# Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

**Национальный исследовательский**

**Томский политехнический университет**

Инженерная школа информационных технологий и робототехники Отделение информационных технологий

Отчет по лабораторной работе №21 по дисциплине

**«Мобильная разработка. Хранение данных и сетевое взаимодействие»**

Сервисы

Выполнил:

Студент группы 8B21 П. А. Жердева

Проверил:

Ст. преп. ОИТ ИШИТР В. А. Дорофеев

# Задание

Напишите программу-таймер. Интерфейс программы должен выглядеть примерно так:

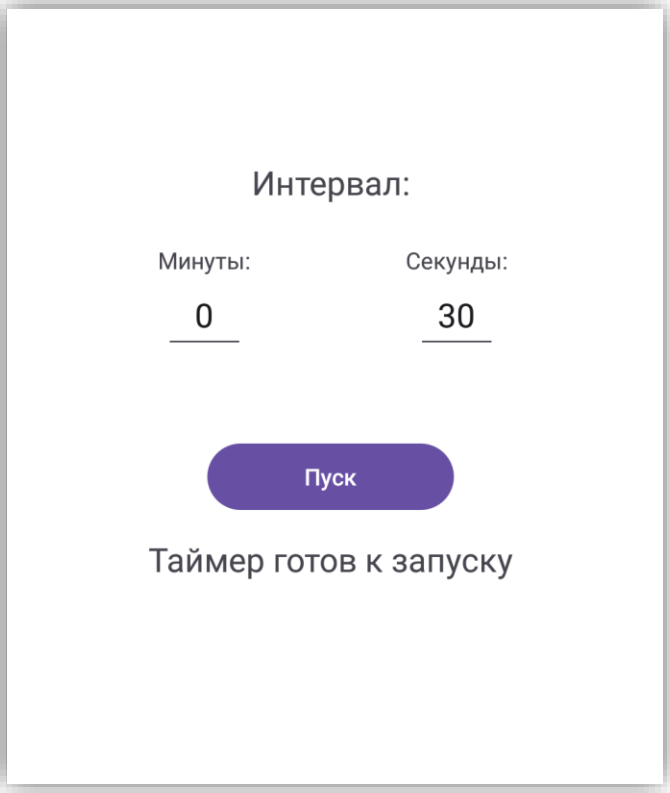


Рис.1. Примерный внешний вид приложения

Пользователь может ввести числовые значения в поля минут и секунд. При нажатии кнопки «Пуск» начинается обратный отсчёт с тех значений, которые введены в поля минут и секунд, до нуля. На время отсчёта поля ввода блокируются, кнопка «Пуск» становится кнопкой «Стоп», а вместо надписи «Таймер готов к запуску» отображается оставшееся время:

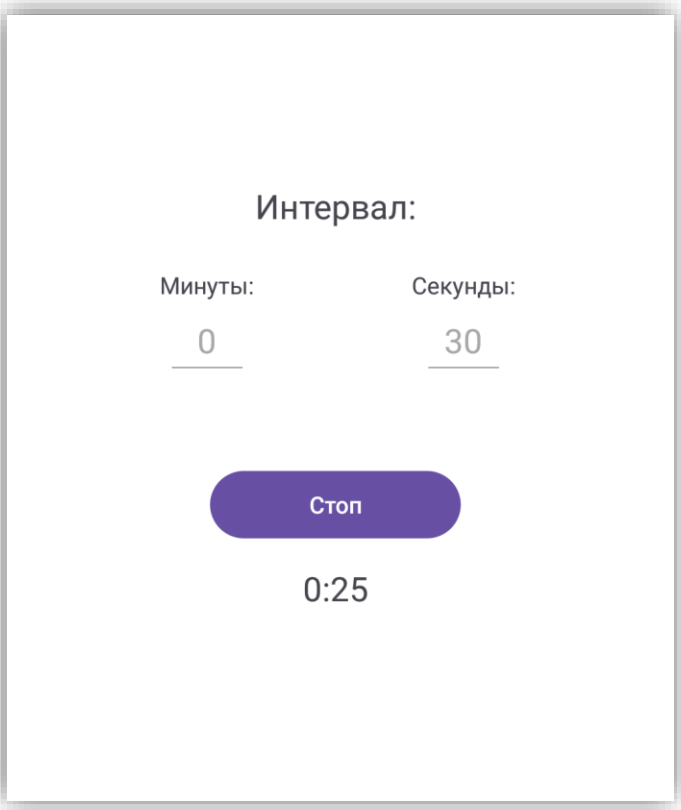


Рис. 2. Внешний вид экрана после запуска таймера

Одновременно с этим отображается уведомление, в котором также идёт обратный отсчёт:

