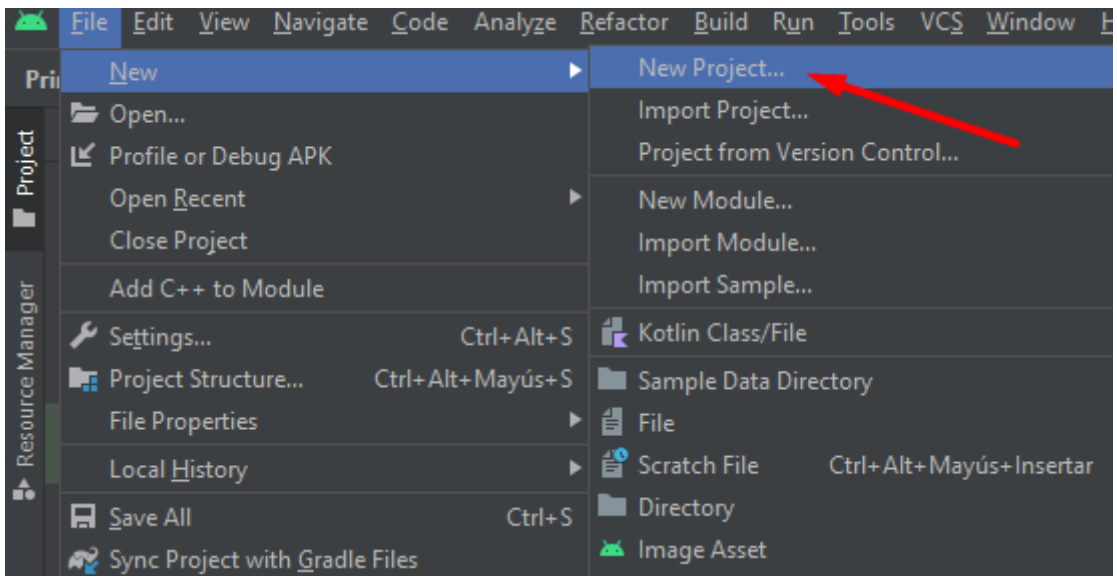
Primer Proyecto Android

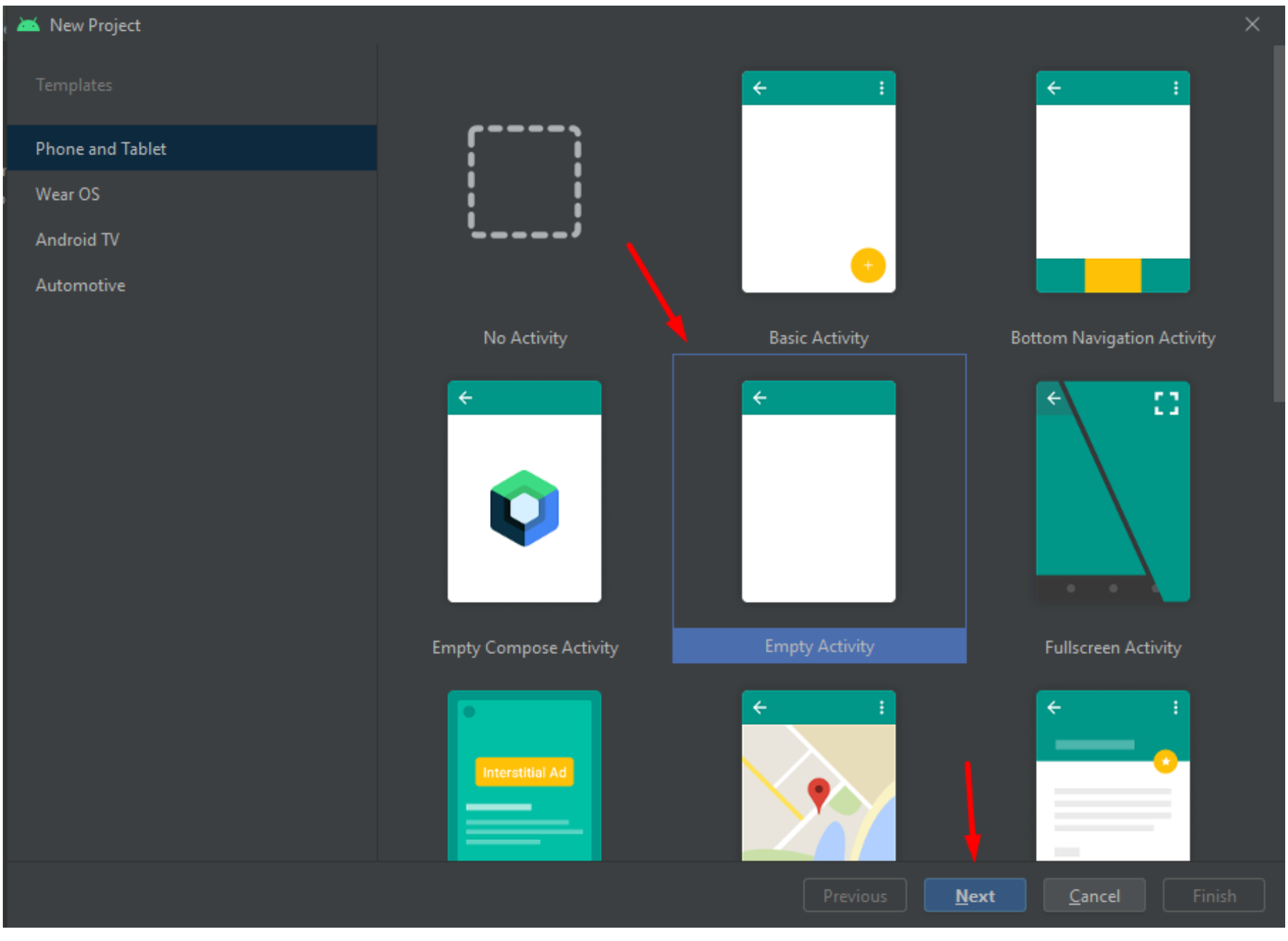
En la siguiente guía veremos la creación de una aplicación simple con captura de datos y eventos de bóton.

Creación del Proyecto.

Al iniciar Android Studio, si esta la primera vez, la herramienta nos da la posibilidad de crear un proyecto nuevo, sino entonces desde la barra de herramientas seleccionamos la new Proyect…

