29.- Diálogos

Los diálogos son elementos en la interfaz de Android que se sobreponen a las actividades para exigir la confirmación de una acción o solicitar información al usuario.

Estos cuadros de diálogo se caracterizan por interrumpir una tarea del usuario, alterando el foco de la aplicación e invitándolo de forma intrusiva a que vea cierta información, decida ante una circunstancia critica o especifique datos.

En principio, los diálogos de Android los podremos utilizar con distintos fines:

- Mostrar un mensaje.
- Pedir una confirmación rápida.
- Solicitar al usuario una elección (simple o múltiple) entre varias alternativas.

El uso actual de los diálogos en Android se basa en **fragments**. En este caso nos vamos a basar en la clase `DialogFragment`, para instanciar los dialogo. Aunque es `Dialog` la clase que genera la interfaz, `DialogFragment` es quien permite controlar los eventos de forma fluida. Por ello no se recomienda crear objetos de clase base.

Para crear un diálogo lo primero que haremos será crear una nueva clase que herede de `DialogFragment` y sobrescribiremos uno de sus métodos `onCreateDialog()`, que será el encargado de construir el diálogo con las opciones que necesitemos.

La forma de construir cada diálogo dependerá de la información y funcionalidad que necesitemos.

Dialogo de Alerta

Este tipo de diálogo se limita a mostrar un título, un mensaje sencillo al usuario, y un único botón de Ok para confirmar su lectura.

En los botones podemos ver acciones como la de aceptación (Botón positivo) que determina que el usuario está de acuerdo con la acción. La cancelación (Botón negativo) para evitar la realización de una acción, y la neutralidad (Botón neutro) para determinar que aún no se está listo para proseguir o cancelar la acción.

Un dialogo de alerta o construiremos mediante la clase `AlertDialog`, y más concretamente con la su subclase `AlertDialog.Builder`, de forma similar a las notificaciones de barra de estado. Su utilización es muy sencilla, bastará con crear un objeto de tipo `AlertDialog.Builder` y establecer las propiedades del diálogo mediante sus métodos **`set*()`**. Siendo los más frecuentes:

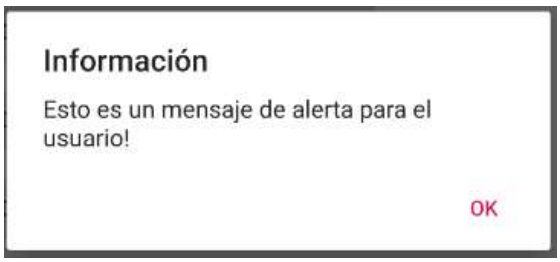
- `setTitle()`: Título del diálogo.
- `setMessage()`: Contenido del mensaje que se quiere transmitir.
- `setPositiveButton()`: Crea una instancia del botón de confirmación. El primer parámetro que recibe es el texto del botón y el segundo una escucha `OnClickListener` para determinar qué acción se tomará al ser pulsado.

Dentro de este evento, nos limitamos a cerrar el dialogo mediante su método `cancel()`, aunque podríamos realizar cualquier otra acción.

Para lanzar este diálogo por ejemplo desde nuestra actividad principal, obtendríamos una referencia al *Fragment Manager* mediante una llamada a `getSupportFragmentManager()`, creamos un nuevo objeto de tipo `DialogoAlerta` y por último mostramos el diálogo mediante el método `show()` pasándole la referencia al fragment manager y una etiqueta identificativa del diálogo.

```
package com.example.ejdialogos;

import android.app.Dialog;
import android.content.DialogInterface;
import android.os.Bundle;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AlertDialog;
import androidx.fragment.app.DialogFragment;

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

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

        builder.setMessage("Esto es un mensaje de alerta para el usuario!")
                .setTitle("Información")
                .setPositiveButton("OK", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        dialog.cancel();
                    }
                })

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

El aspecto de este dialogo de alerta será:



Dialogo de Confirmación

Un diálogo de confirmación es muy similar al anterior, con la diferencia de que lo utilizaremos para solicitar al usuario que nos confirme una determinada acción, pudiendo incluir hasta 3 botones de confirmación.

En los botones podemos ver acciones como la de aceptación (Botón positivo) que determina que el usuario está de acuerdo con la acción. La cancelación (Botón negativo) para evitar la realización de una acción, y la neutralidad (Botón neutro) para determinar que aún no se está listo para proseguir o cancelar la acción.

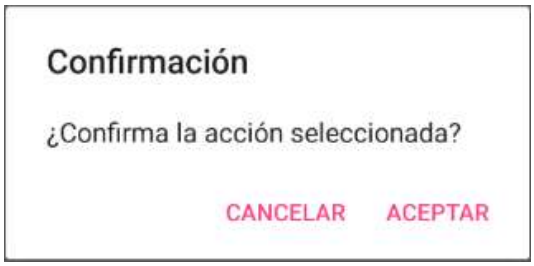
La implementación de estos diálogos es prácticamente igual a la comentada para las alertas, salvo que en esta ocasión se añadirán hasta dos botones más

- `setNegativeButton()`:Ídem al botón de confirmación pero con la decisión negativa.
- `setNeutralButton()`: Genera la escucha para el botón neutro. Al igual que los dos anteriores recibe el nombre y la escucha.

Veamos un ejemplo:

