

Diseño de Interfaces Gráficas



KEEPCODING
Tech School

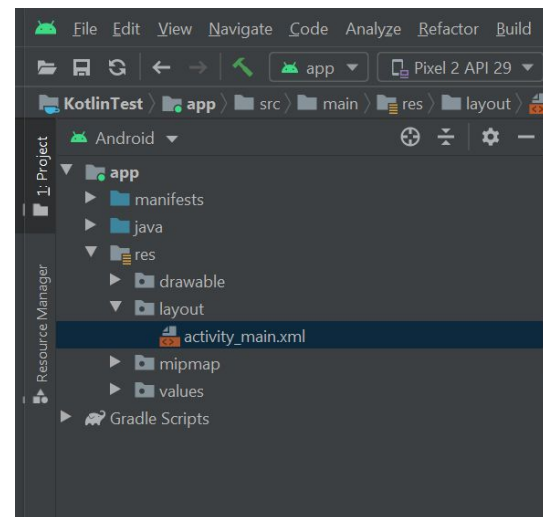


Conceptos básicos

¿Qué es Android Studio?

Directamente por Código Kotlin - Opción poco recomendable.

Utilizando los archivos XML y el editor gráfico de Android Studio.



El editor de interfaces gráficas de Android

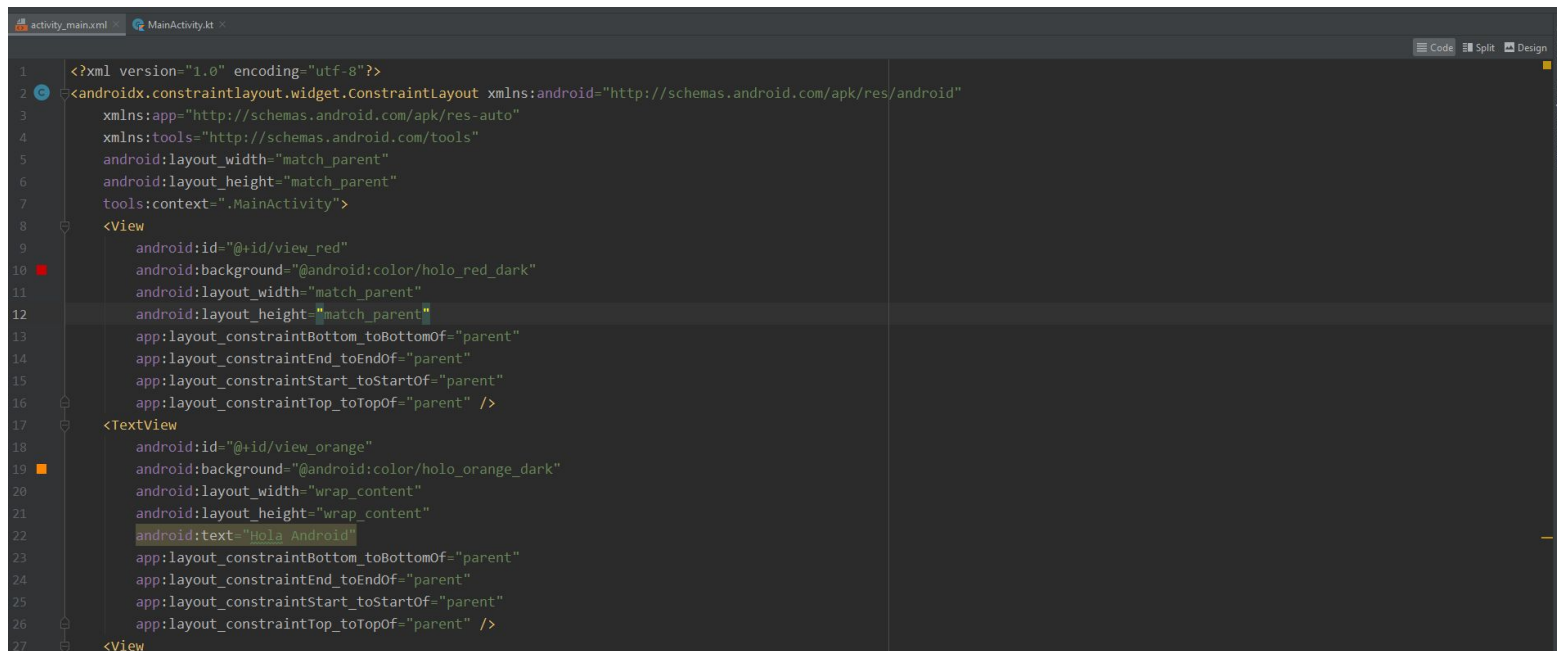
¿Dónde encontrarlo?

- app/res/layout

¿Qué vistas existen?