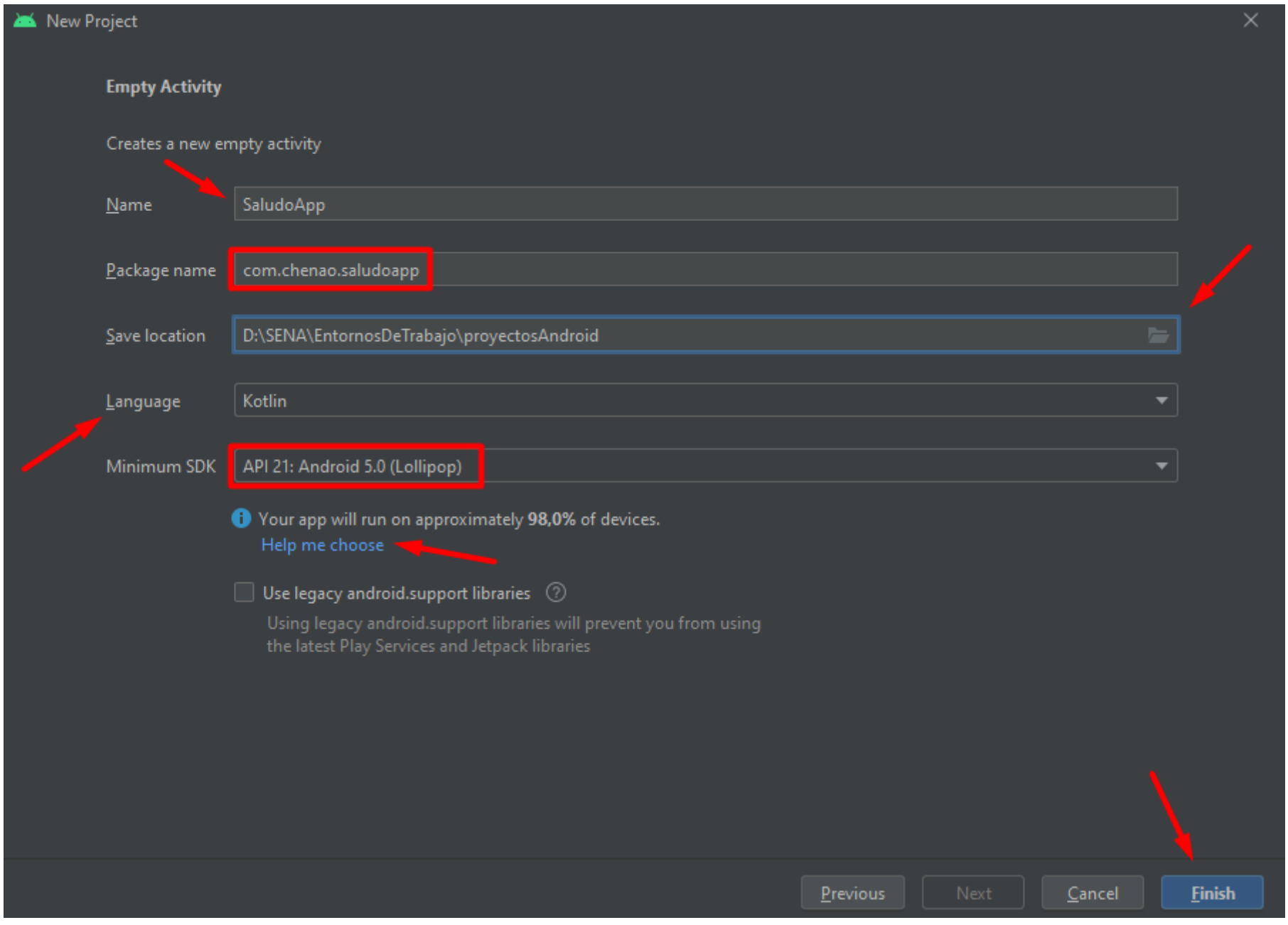


Al hacerlo se carga la ventana donde elegimos el tipo de proyecto que queremos, en este caso seleccionamos Empty Activity y damos clic en Next



Posteriormente se carga la ventana de creación, aquí definimos el nombre del proyecto, el nombre del paquete (muy importante ya que es el que va a referenciar el paquete y la empresa cuando se publique en la tienda de aplicaciones) la ruta donde quedara almacenado el proyecto, el lenguaje de programación y el mínimo SDK con el que vamos a trabajar.

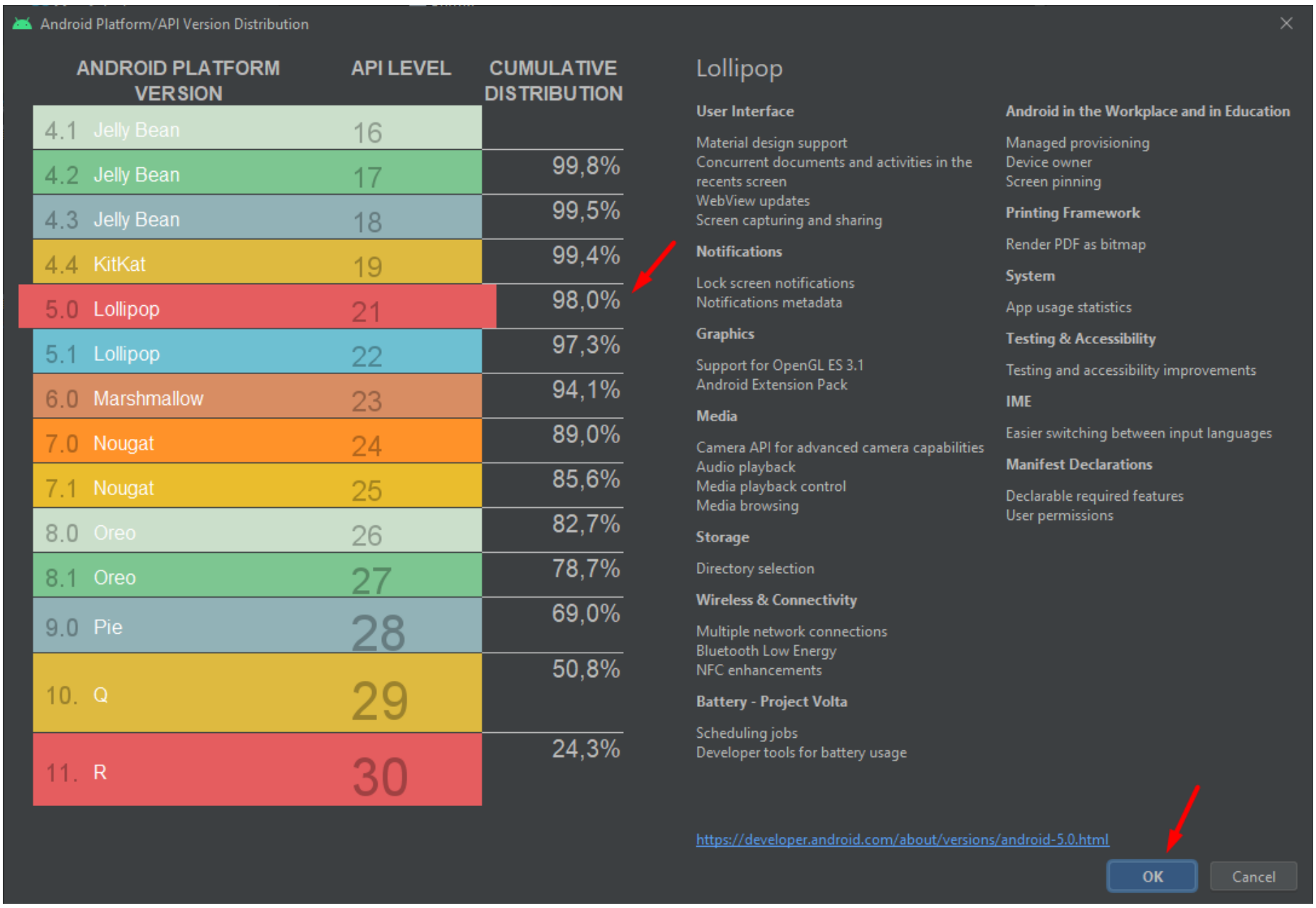
En caso que no sepamos cual es el mínimo SDK entonces podemos dar clic en “Help me choose”



Al hacerlo se carga una ventana donde podemos ver las versiones de android de las más usadas hasta las más recientes así como los detalles de cada versión.

en la columna “CUMULATIVE DISTRIBUTION” se muestra el porcentaje de uso de la versión de android en los dispositivos actuales, estos datos se toman basado en las estadísticas recopiladas por la plataforma, seleccionamos la versión a trabajar y damos clic en OK.

Nota: Al definir que versión es la que queremos como mínimo SDK estamos restringiendo el uso de la App desde la versión seleccionada en adelante, para nuestro ejemplo quiere decir que solo podemos instalar la app en versiones 5.0 en adelante, no se podrá instalar en la 4.4 por ejemplo.

