

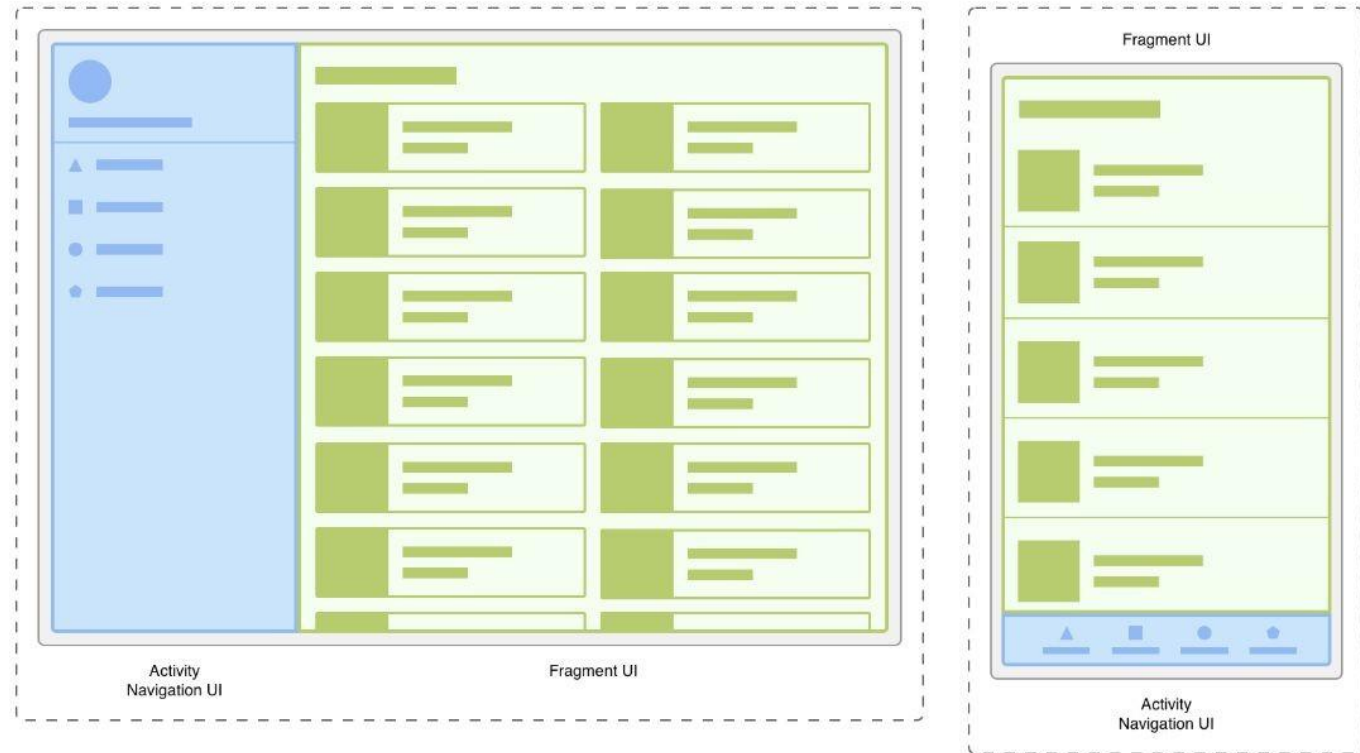
# Desarrollo de Aplicaciones para Dispositivos Móviles

Ingeniero Walter Medina  
[walter.medina@correounivalle.edu.co](mailto:walter.medina@correounivalle.edu.co)



# Fragment

Un Fragment representa una parte reutilizable de la IU de tu app. Un fragmento define y administra su propio diseño, tiene su propio ciclo de vida y puede administrar sus propios eventos de entrada. Los fragmentos no pueden existir por sí solos. Deben estar alojados por una actividad u otro fragmento.



- ★ Se comparte enlace del repositorio con los ejemplos realizados en clase.



