4.5. Уведомления (Notification)

Сайт: Samsung Innovation Campus
Курс: Мобильная разработка на Kotlin
Книга: 4.5. Уведомления (Notification)

Напечатано:: Murad Rezvan
Дата: понедельник, 3 июня 2024, 17:55

Оглавление

1. Назначение и «анатомия» уведомлений

Кроме Toast-сообщений, в Android существует также другой тип уведомлений, который выводится в строке состояния в виде значка с кратким пояснением. Если открыть окно уведомлений, можно увидеть более подробную информацию об уведомлении. Когда пользователь открывает расширенное сообщение, Android запускается объект `Intent`, который определяется в соответствии с уведомлением. Можно также создавать уведомление с использованием звука, вибрации и т.д. Такой вид уведомлений используется в том случае, когда приложение работает в фоновом режиме и должно оповестить пользователя о каком-либо событии. Уведомление будет отображаться в строке состояния до тех пор, пока пользователь не отреагирует на него.

В Android существует множество разных типов и стилей уведомлений. В данном разделе приведен краткий обзор уведомлений и способов их использования. Прежде чем перейдем к различным типам уведомлений, рассмотрим структуру уведомлений в Android и их параметры, которые можно использовать для их настройки. Стиль и структура Android-уведомлений определяются системой и могут различаться в зависимости от версии ОС и производителя. Поэтому можно определять только содержание уведомлений, а не их внешний вид и дизайн.

Ниже приведены основные части уведомления:

- маленький значок: обязателен и устанавливается с помощью метода `setSmallIcon()`;
- большой значок: необязателен и устанавливается с помощью метода `setLargeIcon()`;
- заголовок: необязателен и устанавливается с помощью метода `setContentTitle()`;
- текст уведомления: необязательный и устанавливается с помощью метода `setContentText()`;
- имя приложения: предоставляется системой;
- Отметка времени: предоставляется системой, но может быть отменена с помощью метода `setWhen()`.

Встроенные в систему уведомления охватывают практически все возможные варианты уведомлений, однако существует возможность создания собственную разметку для уведомлений. Далее мы рассмотрим различные типы уведомлений и способы их реализации в Android-приложениях.

2. Класс NotificationCompat. Создание уведомления.

Как было отмечено ранее, уведомления (Notifications) предоставляют краткую информацию о событиях в вашем приложении, когда оно не используется.

Для создания уведомлений в Android существует класс NotificationCompat - помощник для доступа к функциям в уведомлений. API-интерфейсы класса `NotificationCompat` из библиотеки поддержки Android позволяют добавлять функции, доступные для версий Android 4.0 и выше.

Для создания уведомлений используется билдер класса `NotificationCompat`, в котором указываем иконку, заголовок и текст для уведомления. Уведомления могут отображаться в разных областях экрана и в разных форматах, например, в виде значка в строке состояния или более подробной записи в панели уведомлений. Поэтому мы рассмотрим различные виды уведомлений и их реализацию.

Уведомление в строке состояния

Это основной тип уведомления. Сначала оно отображается в виде значка, и его можно просмотреть более подробно с помощью панели уведомлений. Далее приведен код создания уведомления, в котором мы определяем небольшую иконку, заголовок и текст содержимого.

```
val CHANNEL_ID = "CHANNEL_ID"
val builder = NotificationCompat.Builder(this, CHANNEL_ID)
    .setSmallIcon(R.drawable.my_icon)
    .setContentTitle("My title")
    .setContentText("My content for this notification")
    .setPriority(NotificationCompat.PRIORITY_DEFAULT)
```

Обратите внимание, что нам нужно передать строку `CHANNEL_ID` конструктору NotificationCompat, чтобы убедиться, что наше уведомление совместимо с Android 8.0 или выше.

Всплывающее уведомление

Всплывающие уведомления на короткое время появляются в верхней части экрана в плавающем окне. Они появляются в тот момент, когда приложение отправляет запрос, и могут отображаются только если устройство разблокировано. В программном коде они не сильно отличаются от обычных уведомлений. Необходимо установить `Priority` значение `PRIORITY_HIGH` и запросить разрешение на вибрацию. В примере ниже функция отключается, поэтому не нужно запрашивать разрешение на вибрацию.

```
val builder = NotificationCompat.Builder(this, CHANNEL_ID)
    .setSmallIcon(R.drawable.my_icon)
    .setContentTitle("My title")
    .setContentText("My content for this notification")
    .setPriority(NotificationCompat.PRIORITY_HIGH)

if (Build.VERSION.SDK_INT >= 21) builder.setVibrate(LongArray(0))
```

Уведомление на экране блокировки

Для отображения уведомлений на экране блокировки устройства, можно использовать параметр `Visibility`. Он может принимать одно из следующих значений:

- `VISIBILITY_PUBLIC` показывает полное содержание уведомления
- `VISIBILITY_PRIVATE` показывает значок и заголовок.
- `VISIBILITY_SECRET` не показывает уведомление на экране блокировки.