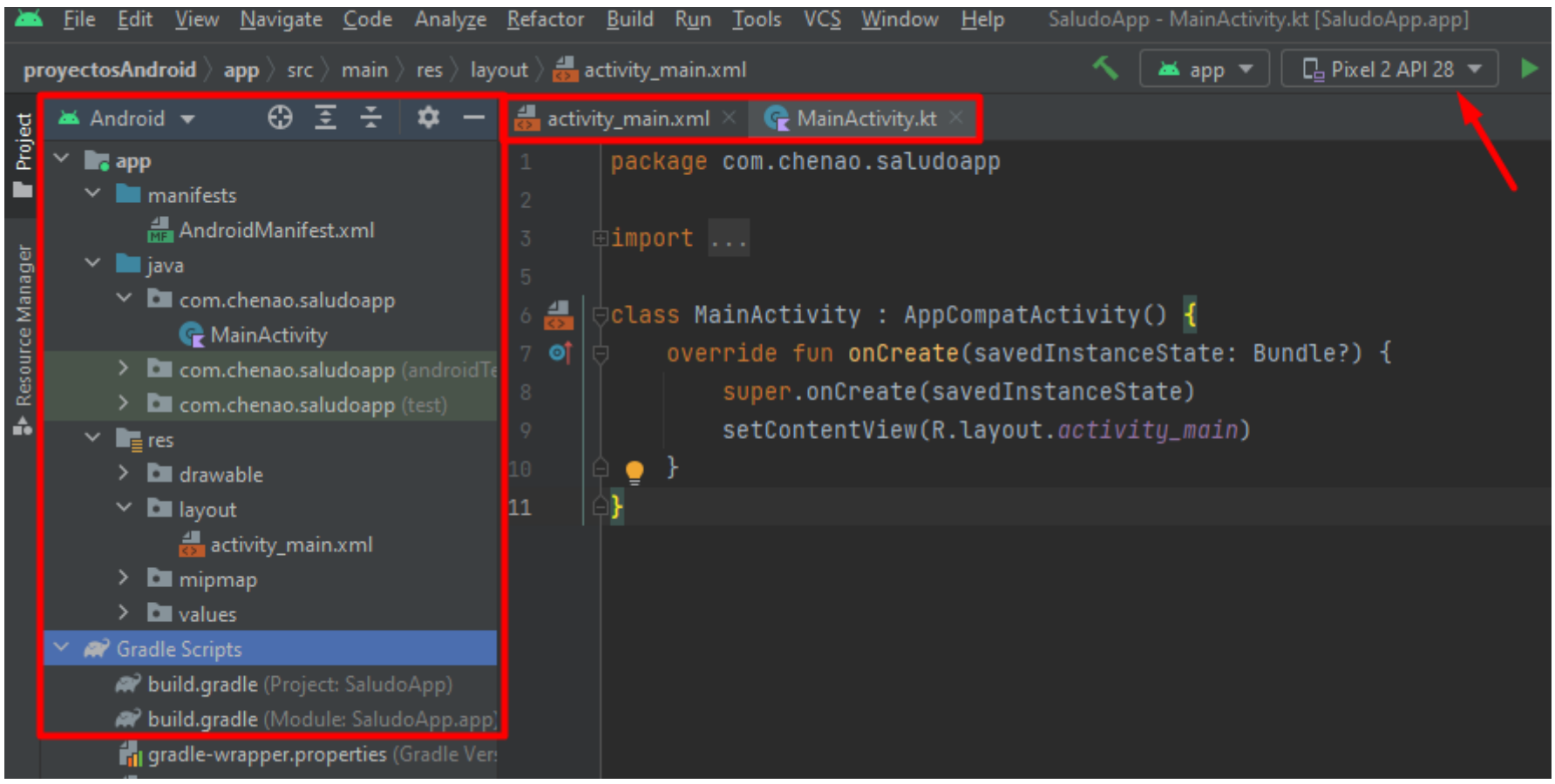


Estructura Básica.

Después de un rato se construye el proyecto, es muy importante que tengamos acceso a internet ya que la herramienta empieza a descargar las dependencias necesarias para la construcción del mismo, al final podemos ver la estructura del proyecto, el archivo **activity_main.xml** y la clase **MainActivity.kt** con el código por defecto.

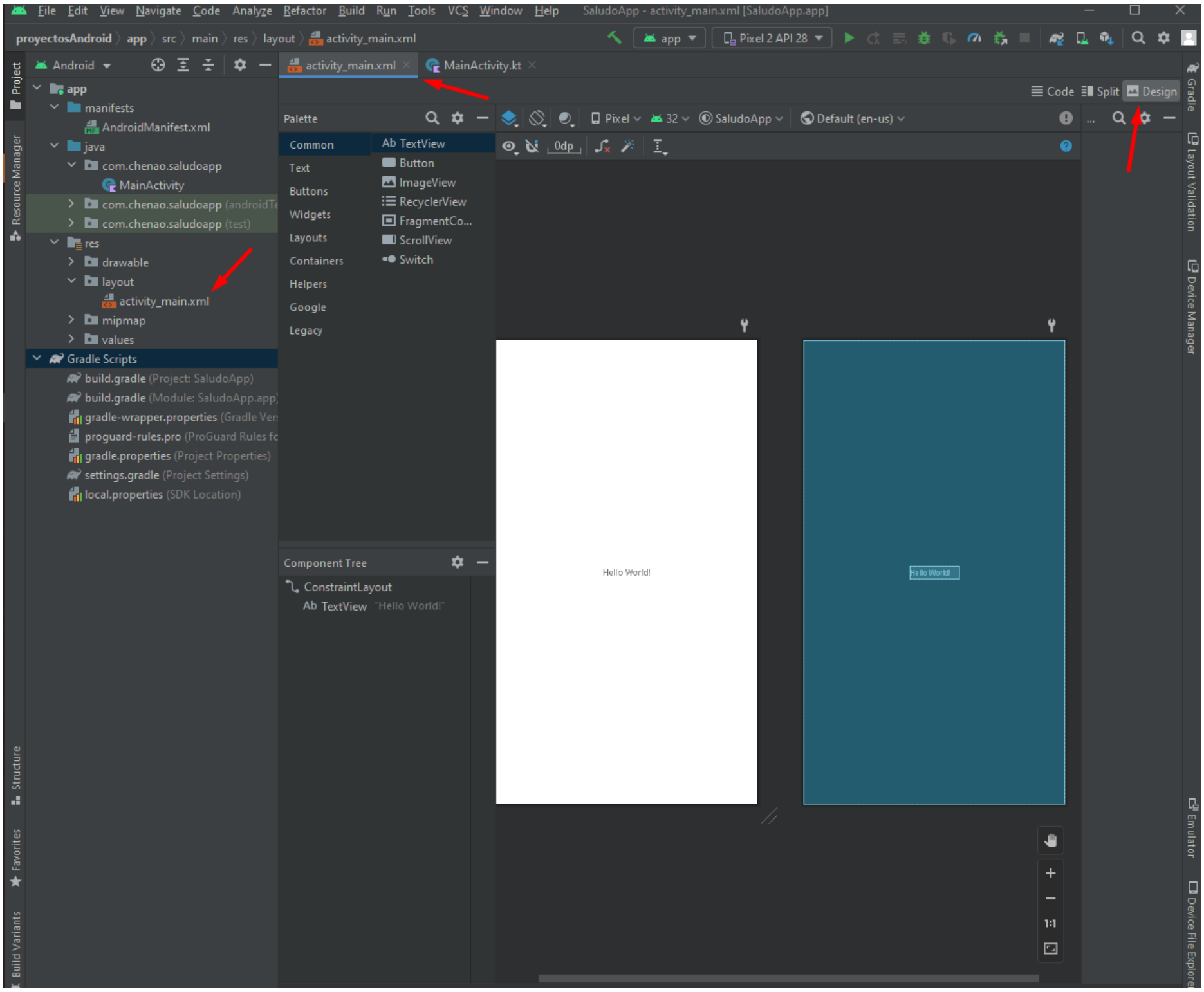
También podemos ver archivos como el AndroidManifest.xml, y el **build.gradle** entre otros que revisaremos más adelante.