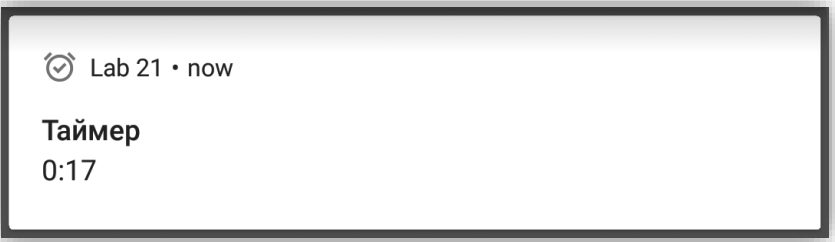


Рис. 3. Уведомление с обратным отсчетом

По окончании отсчёта программа возвращается в исходное состояние, готовая к новому запуску. Если в процессе отсчёта пользователь закрывает программу, отсчёт должен продолжаться, уведомление должно оставаться активным (вместе с продолжающимся отсчётом), и при повторном запуске программы – отсчёт в окне программы также должен быть продолжен. При реализации программы необходимо использовать сервис переднего плана, реализация отсчёта и уведомлений должны быть сделаны в сервисе.

**Ход выполнения**

Для работы были указаны несколько разрешений в файле AndroidManifest.xml:

1. <uses-permission android:name="android.permission.POST\_NOTIFICATIONS"/> - разрешение на отправку уведомлений пользователю
2. <uses-permission android:name="android.permission.FOREGROUND\_SERVICE" /> - разрешение на работу сервиса в переднем плане
3. <uses-permission android:name="android.permission.FOREGROUND\_SERVICE\_DATA\_SYNC" /> - уточняющее разрешение, нужно для foreground-сервисов, выполняющих синхронизацию данных или таймеры/планировщики, которые продолжают работу в фоне

**Как работает приложение:**

1. позволяет ввести минуты и секунды.
2. Запускает отсчёт по нажатию на кнопку «Пуск».
3. Блокирует поля ввода на время работы таймера.
4. Отображает текущее время на экране и в уведомлении.
5. Работает в фоне даже после закрытия приложения — благодаря Foreground Service (сервису переднего плана).
6. Уведомление не пропадает, пока таймер не завершится или не будет остановлен.

Таблица 1. Структура проекта

|  |  |
| --- | --- |
| **Файл** | **Назначение** |
| **MainActivity.kt** | Основная активность (UI и логика взаимодействия пользователя с сервисом). |
| **TimerService.kt** | Сервис переднего плана — выполняет обратный отсчёт, отправляет данные в активность и обновляет уведомление. |
| **activity\_main.xml** | Разметка пользовательского интерфейса (поля ввода, кнопка, текст). |

**MainActivity.kt -** **главная активность приложения**, которая

1. Отображает поля ввода минут и секунд.
2. Запускает или останавливает таймер. Блокирует ввод на время работы таймера.
3. Принимает сообщения от TimerService и обновляет экран. Работа с этим сервисом происходит при нажатии кнопки «Пуск»/ «Стоп»:

* Проверяется разрешение на уведомления (POST\_NOTIFICATIONS).
* Если разрешение есть — вызывается toggleTimer().
* Если таймер не запущен — создаётся Intent с действием ACTION\_START\_TIMER и передаётся в TimerService.
* Если таймер уже работает — посылается ACTION\_STOP\_TIMER и сервис останавливается.