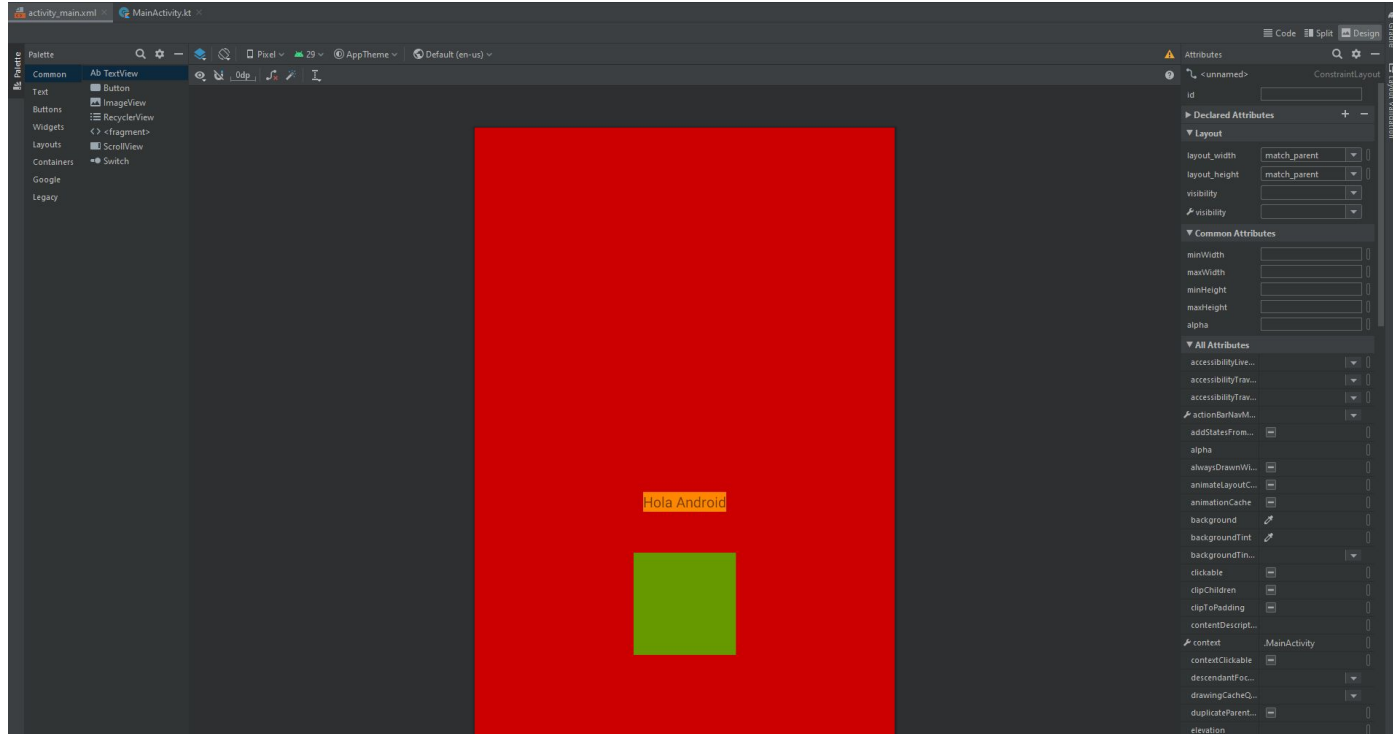
- “Code”
- “Slip”
- “Design”

Vista Code

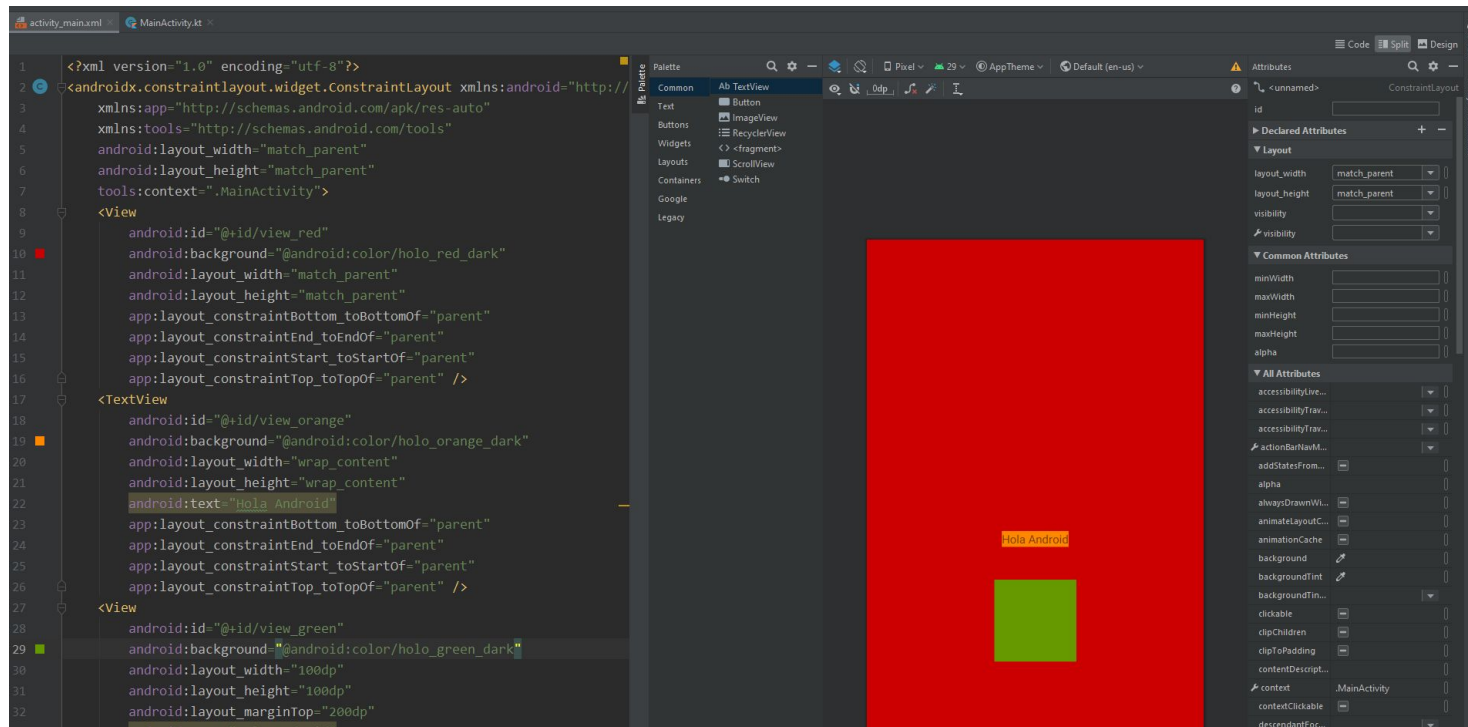


```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <View
        android:id="@+id/view_red"
        android:background="@android:color/holo_red_dark"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <TextView
        android:id="@+id/view_orange"
        android:background="@android:color/holo_orange_dark"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hola Android"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</View>
```