Adicional, si ya se ha creado un emulador entonces se cargará automáticamente el emulador disponible o el dispositivo físico configurado.



Creación Interfaz Gráfica.

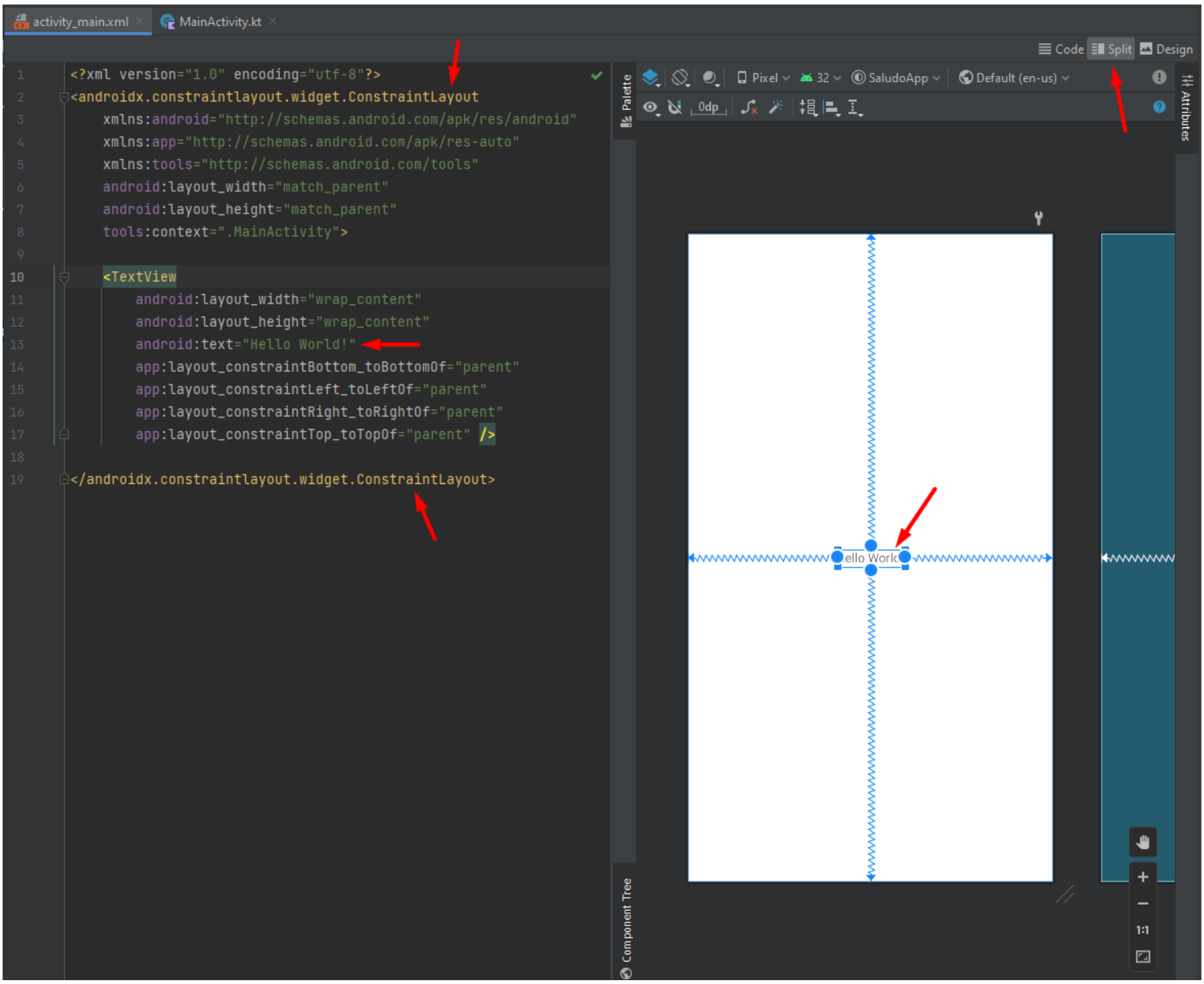
Si ingresamos al archivo **activity_main.xml** veremos un cliente gráfico desde donde podemos crear nuestras pantallas, aquí veremos la pestaña “**Desing**” la que nos permite arrastrar y soltar componentes, así como modificar los componentes mediante sus propiedades.



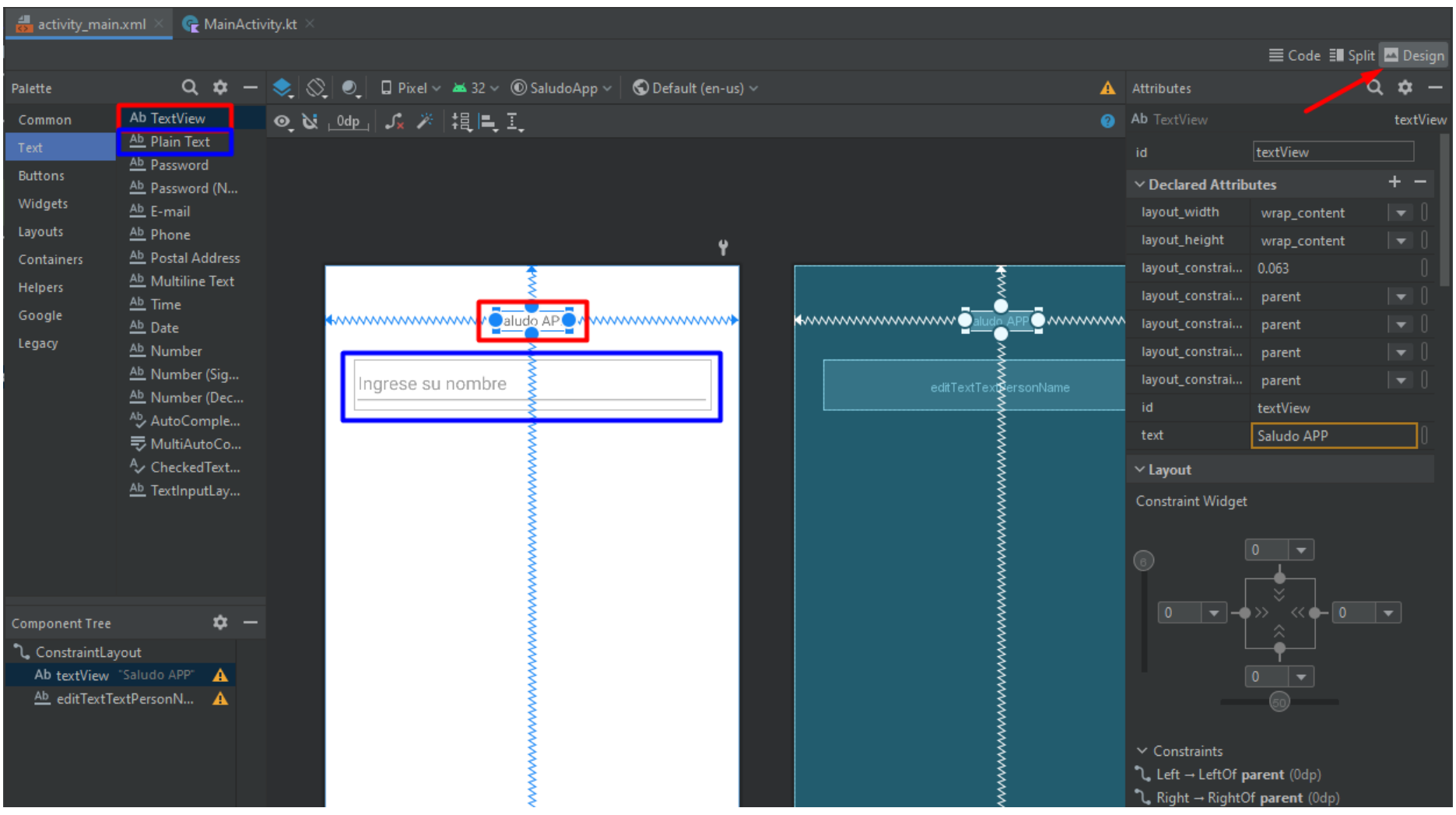
También podemos encontrar la pestaña “**Split**” que permite dividir la pantalla en la sección de código xml y la sección de visualización de pantalla, aquí podemos crear los componentes mediante código aunque podemos alternar en ambas vistas.

