

# CFGS DESARROLLO DE APLICACIONES MULTIPLATAFORMA

## UD 4. Fragments

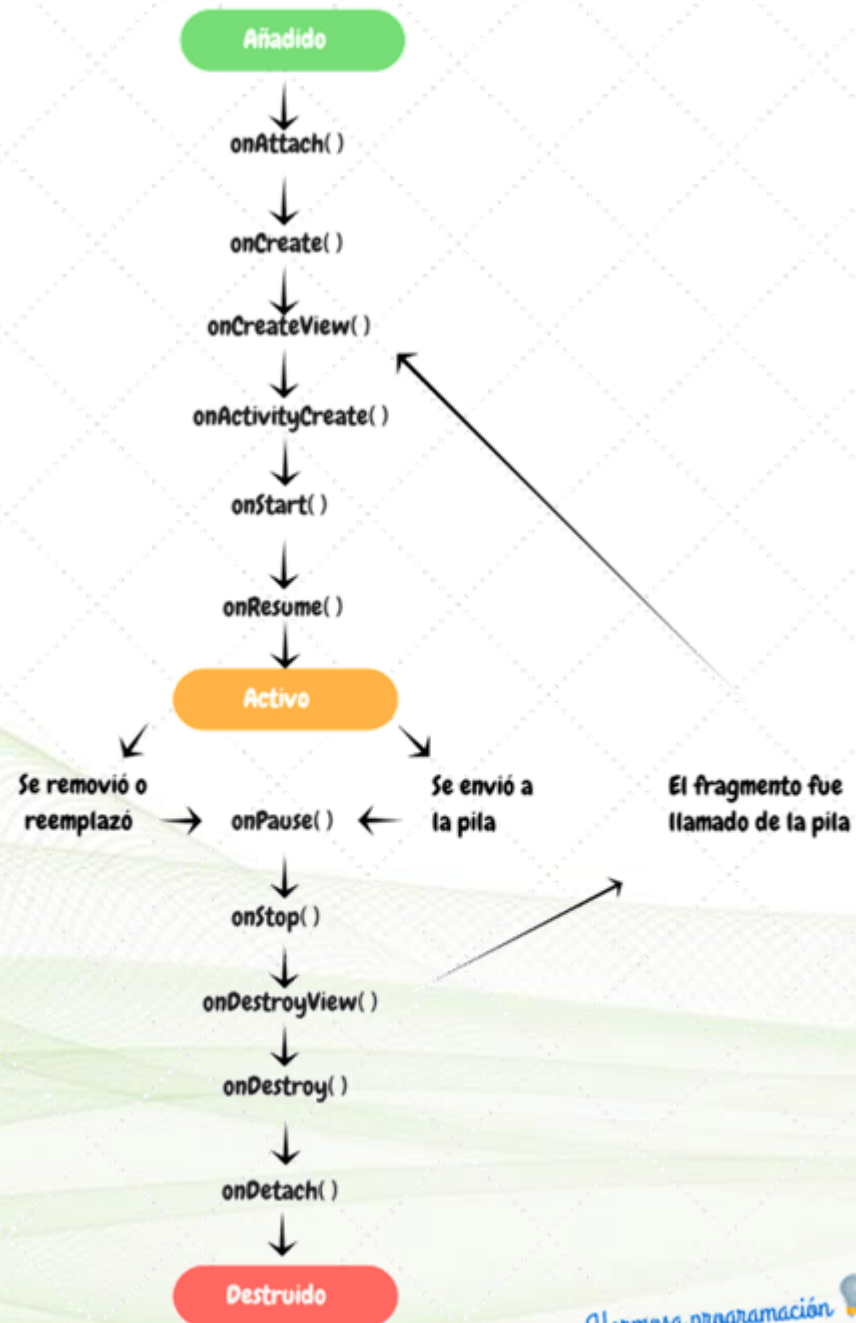
**Módulo: Programación multimedia y dispositivos móviles**