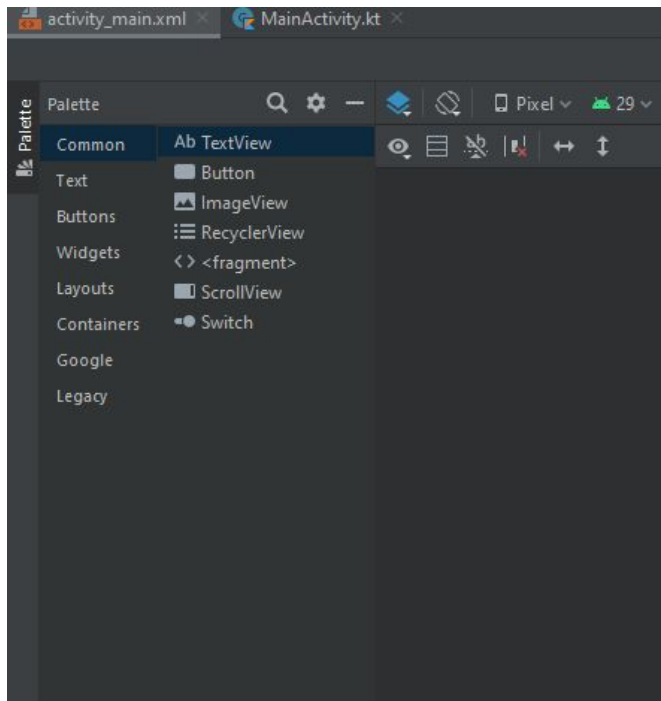
Vista Design



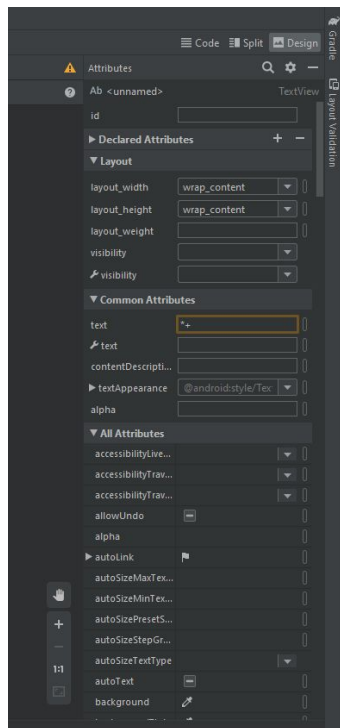
Vista Split



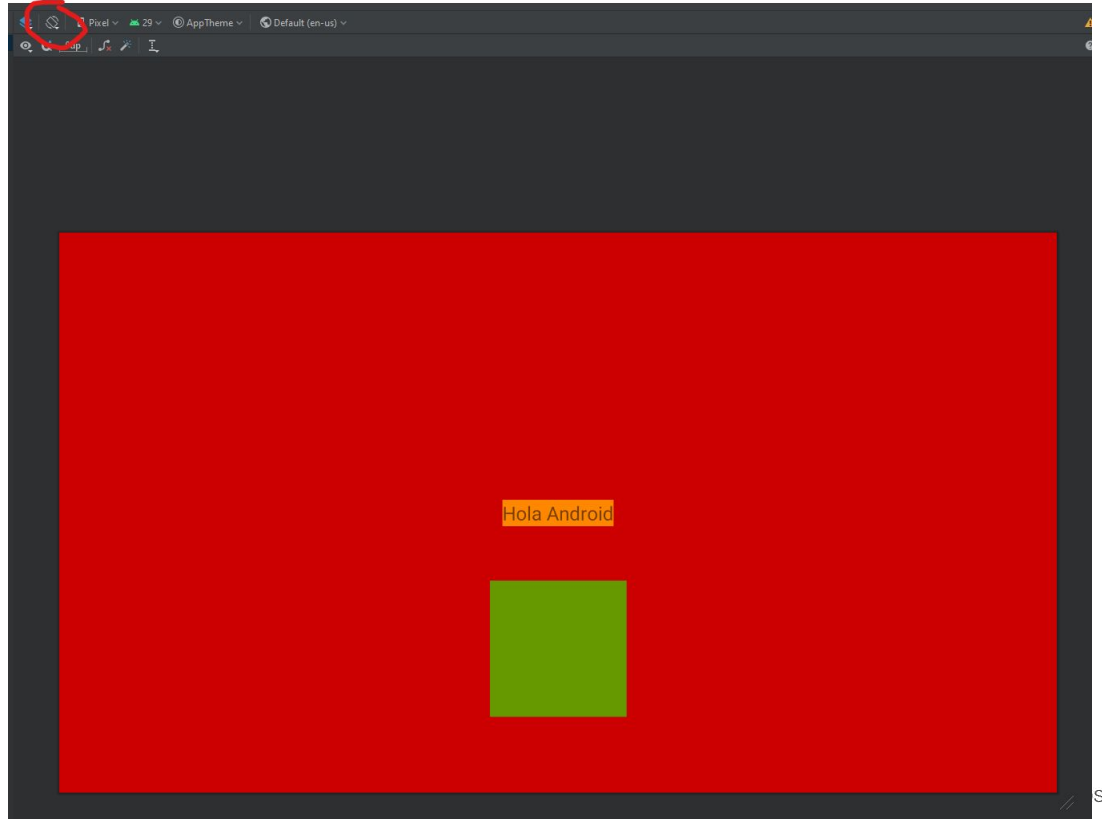
Vista **Design**: La paleta de diseños



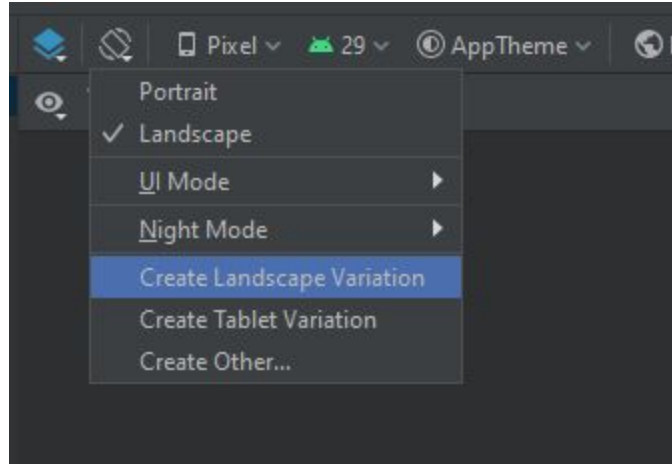
Vista **Design**: Los atributos



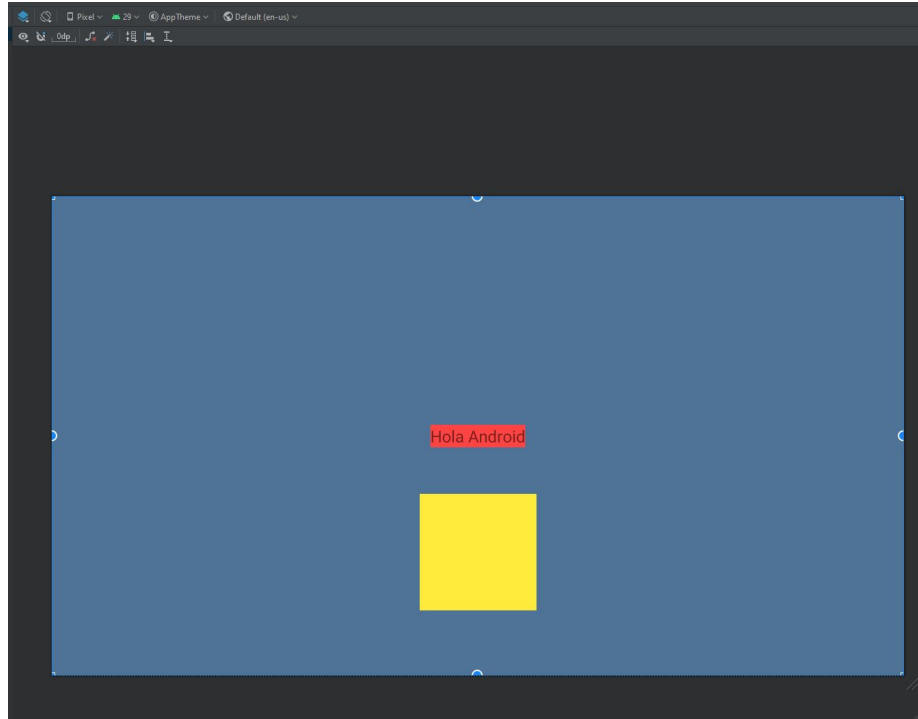
Simulando la Orientación



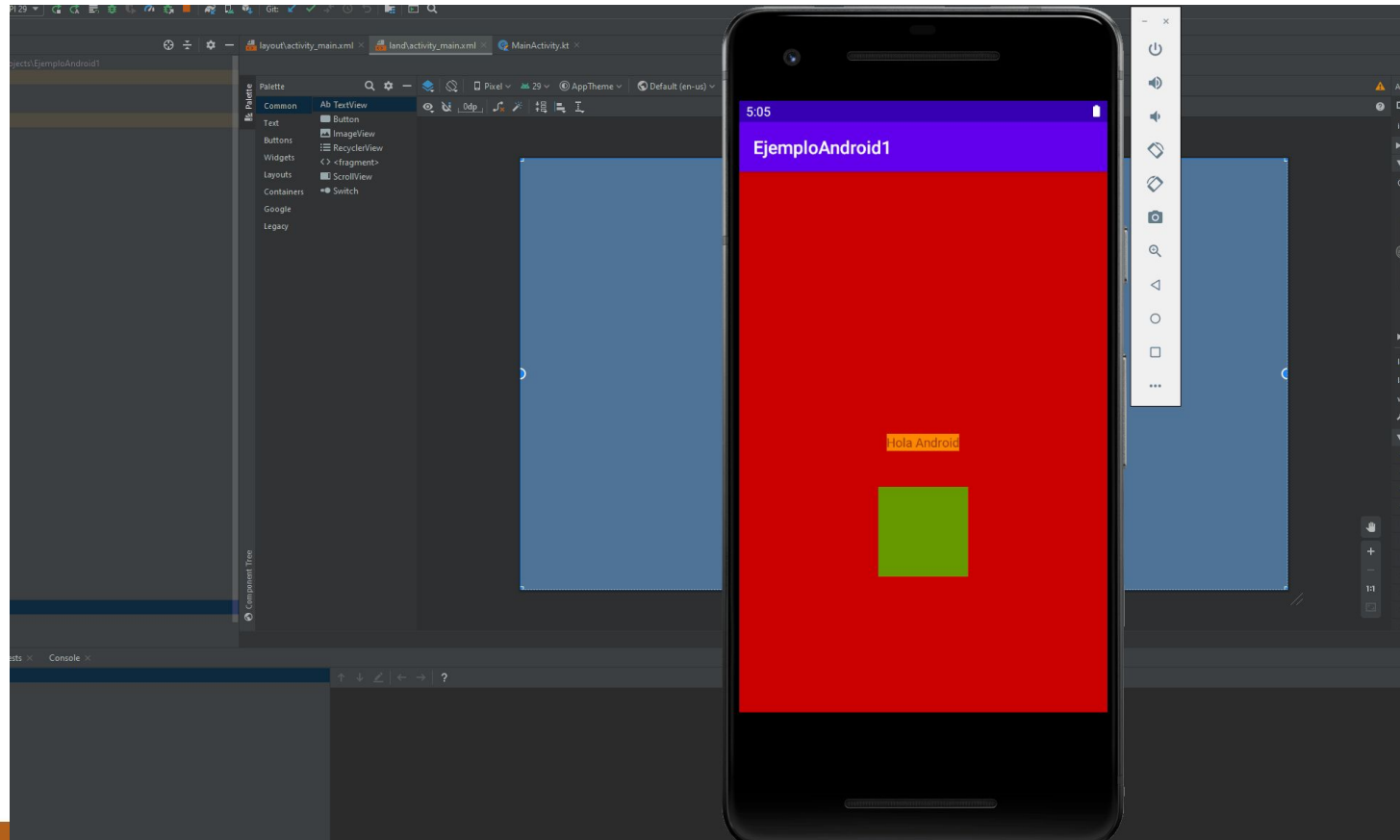
Modo apaisado I: Creación



Modo apaisado II: Modificación



Modo apaisado III: Ejemplo en vivo



Simulando las pantallas



Puedes configurar tu Simulador:

- Tamaño de pantalla
- Cantidad de Píxeles
- Versión de Android

IMPORTANTE:

DP - Es un concepto que nace para solucionar el problema de la gran variedad de pantallas disponibles en los dispositivos Android, de modo que una pulgada haya 160 “dp”, independientemente de la cantidad de píxeles existentes en esa pantalla.

[Documentación DP.](#)



Componentes Principales

Views

<https://developer.android.com/reference/android/view/View>

Layouts

Contienen otras views en su interior y organizan cómo se distribuyen.