También podríamos trabajar en la pestaña “**Code**” la que nos brinda el trabajo solo con código, pero la vista “Split” es más recomendada.

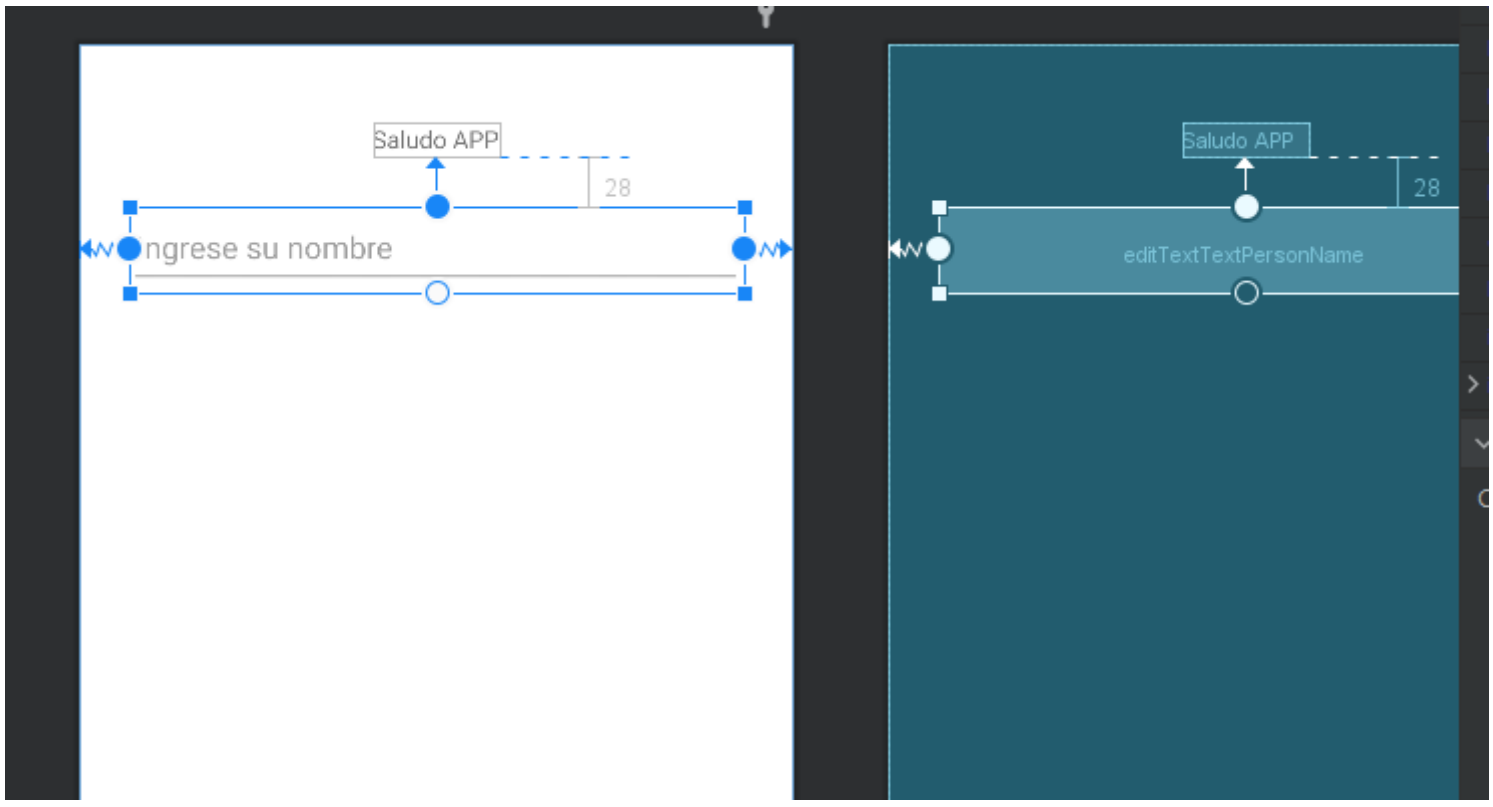
En esta vista podemos conocer la estructura de la pantalla y los componentes que la componen así como sus propiedades, notemos que por ejemplo para el “**Hello World!**” que se crea por defecto, se usa un componente **<TextView/>** y este se encuentra en un gestor de contenido llamado **ConstraintLayout** que permite ubicar los componentes con posiciones relativas a la pantalla u otros componentes.



Si pasamos a la vista de diseño, podemos modificar nuestra pantalla a nuestro gusto, en este caso crearemos la siguiente interfaz.

