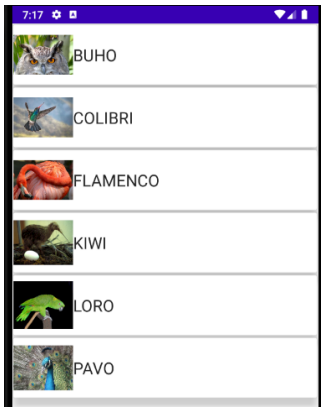
Bloque 8 . Ejercicio Resuelto Pajaros

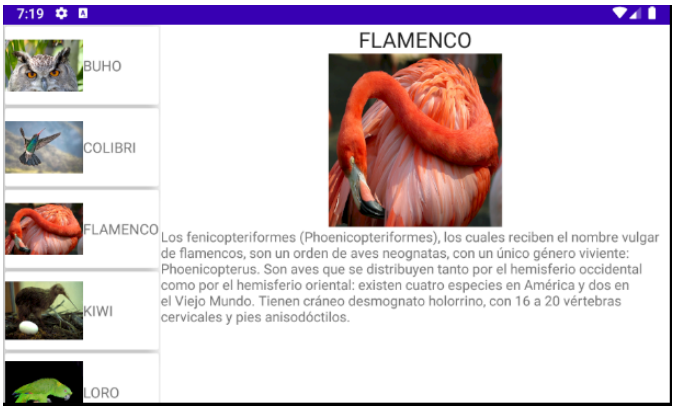
Descargar este ejercicio

Vamos a crear un ejemplo práctico de una lista en la cual, al seleccionar un elemento se muestra el detalle del mismo. La actividad principal MainActivity, usará diferentes layouts para los modos portrait y landscape. Las imágenes y los datos se pasan como recurso, también la clase **Datos**.

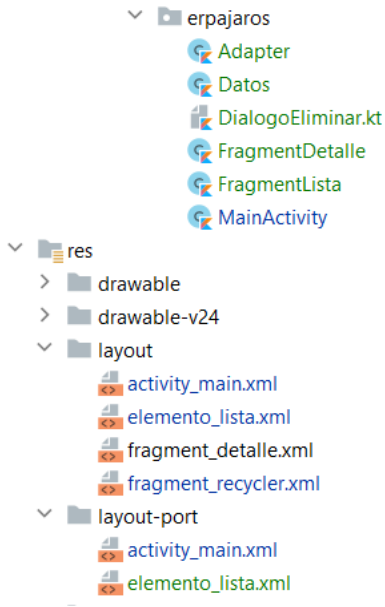
- **Portrair**, MainActivity mostrará un Fragment con un recycler y al seleccionar un elemento de este, se cargará otro Fragment que mostrará el detalle del elemento seleccionado, como se ve en las imagenes:



- **Landscape**, La activity principal cargará dos Fragments, el de la lista y el de detalle. Mostrando el detalle del elemento seleccionado de la lista.



Empezaremos haciendo una descripción de las clases y archivos xml que vamos a ir desarrollando.



Vamos primero a construir el código de las vistas necesarias para los dos fragments:

fragment_detalle.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:gravity="center|top"
    android:orientation="vertical" >
    <TextView
        android:id="@+id/textView_superior"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Large Text"
        android:textAppearance="?android:attr/textAppearanceLarge" />
    <ImageView
        android:id="@+id/imageView_imagen"
        android:layout_width="178dp"
        android:layout_height="178dp"
        android:scaleType="fitXY"
        android:src="@android:drawable/ic_menu_gallery" />
    <TextView
        android:id="@+id/textView_inferior"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Small Text"
        android:textAppearance="?android:attr/textAppearanceSmall" />
</LinearLayout>
```

fragment_recycler.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.recyclerview.widget.RecyclerView xmlns:android="http://schemas.android.com/apk/res/
    android:orientation="vertical" android:layout_width="match_parent"
    android:id="@+id/recycler"
    android:layout_height="match_parent">
</androidx.recyclerview.widget.RecyclerView>
```

elemento_lista.xml (port)

