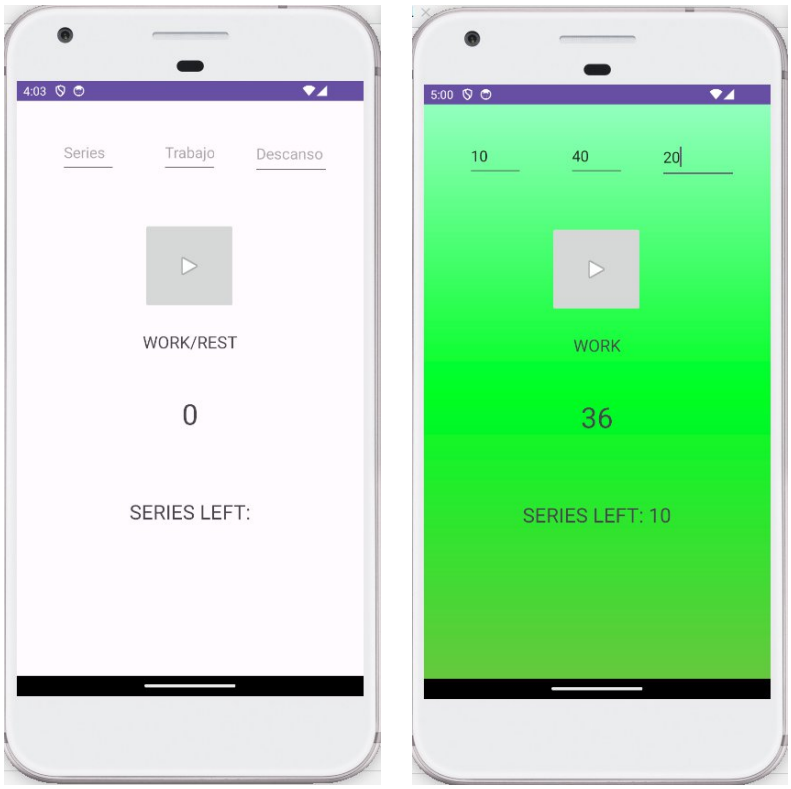
PRÁCTICA 1: El cronómetro TABATA

En esta primera práctica debes programar una aplicación sencillita consistente en una única Activity para familiarizarte con el entorno Android. Tu proyecto de Android Studio deberá llamarse PRC1_NombreApellido1 (con tu nombre y tu primer apellido).

ENUNCIADO

"Tabata" se refiere a un tipo específico de entrenamiento de intervalos de alta intensidad (HIIT), que lleva el nombre del Dr. Izumi Tabata, un científico del deporte japonés que desarrolló este método. El protocolo de Tabata implica series de ejercicios intensivos, que se dividen en N intervalos de X segundos de esfuerzo máximo, seguidos de Y segundos de descanso. Para poder hacer entrenamientos con el protocolo tabata vamos a diseñar una aplicación que nos muestre una actividad con la siguiente interfaz de usuario:



Esta interfaz de usuario consta de:

- 3 campos de edición de texto EditText de tipo Number, (Series, Trabajo y Descanso), un botón con imagen ImageButton con un icono ***de play*** para lanzar el entrenamiento y 3 campos de texto TextView, uno para mostrar si estás trabajando ***(work)*** o descansando (***rest),*** los segundos restantes y el número de series restantes.

La aplicación debe iniciar un entrenamiento para que, durante cada serie de trabajo, se ejecute un periodo de trabajo y acto seguido, un periodo de descanso. Por ejemplo, si el usuario selecciona 5 series de 40 segundos de trabajo y 20 segundos de trabajo, la aplicación deberá:

- Por cada una de las 5 series
  - Emitir un sonido “beep”
  - Lanzar una cuenta atrás de 40 segundos, cambiando la etiqueta WORK/REST a WORK y cambiar el fondo de pantalla a VERDE
  - Emitir un sonido “beep”
  - Lanzar una cuenta atrás de 20 segundos, cambiando la etiqueta WORK a REST y cambiar el fondo de pantalla a ROJO
  - Actualizar el texto con el número de series restantes SERIES LEFT X

– Al acabar la última serie, la aplicación emitirá un sonido “gong” y la etiqueta WORK/REST cambiará a FINISHED y el fondo volverá a ser blanco.

EJEMPLO DE FUNCIONAMIENTO

Puedes ver el siguiente vídeo de cómo quedaría la aplicación, con un ejemplo de la ejecución del programa, habiendo el usuario seleccionado 3 series, con 5 segundos de trabajo y 3 segundos de descanso:

https://youtu.be/Tw_C1Twzszk

¿QUÉ NECESITO SABER PARA PROGRAMAR ESTA APLICACIÓN?

