Tema 7. Fragments

7. Fragments

Aparecieron a partir de la ***API 11*** (Android 3.0), permitiendo tener interfaces de usuario más flexibles. Añaden la posibilidad de dividir la interfaz de usuario en porciones independientes que se organizan según el tamaño y la orientación de la pantalla.

Desde el punto de vista constructivo, los fragmentos son secciones que se incluyen en una *actividad (activity)*. Aunque tienen su propio ciclo de vida, están ligados al ciclo de vida del activity donde se encuentran contenidos. Es importante indicar que un *Fragment* no puede existir por sí mismo, sino que debe crearse dentro de un Activity. Los fragmentos se pueden utilizar en una o más actividades. Además, tienen sus propios eventos.

El ciclo de vida de un fragment es parecido al de un activity, pero incluye nuevos estados.

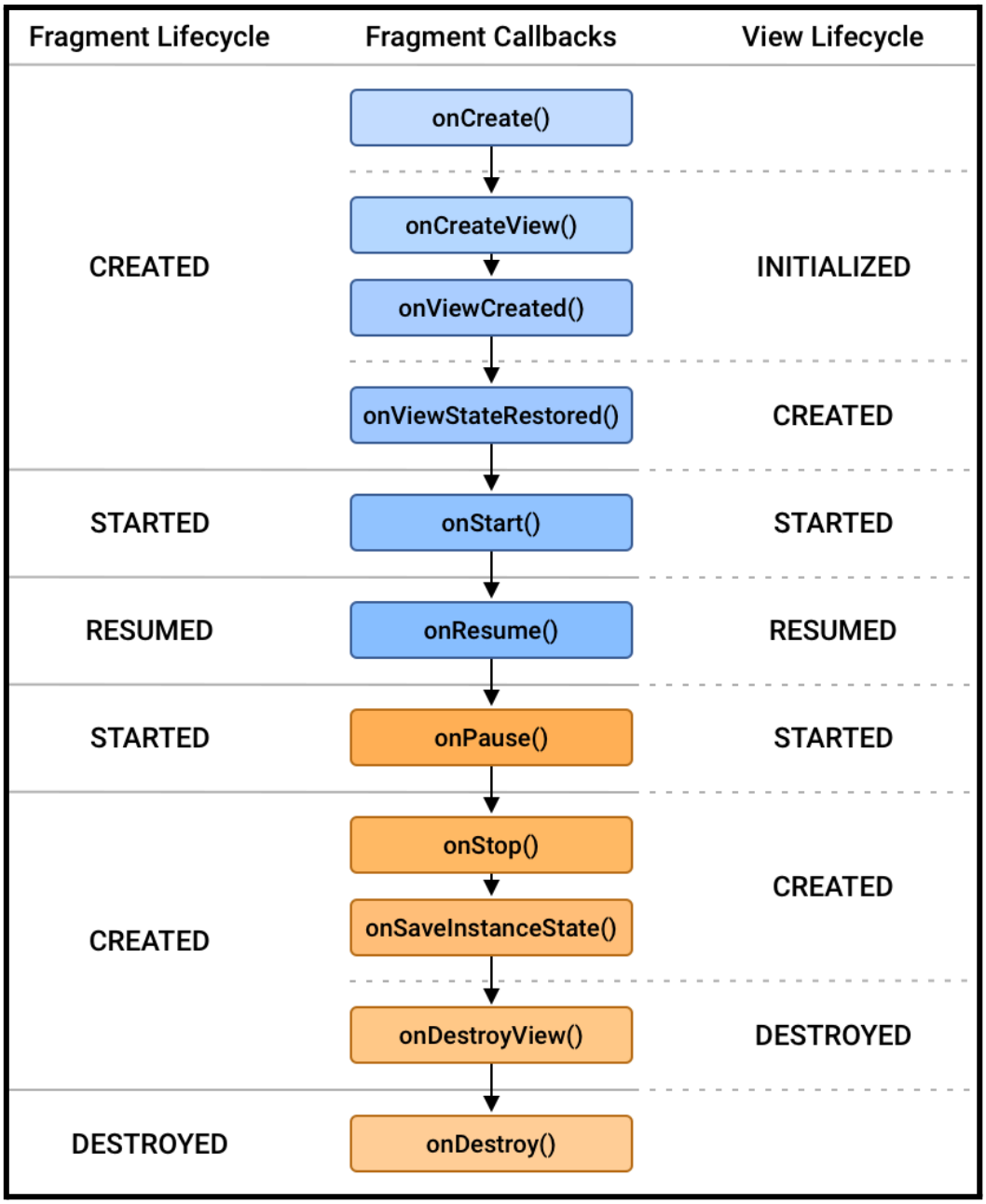
1. **onAttach( )**: liga el activity a nuestro fragment.

2. **onCreateView**( ): devuelve la vista que contendrá dicho fragment.

