DIÁLOGOS CON FRAGMENTOS

En la documentación de Android tenemos gran variedad de ejemplos donde basarnos según las necesidades de nuestra aplicación

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

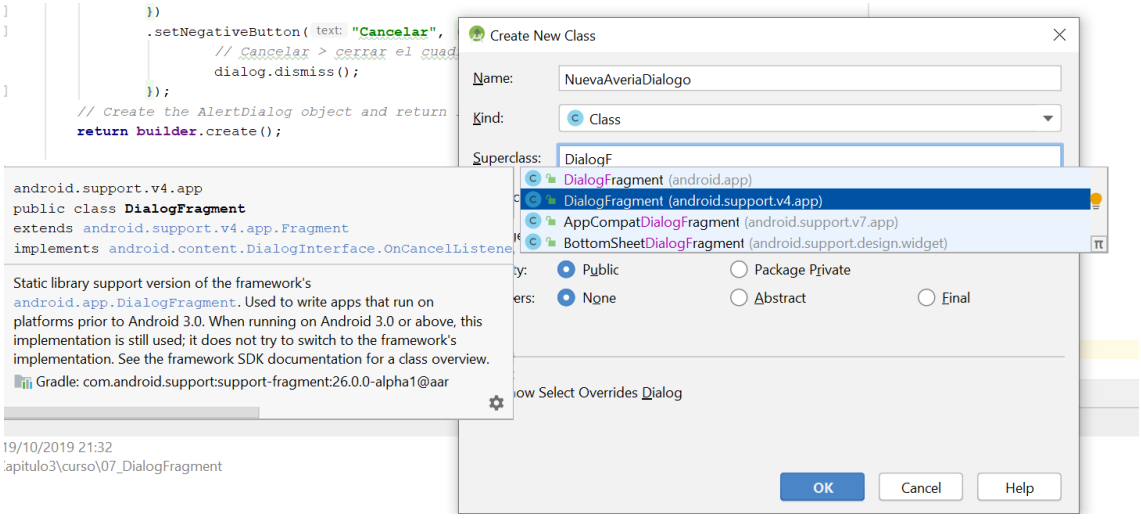
Vamos por tanto a definirlos como fragmentos ya que permiten tener una clase java que extiende de DialogFragment y en el enlace anterior tenemos la implementación de todos los tipos que podemos implementar.

Todos los cuadros de dialogo podemos ver en la documentación que extienden de DialogFragment. En su método onCreateDialog se lanza cuando instanciamos el diálogo, y luego podemos definir distintas configuración dentro de los cuadros de diálogo: añadir botonos, cuadros de texto, checkbox, vistas personalizadas, ….

Paso 1:

Creamos un proyecto con una Basic Activity, ya que lo que queremos es que cuando se pulse el botón flotante se lance el cuadro de diálogo.

Vamos a crear una clase: sobre carpeta java, dentro del paquete donde se almacenan los archivos de mi proyecto hacemos click con segundo botón del ratón y seleccionamos: new→java class



En Superclass le diremos que herede de la clase DialogFragment, en concreto de la que vemos marcada para que permita compatibilidad con aplicaciones hacia atrás.

Paso 2:

A partir de ahora vamos a empezar a implementar el dialogo. Para ello tener muy presente la documentación anterior ya que nos va a ir diciendo todos los métodos y pasos que necesitamos.

1. Primero vemos como crear un fragmento de un dialogo y nos dice que primero debemos implementar el onCreateDialog y copiamos y pegamos todo el código que viene en la documentación en el apartado “Crear un fragmento de dialogo” y a continuación lo vamos adaptando.
2. AlertDialog aparece en rojo por lo que tenemos que importar la clase. Cuando aparecen dos clases distintas solemos importar aquella que viene de la librería de soporte para que sea compatible con versiones antiguas de Android.

3. En lugar de setMenssage nosotros pondremos setTitle(“Nueva Averia”)
4. Los botones son el negativo, que sería la opción de cancelar que sería el que cierra el diálogo, y el positivo o guardar
    a. Cancelar (negativo): como primer parámetro ponemos el texto que queremos que aparezca en el botón y si pulsa el botón tenemos que cerrar el dialogo y para ello se hace que implemente la siguiente línea de código:
    ```
    dialog.dismiss();
    ```
    b. Guardar (positivo): primer parámetro el nombre y en la implementación haceos el siguiente código:
    ```
    Toast.makeText(getActivity(), "Avería
        guardada",Toast.LENGTH_SHORT).show();
    dialog.dismiss();
    ```
5. Importamos la clase DialogInterface.

Paso 3: lanzar el dialogo

Opción 1: crear en Nueva AveriaDialog un método estático que devolviera el diálogo.

