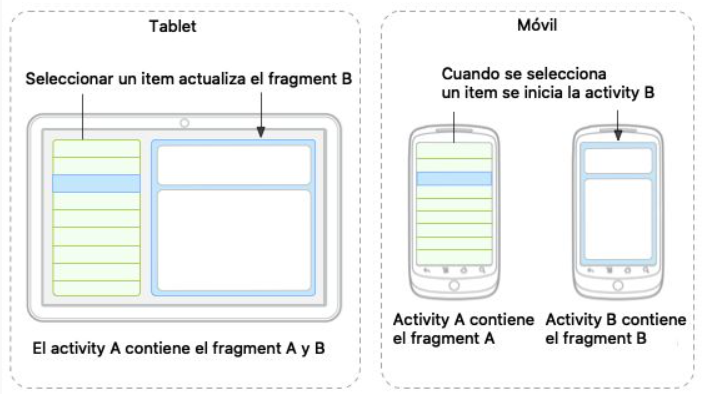
# 2.5 Fragments

Un **fragment** podría definirse como una porción de la interfaz de usuario que puede añadirse o eliminarse de la interfaz de forma independiente al resto de elementos de la actividad, y que por supuesto puede reutilizarse en otras actividades.



Para que entendamos lo que tenemos que hacer. En una aplicación en la que queramos utilizar “Fragments” tendremos que crear el diseño que deseemos que tenga dicho fragment (fragment\_porcion.xml) asociado a una clase que extienda de Fragment.

En una aplicación tendremos:

Fragment\_porcionX.xml asociado a clase FragmentPorcionX (**tantos como necesitemos**)

…

Luego tendremos nuestro activity\_main.xml asociado a clase MainActivity. En diseño, llamaremos al fragment que queramos que sea el de por defecto.

La clase que extiende del Fragment es la que enlaza con el diseño asociado.

Lo vemos mejor con un ejemplo.

## Ejemplo de uso

Vamos a crear una aplicación visor de correos. Tiene que contener:

### Un Fragment que muestre el listado

Su diseño contendrá un RecyclerView

|  |
| --- |
| <LinearLayout 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:orientation="vertical"  tools:context=".MainActivity">   <androidx.recyclerview.widget.RecyclerView  android:id="@+id/lstListado"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content" />  </LinearLayout> |

Su clase:

|  |
| --- |
| package com.example.probandofragments  import androidx.fragment.app.Fragment import android.os.Bundle import android.view.LayoutInflater import android.view.View import android.view.ViewGroup import androidx.recyclerview.widget.DividerItemDecoration import androidx.recyclerview.widget.LinearLayoutManager import androidx.recyclerview.widget.RecyclerView   class FragmentListado : Fragment() {   private lateinit var lstListado : RecyclerView   private val datos =  *MutableList*(5) **{** i **->** Correo("Persona $i", "Asunto del correo $i", "Texto del correo $i") **}** */\* Vamos a asignar un evento desde fuera del fragment.\*/* var listener : CorreosListener? = null   override fun onCreateView(  inflater: LayoutInflater,  container: ViewGroup?,  savedInstanceState: Bundle?  ): View {  return inflater.inflate(R.layout.*fragment\_listado*, container, false)  }   override fun onViewCreated(view:View,savedInstanceState: Bundle?) {  super.onViewCreated(view,savedInstanceState)   lstListado = view?.findViewById<View>(R.id.*lstListado*) as RecyclerView   val adaptador = AdaptadorCorreos(datos) **{** listener?.onCorreoSeleccionado(**it**)  **}** lstListado.*layoutManager* =  *//GridLayoutManager(this.context, 3)* LinearLayoutManager(this.*context*, LinearLayoutManager.*VERTICAL*, false)   lstListado.addItemDecoration(  DividerItemDecoration(this.*context*, DividerItemDecoration.*VERTICAL*)  )   lstListado.*adapter* = adaptador  }   fun setCorreosListener(l: CorreosListener) {  listener = l  }   */\*  \* Defino una función que establece un listener que se activa cuando se seleccione un Correo  \*/* fun setCorreosListener(seleccion: (Correo) -> Unit) {   */\*  \* object:CorreosListener crea una implementación de la interface CorreosListener  \* sin necesidad de crear una clase separada.  \* \*/* listener = object:CorreosListener {  override fun onCorreoSeleccionado(correo: Correo) {  seleccion(correo)  }  }  }    } |