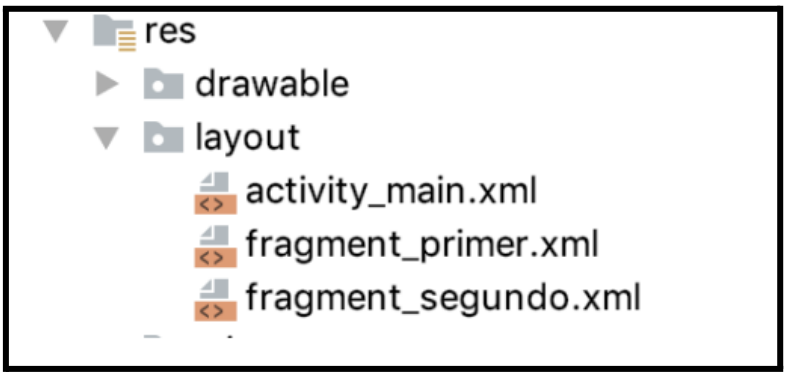
3. **onViewCreated**( ): nos avisa de que ya está todo disponible. Este es un buen lugar donde añadir las funciones *onClick()* o inicializar variables del fragment.

Estados de destrucción

Contiene tres funciones distintas que se irán llamando de manera secuencial una vez los procesos van terminando. El primero será **onDestroyView()** que se llamará cuando la vista vaya a ser destruida, al terminar pasará por **onDestroy()** que pondrá fin a la vida del fragment, terminando por **onDetach()**, que hará lo contrario a la función **onAttach()** que vimos al principio, básicamente desliga la activity con el fragment.



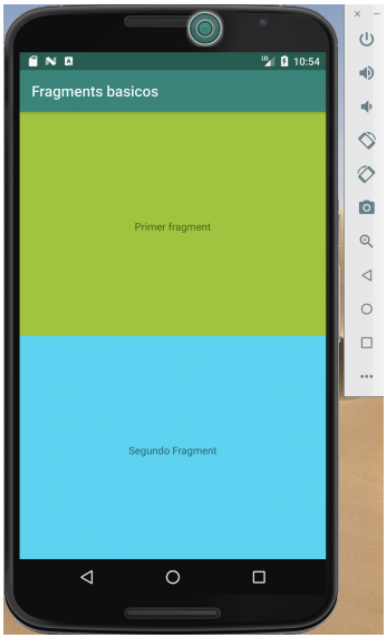
Cada fragmento tendrá asociado un fichero XML (vista) y un fichero en Kotlin (comportamiento). En la siguiente imagen, puedes ver cómo aparecen los ficheros XML asociados a los fragments en la carpeta /res/layouts.



7.1 Fragments estáticos

Solemos hablar de *fragments estáticos*, cuando dividimos la pantalla de un *Activity* en dos o más partes, pero dichos fragments siempre están visibles y no se intercambian unos por otros, en contraposición a los *fragments dinámicos*, donde podemos hacer aparecer o desaparecer alguno de ellos durante la ejecución de la app.

A continuación, se muestra un ejemplo de activity, en el que se ha dividido la pantalla usando dos *fragments estáticos*.

