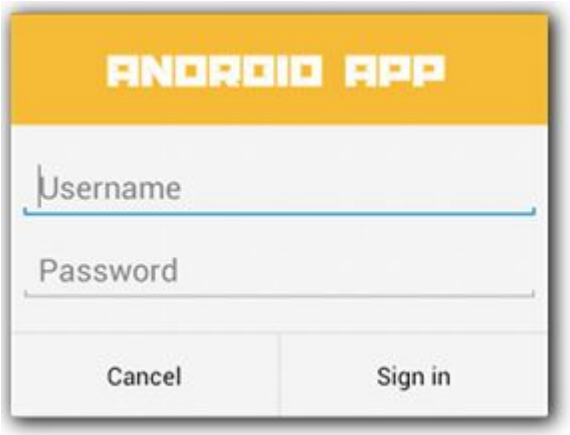
```
@Override
public void onDismiss(@NonNull DialogInterface dialog) {
    super.onDismiss(dialog);

    if (listener != null)
        this.listener.recargar();
}
```

3. VENTANAS MODALES

Es posible utilizar las funcionalidades que nos ofrece la clase AlertDialog para crear ventanas de diálogo personalizadas. Esto nos permite mostrar cualquier layout de forma modal.

Esto es especialmente útil cuando queremos mostrar una nueva ventana relacionada con un elemento de la ventana anterior, pues acentúa la interrelación entre ambas.



3. VENTANAS MODALES

Para crear un cuadro de diálogo personalizado, tan solo debemos definir un layout XML como lo hacemos para un Activity, Fragment o Adaptador y cargarlo dentro de nuestro cuadro de diálogo mediante el método setView() dentro de onCreateDialog().

```
public class ModificarDialog extends DialogFragment implements AlertDialog.OnClickListener {

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

        LayoutInflater inflater = requireActivity().getLayoutInflater();
        View view = inflater.inflate(R.layout.modificar_dialog,  root: null);
        builder.setView(view);

        builder.setTitle(getResources().getString(R.string.titleDialogModificar));
        builder.setMessage(getResources().getString(R.string.mensajeDialogModificar));
        builder.setPositiveButton(getResources().getString(R.string.positiveButton),  listener: this);
        builder.setNeutralButton(getResources().getString(R.string.negativeButton),  listener: this);
        builder.setNegativeButton(getResources().getString(R.string.neutralButton),  listener: this);

        return builder.create();
    }
```

Como vemos, es el mismo esquema que usábamos anteriormente, salvo que añadimos el método setView(View v) a nuestro objeto builder.

4. DATEPICKERDIALOG

Android nos proporciona un tipo de diálogo con un calendario para poder establecer una fecha, la clase es DatePickerDialog.

El constructor de este tipo de diálogos, necesita los siguientes parámetros:

- Context → contexto de la actividad donde se va a mostrar el diálogo.
- DatePickerDialog.OnDateSetListener → evento a ejecutarse al pulsar en el botón aceptar.
- 3 enteros, en el siguiente orden → año, mes y día para inicializar el calendario.

Para inicializar el calendario a la fecha actual, necesitamos crear una instancia de la clase Calendar, y acceder a sus atributos.

```
Calendar calendar = Calendar.getInstance();

int year = calendar.get(Calendar.YEAR);
int month = calendar.get(Calendar.MONTH);
int day = calendar.get(Calendar.DAY_OF_MONTH);
```

4. DATEPICKERDIALOG

Si la actividad que va a mostrar el dialog implementa la interface DatePickerDialog.OnDateSetListener, la forma de crear el dialog será la siguiente.

```
DatePickerDialog date = new DatePickerDialog( context: this,  listener: this, year, month, day);
date.show();
```

Para poner la fecha seleccionada en un EditText, lo haremos en el evento onDateSet(). Los datos que recibe por parámetro son los de la fecha seleccionada.

```
@Override
public void onDateSet(DatePicker view, int year, int month, int dayOfMonth) {
    fecha.setText(String.format("%02d/%02d/%4d", dayOfMonth,month,year));
}
```

5. TIMEPICKERDIALOG

Android nos proporciona un tipo de diálogo con un calendario para poder establecer una hora, la clase es TimePickerDialog.

El constructor de este tipo de diálogos, necesita los siguientes parámetros:

- Context → contexto de la actividad donde se va a mostrar el diálogo.
- TimePickerDialog.OnTimeSetListener → evento a ejecutarse al pulsar en el botón aceptar.
- 2 enteros, en el siguiente orden → hora y minutos para inicializar el calendario.
- boolean → true si queremos formato 24 horas y false si queremos formato 12 horas.

Para inicializar el calendario a la hora actual, necesitamos crear una instancia de la clase Calendar, y acceder a sus atributos.

```
Calendar calendar = Calendar.getInstance();
int hour = calendar.get(Calendar.HOUR);
int minute = calendar.get(Calendar.MINUTE);
```

5. TIMEPICKERDIALOG

Si la actividad que va a mostrar el dialog implementa la interface TimePickerDialog.OnTimeSetListener, la forma de crear el dialog será la siguiente.

```
TimePickerDialog hora = new TimePickerDialog( context: this,  listener: this, hour, minute,  is24HourView: true);
hora.show();
```

Para poner la fecha seleccionada en un EditText, lo haremos en el evento onTimeSet(). Los datos que recibe por parámetro son los de la hora seleccionada.

```
@Override
public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
    hora.setText(String.format("%02d:%02d", hourOfDay, minute));
}
```

6. DOCUMENTACIÓN

Más información:

https://developer.android.com/guide/topics/ui/dialogs.html?hl=es-419

7. DUDAS Y PREGUNTAS