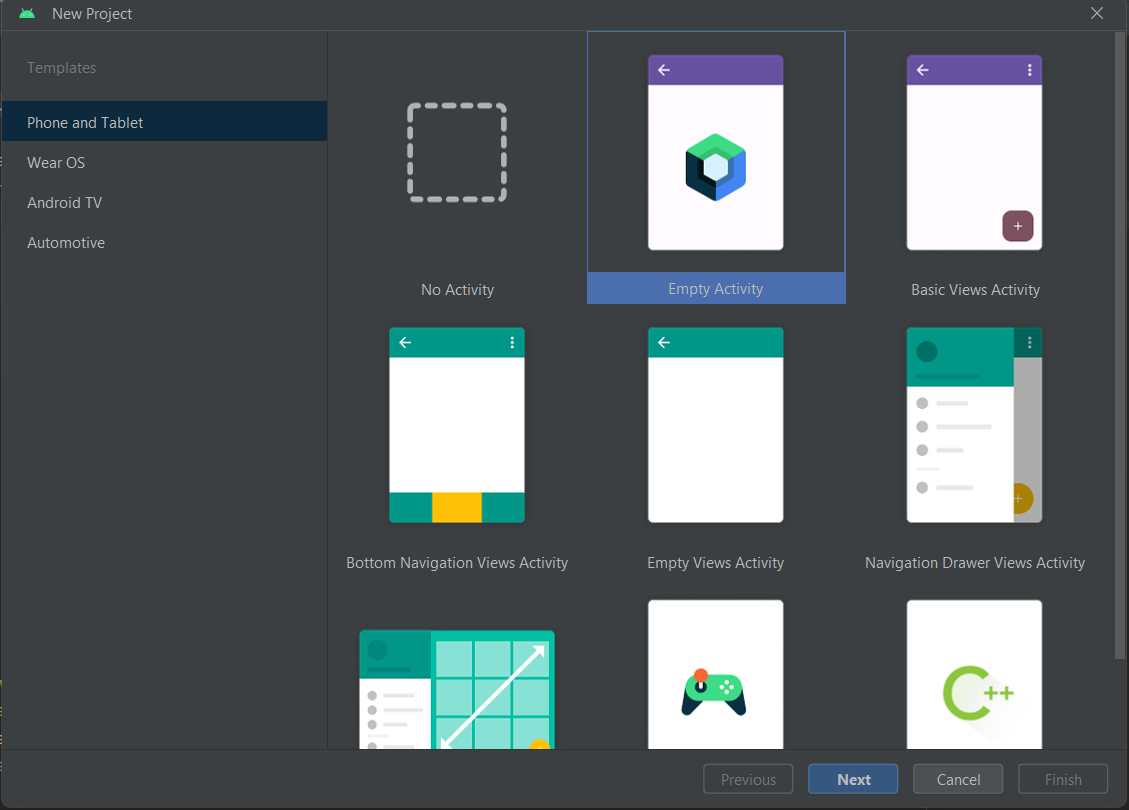
# Práctica 1: Introducción a Android Studio

La finalidad de esta práctica es aprender a utilizar el IDE Android Studio, que es el que utilizaremos a lo largo del curso. Podéis utilizar el documento *Manual de Android Studio* donde se incluye una descripción textual de lo que veremos en clase.

