# 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 №20 по дисциплине

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

Уведомления

Выполнил:

Студент группы 8B21 П. А. Жердева

Проверил:

Ст. преп. ОИТ ИШИТР В. А. Дорофеев

# Задание

Напишите программу, которая по нажатию кнопки отображает уведомление – конечно, с обязательной проверкой разрешена ли отправка уведомлений, запросом такого разрешения и т.п. формальностями. Уведомление должно выглядеть примерно следующим образом:

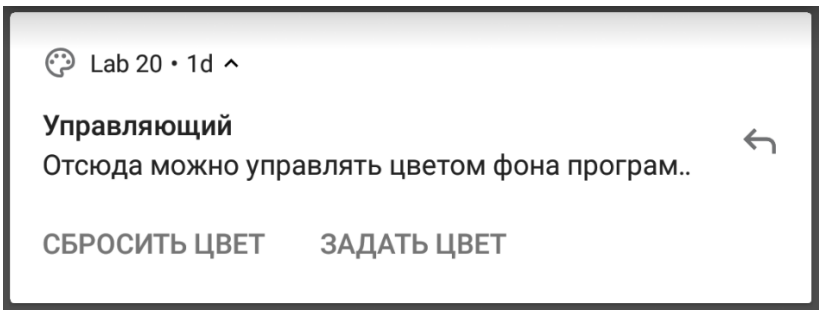


Рис. 1. Примерный вид уведомления

При нажатии уведомления должна запускаться программа (даже если она уже запущена, что поделать…). При нажатии кнопки «Сбросить цвет» – фон активности становится белым. При нажатии кнопки «Задать цвет» – появляется поле ввода, в котором можно ввести цвет в формате RRGGBB:

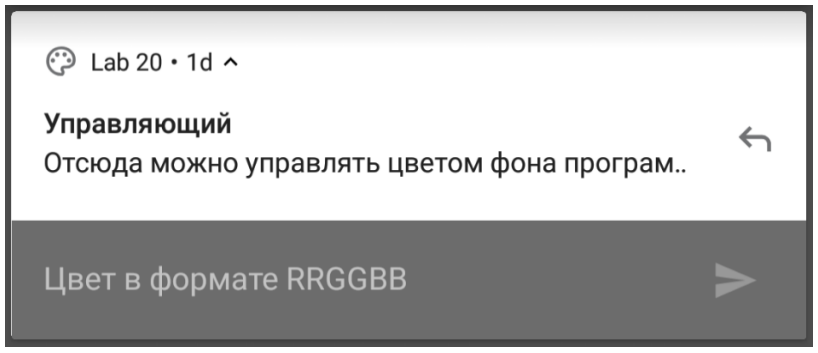


Рис. 2. Поле ввода, в котором можно ввести цвет в

формате RRGGBB

После отправки цвета – фон активности становится выбранного цвета, а в центре появляется надпись вида «Выбран цвет: #RRGGBB», где RRGGBB – это hex-код выбранного цвета:



Рис. 3 Активность с отображением выбранного цвета

**Ход выполнения**

Для работы с уведомлениями было добавлено разрешение: <uses-permission android:name="android.permission.POST\_NOTIFICATIONS" /> в AndroidManifest.xml

**Как устроено приложение:**

1. Запрашивает у пользователя разрешение на отправку уведомлений (POST\_NOTIFICATIONS).
2. Нажатие на кнопку «Показать уведомление» отправляет интерактивное всплывающее уведомление, содержащее:

* кнопку «Сбросить цвет» (фон меняется на белый)
* кнопку «Задать цвет», при нажатии на которую появляется поле ввода
* поле ввода HEX-кода цвета.

1. При нажатии на кнопку — активность меняет цвет фона на введенное значение.

Таблица 1. Общая структура проекта

|  |  |
| --- | --- |
| Файл | Назначение |
| **MainActivity.kt** | Главный экран, обрабатывает разрешения и события от уведомлений |
| **NotificationHelper.kt** | Создаёт и показывает уведомления |
| **NotificationReceiver.kt** | Принимает действия пользователя из уведомлений |
| **EventBus.kt** | Передаёт события между компонентами |
| **activity\_main.xml** | Разметка главного экрана |

**MainActivity.kt -** главная активность, которая управляет всем процессом.

Что делает:

1. Проверяет разрешения для уведомлений RequestPermission(). Если разрешение не выдано — запрашивает у пользователя.
2. Показывает уведомление при нажатии на кнопку showNotification().
3. Реагирует на действия пользователя из уведомления:
   * сброс цвета;
   * установка нового цвета setColor() – метод преобразует HEX-строку в цвет и меняет фон. Если код неверный — показывает сообщение об ошибке.