Proyecto Resumen

Proyecto en GitHub

<https://github.com/KeepCodingMobile16/Fundamentos-Android/commit/b27766c84c009f21695a148a492257cba73ae964>



GitHub

Ejercicio Destacado



Crea 2 pantallas para tu aplicación, una que se aplique en la orientación vertical y otra para la orientación horizontal

Haz que el texto cambie de “Hello world” a “Vertical” cuando está en vertical, “Horizontal” cuando esté en horizontal



Layouts principales



Linear Layout

¿Se usa frecuentemente?

Sí, suele ser un layout de soporte bastante utilizado.

Filosofía: Colocar los elementos en línea uno detrás de otro

Tipos: **Vertical** y **horizontal**

Linear Layout tips

- Para diseños complejos suele ser necesario mezclar layouts verticales con layouts horizontales
- La propiedad weight es muy útil para distribuir los espacios respetando el tamaño mínimo de cada elemento
- Evita usar siempre que puedas valores fijos, salvo para imágenes y márgenes

Proyecto Resumen

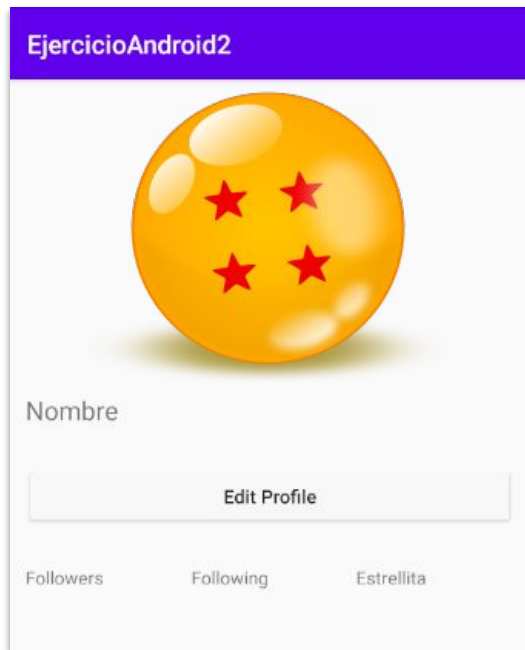
