

Vistas, Layouts y Temas en Android

DESARROLLO DE APLICACIONES MÓVILES NATIVAS - 7CV3

CONTENIDO

- | | | | |
|----|----------------|----|-------------------|
| 1. | INTRODUCCIÓN | 7. | CONTAINERS |
| 2. | VIEWS O VISTAS | 8. | LAYOUTS |
| 3. | ETIQUETAS XML | 9. | CONSTRAINT LAYOUT |
| 4. | VIEWS COMUNES | 10 | TEMAS |
| 5. | BOTONES | 11 | TEMAS VS ESTILOS |
| 6. | WIDGETS | 12 | CONCLUSIÓN |

INTRODUCCIÓN

El desarrollo de aplicaciones móviles con Android Studio implica comprender ciertos conceptos clave que determinan cómo se construyen y presentan las interfaces de usuario.

Al final del día, las interfaces gráficas son la forma en que el usuario interactúa con una aplicación, lo que es de vital importancia conocer herramientas para optimizar este apartado.

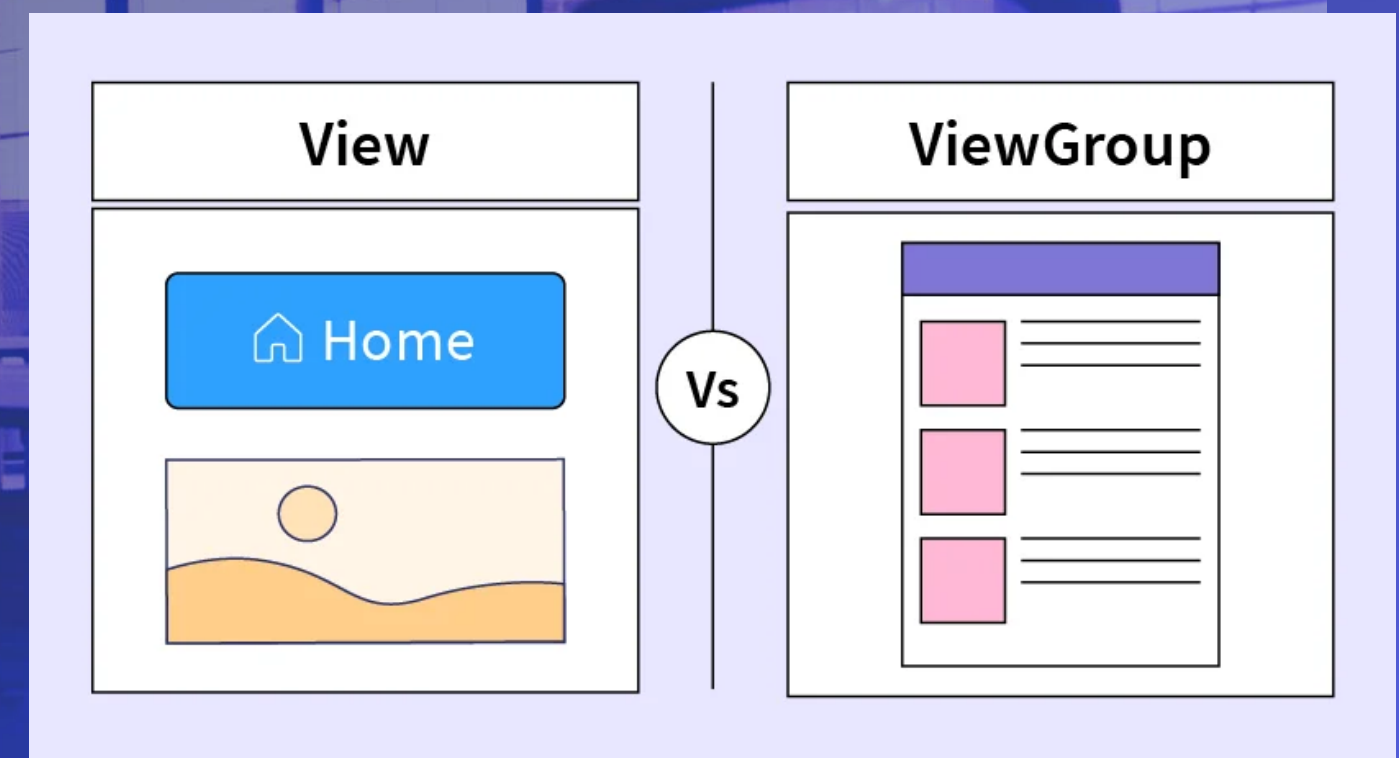
Entre estos conceptos, las vistas, la distribución de componentes (layouts) y los temas son fundamentales para crear aplicaciones que no solo sean funcionales, sino también visualmente atractivas y fáciles de usar.

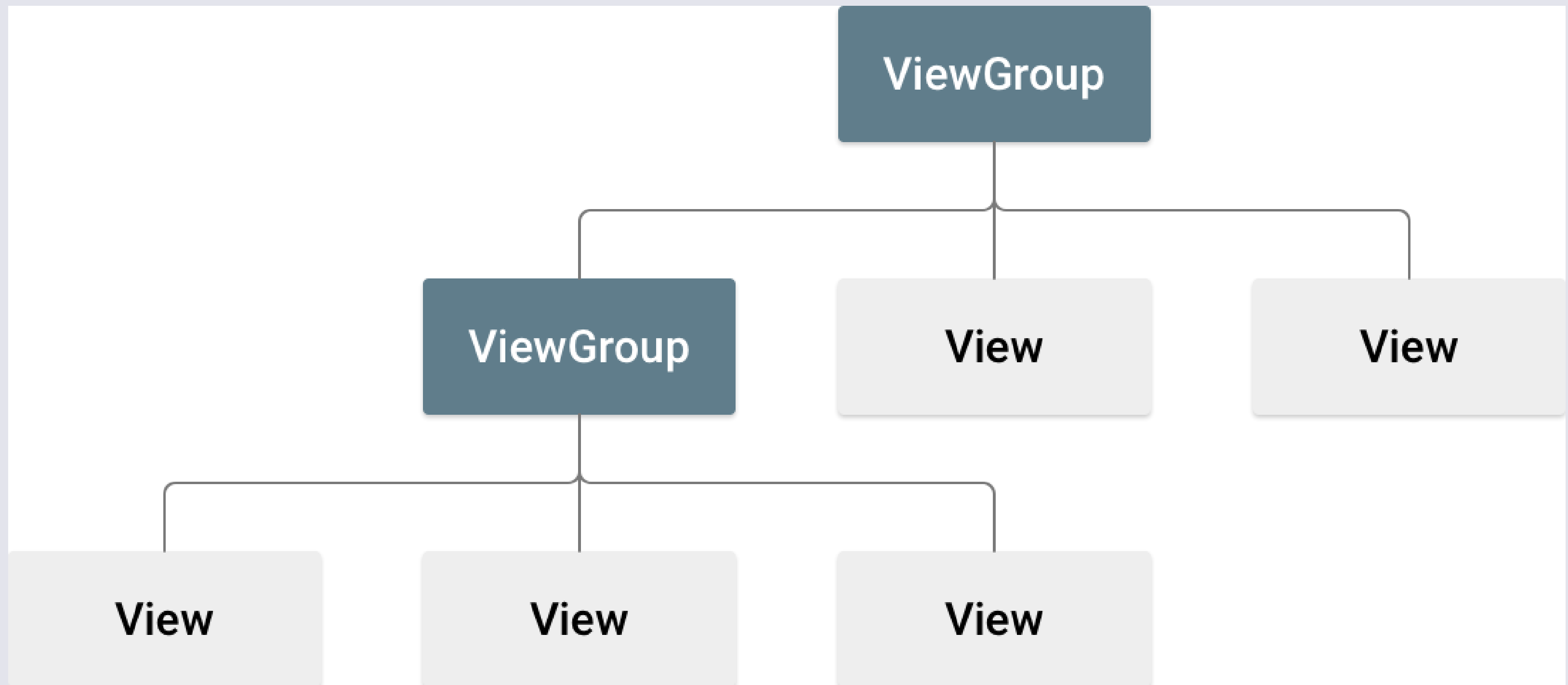


VIEWS (VISTAS)