**NotificationHelper.kt -** класс, который создаёт и отображает уведомления.

Что делает:

1. Создаёт канал уведомлений createNotificationChannel().
2. Формирует само уведомление **—** текст, иконку, кнопки NotificationCompat.Builder().
3. Добавляет интерактивное поле ввода(RemoteInput) для HEX-кода цвета.
4. Отправляет уведомлениепользователю через NotificationManagerCompat.

**NotificationReceiver.kt -** приёмник, который реагирует на действия пользователя из уведомления.

Что делает:

1. При нажатии на «Сбросить цвет» — отправляет событие ACTION\_RESET\_COLOR.
2. При вводе цвета — отправляет ACTION\_SET\_COLOR и сохраняет последний цвет.

**EventBus.kt** – синглтон, который является простым механизмом обмена событиями между компонентами приложения.

Что делает:

1. Создаёт поток событий MutableSharedFlow для передачи сигналов типа: "RESET\_COLOR", "SET\_COLOR:#FF0000".
2. NotificationReceiver отправляет в него команды.
3. MainActivity слушает этот поток и реагирует.

**Текст программы**

**EventBus.kt:**

package ru.polinazherdeva.lr20  
import kotlinx.coroutines.flow.MutableSharedFlow  
import kotlinx.coroutines.flow.asSharedFlow  
  
*// Создаём объект (Singleton), который будет служить шиной событий (EventBus)*object EventBus {  
 private val \_events = *MutableSharedFlow*<String>() *// Создаём изменяемый поток данных, в который можно отправлять события* val events = \_events.*asSharedFlow*() *// Делаем неизменяемую публичную версию потока, чтобы другие классы могли только слушать события  
  
 // suspend-функция для отправки события (выполняется в корутине)* suspend fun sendEvent(action: String) {  
 \_events.emit(action) *// Отправляем событие всем подписчикам* }  
}

**MainActivity.kt:**

package ru.polinazherdeva.lr20  
import android.Manifest  
import android.graphics.Color  
import android.os.Build  
import android.os.Bundle  
import android.widget.Button  
import android.widget.LinearLayout  
import android.widget.TextView  
import android.widget.Toast  
import androidx.activity.result.contract.ActivityResultContracts  
import androidx.appcompat.app.AppCompatActivity  
import androidx.core.app.ActivityCompat  
import androidx.lifecycle.*lifecycleScope*import kotlinx.coroutines.flow.collectLatest  
import kotlinx.coroutines.launch  
  
class MainActivity : AppCompatActivity() {  
  
 private lateinit var rootLayout: LinearLayout  
 private lateinit var colorText: TextView  
 private lateinit var notificationHelper: NotificationHelper  
  
 *// Лаунчер для запроса разрешения на уведомления* private val requestPermission =  
 registerForActivityResult(ActivityResultContracts.RequestPermission()) **{** granted **->** if (granted) showNotification() *// Разрешение получено → показываем уведомление* else Toast.makeText(this, getString(R.string.*toast\_no\_permission*), Toast.*LENGTH\_SHORT*).show()  
 **}** override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_main*) *// Подключаем макет  
  
 // Находим элементы из разметки* rootLayout = findViewById(R.id.*rootLayout*)  
 colorText = findViewById(R.id.*colorText*)  
 notificationHelper = NotificationHelper(this)  
  
 *// Настраиваем кнопку для вызова уведомления* findViewById<Button>(R.id.*btnShowNotification*).*apply* **{** *text* = getString(R.string.*text\_show\_notification*)  
 setOnClickListener **{** handleNotificationRequest() **}** *// Обработчик нажатия* **}** *// Подписываемся на события из EventBus  
 lifecycleScope*.*launch* **{** EventBus.events.collectLatest **{** action **->** when (action) {  
 *// Если пришло событие "сбросить цвет" — ставим белый фон* NotificationReceiver.ACTION\_RESET\_COLOR -> setColor("#FFFFFF")  
 *// Если пришло "установить цвет" — применяем последний введённый цвет* NotificationReceiver.ACTION\_SET\_COLOR -> {  
 val color = NotificationReceiver.lastColor ?: "#FFFFFF"  
 setColor(color)  
 }  
 }  
 **}  
 }** }  
  