<https://github.com/KeepCodingMobile16/Fundamentos-Android/commit/2ddc2b33eaaacc7d68b86a9c86192929c16eed5b>



GitHub

Ejercicio

Utilizando LinearLayout, crea un diseño similar a:



Solución

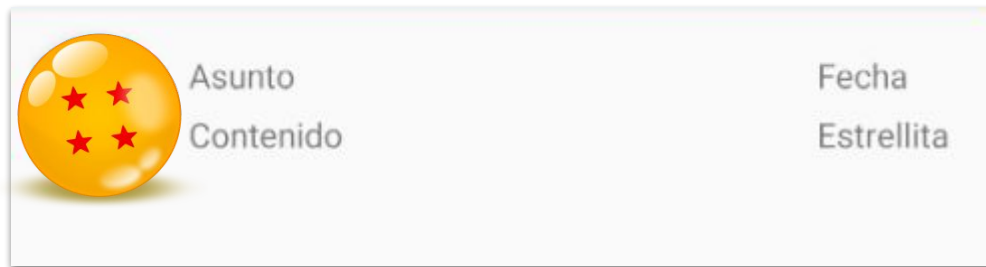
<https://github.com/KeepCodingMobile16/Fundamentos-Android/commit/d328651df82deccf5b09a669f7233e8e5669a1bf>



GitHub

Ejercicio

Utilizando LinearLayout, crea un diseño similar a:



Solución

Proyecto en GitHub

<https://github.com/KeepCodingMobile16/Fundamentos-Android/commit/b4354d07b46e25bfe8046e43ea7a97f7a649f254>



GitHub

Ejercicio Destacado



Utilizando un LinearLayout, cambias las pantallas que teníamos por los siguiente elementos:

- Título
- EditText: Usuario
- EditText: Contraseña
- Botón: Login

Caso horizontal ->



Caso vertical ->



Constraint Layout tip

¿Se usa frecuentemente?

Sí, suele ser la base de todos los layouts complejos.

Filosofía: Los elementos se colocan en relación a los otros elementos que conforman la interfaz y respecto a los límites del layout. Por esa razón son los elementos quienes dicen al layout dónde deben situarse.