Opción 2: programar el botón flotante para que lance el diálogo

Nos quedamos con la opción dos y haremos las siguientes modificaciones:

1. Creamos en MainActivity una instancia de la clase de DialogFragment
    ```
    DialogFragment dialogoNuevaAveria;
    ```
2. Dentro de la implementación del botón flotante implementamos la instancia al dialogo de la siguiente manera:
    ```
    FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
    fab.setOnClickListener((view) → {
            dialogoNuevaAveria = new NuevaAveriaDialogo();
            dialogoNuevaAveria.show(getSupportFragmentManager(), tag: "NuevaAveriaDialogo");
    });
    ```
    Pasándole, como vemos al método show, el gestor de fragmentos y una etiqueta, que indica simplemente como queremos llamar a esa transacción que vamos a realizar
3. Si ejecutamos vemos que nuestro cuadro se muestra con los dos botones pero aún no tiene nada. Vamos a personalizar.

Paso 4: Personalizar el cuadro de diálogo.

Para esto podemos en la documentación ver cual se adapta más a nuestras necesidades.

Lo que vamos a hacer es añadirle una vista con las opciones que queremos que aparezcan. En nuestro caso sería similar a este tipo

Para ello vamos a ver la implementación en java y vemos que debemos añadir una vista personalizada.

Esto último requiere dos líneas de código: una para obtener el layout del activity sobre el que estamos trabajando y segundo, añadirle una vista personalizada sobre el cuadro de diálogo.

Para hacer la vista personalizada creamos un nuevo layout que le añadimos dos TextView (título de la avería y otro para la descripción). Una vez creada en la segunda línea de texto añadida anteriormente decimos que cargue este layout

```
@Override
public Dialog onCreateDialog(Bundle savedInstanceState) {

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

    // Get the layout inflater
    LayoutInflater inflater = getActivity().getLayoutInflater();

    builder.setView(inflater.inflate(R.layout.dialogo_nueva_averia,  root: null));


    builder.setTitle("Nueva avería")
```

Paso 5: cómo pasar la información que hemos agregado en el diálogo a nuestra actividad principal.

Los primero será crear una interface, como ya vimos en ejemplos anteriores en la comunicación entre fragments. A esta interface le creo los siguientes métodos:

```
public interface OnNuevaAveriaListener {
    public void onAveriaGuardarClickListener();
}
```

A continuación en el MainActivity.java, implementamos dicha interface que me obliga a implementar dicho método:

```
@Override
public void onAveriaGuardarClickListener() {
    Toast.makeText( context: this,  text: "Se ha recibido la avería guardada", Toast.LENGTH_SHORT).show();
}
```

Ahora, tal como se hizo en el ejemplo antes mencionado, en NuevaAveriaDialog, nos creamos un objeto de dicha interface

```
public class NuevaAveriaDialogo extends DialogFragment {
    AlertDialog.Builder builder;
    OnNuevaAveriaListener mListener;
```

y también definimos el método onAttach que es el que nos permite verificar que el activity que ha cargado esta avería implementa la interface onNuevaAveriaListener y en caso de que no sea así capturará un excepción. Este código podemos obtenerlo también de la documentación.

Por último para que pueda lanzar ese método, tendremos que decir que se ha de producir cuando se pulse en concreto el botón guardar, que es cuando queremos que la información del diálogo pase al activity, para ello hemos de añadir la siguiente línea de código:

```
builder.setTitle("Nueva avería")
        .setPositiveButton( text: "Guardar", (dialog, id) → {
                Toast.makeText(getActivity(),  text: "Avería guardada", Toast.LENGTH_SHORT).show();
                dialog.dismiss();
                mListener.onAveriaGuardarClickListener();
        })
```

Para probar a pasar información, modifica el método de la interface pasándole dos String y en onCreateDialog haz las siguientes modificaciones:

```
public Dialog onCreateDialog(Bundle savedInstanceState) {

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

    // Get the layout inflater
    LayoutInflater inflater = getActivity().getLayoutInflater();

    //builder.setView(inflater.inflate(R.layout.dialogo_nueva_averia, null));
    View view = inflater.inflate(R.layout.dialogo_nueva_averia,  root: null);
    builder.setView(view);

    final EditText ETtexto = (EditText)view.findViewById(R.id.editText);
    final EditText ETdescripcion = (EditText)view.findViewById(R.id.editText2);

    builder.setTitle("Nueva avería")
            .setPositiveButton( text: "Guardar", (dialog, id) → {
                    Toast.makeText(getActivity(),  text: "Avería guardada", Toast.LENGTH_SHORT).show();
                    dialog.dismiss();
                    mListener.onAveriaGuardarClickListener(ETtexto.getText().toString(), ETdescripcion.getText().toString() );
            })
```