Las vistas (views) en Android representan los componentes visuales básicos que constituyen la interfaz gráfica de una aplicación. Los objetos View a menudo se denominan widgets y pueden ser uno de muchas subclases, como Button, TextView, TextArea, etc. Una vista es la clase base para todos los widgets y contiene todos los elementos que pueden ser interactuados por el usuario.

Todos los elementos de diseño se construyen mediante una jerarquía de View y ViewGroup objetos. Un View suele dibujar algo que el usuario puede ver y para interactuar. Un ViewGroup es un contenedor invisible que define la estructura de diseño de View y otros ViewGroup objetos






```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Introduce tu nombre:"
        android:textSize="18sp" />

    <EditText
        android:id="@+id/editTextName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Nombre" />

    <Button
        android:id="@+id/buttonSubmit"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Enviar" />

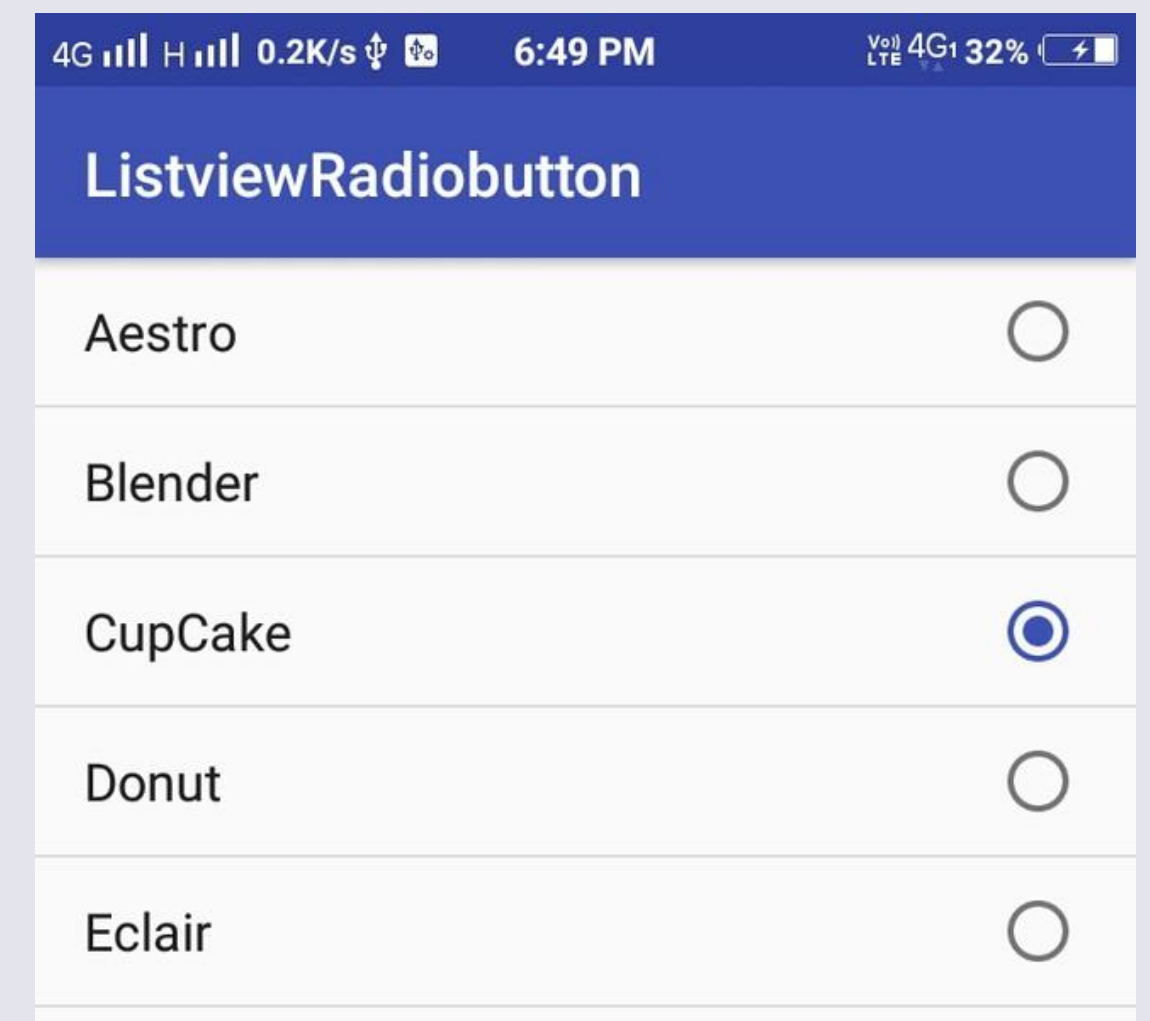
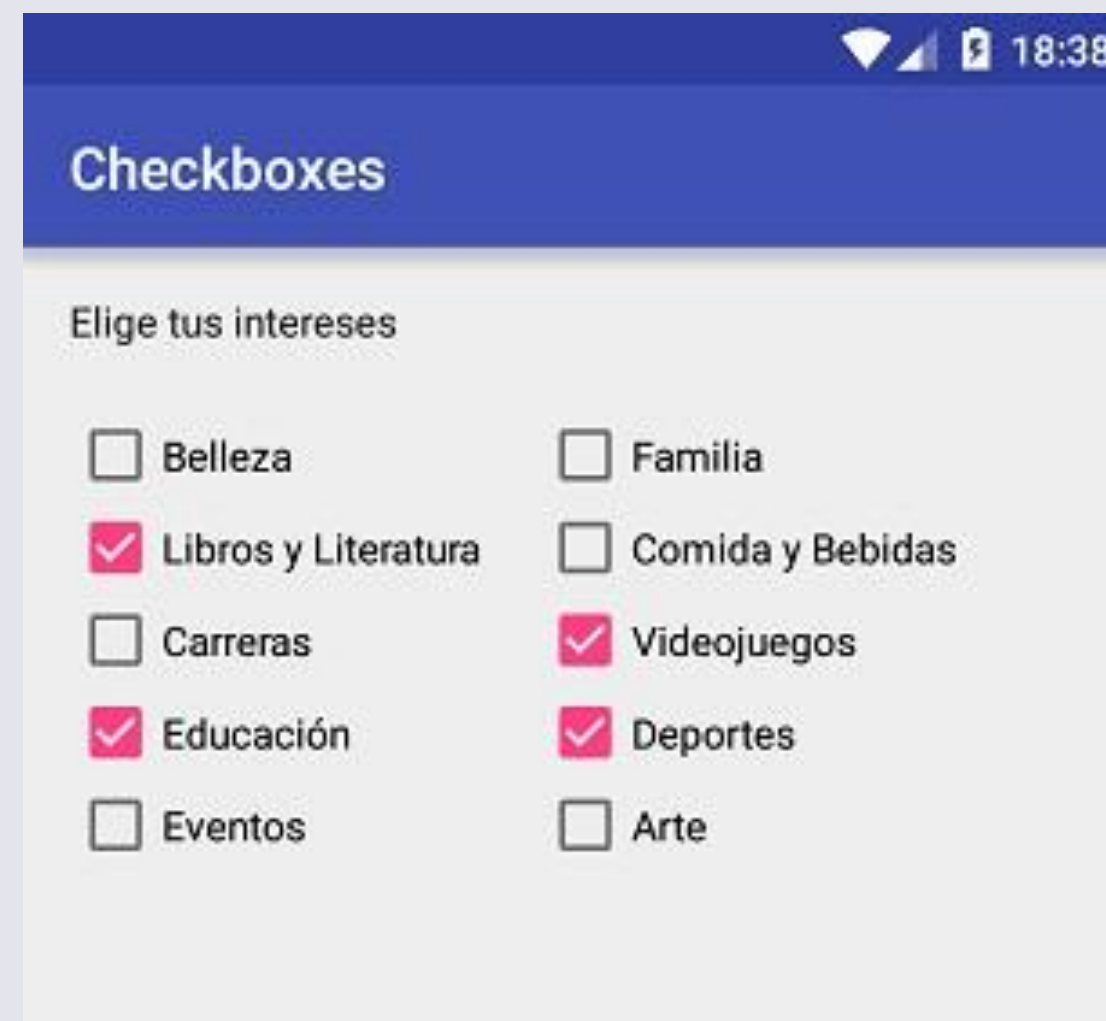
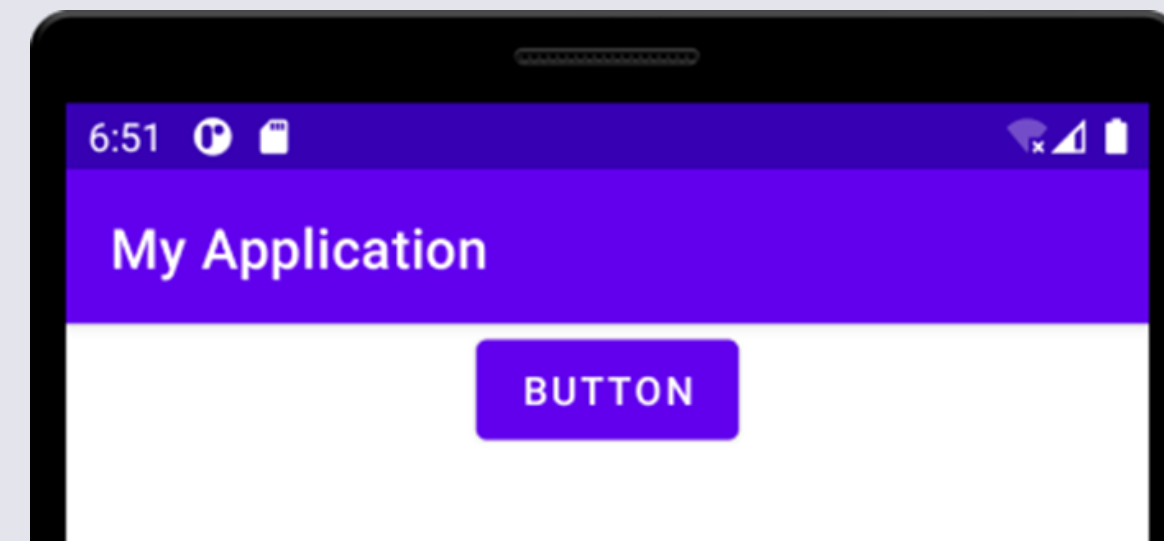