**TimerService.kt** - это Foreground Service (сервис переднего плана) — главный исполнитель логики таймера.

Что делает:

1. Получает команду (ACTION\_START\_TIMER или ACTION\_STOP\_TIMER).
2. Если запущен — создаёт уведомление и начинает отсчёт с помощью CountDownTimer.
3. Каждую секунду:

* Отправляет широковещательное сообщение с оставшимся временем (ACTION\_TIMER\_TICK);
* Обновляет текст уведомления.
* Когда отсчёт заканчивается — останавливает себя.

**Текст программы**

**MainActivity.kt:**

package ru.polinazherdeva.lr21  
  
import android.Manifest  
import android.content.BroadcastReceiver  
import android.content.Context  
import android.content.Intent  
import android.content.IntentFilter  
import android.content.pm.PackageManager  
import android.os.Build  
import android.os.Bundle  
import android.widget.Button  
import android.widget.EditText  
import android.widget.TextView  
import android.widget.Toast  
import androidx.activity.result.contract.ActivityResultContracts  
import androidx.appcompat.app.AppCompatActivity  
import androidx.core.content.ContextCompat  
  
class MainActivity : AppCompatActivity() {  
  
 private lateinit var timerText: TextView  
 private lateinit var startButton: Button  
 private lateinit var minutesInput: EditText  
 private lateinit var secondsInput: EditText  
 private var timerRunning = false *// флаг — запущен ли таймер  
  
 // Приёмник сообщений от службы TimerService (каждую секунду получает оставшееся время)* private val timerBroadcastReceiver = object : BroadcastReceiver() {  
 override fun onReceive(context: Context?, intent: Intent?) {  
 if (intent?.*action* == TimerService.ACTION\_TIMER\_TICK) {  
 val millisLeft = intent.getLongExtra(TimerService.EXTRA\_TIME\_LEFT, 0L)  
 if (millisLeft > 0) {  
 timerText.*text* = formatTime(millisLeft) *// обновляем текст с оставшимся временем* } else {  
 timerText.*text* = getString(R.string.*timer\_ready*) *// выводим "таймер готов"* timerRunning = false  
 startButton.*text* = getString(R.string.*start\_button*)  
 setInputsEnabled(true) *// разблокируем поля после окончания таймера* }  
 }  
 }  
 }  
  
 *// Запрос разрешения на уведомления* private val requestNotificationPermission =  
 registerForActivityResult(ActivityResultContracts.RequestPermission()) **{** granted **->** if (granted) toggleTimer() *// если разрешение дано — запускаем таймер* else Toast.makeText(this, "Нужны разрешения для уведомлений", Toast.*LENGTH\_SHORT*).show()  
 **}** override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_main*) *// подключаем разметку  
  
 // Находим элементы интерфейса* timerText = findViewById(R.id.*timerText*)  
 startButton = findViewById(R.id.*startButton*)  
 minutesInput = findViewById(R.id.*minutesInput*)  
 secondsInput = findViewById(R.id.*secondsInput*)  
  
 *// Обработка нажатия на кнопку* startButton.setOnClickListener **{** checkNotificationPermissionAndToggle() **}** }  
  
 *// Проверка разрешения на уведомления и запуск/остановка таймера* private fun checkNotificationPermissionAndToggle() {  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*TIRAMISU*) {  
 *// Если разрешения нет — запрашиваем его* if (checkSelfPermission(Manifest.permission.*POST\_NOTIFICATIONS*) != PackageManager.*PERMISSION\_GRANTED*) {  
 requestNotificationPermission.launch(Manifest.permission.*POST\_NOTIFICATIONS*)  
 } else {  
 toggleTimer() *// иначе — запускаем таймер* }  
 } else {  
 toggleTimer() *// для старых версий Android разрешения не нужны* }  
 }  
  
 *// Запуск или остановка таймера* private fun toggleTimer() {  
 if (!timerRunning) {  
 *// Запуск таймера* val minutes = minutesInput.*text*.toString().*toLongOrNull*() ?: 0  
 val seconds = secondsInput.*text*.toString().*toLongOrNull*() ?: 0  
 val totalMillis = (minutes \* 60 + seconds) \* 1000  
  
