ALERT DIALOGS

En esta app, crearemos un activity, que tendrá varios botones, en el que cada uno de ellos nos llevará a un ejemplo de uso de los Alert Dialogs más comunes.

Un **AlertDialog** en Android, es una ventana emergente (diálogo) que interrumpe la interacción del usuario con la aplicación para mostrarle un mensaje importante, pedirle una confirmación o que realice una acción. Este tipo de diálogo es comúnmente utilizado para situaciones que requieren una respuesta del usuario, como confirmar la eliminación de un archivo, advertir sobre una acción irreversible, pedir opciones o proporcionar información adicional etc.

Componentes de un AlertDialog:

1. **Título (Title):** Una breve descripción o título de lo que trata el diálogo.
2. **Mensaje (Message):** El contenido principal o la información que se desea transmitir.
3. **Botones (Buttons):** Opciones que el usuario puede seleccionar. Un AlertDialog suele tener hasta tres tipos de botones:
    - **Positive Button:** Generalmente representa una acción afirmativa, como "Aceptar" o "Sí".
    - **Negative Button:** Representa una acción negativa, como "Cancelar" o "No".
    - **Neutral Button:** Puede ser utilizado para acciones adicionales como "Más tarde".

Ejemplo básico de un AlertDialog en Android (Kotlin):

```
val builder = AlertDialog.Builder(this)
builder.setTitle("Confirmación")
builder.setMessage("¿Estás seguro de que deseas eliminar este archivo?")
builder.setPositiveButton("Sí") { dialog, which ->
    // Acción si el usuario elige "Sí"
}
builder.setNegativeButton("No") { dialog, which ->
    // Acción si el usuario elige "No"
}
builder.setNeutralButton("Cancelar") { dialog, which ->
    // Acción si el usuario elige "Cancelar"
}
builder.show()
```

Funcionalidades avanzadas:

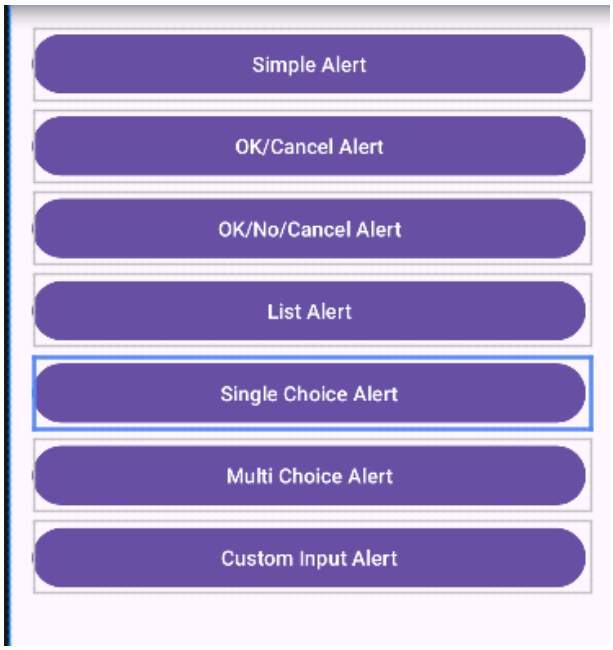
- **Input dialogs**: Puedes añadir campos de texto dentro de un AlertDialog para permitir que el usuario ingrese datos.
- **Listas de selección**: Se pueden añadir listas de elementos (con checkboxes o radio buttons) para que el usuario seleccione opciones.

El AlertDialog interrumpe la interacción con la actividad actual hasta que el usuario responde, después de lo cual el diálogo se cierra automáticamente.

Para comprender los diferentes usos, crearemos un proyecto sobre el que explicaremos diferentes aspectos:

1. Crea un proyecto: File→New Project → Empty Views Activity , lo llamaremos, por ejemplo, Dialogos.

2. Crea un activity con el siguiente aspecto (puedes coger el código para tardar menos, puesto en recursos). Los botones están alineados y usando una cadena para que se repartan el espacio disponible.



3. Una vez creado el layout, añadiremos el siguiente código en el OnCreate, usando binding. En cada uno de esos métodos, implementaremos un ejemplo de cada tipo de Alert Dialog.