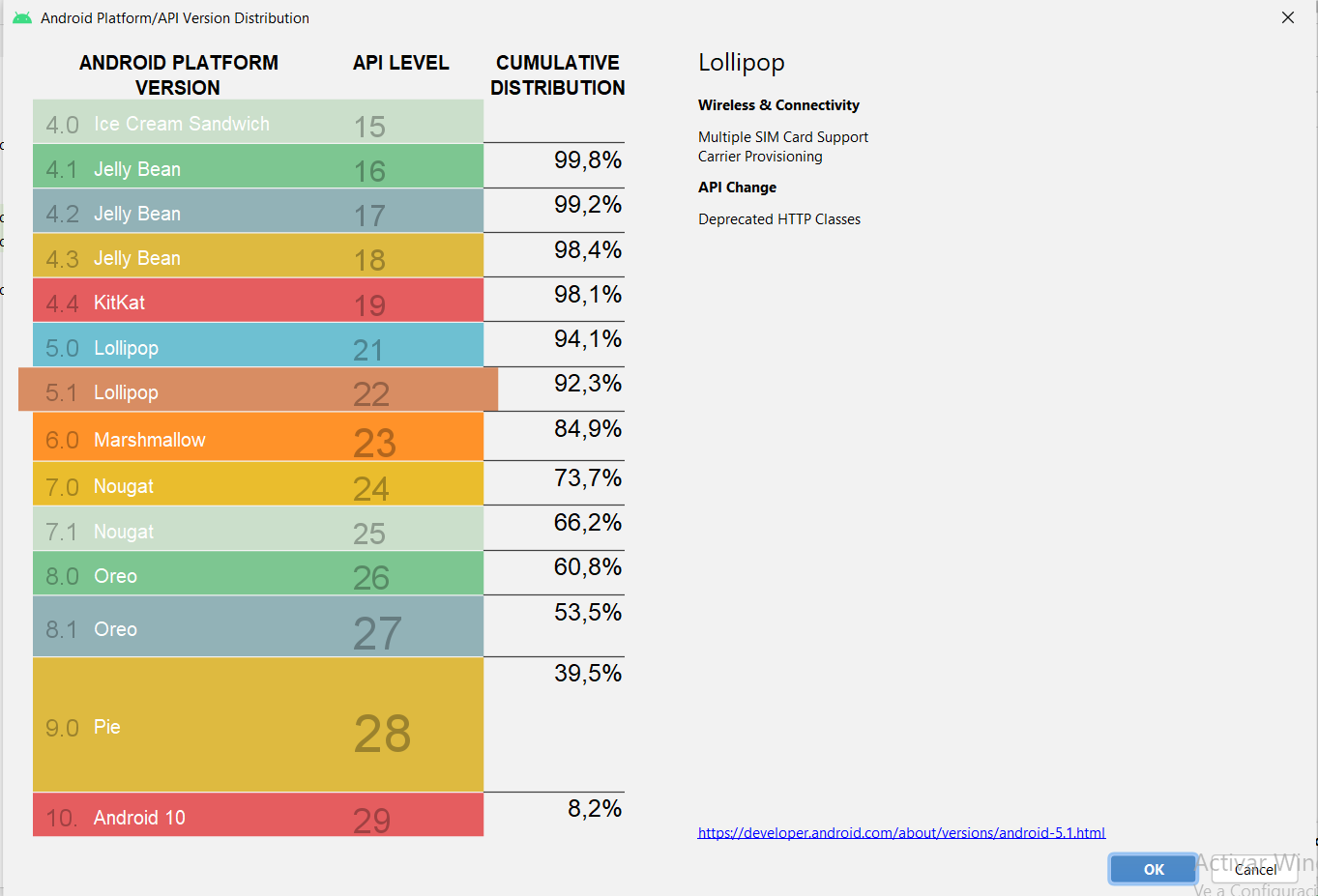
## Ejercicio 1

Vamos a analizar uno de los proyectos que hay de prueba y vamos a estudiar su comportamiento.

Para ello crea un nuevo proyecto de tipo *EmptyActivity* y llámalo PruebaBasica.



Después vamos a tener que elegir la versión de Android:



Cuanto más nueva; menos dispositivos podrán ejecutar la aplicación, pero nuevas clases y herramientas disponibles. Una buena elección es escoger la 7.0 Nougat

Una vez iniciado el proyecto vamos a ver los ficheros que hay, que serán un fichero Kotlin en la dirección del paquete, y un subpaquete ***ui.theme*** con 3 ficheros .kt, relacionados con la UI de nuestra app.

Crea un AVD si no existía ya, ejecuta el proyecto y responde a estas preguntas:

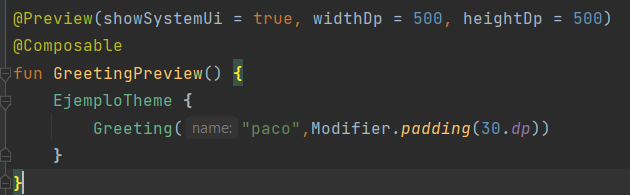
1. Estudia su código. ¿Cuántos archivos kotlin hay? ¿Por qué? Busca información sobre el método OnCreate(), y sobre el ciclo de vida de una actividad en Android.