 *// Проверяем, что введено корректное время* if (totalMillis <= 0) {  
 Toast.makeText(this, "Введите время больше 0", Toast.*LENGTH\_SHORT*).show()  
 return  
 }  
 *// Создаём Intent для запуска сервиса* val intent = Intent(this, TimerService::class.*java*).*apply* **{** *action* = TimerService.ACTION\_START\_TIMER  
 putExtra(TimerService.EXTRA\_TIME\_LEFT, totalMillis)  
 **}** ContextCompat.startForegroundService(this, intent) *// запускаем сервис переднего плана* timerRunning = true  
 startButton.*text* = getString(R.string.*stop\_button*) *// меняем текст кнопки на стоп* setInputsEnabled(false) *// блокируем поля во время работы таймера* } else {  
 *// Остановка таймера* val intent = Intent(this, TimerService::class.*java*).*apply* **{** *action* = TimerService.ACTION\_STOP\_TIMER  
 **}** startService(intent) *// отправляем сигнал сервису* timerRunning = false  
 startButton.*text* = getString(R.string.*start\_button*)  
 timerText.*text* = getString(R.string.*timer\_ready*)  
 setInputsEnabled(true) *// разблокируем поля после остановки* }  
 }  
  
 *// Форматирование времени в мм:сс* private fun formatTime(millis: Long): String {  
 val totalSeconds = millis / 1000  
 val minutes = totalSeconds / 60  
 val seconds = totalSeconds % 60  
 return String.*format*("%02d:%02d", minutes, seconds)  
 }  
  
 *// Регистрация приёмника* override fun onStart() {  
 super.onStart()  
 ContextCompat.registerReceiver(  
 this,  
 timerBroadcastReceiver, *// указываем приёмник* IntentFilter(TimerService.ACTION\_TIMER\_TICK), *// фильтр действий* ContextCompat.*RECEIVER\_EXPORTED // приёмник доступен извне* )  
 }  
  
 *// Отмена регистрации приёмника* override fun onStop() {  
 super.onStop()  
 unregisterReceiver(timerBroadcastReceiver)  
 }  
  
 *// Метод для включения/отключения полей ввода* private fun setInputsEnabled(enabled: Boolean) {  
 minutesInput.*isEnabled* = enabled  
 secondsInput.*isEnabled* = enabled  
 }  
}

**TimeService.kt:**

package ru.polinazherdeva.lr21  
  
import android.app.NotificationChannel  
import android.app.Service  
import android.content.Intent  
import android.os.Build  
import android.os.CountDownTimer  
import android.os.IBinder  
import androidx.core.app.NotificationCompat  
import androidx.core.app.NotificationManagerCompat  
  
*// Сервис — компонент, выполняющий задачу в фоне*class TimerService : Service() {  
  
 *// companion object — хранит константы, общие для всего приложения* companion object {  
 const val ACTION\_START\_TIMER = "ACTION\_START\_TIMER" *// действие для запуска таймера* const val ACTION\_STOP\_TIMER = "ACTION\_STOP\_TIMER" *// действие для остановки таймера* const val ACTION\_TIMER\_TICK = "ACTION\_TIMER\_TICK" *// действие — событие «тик» (каждую секунду)* const val EXTRA\_TIME\_LEFT = "EXTRA\_TIME\_LEFT" *// ключ для передачи оставшегося времени* const val CHANNEL\_ID = "timer\_channel" *// ID канала уведомлений* const val NOTIFICATION\_ID = 1 *// ID уведомления* }  
  
 private var timer: CountDownTimer? = null *// объект обратного отсчёта* private var remainingTime: Long = 60000L *// оставшееся время (по умолчанию 60 сек)* private lateinit var notificationBuilder: NotificationCompat.Builder *// конструктор уведомлений  
  
 // Основной метод запуска сервиса. Срабатывает при вызове startForegroundService() или startService()* override fun onStartCommand(intent: Intent?, flags: Int, startId: Int): Int {  
 when (intent?.*action*) { *// проверяем, какое действие пришло в Intent* ACTION\_START\_TIMER -> { *// если действие — "запустить таймер"* remainingTime = intent.getLongExtra(EXTRA\_TIME\_LEFT, 60000L) *// получаем время из Intent* startForegroundServiceSafe() *// запускаем сервис в переднем плане с уведомлением* startCountDown() *// запускаем сам отсчёт* }  
 ACTION\_STOP\_TIMER -> { *// если действие — "остановить таймер"* stopTimer() *// останавливаем таймер* }  
 }  
 return *START\_STICKY // START\_STICKY — если система убьёт сервис, она перезапустит его без нового Intent* }  
  