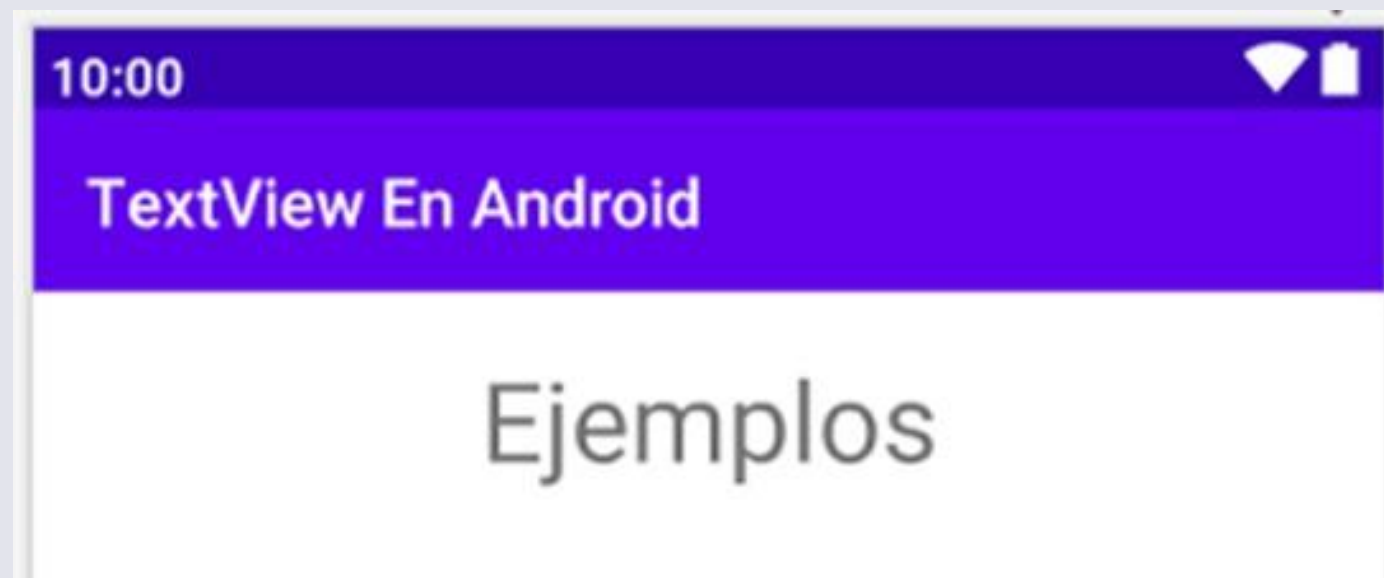
    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/ic_launcher_foreground"
        android:contentDescription="@string/image_desc" />

</LinearLayout>

```

CARACTERÍSTICAS

- Permiten recibir y manejar entradas del usuario, como clics, deslizamientos, o toques.
- Se organizan en una jerarquía que permite heredar de otras clases
- A través de los listeners, puedes definir comportamientos interactivos



DEFINICIÓN DE UNA ETIQUETA EN XML

Scroll View: Permite que los elementos dentro de él se desplacen verticalmente. Es útil para pantallas con mucho contenido.

```
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"
```

```
<EditText  
    android:id="@+id/editText"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:hint="Introduce tu texto aquí"  
    android:inputType="text"/>
```








COMUNES

TextView: Muestra texto en la pantalla. Se usa para mostrar etiquetas, títulos o cualquier texto que no sea editable por el usuario.

EditText: Campo de texto editable que permite al usuario introducir datos. Muy utilizado en formularios para recibir entradas de texto.