# Fragment Estáticos

Para agregar un fragmento de forma estática en el XML de diseño de tu actividad, usa el contenedor `FragmentManager`.

```
<!-- res/layout/example_activity.xml -->
<androidx.fragment.app.FragmentContainerView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/fragment_container_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:name="com.example.ExampleFragment" />
```

El atributo `android:name` especifica el nombre del Fragment que quieres que se muestre en la actividad

# Fragment Dinámicos

Para agregar un fragmento de manera programática o dinámica al diseño de tu actividad, debes incluir un `FragmentManager`, como se observa en este caso ya no se usa el atributo `android:name`, debido a que esa transición de fragment se hará desde el código kotlin.

```
<!-- res/layout/example_activity.xml -->
<androidx.fragment.app.FragmentContainerView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/fragment_container_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

# Ir de un Fragment a otro Fragment (Standar)

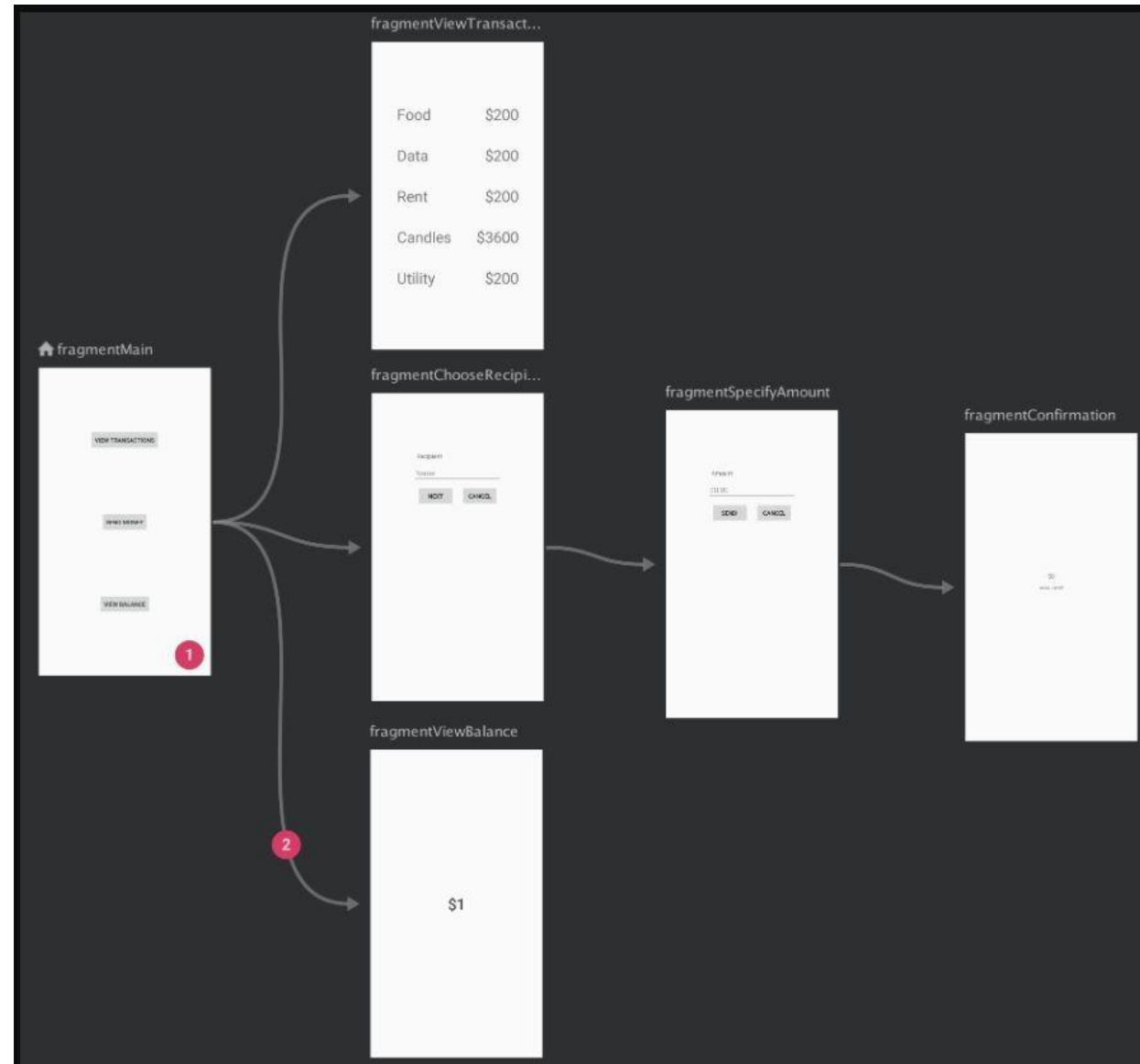
Para ir de un fragmento A a un Fragmento B de la manera tradicional se debe hacer uso de la propiedad **transaction** , como se observa en el siguiente código

