



Ejercicio: ConstraintLayout

A la hora de crear el diseño para nuestras actividades es de gran importancia tener en claro que son los **Layouts en Android**.

Un **Layout** es un elemento que representa el diseño de la interfaz de usuario de componentes gráficos como una actividad o fragmento. Un layout se encarga de actuar como contenedor de views (componentes) para establecer un orden visual, que facilite la comunicación del usuario con la interfaz de la aplicación.

La idea principal de este contenedor es evitar el uso de **layouts anidados** y así mejorar el performance al pintar vistas complejas.

Debemos tener en cuenta, que un *constraint* es una relación de un objeto a otro, de una manera restrictiva o de referencia, que nos ayuda a posicionar los elementos en relación o restricción de otros.

Por tanto, ConstraintLayout es un diseño que te permite colocar elementos en relación con otros elementos en la pantalla. Se utiliza en diseños "responsive" (adaptables), esto significa hacer que una app se adaptable a todos los dispositivos de diferentes tamaños: tabletas, smartphones, etc.

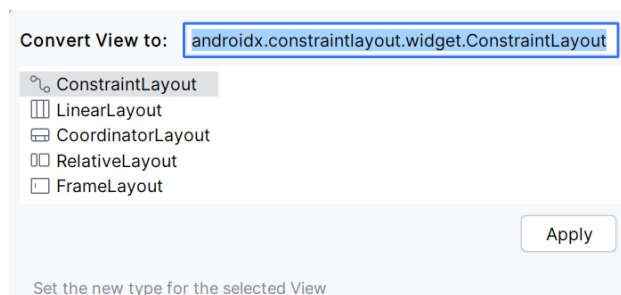
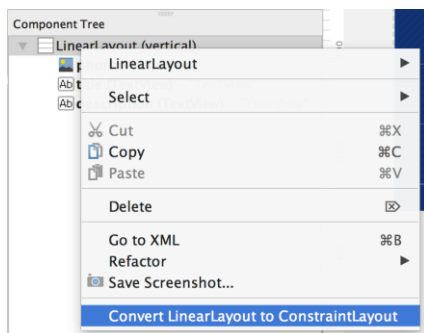
Es similar a RelativeLayout (legacy), aunque más flexible.

Cuando creas un proyecto con la opción "Empty Views Activity" utiliza ConstraintLayout.

Si por contrario ya tienes una pantalla que hace uso de algún otro layout, podemos fácilmente migrar los elementos a *ConstraintLayout*, ya que Android Studio tiene una herramienta para esto en su editor visual.

Para convertir un diseño existente en uno de ConstraintLayout, sigue estos pasos:

1. Abre tu diseño en Android Studio y haz clic en la pestaña **Design**
2. En la ventana **Component Tree**, haz clic con el botón derecho en el layout y, luego, en **Convierte XXXXLayout en ConstraintLayout**, o bien usa la opción "Convert View..."



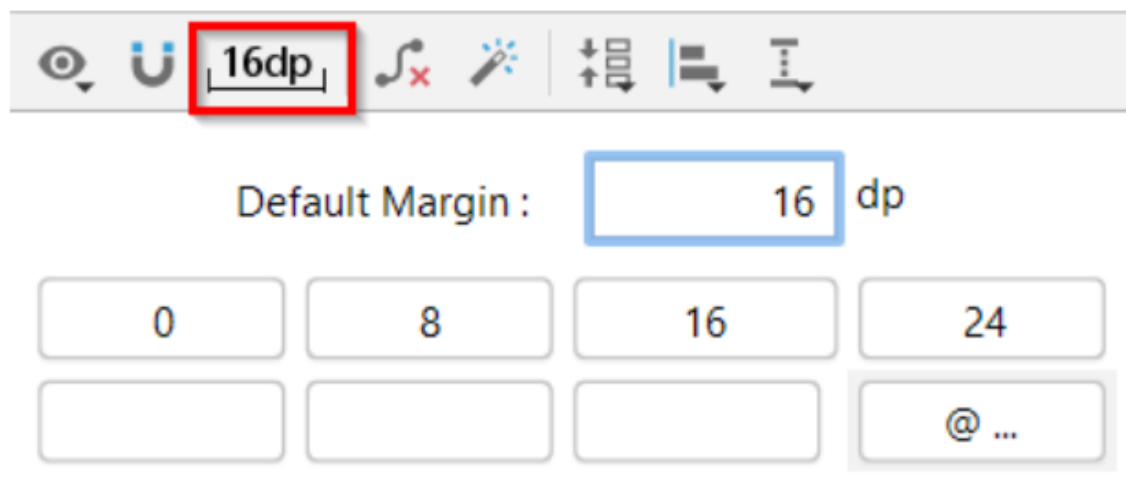
Para definir la posición de un componente en `ConstraintLayout`, hay que agregar al menos una *constraint* horizontal y una vertical en los atributos `Constraint widget`, si no lo hacemos, veremos como el editor indica que faltan constraints como un error en la barra de herramientas.

