Programación multimedia y dispositivos móviles

UT-2. Controles de entrada II

https://developer.android.com/guide/
https://www.sgoliver.net/blog/curso-de-programacion-android/indice-de-contenidos/

Interfaz de usuario en Android: Controles básicos (II)

En este apartado vamos a ver cómo utilizar otros dos tipos de controles básicos en muchas aplicaciones, los checkboxes y los radio buttons:

- **CheckBox**: Un control checkbox se suele utilizar para marcar o desmarcar opciones en una aplicación, y en Android está representado por la clase del mismo nombre, CheckBox. La forma de definirlo en nuestra interfaz y los métodos disponibles para manipularlos desde nuestro código son análogos a los ya comentados para el control ToggleButton.

  Además, podremos utilizar la propiedad "**checked**" para inicializar el estado del control a marcado (true) o desmarcado (false). Si no establecemos esta propiedad el control aparecerá por defecto en estado desmarcado.

Interfaz de usuario en Android: Controles básicos (II)

- **RadioButton:** al igual que los controles checkbox, un radio button puede estar marcado o desmarcado, pero en este caso suelen utilizarse dentro de un grupo de opciones donde una, y sólo una, de ellas debe estar marcada obligatoriamente, es decir, que si se marca una de las opciones se desmarcará automáticamente la que estuviera activa anteriormente. En Android, un grupo de botones radio button se define mediante un elemento "**RadioGroup"**, que a su vez contendrá todos los elementos RadioButton necesarios.

  Una vez definida la interfaz podremos manipular el control desde nuestro código bien haciendo uso de los diferentes métodos del control RadioGroup bien con los métodos del propio RadioButton.

Interfaz de usuario en Android: Controles básicos (II)

Si decides trabajar con el RadioGroup algunas propiedades o métodos que necesitaras: **check**(id) para marcar una opción determinada mediante su ID, **clearCheck** para desmarcar todas las opciones, y **checkedRadioButtonId** que como su nombre indica devolverá el ID de la opción marcada (o el valor -1 si no hay ninguna marcada).

Para trabajar con los métodos del RadioButton, los métodos son similares a los del Toogle Button, Switch o CheckBox

https://developer.android.com/guide/topics/ui/controls/radiobutton

https://developer.android.com/guide/topics/ui/controls/checkbox?hl=es-419

Interfaz de usuario en Android: Controles básicos (II)

- **ImageView**: El control ImageView permite mostrar imágenes en la aplicación. La propiedad más interesante es "**srcCompat**", que permite indicar la imagen a mostrar. Nuevamente, lo normal será indicar como origen de la imagen el identificador de un recurso de nuestra carpeta /res/drawable. Además de esta propiedad, existen algunas otras útiles en algunas ocasiones como las destinadas a establecer el tamaño máximo que puede ocupar la imagen, "**maxWidth**" y "**maxHeight**", o para indicar cómo debe adaptarse la imagen al tamaño del control, "**scaleType"** (CENTER, CENTER_CROP, CENTER_INSIDE, ...). Además, como ya comentamos para el caso de los controles ImageButton, al tratarse de un control de tipo imagen deberíamos establecer siempre la propiedad "**contentDescription**" para ofrecer una breve descripción textual de la imagen, algo que hará nuestra aplicación mucho más accesible.

Interfaz de usuario en Android: Controles básicos (II)

De forma general puedes cambiar los atributos de un control bien accediendo a ellos o bien con determinados métodos “setXXX” que nos proporciona el SDK de Android (depende un poco del atributo).

Si en vez de establecer la imagen a mostrar en el propio layout XML de la actividad quisiéramos establecerla mediante código utilizaríamos el método “**setImageResorce(...)**”, pasándole el ID del recurso a utilizar como contenido de la imagen:

.
```
miImageView.setImageResource(R.drawable.programador)
```

En cuanto a posibles eventos, al igual que comentamos para los controles de tipo botón en el apartado anterior, para los componentes ImageView también podríamos implementar su evento **onClick**, de forma idéntica a la que ya vimos, aunque en estos casos suele ser menos frecuente la necesidad de capturar este evento.

