

Aplicaciones navegables



KEEPCODING

Tech School

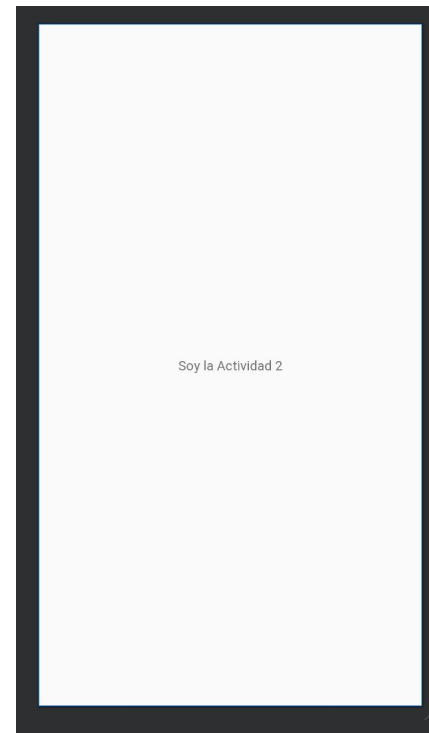




Aplicación con varias Activities

| ¿Cómo crear Activities? I

(Paso opcional) Se crea un nuevo layout



¿Cómo crear Activities? II

Se crea la nueva Actividad

```
class SecondActivity : AppCompatActivity() {  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
  
        super.onCreate(savedInstanceState)  
  
        setContentView(R.layout.activity_second)  
  
    }  
  
}
```

¿Cómo crear Activities? III

Se actualiza el Manifest

```
<application
```

```
...
```

```
<activity android:name=".SecondActivity"/>
```

```
</application>
```

¿Cómo crear Activities IV?

Se modifica MainActivity para llamar a la nueva Actividad

```
class MainActivity : AppCompatActivity() {  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
  
        ...  
  
        button.setOnClickListener {  
  
            val intent = Intent(this, SecondActivity::class.java)  
  
            startActivity(intent)  
  
        }  
  
    }  
}
```

Ejemplo Resumen I

Proyecto en GitHub

<https://github.com/KeepCodingMobile16/Fundamentos-Android/commit/b4d633799799c3e32f92f00c91e6376fe141d58f>



GitHub



Ejercicio

Actualiza el “Ejercicio Ciclo de Vida” para que cubra ambas Activities y pruebalo

¿Cómo enviar datos entre activities? I

La manera más simple de enviar datos de una Activity a otra es enviar un intent desde MainActivity

```
val intent = Intent(context, SecondActivity::class.java)
```

```
intent.putExtra("Clave", saludo)
```

```
startActivity(intent)
```

Recibiendolo del siguiente modo:

```
val saludo = intent.getStringExtra("Clave")
```

¿Cómo enviar datos entre activities? II

Por convenio, es mejor declarar la clave como un `companion object` en `SecondActivity`.

```
companion object {  
  
    const val CLAVE_1 = "ClaveSecondActivityString"  
  
}
```

¿Cómo enviar datos entre activities? III

Alternativamente, se puede optar por crear una función estática en `SecondActivity` agrupando todo lo necesario para ejecutar satisfactoriamente un intent.

```
fun getIntent(context: Context, saludo : String): Intent {  
  
    val intent = Intent(context, SecondActivity::class.java)  
  
    intent.putExtra(CLAVE_1, saludo)  
  
    return intent }
```

Alternativamente Siendo llamado desde `MainActivity`

```
startActivity(SecondActivity.getIntent(this,"Hola, ¿Que tal?"))
```

¿Cómo enviar datos complejos entre activities?

Es posible transformar la clase a un json (String), enviándolo como un String normal

Para ello podemos utilizar librerías como gson

```
dependencies {  
  
    implementation 'com.google.code.gson:gson:2.10.1'  
  
}
```

Ejercicio Resumen

Proyecto en GitHub


<https://github.com/KeepCodingMobile16/Fundamentos-Android/commit/ebb335bbf91781662d7e75e3cd04134c1d739d6e>



GitHub



Fragments

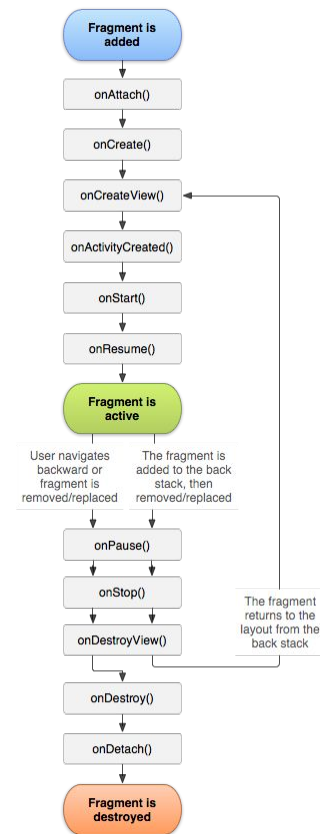


| ¿Qué son?