   NOTA: En recursos he dejado un ejemplo de algunos tipos de Alert Dialog.

```
binding.btnSimpleAlert.setOnClickListener { showSimpleAlertDialog() }
binding.btnOkCancel.setOnClickListener { showOkCancelAlertDialog() }
binding.btnOkNoCancel.setOnClickListener { showOkNoCancelAlertDialog() }
binding.btnListAlert.setOnClickListener { showListAlertDialog() }
binding.btnSingleChoiceAlert.setOnClickListener { showSingleChoiceAlertDialog() }
binding.btnMultiChoiceAlert.setOnClickListener { showMultiChoiceAlertDialog() }
binding.btnCustomInput.setOnClickListener { showCustomInputAlertDialog() }
```

4. En el primer ejemplo, además de establecer un título, un mensaje y la acción del botón aceptar, añadiré un icono, que sirva como ejemplo para algunas de las personalizaciones que podemos realizar directamente con el AlertDialog.

```
private fun showSimpleAlertDialog() {
    AlertDialog.Builder(this)
        .setTitle("Simple Alert")
        .setMessage("This is a simple alert dialog.")
        .setPositiveButton("OK", null)
        .setIcon(R.drawable.cohete)
        .show()
}
```

Ese icono previamente lo hemos debido añadir a la carpeta drawable. (esa propiedad es opcional, la omites no sale icono)

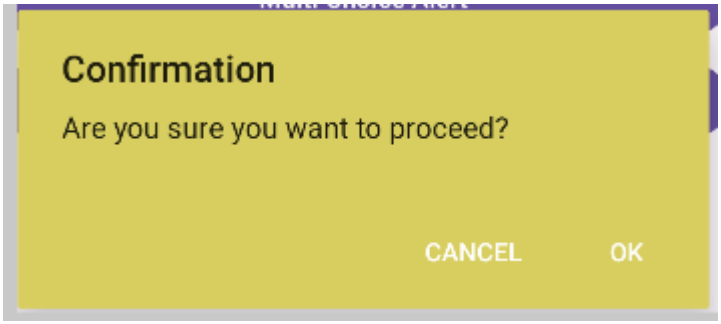
5. En el siguiente ejemplo, además de darle funcionalidad a los botones positivo y negativo, pondré un ejemplo de cómo darle color de fondo a mi AlertDialog directamente, sin tener que crear otra ventana personalizada.
   a. En estilos (themes), añade un estilo propio (previamente he creado el color oro, en el archivo colors.

```
<style name="MyAlertDialogTheme"
parent="Theme.AppCompat.Light.Dialog.Alert">
    <item name="android:background">@color/oro</item>
</style>
```

   b. Ahora crea el AlertDialog con este constructor en el que se hace referencia al estilo:

```
private fun showOkCancelAlertDialog() {
    AlertDialog.Builder(ContextThemeWrapper(this,R.style.MyAlertDialogTheme))
        .setTitle("Confirmation")
        .setMessage("Are you sure you want to proceed?")
        .setPositiveButton("OK") { dialog, which ->
            // Handle OK button click
            Toast.makeText(this, "OK clicked", Toast.LENGTH_SHORT).show()
        }
        .setNegativeButton("Cancel") { dialog, which ->
            // Handle Cancel button click
            Toast.makeText(this, "Cancel clicked", Toast.LENGTH_SHORT).show()
        }
        .show()
}
```

El AlertDialog ha quedado de la siguiente manera:



Investiga ahora como podrías cambiar el color de los mensajes.

6. Cabe destacar que, en el último ejemplo, lo que hacemos en abrir un layout personalizado. Para ello hay que crear un layout en nuestro proyecto de la siguiente manera:

a. Dentro de la carpeta layout →botón derecho→New→XML→LayoutXML File.

b. Dar el nombre deseado (por ejemplo: dialogo_personalizado.xml) y añádele un icono y un Plain Text (por ejemplo).

c. Ponle un color de fondo al layout.

d. Ahora podremos seguir el código, en el que se abre ese layout y se captura lo que se ha escrito en un Plain Text que habíamos puesto en el mismo. En mi caso ha quedado así: