Diálogos y menús

Desarrollo para plataformas móviles

Borja Martin Herrera
Borja.herrera@u-tad.com

CENTRO UNIVERSITARIO
DE TECNOLOGÍA Y ARTE DIGITAL

Cuadros de diálogo

- Los cuadros de diálogo son elementos que permiten la visualización de información de forma modal, sin necesidad de emplear una activity adicional. Existen dos grandes tipos de cuadros de diálogo:
  - Preconstruidos: con una sección para el título, mensaje y botones
  - Personalizados:: con una vista totalmente personalizada

Cuadros de diálogo. Por defecto

- Necesitamos una clase que extienda de DialogFragment. Esta clase tiene su propio ciclo de vida. El método onCreateDialog es donde creamos y configuramos el diálogo

```
class DialogoDefecto : DialogFragment() {

    override fun onCreateDialog(savedInstanceState: Bundle?): Dialog {

        val builder = AlertDialog.Builder(requireContext())

        return builder.create()
    }
}
```

- El objeto de tipo AlertDialog.Builder ayuda a construir el diálogo, seteando cada una de las partes del mismo.

Cuadros de diálogo. Por defecto

- Las partes que se pueden personalizar son: titulo, mensaje y botones

```
override fun onCreateDialog(savedInstanceState: Bundle?): Dialog {

    val builder = AlertDialog.Builder(requireContext())
    // titulo
    builder.setTitle("Diálogo ayuda")
    // mensaje ó items
    builder.setMessage("Este es un diálogo de ayuda de la aplicación de dispositivos móviles")

    // botones
    builder.setPositiveButton( text: "YES", DialogInterface.OnClickListener { _, _ ->
        Log.v( tag: "dialogo",  msg: "Pulsado positivo")
    })
    builder.setNegativeButton( text: "NO", DialogInterface.OnClickListener { _, _ ->
        Log.v( tag: "dialogo",  msg: "Pulsado negativo")
    })
    builder.setNeutralButton( text: "CANCEL", DialogInterface.OnClickListener { _, _ ->
        Log.v( tag: "dialogo",  msg: "Pulsado neutral")
    })

    return builder.create()
}
```

- En los métodos que controlan la pulsación de los botones (setPositiveButton por ejemplo), además e indicar el texto que tendrá el botón es necesario indicar la acción que tendrá dicho boton

Cuadros de diálogo. Por defecto

- Una vez creado el cuadro de diálogo, tendrá que ser lanzado desde la activity o el sitio desde el cual se lanzará. Para ello se utiliza el método show()

```
DialogoDefecto().show(supportFragmentManager,"")
```

- Una vez echo ejecutado esto cuando se pulsa un botón (por ejemplo), el cuadro de diálogo será mostrado

Cuadros de diálogo. Personalizado

- El proceso es muy similar al anterior, con la única diferencia que en vez de poner titulo, mensaje y/o botones, se pone un xml personalizado. Para ello los pasos son los siguientes

1. Creamos un xml personalizado con el aspecto que se quiere para el cuadro de diálogo
2. En la clase que representa el cuadro de diálogo, se crea una variable de tipo view y se infla mediante un objeto de tipo LayoutInflater. Además se pone en el cuadro de diálogo con el método setView()

```
class DialogoPersolanizado : DialogFragment() {

    private lateinit var vista: View

    override fun onCreateDialog(savedInstanceState: Bundle?): Dialog {
        val builder =  AlertDialog.Builder(requireContext())
        vista = LayoutInflater.from(requireContext()).inflate(R.layout.dialogo_proyecto, root: null)
        builder.setView(vista)
        return builder.create()
    }


}
```

Cuadros de diálogo. Personalizado

El último paso es instanciar cada uno de los elementos de la interfaz y utilizarlos. Para ello es necesario recurrir al método findViewById sobre el dato correspondiente

```
class DialogoPersolanizado : DialogFragment() {

    private lateinit var vista: View
    private lateinit var boton: Button
    private lateinit var editNombre: EditText
    private lateinit var editResponsable: EditText
    private lateinit var editPresupuesto: EditText

    override fun onCreateDialog(savedInstanceState: Bundle?): Dialog {
        val builder =  AlertDialog.Builder(requireContext())
        vista = LayoutInflater.from(requireContext()).inflate(R.layout.dialogo_proyecto, root: null)
        builder.setView(vista)
        return builder.create()
    }

    override fun onStart() {
        super.onStart()
        boton = vista.findViewById(R.id.boton_add)
        editNombre = vista.findViewById(R.id.edit_nombre)
        editResponsable = vista.findViewById(R.id.edit_responsable)
        editPresupuesto = vista.findViewById(R.id.edit_presupuesto)

        boton.setOnClickListener { it: View!
            // sacar datos y llevarlos al recycler
            val responsble = editResponsable.text.toString()
            val nombre = editNombre.text.toString()
            val presupuesto = editPresupuesto.text.toString().toInt()
            dismiss()
        }
    }
}
```

Cuadros de diálogo. Comunicación

En el caso de querer mandar datos de un cuadro de diálogo a una activity se utiliza las interfaces de callback. Para ello es necesario realizar los siguientes pasos:

- 1. Crear una interfaz en el origen de los datos, en este caso el cuadro de diálogo, con los métodos que se quieran utilizar

```
interface OnDialogoProyectoListener{
    fun onProyectoAdd(nombre: String, responsable: String, presupuesto: Int)
}
```