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="2dp"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    app:cardElevation="10dp"
    app:cardCornerRadius="2dp"
    android:id="@+id/cardView">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="left|center"
        android:orientation="horizontal">
    <ImageView
        android:id="@+id/imageView_imagen_miniatura"
        android:layout_width="80dp"
        android:layout_height="80dp"
        android:src="@android:drawable/ic_menu_gallery" />
    <TextView
        android:id="@+id/textView_titulo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Large Text"
        android:textSize="15dp" />
    </LinearLayout>
</androidx.cardview.widget.CardView>
```

Como se puede ver en la imagen de la estructura de las carpetas, nos encontramos con **dos activity_main.xml** y dos **elemento_lista.xml", esto es debido a la diferenciación que le queremos dar si se cambia de vista horizontal a vertical. Por este motivo tendrémos que crear una activity_main con un contender para el fragment (en este único contenedor se intercambiarán ambos fragments) y otra con dos (lista y detalle).

elemento_lista.xml (land)

En este caso solo se diferencia por el tamaño de el texto, aunque podrían haber más elementos distintos.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="2dp"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    app:cardElevation="10dp"
    app:cardCornerRadius="2dp"
    android:id="@+id/cardView">
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="left|center"
        android:orientation="horizontal">
    <ImageView
        android:id="@+id/imageView_imagen_miniatura"
        android:layout_width="80dp"
        android:layout_height="80dp"
        android:src="@android:drawable/ic_menu_gallery" />
    <TextView
        android:id="@+id/textView_titulo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Large Text"
        android:textAppearance="?android:attr/textAppearanceLarge" />
    </LinearLayout>
</androidx.cardview.widget.CardView>
```

main_activity.xml (port)

Con un solo contenedor para los fragments

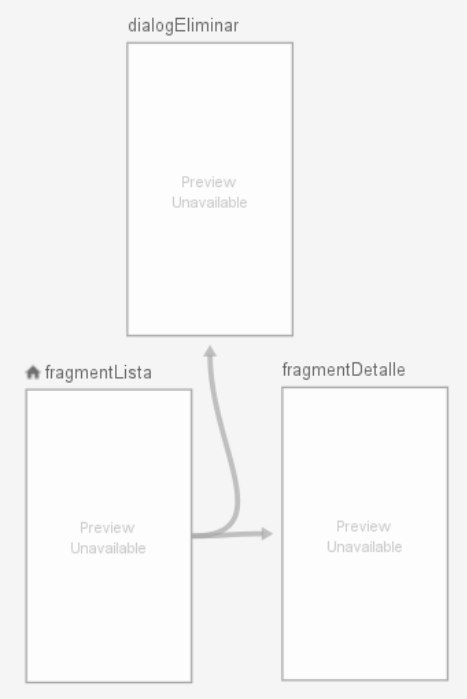
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:orientation="horizontal"
    tools:context=".MainActivity">
    <androidx.fragment.app.FragmentContainerView
        android:layout_weight="0.3"
        android:name="androidx.navigation.fragment.NavHostFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:defaultNavHost="true"
        app:navGraph="@navigation/navigation"
        android:id="@+id/contenedor"/>
</LinearLayout>
```

main_activity.xml (land)

Con un contenedor estático para el detalle y otro de navegación para la lista.

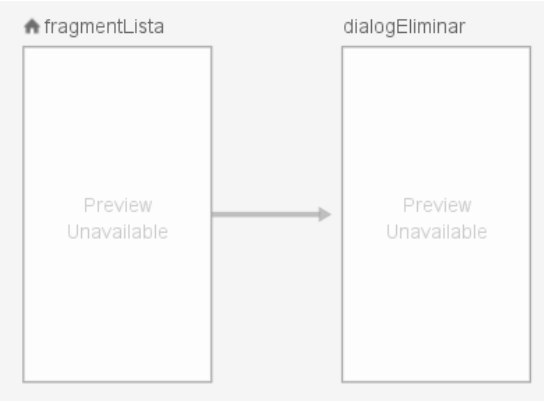
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:orientation="horizontal"
    tools:context=".MainActivity">

    <androidx.fragment.app.FragmentContainerView
        android:id="@+id/lista"
        android:name="androidx.navigation.fragment.NavHostFragment"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        app:defaultNavHost="true"
        app:navGraph="@navigation/navigation_land_lista" />

    <androidx.fragment.app.FragmentContainerView
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:name="com.ejemplos.b8.erpajaros.FragmentDetalle"
        android:id="@+id/detalle"/>
</LinearLayout>
```

Para este ejercicio vamos a usar dos elementos `navigation`. Para el caso de vista **vertical** utilizaremos un gráfo de navegación como el siguiente:



Como se puede ver en la imagen, también tendremos que crear un `FragmentDialog`, para que se muestre un dialogo al realizar una pulsación larga sobre algún elemento de la lista para ser eliminado.

Y para la vista horizontal usaremos otro, solo con la parte de navegación entre fragmentLista y el dialogEliminar. La navegación del elemento Detalle, se ha realizado usando el `FragmentManager` y `FragmentTransaction`



Para poder hacer los gráficos de navegación, antes debemos crear el código `Kotlin` necesario. Vamos a pasar a ello, junto con sus comentarios por líneas.