Ниже приведен пример уведомления, которое отображается на экране блокировки.

```
val builder = NotificationCompat.Builder(this, CHANNEL_ID)
    .setSmallIcon(R.drawable.my_icon)
    .setContentTitle("My lock screen Notification")
    .setContentText("My content for this notification")
    .setPriority(NotificationCompat.PRIORITY_DEFAULT)
    .setVisibility(VISIBILITY_PUBLIC)
```

Мы также можем создать уведомление, в котором скрыты определенные детали. В примере ниже показано уведомление, которое скрывает содержимое обычного уведомления на экране блокировки.

```
val publicBuilder = NotificationCompat.Builder(this, CHANNEL_ID)
    .setSmallIcon(R.drawable.my_icon)
    .setContentTitle("My alternative notification")
    .setPriority(NotificationCompat.PRIORITY_DEFAULT)

val builder = NotificationCompat.Builder(this, CHANNEL_ID)
    .setSmallIcon(R.drawable.my_icon)
    .setContentTitle("My lock screen notification")
    .setContentText("My content for this notification")
    .setPriority(NotificationCompat.PRIORITY_DEFAULT)
    .setVisibility(VISIBILITY_PRIVATE)
    .setPublicVersion(publicBuilder.build())
```

Уведомления на иконках приложений (App icon badge)

Зачастую, когда у вас появляются уведомления вы можете увидите соответствующий значок на иконке приложения. Такие уведомления доступны на Android 8.0 и выше. К ним можно получить доступ, путем долгого нажатия на значок приложения.

Данный тип уведомлений включен с систему по умолчанию, и нет необходимости ничего настраивать для их использования. Однако это не значит, что мы не можем их настраивать. Вот пример, в котором мы настраиваем уведомление так, чтобы он отображал маленький значок вместо большого.

```
val builder = NotificationCompat.Builder(this@MainActivity, CHANNEL_ID)
    .setContentTitle("Badged Notification")
    .setContentText("New badged notification")
    .setSmallIcon(R.drawable.notification_icon)
    .setBadgeIconType(NotificationCompat.BADGE_ICON_SMALL)
```

Также существует возможность отключения уведомлений на иконках приложений с помощью метода `showBadges` канале уведомлений.

```
val channel = NotificationChannel(CHANNEL_ID, name, importance).apply {
    description = descriptionText
    setShowBadge(false)
}
```

3. Добавления действий к уведомлению

В данном разделе покажем, как добавить вашему уведомлению действия такие, как открытие активности или отображение индикатора выполнения.

OnClick

Во-первых, давайте посмотрим, каким образом мы можем совершить какое-либо действие по нажатию пользователя на содержание нашего уведомления. Для нам необходимо создать действие объект типа `pendingIntent`, чтобы открыть MainActivity как показано ниже:

```
val intent = Intent(this, MainActivity::class.java)
val pendingIntent: PendingIntent = PendingIntent.getActivity(this, 0, intent, 0)
```

После этого нам нужно передать `pendingIntent` в метод `setContentIntent` при создании уведомления.

```
val builder = NotificationCompat.Builder(this@MainActivity, CHANNEL_ID)
.setContentIntent(pendingIntent)
```

Можно настроить автоматическое удаление уведомления, когда пользователь нажимает на них с помощью метода `setAutoCancel()`.

```
val builder = NotificationCompat.Builder(this@MainActivity, CHANNEL_ID)
.setAutoCancel(true)
```

Кнопки действий при уведомлении

В Android можно настроить до трех кнопок действий для уведомления. Это позволяет пользователю быстро реагировать на события. Для добавления кнопок к уведомлению, необходимо передать объект `PendingIntent` методу `addAction()` при создании. Функция addAction () принимает три аргумента: иконка для действия, имя кнопки и объект типа `pendingIntent` для действия.

```
val intent = Intent(this, MainActivity::class.java).apply {
    flags = Intent.FLAG_ACTIVITY_NEW_TASK or Intent.FLAG_ACTIVITY_CLEAR_TASK
}

val pendingIntent: PendingIntent = PendingIntent.getActivity(this, 0, intent, 0)

val builder = NotificationCompat.Builder(this@MainActivity, CHANNEL_ID)
    .addAction(R.drawable.my_icon, "Open Activity", pendingIntent)
    .setAutoCancel(true)
```

Расширяемое уведомление

Можно расширить уведомление для отображения большего количества информации, например больших изображений или блоков текста. Для этого необходимо вызвать метод `setStyle()` в `Notification.Builder`. В примере ниже для Style устанавливается значение `NotificationCompat.BigPictureStyle` (большое изображение).

```
val bitmap = BitmapFactory.decodeResource(resources, R.drawable.my_icon)

val builder = NotificationCompat.Builder(this, CHANNEL_ID)
    .setSmallIcon(R.drawable.my_icon)
    .setContentTitle("My expandable notification")
    .setContentText("My notification that can be expended")
    .setLargeIcon(bitmap)
    .setStyle(NotificationCompat.BigPictureStyle()
        .bigPicture(bitmap)
        .bigLargeIcon(null))
```

Также можно отображать большие блоки текста, установив стиль `NotificationCompat.BigTextStyle`.

Индикатор выполнения(Progress bar)

Еще одно распространенное действие - отображать индикаторы выполнения в ваших уведомлениях и обновлять их. Для этого можно воспользоваться методом setProgress(). Далее приведем пример уведомления с индикатором выполнения.

```
val builder = NotificationCompat.Builder(this, CHANNEL_ID)
    .setSmallIcon(R.drawable.my_icon)
    .setContentTitle("My progress bar notification")
    .setContentText("My notification with a progress bar ")

// константы для индикатора прогресса
val PROGRESS_MAX = 100
val PROGRESS_CURRENT = 0

NotificationManagerCompat.from(this).apply {
    // Устанавливает начальный прогресс на 0
    builder.setProgress(PROGRESS_MAX, PROGRESS_CURRENT, false)
    notify(7 , builder.build())

    for(i in 0 until 100){
        builder.setProgress(PROGRESS_MAX, i, false)
        HandlerThread.sleep(100)
    }

    // Обновляет уведомление, когда прогресс завершен
    builder.setContentText("Download complete")
        .setProgress(0, 0, false)
    notify(7, builder.build())
}
```

В примере использован простой цикл for, который при каждой итерации останавливает поток на 100 миллисекунд, имитируя загрузку.