```
package com.example.ejdialogos;

import android.app.Dialog;
import android.content.Context;
import android.content.DialogInterface;
import android.os.Bundle;
import android.util.Log;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AlertDialog;
import androidx.fragment.app.DialogFragment;


public class DialogoConfirmacion extends DialogFragment {

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

        builder.setMessage("¿Confirma la acción seleccionada?")
                .setTitle("Confirmación")
                .setPositiveButton("ACEPTAR",
                        new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        //dialog.cancel();
                        Log.i("Dialogos", "Confirmación Aceptada.");
                        listener.onPossitiveButtonClick();
                    }
                })
                .setNegativeButton("CANCELAR",
                        new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        //dialog.cancel();
                        Log.i("Dialogos", "Confirmación Denegada");
                        listener.onNegativeButtonClick();
                    }
                });
        return builder.create();
    }

}
```

En este caso, generamos un par de mensajes de log para verificar qué botón del diálogo de confirmación se ha seleccionado. Y su aspecto será:



Dialogo de Selección

En los diálogos de selección, como su propio nombre indica, el usuario podrá elegir entre una lista de diferentes opciones.

Para ello también utilizaremos la clase `AlertDialog`, pero esta vez no asignaremos ningún mensaje ni definiremos las acciones a realizar por cada botón individual, sino que directamente 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.

Aquí disponemos de 3 opciones:

1) El resultado será similar a una lista,
2) Similar a los radioButtons y
3) Similar a checkbox.

En la OPCION 1, el diálogo permite al usuario elegir una opción entre las opciones disponibles que se muestran en pantalla. Pero, ¿y si quisiéramos recordar cuál es la opción u opciones seleccionadas por el usuario para que aparezcan marcadas al visualizar de nuevo el cuadro de diálogo? Para ello podemos utilizar los métodos `setSingleChoiceItems()` (OPCION 2) o `setMultiChoiceItems()` (OPCION 3), en vez de el `setItems()`.

```
package com.example.ejdialogos;

import android.app.Dialog;
import android.content.DialogInterface;
import android.os.Bundle;
import android.util.Log;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AlertDialog;
import androidx.fragment.app.DialogFragment;

import java.util.ArrayList;

public class DialogoSeleccionLista extends DialogFragment {
    @NonNull
    @Override
    public Dialog onCreateDialog(@Nullable final Bundle savedInstanceState) {

        final ArrayList<String> itemsSeleccionados = new ArrayList<String>();
        final String[] items = {"Español", "Ingles", "Frances", "Aleman"};
```

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

        //OPCION 1: Solo se puede elegir una opción
        builder.setTitle("Selección")
                .setItems(items, new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int item) {
                        Log.i("Dialogos", "Opción elegida: "+items[item]);
                        Toast.makeText(getActivity(),
                                "Has seleccionado "+ items[item] ,
                                 Toast.LENGTH_SHORT).show();
                    }
                });

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

En la OPCION 2, la forma de utilizarlos es muy similar a la ya comentada. En el caso de `setSingleChoiceItems()`, el método tan sólo se diferencia de `setItems()` en que recibe un segundo parámetro adicional que indica el índice de la opción marcada por defecto. Si no queremos tener ninguna de ellas marcadas inicialmente pasaremos el valor -1.

