

**Nombre:** Iosef Yamil Ramírez Vega **Grado y Grupo:** 9 B IDGS **Materia:** Desarrollo de Dispositivos Inteligentes  
**Reporte Técnico: Cronómetro con cuenta regresiva en Jetpack Compose para Wear OS**

**Reporte Técnico: Cronómetro con cuenta regresiva en Jetpack Compose para Wear OS**

**Objetivo del Código**

La finalidad de este código es implementar una **aplicación de cronómetro híbrido** para dispositivos con **Wear OS**. Esta app permite tanto:

* **Cronometrar en ascenso** (modo clásico),
* Como realizar **cuenta regresiva** a partir de 15, 20 o 30 minutos.

**Estructura del Código**

**Clase Principal: MainActivity**

* Hereda de ComponentActivity.
* Dentro de onCreate(), configura la interfaz de usuario utilizando Jetpack Compose y un **tema personalizado (CronometroTheme)**.

kotlin

setContent { CronometroTheme { ... } }

**Componentes Composables**

**StopwatchScreen()**

Es la función principal que dibuja la interfaz del cronómetro y maneja su lógica interna.

**Estados (usando remember)**

* isRunning: Booleano que indica si el cronómetro está activo.
* isCountdownMode: Booleano que indica si se trata de una cuenta regresiva.
* elapsedTime: Tiempo transcurrido para el cronómetro clásico.
* remainingTime: Tiempo restante para el modo cuenta regresiva.
* countdownTarget: Tiempo objetivo en milisegundos para el temporizador.

**Lógica de Tiempo**

Dentro de un LaunchedEffect, se ejecuta un bucle que actualiza el tiempo cada 50 milisegundos usando SystemClock.elapsedRealtime().

kotlin

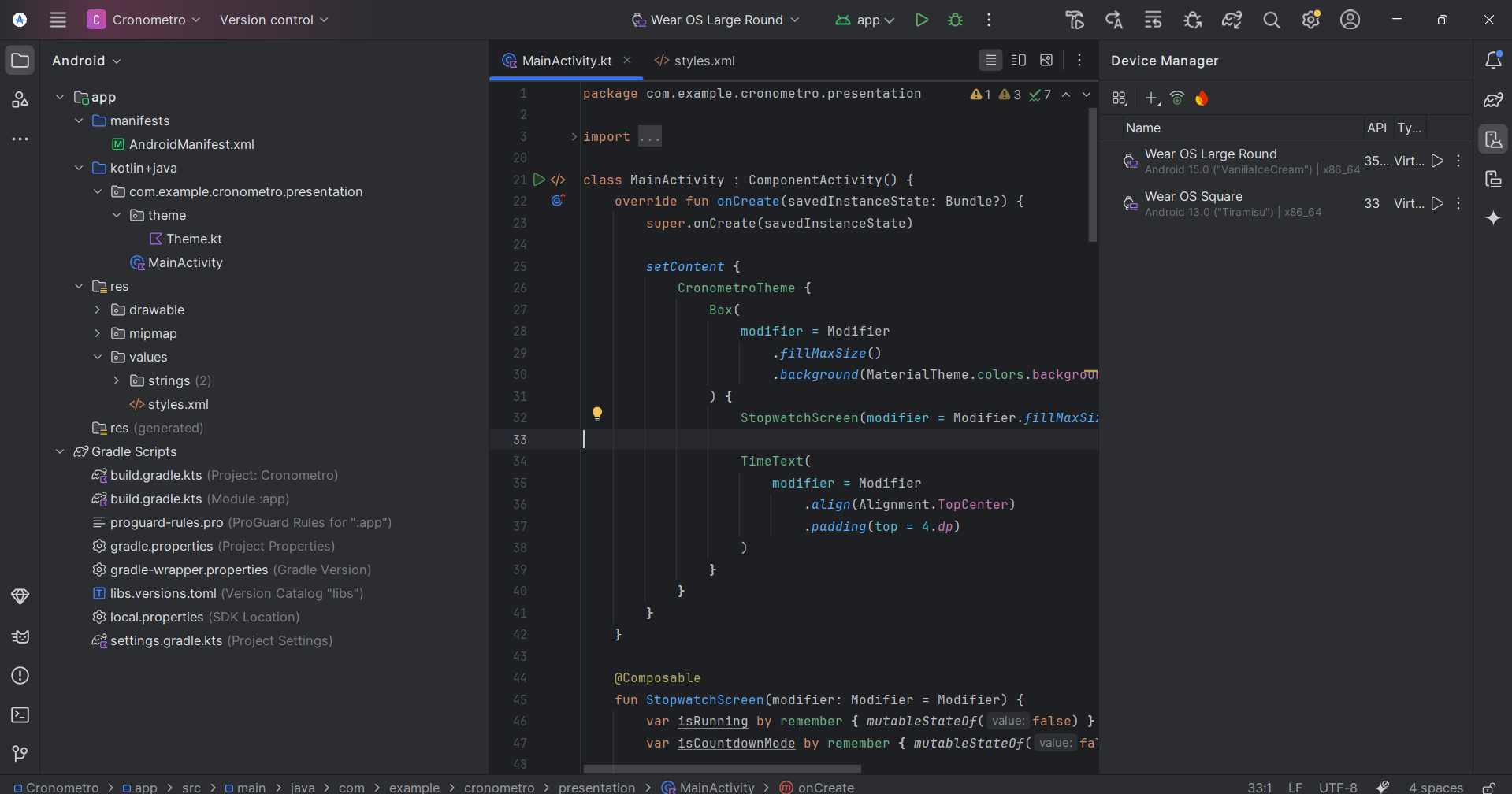
LaunchedEffect(isRunning, isCountdownMode) { ... }

