CFGS DESARROLLO DE APLICACIONES MULTIPLATAFORMA

UD 3. Activitys y Fragments

Módulo: Programación multimedia y dispositivos móviles

Víctor J. Vergel Rodríguez

Centro de Enseñanza
Gregorio Fernández

Activitys

- Los intents permiten enviar información.

```
var miIntent: Intent= Intent(this, Activity2::class.java)
miIntent.putExtra("dato", binding.tietDato.text.toString())
startActivity(miIntent)
```

- Devolver información desde el segundo Activity

```
var launcher: ActivityResultLauncher<Intent> = registerForActivityResult(
            ActivityResultContracts.StartActivityForResult()
        ){
            if(it.resultCode== Activity.RESULT_OK){
                binding.tvDatoRecibido.text= it.data?.getStringExtra("dato").toString()
            }
        }
```

Intents del sistema

- Envío de emails

```
val intent = Intent(Intent.ACTION_SEND)
intent.type = "text/plain"
intent.putExtra(Intent.EXTRA_SUBJECT, "Asunto del correo")
intent.putExtra(Intent.EXTRA_TEXT, "Cuerpo del correo")
intent.putExtra(Intent.EXTRA_EMAIL, arrayOf("victorvergel@verinsis.com"))
```

- Realizar una llamada

```
val intent = Intent(Intent.ACTION_DIAL, Uri.parse("tel:123456789"))
```

Intents del sistema

- Lanzar un navegador web

```
val intent = Intent(Intent.ACTION_VIEW, Uri.parse("http://www.google.com"))
```

- Lanzar Google maps

```
val intent = Intent(Intent.ACTION_VIEW,  Uri.parse("geo:41.6525,-4.7245"))
```

- Lanzar WhatsApp

```
val intent = Intent(Intent.ACTION_VIEW, Uri.parse("https://wa.me/34666666666"))
```

Diálogos

- Confirmación

```
AlertDialog.Builder(this)
                .setTitle("Título del diálogo")
                .setMessage("Mensaje del diálogo")
                .setIcon(android.R.drawable.ic_dialog_alert)
                .setPositiveButton("Aceptar") { dialog, which ->
                    Snackbar.make(binding.root, "Has pulsado Aceptar", Snackbar.LENGTH_SHORT).show()
                }
                .setNegativeButton("Cancelar", DialogInterface.OnClickListener() { dialog,
               which ->
                    Snackbar.make(binding.root, "Has pulsado Cancelar", Snackbar.LENGTH_SHORT).show()
                })
                .show()
```

Diálogos

- Selección múltiple

```
            val colores = arrayOf("Rojo", "Verde", "Azul", "Amarillo")
              val seleccionados = booleanArrayOf(true, false, false, false)
              AlertDialog.Builder(this)
                  .setTitle("Selecciona colores")
                  .setMultiChoiceItems(colores, seleccionados) { dialog, which, isChecked ->   seleccionados[which] =
isChecked}
                  .setPositiveButton("Aceptar") { dialog, which ->
                      var coloresSeleccionados = ""
                      for (i in colores.indices) {
                          if (seleccionados[i]) {       coloresSeleccionados += colores[i] + ", "
}
                      }
                      Snackbar.make(binding.root, "Has seleccionado: $coloresSeleccionados",
Snackbar.LENGTH_SHORT).show()
                  }
                  .setNegativeButton("Cancelar", null)
                  .show())
```

Diálogos

- Selección simple

En lugar del setMultipleChoise meteríamos:

```
.setSingleChoiceItems(colores, 0) { dialog, which ->
                    Snackbar.make(binding.root, "Has seleccionado: ${colores[which]}", Snackbar.LENGTH_SHORT).show()
                }
```

Menús

- Contextuales

En lugar del setMultipleChoise meteríamos:

```
.setSingleChoiceItems(colores, 0) { dialog, which ->
                    Snackbar.make(binding.root, "Has seleccionado: ${colores[which]}", Snackbar.LENGTH_SHORT).show()
                }
```

- Aplicación

Menús

• ActionBar

Ubicado en la parte superior de aplicaciones con la finalidad de facilitar la navegación entre activitys. **Menús personalizados**:

```
<item
….
android:icon="@drawable/icono"
app:showAsAction="ifRoom" />
```

ifRoom: acción en el ActionBar con icono el elemento si tiene espacio la ActionBar

withText: Incluir el texto en la ActionBar.

never: no mostrar en la ActionBar el elemento, por lo tanto permanecerá en el componente Overflow.

always: Mostrar en la ActionBar el elemento siempre, si no hubiera espacio recortará el resto de componentes (title).

La propiedad orderInCategory nos indica la ordenación, cuanto más elevada más a la derecha.