Constraint Layout: Definición



Constraint Layout

Puedes encadenar elementos entre sí lanzando un doble enlace entre ellos

```
app:layout_constraintEnd_toEndOf="parent"
```

Dentro del espacio que tiene un elemento, puedes moverlo utilizando la propiedad bias (vertical u horizontal)

```
app:layout_constraintVertical_bias="0"
```

Cuando un elemento quieres que ocupe todo el espacio disponible limitado por sus constraints, utiliza 0dp

```
android:layout_height="0dp"
```

Constraint Layout

Las cadenas se realizan enlazando los elementos doblemente.

Ejemplo:

A se junta a B

B se junta a A

Dándole una distribución únicas

Proyecto Resumen

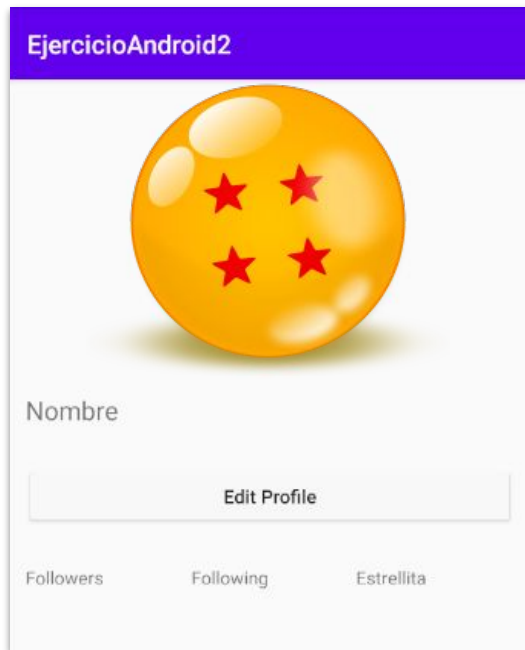
<https://github.com/KeepCodingMobile16/Fundamentos-Android/commit/85b144a0efaf0d796a5a56a31350ddf2913b51ba>



GitHub

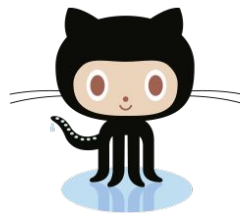
Ejercicio

Utilizando ConstraintLayout, crea un diseño similar a:



Solución

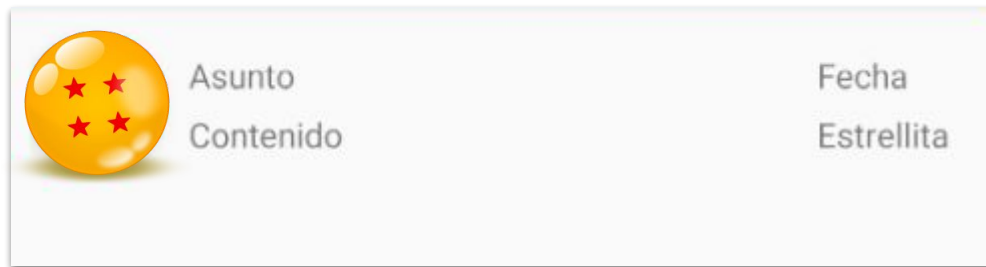
<https://github.com/KeepCodingMobile16/Fundamentos-Android/commit/7b0a048cde8c28d76f179cd2ed432871ae964238>



GitHub

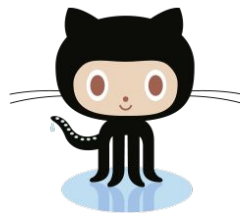
Ejercicio

Utilizando Constraint, crea un diseño similar a:



Solución

<https://github.com/KeepCodingMobile16/Fundamentos-Android/commit/dd2e0a719d1aead9126affc335a61ec0315d4a41>



GitHub

Ejercicio Destacado



Utilizando un Constraint Layout, cambia las pantallas que teníamos por los siguiente elementos:

- Título
- EditText: Usuario
- EditText: Contraseña
- Botón: Login

Caso horizontal ->



Caso vertical ->





Apoyo al Constraint Layout

Guías

Las guías son líneas verticales u horizontales.

Sirven para dividir el layout en subzonas de uso opcional.

```
<androidx.constraintlayout.widget.Guideline
    android:id="@+id/guideline2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintGuide_begin="74dp" />
```

Espacios

Los espacios son superficies bidimensionales que permiten delimitar zonas completas que utilizar como soporte para la organización efectiva del layout.

```
<Space  
    android:id="@+id/space"  
    android:layout_width="100dp"  
    android:layout_height="100dp" />
```

Proyecto Resumen

<https://github.com/KeepCodingMobile16/Fundamentos-Android/commit/691ecb50fe81c1b6532ff9b8b0d96caaaa9b7fa3>



GitHub



Layouts Legacy

Relative Layout

Filosofía: Los elementos se colocan en relación a los otros elementos que conforman la interfaz y respecto a los límites del layout.

Sustituto: Constraint Layout.

Principales problemas:

- Rendimiento.
- Adaptación a Android Studio y a las nuevas versiones de Android.
- Complejidad para realizar interfaces complejas.



List View

Filosofía: Los elementos se colocan uno detrás de otro en forma de lista

Sustituto: RecyclerView.

Principales problemas:

- Rendimiento.
 - Gestión de memoria.



Views

TextView I

Uso principal:

Mostrar información escrita al usuario.

```
<TextView
```

```
    android:id="@+id/textView"
```

```
    android:text="TextView" />
```



TextView II

Propiedades Principales

```
android:textSize="30sp"
```

```
android:textColor="@color/colorAccent"
```

```
android:textStyle="bold|italic"
```

```
android:typeface="monospace"
```

| EditText I

Uso principal:

Permitir interaccionar con el usuario a través de texto.

```
<EditText
```

```
    android:id="@+id/editTextTextPassword"
```

```
    android:ems="10"
```

```
    android:inputType="textPassword" />
```

EditText II

Uso principal:

Permitir interaccionar con el usuario a través de texto.

```
android:inputType="numberDecimal"
```

Existen numerosas posibilidades de inputType. Estas no solo personalizan el aspecto de la vista, sino que también modifican los valores válidos así como el teclado que muestra al usuario.