Button: Botón que el usuario puede presionar para realizar alguna acción. También hay variantes como **ImageButton**, que permite usar una imagen en lugar de texto.

ImageView: Muestra imágenes en la interfaz. Puedes cargar imágenes desde los recursos o desde una URL si lo configuras.

Palette	
Common	Ab TextView
Text	 Button
Buttons	 ImageView
Widgets	 RecyclerView
Layouts	 FragmentContainerView
Containers	 ScrollView
Helpers	 Switch
Google	
Legacy	

BOTONES

Button

Es el botón estándar que el usuario puede presionar para ejecutar una acción. Es personalizable en cuanto a color, tamaño, y texto.

ImageButton

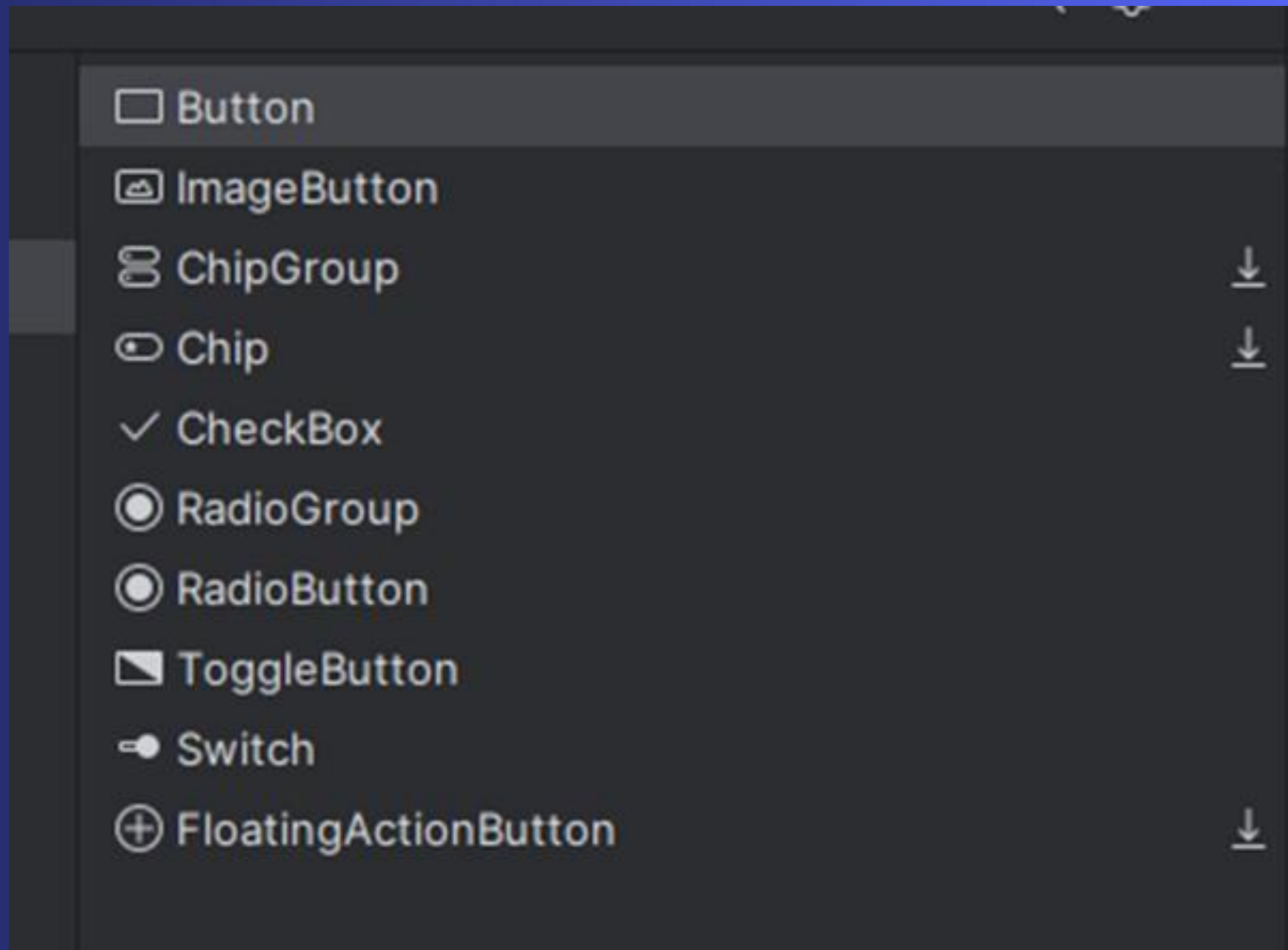
Similar al botón estándar, pero en lugar de texto, muestra una imagen. Es útil cuando quieres que el botón tenga un ícono o gráfico.

FloatingActionButton

Botón flotante circular que se usa comúnmente para acciones principales en la aplicación, como agregar un nuevo elemento.

ToggleButton

Botón de encendido/apagado que permite al usuario alternar entre dos estados (activado/desactivado). Es útil para configuraciones o ajustes rápidos.



BOTONES

Switch

Botón de interruptor que también permite alternar entre dos estados, pero tiene un diseño de deslizador.

RadioButton

Parte de un grupo de opciones donde el usuario puede seleccionar solo una.

Checkbox

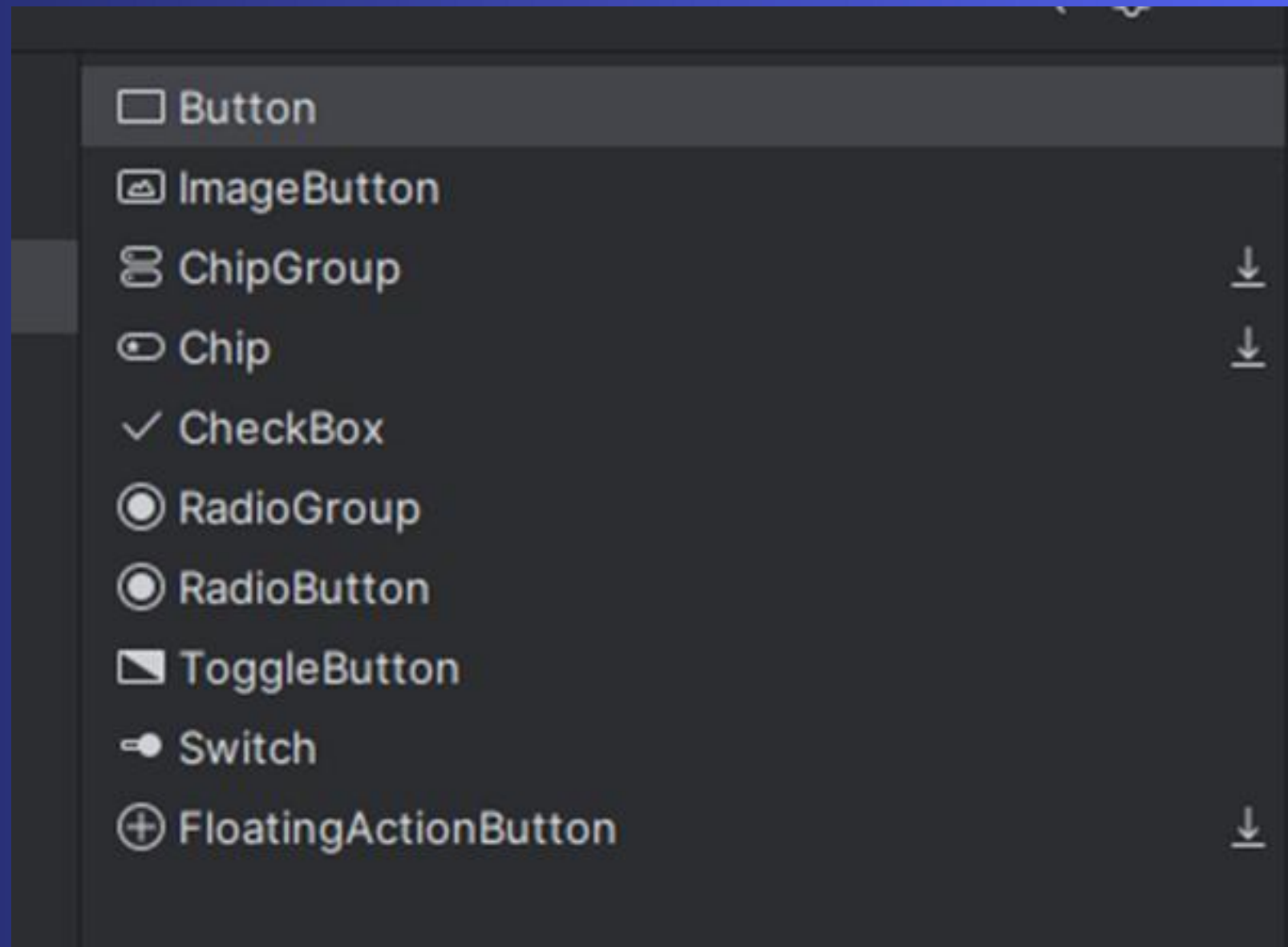
Casilla de verificación que permite al usuario seleccionar o deseleccionar opciones. Permite seleccionar varias opciones al mismo tiempo.