 *// Метод, который создаёт уведомление и переводит сервис в режим переднего плана* private fun startForegroundServiceSafe() {  
 val manager = NotificationManagerCompat.from(this) *// менеджер уведомлений* createNotificationChannel(manager) *// создаём канал уведомлений  
  
 // Настраиваем уведомление* notificationBuilder = NotificationCompat.Builder(this, CHANNEL\_ID)  
 .setContentTitle("Таймер") *// заголовок уведомления* .setContentText(formatTime(remainingTime)) *// текст — оставшееся время* .setSmallIcon(R.drawable.*ic\_art*) *// иконка* .setOnlyAlertOnce(true) *// не издавать звук при каждом обновлении  
 // Запускаем сервис в режиме переднего плана* try {  
 startForeground(NOTIFICATION\_ID, notificationBuilder.build())  
 } catch (e: SecurityException) {  
 e.printStackTrace() *// обработка ошибки при отсутствии разрешений* }  
 }  
  
 *// Создаём канал уведомлений* private fun createNotificationChannel(manager: NotificationManagerCompat) {  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*O*) {  
 val channel = NotificationChannel(  
 CHANNEL\_ID, *// ID канала* "Таймер", *// имя канала* android.app.NotificationManager.*IMPORTANCE\_LOW // низкий приоритет* )  
 manager.createNotificationChannel(channel) *// регистрируем канал в системе* }  
 }  
  
 *// Метод, запускающий обратный отсчёт* private fun startCountDown() {  
 timer?.cancel() *// если предыдущий таймер существовал — отменяем его* timer = object : CountDownTimer(remainingTime, 1000) { *// создаём таймер с шагом 1 сек* override fun onTick(millisUntilFinished: Long) {  
 remainingTime = millisUntilFinished *// сохраняем оставшееся время  
  
 // Обновляем текст в уведомлении* notificationBuilder.setContentText("Осталось: ${formatTime(remainingTime)}")  
 try {  
 NotificationManagerCompat.from(this@TimerService)  
 .notify(NOTIFICATION\_ID, notificationBuilder.build()) *// обновляем уведомление* } catch (e: SecurityException) { e.printStackTrace() }  
  
 *// Отправляем широковещательное сообщение (Broadcast) в MainActivity* sendBroadcast(Intent(ACTION\_TIMER\_TICK).*apply* **{** putExtra(EXTRA\_TIME\_LEFT, remainingTime) *// передаём оставшееся время* **}**)  
 }  
 *// Действия после завершения отсчёта* override fun onFinish() {  
 remainingTime = 0L  
 *// Отправляем сообщение о том, что время вышло* sendBroadcast(Intent(ACTION\_TIMER\_TICK).*apply* **{** putExtra(EXTRA\_TIME\_LEFT, remainingTime)  
 **}**)  
 stopTimer() *// останавливаем сервис* }  
 }.start() *// запускаем таймер* }  
  
 *// Остановка таймера* private fun stopTimer() {  
 timer?.cancel() *// останавливаем обратный отсчёт* stopForegroundService() *// снимаем сервис с переднего плана* }  
  
 *// Остановка сервиса переднего плана и завершение его работы* private fun stopForegroundService() {  
 try {  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*Q*) {  
 stopForeground(*STOP\_FOREGROUND\_REMOVE*)  
 } else {  
 @Suppress("DEPRECATION")  
 stopForeground(true)  
 }  
 } catch (e: SecurityException) {  
 e.printStackTrace()  
 }  
 stopSelf() *// полностью останавливаем сервис* }  
  
 *// Форматируем миллисекунды в строку "мм:сс"* private fun formatTime(millis: Long): String {  
 val totalSeconds = millis / 1000  
 val minutes = totalSeconds / 60  
 val seconds = totalSeconds % 60  
 return String.*format*("%02d:%02d", minutes, seconds)  
 }  
  
 override fun onBind(intent: Intent?): IBinder? = null *// Сервис не поддерживает привязку, поэтому возвращаем null*}