```
private fun navigationStandar() {  
    binding.btnFragmentB.setOnClickListener { it: View!  
        val fragmentB = FragmentB()  
        val transaction = requireActivity().supportFragmentManager.beginTransaction()  
        transaction.replace(R.id.fragmentContainerStatic, fragmentB)  
        transaction.addToBackStack(name: null) // Opcional, para permitir retroceder  
        transaction.commit()  
    }  
}
```



# Navigation

Un Navigation o también llamado gráfico de navegación, es un archivo de recursos que contiene todos tus destinos y acciones. El gráfico representa todas las rutas de navegación de tu app. Principalmente esas rutas de navegación están compuestos por fragments



# Navegar del FragmentA al FragmentB con Navigation

Para ir de un fragmento A a un Fragmento B usando Navigation se hace uso del método **findNavController()**, donde se le pasa el id del fragment de destino, es id se genera en el gráfico de navegación.

```
private fun navigationFragmentB(){  
    binding.btnNavFragmentB.setOnClickListener { it: View!  
        findNavController().navigate(R.id.action_fragmentA_to_fragmentB)  
    }  
}
```

# Pasar información de un fragment a otro con Navigation

Para pasar información de un fragmento a otro usando Navigation, se hace uso de la clase Bundle, la cual requiere que se le pase la estructura clave, valor. Luego en el findNavController se le pasa el bundle con la data.

```
private fun pasarDataEntreFragments() {  
  
    binding.btnPasarDatos.setOnClickListener { it: View!   
  
        val dataEditText = binding.etName.text.toString()  
        val bundle = Bundle()  
        bundle.putString("clave", dataEditText)  
        findNavController().navigate(R.id.action_fragmentA_to_fragmentB, bundle)  
    }  
}
```

Finalmente para capturar la data desde el fragment de destino se hace como se observa en el código de la derecha, observe que se debe pasar la clave.

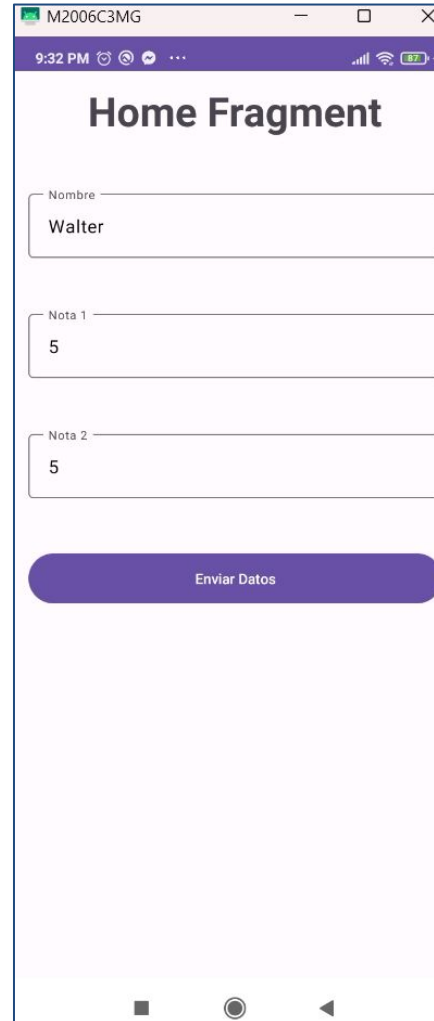
```
private fun capturarData(){  
  