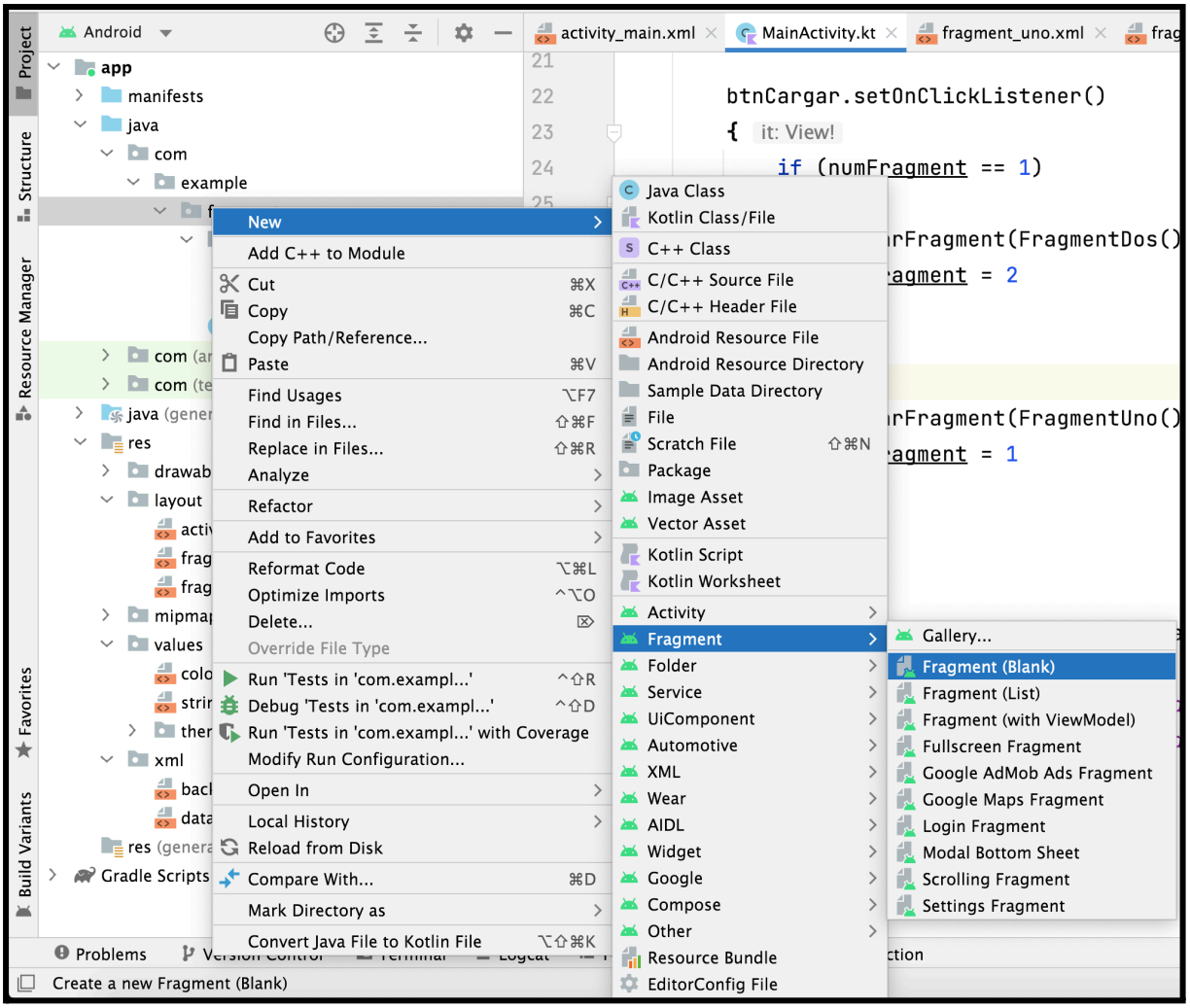


Enlace a la documentación oficial.

7.2 Fragments dinámicos

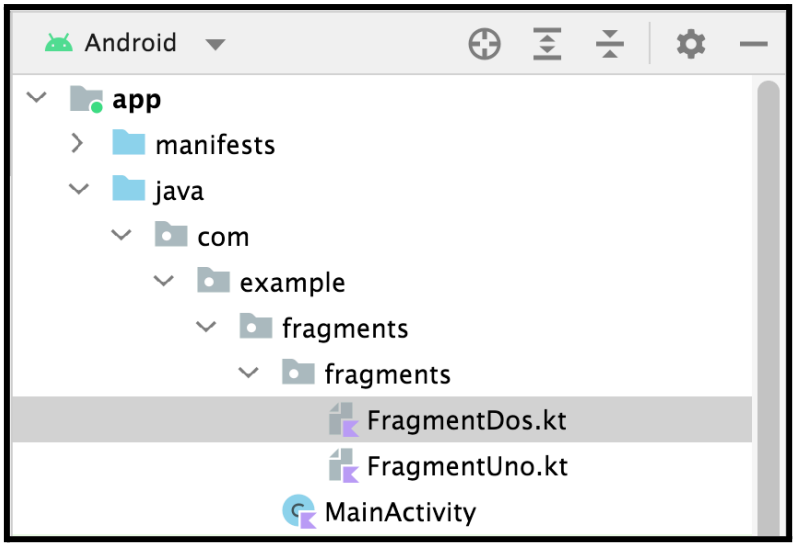
Los fragments de este tipo son aquellos que suelen estar dentro de un *FrameLayout*, y que pueden intercambiarse, mostrándose un Fragment u otro, en función de un evento, como puede ser pulsar un botón.

Para añadir un fragment a tu proyecto debes seguir los siguientes pasos:

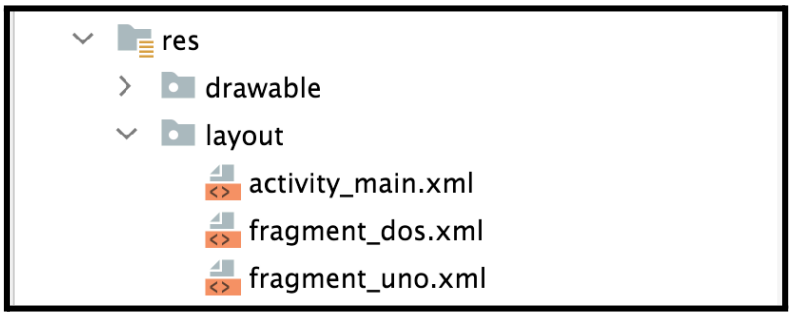


En el siguiente enlace tienes un proyecto de ***AndroidStudio***, que te puede servir de ejemplo.