Вывод уведомлений на экран

Для вывода уведомлений на экран нужно вызвать функцию `notify` и передать идентификатор и сборщик уведомлений:

```
with(NotificationManagerCompat.from(this)) {
    notify(id, builder.build())
}
```

Также можно группировать уведомления, которые могут отправляться приложениями за короткий промежуток времени (например социальными сетями). В примере ниже создается идентификатор группы, в которую добавляется уведомление.

```
val GROUP_KEY = "ru.samsung.itacademy.notificationexample.KEY"
val group = NotificationCompat.Builder(this@MainActivity, CHANNEL_ID)
    .setSmallIcon(R.drawable.my_icon)
    .setContentTitle("My group notification")
    .setContentText("My notification group")
    .setGroup(GROUP_KEY)
```

Каналы уведомлений

Еще одна очень важная часть уведомлений - это канал уведомлений, который требуется для всех уведомлений для версий Android Oreo (8.0) и выше. Канал определяет визуальное и слуховое поведение уведомлений, находящихся в нем. После создания каналов пользователь получает может включать и отключать различные каналы уведомлений вашего приложения, а также управлять настройками уведомлений. Для каждого типа уведомлений необходимо создать собственный канал например как показано ниже.

```
private val CHANNEL_ID = "CHANNEL_ID"
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
    // имя канала
    val name = "Channel"
    val descriptionText = "My channel example"
    val importance = NotificationManager.IMPORTANCE_DEFAULT
    val channel = NotificationChannel(CHANNEL_ID, name, importance).apply {
        description = descriptionText
    }
    // регистрация канала в системе
    val notificationManager: NotificationManager =
        getSystemService(Context.NOTIFICATION_SERVICE) as NotificationManager
    notificationManager.createNotificationChannel(channel)
}
```

В этом примере мы создаем простой канал уведомлений и регистрируем его в системе с помощью функции `createNotificationChannel()`.

В случае если канал уведомлений больше не используется, необходимо его удалить с помощью функции `deleteNotificationChannel()` класса `NotificationManager`.

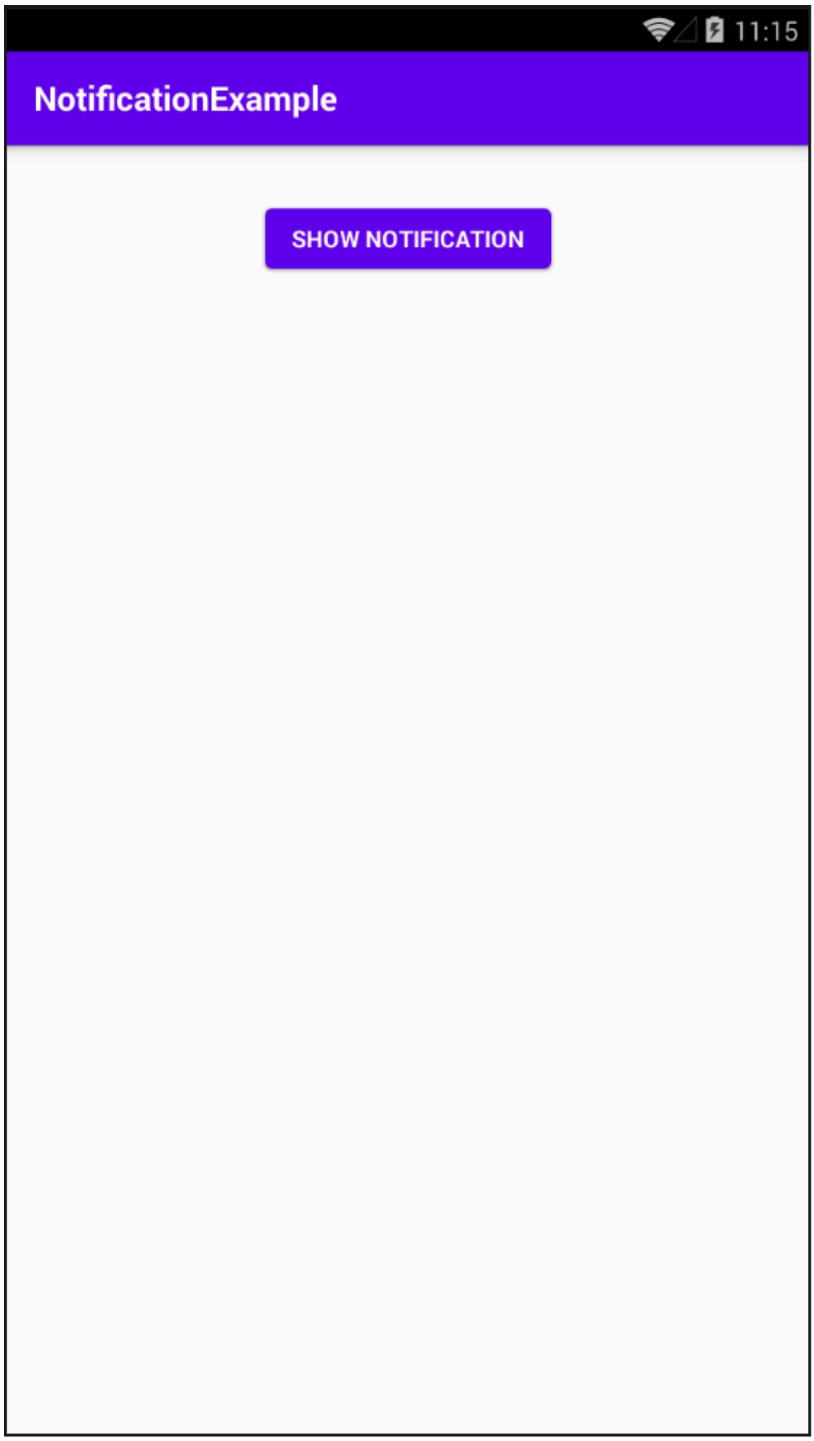
```
private val CHANNEL_ID = "CHANNEL_ID"
val notificationManager = getSystemService(Context.NOTIFICATION_SERVICE) as NotificationManager

notificationManager.deleteNotificationChannel(CHANNEL_ID)
```