Cada *constraint* representa una conexión o alineación con otro componente, el nivel superior de la pantalla o una guía horizontal o vertical invisible.

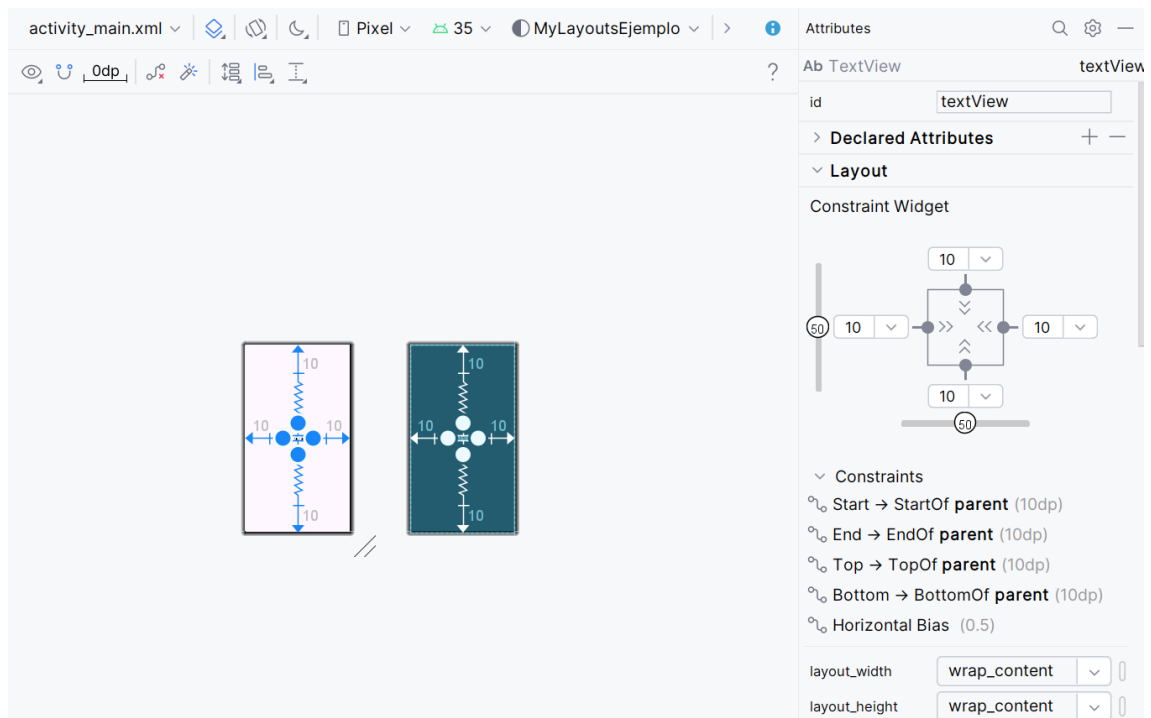
Los valores utilizados en `layout_width` / `layout_height` son:

- Fija: Puedes especificar una dimensión específica
- `wrap_content`: El componente se expande solo lo necesario para ajustarse a su contenido
- `match_constraint`: El componente se expande tanto como sea posible para cumplir con restricciones a cada lado, después de considerar los márgenes (se ajusta al ancho del contenedor).

Es conveniente, establecer al principio el margen por defecto. Especifica **Default Margins** en la toolbar.



