Ambiente de desarrollo y sus elementos de configuración

Los assets de un proyecto Android (Parte II)

{desafío} latam_

Implementar una interfaz de usuario utilizando buenas prácticas en el manejo de estilos para brindar un aspecto visual e interacciones acordes a lo requerido

- **Unidad 1**: Ambiente de desarrollo y sus elementos de configuración.
- **Unidad 2:** Elementos de la interfaz, navegación e interacción.
- **Unidad 3**: Fundamentos de GIT y GitHub.

Te encuentras aquí

{desafío} latam_

¿Qué aprenderás en esta sesión?

- *Utiliza archivos de assets externos a un proyecto Android obtenidos desde Material Icons para complementar el proyecto*
- *Utiliza componentes adicionales de acuerdo a requerimientos dados*

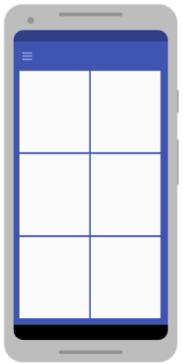
¿Para qué sirve un layout?

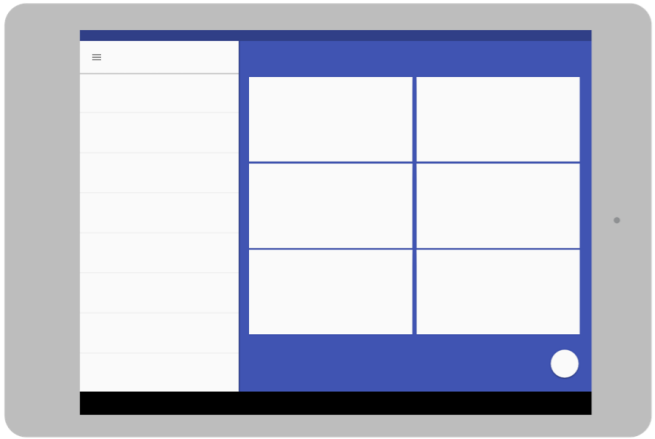
¿Cuántas vistas puede tener un layout?

/* Manejo de layouts para distintas orientaciones y tamaños */

Orientaciones y tamaños

Orientación **portrait** en un teléfono (más espacio vertical que horizontal).





Orientación **landscape** en una tablet (más espacio horizontal que vertical).