Incorporar recursos a nuestro proyecto

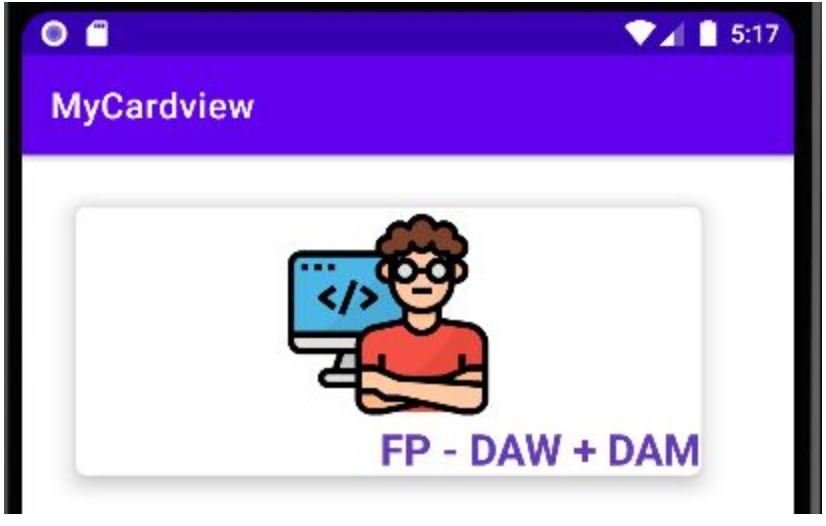
Simplemente hay que abrir el ResourceManager (bien en la pestaña derecha debajo de proyectos o bien desde la ruta de menú: **View > Tool Windows > Resource Manager**)

Dentro del ResourceManager pulsas el botón de (+) e importar los archivos (que previamente debes tener)

https://developer.android.com/studio/write/resource-manager?hl=es

Interfaces de usuario: CardView

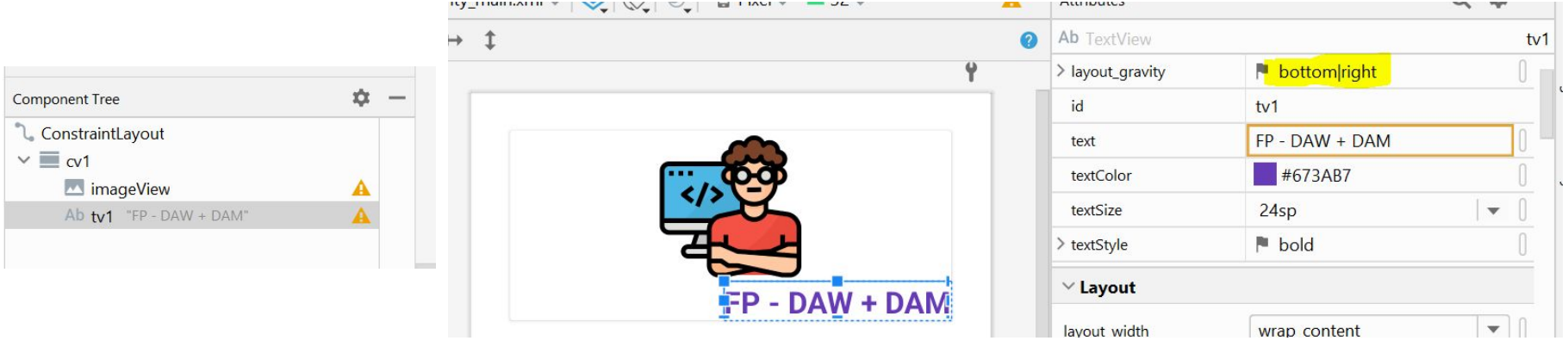
El componente llamado CardView es la implementación que nos proporciona Google del elemento visual en forma de tarjetas de información que tanto utiliza en muchas de sus aplicaciones.