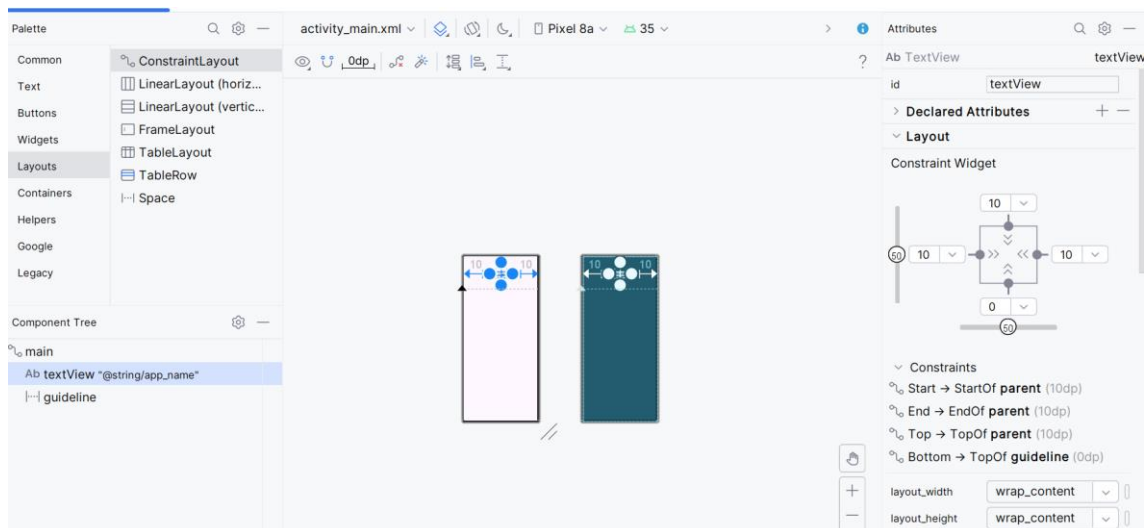
Ejemplo: A continuación, se muestra un ejemplo, en el que el TextView está alineado con el nivel superior e inferior de la pantalla, así como a la izquierda y a la derecha:



A continuación, se muestra un ejemplo, en el que el TextView está alineado con una guía horizontal invisible a los usuarios de tu app:

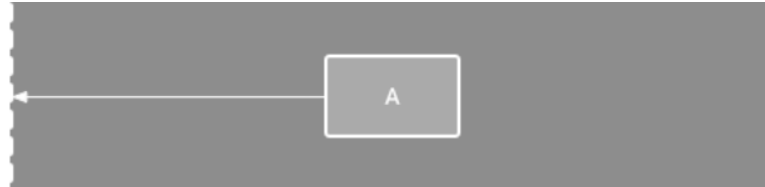
Para crear una línea guía, haz clic en **Guidelines**.  en la barra de herramientas y haz clic en **Add Vertical Guideline** o **Add Guideline horizontal**.

Arrastra la línea punteada para reubicar la guía y después conecta el componente con la guía.

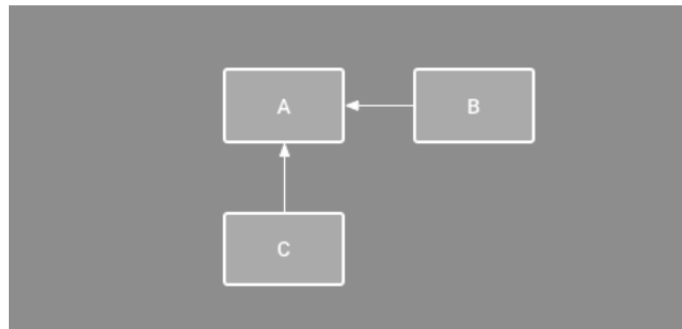


De esta manera, podemos ir colocando los distintos componentes que van a formar parte de mi pantalla. Podemos usar constraints para lograr diferentes tipos de diseños, a continuación, se muestran algunos de ellos:


-. **Restringe el lado de un componente al borde correspondiente de la pantalla:** En la imagen de abajo vemos como el lado izquierdo del componente está conectado al borde izquierdo de la pantalla. Puedes definir la distancia desde el borde.



