

Отчёт по проделанной работе на семинаре по мобильной разработке (24.02.2022).

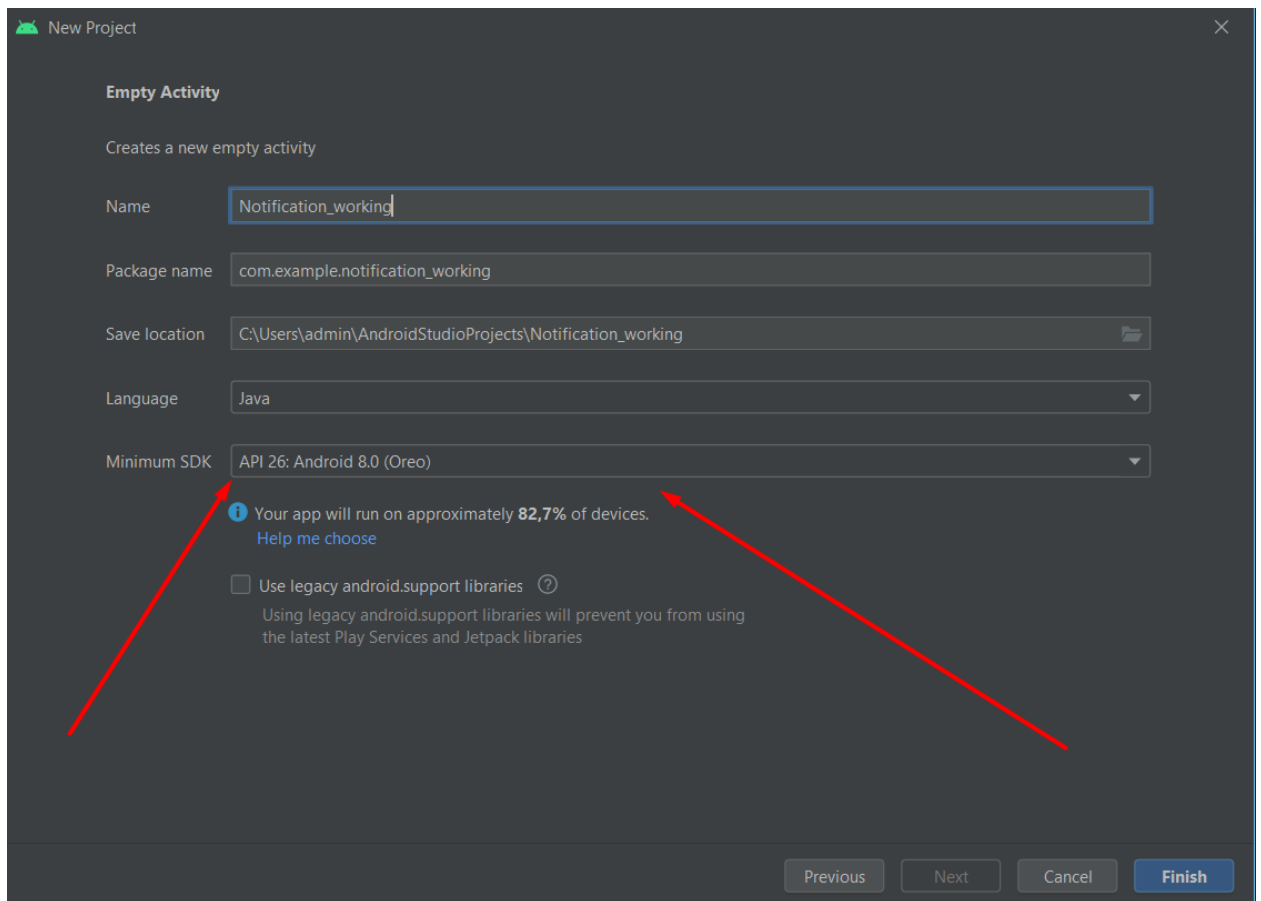
(выполнил Валяев Георгий Анатольевич)

Сегодня уведомления, в свою очередь, пожалуй, дают весьма ценную возможность следить за новостным полем развития событий во множестве приложений как на мобильном телефоне, так и на других устройствах с Операционными системами. При разработке сервисов приложения создание и логика работы уведомлений стоят на одном уровне с самими сервисами, так как они взаимосвязаны по некоторым пунктам.