4. Пример 4.5.1

Рассмотрим пример приложения, в котором создадим уведомление, создадим для него канал и установим действие по нажатию.

В разметке главной активности разместим кнопку "Show notification" для показа нашего уведомления:

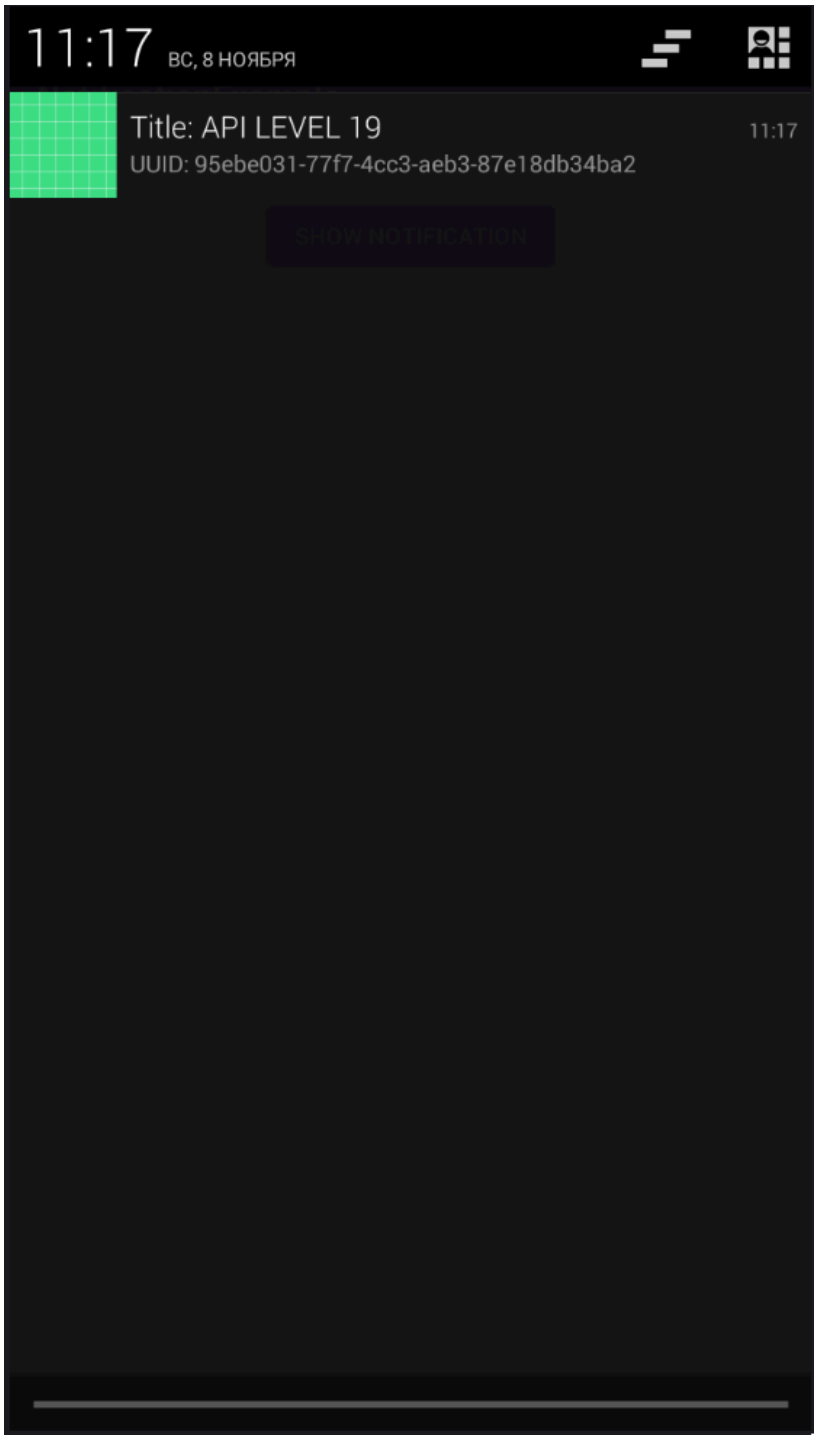
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/btn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="32dp"
        android:text="Show Notification"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        />

</androidx.constraintlayout.widget.ConstraintLayout>
```



Код активности будет следующим:

```
package ru.samsung.itacademy.notificationexample

import android.app.NotificationChannel
import android.app.NotificationManager
import android.app.PendingIntent
import android.content.Context
import android.content.Intent
import android.os.Build
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Button
import androidx.core.app.NotificationCompat
import androidx.core.app.NotificationManagerCompat
import java.util.*

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val btn: Button = findViewById(R.id.btn)

        // Создание и регистрация канала уведомлений api 26+
        val channelId = "My_Channel_ID"
        createNotificationChannel(channelId)


        // Создайте явное намерение
        val intent = Intent(this,MainActivity::class.java)
        intent.apply {
            flags = Intent.FLAG_ACTIVITY_NEW_TASK or Intent.FLAG_ACTIVITY_CLEAR_TASK
        }
        val pendingIntent = PendingIntent.getActivity(this,0,intent,0)

        btn.setOnClickListener{
            val notificationBuilder = NotificationCompat.Builder(this,channelId)
                .setSmallIcon(R.drawable.ic_launcher_background)
                .setContentTitle("Title: API LEVEL " + Build.VERSION.SDK_INT)
                .setContentText("UUID: " + UUID.randomUUID())
                .setPriority(NotificationCompat.PRIORITY_DEFAULT)
                .setContentIntent(pendingIntent)

            with(NotificationManagerCompat.from(this)){
                notify(1, notificationBuilder.build())
            }
        }
    }


    private fun createNotificationChannel(channelId:String) {
        // Создаем NotificationChannel, но только в API 26+ (Android 8.0),
        // потому что класс NotificationChannel является новым и отсутствует в библиотеке поддержки.
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O){
            val name = "My Channel"
            val channelDescription = "Channel Description"
            val importance = NotificationManager.IMPORTANCE_DEFAULT

            val channel = NotificationChannel(channelId,name,importance)
            channel.apply {
                description = channelDescription
            }

            // Наконец регистрируем канал в системе
            val notificationManager = getSystemService(Context.NOTIFICATION_SERVICE) as NotificationManager
            notificationManager.createNotificationChannel(channel)
        }
    }

