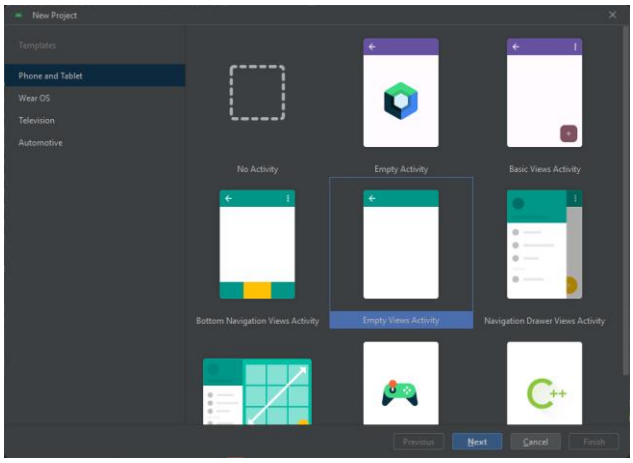
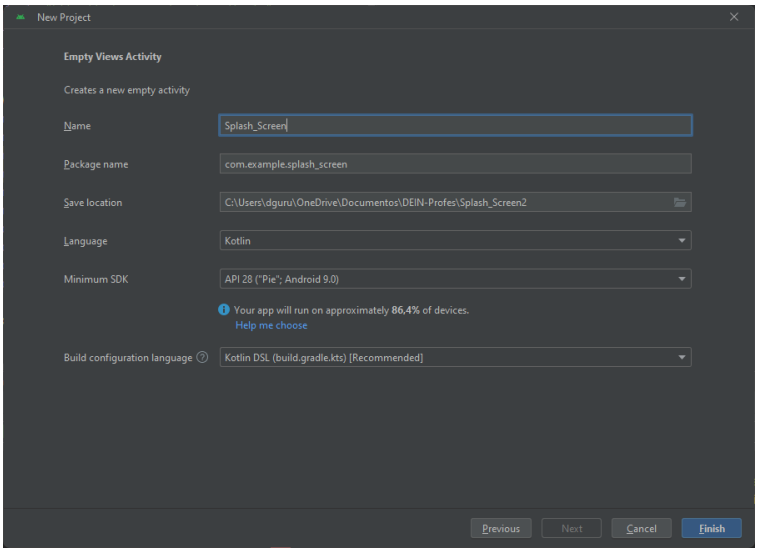
Actividad 3.1. Primeras Vistas(Views)

En esta actividad vas a crear un nuevo proyecto e insertaremos sobre él varios elementos (Views) desde el modo de "Diseño" de Android Studio.

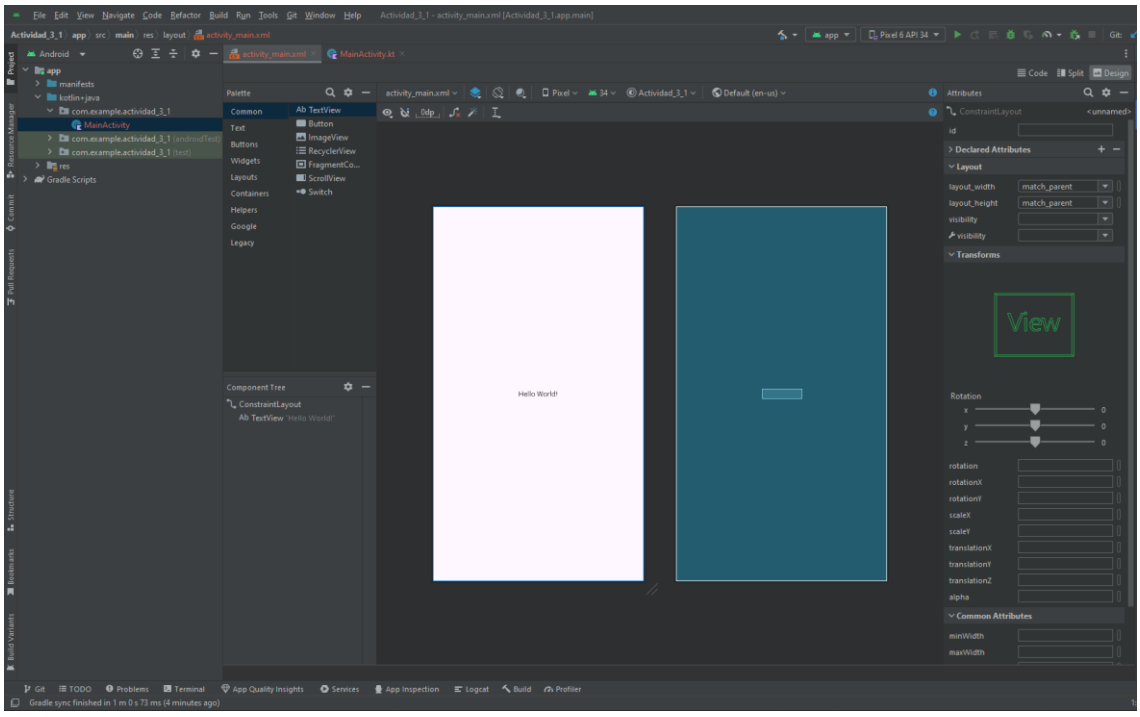
1. **Primero crea un nuevo proyecto en Android Studio del tipo "Empty Views Activity"**



Llámale "**Primeras_Vistas"** el Lenguaje "Kotlin" y aunque tenemos diversas plataformas y APIs en las que utilizar nuestra aplicación, nosotros nos centraremos en aplicaciones para teléfonos y tablets, en cuyo caso tan sólo tendremos que seleccionar la API mínima (es decir, la versión mínima de Android) que soportará la aplicación. Android 9.0. como versión mínima (API 28):

