07. Poder realizar swipe en los ítems del RecyclerView.

Se quiere que el usuario pueda realizar un swipe (desplazar a la derecha) cada uno de los ítems del RecyclerView para que muestre a su izquierda una serie de opciones que podrán realizarse en dicho item.

Para ello en el layout correspondiente al item de tipo film (layout con identificador R.layout.film_item) hemos definido dos guide lines verticales: R.id.glStart y R.id.glEnd. Estos guide lines están ubicados respectivamente al 0% y 100% (coincidiendo de esta forma con el start y el end del parent)

```
<androidx.constraintlayout.widget.Guideline
    android:id="@+id/glStart"
    android:orientation="vertical"
    app:layout_constraintGuide_percent="0"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
```

```
<androidx.constraintlayout.widget.Guideline
    android:id="@+id/glEnd"
    android:orientation="vertical"
    app:layout_constraintGuide_percent="1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
```

El resto de widgets se ubican en el layout respecto a estos guide lines.

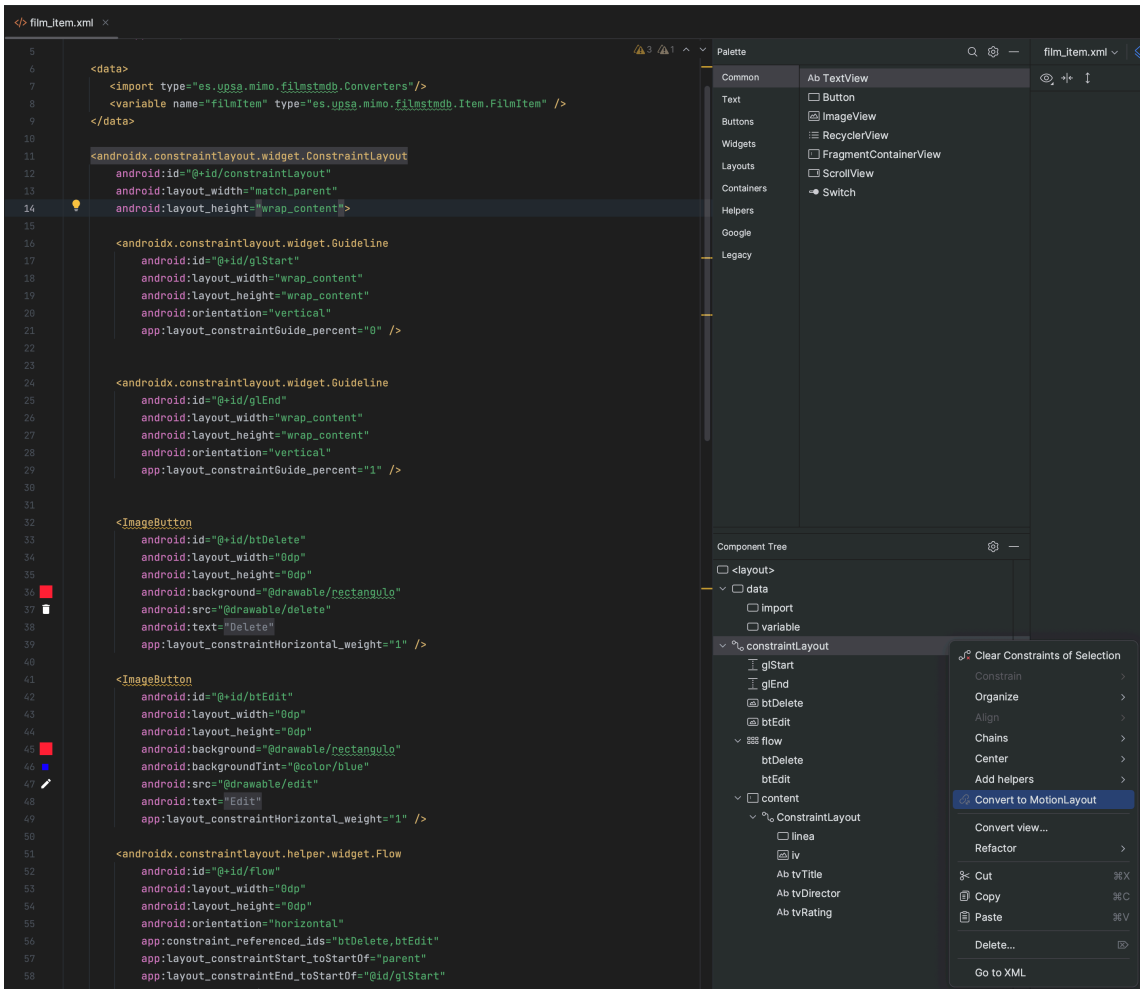
Por otra parte, en el layout también se incluye los botones que se quiere que aparezcan al realizar el swipe (R.id.btDelete y R.id.btEdit). Estos dos botones se incluirán dentro de un androidx.constraintlayout.helper.widget.Flow con orientación horizontal. Este Flow se ubicará dentro del layout entre el start del parent y el guide line R.id.glStart

```
<ImageButton
  android:id="@+id/btDelete"
  app:layout_constraintHorizontal_weight="1"
  android:src="@drawable/delete"
  android:background="@drawable/rectangulo"
  android:layout_width="0dp"
  android:layout_height="0dp"/>
```

```
<ImageButton
  android:id="@+id/btEdit"
  app:layout_constraintHorizontal_weight="1"
  android:src="@drawable/edit"
  android:background="@drawable/rectangulo"
  android:backgroundTint="@color/blue"
  android:layout_width="0dp"
  android:layout_height="0dp"/>
```

```
<androidx.constraintlayout.helper.widget.Flow
  android:id="@+id/flow"
  android:orientation="horizontal"
  app:constraint_referenced_ids="btDelete,btEdit"
  app:layout_constraintStart_toStartOf="parent"
  app:layout_constraintEnd_toStartOf="@id/glStart"
  app:layout_constraintBottom_toBottomOf="parent"
  app:layout_constraintTop_toTopOf="parent"
  android:layout_width="0dp"
  android:layout_height="0dp"/>
```

A continuación, se convierte el ConstraintLayout en un MotionLayout. Un MotionLayout es una especialización de ConstraintLayout (por tanto también es un ConstraintLayout) que permite definir animaciones.