```
package com.example.ejdialogos;

import android.app.Dialog;
import android.content.DialogInterface;
import android.os.Bundle;
import android.util.Log;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AlertDialog;
import androidx.fragment.app.DialogFragment;

import java.util.ArrayList;

public class DialogoSeleccionRadioButton extends DialogFragment {
    @NonNull
    @Override
    public Dialog onCreateDialog(@Nullable final Bundle savedInstanceState) {
        final ArrayList<String> itemsSeleccionados = new ArrayList<String>();
        final String[] items = {"Español", "Ingles", "Francés", "Alemán"};

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

        //OPCION 2: Solo se puede elegir una opción, similar a radioButton
        //pero se puede sacar preselecciona una opción
        builder.setTitle("Selección")
              .setSingleChoiceItems(items, -1,
                      new DialogInterface.OnClickListener() {
                          @Override
```

```
                        public void onClick(DialogInterface dialog,
                                            int item) {
                            Log.i("Dialogos",
                                    "Opción Elegida: " + items[item]);
                            Toast.makeText(getActivity(),
                                        "Has seleccionado "+ items[item],
                                        Toast.LENGTH_SHORT).show();
                        }
                    })
            .setPositiveButton("Aceptar",
                    new DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialog,
                                            int item) {
                           // Log.i("Dialogos", "Opcion: " + items[item]);
                           Toast.makeText(getActivity(),
                                        "Opcion Aceptar",
                                        Toast.LENGTH_SHORT).show();
                            //dialog.cancel();
                        }
                    });

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

Si por el contrario optamos por la OPCION 3, opción de selección múltiple, la diferencia principal estará en que tendremos que implementar un listener del tipo `DialogInterface.OnMultiChoiceClickListener`. En este caso, en el evento `onClick` recibiremos tanto la opción seleccionada (`item`) como el estado en el que ha quedado (`isChecked`). Además, en esta ocasión, el segundo parámetro adicional que indica el estado por defecto de las opciones ya no será un simple número entero, sino que tendrá que ser un array de booleanos. En caso de no querer ninguna opción seleccionada por defecto pasaremos el valor `null`.

```
package com.example.ejdialogos;

import android.app.Dialog;
import android.content.DialogInterface;
import android.os.Bundle;
import android.util.Log;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AlertDialog;
import androidx.fragment.app.DialogFragment;

import java.util.ArrayList;

public class DialogoSeleccionCheckbox extends DialogFragment {
    @NonNull
    @Override
    public Dialog onCreateDialog(@Nullable final Bundle savedInstanceState) {
        final ArrayList itemsSeleccionados = new ArrayList();
```

```
        final String[] items = {"Español", "Ingles", "Francés", "Alemán"};
        String seleccion;

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

        //OPCION 3: Se pueden elegir más de una opción, similar a checkbox
        //y se pueden preseleccionar una o varias opciones
        builder.setTitle("Selección")
                .setMultiChoiceItems(items, null,
                      new DialogInterface.OnMultiChoiceClickListener() {
                          @Override
                          public void onClick(DialogInterface dialog,
                                              int item, boolean isChecked) {
                              if (isChecked){
                                  itemsSeleccionados.add(item);
                                  Toast.makeText(getActivity(),
                                     "Checks Seleccionados: (
                                          "+itemsSeleccionados.size()+")",
                                        Toast.LENGTH_SHORT).show();
                              }
                              else if(itemsSeleccionados.contains(item)){
                                  itemsSeleccionados.remove(
                                              Integer.valueOf(item));
                                  Toast.makeText(getActivity(),
                                        "Checks Seleccionados: ( " +
                                              itemsSeleccionados.size()+")",
                                         Toast.LENGTH_SHORT).show();
                              }
                          }
                      })
                .setPositiveButton("Aceptar",
                         new DialogInterface.OnClickListener() {
                             @Override
                             public void onClick(DialogInterface dialog,
                                                 int item) {
                                 Toast.makeText(getActivity(),
                                     "Checks Seleccionados: ( "+
                                          itemsSeleccionados.size()+")",
                                       Toast.LENGTH_SHORT).show();
                             }
                         });

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

El dialogo en cada uno de los casos quedará de la siguiente manera:

Dialogo Personalizados

Por último, vamos a ver cómo podemos establecer completamente el aspecto de un cuadro de diálogo. Para esto vamos a actuar como si estuviéramos definiendo la interfaz de una actividad, es decir, definiremos un layout XML con los elementos a mostrar en el diálogo. Por ejemplo, vamos a definir un layout de ejemplo llamado `dialogo_personal.xml` que colocaremos como siempre en la carpeta *res/layout*. Contendrá por ejemplo una imagen a la izquierda y dos líneas de texto a la derecha:

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

    <ImageView
        android:id="@+id/imageView1"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:src="@drawable/ic_face"
        android:layout_marginTop="20dp"
        android:layout_marginLeft="20dp"
        android:layout_marginRight="20dp"/>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:padding="3dp"
        android:layout_marginTop="20dp"
        android:layout_marginLeft="20dp">

        <TextView
            android:id="@+id/textView1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Dialogo_linea_1"
            android:textSize="18dp"/>
```

```
        <TextView
            android:id="@+id/textView2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Dialogo_linea_2" />

    </LinearLayout>
</LinearLayout>
```