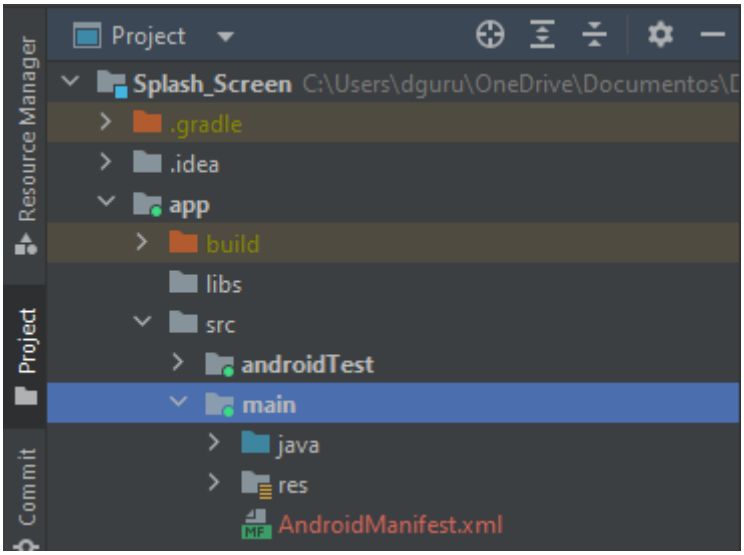


Una vez configurado todo pulsamos el botón **Finish** y Android Studio creará por nosotros toda la estructura del proyecto y los elementos indispensables que debe contener. Si todo va bien aparecerá la pantalla principal de Android Studio con el nuevo proyecto creado:

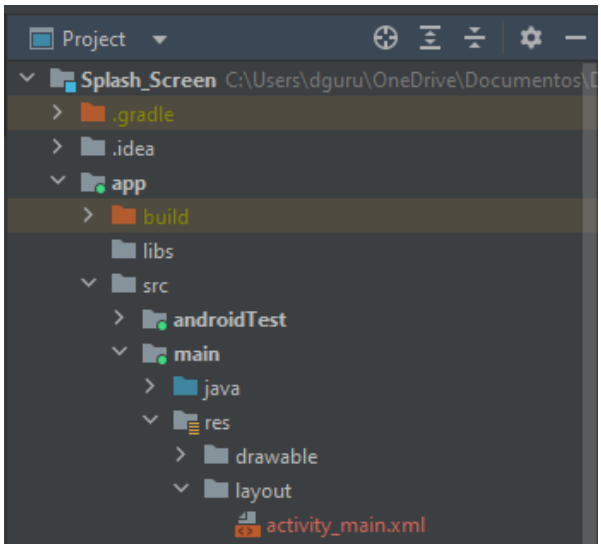


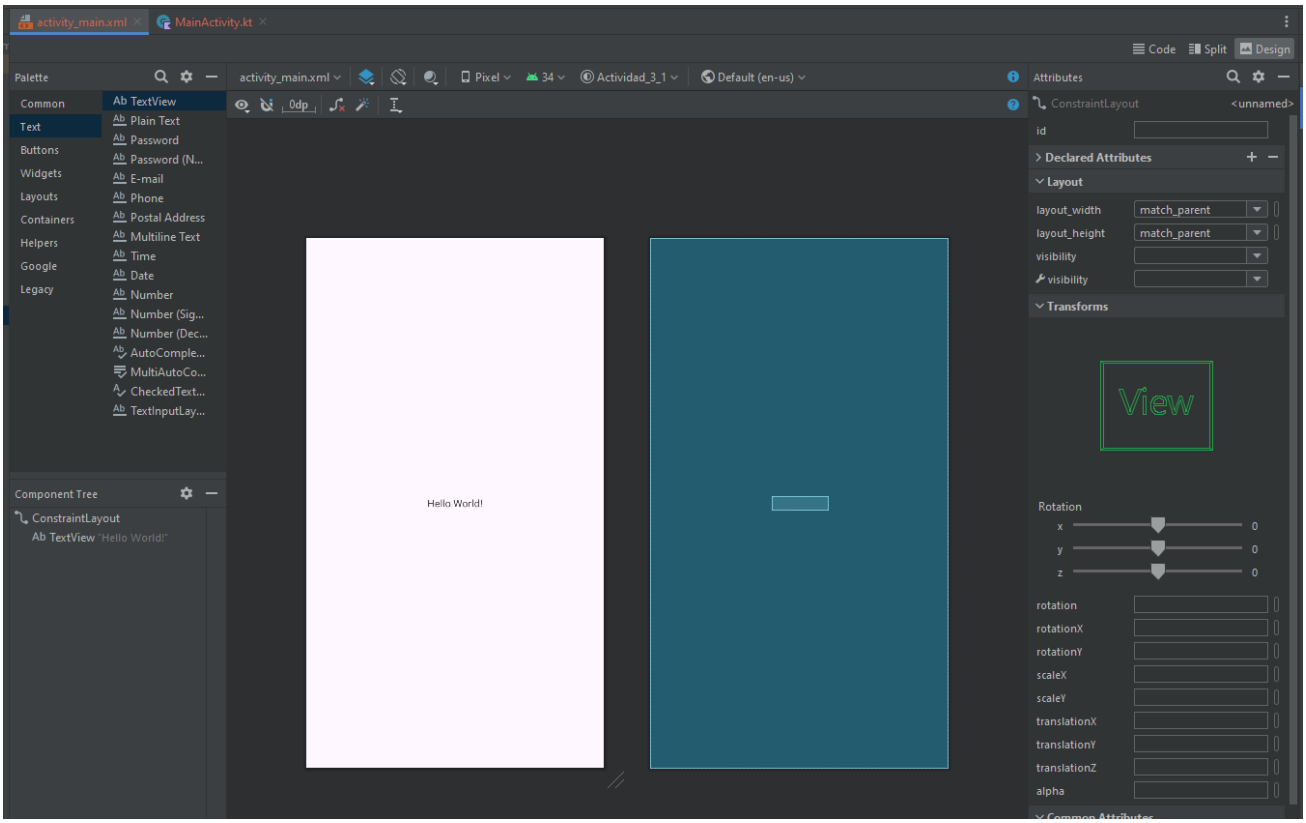
En la parte izquierda, podemos observar todos los elementos creados inicialmente para el nuevo proyecto Android, sin embargo, por defecto los vemos de una forma un tanto peculiar que podría llevarnos a confusión. Para entender mejor la estructura del proyecto vamos a cambiar momentáneamente la forma en la que Android Studio nos la muestra. Para ello, pulsaremos sobre la lista desplegable situada en la parte superior izquierda, y cambiaremos la vista de proyecto al modo **"Project".**