    val textView = binding.tvData  
    val dataFragmentA = arguments?.getString(key: "clave")  
    textView.text =dataFragmentA  
}
```



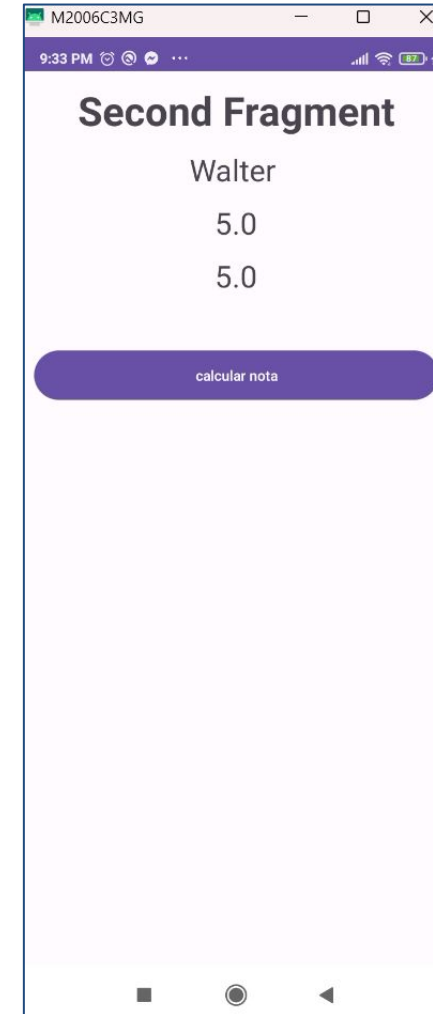
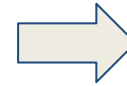
## Ejercicio en clase

Realice un formulario que permita ingresar el nombre del estudiante, la nota1, nota2 y que al dar clic en el botón “Enviar Datos”, la información se envíe y visualice en una segunda ventana. Al dar clic en el botón “calcular nota” debe salir un mensaje emergente informando si aprobó o no el curso. (nota < 3 = reprobó, nota >=3 = aprobó)

Recuerde que debe usar fragment y navigation para este ejercicio.



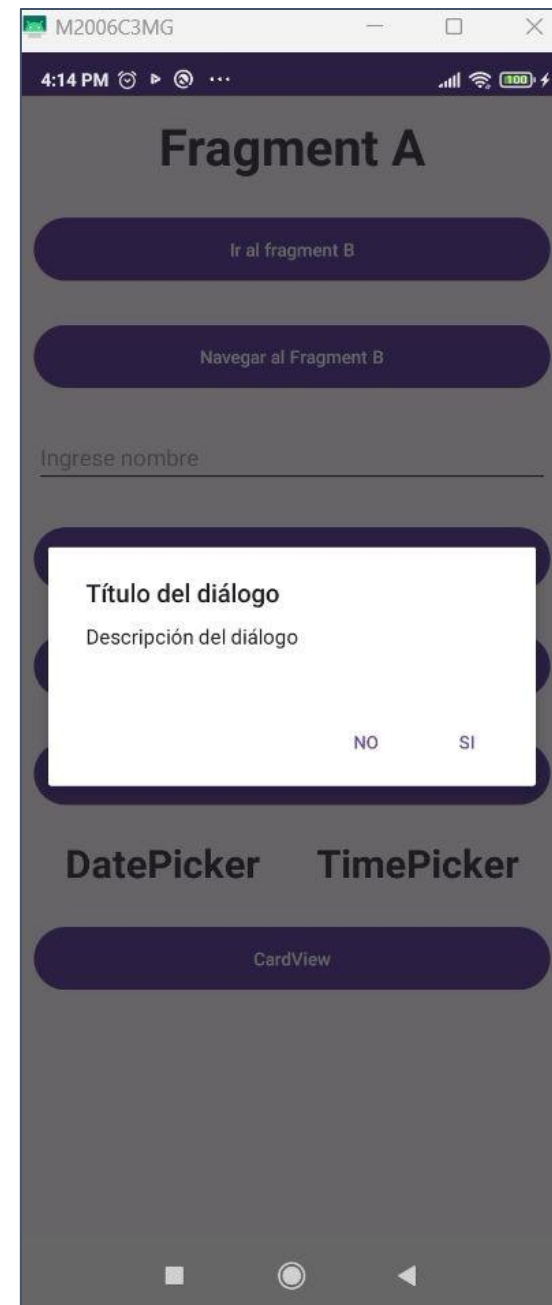
The screenshot shows a mobile application window titled "M2006C3MG" with a status bar at 9:32 PM. The main heading is "Home Fragment". Below it are three input fields: "Nombre" with the text "Walter", "Nota 1" with the value "5", and "Nota 2" with the value "5". At the bottom is a purple button labeled "Enviar Datos".



The screenshot shows a mobile application window titled "M2006C3MG" with a status bar at 9:33 PM. The main heading is "Second Fragment". Below it, the text "Walter" is displayed, followed by two lines showing the value "5.0". At the bottom is a purple button labeled "calcular nota".

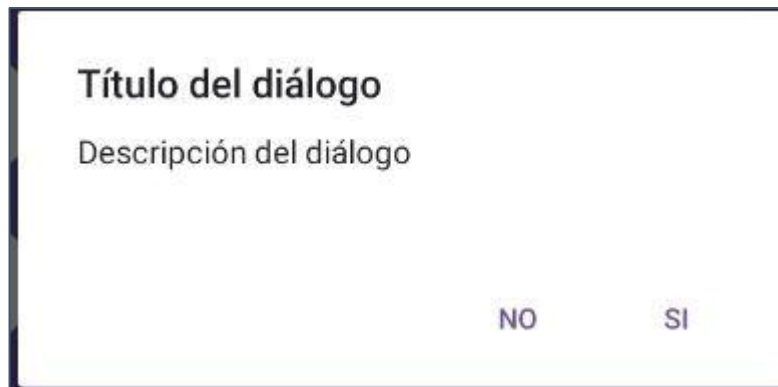
# Diálogos

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 que requieren que los usuarios realicen alguna acción para poder continuar.



# Diálogos Standar

Son diálogos que android trae por defecto para ser usados en las aplicaciones. Generalmente se componen de un Título, una área de contenido y unos botones de acción