Button

Uso principal:

Capturar la pulsación del usuario

```
<Button
```

```
    android:id="@+id/button"
```

```
    android:text="Button" />
```

Radio Group

Uso principal:

Establece una zona dentro de la cual únicamente un radio button puede estar seleccionado al mismo tiempo.

```
<RadioGroup
```

```
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent" >
```

```
RADIO BUTTON AQUÍ
```

```
</RadioGroup>
```

Radio Button

Uso principal:

Permite escoger al usuario una opción de entre múltiples posibilidades excluyentes entre sí.

```
<RadioButton
```

```
    android:id="@+id/radioButton"
```

```
    android:text="RadioButton" />
```

Toggle Button

Uso principal:

Permite escoger al usuario entre dos valores.

```
<ToggleButton
```

```
    android:id="@+id/toggleButton"
```

```
    android:text="ToggleButton" />
```

Switch

Uso principal:

Permite escoger al usuario entre dos valores.

```
<Switch
```

```
    android:id="@+id/switch1"
```

```
    android:text="Switch" />
```

Floating Action Button

Uso principal:

Capturar la pulsación del usuario. Se diferencia del botón en que se sitúa, por defecto, superpuesto al contenido de la vista.

```
<com.google.android.material.floatingactionbutton.FloatingActionButton
```

```
    android:id="@+id/floatingActionButton"
```

```
    app:srcCompat="@android:drawable/ic_menu_add" />
```

Scroll View

El Scroll View es un tipo especial de vista dentro de la cual se le inserta un layout, permitiendo realizar scroll sobre el layout de su interior.

```
<ScrollView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
>
```

```
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
    >
```

Ejercicio

<https://github.com/KeepCodingMobile16/Fundamentos-Android/commit/948df1764bab61bf59f039a48c4061f23174465c>



GitHub

| Include

Es posible añadir un layout a otro layout.

```
<include layout="@layout/activity_menu"/>
```

Es también es posible modificar los atributos del layout para adaptarlo al su nueva ubicación.

Proyecto Resumen

<https://github.com/KeepCodingMobile16/Fundamentos-Android/commit/d4b56ecd31ffc1375edaae91250963bebd6c7fb0>



GitHub

Ejercicio I

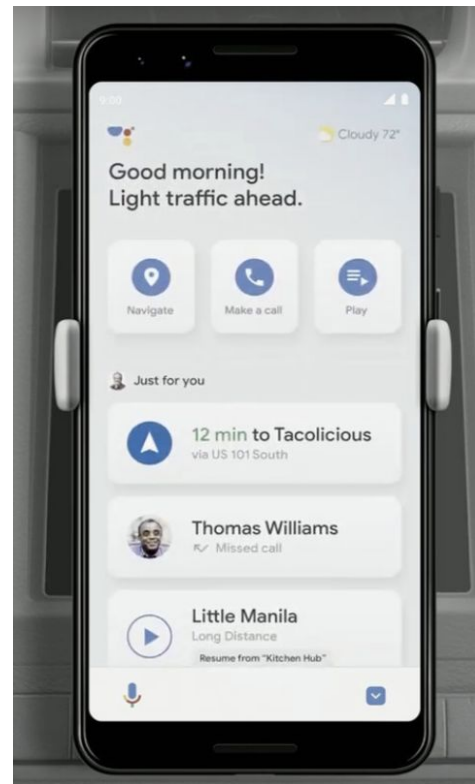
A partir de esta imagen, intenta diseñar un layout lo más parecida posible.

Adjunta a la entrega:

Capturas de pantalla / gifs de tu Android Studio para mostrar tu trabajo.

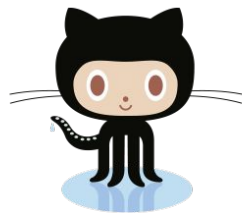
Adjunta el archivo XML completo.

Advertencia! Confirma conmigo si la imagen escogida es válida para la tarea.



Proyecto Resumen

<https://github.com/KeepCodingMobile16/Fundamentos-Android/commit/c03d0802c4a306c7a2c9cb2ab39dd0429ac8f661>



GitHub



Unidades de medida



Px

Es una unidad de medida de referencia en Android que se corresponde a un píxel de la pantalla.

Ventajas:

- Precisión

Desventajas:

- Dependencia extrema del hardware

Dp

Un móvil con una pantalla de 4 pulgadas de alto tendrá 640dp mientras que uno de 2 pulgadas 320dp.

Pregunta... ¿Y un móvil 6 pulgadas de alto cuantos dp tendrá?

Pregunta... ¿Y un móvil 6 pulgadas de alto con resolución 4K cuantos dp tendrá?



Ejercicio II

Escoge una aplicación famosa y realiza una captura de pantalla de alguno de sus pantallas.

A partir de esa imagen, intenta diseñar un layout lo más parecida posible.

Adjunta a la entrega:

Capturas de pantalla / gifs de tu Android Studio para mostrar tu trabajo.

Adjunta el archivo XML completo.

Advertencia! Confirma conmigo si la imagen escogida es válida para la tarea.



Gestión de recursos



Strings

Las strings en Android se deben añadir en:
`res/values/string.xml`

Además es posible gestionar la traducción de String mediante archivos por idioma

Organización de imágenes I

Las imágenes en Android se deben añadir en:

- Si estás en la vista Android.

```
res>drawable
```

*Al pegar una imagen te aparecerá un pop-up para seleccionar. Desde ahí selecciona xhdpi.