Tras hacer esto, la estructura del proyecto cambia un poco de aspecto y pasa a ser como se observa en la siguiente imagen:



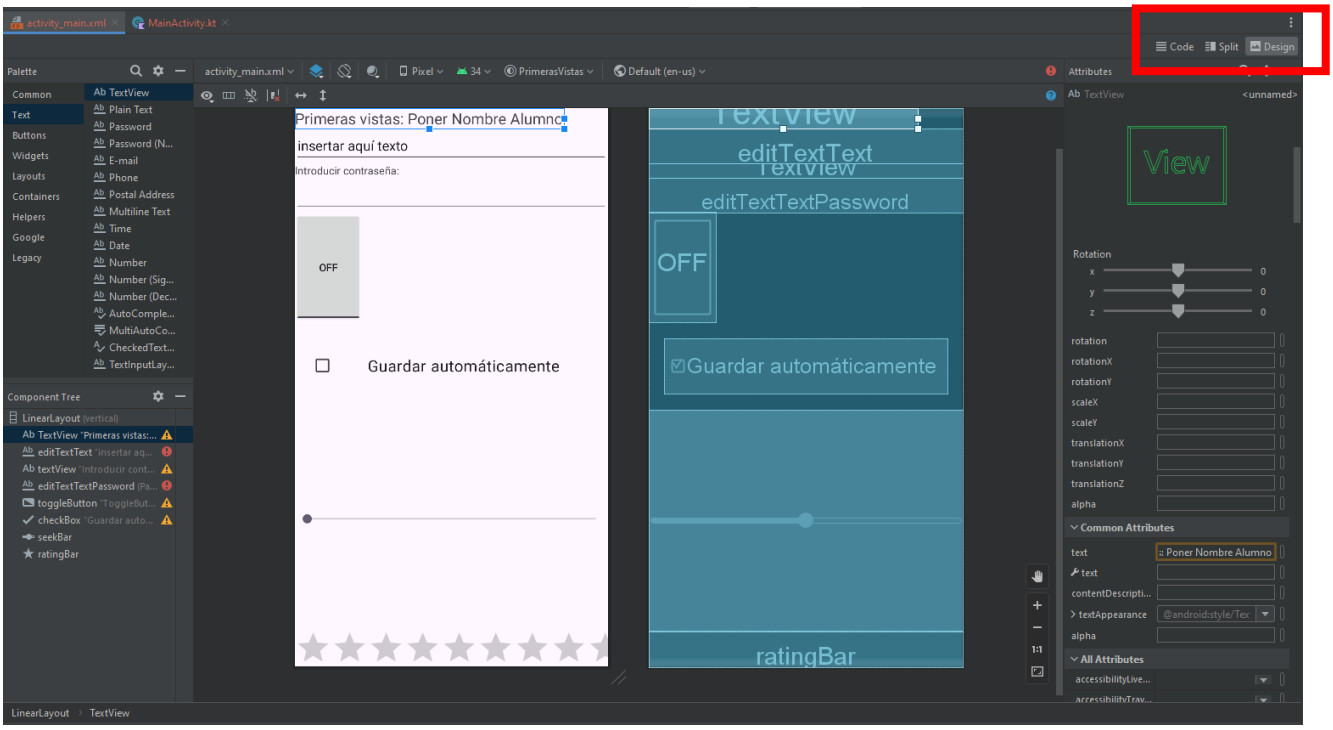
En la estructura de carpetas del proyecto la c***arpeta /app/src/main/res/*** contiene todos los ficheros de recursos necesarios para el proyecto: imágenes, layouts, cadenas de texto, etc. Los diferentes tipos de recursos se pueden distribuir entre diversas subcarpetas. Entre los recursos creados por defecto cabe destacar los **layouts**, donde encontramos el fichero "**activity_main.xml**", que contienen la definición de la interfaz gráfica de la pantalla principal de la aplicación. Si hacemos doble clic sobre este fichero Android Studio nos mostrará esta interfaz en su editor gráfico, y como podremos comprobar, en principio contiene tan sólo una etiqueta de texto con el mensaje "Hello World!"





Ventana gráfica de "activity_main.xml

Pulsando sobre las pestañas superior derecha "Design", "Code" y “Split” podremos alternar entre el editor gráfico (tipo arrastrar-y soltar), mostrado en la imagen anterior, y el editor XML que se muestra en la imagen siguiente:



Vista Diseño y configuración final de la MainActivity.xml

Una vez ahí, utilizando la Vista de “**Design**” vas a ir creando un diseño para que el resultado final sea similar al de la imagen superior.

Primero vamos a hacer que la raíz del *layout* se base en un **LinearLayout vertical**:

Este tipo de *layout* es uno de los más sencillos de utilizar. Te permite representar las vistas **una debajo de la otra**. Para ello, en el marco “***Component Tree***” pulsa con el botón derecho sobre “**ConstraintLayout**” y selecciona “***Convert View…***”. Indica que quieres usar el **LinearLayout.** El layout que ha añadido es horizontal pero en nuestro caso lo queremos de tipo vertical, para cambiarlo pulsa con el botón derecho sobre “**LinearLayout**”. y selecciona “***LinearLayout>Convert* orientation to vertical”.**

Todas las operaciones que hacemos en modo diseño visual (modo Design) también las podemos hacer con el editor de texto. Para probarlo deshaz el trabajo anterior, usando la opción Edit/Undo Ctrl+Z). Selecciona la lengüeta **Code** y cambiar las etiquetas:

```
<androidx.constraintlayout.widget.ConstraintLayout
```

Por:

```
</LinearLayout>
```

Y añade la orientación del layout. Debería quedar algo así:

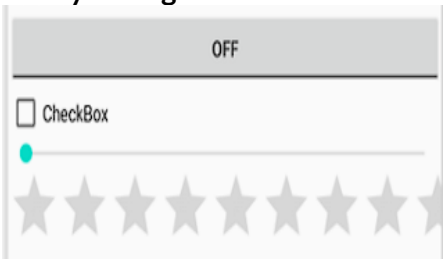
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!" />

</LinearLayout>
```

Ahora, vuelve al modo **Design** y vamos a insertar todos los elementos de la **Activity**:

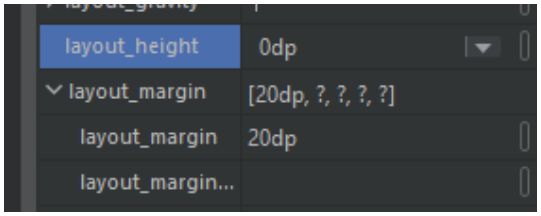
a. Primero borra el texto que aparece por defecto en la plantilla.
b. Inserta un **TextView (Palette>Text),** en el que tiene que aparecer el texto "Primeras vistas: tu nombre y apellidos"
c. Añade en la parte superior del Layout anterior otra vista de tipo entrada de texto, de tipo normal (**Plain Text).** Lo encontrarás dentro del **grupo Text**. Haz que aparezca el texto "insertar aquí texto" luego podemos borrar este campo para utilizarlo como queramos.
d. Inserta otro **TextView (Palette>Text),** en el que tiene que aparecer el texto "Introducir contraseña:"
e. Añade un campo de tipo "**Password" (Palette>Text)**
f. Ahora desde la paleta de izquierda, arrastra al área de diseño, los siguientes elementos: **ToggleButton, CheckBox,SeekBar y RatingBar:**



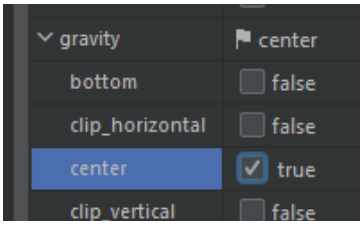
g. Selecciona la vista **ToggleButton**. y configúralo como **layout width** a "wrap_content". Conseguirás que la anchura del botón se ajuste a la anchura de su texto. Seleccionando en **layout width** a "match_parent", haremos que el ancho del botón se ajuste a su contenedor. Observa en el marco Attributes como cambia la propiedad **layout_width**. Deja el valor wrap_content.

h. Con la vista **ToggleButton** seleccionada. Pulsa el botón **layout Height** a "match_parent"; Conseguirás que la altura del botón se ajuste al a altura de su contenedor. El problema es que el resto de los elementos pueden dejar de verse (en ese caso puedes volver al estado anterior).