Al realizar la conversión, se genera automáticamente el recurso de tipo xml: film_item_xml_constraintlayout_scene.xml (al que genéricamente se le denomina «*motion scene*»). En este fichero se podrán definir «transiciones». Una transición implica que el MotionLayout pase de un estado «A» a otro «B» cuando suceda algo. El estado del MotionLayout viene determinado por el valor que tienen los atributos de los widgets descritos en él. En el fichero «*motion scene*» podemos definir varios estados para el Motionlayout. A cada estado le asignamos un identificador. En un estado (representado por un elemento `ConstraintSet`) se define por cada widget del MotionLayout los valores que se desea que tengan sus atributos.

En nuestro caso se han definido dos estados:

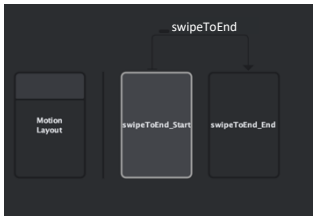
- R.id. swipeToEnd_Start: no altera el valor de ningún atributo de los widgets del MotionLayout

- R.id.swipeToEnd_End: se modifica dos widgets: los guidelines R.id.glStart y R.id.glEnd. Concretamente para R.id.glStart se modifica el atributo `layout_constraintGuide_percent con el valor 0.4 y para` R.id.glEnd se modifica su atributo

`layout_constraintGuide_percent` con el valor 1.4. A través de este estado se intenta representar cómo queremos que se muestre el Motionlayout una vez se haya realizado la operación swipe.

Por otra parte en el «*motion scene*» se ha definido la transición con identificador R.id.swipeToEnd. En esta transición se indica que el MotionLayout debe pasar del estado inicial identificado por `R.id.swipeToEnd_Start` al estado final identificado por `R.id.swipeToEnd_End`. Esta transición tendrá una duración de `500` milisegundos y se desencadenará cuando se detecte que el usuario se realiza un `swipe` (elemento `<OnSwipe>`) con dirección «`dragRight`» en el view `R.id.content` del Motionlayout.

```
<MotionScene
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:motion="http://schemas.android.com/apk/res-auto">

    <Transition android:id="@+id/swipeToEnd"
                motion:constraintSetStart="@id/swipeToEnd_Start"
                motion:constraintSetEnd="@+id/swipeToEnd_End"
                motion:duration="500">

        <OnSwipe motion:touchAnchorId="@+id/content"
                 motion:touchAnchorSide="start"
                 motion:dragDirection="dragRight"/>

    </Transition>

    <ConstraintSet android:id="@+id/swipeToEnd_Start">

    </ConstraintSet>

    <ConstraintSet android:id="@+id/swipeToEnd_End">
        <Constraint android:id="@+id/glStart">
            <Layout motion:layout_constraintGuide_percent="0.4"/>
        </Constraint>
        <Constraint android:id="@+id/glEnd">
            <Layout motion:layout_constraintGuide_percent="1.4"/>
        </Constraint>
    </ConstraintSet>
</MotionScene>
```



Cuando se detecte que el usuario realiza el gesto swipe con dirección dragRight en el view `R.id.content` del MotionLayout, iniciará una animación que durará 500 milisegundos que refleje la transición entre los estados indicados. Al estar desplazando los guidelines, se simula el deslizamiento del item y la aparición de los botones que representarán las acciones que se podrán realizar.

Por otra parte, se ha modificado el view holder FilmItemViewHolder (el encargado de contener la vista con la que mostrar una película). El cambio supone añadirle un nuevo método reset(). Lo que se pretende hacer en este método es iniciar una nueva animación en el MotionLayout que le devuelva desde su estado actual hacia el estado inicial de su transición activa. En concreto, si en el Motionlayout el usuario ha desencadenado la transición

descrita anteriormente (ha realizado es swipe) se quiere que el MotionLayout vuelva al estado inicial descrito en esa transición (es decir, volver al estado `R.id.swipeToEnd_Start`)

El motivo por el que se quiere que el view holder tenga este método es para devolver al MotionLayout a su estado inicial. Se tendrá que hacer esto en dos momentos:

- cuando se vaya a reutilizar el view holder para mostrar una nueva película
- cuando al hacer scroll, deje de estar visible la vista del view holder en pantalla.

Estos dos momentos están representados por los métodos onViewDetachedFromWindow() y onViewRecycled() del adapter. onViewDetachedFromWindow() se ejecutará cuando la vista deje de estar visible como consecuencia del scroll realizado por el usuario en el RecyclerView. Por su parte onViewRecycled() se jecuta cuando se va a reciclar un viewholder. En ambos casos, se quiere que el MotionLayout esté en su estado inicial.