Menús

- Toolbar

Android y concretamente Android Lollipop trae consigo un nuevo concepto de ActionBar, en donde nos permite generar mayor integración con el resto de widgets. Podemos utilizar el ToolBar como una vista más dentro de nuestro layout. A tener en cuenta:

- Es incompatible con el ActionBar, por tanto en el tema tenemos que deshabilitar el ActionBar

```
<style name="Base.Theme.Ejem07_menus" parent="Theme.Material3.DayNight.NoActionBar">
```

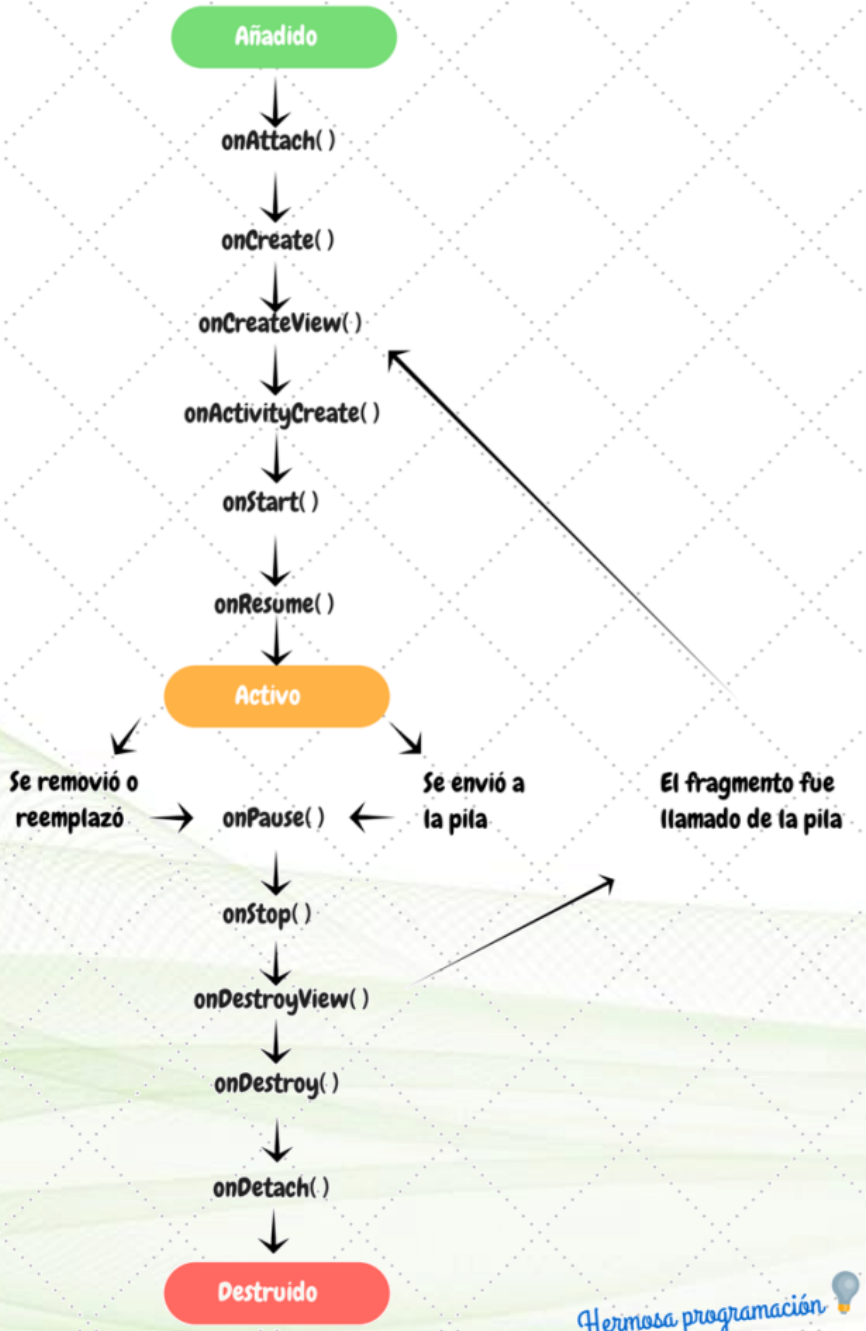
- Marcamos que el Toolbar registre el menú:

```
setSupportActionBar(binding.toolbar.miToolbar)
supportActionBar?.title = "Menús Activity 2"
supportActionBar?.setDisplayHomeAsUpEnabled(true)
```

- Implementamos en el segundo activity la finalización del mismo para volver al primero

```
override fun onSupportNavigateUp(): Boolean {
        finish()
        return super.onSupportNavigateUp()
    }
```