Interfaces de usuario: CardView

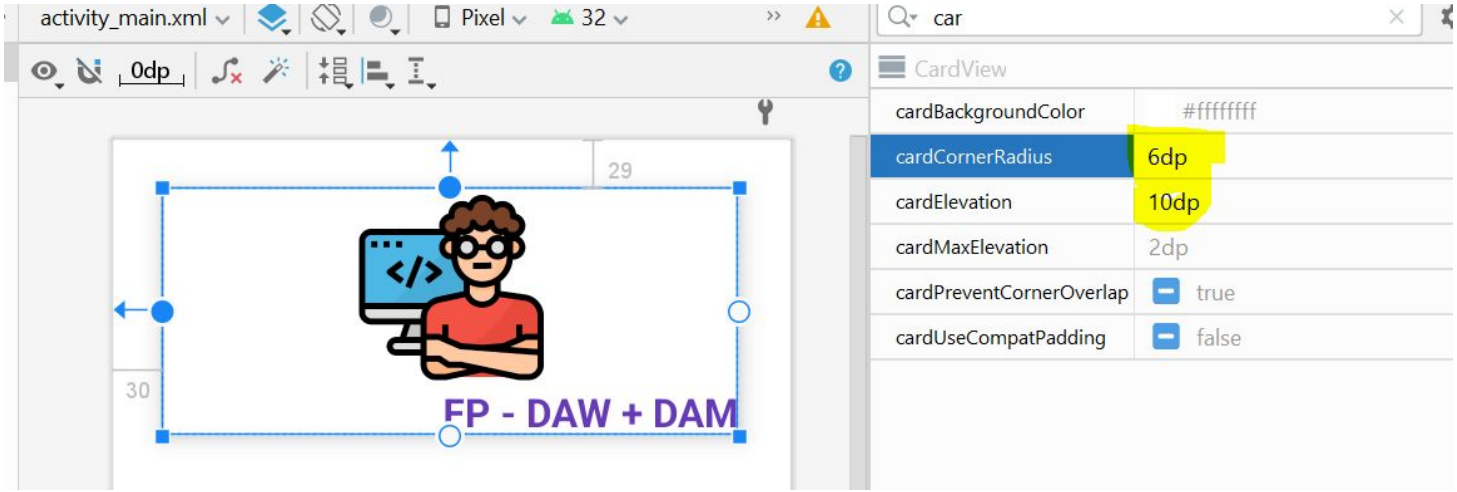
Al extender FrameLayout, cada componente que añadimos al CardView se nos coloca arriba y a la izquierda. De hecho si añadimos más de un componente estos se apilan arriba y a la izquierda.

Para colocar los componentes en este contenedor (y esto también es válido para el FrameLayout) podemos utilizar la propiedad Layout_gravity de cada componente indicando dónde debemos colocarlos.



Interfaces de usuario: CardView

Para definir la elevación de la tarjeta y el radio de las esquinas redondeadas, utilizando las propiedades cardElevation y cardCornerRadius respectivamente.



Interfaces de usuario: CardView

No hay mucho más que hacer con este control. CardView es un contenedor que nos ofrece por defecto bordes redondeados y/o elevaciones. Si no buscamos esto, no tiene mucho sentido utilizarlo.

Muchas veces su uso está asociado al RecyclerView ya que son componentes que encajan muy bien juntos.

https://developer.android.com/guide/topics/ui/layout/cardview?hl=es#groovy

Un enlace donde explica para qué sirven los atributos más importantes de una CardView

https://programmerclick.com/article/54771093788/

Interfaces de usuario: ScrollView

El ScrollView en Android te permite añadir controles en tu actividad ocupando más espacio del que está disponible. Este componente se encarga de desplazar su contenido a lo largo de la pantalla.

Se pueden añadir scroll vertical, scroll horizontal y también hacer anidación de scrolls. El usuario hará scroll a través de un gesto de desplazamiento vertical u horizontal para revelar la información limitada por el tamaño del contenedor.