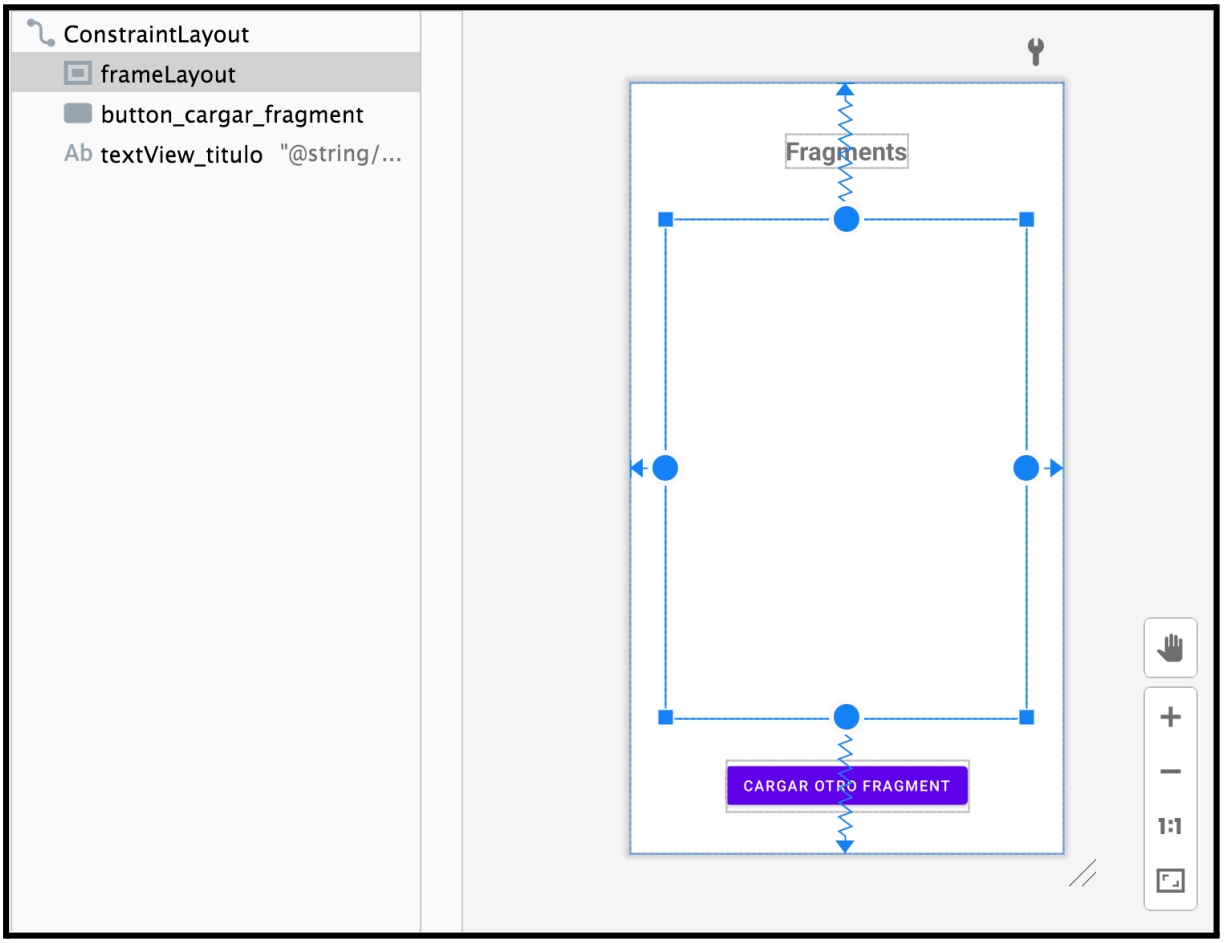
En este ejemplo verás que hay un activity y dos fragments.



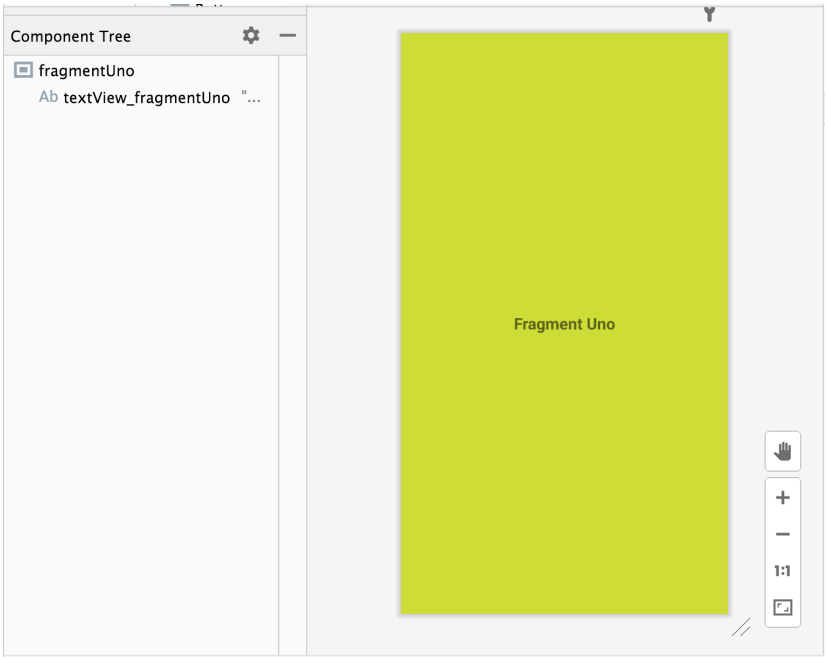
Igualmente, para cada fragment y activity, se tiene un fichero XML para definir la vista.