i. Selecciona la vista **CheckBox**. Y en el marco Atributes, en la parte inferior, pulsa **en All attributes**. Busca la propiedad **layout_margin** en el campo con el mismo nombre introduce "**20dp**". Se añadirá un margen alrededor de la vista. Modifica el resto de los parámetros:



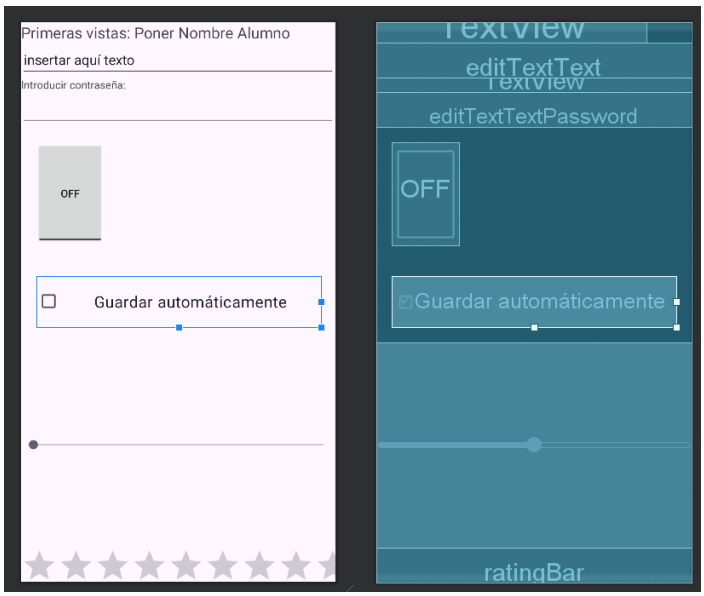
j. Busca la propiedad **gravity** y selecciona center, true:



k. Selecciona la vista **CheckBox**. Asigna en el marco Attributes , **layout_height** = 0dp; y **layout_weight** = 1. Observa como toda la altura restante es asignada a la vista seleccionada.

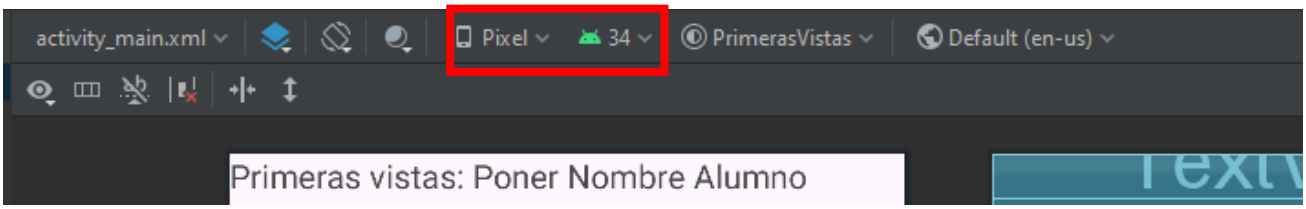
l. Selecciona la vista **CheckBox** y observa las diferentes propiedades que podemos definir en el marco **Attributes**. Algunas ya han sido definidas por medio de la barra de botones. En concreto y siguiendo el mismo orden que en los botones hemos modificado: **Layout margin** = 20dp, **gravity** = center y **Layout weight** = 0.5. Busca el atributo **Text** y sustituye el valor "CheckBox" por "Guardar automáticamente" y **Text size** por "9pt".

El resultado obtenido debería ser similar a este:



Revisa el Modo "**Code**" y "**Split**" y relaciona los elementos de uno y otro y como podrías **cambiar** los textos, formatos, etc, desde el propio código.

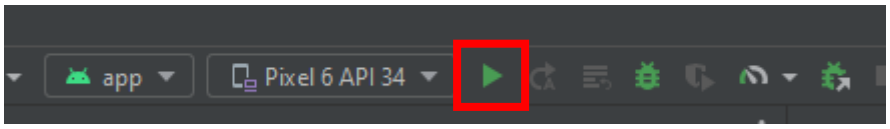
Utiliza los botones de la barra superior para observar cómo se representará el layout en diferentes situaciones y tipos de dispositivos:



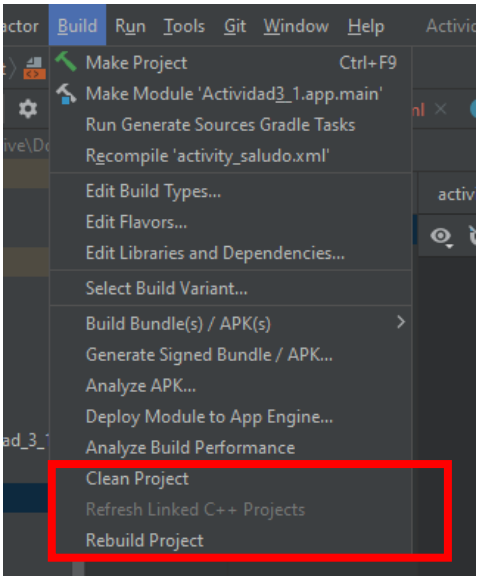
Guarda el proyecto, y ahora pasaremos a verlo en un dispositivo virtual:

2. Compilar y probar el funcionamiento de la App en un dispositivo virtual

Para ver el aspecto y el comportamiento de tu app en un dispositivo, debes compilarla y ejecutarla. Android Studio configura proyectos para que puedas implementar tu app en un dispositivo virtual o físico. Si tuviéramos un dispositivo configurado podríamos ejecutar "Run App" para probar la App:



También es importante utilizar y conocer las opciones "Clean Project" y rebuild Project que nos ayudan a compilar el proyecto sin problemas (o ayudarnos a solucionarlos)

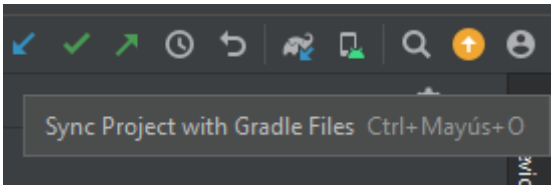


Clean y Rebuild Project

Clean Project (Limpiar Proyecto), limpia el proyecto eliminando todos los archivos compilados y los artefactos generados en la compilación anterior. Se utiliza cuando hay problemas en la compilación o cuando se quieren eliminar los archivos generados para comenzar desde cero. Puede ser útil después de realizar cambios significativos en el proyecto.

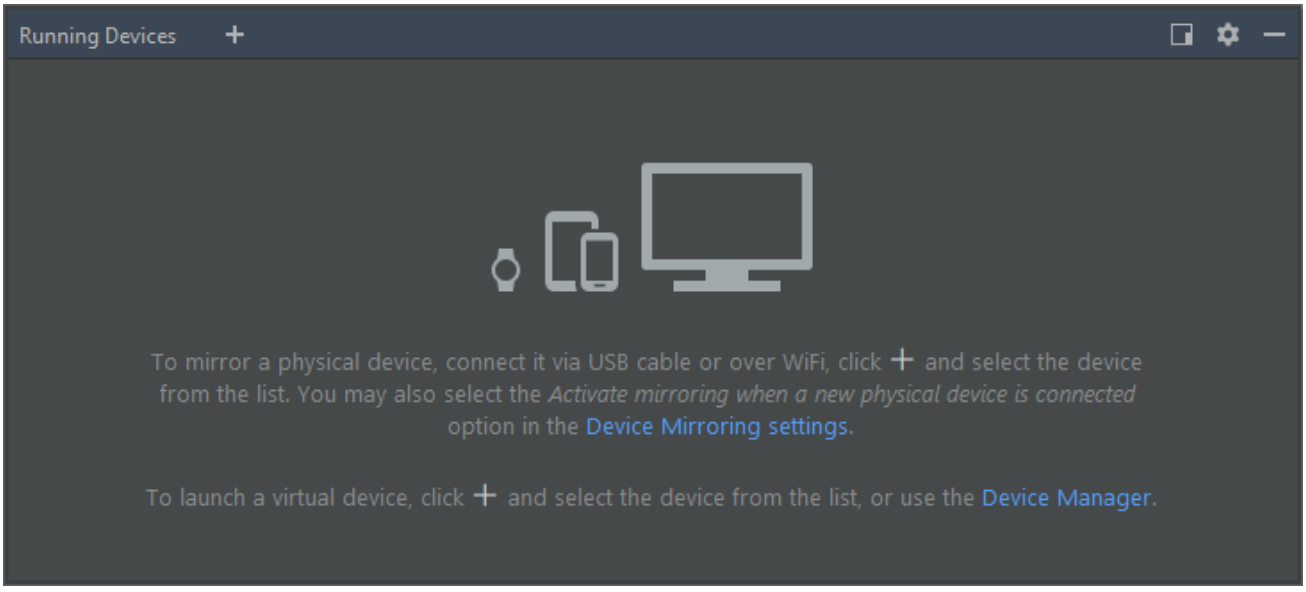
Rebuild Project (Reconstruir Proyecto), realiza la acción de "Clean Project" y luego realiza una compilación completa desde cero. Elimina todos los archivos compilados y regenera todo, incluyendo los archivos de clase, recursos y otros artefactos de construcción. Se utiliza cuando los problemas persisten incluso después de realizar la "Clean Project". Puede ser beneficioso cuando se sospecha que hay problemas profundos en la configuración del proyecto.

También es conveniente a veces cuando nos da algún error revisar la sincronización del proyecto con los ficheros del compilador:

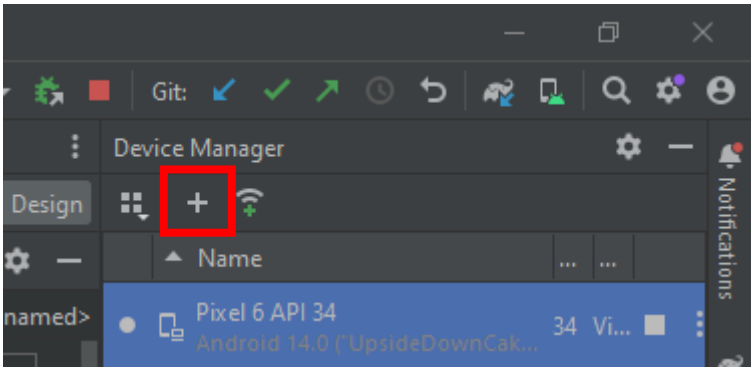