*Víctor J. Vergel Rodríguez*



Centro de Enseñanza  
Gregorio Fernández

# Fragment



# Fragments

- **Fragment estático**

Enlazamos un `FragmentManager` 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")  
        }  
    }  
}
```



# 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 onCreateView(view: View, savedInstanceState: Bundle?) {
        super.onCreateView(view, savedInstanceState)
        sharedViewModel =
            ViewModelProvider(requireActivity())[SharedViewModel::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
}
```



# Menú de aplicación y contextual

- Android Resource Directory de tipo menú

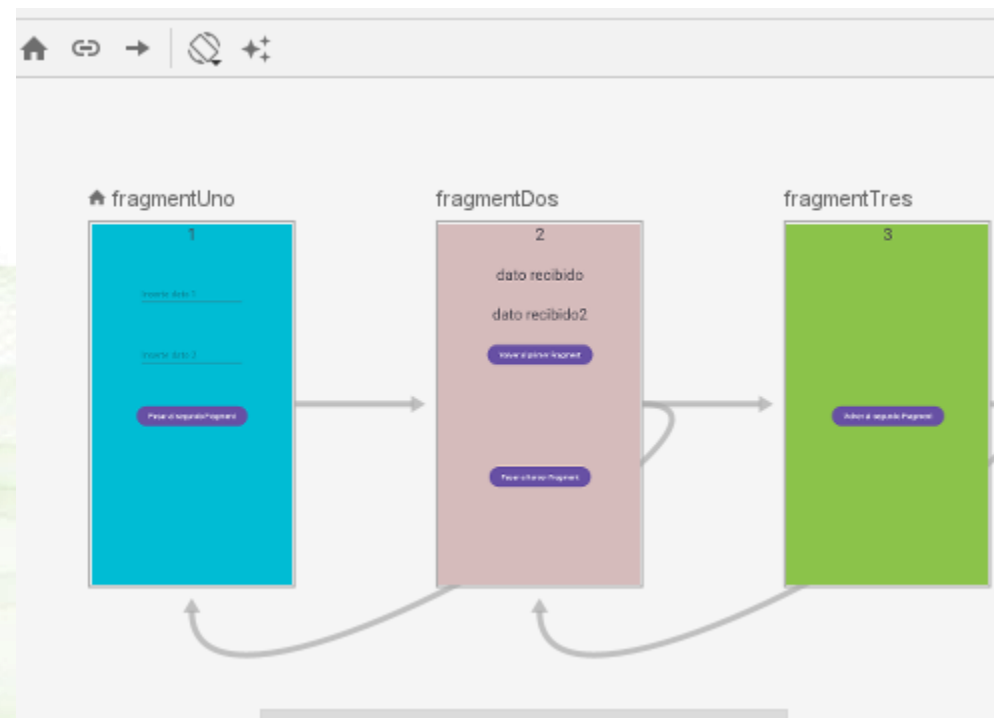
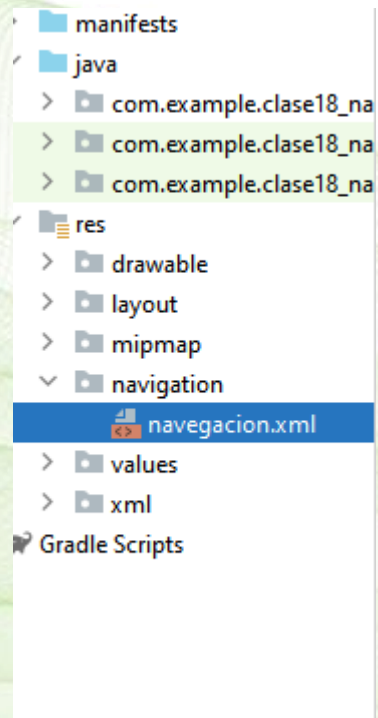
```
override fun onCreateOptionsMenu(menu: Menu?): Boolean {  
    // El menuInflater "infla" el XML y lo convierte en objetos reales  
    menuInflater.inflate(R.menu.main_menu, menu)  
    return true  
}
```

```
override fun onOptionsItemSelected(item: MenuItem): Boolean {  
    return when (item.itemId) {  
        R.id.action_settings -> {  
            // Aquí pones el código para abrir los ajustes  
            abrirConfiguracion()  
            true  
        }  
        else -> super.onOptionsItemSelected(item)  
    }  
}
```