Dentro del layout del *Activity* se han definido tres views (objetos gráficos): un *TextView* que usa el nombre de la app, un *FrameLayout* (donde se van a cargar los fragments) y un *botón* (para cambiar el fragment cargado).



Los fragments simplemente incluyen un texto con el nombre del fragment, y un color de fondo.



Como puede verse a continuación, el fragment cargado, cambia al pulsar el botón.

Para conseguir este comportamiento en nuestra app, se ha usado el siguiente código:

```
class MainActivity : AppCompatActivity()
{
    override fun onCreate(savedInstanceState: Bundle?)
    {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        //Se carga el FragmentUno
        cargarFragment(FragmentUno())
        var numFragment=1
        val btnCargar: Button = findViewById(R.id.button_cargar_fragment)

        btnCargar.setOnClickListener()
        { it: View!
            if (numFragment == 1)
            {
                cargarFragment(FragmentDos())
                numFragment = 2
            }
            else
            {
                cargarFragment(FragmentUno())
                numFragment = 1
            }
```

Se tiene una variable llamada *numFragment*, que contendrá un 1 cuando esté cargado el FragmentUno y un 2, cuando esté cargado el FragmentDos.

La carga del fragment se realiza con el método *cargarFragment()*. Este método hace uso de la clase *supportFragmentManager* para llevar a cabo la operación.

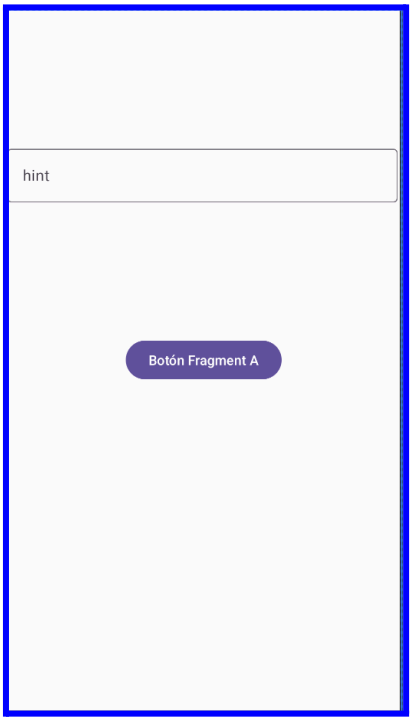
```
    private fun cargarFragment(fragment: Fragment)
    {
        val fragmentTransaction = supportFragmentManager.beginTransaction()
        fragmentTransaction.add(R.id.frameLayout, fragment)
        fragmentTransaction.commit()
    }
```

7.3 Paso de parámetros entre fragments

En Android, los ***fragments*** pueden comunicarse entre sí a través de diferentes formas. En este caso, vamos a ver cómo se produce esa comunicación, utilizando el atributo arguments, que poseen todos los fragments.

Supongamos que tienes dos fragments: FragmentA y FragmentB.

El diseño del FragmentA es el siguiente:



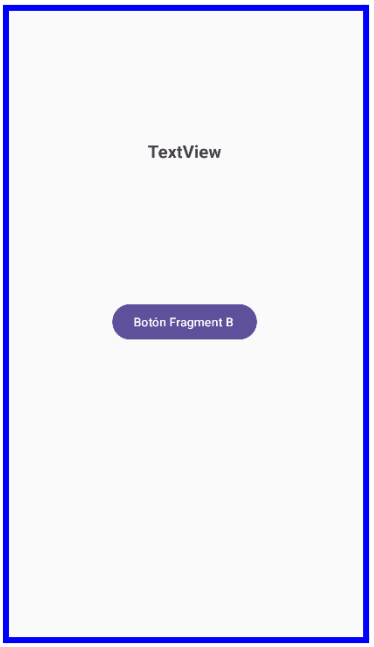
Como se aprecia en la imagen, el diseño consiste en un TextInputLayout donde se va a escribir el texto a pasar al otro fragment (FragmentB) y un botón, que al ser pulsado, copiará la información del FragmentA al FragmentB.

El código en Kotlin de dicho fragment es:

```kotlin
class FragmentA : Fragment()
{
    companion object {

        //Patrón Singleton
        private var instance: FragmentA? = null

        fun getInstance(): FragmentA {
            if (instance == null) {
                instance = FragmentA()
            }
            return instance!!
        }
    }

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflar el diseño del fragmento aquí
        return inflater.inflate(R.layout.fragment_a, container,  attachToRoot: false)
    }
```

```kotlin
    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        // Acción que activa la comunicación con FragmentB
        val texto = view.findViewById<TextInputEditText>(R.id.textoLeido)

        val buttonSendData = view.findViewById<Button>(R.id.buttonSendData)

        buttonSendData.setOnClickListener { it: View!
            // Crear un Bundle para enviar datos a FragmentB
            val bundle = Bundle()
            Toast.makeText(context, texto.text.toString(), Toast.LENGTH_LONG).show()
            bundle.putString("data", texto.text.toString())

            // Iniciar FragmentB y pasarle el Bundle
            val fragmentB = FragmentB.getInstance()
            fragmentB.arguments = bundle
        }
    }

}
```