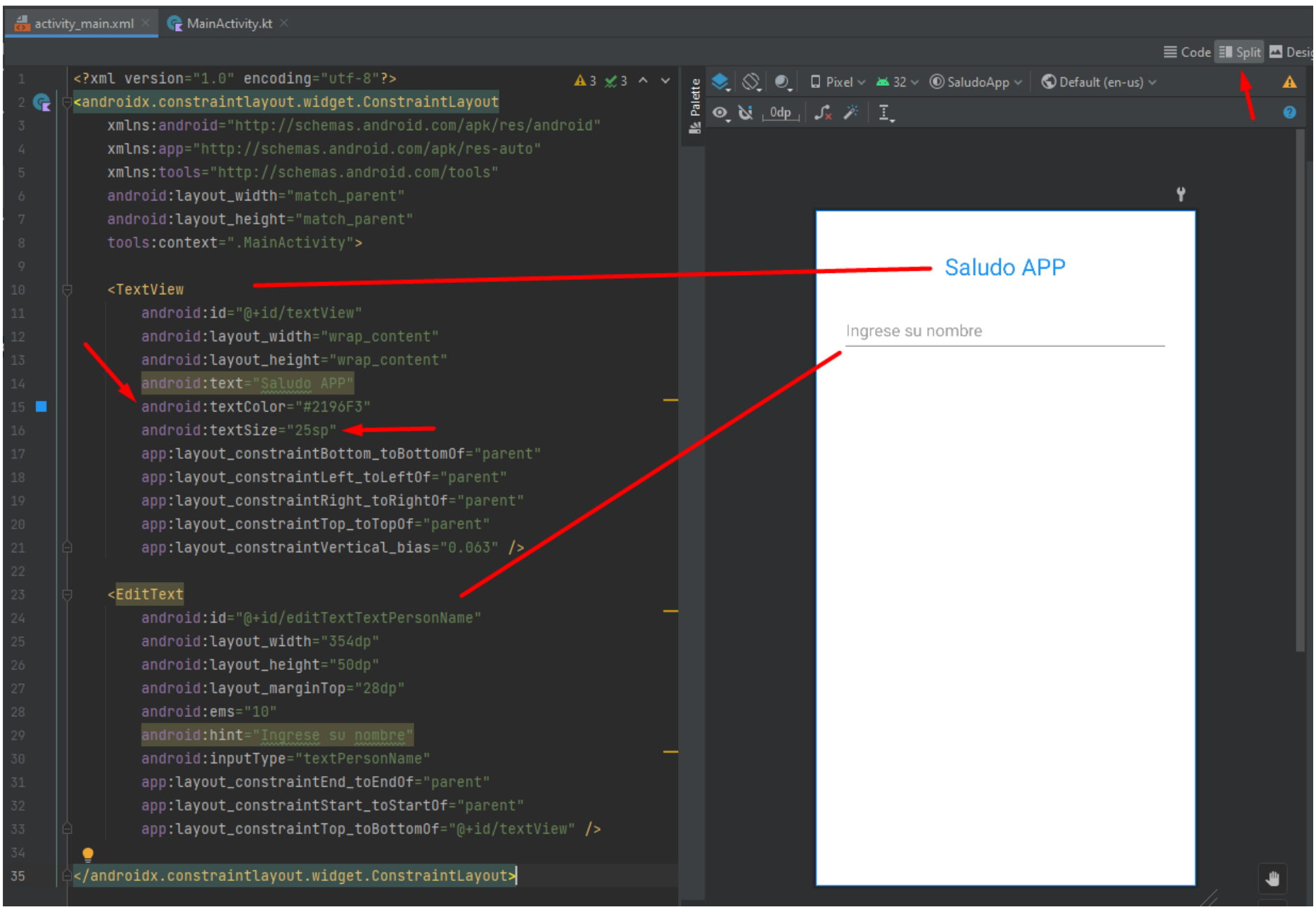


Como vemos agregamos un componente PlainText qué en realidad corresponde a un campo de edición o <EditText/> el cual puede tener propiedades como texto plano, password, Teléfono, Correo entre otros, así mismo el Layout definido nos permite modificar el tamaño de los componentes y asignarle una posición relativa, en este caso el campo de texto tiene una posición relativa al textView del titulo.

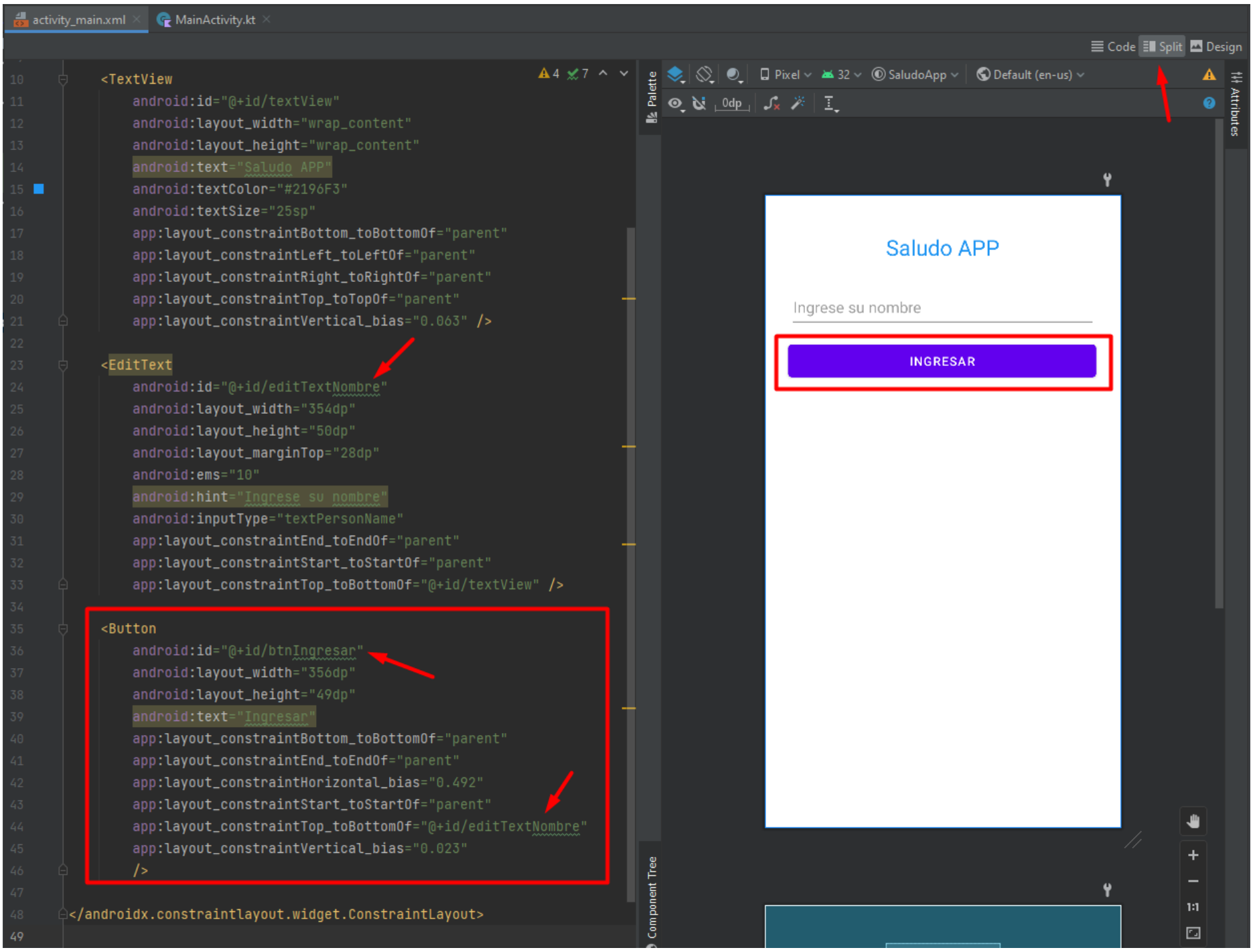


Si pasamos a la vista Split vemos entonces el código xml generado y desde aquí podemos modificar las propiedades de forma manual, en este caso se agregó un color al titulo y un tamaño de 25sp, nótese que cada componente tiene una propiedad id, así como propiedades para controlar la ubicación del componente en la pantalla.

Encontramos otros componentes como el **hint** que corresponde a un **placeHolder** el cual se muestra mientras el campo esté vacío, se podrían agregar de la misma manera otras propiedades para estilizar el componente.



Ahora se agrega un componente de tipo <Button/> el cual también tiene la propiedad id la cual es relativa al componente del campo de texto, para este caso modificamos el identificador.



Lógica de Aplicación.