- 2. Crear un objeto del tipo interfaz, instanciándolo en el método onAttach y utilizarlo cuando se considere necesario

```
private lateinit var listener: OnDialogoProyectoListener

override fun onAttach(context: Context) {
    super.onAttach(context)
    listener = context as OnDialogoProyectoListener
}
```

```
override fun onStart() {
    super.onStart()
    boton = vista.findViewById(R.id.boton_add)
    editNombre = vista.findViewById(R.id.edit_nombre)
    editResponsable = vista.findViewById(R.id.edit_responsable)
    editPresupuesto = vista.findViewById(R.id.edit_presupuesto)

    boton.setOnClickListener { it: View!
        // sacar datos y llevarlos al recycler
        val responsble = editResponsable.text.toString()
        val nombre = editNombre.text.toString()
        val presupuesto = editPresupuesto.text.toString().toInt()
        listener.onProyectoAdd(nombre,responsble,presupuesto)
        dismiss()
    }
}
```

Cuadros de diálogo. Comunicación

- 3. En el destino de la comunicación, en este caso la activity, implementar la interfaz creada, lo que obligará a traer los métodos y por lo tanto la comunicación realizada

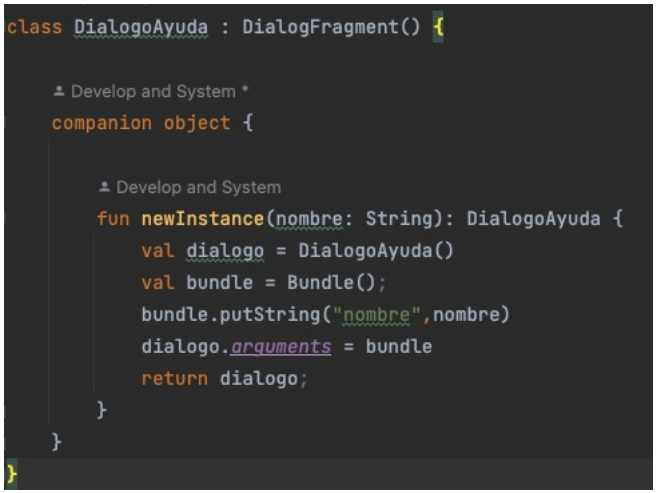
```
class MainActivity : AppCompatActivity(), OnClickListener, DialogoProyecto.OnDialogoProyectoListener {

    new *
    override fun onProyectoAdd(nombre: String, responsable: String, presupuesto: Int) {
        // llevar los datos a la lista (recycler)
        adaptadorProyecto.addProyecto(nombre,responsable, presupuesto)

    }
```

Cuadros de diálogo. Comunicación

En el caso de querer hacer una comunicación en sentido contrario,, es decir mandar datos de la activity al cuadro de diálogo, es necesario utilizar un constructor estático.



```
class DialogoAyuda : DialogFragment() {

    Develop and System *
    companion object {

        Develop and System
        fun newInstance(nombre: String): DialogoAyuda {
            val dialogo = DialogoAyuda()
            val bundle = Bundle();
            bundle.putString("nombre",nombre)
            dialogo.arguments = bundle
            return dialogo;
        }
    }
}
```

Este método permite crear un cuadro de diálogo con datos pasados. Los datos se ponen como argumentos.

Cuadros de diálogo. Comunicación

Lo siguiente es recuperarlos cuando empiece el ciclo de vida del cuadro. Para ello se realiza la extracción en el método onAttach, igualando el dato a una variable de clase para que esté disponible en toda la clase

```
private lateinit var nombre: String ;

new *
override fun onAttach(context: Context) {
    super.onAttach(context)
    this.nombre = arguments?.getString( key: "nombre") ?: "defecto"
}
```

Una vez recuperado los datos, se puede utilizar con la comunicación ya realizada. Por último, para poder lanzar el cuadro de diálogo es necesario llamar el método newInstance creado

```
DialogoAyuda.newInstance( nombre: "Ejemplo",10).show(supportFragmentManager, tag: null)
```

Menús

Los menús representan un elemento de interacción con los usuarios. Para poder utilizarlos, por defecto se ponen en la parte superior (llamada actionbar), pero antes es necesario configurarlos. Para ello seguiremos estos pasos:

- 1. Creamos un xml que representará el contenido del menú. Este elemento tiene <item> los cuales tienen configuradas las características de cada elemento

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <item android:title="Opcion 1"
        android:id="@+id/opcion1"
        android:icon="@android:drawable/ic_dialog_email"
        />
    <item android:title="Opcion 2"
        android:id="@+id/opcion2"
        android:icon="@android:drawable/ic_dialog_info"
        app:showAsAction="ifRoom"
        />
</menu>
```

Menús

- 2. Una vez está creado el recurso, el siguiente paso es ponerlo dentro de la activity. Para ello se sobreescriben los métodos onCreateOptionMenu y onOptionItemSelected

```
override fun onCreateOptionsMenu(menu: Menu?): Boolean {
    menuInflater.inflate(R.menu.main_menu,menu)
    return true
}

 Develop and System *
override fun onOptionsItemSelected(item: MenuItem): Boolean {

    when(item.itemId){
        R.id.opcion1->{
            DialogoProyecto().show(supportFragmentManager, tag: "")
        }
        R.id.opcion2->{
            DialogoAyuda.newInstance( nombre: "Menus").show(supportFragmentManager, tag: "")
        }
    }
    return true
}
```