El diseño del FragmentB es el siguiente:



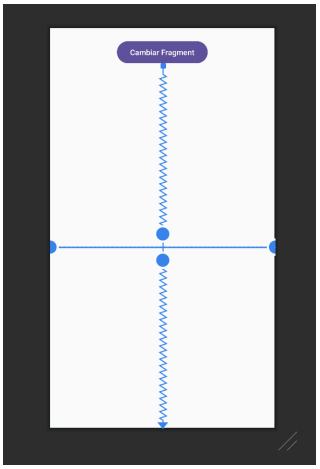
Como se aprecia en la imagen, el diseño consiste en un TextView donde se va a mostrar el texto pasado desde el otro fragment (FragmentA) y un botón, que al ser pulsado, mostrará la información del FragmentA en dicho TextView.

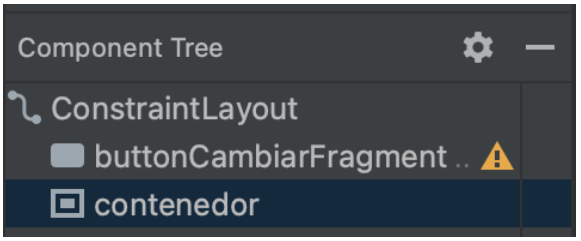
```
class FragmentB : Fragment()
{
    companion object {

        //Patrón Singleton
        private var instance: FragmentB? = null

        fun getInstance(): FragmentB {
            if (instance == null) {
                instance = FragmentB()
            }
            return instance!!
        }
    }

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflar el diseño del fragmento aquí

        return inflater.inflate(R.layout.fragment_b, container,  attachToRoot: false)
    }
```

```
override fun onViewCreated(view: View, savedInstanceState: Bundle?)
{
    super.onViewCreated(view, savedInstanceState)

    val buttonReceiveData = view.findViewById<Button>(R.id.buttonReceiveData)

    // Acción que activa la comunicación con FragmentA
    buttonReceiveData.setOnClickListener{ it: View!

        // Obtener datos del Bundle enviado desde FragmentA
        val bundle = arguments
        val inputData = bundle?.getString( key: "data")

        val texto = view.findViewById<TextView>(R.id.tvTexto)

        if (inputData != null)
        {
            texto.setText(inputData)
        }
        else
        {
            texto.setText("Ha habido un error")
        }

    }
}
```

Como se ha indicado al inicio de la sección, la comunicación se ha realizado, sin utilizar al Activity que contiene a ambos fragments. Sin embargo, sí es necesario gestionar desde este Activity, los pasos para que se muestre un fragment u otro.

Además, es interesante ver cómo suele ser el diseño de un Activity que va a contener diferentes fragments. Como se aprecia en la imagen, se dispone de un botón para gestionar el evento de cambiar de vista (fragment), y por otro lado, un elemento de tipo FrameLayout, que es el que contiene propiamente la vista de cada fragment.





```
class MainActivity : AppCompatActivity(), View.OnClickListener
{
    private var tipo = 0
    private var fragmentA = FragmentA.getInstance()
    private var fragmentB = FragmentB.getInstance()

    override fun onCreate(savedInstanceState: Bundle?)
    {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val boton = findViewById<Button>(R.id.buttonCambiarFragment)
        boton.setOnClickListener(this)

        supportFragmentManager.beginTransaction()
            .replace(R.id.contenedor, fragmentA)
            .addToBackStack( name: null)
            .commit()

    }
```

Se ha creado un botón, llamado *buttonCambiarFragment*, que al ser pulsado alternará entre los fragments A y B.

Ese intercambio, se lleva a cabo dentro del método ***onClick()***, donde se trabaja con una variable llamada tipo, que almacena el fragment que ocupa el Activity en cada momento.

```
    override fun onClick(p0: View?)
    {
        if (tipo == 0)
        {
            supportFragmentManager.beginTransaction()
                .replace(R.id.contenedor, fragmentB)
                .commit()

            tipo = 1
        }
        else
        {
            supportFragmentManager.beginTransaction()
                .replace(R.id.contenedor, fragmentA)
                .commit()

            tipo = 0
        }
    }

}
```

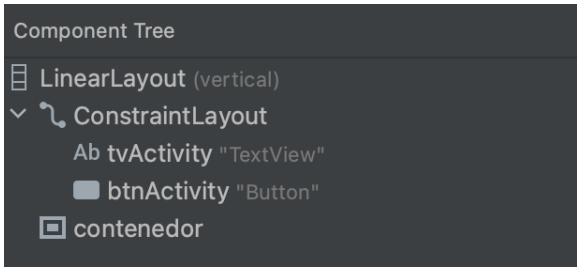
7.4 Comunicación bidireccional entre fragment y activity

