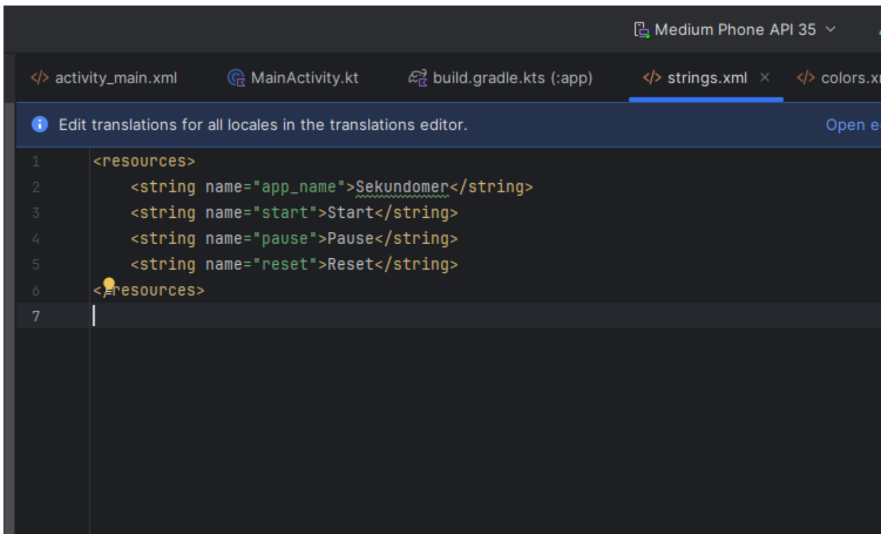
```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center"
    android:padding="16dp"
    android:background="@color/background"
    tools:context=".MainActivity">

    <Chronometer
        android:id="@+id/textTime"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="36sp"
        android:layout_marginBottom="24dp"
        android:textColor="@color/black"/>

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <Button
            android:id="@+id/btnPause"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/pause"
            android:backgroundTint="@color/orange"
            android:layout_margin="8dp"/>

    </LinearLayout>

    <Button
        android:id="@+id/btnStart"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="8dp"
        android:backgroundTint="@color/green"
        android:text="@string/start" />

    <Button
        android:id="@+id/btnReset"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="8dp"
        android:backgroundTint="@color/red"
        android:text="@string/reset" />

</LinearLayout>
```

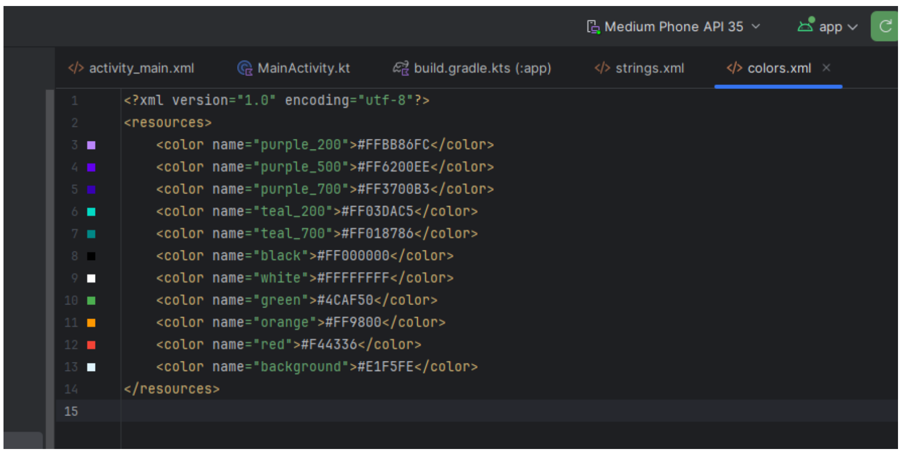
```kotlin
package com.example.tutinalitvin

import android.os.Bundle
import android.os.SystemClock
import android.widget.Button
import android.widget.Chronometer
import androidx.appcompat.app.AppCompatActivity
//import com.example.tutinalitvin.R
```

```
class MainActivity : AppCompatActivity() {
    private lateinit var chronometr: Chronometer
    private var running = false
    private var offset: Long = 0

    // Ключи для сохранения состояния
    private val OFFSET_KEY = "offset"
    private val RUNNING_KEY = "running"
    private val BASE_KEY = "base"

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main) // Убедитесь, что файл
activity_main.xml существует

        // Инициализация элементов
        chronometr = findViewById(R.id.textTime)
        val btnStart = findViewById<Button>(R.id.btnStart)
        val btnPause = findViewById<Button>(R.id.btnPause)
        val btnReset = findViewById<Button>(R.id.btnReset)

        // Восстановление состояния при повороте экрана
        if (savedInstanceState != null) {
            offset = savedInstanceState.getLong(OFFSET_KEY)
            running = savedInstanceState.getBoolean(RUNNING_KEY)
            if (running) {
                chronometr.base = savedInstanceState.getLong(BASE_KEY)
                chronometr.start()
            } else {
                setBaseTime()
            }
        } else {
            setBaseTime()
        }

        // Обработчики кнопок
        btnStart.setOnClickListener {
            if (!running) {
                setBaseTime()
                chronometr.start()
                running = true
            }
        }

        btnPause.setOnClickListener {
            if (running) {
                saveOffset()
                chronometr.stop()
                running = false
            }
        }

        btnReset.setOnClickListener {
            offset = 0
            setBaseTime()
            running = false
        }
    }

    // Сохранение состояния перед уничтожением активности
    override fun onSaveInstanceState(outState: Bundle) {
        super.onSaveInstanceState(outState)
        outState.putLong(OFFSET_KEY, offset)
        outState.putBoolean(RUNNING_KEY, running)
        outState.putLong(BASE_KEY, chronometr.base)
    }

    // Сохранение текущего смещения времени
    private fun saveOffset() {
        offset = SystemClock.elapsedRealtime() - chronometr.base
    }

    // Установка базового времени с учетом смещения
    private fun setBaseTime() {
        chronometr.base = SystemClock.elapsedRealtime() - offset
    }
}
```