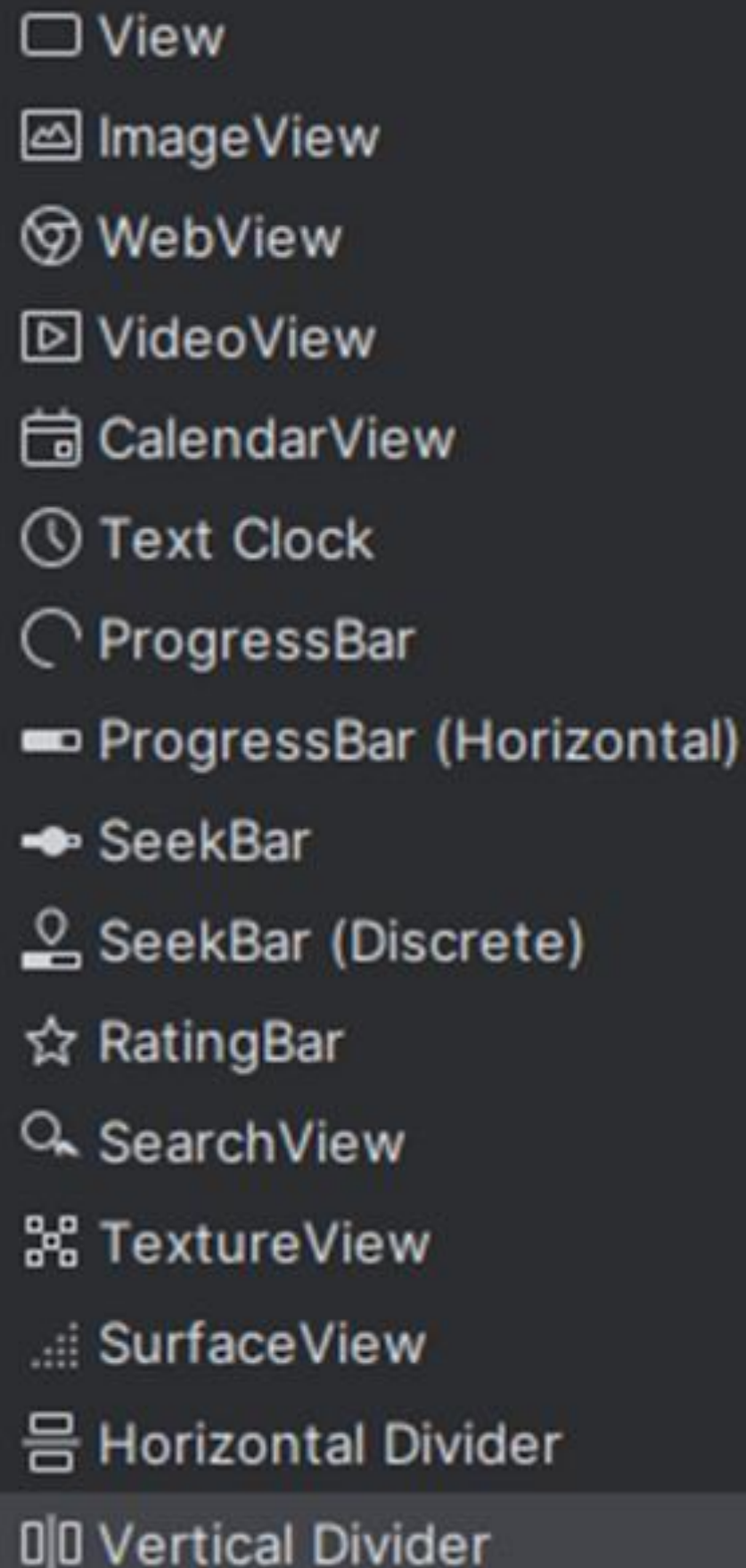
Chip

Elemento interactivo que funciona como un botón, pero con una apariencia de etiqueta. Los chips son útiles para mostrar opciones seleccionables o etiquetas que el usuario puede activar o desactivar.

MaterialButton

Un botón más personalizable que el estándar, parte de la librería de Material Design, que permite agregar íconos, cambiar bordes, y estilizar fácilmente en diferentes estados.





- View
- ImageView
- WebView
- VideoView
- CalendarView
- Text Clock
- ProgressBar
- ProgressBar (Horizontal)
- SeekBar
- SeekBar (Discrete)
- RatingBar
- SearchView
- TextureView
- SurfaceView
- Horizontal Divider
- Vertical Divider

WIDGETS

View: El elemento básico en Android, usado como base para otros componentes y contenedores. No tiene funcionalidades específicas por sí solo.

ImageView: Muestra una imagen en la interfaz. Puedes configurar su tamaño, posición y fuente de imagen.

VideoView: Permite cargar y mostrar contenido web dentro de la app. Es como un navegador integrado.

VideoView: Widget que reproduce videos, ya sea desde un archivo local o una URL.

CalendarView: Muestra un calendario interactivo para seleccionar fechas.

TextClock: Muestra la hora actual en la interfaz, y se actualiza en tiempo real.

ProgressBar: Indicador visual de progreso que muestra una animación de carga. Puede ser circular o en barra.

WIDGETS

ProgressBar (Horizontal): Variante de ProgressBar que muestra el progreso en una barra horizontal.

SeekBar: Barra deslizable que permite al usuario seleccionar un valor dentro de un rango, útil para ajustes como volumen o brillo.

SeekBar (Discrete): Una versión de SeekBar con valores discretos (escalonados), en lugar de ser continua.

RatingBar: Permite al usuario dar una calificación, usualmente en forma de estrellas.