En el caso de los diálogos personalizados, en el método `onCreateDialog()` correspondiente utilizaremos el método `setView()` del builder para asociarle nuestro layout personalizado, que previamente tendremos que inflar como ya hemos comentado utilizando el método `inflate()`. Finalmente podremos incluir botones como hemos vimos para los diálogos de alerta o confirmación. En este caso de ejemplo incluiremos un botón de *Aceptar*.

```
package com.example.ejdialogos;

import android.app.Dialog;
import android.content.DialogInterface;
import android.os.Bundle;
import android.view.LayoutInflater;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AlertDialog;
import androidx.fragment.app.DialogFragment;

public class DialogoPersonalizado extends DialogFragment {

    @NonNull
    @Override
    public Dialog onCreateDialog(@Nullable Bundle savedInstanceState) {
        //return super.onCreateDialog(savedInstanceState);
        AlertDialog.Builder builder =
                new AlertDialog.Builder(getActivity());
        LayoutInflater inflater = getActivity().getLayoutInflater();

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

}
```

De esta forma, si ejecutamos de nuevo nuestra aplicación de ejemplo y lanzamos el diálogo personalizado veremos algo como lo siguiente:

Enviar eventos desde un DialogFragment hacia su Actividad Contenedora

Si queremos pasar los eventos que ocurren en el diálogo hacia una actividad debemos recurrir a una **interfaz de comunicación** que permita compartir los métodos de acción.

Para ello, debemos seguir los siguientes pasos:

Paso 1: Declarar una interfaz dentro del `DialogFragment`. La declaración de la interfaz debe tener definido un método para cada acción que reciba el dialogo.

Si queremos procesar en la actividad los métodos del botón positivo y el negativo, entonces tenemos que creas una interfaz con los dos respectivos métodos:

```
public interface OnDialogoConfirmacionListener{
    void onPossitiveButtonClick(); //Eventos Botón Positivos
    void onNegativeButtonClick();  //Eventos Botón Negativo
}
```

Paso 2: Declarar un atributo del tipo de la interfaz para conseguir la instancia directa de la actividad

```
// Interfaz de comunicación
OnDialogoConfirmacionListener listener;
```

Paso 3: Comprobar que la actividad ha implementado la interfaz podemos usar el método `onAttach()`. Recuerda que este recibe la instancia de la actividad contenedora del fragmento. Simplemente asignas la actividad a la instancia de la interfaz.

Si este proceso no es posible, entonces lanza una excepción del tipo `ClassCastException`.

```
@Override
public void onAttach(Context context) {
    super.onAttach(context);

    try{
        listener = (OnDialogoConfirmacionListener) context;
    } catch (ClassCastException e) {
        throw new ClassCastException(context.toString() +
                " no Implemento OnDialogoConfirmacionListener");
    }

}
```

Paso 4: Invocar los métodos de la interfaz en las partes del dialogo donde deseamos implicar a la actividad.

```
public Dialog onCreateDialog(@Nullable Bundle savedInstanceState) {
    AlertDialog.Builder builder =
            new AlertDialog.Builder(getActivity());

    builder.setMessage("¿Confirma la accion selecionada?")
            .setTitle("Confirmacion")
            .setPositiveButton("ACEPTAR",
                    new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    listener.onPossitiveButtonClick();
                }
            })
            .setNegativeButton("CANCELAR",
                    new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    listener.onNegativeButtonClick();
                }
            });
    return builder.create();
}
```

Paso 5: Implementar sobre la actividad contenedora, la interfaz declarada en el fragmento. Y sobreescribir dentro de la actividad los métodos de la interfaz, con las acciones que se requieran

```
public class MainActivity extends AppCompatActivity
                implements DialogoConfirmacion.OnDialogoConfirmacionListener {

    . . .

    @Override
    public void onPossitiveButtonClick() {
        Toast.makeText(this, "Boton Positivo pulsado",
                       Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onNegativeButtonClick() {
        Toast.makeText(this, "Botón Negativo pulsado",
                       Toast.LENGTH_SHORT).show();
    }
}
```

Actividad Propuesta

Crear una aplicación libre, utilizando todo lo visto hasta ahora (Controles básicos, controles de selección, pestañas, Toolbar, menus, temas, estilos,…), en la que antes de empezar se pida autentificación al usuario (usuario: usuario1 y password:123456) y a la hora de finalizar también pida confirmación para terminar la aplicación.