Hay 3 archivos Kotlin: MainActivity, que se dedica a la funcionalidad real de la aplicación, junto a la posibilidad de ver la preview de ella sin tener que ejecutarla, ExampleInstrumentedTest y ExampleUnitTest cuya función es testear el funcionamiento de tanto los componentes de la aplicación como la “instrumental” del proyecto, que inyecta datos y comprueba el rendimiento de la aplicación por completo. Aunque no sea obligatorio testear una app, si es recomendable hacer uso de estos tests para asegurar que una funcionalidad no estalle toda nuestra app, o comprobar qué tan bien rinde antes de lanzarla. MainActivity es altamente obligatorio, debido a que dentro de esta clase se determina qué hará la aplicación en los distintos estados como onCreate(), onStart() o onStop(). Al arrancar una App Android bien diseñada, ésta hará usos de estados, que son bloques de código que se ejecutarán cuando pasen ciertos eventos en la app. onCreate() se ejecutará nada más iniciar por primera vez la aplicación, seguida por onStart(). Cuando se deja la app en segundo plano, se ejecutará onPause() y al “matarla”, cerrarla completamente, onStop().

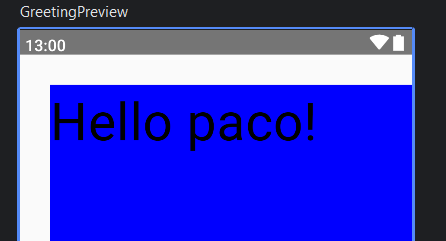
1. ¿Qué elemento gráfico conforma la interfaz de la pantalla? ¿Cómo se llama?

En Compose la interfaz de conforma de componentes. En este caso se llama EjemploTheme, que contiene un contenedor Surface y éste renderiza la función Greeting que devuelve un Text.

1. Crea dos AVD para lanzar tu aplicación, uno similar a un dispositivo tablet y otro un móvil normal.
2. [Investiga](https://developer.android.com/jetpack/compose/tooling/previews) qué parámetros pueden tener los @Composables con la etiqueta @Preview. ¿Se pueden tener varios? Crea uno en el que se pueda ver la pantalla del dispositivo, y otro con un tamaño fijo de 500 x 500.



1. Como ya hemos visto en clase, los Modifier nos permiten cambiar el estilo de nuestros componentes: tamaño, alineación, padding, bordes… Aquí en este [enlace](https://developer.android.com/jetpack/compose/modifiers-list) te dejo la documentación oficial de Android, y con ello investiga:
   1. Cómo se puede poner el fondo del texto en azul.



Añadimos al componente Text el modifier Modifier.background(Color.Blue), pinta en el fondo del componente de un color, si le pasamos una forma lo hará de una forma concreta como un cuadrado, círculo...

* 1. Añade un borde de color amarillo.



Añadimos el modificador Modifier.border al componente Text, pasándole el grosor en dp y el color en este caso el amarillo. Podemos ver que Modifier se puede concatenar como si fueran atributos CSS.

* 1. ¿Cuál es la diferencia entre fillMaxSize(), fillMaxHeight() y fillMaxWidth())?

fillMaxSize() hará que todo el contenido ocupe todo el tamaño que se haya asignado por parámetro por el ancho y alto de la pantalla. En cambio si por ejemplo solo hacemos fillMaxWidth(), solo modificará el ancho, sin tocar el alto.

* 1. ¿Cuál es la diferencia entre padding y offset?

Explica con tus palabras cómo funciona cada modificador y el efecto que tiene a nivel visual, añadiendo explicaciones con capturas.

1. ¿Qué ocurre si añadimos otro elemento textual justo después del que ya hay creado?