SearchView: Barra de búsqueda que permite al usuario ingresar consultas. Es ideal para buscar contenido dentro de la app.

TextureView: Permite mostrar contenido multimedia que se renderiza directamente desde una textura. Es útil para gráficos avanzados y multimedia.

SurfaceView: Vista que permite dibujar directamente en un área de la pantalla. Se usa para gráficos y videos personalizados.

- View
- ImageView
- WebView
- VideoView
- CalendarView
- Text Clock
- ProgressBar
- ProgressBar (Horizontal)
- SeekBar
- SeekBar (Discrete)
- RatingBar
- SearchView
- TextureView
- SurfaceView
- Horizontal Divider
- Vertical Divider

CONTAINERS

ScrollView: Permite que una vista contenedora sea desplazable, útil para mostrar contenido que excede el tamaño de la pantalla.

RecyclerView: Un contenedor más avanzado para listas de elementos. Ofrece un rendimiento mejorado y es más flexible que ListView.

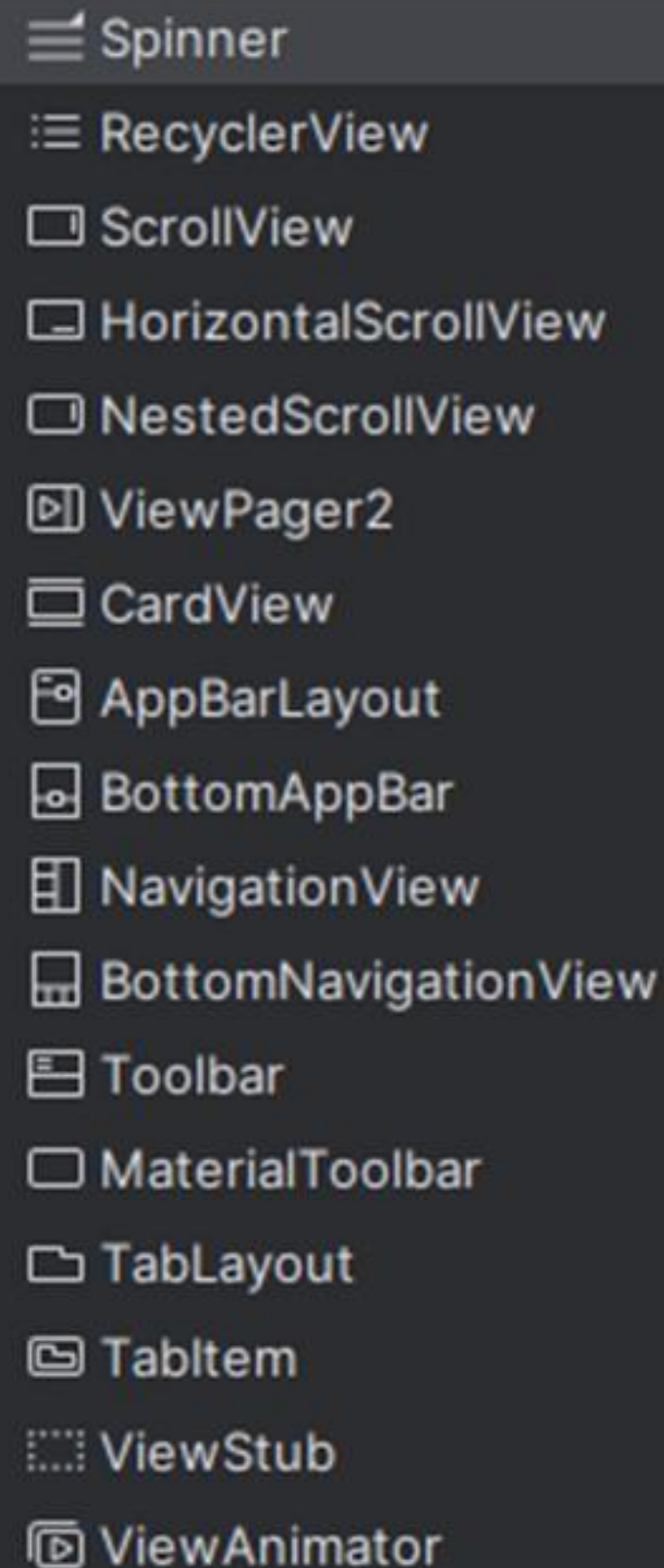
ListView: Muestra una lista de elementos que se pueden desplazar, pero es menos eficiente que RecyclerView para grandes conjuntos de datos.

Spinner: Un componente que permite seleccionar un valor de una lista desplegable.

ViewPager: Permite deslizar entre fragmentos en una vista paginada.

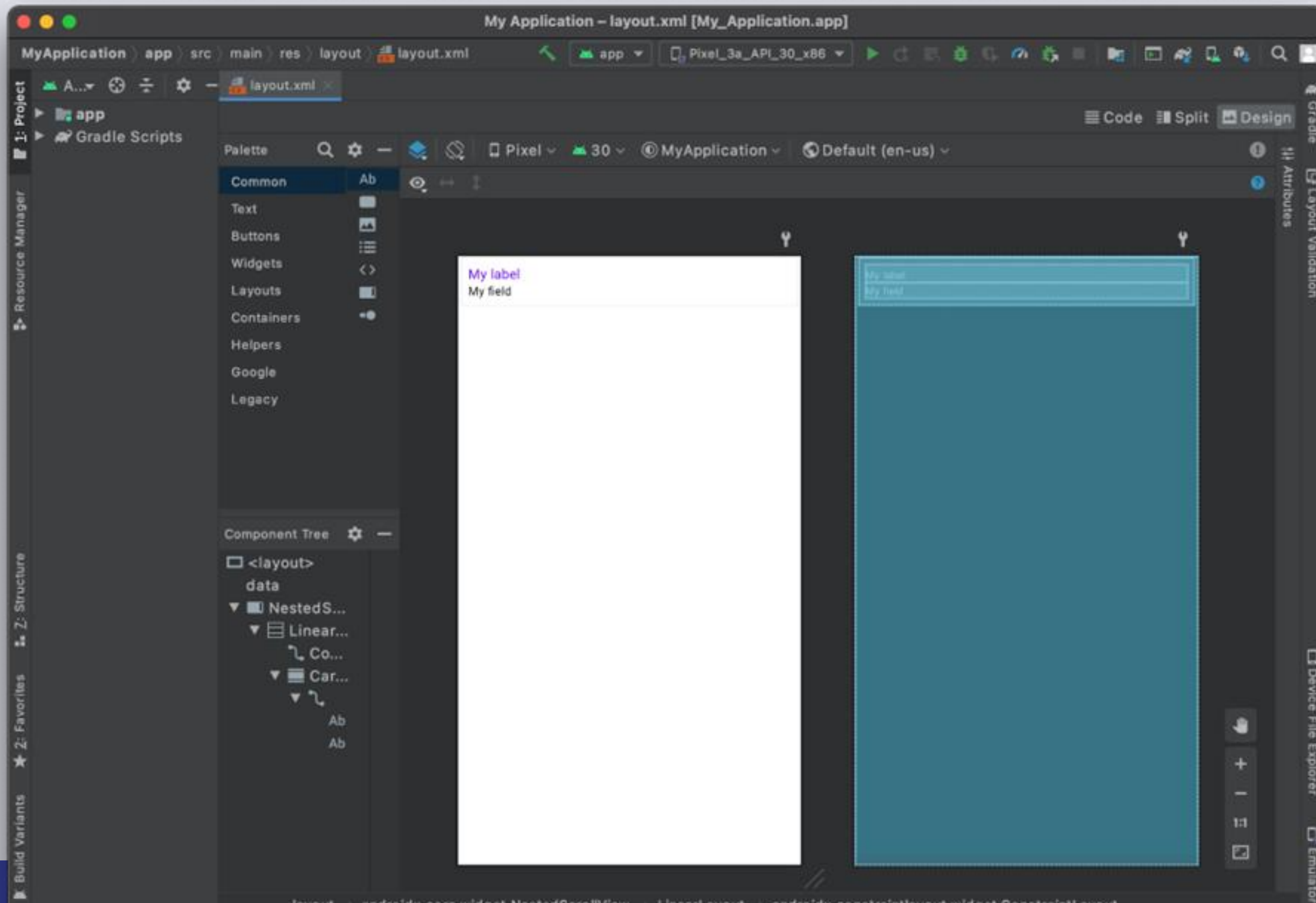
TabLayout: Proporciona una interfaz para mostrar pestañas, que se puede usar junto con ViewPager.