Parte superior derecha del programa

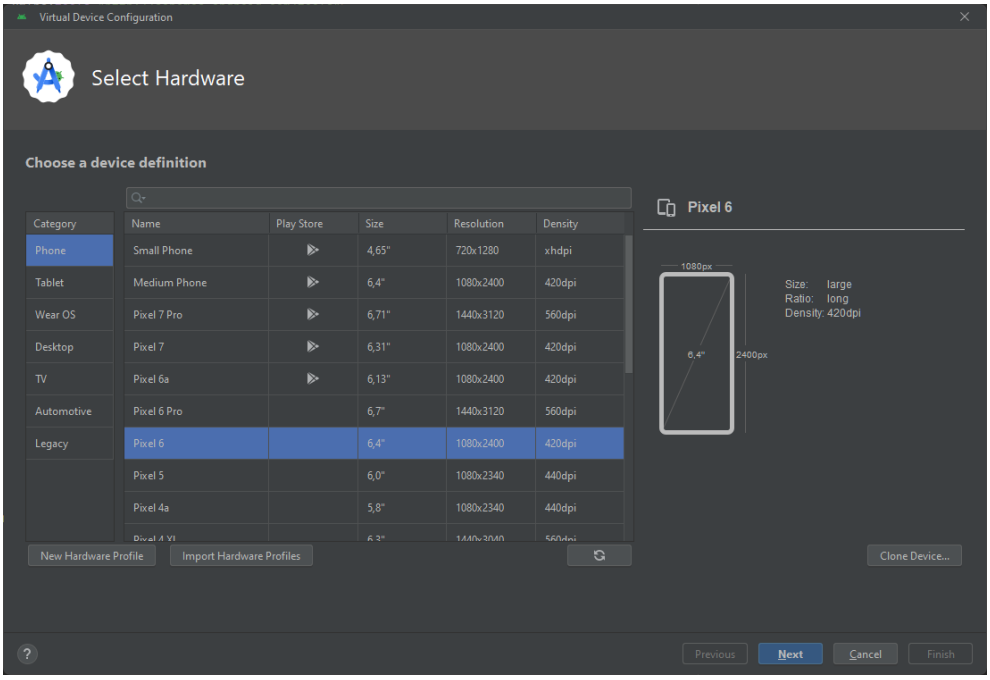
Si todos estos pasos se han ejecutado sin errores, guarda el proyecto y ahora vamos a **probar la aplicación Android** en nuestro PC sin tener que recurrir a un dispositivo físico, por lo que tenemos que definir lo que se denominan **AVD (Android Virtual Device).** Para crear un AVD seleccionaremos el menú Tools / Android / AVD Manager. Si es la primera vez que accedemos a esta herramienta veremos la pantalla siguiente:



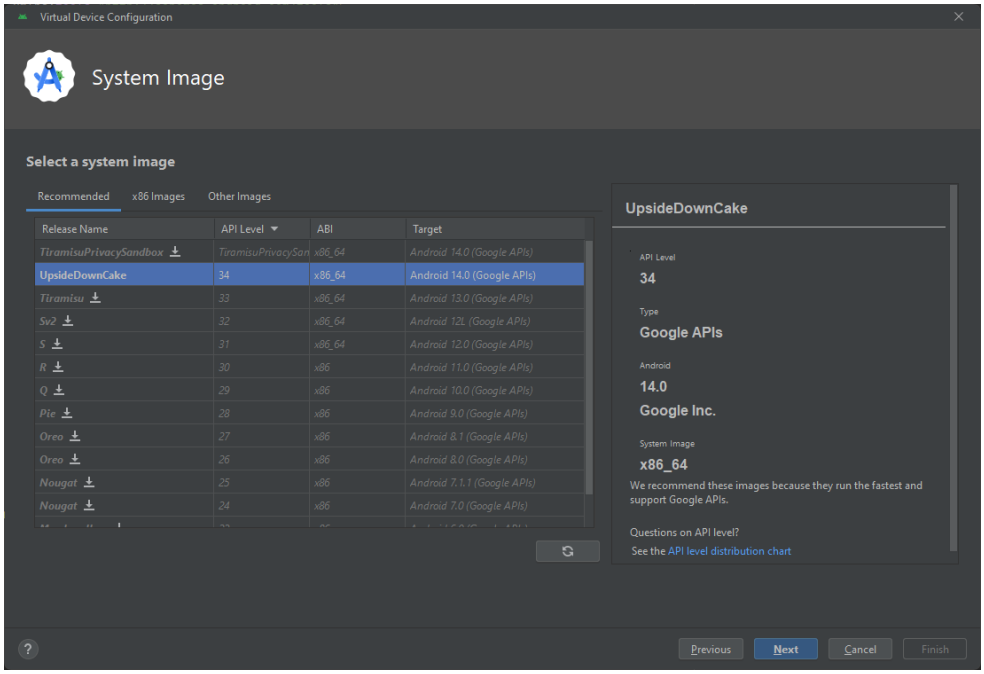
Si no te ha aparecido esta pantalla, puedes hacerlo desde aquí:



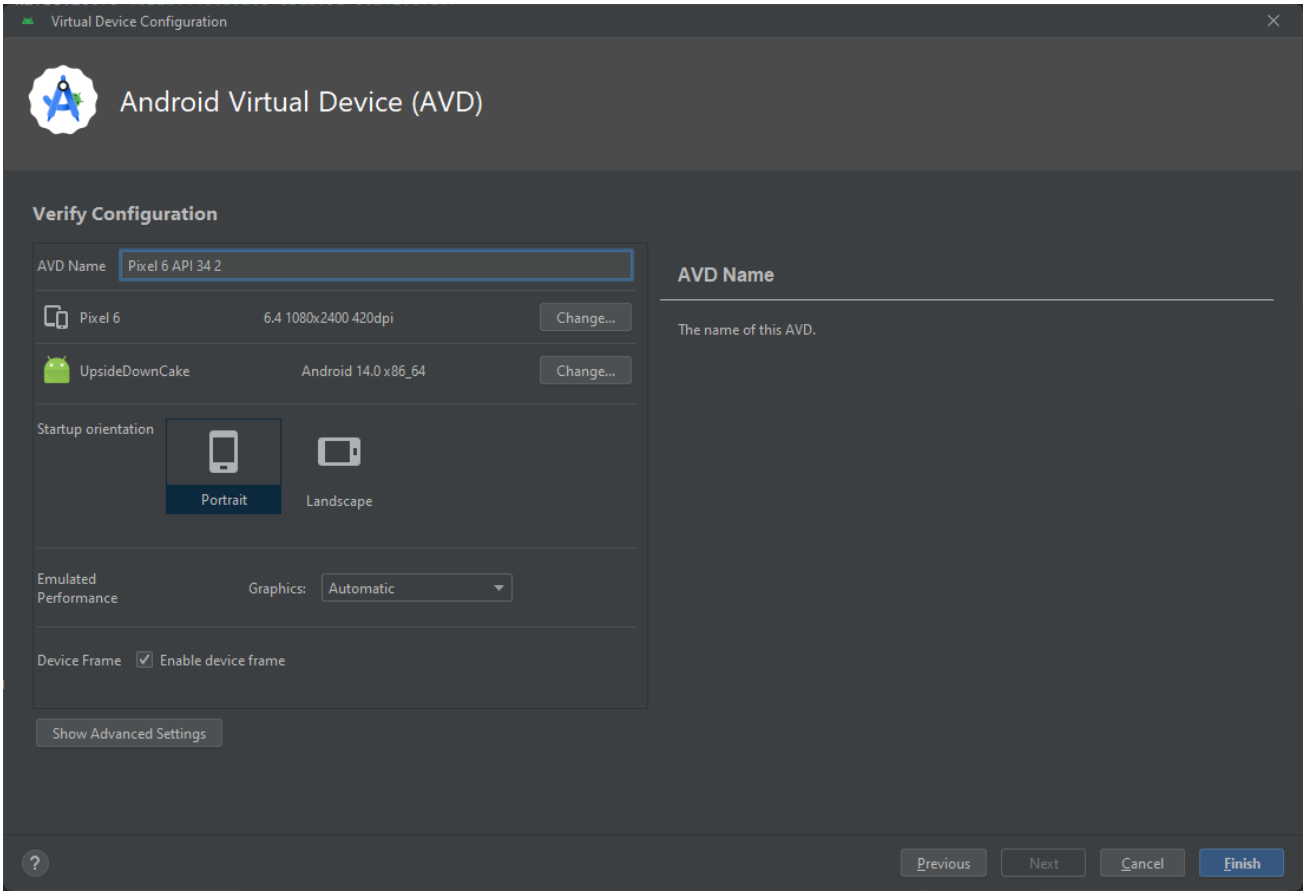
Pulsando el botón central "**Create a virtual device**" de la primera pantalla o dándole al "+" en el "Device Manager" accederemos al asistente para crear un AVD. En el primer paso tendremos que seleccionar a la izquierda qué tipo de dispositivo queremos que "simule" nuestro AVD (teléfono, tablet, reloj, ...) y el tamaño, resolución, y densidad de píxeles de su pantalla. En mi caso seleccionaré por ejemplo las características de un Pixel 6 y pasaremos al siguiente paso pulsando "Next".



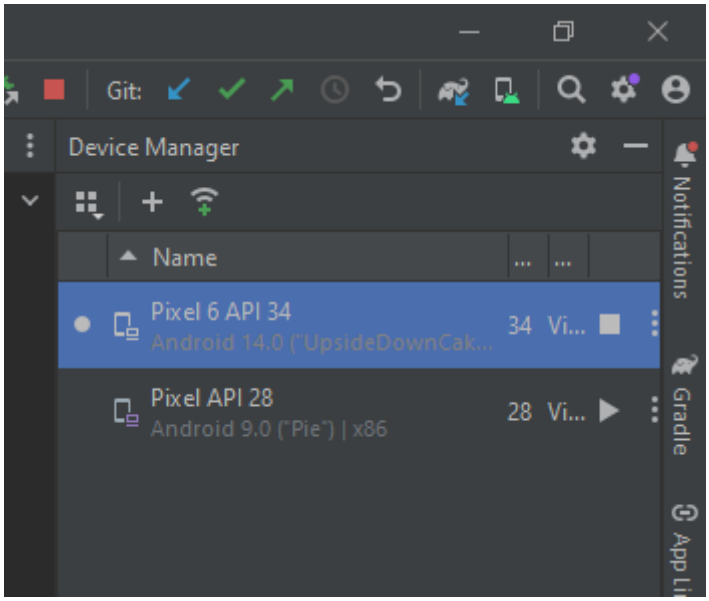
En la siguiente pantalla seleccionaremos la versión de Android que utilizará el AVD. Aparecerán directamente disponibles las que instalamos desde el SDK Manager al instalar el entorno, aunque tenemos la posibilidad de descargar e instalar nuevas versiones desde esta misma pantalla. En mi caso utilizaré la última versión de Android (UpsideDownCake) (API Level 34) para este primer AVD (podemos crear tantos como queramos para probar nuestras aplicaciones sobre distintas condiciones).