Para realizar la lógica y procesamiento de información, vamos a la clase MainActivity.kt, desde allí obtenemos los componentes gráficos por medio de su id utilizando el método findViewById()

Agregamos el evento clic del botón y creamos una función llamada onClick() (Puede ser llamada de cualquier forma)

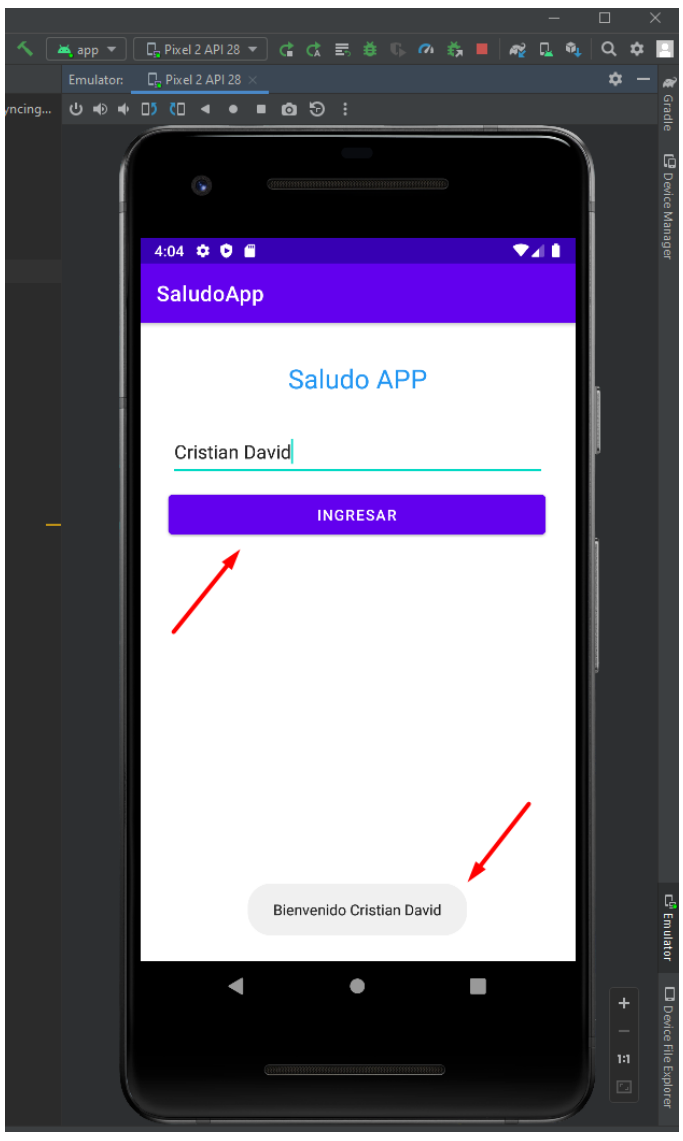
En esta función se agrega la lógica para capturar el valor del campo de texto el cual se referencia igual que el botón pero en este caso creando un componente de tipo EditText que referencia el id del campo de texto.

Posteriormente se obtiene el valor del campo y se agrega a una variable String que luego se muestra en un componente de tipo Toast el cual permite mostrar un mensaje temporal en pantalla.

```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help   SaludoApp - MainActivity.kt [SaludoApp.app]
proyectosAndroid 〉 app 〉 src 〉 main 〉 java 〉 com 〉 chenao 〉 saludoapp 〉 MainActivity
activity_main.xml × | MainActivity.kt ×
```

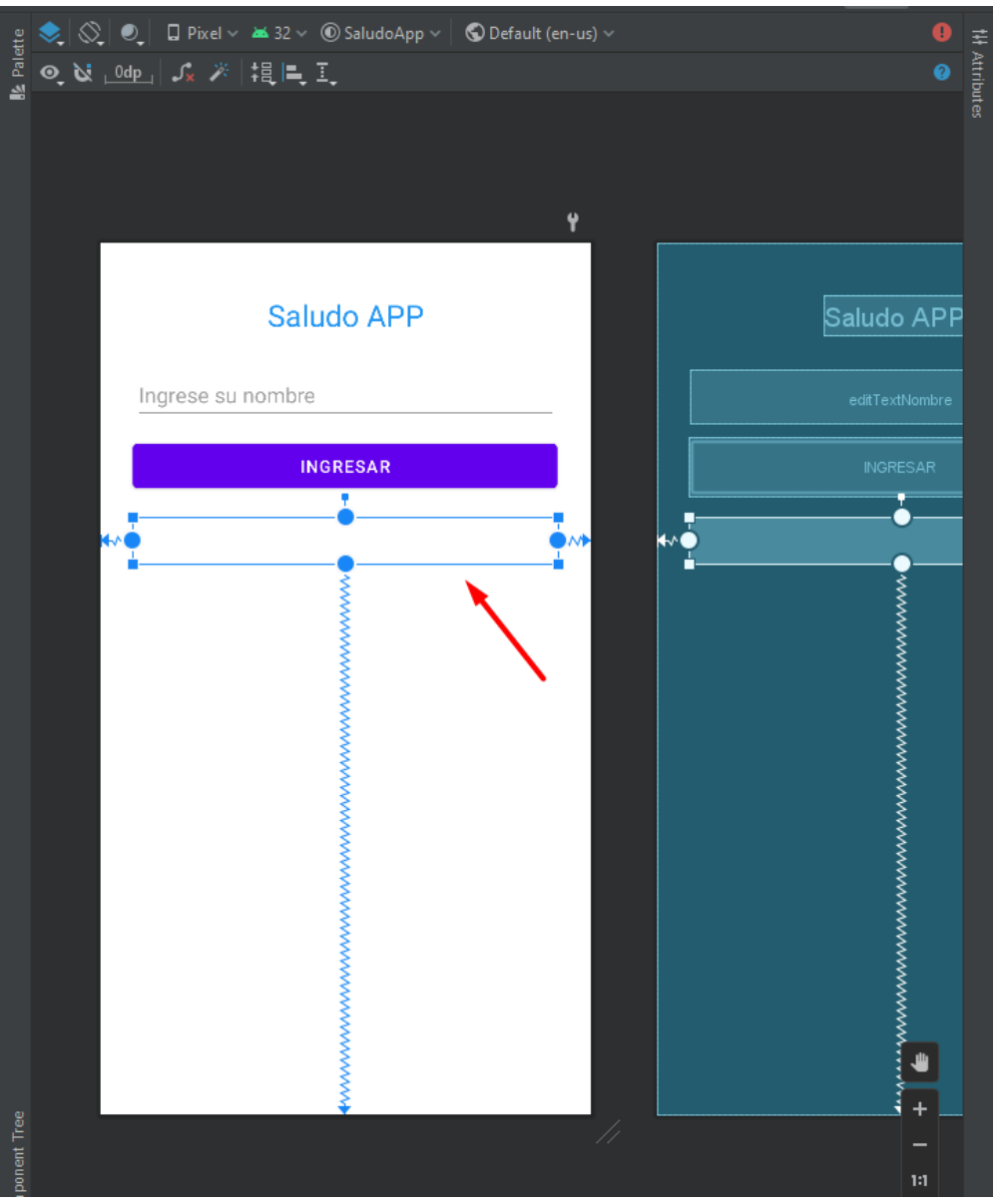
```kotlin
package com.chenao.saludoapp

import ...

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val miBoton:Button= findViewById(R.id.btnIngresar)
        miBoton.setOnClickListener{onClick()}

    }

    private fun onClick() {
        val campoTexto:EditText=findViewById(R.id.editTextNombre)
        var nombre:String=campoTexto.text.toString()
        Toast.makeText( context: this, text: "Bienvenido $nombre",Toast.LENGTH_LONG).show()
    }
}
```

Cuando ejecutamos la aplicación se lanza el emulador y podemos probar el sistema agregando el nombre del usuario y presionando el botón “ingresar”, vemos entonces como se muestra el Toast el cual dura algunos segundos.



Luego de esto modificamos la pantalla agregando otro componente TextView sin texto, el cual servirá para mostrar la información del nombre ingresado pero ahora directamente en la pantalla.