BottomNavigationView: Proporciona una barra de navegación en la parte inferior de la pantalla con varias opciones.



- Spinner
- RecyclerView
- ScrollView
- HorizontalScrollView
- NestedScrollView
- ViewPager2
- CardView
- AppBarLayout
- BottomAppBar
- NavigationView
- BottomNavigationView
- Toolbar
- MaterialToolbar
- TabLayout
- TabItem
- ViewStub
- ViewAnimator

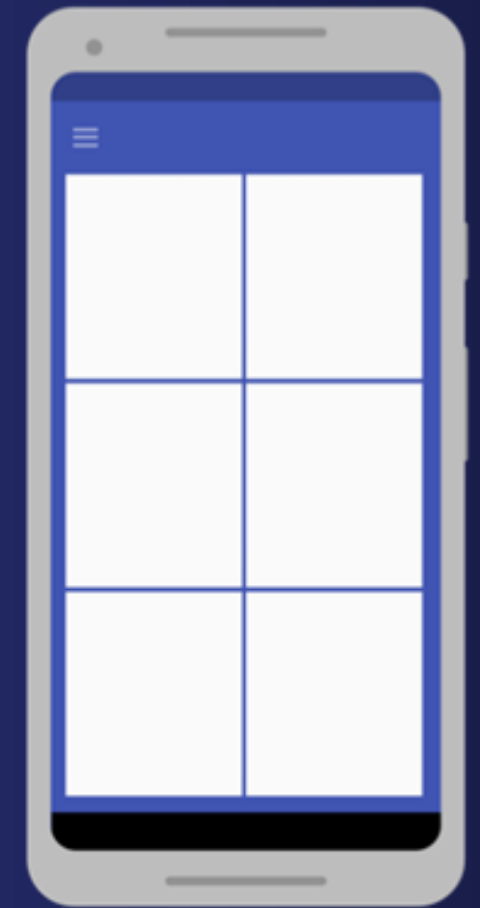
LAYOUTS



En Android Studio, los layouts son la estructura que define la disposición visual de los elementos en la interfaz gráfica de usuario (GUI) de una aplicación. Son el primer paso para desarrollar las vistas en Android

LAYOUTS

Se pueden entender como contenedores que organizan las vistas dentro de una actividad o fragmento, que controlan cómo se posicionan y visualizan los componentes en la pantalla. Algunos ejemplos son: dunt ut labore et dolore magna aliqua. Ut enim ad minim veniam.



LINEAR LAYOUT

Alinea los elementos en una dirección (vertical u horizontal).

CONSTRAINT LAYOUT

Proporciona más control y flexibilidad, ya que las vistas se posicionan con restricciones (constraints), lo que permite crear interfaces más complejas y adaptables a diferentes tamaños de pantalla.

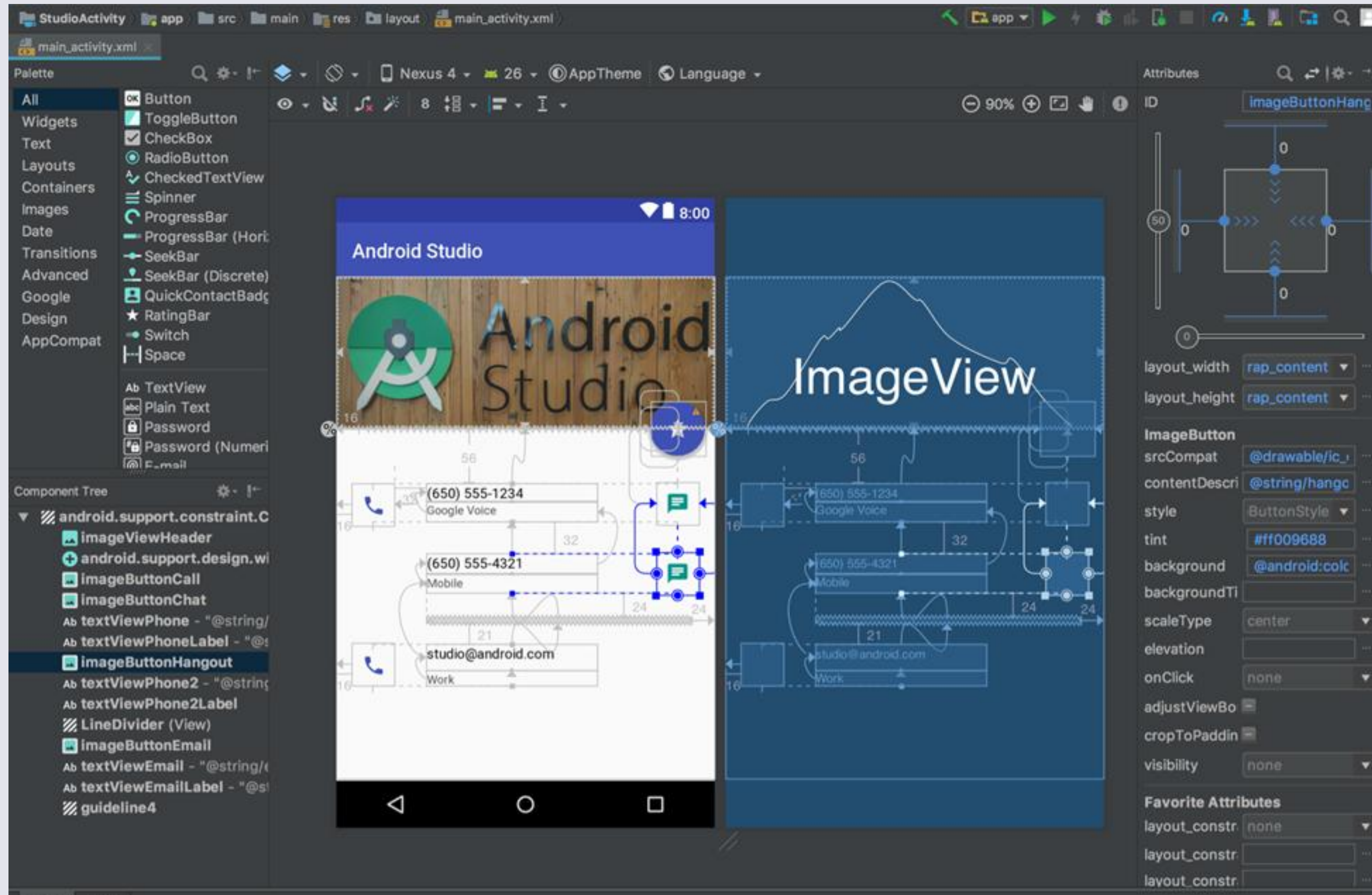
FRAME LAYOUT

Contiene solo una vista o múltiples vistas superpuestas, útil para casos donde se necesitan vistas apiladas.