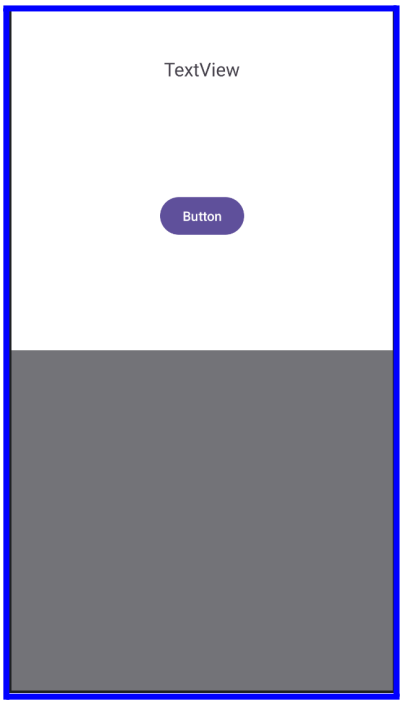
La comunicación bidireccional entre un Fragment y una Activity en Android se puede lograr a través de interfaces.

En primer lugar, creamos la interfaz ***CommunicationListener.kt***

```kotlin
package com.example.fragment_activity

interface CommunicationListener
{
    fun sendDataToFragment(data: String)
    fun sendDataToActivity(data: String)
    fun receiveDataFromFragment(data: String)
    fun receiveDataFromActivity(data: String)
}
```

El diseño del layout del activity es el siguiente:





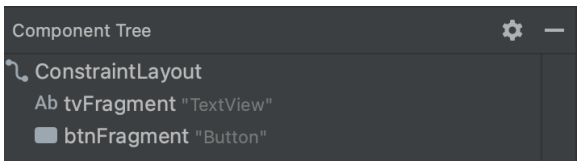
Donde podemos ver que, el contenedor principal es un LinearLayout vertical, dividido en dos partes. Cada una de las partes ocupa un 50% del tamaño.

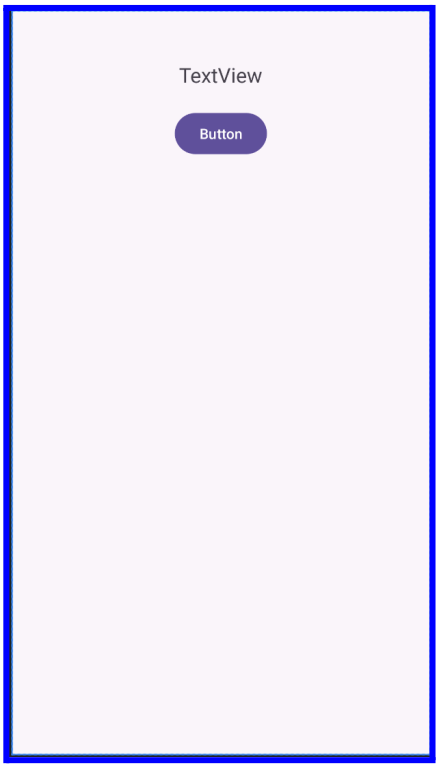
La primera parte serían los elementos incluidos en la vista activity, y la segunda, un ***FrameLayout***, donde se va a cargar el fragment.

A continuación, modificamos el activity, para implementar todos los métodos de las interfaces ***CommunicationListener*** y ***OnClickListener***.

```
class MainActivity : AppCompatActivity(), CommunicationListener, View.OnClickListener {

    override fun onCreate(savedInstanceState: Bundle?)
    {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val fragment = MyFragment()
        supportFragmentManager.beginTransaction().replace(R.id.contenedor, fragment).commit()

        val btn = findViewById<Button>(R.id.btnActivity)
        btn.setOnClickListener(this)
    }

    override fun sendDataToFragment(data: String)
    {
        val fragment = supportFragmentManager.findFragmentById(R.id.contenedor)

        if (fragment is MyFragment)
        {
            fragment.receiveDataFromActivity(data)
        }
    }

    override fun receiveDataFromFragment(data: String)
    {
        val texto = findViewById<TextView>(R.id.tvActivity)

        texto?.text = data
    }

    override fun onClick(p0: View?)
    {
        sendDataToFragment( data: "Llamando al Fragment")
    }
```

El diseño del fragment es:





Finalmente, tenemos la implementación en kotlin del *fragment*.

```kotlin
class MyFragment : Fragment(), CommunicationListener, View.OnClickListener {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
    }

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_my, container, attachToRoot: false)
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?)
    {
        super.onViewCreated(view, savedInstanceState)

        val btn = view.findViewById<Button>(R.id.btnFragment)
        btn.setOnClickListener(this)
    }
```

