


Open Opened 3 weeks ago by  **Rubén Montero**

ConstraintLayout

Resumen

- Introduciremos el funcionamiento de `ConstraintLayout` y el uso de *constraints*
- Añadiremos 3 *constraints* al botón

Descripción

El elemento raíz de las interfaces Android con las que trabajaremos, [ConstraintLayout](#), sirve para albergar vistas y posicionarlas de forma sencilla y escalable.

Constraint

Son *atributos* de los elementos que especifican su posición *dentro* del `ConstraintLayout`, y tienen una forma como:

```
app:layout_constraintBottom_toBottomOf="parent"
```

(Nótese que el prefijo es `app`: en vez de `android`:)

La alerta

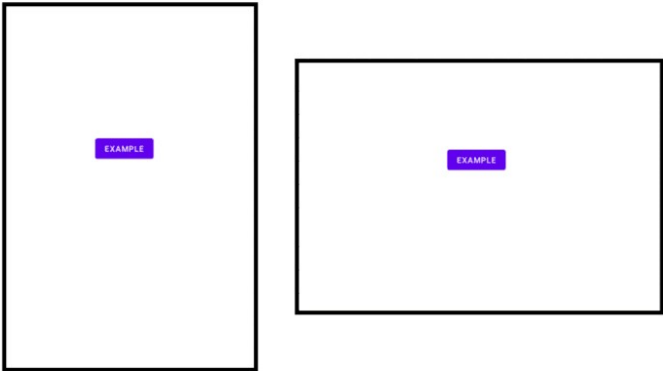
En nuestro `<Button>` no hemos especificado ninguna *constraint* y por eso aparece esta advertencia:

```
This view is not constrained. It only has designtime position, so it will jump to (0,0) at runtime unless you add the constraints
```

Es decir, que la razón por la que el botón aparece *arriba a la izquierda* (el origen de coordenadas (0,0)) no es arbitraria, sino un efecto directo de que faltan *constraints*. Podemos imaginar que el `<Button>` está *flotando libremente* porque ninguna *constraint* lo restringe.

En [la documentación](#) hay muchos casos de uso bien explicados. Comencemos viendo unas *constraint* básicas:

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Example" />
```



Como puedes observar, una *constraint* indica:

- **Qué (1)** parte del componente se unirá a...
- **...qué (2)** parte de otro componente **(3)**.

En el ejemplo, se une el **Top (1)** del botón con el **Top (2)** del `"parent"`, es decir, el contenedor **(3)**. Si hubiera varios niveles de jerarquía, `"parent"` haría referencia al inmediatamente superior (no necesariamente al `ConstraintLayout` raíz).

```
(1)  (2)  (3)
app:layout_constraintTop_toTopOf="parent"
```

La siguiente *constraint* une el **Start (1)** del botón con el **Start (2)** del contenedor **(3)**:

```
(1)  (2)  (3)
```

```
app:layout_constraintStart_toStartOf="parent"
```

Start es *izquierda*, con la ventaja de que si nuestra aplicación se ejecuta en un dispositivo con idioma local *Right-To-Left* (como árabe o hebreo), **Start** se posicionará en la *derecha* automáticamente, para ofrecer a esos usuarios su sentido de lectura natural.

Las cuatro dimensiones que podemos usar para *arriba*, *abajo*, *izquierda* y *derecha* respectivamente son:

- **Top**
- **Bottom**
- **Start**
- **End**

La tarea

En `activity_main.xml`, **añade** las *constraints* expuestas anteriormente a tu `<Button>`:

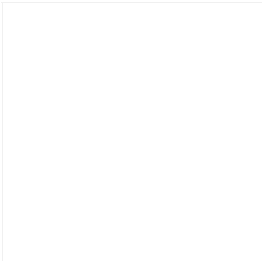
```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Ver tostada"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintStart_toStartOf="parent" >
</Button>
```

Si ejecutas la aplicación, verás que el botón aparece *arriba a la izquierda*. Igual que antes, pero ahora no hay ninguna advertencia en el XML indicando que la posición del botón es *ambigua*.

Centrar el botón

Es posible especificar simultáneamente una *constraint* **Start_toStart** y **End_toEnd**.

Dado que el *ancho* está especificado como `wrap_content`, cada *constraint* "tirará" del componente hacia su lado y se centrará¹.




Añade una *constraint* **End_toEnd** para *centrar* tu botón en la parte superior.

Puedes lanzar la aplicación y verificar que el botón aparece centrado arriba.

Por último

Haz `git add *` y `git commit -m "Tarea 3, botón centrado"` para crear un *commit*. Sube tus cambios al repositorio con `git push`.

1. Podrías esperar que el `<Button>` se deformase y se *pegase* a ambos lados, ocupando todo el ancho. Eso puede suceder si el **ancho** es *algo distinto* a `wrap_content`. Lo veremos más adelante. 



Rubén Montero @ruben.montero changed milestone to [%Sprint 1](#) 3 weeks ago



Ania Blanco @ania.blanco mentioned in commit [bbb52f3e](#) 2 weeks ago