Fragment



Hermosa programación

Centro de Enseñanza
Gregorio Fernández

Fragments

• Fragment estático

Enlazamos un FragmentContainerView al XML que representa ese Fragment.

```
<androidx.fragment.app.FragmentContainerView
            android:name="com.example.ejem08_fragmentsholamundo.FragmentInicial"
```

• Fragment dinámico

```
supportFragmentManager.beginTransaction().apply {
    add(R.id.fragmentContainerView, FragmentInicial())   -> add/remove/replace
    commit()
}
```

Fragments (pasando datos)

- Del Activity al Fragment: creación del Fragment con newInstance
- Del Fragment al Activity (Con interface – 1ª forma):

```
class MyFragment : Fragment() {
    private lateinit val activityDependiente: EnviandoDatos
    interface EnviandoDatos {
        fun enviarDatos(datos: String)
    }
    override fun onAttach(context: Context) {
        super.onAttach(context)
        if (context is EnviandoDatos) {
 activityDependiente = context
        } else {
            throw ClassCastException("$context must implement
OnDataPassListener")
        }
    }
}
```

ViewModel

```
class Contador : ViewModel(){
    var contador=0
    fun incrementar(){
        contador++
    }
}
override fun onCreate(savedInstanceState: Bundle?) {
        ……

        var c: Contador = ViewModelProvider(this).get(Contador::class.java)
        binding.tvContador.text = c.contador.toString()

        binding.bIncrementar.setOnClickListener() {
            c = ViewModelProvider(this).get(Contador::class.java)
            c.incrementar()
            binding.tvContador.text = c.contador.toString()
        }
   }
```

Fragments (pasando datos)

- Del Fragment al Activity (Con ViewModel – 2ª forma):

```
class SharedViewModel : ViewModel() {
    val data = MutableLiveData<String>()
    fun setData(newData: String) {
        data.value = newData
    }
}
class MyActivity : AppCompatActivity() {
    private lateinit var sharedViewModel:
SharedViewModel
    override fun onCreate(savedInstanceState:
Bundle?) {
        super.onCreate(savedInstanceState)
        sharedViewModel =
ViewModelProvider(this)[SharedViewModel::class.java
]


        sharedViewModel.data.observe(this) { data -
>
            // Manejar el dato recibido
        }
    }
}
class MyFragment : Fragment() {
    private lateinit var sharedViewModel:
SharedViewModel
    override fun onViewCreated(view: View,
savedInstanceState: Bundle?) {
        super.onViewCreated(view,
savedInstanceState)
        sharedViewModel =
ViewModelProvider(requireActivity())[SharedViewMode
l::class.java]


        // Para pasar datos a la actividad
        sharedViewModel.setData("mi dato")
    }
}
```

Fragments (pasando datos)

- Del Fragment al Activity o a otro Fragment (3ª forma)

EMISOR:

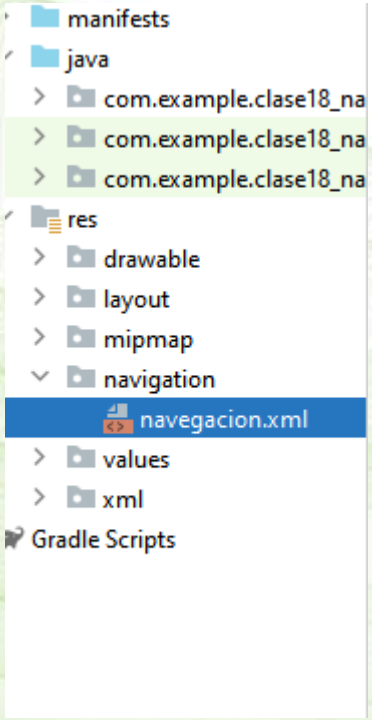
```
parentFragmentManager.setFragmentResult("requestKey", bundleOf("dataKey" to "mi dato"))
```