## Ejercicio 2: Depuración de una actividad

Una herramienta muy importante en cualquier IDE es conocer la forma de **depurar** y buscar errores en nuestras aplicaciones. Para ello podéis bajar el código de <https://github.com/ANavarro96/EjemploDebugKotlin>(clonar el repositorio, o bajar el proyecto y abrirlo desde Android Studio), que es una aplicación que contiene un error de ejecución.

## ¿Qué es depurar una aplicación?

Un debugger (o depurador) es una herramienta que se utiliza para la detección de errores y la ejecución de los programas de forma más controlada. Nos permite ejecutar un programa instrucción a instrucción y ver el valor de las variables, en programas más complejos ver el estado de la memoria.

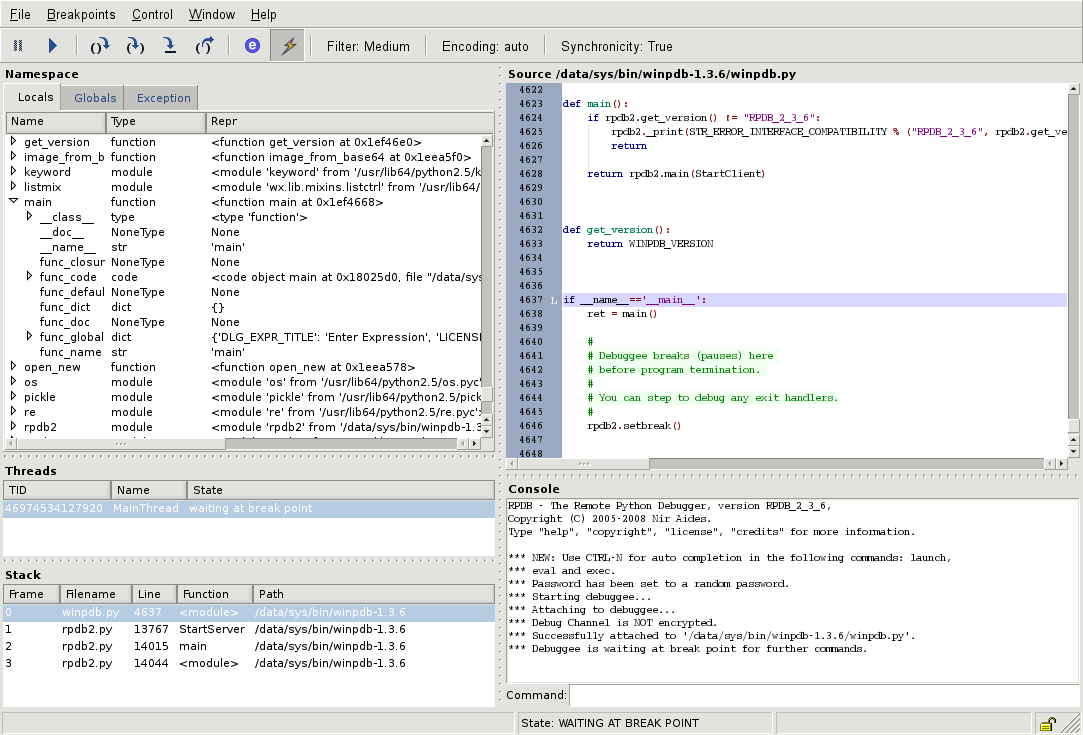
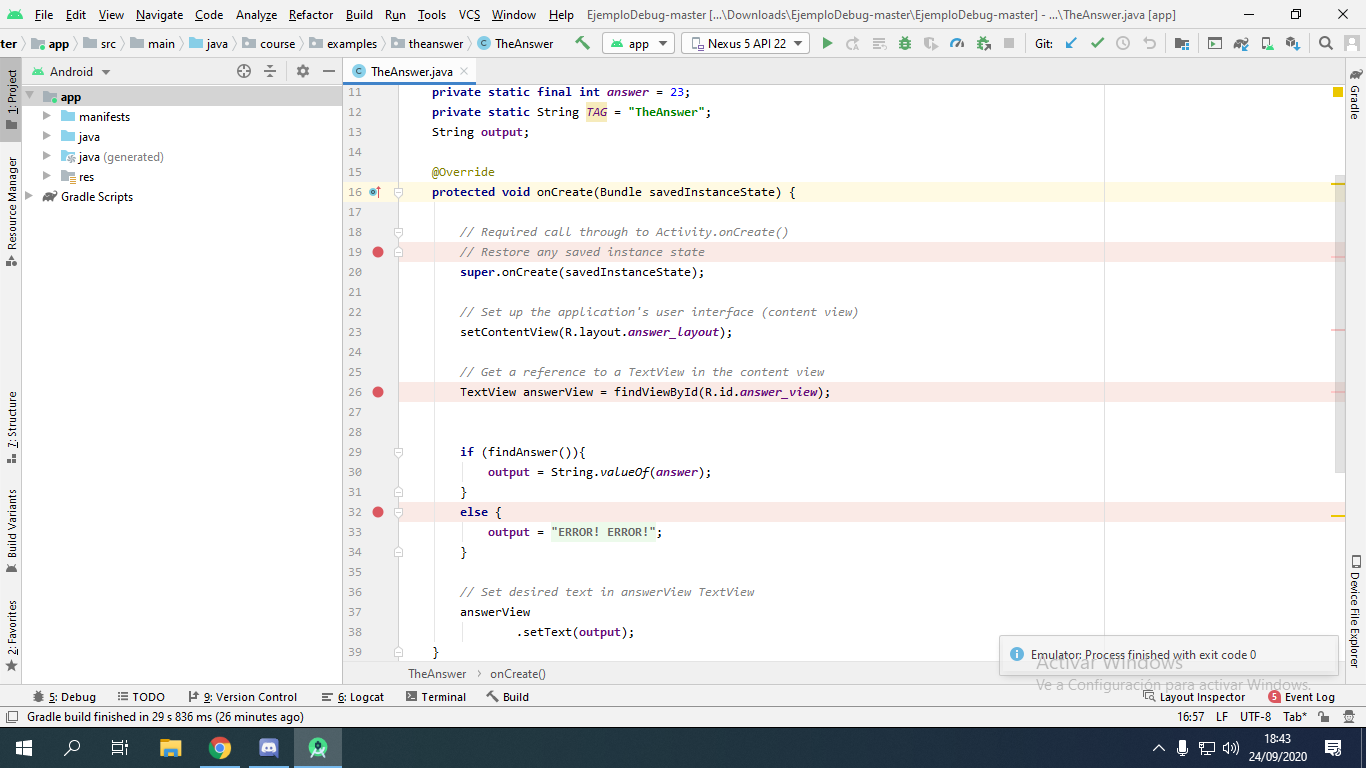