```
fun showDialog(context: Context): AlertDialog {  
  
    val builder = AlertDialog.Builder(context)  
    builder.setCancelable(false)  
    builder.setTitle("Título del diálogo")  
        .setMessage("Descripción del diálogo")  
        .setPositiveButton(text: "SI") { dialog, _ ->  
            Toast.makeText(context, text: "SI", Toast.LENGTH_SHORT).show()  
            dialog.dismiss()  
        }  
        .setNegativeButton(text: "NO") { dialog, _ ->  
            Toast.makeText(context, text: "NO", Toast.LENGTH_SHORT).show()  
            dialog.dismiss()  
        }  
    return builder.create()  
}
```

# Diálogos Personalizados

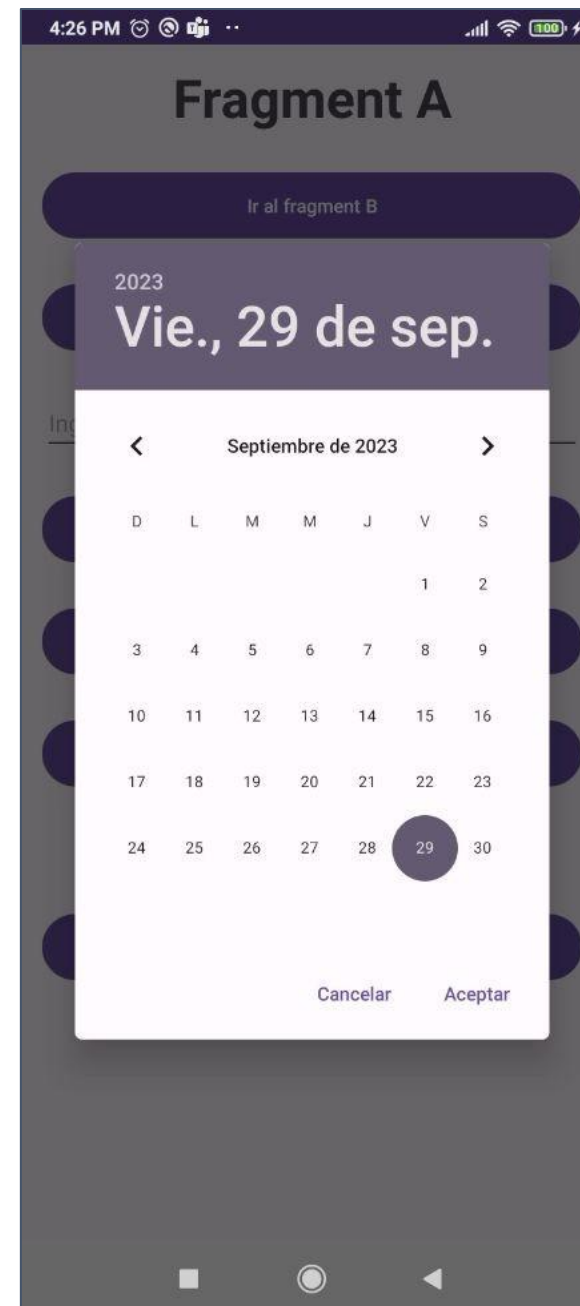
En Android puedes crear Diálogos con diseños personalizados, donde puedes agregar cualquier tipo de componente que requiera el diálogo.



# Diálogo DatePicker

Los DatePicker son diálogos que permiten al usuario seleccionar una fecha en particular.

En clases pasadas se había mostrado los DatePicker desde el lado de la Interface, en esta ocasión se mostrará cómo se implementan desde la lógica.

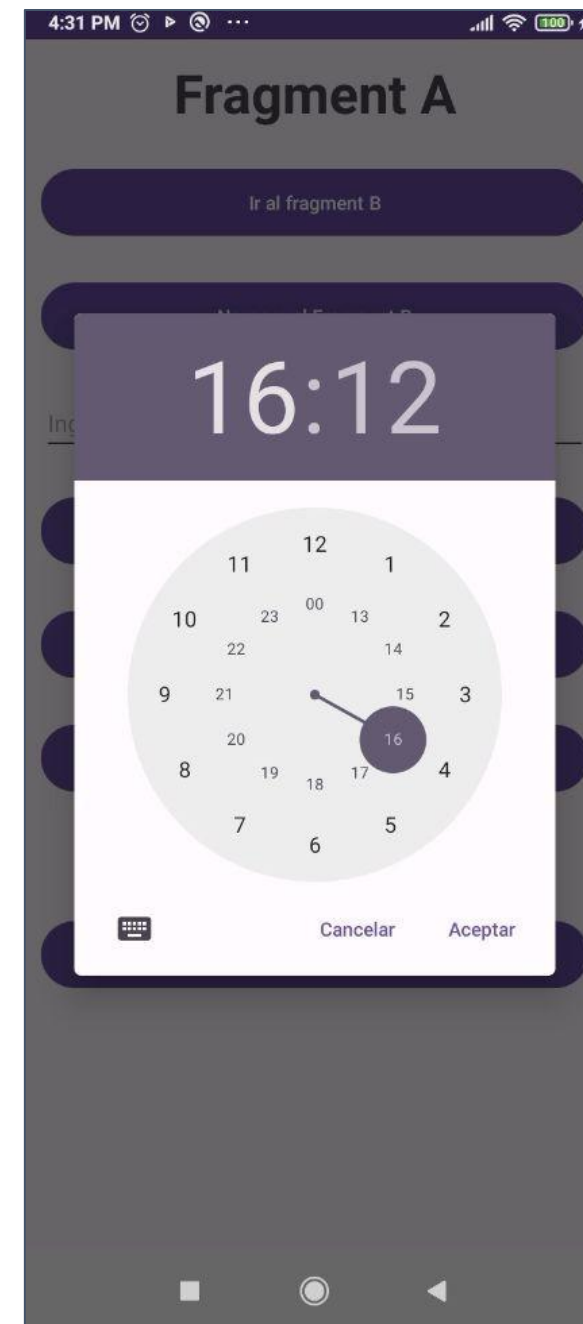




# Diálogo TimePicker

Los TimePicker son diálogos que permiten al usuario seleccionar una hora en particular.

En clases pasadas se había mostrado los TimePicker desde el lado de la Interface, en esta ocasión se mostrará cómo se implementan desde la lógica.



# CardView

Con frecuencia, las apps deben mostrar datos en contenedores con un estilo similar. Estos contenedores se usan a menudo en listas para retener la información de cada elemento.

Android tiene las CardView como una manera sencilla de mostrar información en tarjetas. Estas tarjetas tienen una elevación predeterminada por encima del grupo de vistas que las contiene, de modo que el sistema dibuja sombras debajo de ellas. Las tarjetas ofrecen una forma simple de incluir un grupo de vistas.



# GRACIAS !!

*“El esfuerzo vence al talento cuando el  
talento no se está esforzando”.  
Anónimo*



### Bibliografía:

- ❖ <https://developer.android.com/guide/topics/ui/dialogs?hl=es-419>
- ❖ <https://developer.android.com/guide/navigation/navigation-pass-data?hl=es-419>
- ❖ <https://developer.android.com/guide/topics/ui/layout/cardview?hl=es-419>
- ❖ <https://developer.android.com/guide/fragments?hl=es-419>