# NavigationComponent

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



# NavigationComponent - Requerimientos

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

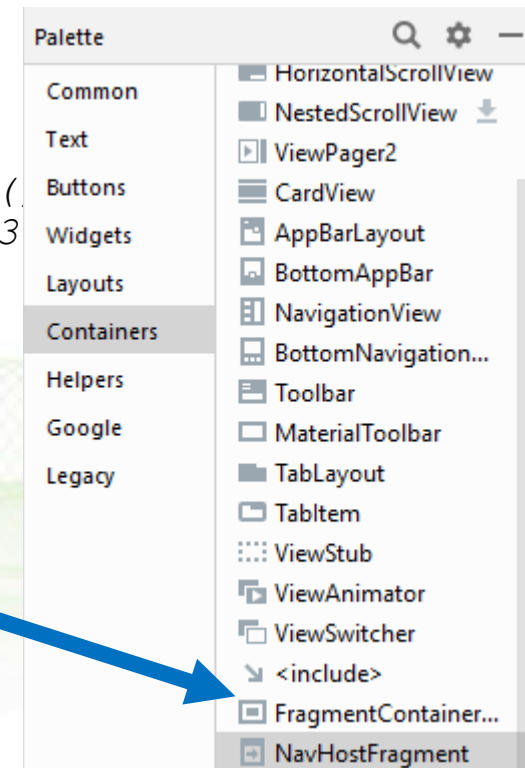
```
implementation("androidx.navigation:navigation-fragment:2.7.6")  
implementation("androidx.navigation:navigation-ui:2.7.6")
```

- En el código:

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

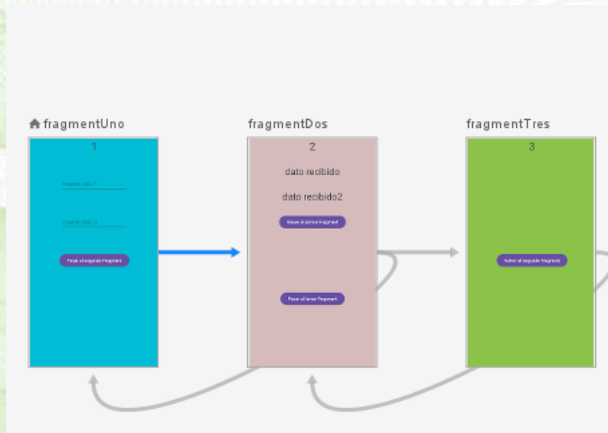
En el **contenedor**:

```
<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.
- .....



→ action	action_fri	
id	action_fragmentUno_to_fragment	
destination	fragmentDos	
▼ Animations		
enterAnim		
exitAnim	@android:anim/slide_in_left	
popEnterAnim		
popExitAnim		
▼ Argument Default Values		
dato	string	defau
▼ Pop Behavior		
popUpTo		



# Pasar a NavView

- Creamos un menú con los mismos identificadores que aparecen en el xml de navegación

```
<item android:title="fragment3"
      android:icon="@drawable/ic_android_morado_24dp"
      android:id="@+id/fragment3"
/>
```

- Añadimos al activity\_main.xml el bottomNavigation:

```
<com.google.android.material.bottomnavigation.BottomNavigationView
.....

    app:menu="@menu/bottom_nav_menu" />
```

- Cambiamos androidx.fragment.app.FragmentContainerView por fragment y añadimos en el MainActivity:

```
val navView: BottomNavigationView = binding.navView

val navController = findNavController(R.id.fragmentContainerView3)
val appBarConfiguration = AppBarConfiguration(
    setOf(
        R.id.fragment1, R.id.fragment2, R.id.fragment3
    )
)
setupActionBarWithNavController(navController, appBarConfiguration)
navView.setupWithNavController(navController)
```

- Y habilitar en el tema el ActionBar quitando NoActionBar