Ilustración 1: Ejemplo de depurador

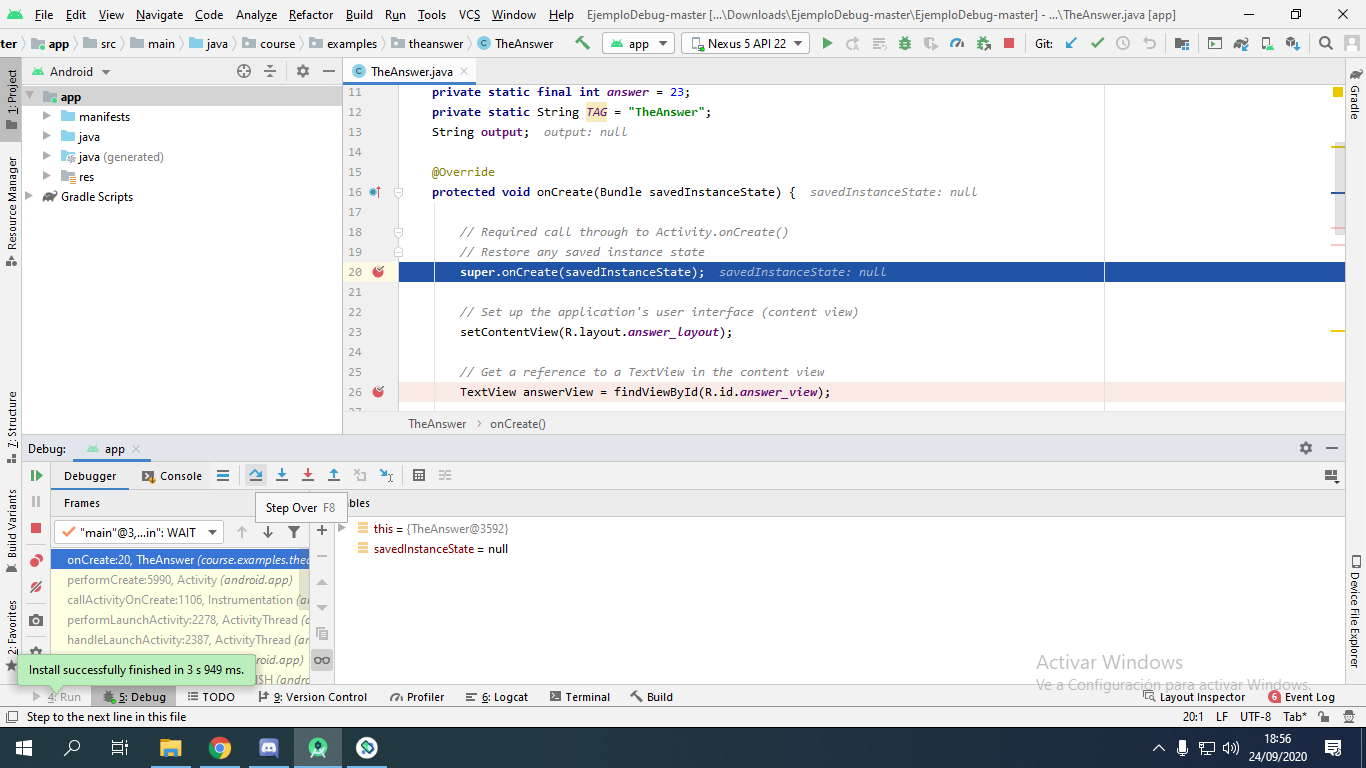
Android Studio incluye un depurador integrado, que permite controlar el flujo de nuestra app, la cual estará corriendo en un emulador virtual que hemos creado.

El depurador puede ejecutar línea a línea, o puede ejecutar hasta la línea que decidamos nosotros. Esto se llama breakpoint, por que es un punto en el que puede haber algún error.