- Si estás en la vista Project:

```
res>drawable>mipmap-xhdpi
```

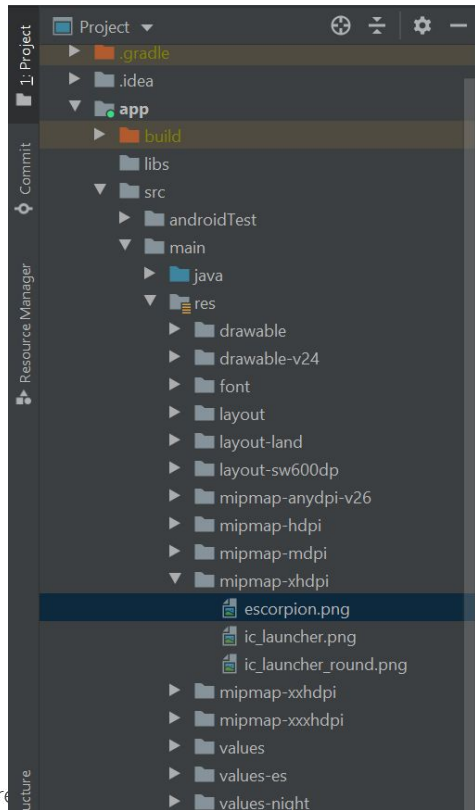
Organización de imágenes II

Es importante escoger la carpeta adecuada

(Opción recomendable)

Si estás en la **vista Project** las imágenes en Android se deben añadir en:

```
res>mipmap-xhdpi
```

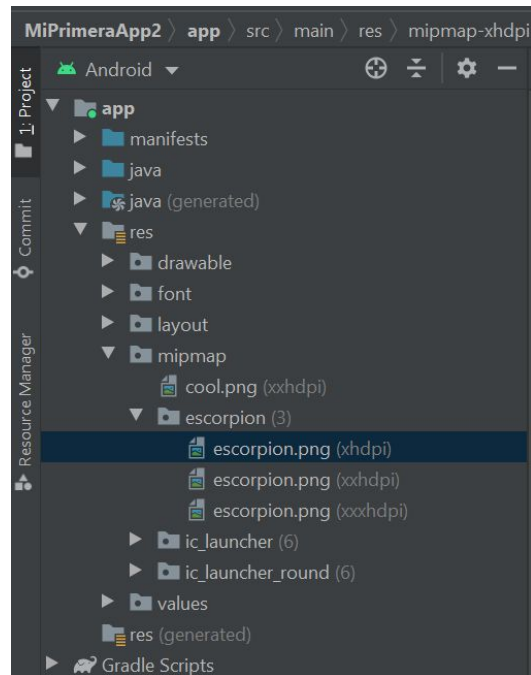
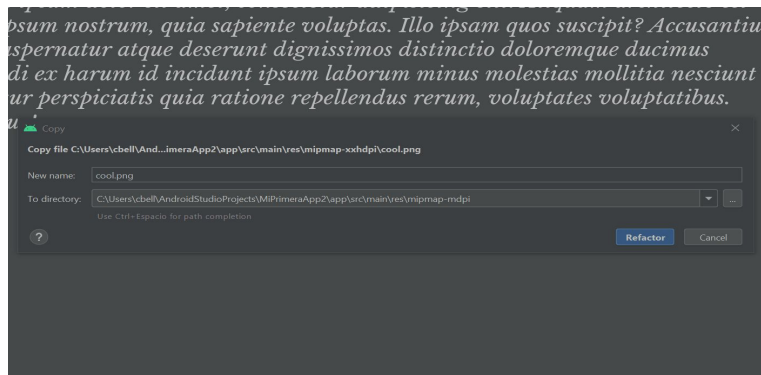


Organización de imágenes III

Si estás en la **vista Android** las imágenes en Android se deben añadir en:

`res>mipmap`

Te saldrá un mensaje en el que te pedirá la ubicación. Es importante seleccionar la opción que termina en “-xhdpi”



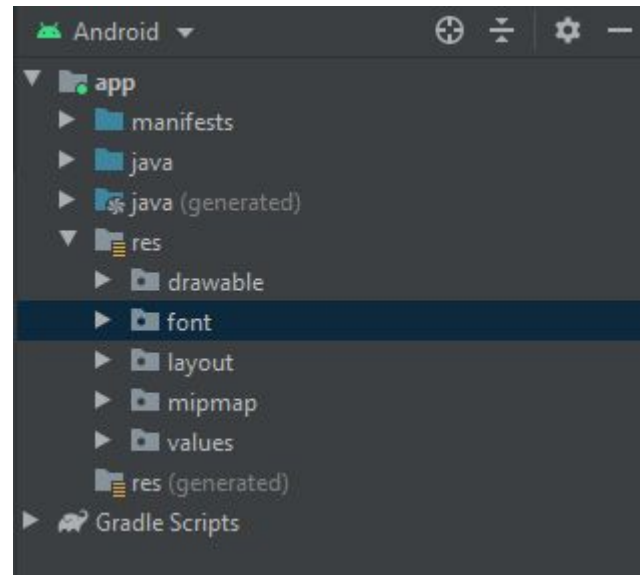
Fuentes

Crea una nueva carpeta en res y llámala font

res>mipmap-xhdpi

Arrastra a esa nueva carpeta tu fuente en formato “.tff”

Podrás acceder a ella mediante R.



Ejercicio Destacado



- Traduce tu aplicación al Español / Inglés
- Añade un CheckBox “Recordar usuario”
- Añade una fuente de tu elección <https://www.1001freefonts.com/>
- Añade la siguiente imagen como fondo de pantalla a tu app
- Junta ambos editText dentro de alguna estructura y ponle un fondo redondeado

Existen varias formas de hacerlo:

- Mediante un layout
- Mediante una CardView
- Mediante un drawable

En ambos casos necesitas usar la propiedad: CornerRadiious

- Cambia el botón por un MaterialButton y mira que diferencias tiene respecto a un button clásico





Fin