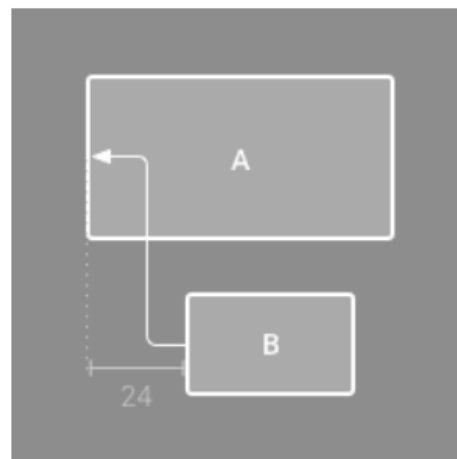
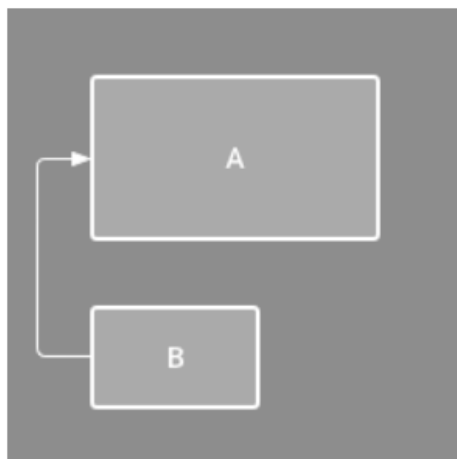
-. **Define la posición de un componente con respecto a otro, ya sea de forma vertical o horizontalmente:** En la imagen de abajo vemos como el componente B está restringido para estar siempre a la derecha de A, y el C está restringido debajo de A. Sin embargo, estas restricciones no implican alineación, por lo que B puede moverse hacia arriba y hacia abajo.



-. **Alinea el borde de un componente con el mismo borde de otro componente:** En la imagen de la izquierda, el lado izquierdo de B está alineado con el lado izquierdo de A.

Para alinear los componentes, selecciona todos los componentes que deseas alinear y, luego, haz clic en **Align**  en la barra de herramientas para seleccionar el tipo de alineación.

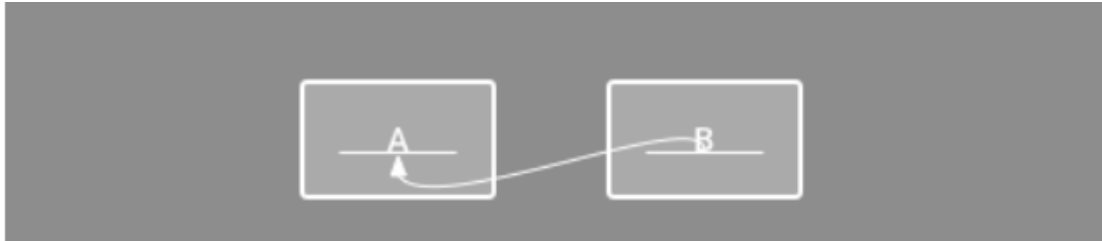
Puedes desplazar la alineación arrastrando el componente desde la restricción hacia adentro. Por ejemplo, en la imagen de la derecha, se muestra B con alineación con desplazamiento de 24 dp. El desplazamiento se define en el margen de Constraint widget.



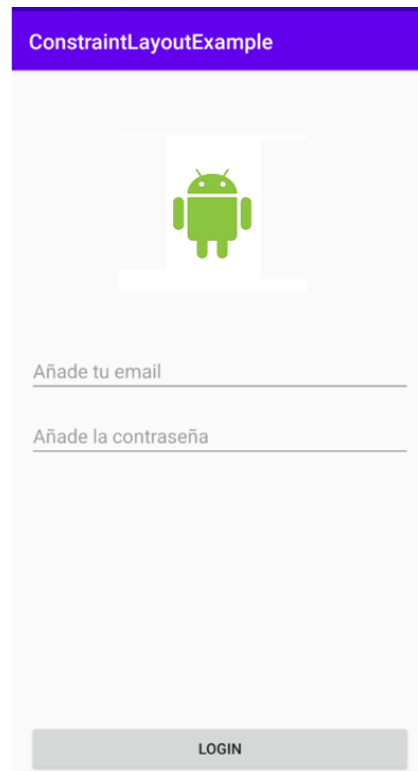
-. **Alineación de línea base:** Alinea la línea de base de texto de un componente con la línea de base de texto de otro.

Por ejemplo, en la imagen de abajo, la primera línea de B está alineada con el texto de A.

Para crear una restricción de este tipo, haz clic con el botón derecho en el componente de texto que desees limitar y, luego, en **Show Baseline**. Luego, haz clic en el texto línea de base y arrastra la línea a otra línea de base.