Para poder trabajar con esta aplicación necesitas saber:

A) ~~Haber leído el primer tema para saber cómo crear un proyecto en Android y conocer el ciclo de vida de una actividad e inicializar los objetos que necesites en la función onCreate() de la Actividad principal y saber cómo registrar listeners para OnClick en componentes. En este caso, no tienes botones Button, pero tienes un ImageButton que se trata exactamente igual, al ser una clase derivada de Button.~~

B) ~~Haber trabajado con el ConstraintLayout y realizado el lab de Google que se proporciona en los recursos.~~

C) Trabajar con la clase CountDownTimer para cuentas atrás:

Para trabajar con cuentas atrás, Java y la suite de Android Studio te proporcionan la clase CountDownTimer:

La clase CountDownTimer en Android se utiliza para realizar una cuenta atrás sobre un intervalo de tiempo definido. Tiene dos métodos que debes sobrescribir:

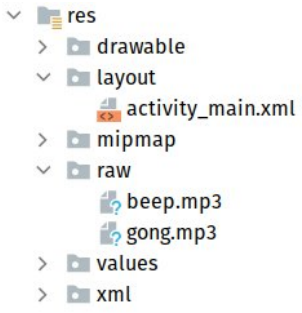
onTick(long millisUntilFinished): este método es llamado en intervalos regulares a lo largo del tiempo definido para la cuenta atrás. El parámetro millisUntilFinished indica la cantidad de milisegundos que quedan hasta que el temporizador termine.

onFinish(): este método es llamado cuando la cuenta atrás ha terminado, es decir, cuando el tiempo definido para el temporizador ha transcurrido.

Tienes disponible un proyecto de Android con un ejemplo de cómo hacer funcionar una cuenta atrás en el fichero **CountDownTimerExample.zip**.

D) ~~Saber reproducir sonidos: Para reproducir sonidos, deberás utilizar la clase MediaPlayer. Esta clase la trataremos más en profundidad en el tema 6, pero de momento, para poder reproducir un sonido tienes que hacer dos cosas, primero crear una instancia de un mediaplayer con el sonido a reproducir y luego, invocar a su método start. Por ejemplo, puedes crear la función playBeep() para invocarla cuando lo estimes oportuno:~~

```
public void playBeep(){
    Mediaplayer mp = MediaPlayer.create(
            getApplicationContext(), R.raw.beep);
    mp.start();
}
```



~~Fíjate en la constante que recibe el método create de MediaPlayer: R.raw.*cancion*. Esta referencia apunta a un recurso de la carpeta res “R”. Por tanto, deberás crear un directorio raw dentro de la carpeta de recursos y ahí, poner los sonidos que quieras reproducir.~~

~~Te facilitamos, para que realices la práctica, dos archivos mp3 “beep.mp3” y “gong.mp3”.~~

E) Saber cambiar el color de fondo de ConstraintLayout: Esta parte es muy sencilla, tan sólo tienes que ponerle un identificador al ConstraintLayout de tu actividad para poder referenciarla desde tu código fuente e invocar al método setBackgroundColor, como se muestra a continuación:

```
private ConstraintLayout constraintLayout;
constraintLayout = findViewById(R.id.constraintLayout);
constraintLayout.setBackgroundColor(Color.RED);
```

F) Si eres valiente, y quieres un reto fácil de conseguir, puedes utilizar en lugar del método setBackgroundColor, el método setBackgroundResource() para poner un degradado de un color determinado. Por ejemplo:

Define en la carpeta res ->drawable un recurso llamado degradado_verde.xml con la siguiente definición:

```
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <gradient
        android:type="linear"
        android:startColor="#66C83D"
        android:centerColor="#00FF23"
        android:endColor="#93FFBF"
        android:angle="90"/>
</shape>
```

Y desde código fuente, realizar:

```
private ConstraintLayout constraintLayout;
constraintLayout = findViewById(R.id.constraintLayout);
constraintLayout.setBackgroundResource(R.drawable.degradado_verde);
```

REQUISITOS:

- La aplicación tiene una única actividad con todos los componentes de la interfaz de usuario mencionados en el enunciado y funciona con el comportamiento descrito.
- La aplicación trata excepciones, por ejemplo, que el número de series no se haya especificado.
- La aplicación funciona sin errores.
- Para entregar la aplicación se utilizará la opción de Android Studio File->export->export to zip file.