En cuanto al código del adaptador y del holder del recycler, será idéntico de como lo hacemos normalmente, en el que implementamos los listeners para el Cick corto y largo:

Adapter.kt

```
class Adapter(var datos: ArrayList<Datos>) :
                                RecyclerView.Adapter<Adapter.Holder>(),
                                View.OnClickListener, OnLongClickListener
{
    var listener: View.OnClickListener? = null
    var listenerLong: OnLongClickListener? = null
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
                                Holder {
        val view: View
        view = LayoutInflater.from(parent.context)
                        .inflate(R.layout.elemento_lista, parent, false)
        val holder = Holder(view)
        view.setOnClickListener(this)
        view.setOnLongClickListener(this)
        return holder
    }
    override fun onBindViewHolder(holder: Holder, position: Int) {
        (holder as Holder?)!!.bind(datos[position])
    }
    override fun getItemCount(): Int {
        return datos.size
    }
    fun miOnClick(listener: View.OnClickListener?) {
        this.listener = listener
    }
    override fun onClick(v: View) {
        if (listener != null) listener!!.onClick(v)
    }
    fun miOnLongClick(listener: OnLongClickListener?) {
        listenerLong = listener
    }
    override fun onLongClick(v: View): Boolean {
        if (listenerLong != null) listenerLong!!.onLongClick(v)
        return false
    }

    class Holder(itemView: View) : RecyclerView.ViewHolder(itemView) {
        var lineTexto: TextView
        var imagen: ImageView
        var cardView: CardView
        fun bind(dato: Datos) {
            lineTexto.text = dato.nombre
            imagen.setImageResource(dato.icono)
        }
        init {
            lineTexto = itemView.findViewById(R.id.textView_titulo)
            imagen = itemView.findViewById(R.id.imageView_imagen_miniatura)
            cardView = itemView.findViewById(R.id.cardView)
        }
    }
}
```

La clase que implementa el detalle al pulsar sobre un elemento de la lista, quedaría con el siguiente código:

FragmentDetalle.kt

```
class FragmentDetalle : Fragment() {
    var mItem: Int? = null
    override fun onCreate(savedInstanceState: Bundle?) {

        super.onCreate(savedInstanceState)
6       if(savedInstanceState!=null) mItem=savedInstanceState?.getInt("POSICION")
        else {
            if (arguments != null) {
9               mItem = arguments?.getInt("POSICION")
10          } else mItem = 0
        }
    }
     override  fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        val rootView: View = inflater.inflate(R.layout.fragment_detalle,
                                              container, false)
        //Mostramos el contenido al usuario
        if(MainActivity.datos!!.size > 0) {
            val elemento = MainActivity.datos?.get(mItem!!)
            (rootView.findViewById<TextView>(R.id.textView_superior)).text =
                                                    elemento?.nombre
            (rootView.findViewById<TextView>(R.id.textView_inferior)).text =
                                                    elemento?.desc
            (rootView.findViewById<ImageView>(R.id.imageView_imagen)).
                                          setImageResource(elemento!!.icono)
        }
        return rootView
    }
32  override fun onSaveInstanceState(outState: Bundle) {
        super.onSaveInstanceState(outState)
        outState.putInt("POSICION", mItem!!)
    }
}
```

📌 las líneas a destacar de este código son la creadas para guardar la posición pulsada en determinado momento, para el caso de que ocurra algún giro de pantalla no perder el elemento pulsado. Guardaremos la posición mediante el método `onSaveInstanceState` por el que se pasará justo antes de que la aplicación se destruya, **Línea 32**. Esta posición se podrá recuperar mediante el Bundle que entra en el método `onCreate` **Línea 6**. Las **Líneas 9 y 10**

> son para recuperar la posición cuando se pulsa el elemento de la lista en el `FragmentLista`, se recupera mediante la propiedad `arguments`

El código del fragment que contendrá el recycler, será una clase que derivará de Fragment y en la que inflaremos el layout recycler e implementaremos el OnClick, mandando la posición seleccionada mediante el Bundle a el Fragment Detalle:

FragmentLista.kt

```
class FragmentLista : Fragment() {
    var listenerLargo: View.OnLongClickListener? = null
    var recyclerView: RecyclerView? = null
    var valores: ArrayList<Datos>? = null
    var posicion:Int=0

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        super.onCreateView(inflater, container, savedInstanceState)
        val rootView: View = inflater.inflate(R.layout.fragment_recycler,
                                              container, false)
        recyclerView = rootView.findViewById(R.id.recycler)
        adapter = Adapter(MainActivity.datos!!)
        recyclerView?.adapter = adapter
        recyclerView?.setHasFixedSize(true)
        recyclerView?.layoutManager =LinearLayoutManager(getActivity(),
                                        LinearLayoutManager.VERTICAL, false)

        adapter!!.miOnClick { v ->
            val bundle = Bundle()
            bundle.putInt("POSICION", recyclerView!!.getChildAdapterPosition(v))
            val navController = NavHostFragment.findNavController(this)
            if (navController.currentDestination?.id == R.id.fragmentLista) {
                if (resources.configuration.orientation ==
                    Configuration.ORIENTATION_PORTRAIT) navController.navigate(
                    R.id.action_fragmentLista_to_fragmentDetalle,
                    bundle
                )
               else
                {
                    val fT = requireActivity().supportFragmentManager.
                                                            beginTransaction()
                    fT.replace(R.id.detalle,FragmentDetalle::class.java,bundle).
                                                            commit()
                }
            }
        }
        adapter!!.miOnLongClick { v ->
            val navController = NavHostFragment.findNavController(this)
            if (navController.currentDestination?.id == R.id.fragmentLista) {
                val bundle = Bundle()
                posicion=recyclerView!!.getChildLayoutPosition(v)
                bundle.putInt("POSICION",posicion)
                if (resources.configuration.orientation ==
                                                Configuration.ORIENTATION_PORTRAIT)
                 navController.navigate(R.id.action_fragmentLista_to_dialogEliminar,
                                        bundle)
                else {
```

```
                        val navHostFragment = requireActivity().
                        supportFragmentManager.findFragmentById(R.id.lista)
                        as NavHostFragment
                        val navControllerLista = navHostFragment.navController
                        navControllerLista.navigate(R.id.
                                        action_fragmentLista_to_dialogEliminar, bundle)
58                  }
                }
                true
            }

            return rootView
        }

         override fun onAttach(activity: Context) {
            super.onAttach(activity)
            try {
                valores = MainActivity.datos
                listenerLargo = activity as View.OnLongClickListener?
            } catch (e: ClassCastException) {  }
        }

        companion object {
            var adapter: Adapter? = null
        }
    }
```

📌Desde **Línea 24 a 38** al pulsar sobre un elemento de la lista se creará un Bundle para guardar la posición del elemento pulsado, dependiendo si el dispositivo está en posición horizontal o vertical: en el primer caso se navegará al fragment detalle utilizando la acción que habremos creado en el grafo de navegación, en el segundo utilizaremos la carga de fragment de forma tradicional. En el código que hay desde la **Línea 43 a 58**, también se tiene que tener en cuenta la posición del dispositivo, pero solo para usar uno u otro `NavHostFragment` , en todo caso navegaremos del elemento pulsado al dialogo que nos permite eliminar un elemento de la lista.

DialogEliminar.kt

```
class DialogEliminar : DialogFragment() {
    override  fun onCreateDialog(savedInstanceState: Bundle?): Dialog {
        val builder: AlertDialog.Builder = AlertDialog.Builder(getActivity())
        if (arguments != null) {
            val posicion = arguments?.getInt("POSICION")
            builder.setMessage("Deseas Eliminar el item?")
                .setTitle("AVISO")
                .setPositiveButton("OK", DialogInterface.OnClickListener {
                     dialog, id -> MainActivity.datos?.removeAt(posicion!!)
                    FragmentLista.adapter!!.notifyItemRemoved(posicion!!)})
        }
        return builder.create()
    }
}
```

> 📌Este dialogo fragment es igual a los que hemos visto en ejercicios anteriores, pasaremos la posición del elemento a eliminar mediante un bundle

En cuanto al código de la MainActivity, solo se encargará de rellenar los datos pasados como recurso y de guardar una instancia de estos datos en caso de que la aplicación pare de forma brusca, para que pueda ser recuperada al volver a iniciar.

```
class MainActivity : FragmentActivity(){
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        //Recuperar datos después giro
        if (savedInstanceState != null) {
            datos = savedInstanceState.getParcelableArrayList("DATOS")
        }
        else  datos = rellenarDatos()
    }

    override fun onSaveInstanceState(outState: Bundle) {
        super.onSaveInstanceState(outState)
        outState.putParcelableArrayList("DATOS", datos)
    }
    fun rellenarDatos(): ArrayList<Datos> {
        ...}

    companion object {
         var datos: ArrayList<Datos>? = null
    }
}
```