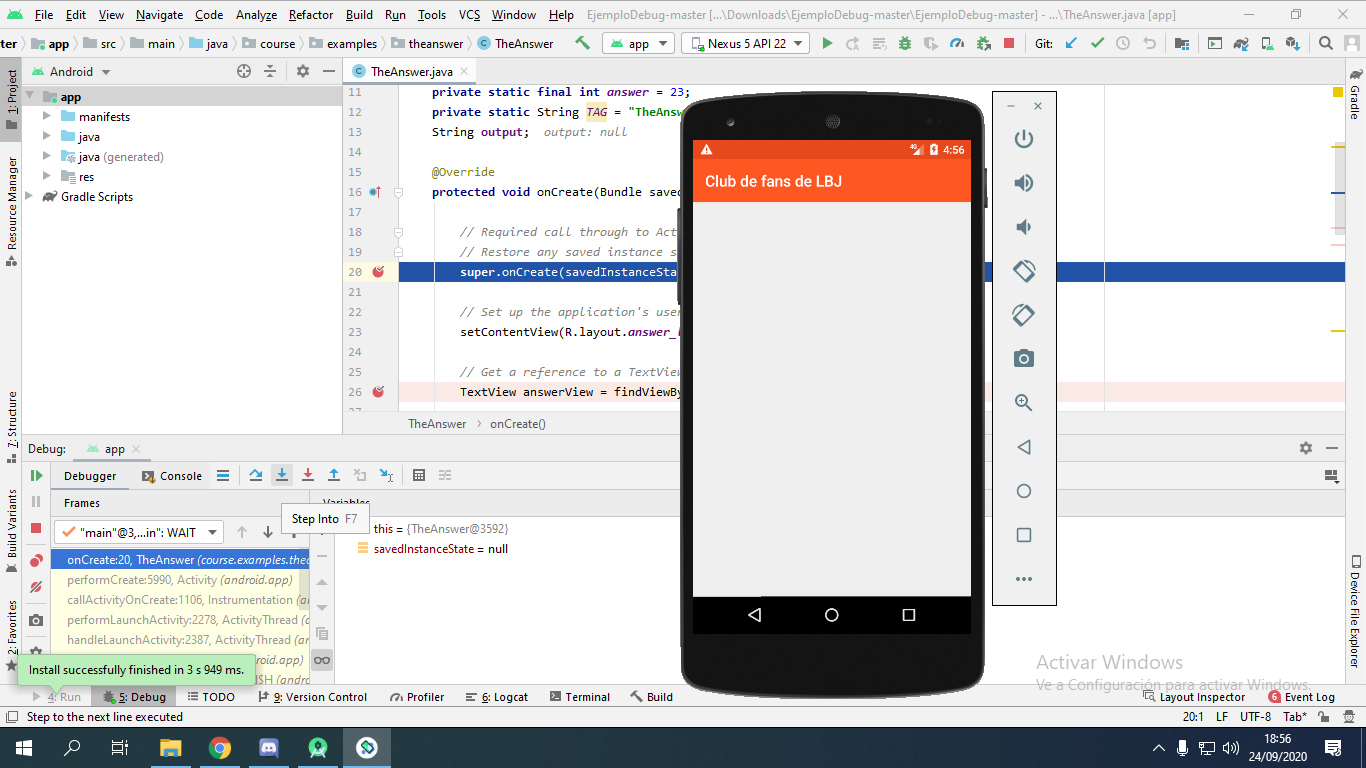


Para crear breakpoints tenemos que hacer click en el borde gris a la izquierda del código, entre la línea de código y el propio código.

Gracias a esto lanzaremos una instancia del depurador que lanzará la actividad, y podremos monitorizar su ejecución y ver los resultados de la traza en la consola de depuración:



Nuestra aplicación se quedará “parada” esperando a que continuemos la ejecución; al llegar al primer breakpoint podremos darle a varias opciones:

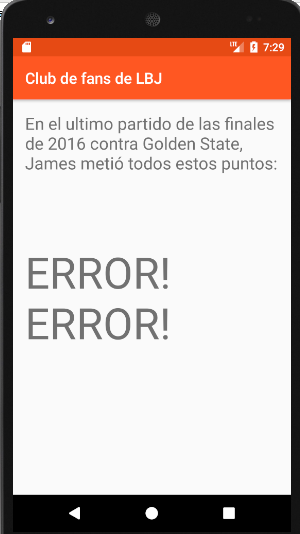


Step Over pasará a la siguiente instrucción, y Step Into lo que nos permite es seguir la ejecución dentro de una función si tenemos un breakpoint ahí. Estas opciones aparecen forma de flecha en la parte superior de la consola de depuración.

Una vez que tengamos seleccionado nuestro breakpoint, tenemos que ejecutar la aplicación y ver donde puede estar el error.

Esta aplicación ha sido creada por la NBA para recordar los puntos que metió Lebron James en el último partido de las finales de 2016, cuando ganaron a Golden State (equipo con mejor récord de la historia) y remontaron una serie en la que iban perdiendo 1-3, y de momento es el único equipo que lo ha hecho a lo largo de la historia de la NBA.

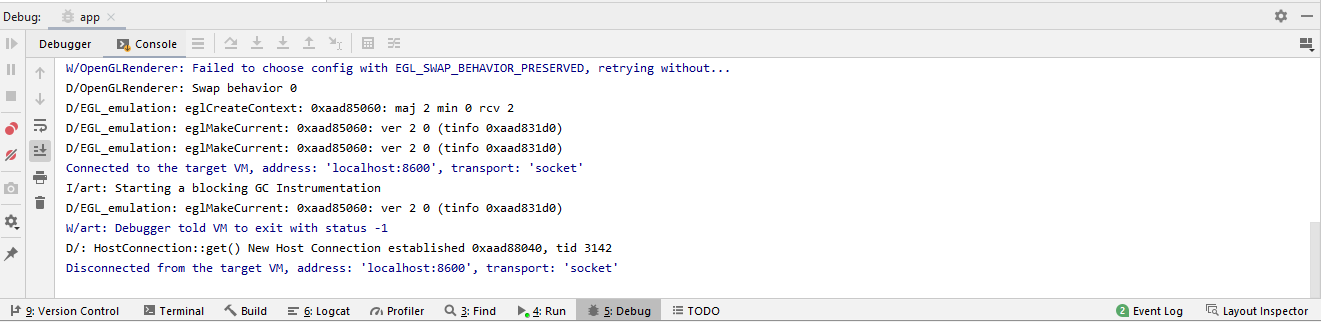
Ha debido de haber algún error de codificación en la aplicación ya que nos devuelve este mensaje:



Claramente ha habido algún error de codificación, pero es la excusa perfecta para hacer debug en Android Studio.



Gracias a esto lanzaremos una instancia del depurador que lanzará la actividad, y podremos monitorizar su ejecución y ver los resultados de la traza en la consola de depuración:



Para ello incluye un punto de break en el OnCreate() y ejecuta la aplicación, pulsando Step Over para ejecutar las instrucciones una a una. Una vez ejecutada responde las preguntas.

1. ¿Dónde se encuentra el error? Modifícalo y pon pantallazos de la traza de la consola de depuración y del error
2. Todas las frases utilizadas en la aplicación son recuperadas de un fichero que se llama strings.xml en la carpeta res/values. ¿Es una práctica común recuperar strings de un fichero en Android? Si lo es, pon algún ejemplo.
3. Crea dos variables en el fichero ***colores.xml*** y modifica los colores de los cuadros de texto, cada uno de un color. Usa el método ***getColor(R.color.XXX)***
4. Para escribir mensajes en la consola de depuración se ha utilizado una instancia de un objeto de la clase Log. Busca información y explica los diferentes tipos de mensajes que se pueden poner. ¿Podemos utilizar el *System.out.println()* en Android? Si es así, ¿dónde podemos observar los mensajes?
5. Reúne el código en un Componente nuevo, y crea un componente con la anotación @Preview. ¿Qué valores puedes modificar sobre el mismo?

## Ejercicio 3

A Lebron James le ha gustado tanto tu aplicación que le gustaría poder tenerla en su teléfono móvil.

Para ello tienes que investigar cómo se puede exportar la aplicación creada en tu IDE en un formato que sea instalable en su teléfono.

¿Cuál es el formato de una app? ¿Cuáles son los pasos para poder exportar la app? ¿Y para instalarlo en un móvil? Incluye capturas.

Ejercicio 4

Desde la compañía padre de Kotlin, se ha desarrollado una framework que permite la creación de aplicaciones híbridas (apps que funcionan en diferentes sistemas operativos a partir de un único desarrollo) usando Kotlin Multiplatform, un [framework](https://kotlinlang.org/docs/multiplatform.html) que se encuentra en los pasos finales de su beta (a mayo del 2023).

Como su versión no es estable, en el módulo trabajaremos únicamente en dispositivos Android, pero es interesante conocer su instalación y uso. Para ello, accede a la [documentación oficial](https://github.com/JetBrains/compose-multiplatform-template" \l "readme), y realiza los pasos necesarios para la instalación de la plantilla que contiene los ficheros necesario para el despliegue en escritorio, android e ios.

Os dejo un par de vídeos:

* [En inglés (oficial)](https://www.youtube.com/watch?v=5_W5YKPShZ4&ab_channel=KotlinbyJetBrains)
* [En español](https://www.youtube.com/watch?v=pA7fbaAMS3o&ab_channel=DevExpert-FormaciónAndroid%26Kotlin)

No hace falta desarrollar nada, únicamente instalar el proyecto de ejemplo y lanzar en Android, IOS y escritorio.

# Fecha de entrega

Hay que responder a todas las preguntas mencionando los links donde hemos obtenido la información y explicarlo lo mejor posible, así como incluir pantallazos donde requiera.

Hay que entregar este archivo con las preguntas respondidas en extensión .pdf con nombre de archivo *NombreYApellidos\_Practica1.pdf* y la fecha límite será el 30**/09/2023**.