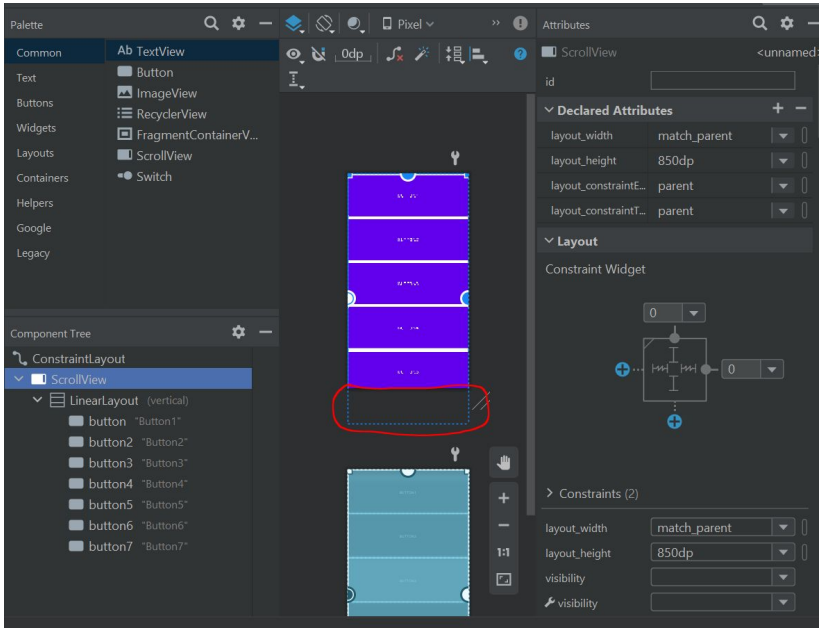
No es conveniente añadir un ScrollView a un RecyclerView. Hacerlo da como resultado un rendimiento deficiente de la interfaz de usuario y una experiencia de usuario pobre.

ScrollView soporta un solo hijo como contexto de desplazamiento, por lo que la estructura general de tu diseño debe encontrarse en él. Android añade por defecto un LinearLayout en la dirección del Scroll (vertical u horizontal). Puedes sustituirlo por otro Layout donde anidar contenido, pero debe haber un Layout.

Interfaces de usuario: ScrollView

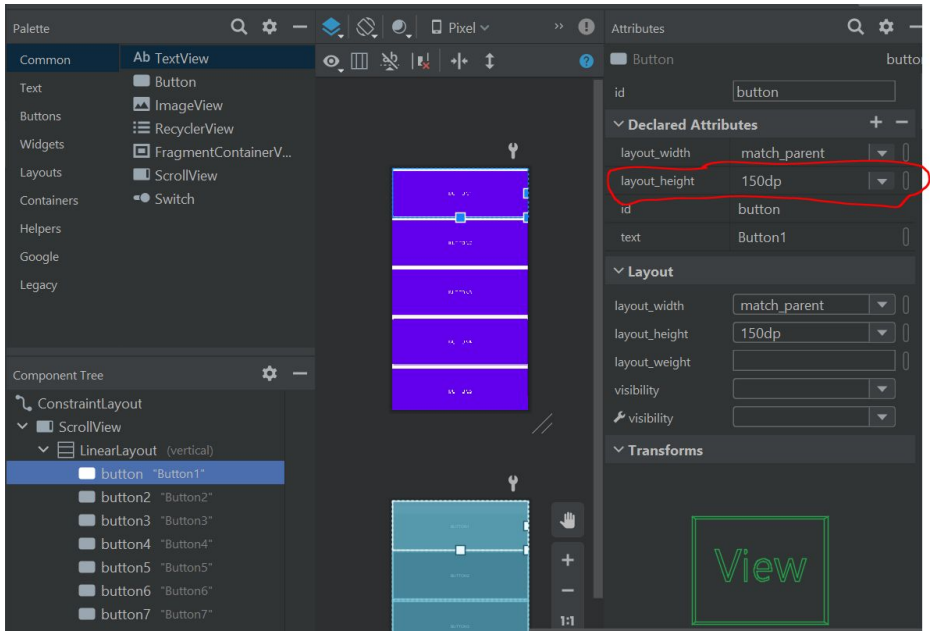
Mi pantalla es de algo más de 700dp[1] . He añadido un SchrollView de 850dp y se añade por debajo del Activity un recuadro en la parte en la que puedo incluir objetos.