En el siguiente paso del asistente podremos configurar algunas características más del AVD, si simulará tener cámara frontal y/o trasera, teclado físico, ... Recomiendo pulsar el botón "Show Advanced Settings" para ver todas las opciones disponibles. Si quieres puedes ajustar cualquiera de estos parámetros, pero por el momento os recomiendo dejar todas las opciones por defecto.

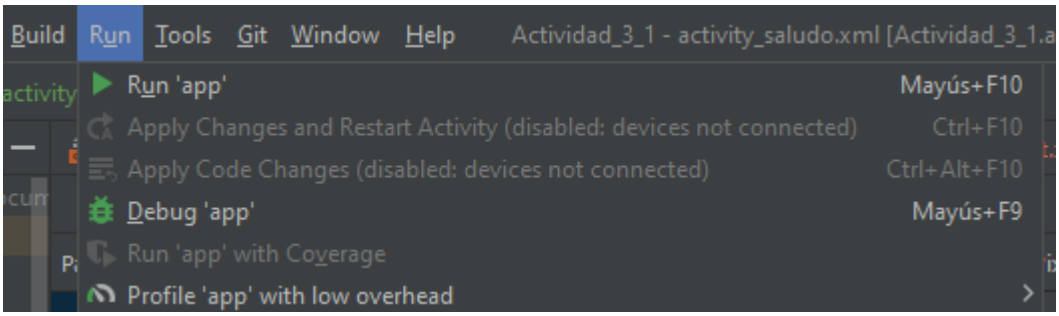


Tras pulsar el botón Finish tendremos ya configurado nuestro AVD, por lo que podremos comenzar a probar nuestras aplicaciones sobre él. En el Device Manager veremos los dispositivos que tenemos:

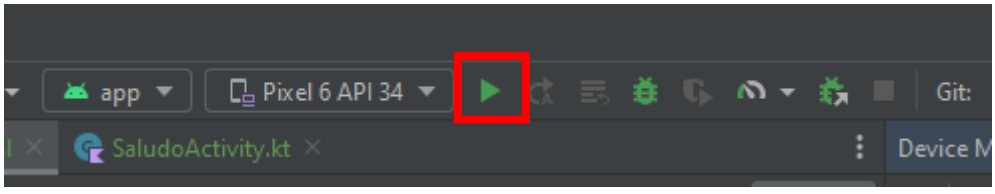


Para ejecutar la App en el dispositivo que hemos creado podemos hacerlo de varias opciones:

1. Pulsaremos simplemente el menú **Run / ejecutar Run** 'app' (o la tecla rápida Mayús+F10). Android Studio.



2. Ejecutarlo desde la ventana de navegación de la parte superior izquierda:

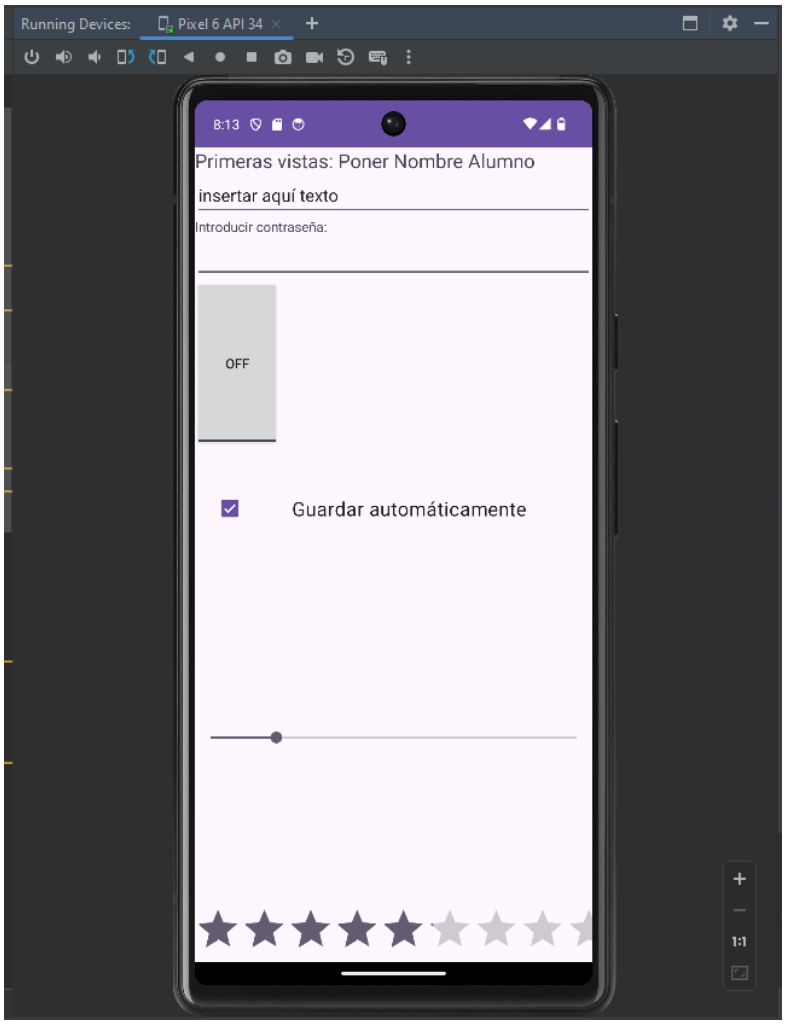


Donde en caso de tener más dispositivos, podemos seleccionar el que queramos en ese momento. Le damos al botón de **"Run App"**

Si todo va bien, tras una pequeña (o no tan pequeña) espera aparecerá el emulador de Android y se iniciará automáticamente nuestra aplicación (si se inicia el emulador pero no se ejecuta

automáticamente la aplicación podemos volver a ejecutarla desde Android Studio, mediante el menú Run, sin cerrar el emulador ya abierto).

Prueba a escribir un nombre y pulsar el botón "Aceptar" para comprobar si el funcionamiento es el correcto.



Guarda el proyecto, expórtalo como Zip, y crea la APK tal como se ha visto en la teoría.