### Un Fragment que muestre el detalle

Su diseño contendrá lo que deseamos que se vea con el mensaje.

|  |
| --- |
| <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:orientation="vertical"  android:background="#FFBBBBBB"  android:padding="8dp">   <TextView  android:id="@+id/txtDetalle"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:textSize="25sp"/>  </LinearLayout> |

Su clase asociada:

|  |
| --- |
| package com.example.probandofragments  import android.os.Bundle import android.util.Log import android.view.LayoutInflater import android.view.View import android.view.ViewGroup import android.widget.TextView import androidx.fragment.app.Fragment import androidx.recyclerview.widget.RecyclerView  class FragmentDetalle : Fragment() {   lateinit var txtDetalle:TextView  private var datosPendientes: String? = null   override fun onCreateView(  inflater: LayoutInflater,  container: ViewGroup?,  savedInstanceState: Bundle?  ): View {  return inflater.inflate(R.layout.*fragment\_detalle*, container, false)  }   override fun onViewCreated(view:View,savedInstanceState: Bundle?) {  super.onViewCreated(view,savedInstanceState)     *//Log.d("FragmentDetalle", "onViewCreated ejecutado")* txtDetalle = view.findViewById(R.id.*txtDetalle*)  *// Aplica los datos pendientes, si existen* datosPendientes?.*let* **{** txtDetalle.*text* = **it** datosPendientes = null  **}** }   fun mostrarDetalle(datos: String) {  if (::txtDetalle.*isInitialized*) {  txtDetalle.*text* = datos  } else {  Log.e("FragmentDetalle", "La vista no está inicializada.")  datosPendientes = datos  }  }  */\*fun mostrarDetalle(texto: String) {  txtDetalle = requireView().findViewById(R.id.txtDetalle) as TextView   txtDetalle.text = texto  }\*/* } |

### Lo necesario para el RecyclerView

El dataclass Correo

|  |
| --- |
| package com.example.probandofragments  data class Correo(  val de: String  , val asunto: String  , val texto: String) |

El adaptador:

|  |
| --- |
| package com.example.probandofragments  import android.view.LayoutInflater import android.view.View import android.view.ViewGroup import android.widget.LinearLayout import android.widget.TextView import androidx.recyclerview.widget.RecyclerView  class AdaptadorCorreos(  private val datos: MutableList<Correo>,  private val clickListener : (Correo) -> Unit) :  RecyclerView.Adapter<AdaptadorCorreos.CorreosViewHolder>() {  class CorreosViewHolder(val item: View) : RecyclerView.ViewHolder(item) {  val lblDe = item.findViewById(R.id.*lblDe*) as TextView  val lblAsunto = item.findViewById(R.id.*lblAsunto*) as TextView  fun bindCorreo(correo: Correo){  lblDe.*text* = correo.de  lblAsunto.*text* = correo.asunto  }  }  override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): CorreosViewHolder {  val item = LayoutInflater.from(parent.*context*)  .inflate(R.layout.*listitem\_correo*, parent, false) as LinearLayout  return CorreosViewHolder(item)  }  override fun onBindViewHolder(holder: CorreosViewHolder, position: Int) {  val correo = datos[position]  holder.bindCorreo(correo)  holder.item.setOnClickListener**{**clickListener(correo)**}**;  }  override fun getItemCount() = datos.size } |

Y el diseño que queremos que tenga cada item del RecyclerView.