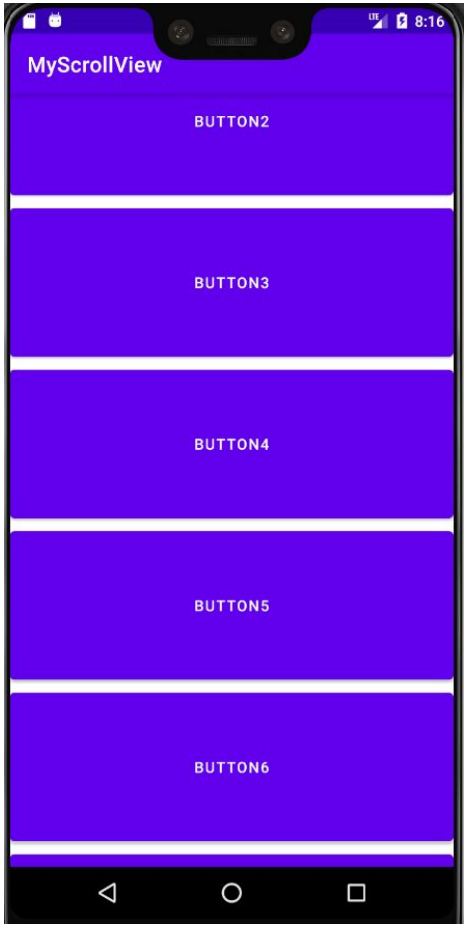
He añadido 7 botones de 150dp. En teoría el 6º no cabe completo y el 7º no cabe nada, pero a la hora de mostrarlo por pantalla no es exactamente así (quizás por la traducción de dp a pixeles).



(1) Un dp es una unidad de píxel virtual que prácticamente equivale a un píxel en una pantalla de densidad media (160 dpi; la densidad "de referencia"). Android traduce este valor a la cantidad apropiada de píxeles reales para cada densidad.