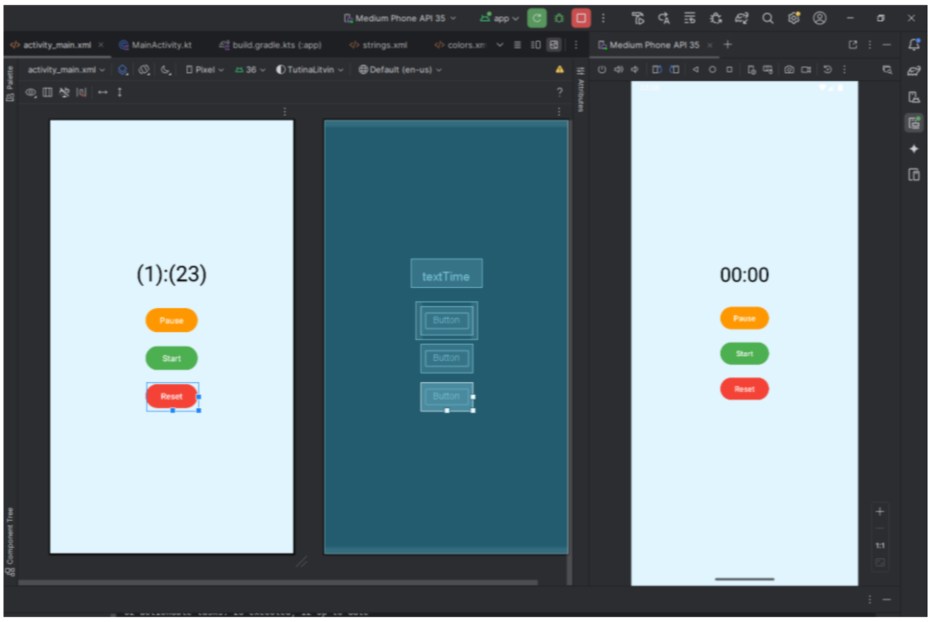


Edit translations for all locales in the translations editor.

```xml
<resources>
    <string name="app_name">Sekundomer</string>
    <string name="start">Start</string>
    <string name="pause">Pause</string>
    <string name="reset">Reset</string>
</resources>
```



```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="purple_200">#FFBB86FC</color>
    <color name="purple_500">#FF6200EE</color>
    <color name="purple_700">#FF3700B3</color>
    <color name="teal_200">#FF03DAC5</color>
    <color name="teal_700">#FF018786</color>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
    <color name="green">#4CAF50</color>
    <color name="orange">#FF9800</color>
    <color name="red">#F44336</color>
    <color name="background">#E1F5FE</color>
</resources>
```


Medium Phone API 35
app
activity_main.xml
MainActivity.kt
build.gradle.kts (:app)
strings.xml
colors.xm
Medium Phone API 35
activity_main.xml
Pixel
36
TutinaLitvin
Default (en-us)
Palette
Attributes
Component Tree
(1):(23)
Pause
Start
Reset
textTime
Button
Button
Button
00:00
Pause
Start
Reset
1:1