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  android:orientation="vertical"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content">  <TextView android:id="@+id/lblDe"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:textStyle="bold"  android:textSize="24sp" />  <TextView android:id="@+id/lblAsunto"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:textStyle="normal"  android:textSize="16sp" />  </LinearLayout> |

### Interfaz para capturar el evento del fragment

|  |
| --- |
| package com.example.probandofragments  interface CorreosListener {  fun onCorreoSeleccionado(correo: Correo) } |

### Últimos detalles

Faltaría asignar al activity\_main el Fragment por defecto que deseamos que se muestre.

|  |
| --- |
| *<?*xml version="1.0" encoding="utf-8"*?>* <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  android:id="@+id/llprincipal"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:orientation="vertical">   <androidx.fragment.app.FragmentContainerView  android:id="@+id/fragmentContainerView"  android:name="com.example.probandofragments.FragmentListado"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent" /> </LinearLayout> |

La clase principal quedaría de esta manera:

|  |
| --- |
| package com.example.probandofragments  import android.content.Intent import android.os.Bundle import androidx.activity.ComponentActivity import androidx.activity.compose.setContent import androidx.activity.enableEdgeToEdge import androidx.appcompat.app.AppCompatActivity import androidx.compose.foundation.layout.fillMaxSize import androidx.compose.foundation.layout.padding import androidx.compose.material3.Scaffold import androidx.compose.material3.Text import androidx.compose.runtime.Composable import androidx.compose.ui.Modifier import androidx.compose.ui.tooling.preview.Preview import androidx.core.app.PendingIntentCompat.getActivity import androidx.core.content.ContentProviderCompat.requireContext import androidx.fragment.app.Fragment import com.example.probandofragments.ui.theme.ProbandoFragmentsTheme import java.security.Provider  class MainActivity : AppCompatActivity() {  override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)   setContentView(R.layout.*activity\_main*)  *//enableEdgeToEdge()* val frgListado = *supportFragmentManager*.findFragmentById(R.id.*fragmentContainerView*) as FragmentListado  frgListado?.setCorreosListener **{** val frgDetalle: Fragment? = *supportFragmentManager*.findFragmentById(R.id.*fragmentContainerView2*) as? FragmentDetalle  if (frgDetalle != null)  (frgDetalle as FragmentDetalle).mostrarDetalle(**it**.texto)  else {   val i = Intent(this@MainActivity, DetalleActivity::class.*java*)  i.putExtra("DETALLE", **it**.texto)  startActivity(i)  }  **}** } } |

En cuanto al diseño de la actividad de detalle:

|  |
| --- |
| *<?*xml version="1.0" encoding="utf-8"*?>* <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  android:orientation="vertical"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent">   <androidx.fragment.app.FragmentContainerView  android:id="@+id/fragmentContainerView2"  android:name="com.example.probandofragments.FragmentDetalle"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content" /> </LinearLayout> |

Y su clase asociada:

|  |
| --- |
| package com.example.probandofragments  import android.content.Intent import android.os.Bundle import android.util.Log import androidx.appcompat.app.AppCompatActivity import androidx.fragment.app.Fragment  class DetalleActivity : AppCompatActivity() {   override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  setContentView(R.layout.*activity\_detalle*)  *//val detalle = supportFragmentManager.findFragmentById(R.id.fragmentContainerView2) as FragmentDetalle  //intent.getStringExtra("DETALLE")?.let { (detalle as FragmentDetalle).mostrarDetalle(it) }* val detalle = *supportFragmentManager*.findFragmentById(R.id.*fragmentContainerView2*) as? FragmentDetalle    if (detalle != null) {  *//(detalle as FragmentDetalle)?.mostrarDetalle(intent.getStringExtra("DETALLE").toString())* val datos = *intent*.getStringExtra("DETALLE").*orEmpty*()  detalle.mostrarDetalle(datos)  } else {  Log.e("DetalleActivity", "El fragmento FragmentDetalle no se encontró.")  }   *//val detalle: Fragment? = supportFragmentManager.findFragmentById(R.id.fragmentContainerView2) as FragmentDetalle  //if (detalle != null)  // (detalle as FragmentDetalle).mostrarDetalle(intent.getStringExtra("DETALLE").toString())* } } |

### Adaptando el diseño a distintos tamaños de pantalla

Si queremos que según la resolución de pantalla se muestre un diseño u otro se lo tenemos que indicar a Android Studio de esta manera:

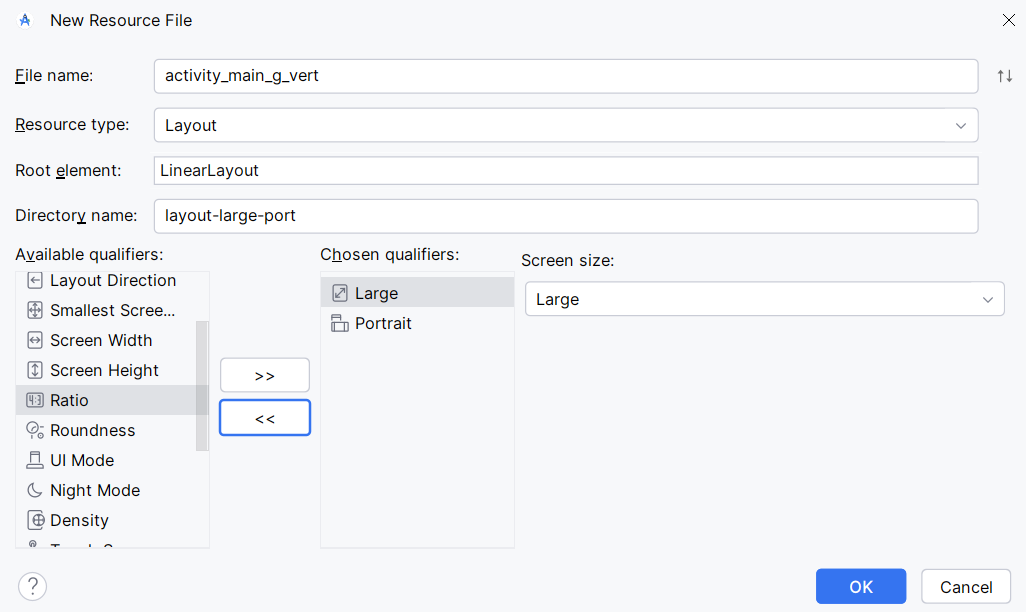
* Pulsa con el botón derecho sobre la carpeta “res” y selecciona “new – android resource file”.

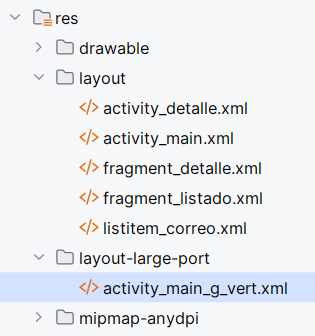
File name: tiene que ser el mismo que el homónimo en pantalla normal.

Resource type: layout

Qualifiers seleccionamos “size” y “orientation”

Primero vamos a generar el diseño para una pantalla grande en vertical.





Se nos genera el diseño deseado (.xml). Lo configuraremos con los fragments que consideremos:

|  |
| --- |
|  |

Probamos que funcione.

### Eventos

Vamos a explicar cómo hemos implementado la lógica de pinchar en un elemento del listado.

Empezaremos implementando el evento de click del listado, en la definición de su adaptador, dentro del método **onViewCreated**() de la clase **FragmentListado**. Lo que hagamos al capturar este evento dependerá de si en la pantalla se está viendo el fragment de detalle o no:

1. Si existe el fragment de detalle habría que obtener una referencia a él y llamar a su método mostrarDetalle() con el texto del correo seleccionado.
2. En caso contrario, tendríamos que navegar a la actividad secundaria DetalleActivity para mostrar el detalle.

Sin embargo existe un problema, **un fragment no tiene por qué conocer la existencia de ningún otro**, es más, deberían diseñarse de tal forma que fueran lo más independientes posible, de forma que puedan reutilizarse en distintas situaciones sin problemas de dependencias con otros elementos de la interfaz.

Por este motivo, el patrón utilizado normalmente en estas circunstancias no será tratar el evento en el propio fragment, sino definir y lanzar un evento personalizado al pulsarse el item de la lista y **delegar a la actividad contenedora la lógica del evento**, ya que ella sí debe conocer qué fragments componen su interfaz.

¿Cómo hacemos esto?

**Primero definimos una interfaz** con el método asociado al evento, en este caso llamada CorreosListener con un único método llamado onCorreoSeleccionado()

Y posteriormente **declaramos un atributo de la clase** con esta interfaz y definimos un método setXXXListener() para poder asignar el evento desde fuera de la clase.

## Actividad

Modifica el ejemplo anterior de forma que, si cambiamos la orientación a “landscape”, en el diseño principal se muestren los dos fragments.

Haz pruebas de funcionamiento.