 *// Проверка разрешения и показ уведомления* private fun handleNotificationRequest() {  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*TIRAMISU*) {  
 if (ActivityCompat.checkSelfPermission(  
 this, Manifest.permission.*POST\_NOTIFICATIONS* ) != android.content.pm.PackageManager.*PERMISSION\_GRANTED* ) {  
 *// Если разрешения нет — запрашиваем его* requestPermission.launch(Manifest.permission.*POST\_NOTIFICATIONS*)  
 } else showNotification() *// Разрешение есть — показываем уведомление* } else showNotification() *// Для старых версий Android* }  
  
  
 *// Показ уведомления через вспомогательный класс* private fun showNotification() = notificationHelper.showNotification()  
  
 *// Изменение цвета фона и текста на экране* private fun setColor(hex: String) {  
 try {  
 val parsed = Color.parseColor(hex) *// Парсим строку в цвет* rootLayout.setBackgroundColor(parsed) *// Меняем цвет фона* colorText.*text* = getString(R.string.*text\_selected\_color*, hex) *// Обновляем текст* } catch (e: Exception) {  
 *// Если пользователь ввёл неверный цветовой код* Toast.makeText(this, getString(R.string.*toast\_invalid\_color*), Toast.*LENGTH\_SHORT*).show()  
 }  
 }  
}

**NotificationHelper.kt:**

package ru.polinazherdeva.lr20  
import android.app.\*  
import android.content.Context  
import android.content.Intent  
import android.graphics.Color  
import android.os.Build  
import androidx.core.app.NotificationCompat  
import androidx.core.app.NotificationManagerCompat  
import androidx.core.app.RemoteInput  
  
*// Вспомогательный класс для создания и показа уведомлений*class NotificationHelper(private val context: Context) {  
  
 companion object {  
 *// ID канала уведомлений (уникальный идентификатор канала)* const val CHANNEL\_ID = "lab20\_channel\_color"  
 *// ID самого уведомления (по нему можно обновлять или удалять уведомление)* const val NOTIFICATION\_ID = 1  
 }  
  
 *// Функция для создания и показа уведомления* fun showNotification() {  
 createNotificationChannel() *// Создаём канал  
  
 // Intent для открытия MainActivity при нажатии на уведомлени* val openAppIntent = Intent(context, MainActivity::class.*java*).*apply* **{** *// Флаги очищают стек активностей и запускают новое окно приложения  
 flags* = Intent.*FLAG\_ACTIVITY\_NEW\_TASK* or Intent.*FLAG\_ACTIVITY\_CLEAR\_TASK* **}** *// Готовый Intent, который будет выполнен системой при нажатии на уведомление* val openAppPendingIntent = PendingIntent.getActivity(  
 context,  
 0,  
 openAppIntent,  
 PendingIntent.*FLAG\_IMMUTABLE // Нельзя менять содержимое Intent позже* )  
  
 *// Intent для кнопки "Сбросить цвет"* val resetIntent = Intent(context, NotificationReceiver::class.*java*).*apply* **{** *action* = NotificationReceiver.ACTION\_RESET\_COLOR *// Задаём действие* **}** *// PendingIntent, который будет отправлен BroadcastReceiver при нажатии кнопки* val resetPendingIntent = PendingIntent.getBroadcast(  
 context, 1, resetIntent, PendingIntent.*FLAG\_IMMUTABLE* )  
  
 *// Создаём поле для ввода текста (HEX-кода цвета)* val remoteInput = RemoteInput.Builder(NotificationReceiver.KEY\_TEXT)  
 .setLabel(context.getString(R.string.*label\_input\_color*)) *// Подпись поля ввода* .build()  
  
 *// Intent для кнопки "Задать цвет"* val setColorIntent = Intent(context, NotificationReceiver::class.*java*).*apply* **{** *action* = NotificationReceiver.ACTION\_SET\_COLOR  
 **}** *// PendingIntent для действия "Задать цвет" (должен быть MUTABLE, чтобы можно было передать введённый текст)* val setColorPendingIntent = PendingIntent.getBroadcast(  
 context, 2, setColorIntent, PendingIntent.*FLAG\_MUTABLE* )  
  