Un Fragmento se puede entender como una mini-actividad, teniendo un ciclo vida y una interfaz propia.

Sin embargo, un fragmento siempre debe estar alojado en una actividad y el ciclo de vida del fragmento se ve afectado directamente por el ciclo de vida de la actividad anfitriona.

Ciclo de vida del Fragment



¿Dónde alojar un Fragment?

El proceso para crear un fragment consiste en:

Se crea en tu activity layout un lugar donde el fragment se aloja:

```
<FrameLayout
```

```
    android:id="@+id/frameLayout"
```

```
    android:layout_width="0dp"
```

```
    android:layout_height="0dp"/>
```

```
</FrameLayout>
```

¿Cómo crear una interfaz para un fragment?

Se crea un layout para el Fragment.

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<androidx.constraintlayout.widget.ConstraintLayout  
xmlns:android="http://schemas.android.com/apk/res/android"
```

```
xmlns:app="http://schemas.android.com/apk/res-auto"
```

```
xmlns:tools="http://schemas.android.com/tools"
```

```
android:layout_width="match_parent"
```

```
android:layout_height="match_parent"
```

```
android:background="#C33A3A">
```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

¿Cómo asignar la interfaz a un Fragment?

Se crea una clase que extiende a Fragment y se le asigna el Layout recién creado.

```
class FragmentRojo : Fragment() {  
  
    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?, savedInstanceState:  
Bundle?): View? {  
  
        return inflater.inflate(R.layout.fragment_rojo, container, false)  
  
    }  
  
}
```

¿Cómo mostrar un Fragment?

Desde la actividad se crea el Fragment y se le aloja en la ubicación especificada anteriormente.

```
button.setOnClickListener {  
    val fragmentTransaction = supportFragmentManager.beginTransaction()  
    fragmentTransaction.add(R.id.frameLayout, nextFragment)  
    fragmentTransaction.commit()  
}
```

Ejercicio Resumen

<https://github.com/KeepCodingMobile16/Fundamentos-Android/commit/a14003cb2901509316bc1e1e1548d656009c2e45>



GitHub

¿Cómo enviar información a un Fragment? I

Es posible utilizar las funciones que implemente el Fragment, sin embargo lo recomendable es enviar los datos en los arguments.

```
PersonajeFragment().apply {  
  
    setCualquierNumero(1)  
  
}
```

Para ello, el fragment tiene que tener una función llamada setCualquierNumero

¿Cómo enviar información a un Fragment? II

La comunicación entre ambos se debe realizar a través de un listener que implementa la Activity y que el fragment ejecuta:

```
interface MainActivityFragmentInterface { fun onClick() }
```

Añadimos una función al Fragment:

```
fun setListener(mainActivityFragmentInterface: MainActivityFragmentInterface) {
```

```
    this.mainActivityFragmentInterface = mainActivityFragmentInterface }
```

Para llamarlo:

```
mainActivityFragmentInterface.onClick()
```

| ¿Cómo enviar información a un Fragment? III

Utilizar un viewModel compartido

byActivityViewModels



View Model en Fragments



| ¿Cómo se usan?

Del mismo modo que con las activities

Por defecto, el viewModel se enlaza al Fragment ofreciendo las mismas ventajas ya vistas

Sin embargo, es posible **enlazar un viewModel creado a nivel de activity a un Fragment**



ViewModel de nivel de Activity

Se ve afectado por el ciclo de la activity en vez del Fragment

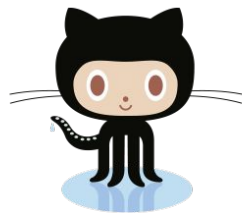
Le permite sobrevivir al Fragment

Le permite se reclamado por otro Fragment

Y por tanto... compartir información de forma sencilla y segura

Ejercicio Resumen

<https://github.com/KeepCodingMobile16/Fundamentos-Android/commit/0b81b7a1a6f3559492b53a03d13fbfea08a43421>



GitHub

<https://github.com/KeepCodingMobile16/Fundamentos-Android/commit/527109833a79830803dce7c781b944c9aa5ff93e>

Fin