Dependiendo del modo activo:

* **Modo cronómetro ascendente**: elapsedTime = now - baseTime
* **Modo cuenta regresiva**: remainingTime = countdownTarget - (now - baseTime)

Si remainingTime <= 0, se detiene el cronómetro.

package com.example.cronometro.presentation  
  
import android.os.Bundle  
import android.os.SystemClock  
import androidx.activity.ComponentActivity  
import androidx.activity.compose.setContent  
import androidx.compose.foundation.background  
import androidx.compose.foundation.layout.\*  
import androidx.compose.runtime.\*  
import androidx.compose.ui.Alignment  
import androidx.compose.ui.Modifier  
import androidx.compose.ui.graphics.Color  
import androidx.compose.ui.tooling.preview.Preview  
import androidx.compose.ui.unit.dp  
import androidx.compose.ui.unit.sp  
import androidx.wear.compose.material.\*  
import androidx.wear.tooling.preview.devices.WearDevices  
import com.example.cronometro.presentation.theme.CronometroTheme  
import kotlinx.coroutines.delay  
  
class MainActivity : ComponentActivity() {  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
  
 *setContent* **{** CronometroTheme **{** Box(  
 modifier = Modifier  
 .*fillMaxSize*()  
 .*background*(MaterialTheme.colors.background)  
 ) **{** StopwatchScreen(modifier = Modifier.*fillMaxSize*())  
  
 TimeText(  
 modifier = Modifier  
 .*align*(Alignment.TopCenter)  
 .*padding*(top = 4.*dp*)  
 )  
 **}  
 }  
 }** }  
  