Fuente:
https://developer.android.com/training/multiscreen/screensizes?hl=es-419

/* Layouts */

Layout

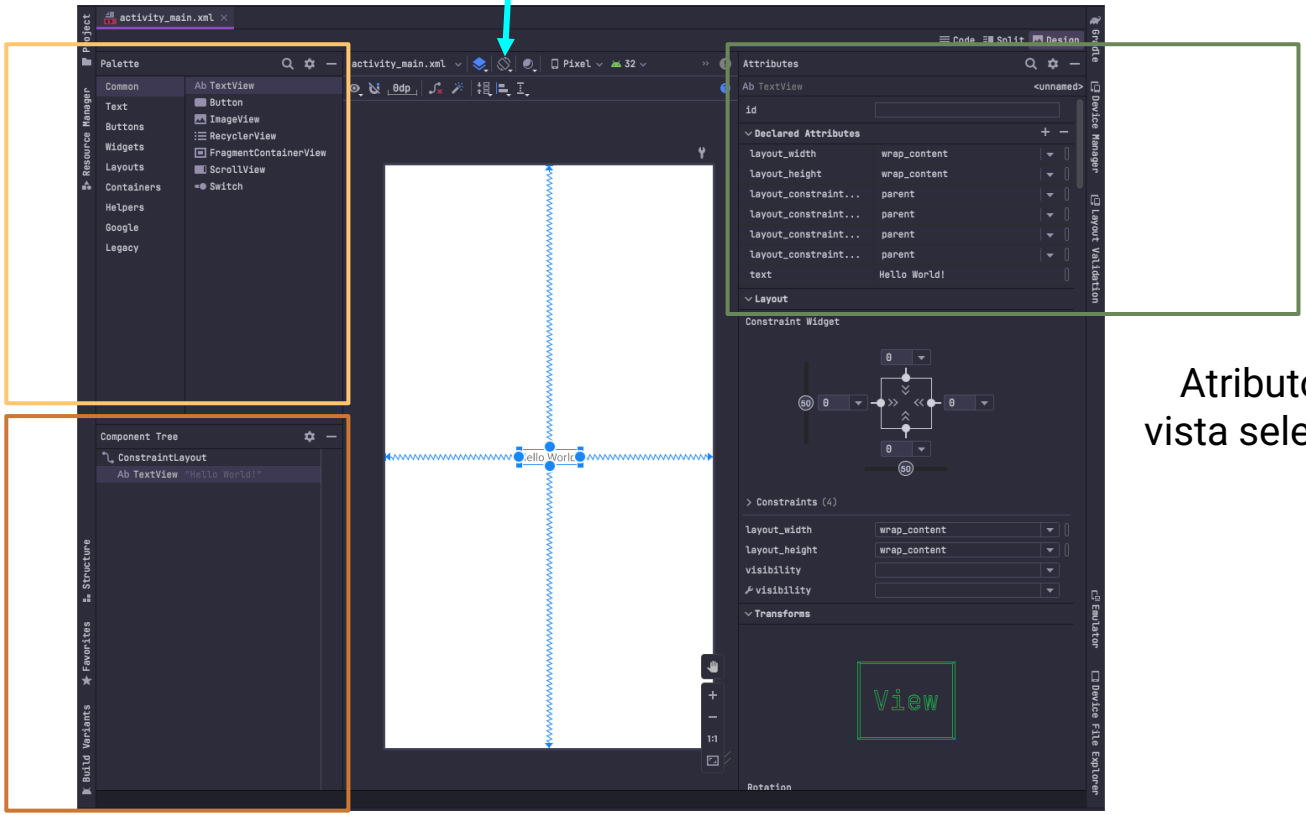
Definición

- Un layout define la **estructura** de la interfaz de usuario en la app.
- Todos los elementos (vista o *widget*) en el layout se construyen heredando de View y ViewGroup.
- Una **vista** generalmente dibuja algo que el usuario puede **ver e interactuar.**
- **ViewGroup** es un **contenedor invisible** (layout) que define la estructura para un widget y/u otros ViewGroup.
- La estructura del layout se define en un archivo XML.

Editor de vistas

Cambiar orientación

Paleta de widgets



Atributos de la vista seleccionada

Árbol jerárquico con todas las vistas

{desafío} latam_

Actividad

Editor de vistas

1. Crear un nuevo proyecto.
    a. Usando la plantilla de Empty Activity.
    b. Min SDK - API 26.
2. Abrir el layout activity_main.xml.
3. Agregar un TextView desde la paleta.
4. Probar en un dispositivo.

- ¿En qué posición está el TextView?
- ¿Se ve igual que en la vista de diseño?

{desafío} latam_

Constraint Layout

- Es el tipo de layout que se utiliza por defecto.
- Muy flexible, permite ubicar vistas en forma relativa a su contenedor o relativa a otras vistas.
- Mejora la performance de la app en comparación con otros layouts como LinearLayout.

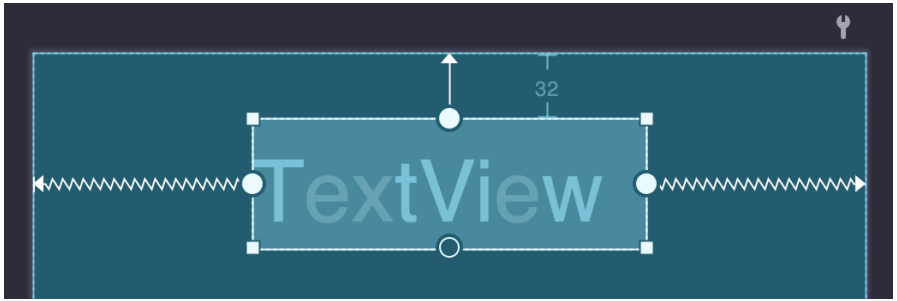
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout


xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    >
</androidx.constraintlayout.widget.ConstraintLayout>
```

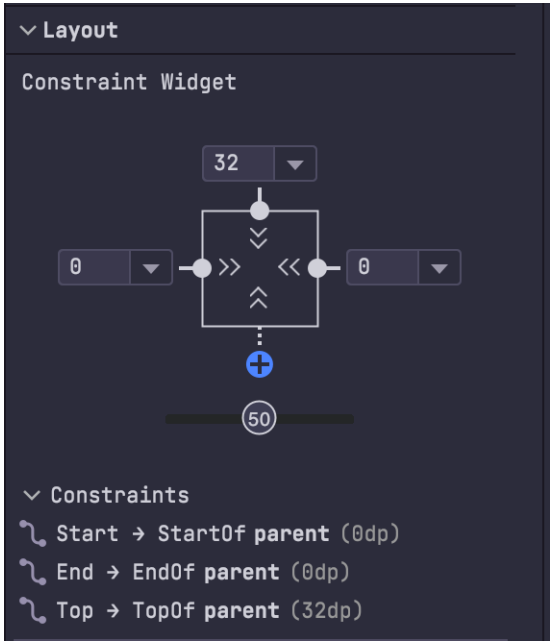
Constraint Layout

Constraints (Restricciones)

Las restricciones permiten a la vista indicar al ConstraintLayout dónde ubicarse.

En este caso, es relativo a la pantalla y tiene un márgen de 32dp.



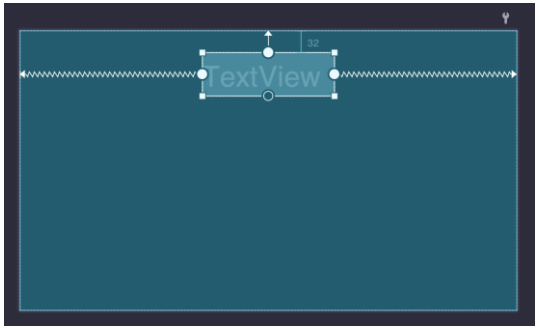


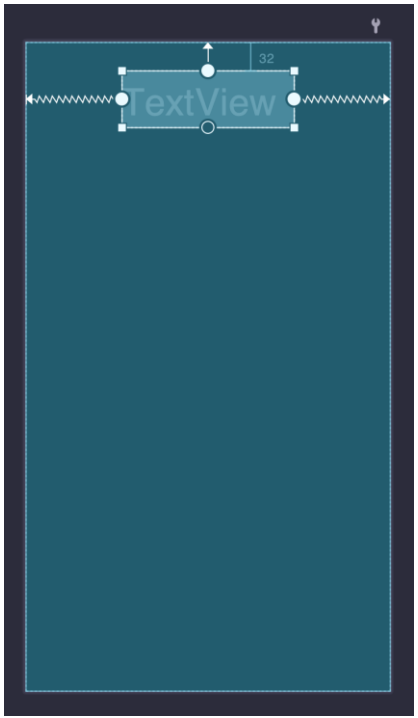
Orientación de layout

Landscape | Portrait

Landscape

Es la orientación más utilizada en tablets.





Portrait

Es la orientación por defecto para teléfonos.

Reutilizar layouts

<include>

Algunas veces se deben reutilizar componentes más grandes con un diseño especial incorporándose dentro del diseño actual.

Por ejemplo:

- Una barra de progreso con un texto personalizado, puede ser incorporada en distintas pantallas de la aplicación.
- El tag <include> permite referenciar a otro layout previamente declarado en el directorio de layouts del proyecto.
- Se incluyen 2 layouts distintos para formar 1 sola pantalla.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   >

   <include
       layout="@layout/titlebar"
       tools:layout_editor_absoluteX="0dp"
       tools:layout_editor_absoluteY="0dp" />

   <include
       layout="@layout/main_content"
       tools:layout_editor_absoluteX="0dp"
       tools:layout_editor_absoluteY="0dp" />
</androidx.constraintlayout.widget.ConstraintLayout
>
```