Interfaces de usuario: ScrollView



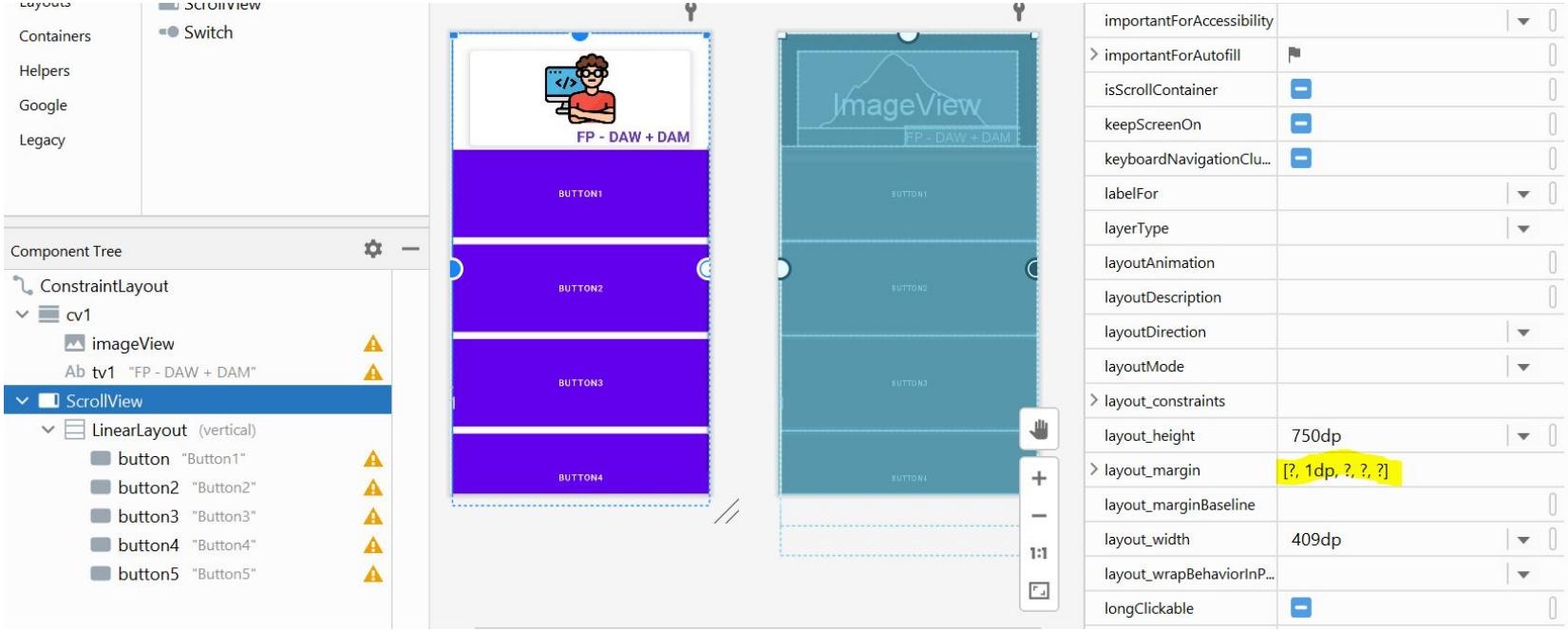


Interfaces de usuario: ScrollView

También es posible que el ScrollView no ocupe toda la pantalla, sino tan solo una parte. En mi caso he vuelto sobre el ejemplo del CardView y debajo de la tarjeta he metido un ScrollView donde ver diferentes jugadores. Cosas a tener en cuenta.

En este caso el ScrollView no funciona si para fijar su posición estamos usando la propiedad android:layout_marginTop, sino que en su lugar hay que configurar (directamente en el xml) la propiedad android:paddingTop.

Interfaces de usuario: ScrollView



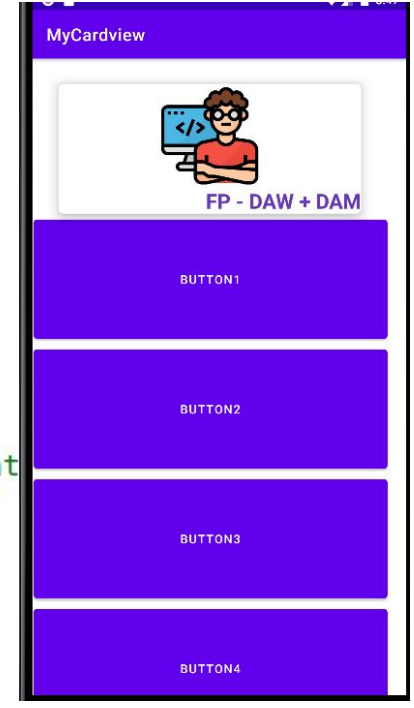
Interfaces de usuario: ScrollView

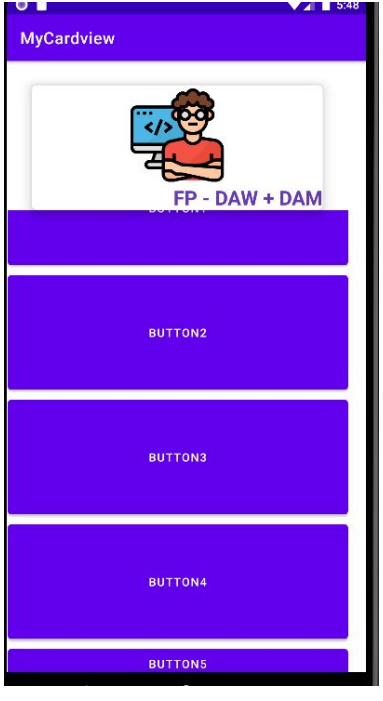
```
        android:textStyle="bold" />

</androidx.cardview.widget.CardView>

<ScrollView
    android:layout_width="409dp"
    android:layout_height="750dp"
    android:layout_marginStart="1dp"
    android:paddingTop="180dp"
    app:layout_constraintStart_toStartOf="parent
    app:layout_constraintTop_toTopOf="parent">

    <LinearLayout
        android:layout_width="match_parent"
```





Interfaces de usuario: ScrollView

El ScrollView horizontal funciona de forma análoga al vertical.

Es importante destacar que siempre que definimos un ScrollView, el Layout que ponemos dentro (con el que definimos el contexto de desplazamiento) debe tener la altura o la anchura (según el caso) = match_parent.

Si es un ScrollView vertical, será la altura del Layout la que deba ser igual a la del ScrollView (match_parent) para no limitarlo.

Si es un ScrollView horizontal, será la anchura del Layout la que deba ser igual a la del ScrollView (match_parent) para no limitarlo.