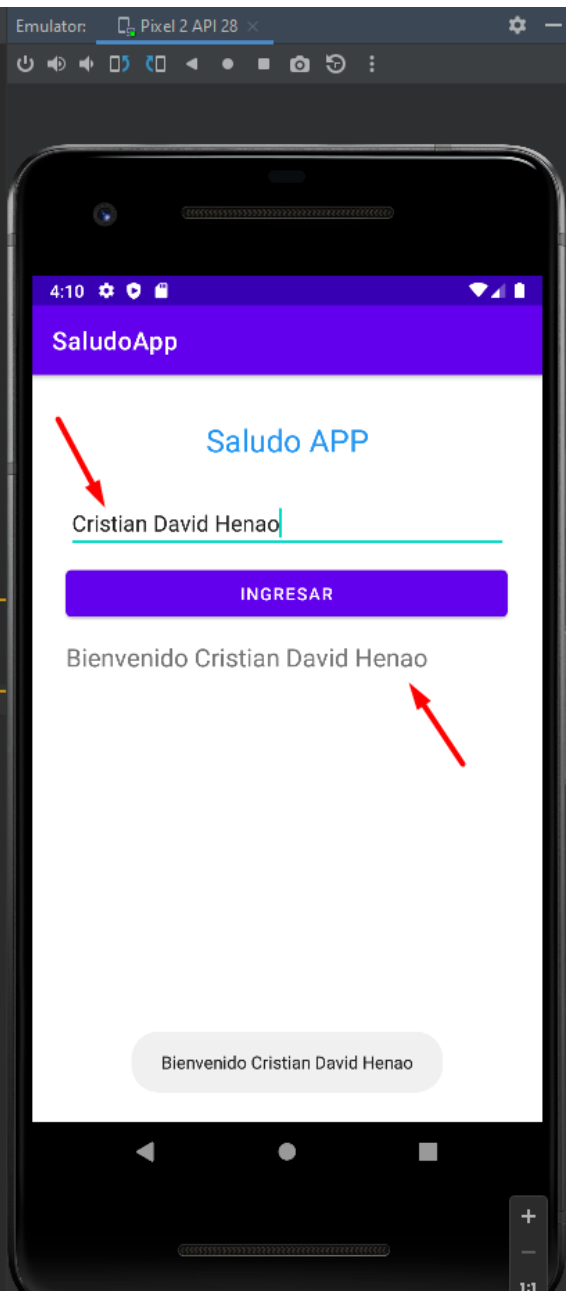
```
        android:layout_marginTop="28dp"
        android:ems="10"
        android:hint="Ingrese su nombre"
        android:inputType="textPersonName"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView" />

    <Button
        android:id="@+id/btnIngresar"
        android:layout_width="356dp"
        android:layout_height="49dp"
        android:text="Ingresar"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.492"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/editTextNombre"
        app:layout_constraintVertical_bias="0.023" />

    <TextView
        android:id="@+id/txtResultado"
        android:layout_width="356dp"
        android:layout_height="39dp"
        android:text=" "
        android:textSize="20sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.49"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/btnIngresar"
        app:layout_constraintVertical_bias="0.037" />

</androidx.constraintlayout.widget.ConstraintLayout>
```



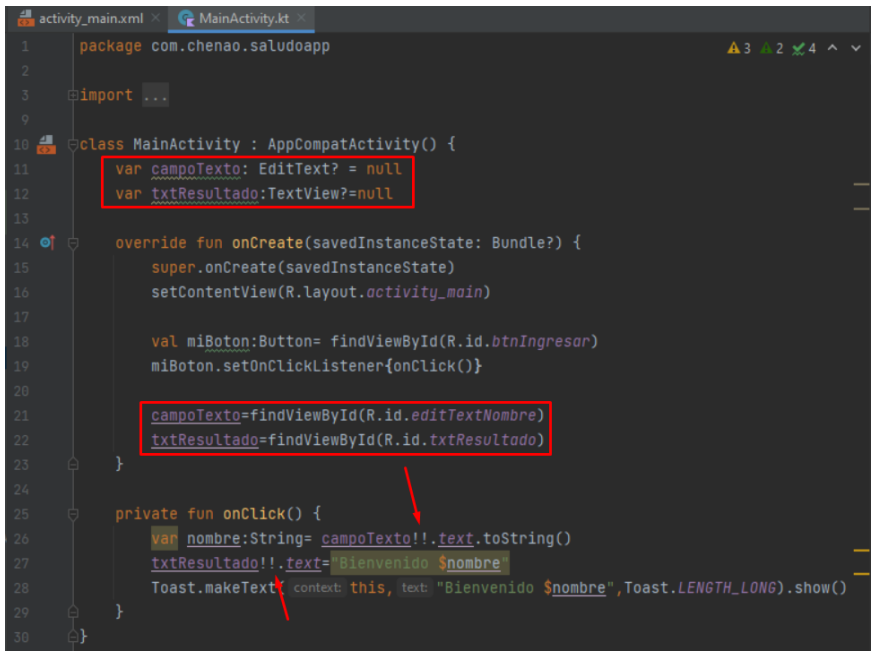
Para realizar esta lógica, regresamos a la clase MainActivity.kt y en la función referenciamos el componente EditText con el id asignado y luego enviamos el mensaje que queremos presentar.

Posteriormente lanzamos otra vez la aplicación y tenemos el siguiente resultado.

```
package com.chenao.saludoapp

import ...

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val miBoton:Button= findViewById(R.id.btnIngresar)
        miBoton.setOnClickListener{onClick()}

    }

    private fun onClick() {
        val campoTexto:EditText=findViewById(R.id.editTextNombre)
        var nombre:String=campoTexto.text.toString()

        val txtResultado:TextView=findViewById(R.id.txtResultado)
        txtResultado.text="Bienvenido $nombre"

        Toast.makeText( context: this, text: "Bienvenido $nombre",Toast.LENGTH_LONG).show()
    }
}
```



Otra alternativa para la captura de los datos es declarando los componentes de forma global evitando así tener que declararlos en cada función:



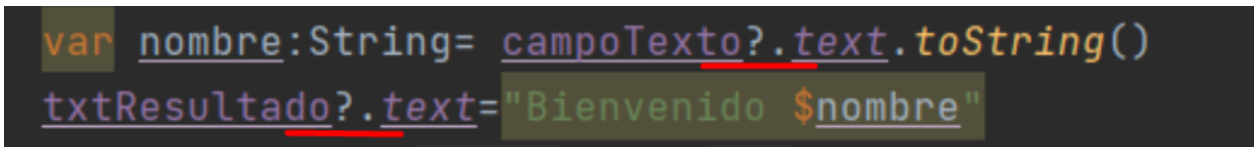
Importante usar !! para evitar que se puedan tener referencias nulas.

Nota:

En las líneas 11 y 12 se utiliza nullsafety, esta es una característica de kotlin que permite controlar los valores nulos dentro de la aplicación, evitando que cargue el popular NullPointerException, directamente no podemos asignar valores nulos, pero de esta manera cuando indicamos EditText? estamos diciéndole al compilador que ese campo podría tener una referencia null y que permita procesarla.

en la línea 26 y 27 se hace uso de !! para indicar al compilador que no compruebe si el valor es null y por lo tanto puede compilar sin problemas pero en caso de ser null, lanzar la excepción en consola, esto es en caso de que sepamos que puede llegar un valor null y aún así lo queremos procesar.

Otra alternativa recomendada seria realizar llamadas seguras, así cuando se intente llamar al objeto, si este es null, no se lanzará la excepción sino que se almacenará el valor null sin terminar el proceso, por esta razón la variable tiene que ser nullable para que compile correctamente.



Actividad.

Modifique la aplicación agregando los siguientes datos:

- Dirección
- Teléfono
- E-mail
- Documento
- Edad

Cuando presione el botón deberá presentar toda la información en pantalla y en el toast, así como un mensaje que indique si la persona es Infante, Adolescente, Adulto o Adulto Mayor, según la siguiente tabla.

Edad	Resultado

Edad	Resultado
de 0 a 15	Infante
de 15 a 18	Adolescente
de 18 a 65	Adulto
de 65 en adelante	Adulto Mayor

Haga uso de las propiedades según el tipo de dato definido.