}
```

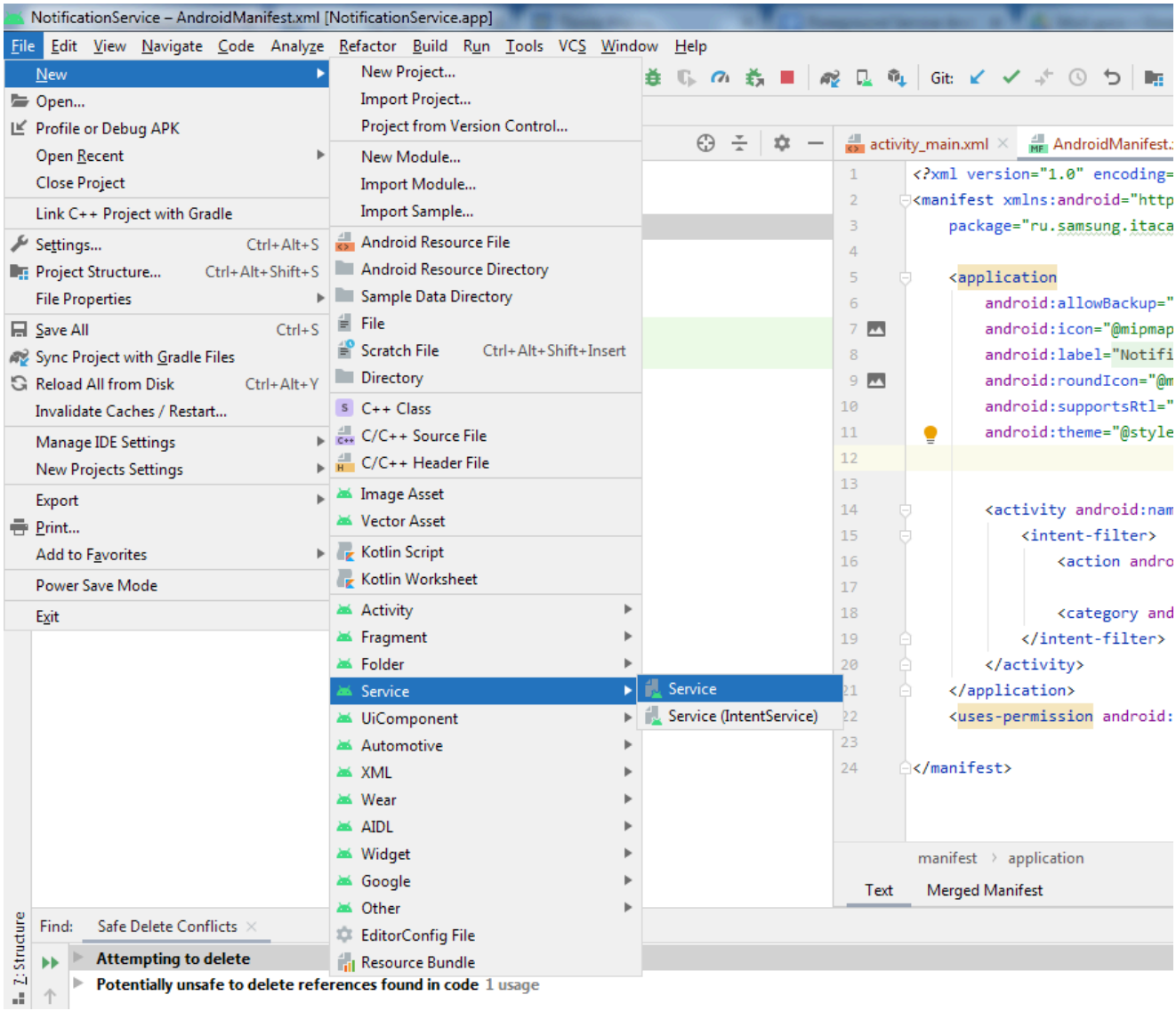
После запуска приложения и нажатия на кнопку в строке состояния появится уведомление:



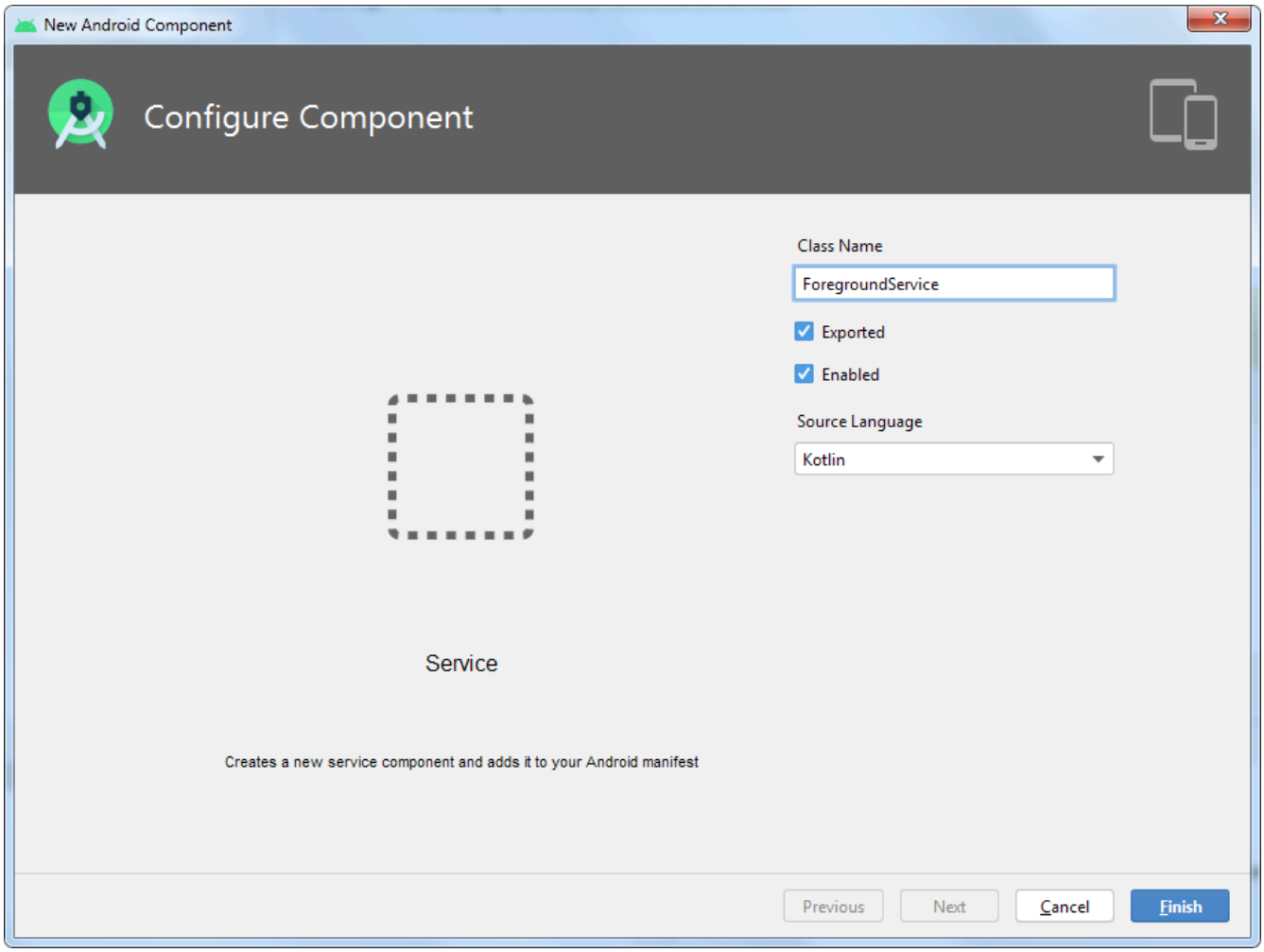
Теперь вы можете менять параметры уведомлений, чтобы на экране появлялись уведомления других типов.

5. Пример 4.5.2

Приведем пример реализации сервиса по отправке уведомлений. Создадим новый проект, а в нем файл сервиса с названием *ForegroundService.kt*. Это можно сделать вручную, прописав сервис в манифесте, а можно автоматически, нажав *File -> New -> Service*.



Далее необходимо ввести имя файла для сервиса:



В итоге появится файл *ForegroundService.kt* со следующим содержимым:

```
package ru.samsung.itacademy.notificationservice

import android.app.Service
import android.content.Intent
import android.os.IBinder

class ForegroundService : Service() {

    override fun onBind(intent: Intent): IBinder {
        TODO("Return the communication channel to the service.")
    }
}
```

Добавим в него следующий код:

```
package ru.samsung.itacademy.notificationservice

import android.app.NotificationChannel
import android.app.NotificationManager
import android.app.PendingIntent
import android.app.Service
import android.content.Context
import android.content.Intent
import android.os.Build
import android.os.IBinder
import androidx.core.app.NotificationCompat
import androidx.core.content.ContextCompat

class ForegroundService : Service() {
    private val CHANNEL_ID = "CHANNEL_ID"

    companion object {

        fun startService(context: Context, message: String) {
            val startIntent = Intent(context, ForegroundService::class.java)
            startIntent.putExtra("inputExtra", message)
            ContextCompat.startForegroundService(context, startIntent)
        }

        fun stopService(context: Context) {
            val stopIntent = Intent(context, ForegroundService::class.java)
            context.stopService(stopIntent)
        }
    }

    override fun onStartCommand(intent: Intent?, flags: Int, startId: Int): Int {

        //do heavy work on a background thread
        val input = intent?.getStringExtra("inputExtra")
        createNotificationChannel()
        val notificationIntent = Intent(this, MainActivity::class.java)
        val pendingIntent = PendingIntent.getActivity(
            this,
            0, notificationIntent, 0
        )

        val notification = NotificationCompat.Builder(this, CHANNEL_ID)
            .setContentTitle("Foreground Service Kotlin Example")
            .setContentText(input)
            .setSmallIcon(R.drawable.ic_launcher_background)
            .setContentIntent(pendingIntent)
            .build()

        startForeground(1, notification)
        //stopSelf();
        return START_NOT_STICKY
    }

    override fun onBind(intent: Intent): IBinder? {
        return null
    }

    private fun createNotificationChannel() {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            val serviceChannel = NotificationChannel(CHANNEL_ID, CHANNEL_ID,
                NotificationManager.IMPORTANCE_DEFAULT)

            val manager = getSystemService(NotificationManager::class.java)
            manager!!.createNotificationChannel(serviceChannel)
        }
    }
}
```

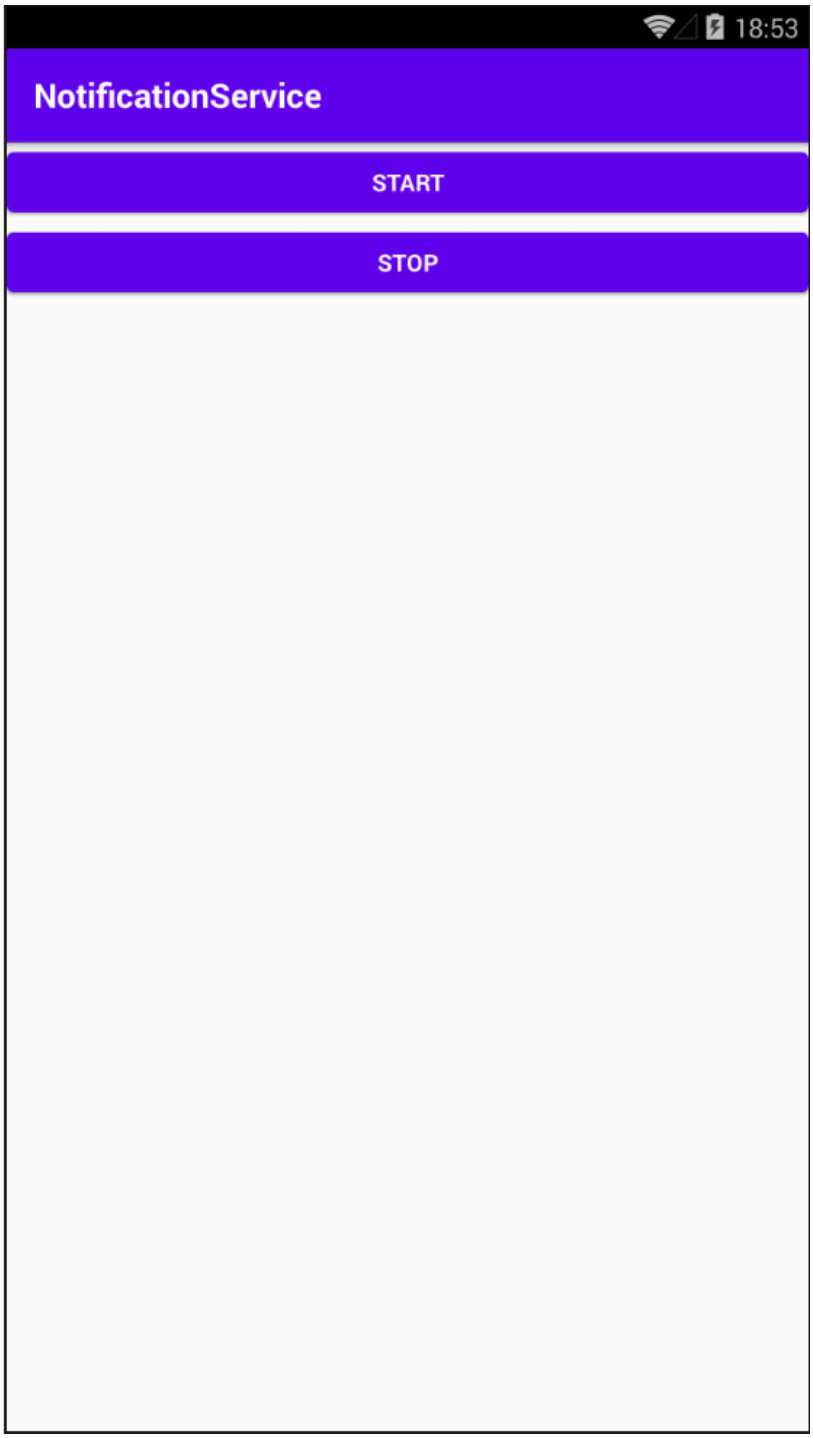
В этом исходном файле создается канал уведомлений. Также запускаем службу. Для отображения уведомления необходимо использовать два параметра id и экземпляр уведомления. Начиная с версии Android 8.0 нельзя выполнять длительные операции в фоновом режиме без уведомления пользователей. Таким образом, мы передаем объект уведомления методу `startForeground()`.

Далее объявим разрешение в AndroidManifest.xml