Constraint Layout

Actividad

En la actividad anterior se agregó un TextView al layout activity_main.xml creado desde una plantilla de Empty Activity. Ahora deberás:

1. Agregar 3 constraints (left, right, top) al TextView.
2. Darle un margen de 24 dp a la parte superior (top).
3. Probar en el dispositivo.

¿Son consistentes las vistas de diseño y lo que se ve en el dispositivo?

/* Utilizar los recursos importados */

Utilizar los recursos importados

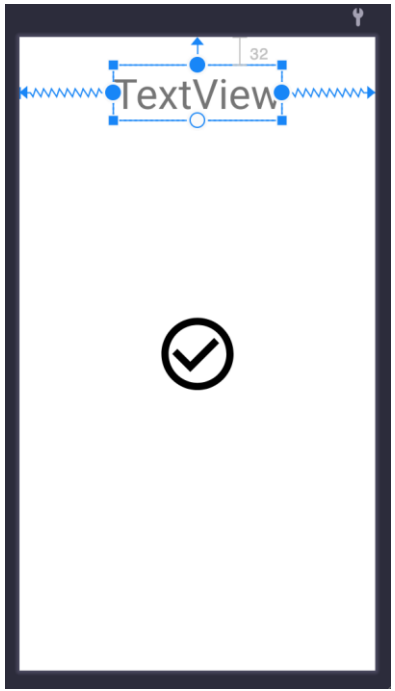
Vector Asset, ImageView y ConstraintLayout

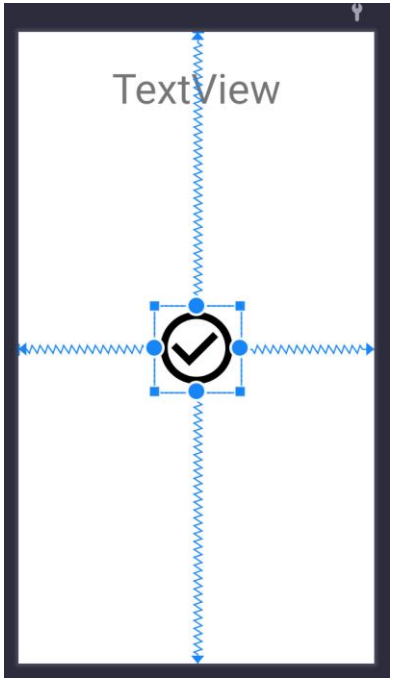
Vamos a colocar en práctica lo que hemos aprendido, para lo que seguiremos con el ejemplo anterior:

1. Agregar un Vector Asset y seleccionar desde la biblioteca de material el icono "*check circle outline*".
2. Agregar un ImageView y seleccionar desde el selector de recursos el vector creado previamente (el nombre por defecto es @drawable/ic_baseline_check_circle_outline_24).
3. Agregar las 4 constraints para que el ImageView se ubique al centro de la pantalla.
4. Cambiar el tamaño del ImageView a altura: 100dp, ancho: 100dp.
5. Agregar un TextView que esté centrado horizontalmente y un *margin top* de 32 dp.
6. Probar en un dispositivo y verificar que la vista de diseño sea consistente.

Utilizar los recursos importados

Agregar las constraints a las vistas





Utilizar los recursos importados

Vista portrait | landscape

- ¿Qué pasa al cambiar la orientación de portrait a landscape?

¿Cuándo debo hacer un diseño distinto para portrait y landscape?

¿Para qué sirven las constraints en un ConstraintLayout?

{desafío} latam_

Próxima sesión...

Guía de ejercicios - Ambiente de desarrollo y sus elementos de configuración

{desafío}
latam_

{desafío}
latam_

Academia de
talentos digitales