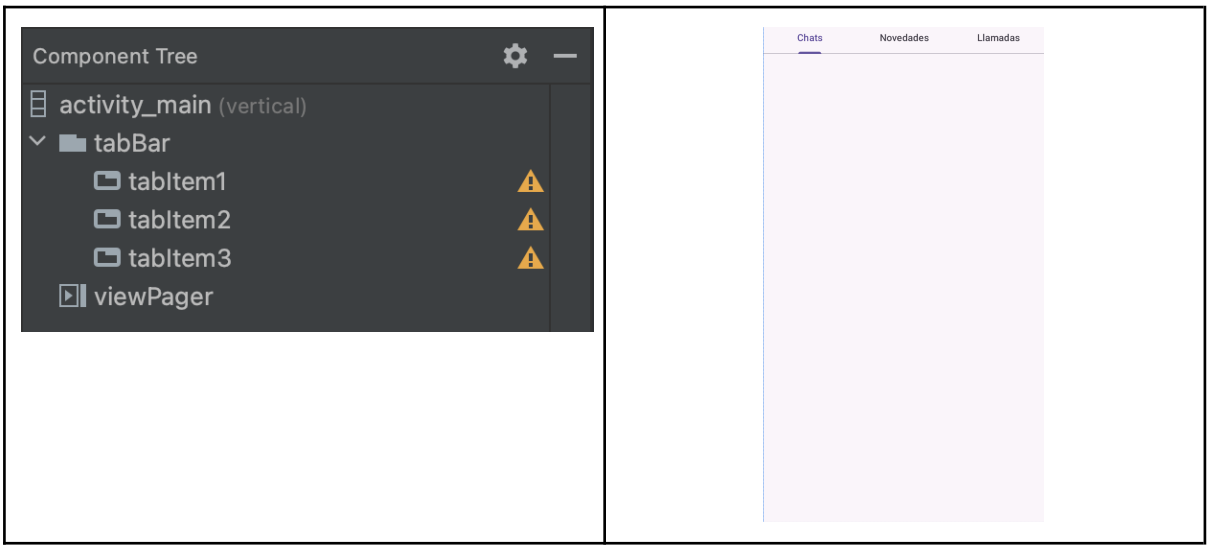
```kotlin
override fun sendDataToActivity(data: String)
{
    val activity = activity
    if (activity is CommunicationListener)
    {
        activity.receiveDataFromFragment(data)
    }
}

override fun receiveDataFromActivity(data: String)
{
    val texto = view?.findViewById<TextView>(R.id.tvFragment)
    texto?.text = data
}

override fun onClick(p0: View?)
{
    sendDataToActivity( data: "Llamando al Activity")
}
```

7.5 Pestañas/Tabs + ViewPager2

Las pestañas, o ***tabs***, permiten mostrar información de una manera más ordenada. Es un sistema muy utilizado actualmente en aplicaciones como WhatsApp.

Los pasos para crear un activity con varias tabs son:

<u>Paso 1. Crear la vista del activity.</u>



La vista contará con dos elementos principales: una ***tabBar*** y un ***viewPager***. El primero es el elemento gráfico, y el segundo el encargado de la gestión de los fragments asociados a cada pestaña.

Paso 2. Creación de los fragments.

Para este ejemplo, se van a crear tres fragments, donde cada uno de ellos contiene únicamente un TextView que permite diferenciar a cada uno de ellos.

Por ejemplo, para la pestaña Chats, se crea el siguiente fragment:

Paso 3. Desarrollar la lógica del fichero MainActivity.kt

```
class MainActivity : AppCompatActivity()
{
    override fun onCreate(savedInstanceState: Bundle?)
    {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        //TabLayout
        val tabLayout = findViewById<TabLayout>(R.id.tabBar)
        val tabChats = findViewById<TabItem>(R.id.tabItem1)
        val tabNovedades = findViewById<TabItem>(R.id.tabItem2)
        val tabLlamadas = findViewById<TabItem>(R.id.tabItem3)
        val viewPager = findViewById<ViewPager2>(R.id.viewPager)

        //Adaptador (como en el RecyclerView)
        val adapter = ViewPagerAdapter( activity: this)
        viewPager.adapter = adapter

        val tabLayoutMediator = TabLayoutMediator(tabLayout, viewPager) { tab, position ->
            tab.text =
                when (position) {
                    0 -> "Chats"
                    1 -> "Novedades"
                    2 -> "Llamadas"
                    else -> ""
                }
        }
        tabLayoutMediator.attach()

    }
```

Se rescatan los elementos de la vista

Se necesita un adaptador como con RecyclerView

Este elemento es parecido al LayoutManager del RecyclerclerView.

Gestiona posibles modificaciones en las pestañas, cuando son pulsadas

Como ocurría con ***RecyclerView***, es necesario crear un adaptador que gestione nuestra estructura de pestañas. Esta clase debe implementar los métodos *getItemCount()* y *createFragment()*.

```
private class ViewPagerAdapter(activity: AppCompatActivity) : FragmentStateAdapter(activity) {

    override fun getItemCount(): Int = 3

    override fun createFragment(position: Int): Fragment {
        return when (position) {
            0 -> Fragment_Chats()
            1 -> Fragment_Novedades()
            2 -> Fragment_Llamadas()
            else -> throw IllegalArgumentException("Invalid position: $position")
        }
    }
}
```