```
<uses-permission android:name="android.permission.FOREGROUND_SERVICE"/>
```

В макет *activity_main.xml* добавим две кнопки "Start" и "Stop" для запуска и остановки сервиса соответственно.

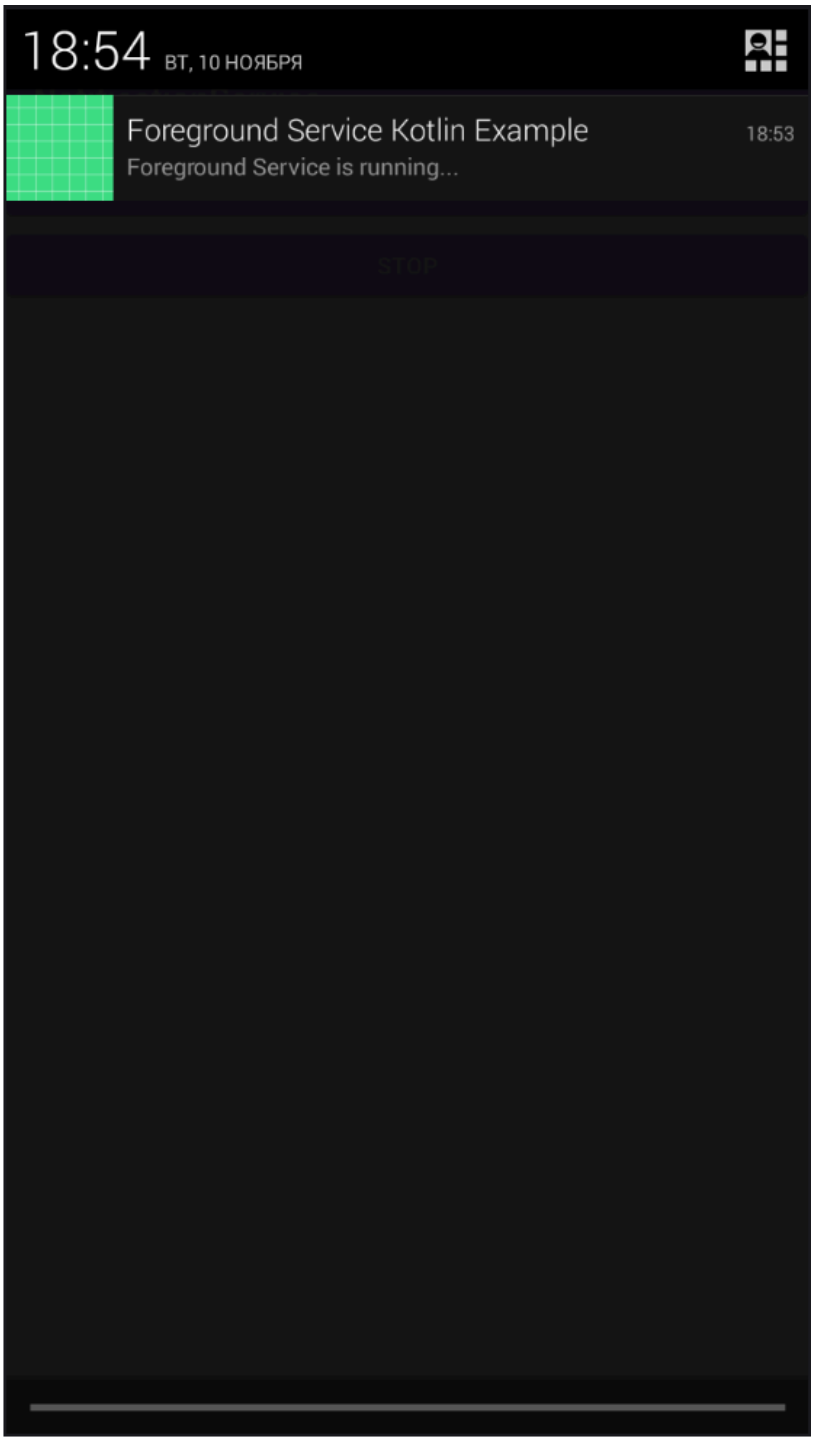
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/buttonStart"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="start" />

    <Button
        android:id="@+id/buttonStop"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="stop" />
</LinearLayout>
```



Наконец, в *MainActivity.kt* напишем следующее:

```
package ru.samsung.itacademy.notificationservice

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.View
import android.widget.Button

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        val buttonStart = findViewById<Button>(R.id.buttonStart)
        val buttonStop = findViewById<Button>(R.id.buttonStop)
        buttonStart.setOnClickListener(View.OnClickListener {
            ForegroundService.startService(this, "Foreground Service is running...")
        })
        buttonStop.setOnClickListener(View.OnClickListener {
            ForegroundService.stopService(this)
        })
    }
}
```

После нажатия кнопки "Start" будет запущена служба, и на панели сверху появится уведомление. Оно исчезнет, как только вы нажмете кнопку «Stop».



Начать тур для пользователя на этой странице