Prueba a crear la siguiente pantalla usando **ConstraintLayout**:



-. **Uso de Cadenas (Chain):** Las cadenas son restricciones que permiten distribuir linealmente un grupo de controles que están conectados bidireccionalmente. Los controles dentro de una cadena se pueden distribuir en sentido horizontal o vertical.

La "cabeza" de la cadena (el componente del extremo izquierdo en una cadena horizontal) (en un diseño de izquierda a derecha) y el componente superior en una cadena vertical: define el estilo de la cadena en XML.

Existen los siguientes tipos principales de cadenas:

1. Spread o Distribuida: Los componentes se distribuyen uniformemente después de definir los márgenes.

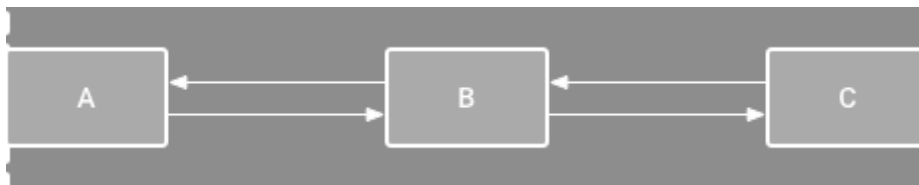


Para crear una cadena horizontal desde el editor de layouts en Android Studio, primero configura el margen izquierdo, arriba y debajo del componente que va a ser la cabeza de la cadena, así como el margen derecho del último componente de la cadena. Después selecciona todos los componentes que formarán parte de la cadena, luego haz click derecho sobre ellos y selecciona la opción **Chains>Create Horizontal Chain**. Después habrá que alinear los componentes (botón derecho: Align>Top Edges por ejemplo)

El diseño por defecto será Spread, si se quisiera cambiar el diseño, selecciona los componentes de la cadena y luego haz click derecho sobre ellos y selecciona la opción **Chains> Horizontal Chain Style** y elige el estilo deseado de entre las opciones disponibles (Spread / Spread Inside / Packed)

Se puede cambiar el estilo, eslabón por eslabón y que cada uno tenga un estilo diferente.

2. Spread inside o Distribuida por dentro: El primer y último componente se fijan en las restricciones en cada extremo de la cadena, y el resto se distribuyen.



3. Packed o empaquetados: Los elementos de la cadena se empaquetan juntos. Lo que permite que el sesgo/bias establecido en la cabecera afecte a todo el grupo como una unidad.

El bias o sesgo es una restricción que entra en acción cuando limitas ambos lados de un eje. Aunque en el centrado no se especifica el valor de este, su valor por defecto es 50% del espacio entre los orígenes de conexión. Su objetivo es determinar la cantidad de desplazamiento que el componente recorrerá en el eje acotado por restricciones.



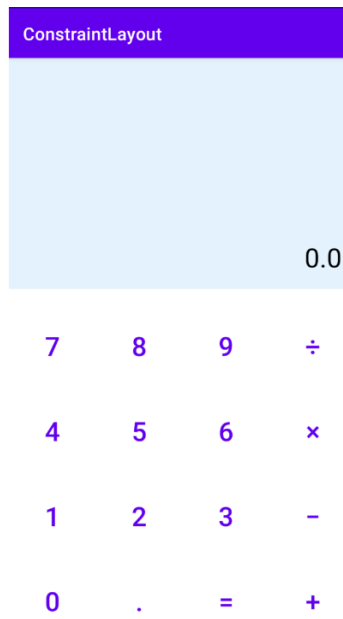
4. Weighted o Ponderada: Si la cabeza está usando los modos CHAIN_SPREAD o CHAIN_SPREAD_INSIDE y las dimensiones de todos los componentes de la cadena tienen MATCH_CONSTRAINT. Es posible manipular el espacio que ocupará el elemento a partir de ponderaciones (pesos).

Pudiéndose asignar un peso a cada componente usando el *layout_constraintHorizontal_weight* y *layout_constraintVertical_weight*.

El componente con el valor de ponderación más alto obtiene la mayor cantidad de espacio, y las vistas con el mismo peso tienen la misma cantidad de espacio.



Prueba a crear la siguiente pantalla usando **ConstraintLayout** y usando cadenas.



ConstraintLayout

<https://developer.android.com/develop/ui/views/layout/constraint-layout?hl=es-419>