RESULTADO

Organiza las vistas en una cuadrícula de filas y columnas.

CONSTRAINT LAYOUTS

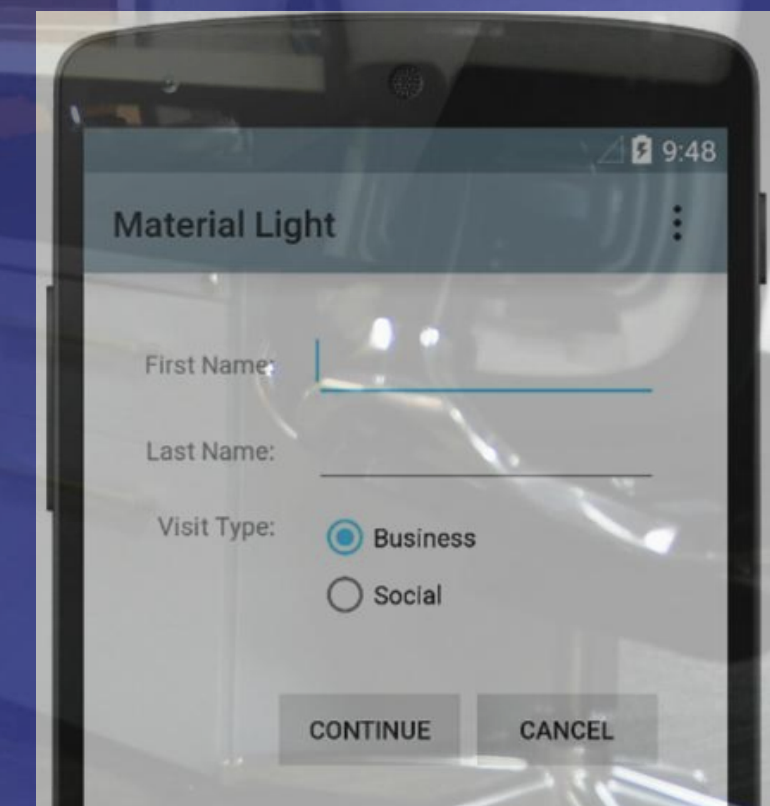
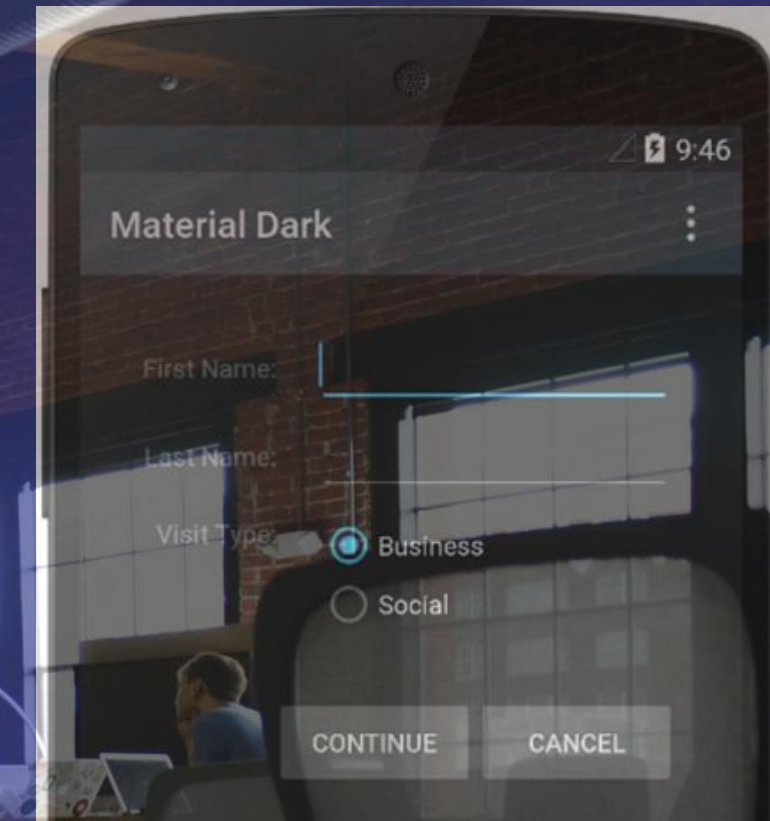


TEMAS

Los temas en Android permiten separar los detalles del diseño de la aplicación de la IU estructura y comportamiento similares a las hojas de estilo en el diseño web.

Un tema es una colección de atributos que se aplican a toda una app, actividad o vista jerarquía, no solo en una vista individual.

Cuando se aplica un tema, cada vista de la app o la actividad aplica cada uno de los atributos del tema que admite. Los temas también pueden aplicar estilos a elementos que no son vistas, como la barra de estado y el fondo de la ventana.



TEMAS VS ESTILOS

Un estilo especifica atributos para un tipo de vista determinado. Un estilo podría especificar los atributos de un botón. Cada atributo que se especifica en un estilo es un atributo que se puede definir en el archivo de diseño. La extracción de todos los atributos de un estilo facilita su uso y mantenimiento en varios widgets.

Un tema por su parte, define una colección de recursos con nombre a los que pueden hacer referencia los estilos, los diseños, widgets, etcétera. Los temas asignan nombres semánticos, como colorPrimary.

Los estilos y los temas se declaran en un archivo de recursos de estilo en res/values/, generalmente con el nombre styles.xml.

TEMAS VS ESTILOS

Ejemplo de cómo aplicar un tema: Se aplica en AndroidManifest.xml

```
<manifest ... >  
    <application android:theme="@style/Theme.AppCompat" ... >  
    </application>  
</manifest>
```

```
<manifest ... >  
    <application ... >  
        <activity android:theme="@style/Theme.AppCompat.Light" ... >  
        </activity>  
    </application>  
</manifest>
```


CONCLUSIÓN

Las vistas permiten que el usuario interactúe y brinde retroalimentación a nuestro sistema, con botones, textos, etc. Los layouts por otra parte, nos brinda una mejor organización de las vistas, lo que asegura que estén correctamente colocadas y alineadas.

Finalmente los temas permiten personalizar la apariencia visual de toda la aplicación, dando coherencia a los colores, estilos y tipografías. Estos conceptos son esenciales para crear una interfaz de usuario atractiva, funcional y personalizada en aplicaciones Android, asegurando un producto de calidad.

FUENTES DE CONSULTA

Diseños en vistas. (n.d.). Android Developers.

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

GeeksforGeeks. (2021, Marzo 17). Difference between View and ViewGroup in Android. GeeksforGeeks.

<https://www.geeksforgeeks.org/difference-between-view-and-viewgroup-in-android/>

Diseño responsivo/adaptable con vistas. (n.d.). Android Developers.

<https://developer.android.com/develop/ui/views/layout/responsive-adaptive-design-with-views?hl=es-419>

Estilos y temas. (n.d.). Android Developers.

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