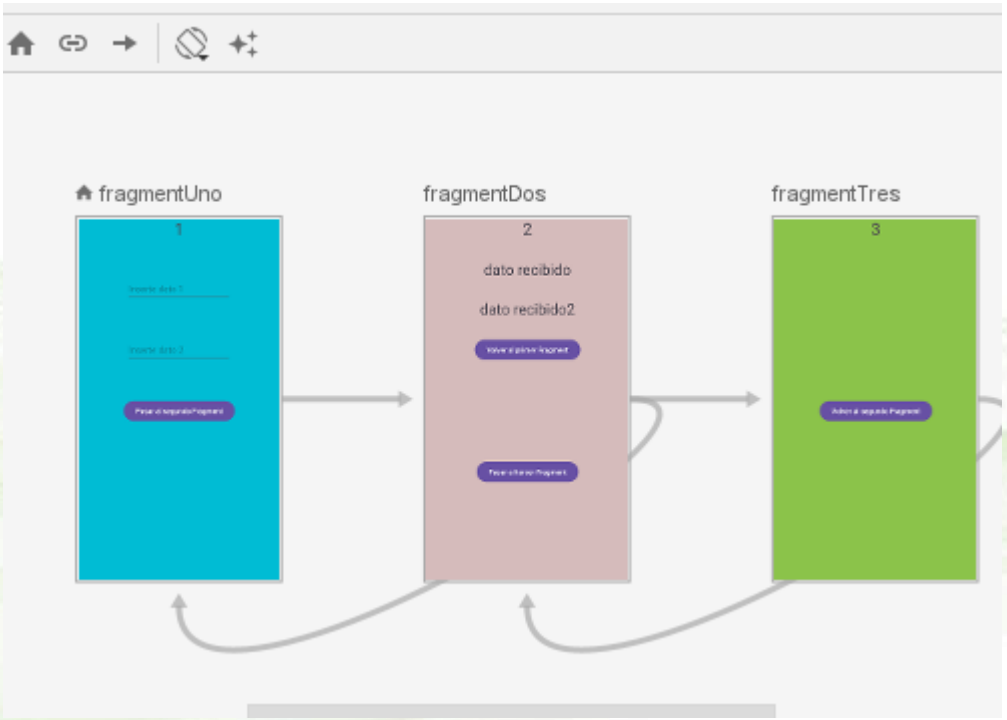
RECEPTOR:

```
supportFragmentManager.setFragmentResultListener("requestKey", this) { key, bundle ->
    val result = bundle.getString("dataKey")
    // Manejar el dato recibido
}
```

NavigationComponent

- Permite movernos entre Fragments/Activitys definidos en nuestra aplicación
- Facilita el paso de datos





NavigationComponent - Requerimientos

- En el fichero **build.grade.kts**

```
implementation("androidx.navigation:navigation-fragment:2.7.6")
implementation("androidx.navigation:navigation-ui:2.7.6")
```

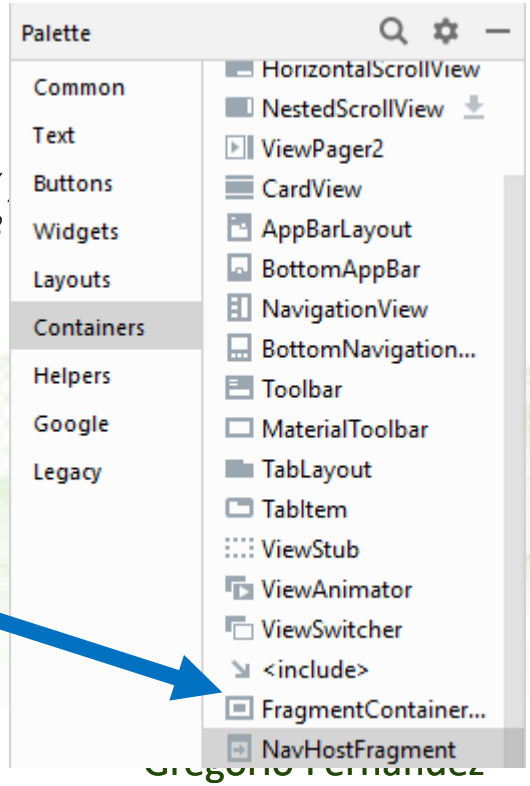
- En el código:

```
val mensaje = requireArguments()?.getString("dato_recibido")
binding.tvDatoRecibido.text = mensaje

binding.bAvanzar.setOnClickListener(){
    val dato:Bundle = Bundle()
    dato.putString("dato_recibido", binding.tietDato.text.toString(
    findNavController().navigate(R.id.action_fragment2_to_fragment3
}
```

En el **contenedor**:

```
<androidx.fragment.app.FragmentContainerView
    android:id="@+id/fragmentContainerView"
    android:name="androidx.navigation.fragment.NavHostFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:defaultNavHost="true"
    app:navGraph="@navigation/navegacion"
….
/>
```



Animaciones

- @android:anim/**slide_in_left**: Desliza el nuevo fragmento desde la izquierda hacia la posición visible
- @android:anim/**slide_in_right**: Desliza el nuevo fragmento desde la derecha hacia la posición visible
- @android:anim/**slide_out_right**: Desliza el fragmento actual hacia la derecha fuera de la pantalla.
- @android:anim/**fade_in**: El nuevo fragmento aparezca gradualmente.
- @android:anim/**fade_out**: El fragmento actual desaparezca gradualmente.
- …….