 *// Создаём действие для уведомления с полем ввода* val setColorAction = NotificationCompat.Action.Builder(  
 android.R.drawable.*ic\_menu\_edit*, *// Иконка кнопки* context.getString(R.string.*action\_set\_color*), *// Текст кнопки* setColorPendingIntent *// Действие по нажатию* ).addRemoteInput(remoteInput) *// Добавление текстового поля ввола* .build()  
  
 *// Конструктор уведомления* val builder = NotificationCompat.Builder(context, CHANNEL\_ID)  
 .setSmallIcon(R.drawable.*ic\_art*) *// значок* .setContentTitle(context.getString(R.string.*title\_notification*)) *// Заголовок* .setContentText(context.getString(R.string.*text\_notification*)) *// Основной текст* .setPriority(NotificationCompat.*PRIORITY\_HIGH*) *// Всплывающее* .setCategory(Notification.*CATEGORY\_MESSAGE*) *// Категория — сообщение* .setAutoCancel(true) *// Уведомление исчезнет после нажатия* .setContentIntent(openAppPendingIntent) *// Действие при нажатии на уведомление  
 // Добавляем кнопку "Сбросить цвет"* .addAction(  
 android.R.drawable.*ic\_menu\_close\_clear\_cancel*,  
 context.getString(R.string.*action\_reset\_color*),  
 resetPendingIntent  
 )  
 *// Добавляем кнопку "Задать цвет" с возможностью ввода* .addAction(setColorAction)  
 .setVisibility(NotificationCompat.*VISIBILITY\_PUBLIC*) *// Делаем уведомление видимым даже на экране блокировки* try {  
 *// Показываем уведомление* NotificationManagerCompat.from(context).notify(NOTIFICATION\_ID, builder.build())  
 } catch (e: SecurityException) {  
 e.printStackTrace() *// Если нет разрешения POST\_NOTIFICATIONS, получаем исключение* }  
 }  
  
 *// Создание канала уведомлений с высоким приоритетом* private fun createNotificationChannel() {  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*O*) {  
 *// Имя и описание канала берём из ресурсов* val name = context.getString(R.string.*title\_notification*)  
 val descriptionText = context.getString(R.string.*text\_notification*)  
 val importance = NotificationManager.*IMPORTANCE\_HIGH // Высокая важность  
 // Создаём объект канала* val channel = NotificationChannel(CHANNEL\_ID, name, importance).*apply* **{** *description* = descriptionText  
 enableVibration(true) *// Включаем вибрацию* enableLights(true) *// Включаем мигание индикатора  
 lightColor* = Color.*MAGENTA // Цвет индикатора  
 lockscreenVisibility* = Notification.*VISIBILITY\_PUBLIC // Видимость на экране блокировки* **}** *// Регистрируем канал в системе* val manager =  
 context.getSystemService(Context.*NOTIFICATION\_SERVICE*) as NotificationManager  
 manager.createNotificationChannel(channel)  
 }  
 }  
}

**NotificationReceiver.kt:**

package ru.polinazherdeva.lr20  
import android.content.BroadcastReceiver  
import android.content.Context  
import android.content.Intent  
import androidx.core.app.RemoteInput  
import kotlinx.coroutines.CoroutineScope  
import kotlinx.coroutines.Dispatchers  
import kotlinx.coroutines.launch  
  
*// Приёмник, который обрабатывает действия из уведомления*class NotificationReceiver : BroadcastReceiver() {  
  
 override fun onReceive(context: Context, intent: Intent) {  
 *// Запускаем корутину для безопасной обработки событий  
 CoroutineScope*(Dispatchers.Default).*launch* **{** when (intent.*action*) {  
 *// Если нажата кнопка "Сбросить" → отправляем событие EventBus* ACTION\_RESET\_COLOR -> EventBus.sendEvent(ACTION\_RESET\_COLOR)  
 *// Если введён цвет → считываем его и отправляем* ACTION\_SET\_COLOR -> {  
 val results = RemoteInput.getResultsFromIntent(intent)  
 val colorText = results?.getCharSequence(KEY\_TEXT)?.toString()  
 if (colorText != null) {  
 lastColor = "#$colorText" *// Сохраняем введённый цвет* EventBus.sendEvent(ACTION\_SET\_COLOR) *// Отправляем событие в EventBus* }  
 }  
 }  
 **}** }  
 companion object {  
 *// Константы для идентификации действий* const val ACTION\_RESET\_COLOR = "ru.polinazherdeva.lr20.RESET\_COLOR"  
 const val ACTION\_SET\_COLOR = "ru.polinazherdeva.lr20.SET\_COLOR"  
 const val KEY\_TEXT = "key\_text\_reply" *// Ключ для поля ввода* var lastColor: String? = null *// Хранит последний введённый цвет* }  
}