**activity\_main.xml:**

<LinearLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 android:orientation="vertical"  
 android:padding="16dp"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:gravity="center\_horizontal">  
  
 <TextView  
 android:text="@string/interval\_label"  
 android:textSize="26sp"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:gravity="center"  
 android:layout\_marginBottom="16dp"/>  
  
 <LinearLayout  
 android:orientation="horizontal"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="8dp">  
  
 <LinearLayout  
 android:orientation="vertical"  
 android:layout\_weight="1"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content">  
  
 <TextView  
 android:text="@string/minutes\_label"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"/>  
  
 <EditText  
 android:id="@+id/minutesInput"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="number"  
 android:minHeight="48dp"  
 android:text="0" />  
 </LinearLayout>  
  
 <LinearLayout  
 android:orientation="vertical"  
 android:layout\_weight="1"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="16dp">  
  
 <TextView  
 android:text="@string/seconds\_label"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"/>  
  
 <EditText  
 android:id="@+id/secondsInput"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="number"  
 android:minHeight="48dp"  
 android:text="0" />  
 </LinearLayout>  
 </LinearLayout>  
  
 <Button  
 android:id="@+id/startButton"  
 android:text="@string/start\_button"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"/>  
  
 <TextView  
 android:id="@+id/timerText"  
 android:text="@string/timer\_ready"  
 android:textSize="24sp"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:gravity="center"  
 android:layout\_marginTop="16dp"/>  
</LinearLayout>

**Strings.xml:**

<resources>  
 <string name="app\_name">lr21</string>  
 <string name="interval\_label">Интервал:</string>  
 <string name="minutes\_label">Минуты:</string>  
 <string name="seconds\_label">Секунды:</string>  
 <string name="start\_button">Пуск</string>  
 <string name="stop\_button">Стоп</string>  
 <string name="timer\_ready">Таймер готов к запуску</string>  
</resources>

**Результат работы:**

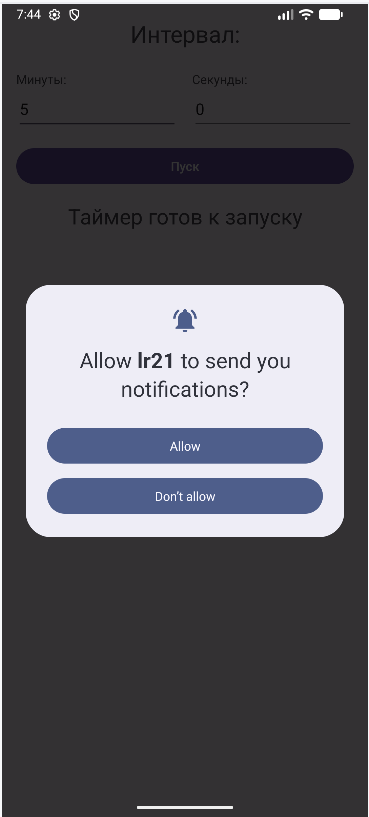
****

Рис. 4. Запрос разрешения на отправку уведомлений



Рис. 5. Главный экран приложения



Рис. 6. Запуск таймера

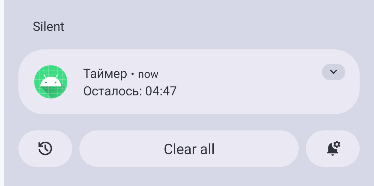


Рис. 7. Отображение обратного отсчета в уведомлении

# Выводы

В ходе лабораторной работы было создано Android-приложение «Таймер» с использованием сервиса переднего плана (Foreground Service).

Приложение позволяет задать время обратного отсчёта, запускать и останавливать таймер, отображая оставшееся время как в интерфейсе, так и в уведомлении.

Таймер продолжает работать при свёрнутом приложении, а после завершения отсчёта интерфейс возвращается в исходное состояние.

В работе реализованы уведомления, обработка разрешений и взаимодействие активности с сервисом через BroadcastReceiver.