 @Composable  
 fun StopwatchScreen(modifier: Modifier = Modifier) {  
 var isRunning by remember **{** *mutableStateOf*(false) **}** var isCountdownMode by remember **{** *mutableStateOf*(false) **}** var elapsedTime by remember **{** *mutableStateOf*(0L) **}** var remainingTime by remember **{** *mutableStateOf*(0L) **}** var countdownTarget by remember **{** *mutableStateOf*(0L) **}** var countdownButtonsEnabled by remember **{** *mutableStateOf*(true) **}** var selectedCountdown by remember **{** *mutableStateOf*<Int?>(null) **}** LaunchedEffect(isRunning, isCountdownMode) **{** if (isRunning) {  
 val baseTime = SystemClock.elapsedRealtime()  
 while (isRunning) {  
 val now = SystemClock.elapsedRealtime()  
 if (isCountdownMode) {  
 remainingTime = countdownTarget - (now - baseTime)  
 if (remainingTime <= 0L) {  
 remainingTime = 0L  
 isRunning = false  
 }  
 } else {  
 elapsedTime = now - baseTime  
 }  
 delay(50L)  
 }  
 }  
 **}** val displayTime = if (isCountdownMode) remainingTime else elapsedTime  
 val minutes = (displayTime / 60\_000).toInt()  
 val seconds = ((displayTime % 60\_000) / 1\_000).toInt()  
 val centis = ((displayTime % 1\_000) / 10).toInt()  
 val timeString = String.*format*("%02d:%02d:%02d", minutes, seconds, centis)  
  
 Box(  
 modifier = modifier.*padding*(8.*dp*),  
 contentAlignment = Alignment.Center  
 ) **{** Column(  
 horizontalAlignment = Alignment.CenterHorizontally,  
 verticalArrangement = Arrangement.spacedBy(12.*dp*)  
 ) **{** Text(  
 text = timeString,  
 fontSize = 36.*sp* )  
  
 Row(horizontalArrangement = Arrangement.spacedBy(8.*dp*)) **{** Button(onClick = **{** isRunning = !isRunning  
 **}**) **{** Text(  
 text = if (isRunning) "Detener" else "Iniciar",  
 fontSize = 12.*sp* )  
 **}** Button(onClick = **{** isRunning = false  
 isCountdownMode = false  
 elapsedTime = 0L  
 remainingTime = 0L  
 countdownTarget = 0L  
 countdownButtonsEnabled = true  
 selectedCountdown = null  
 **}**) **{** Text(text = "Reiniciar", fontSize = 12.*sp*)  
 **}  
 }** // Botones de cuenta regresiva  
 Row(horizontalArrangement = Arrangement.spacedBy(8.*dp*)) **{** *listOf*(15, 20, 30).*forEach* **{** min **->** val isSelected = selectedCountdown == min  
 Button(  
 onClick = **{** isCountdownMode = true  
 countdownTarget = min \* 60\_000L  
 remainingTime = countdownTarget  
 isRunning = true  
 countdownButtonsEnabled = false  
 selectedCountdown = min  
 **}**,  
 enabled = countdownButtonsEnabled || isSelected,  
 colors = ButtonDefaults.buttonColors(  
 backgroundColor = if (isSelected) Color.Green else MaterialTheme.colors.primary  
 )  
 ) **{** Text("$min", fontSize = 12.*sp*)  
 **}  
 }  
 }  
 }  
 }** }  
  
 @Preview(  
 showSystemUi = true,  
 showBackground = true,  
 device = WearDevices.SMALL\_ROUND  
 )  
 @Composable  
 fun Preview\_StopwatchScreen() {  
 CronometroTheme **{** Box(  
 modifier = Modifier  
 .*fillMaxSize*()  
 .*background*(MaterialTheme.colors.background)  
 ) **{** StopwatchScreen(modifier = Modifier.*fillMaxSize*())  
  
 TimeText(  
 modifier = Modifier  
 .*align*(Alignment.TopCenter)  
 .*padding*(top = 4.*dp*)  
 )  
 **}  
 }** }  
}



**Interfaz de Usuario**

****

**Tiempo mostrado**

kotlin

val timeString = String.format("%02d:%02d:%02d", minutos, segundos, centisegundos)

Se muestra el tiempo en formato MM:SS:CC (minutos, segundos, centésimas).

**Botones disponibles**

* **Iniciar / Detener**: Alterna entre correr y pausar.
* **Reiniciar**: Resetea todo el cronómetro a 0.
* **Cuenta regresiva**:
  + Botones etiquetados como 15, 20, 30, que inician la cuenta regresiva con esa cantidad de minutos.

kotlin

listOf(15, 20, 30).forEach { min -> ... }