**activity\_main.xml:**

*<?*xml version="1.0" encoding="utf-8"*?>*<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:id="@+id/rootLayout"  
 android:orientation="vertical"  
 android:gravity="center"  
 android:padding="24dp"  
 android:background="@android:color/white"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
 <TextView  
 android:id="@+id/colorText"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="@string/text\_background\_white"  
 android:textSize="18sp"  
 android:layout\_marginTop="20dp" />  
  
 <Button  
 android:id="@+id/btnShowNotification"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="@string/text\_show\_notification" />  
</LinearLayout>

**Strings.xml:**

<resources>  
 <string name="app\_name">lr20</string>  
 <string name="title\_notification">Управляющий</string>  
 <string name="text\_notification">Отсюда можно управлять цветом фона програм..</string>  
 <string name="action\_reset\_color">СБРОСИТЬ ЦВЕТ</string>  
 <string name="action\_set\_color">ЗАДАТЬ ЦВЕТ</string>  
 <string name="label\_input\_color">Введите цвет (RRGGBB)</string>  
 <string name="text\_show\_notification">Показать уведомление</string>  
 <string name="toast\_invalid\_color">Неверный формат цвета</string>  
 <string name="toast\_no\_permission">Разрешение не выдано</string>  
 <string name="toast\_no\_notification\_permission">Нет разрешения на уведомления</string>  
 <string name="text\_selected\_color">Выбран цвет: %1$s</string>  
 <string name="text\_background\_white">Фон: #FFFFFF</string>  
</resources>

**Результат работы:**

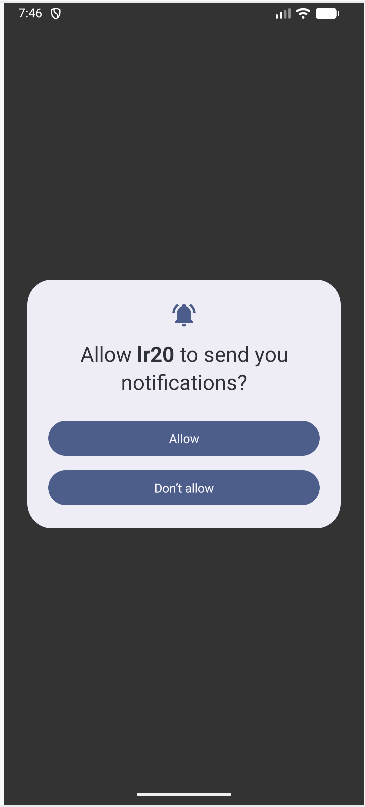


Рис. 4. Запрос разрешения на отправку уведомлений

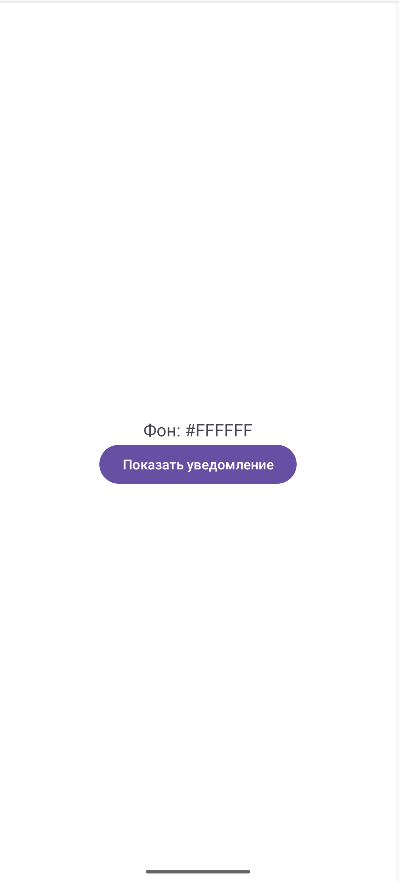


Рис. 5. Стартовый экран с белым фоном

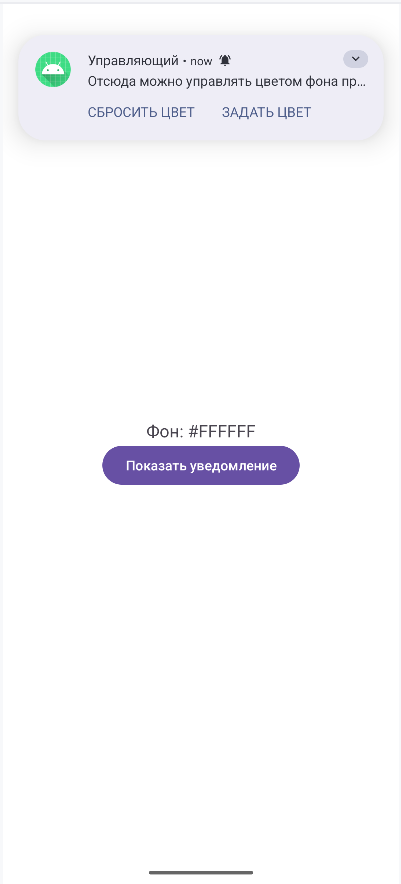


Рис. 6. Всплывающее сообщение после нажатия на кнопку «Показать уведомление»

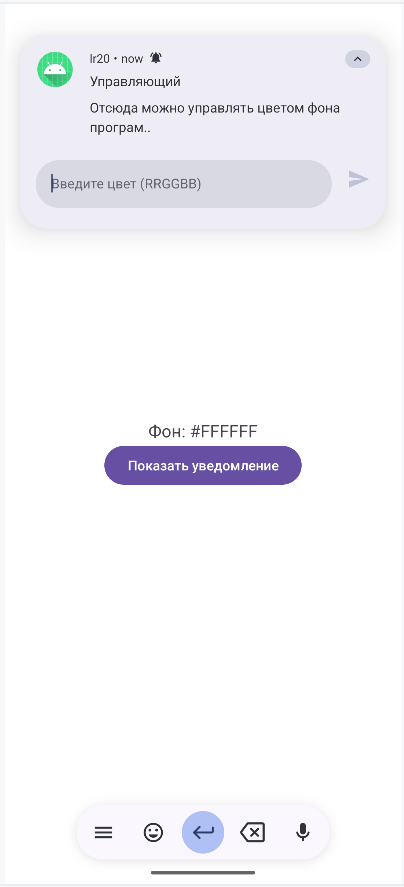


Рис. 7. Поле ввода, в котором можно ввести цвет в

формате RRGGBB

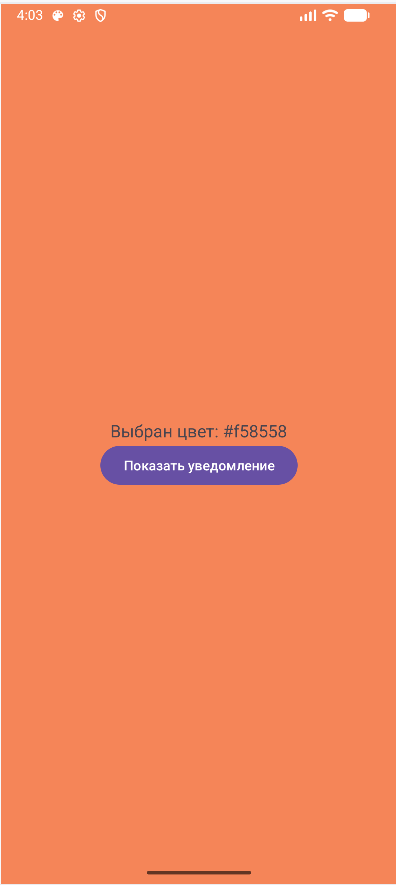


Рис. 8. Окрашивание экрана активности в выбранный цвет и его вывод на экран в формате hex

# Выводы

В ходе лабораторной работы создано Android-приложение, демонстрирующее работу с уведомлениями.

Реализовано:

* запрос разрешения POST\_NOTIFICATIONS;
* интерактивное уведомление с кнопками и вводом текста;
* обработка событий через BroadcastReceiver и EventBus;
* изменение цвета фона в зависимости от действий пользователя.

Приложение корректно показывает уведомления, реагирует на взаимодействие и изменяет интерфейс без перезапуска активности.