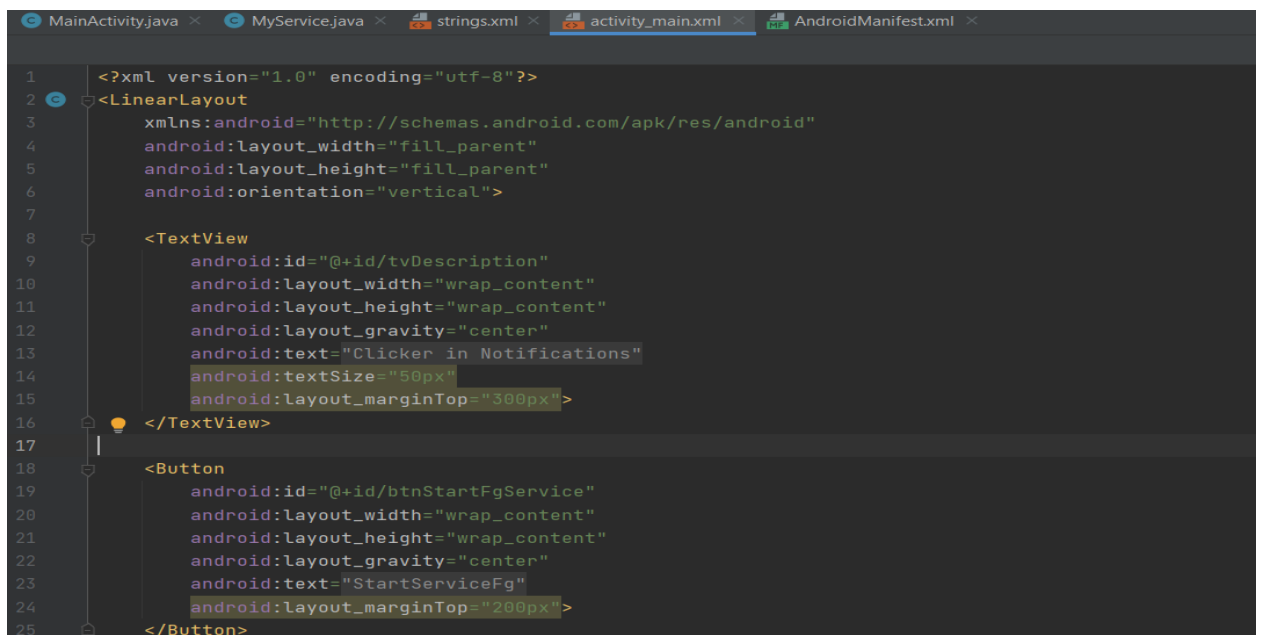
В этой методичке мы попробуем поработать с уведомлениями и сервисами переднего плана, реализуя тем самым Кликер с показом результата в уведомлениях и (при нажатии на него) переводом на новую активность.

Для начала стоит упомянуть важную деталь!

Начиная с 26 версии API, программная работа с уведомлениями была резко изменена, так как из нескольких методов работы с уведомлениями в рамках его существования в приложении разработчики реализовали многофункциональный конструктор уведомления (notificationBuilder), о котором будет упомянуто в дальнейшем. Таким образом, нужно при создании проекта указать минимально поддерживаемый 26 API:



Начнём нашу работу с визуала главной активности – текстовое окно с основной информацией о функционале нашего приложения и 2 кнопки с реализацией сервиса переднего плана и старта Кликера (при последующих нажатиях на неё в уведомлениях будет показано увеличение счётчика нажатий):



```

17
18     <Button
19         android:id="@+id/btnStartFgService"
20         android:layout_width="wrap_content"
21         android:layout_height="wrap_content"
22         android:layout_gravity="center"
23         android:text="StartServiceFg"
24         android:layout_marginTop="200px">
25     </Button>
26
27     <Button
28         android:id="@+id/btnClick"
29         android:layout_width="wrap_content"
30         android:layout_height="wrap_content"
31         android:layout_gravity="center"
32         android:text="To Click"
33         android:layout_marginTop="100px">
34     </Button>
35
36 </LinearLayout>

```

Переходим в MainActivity.java:

```

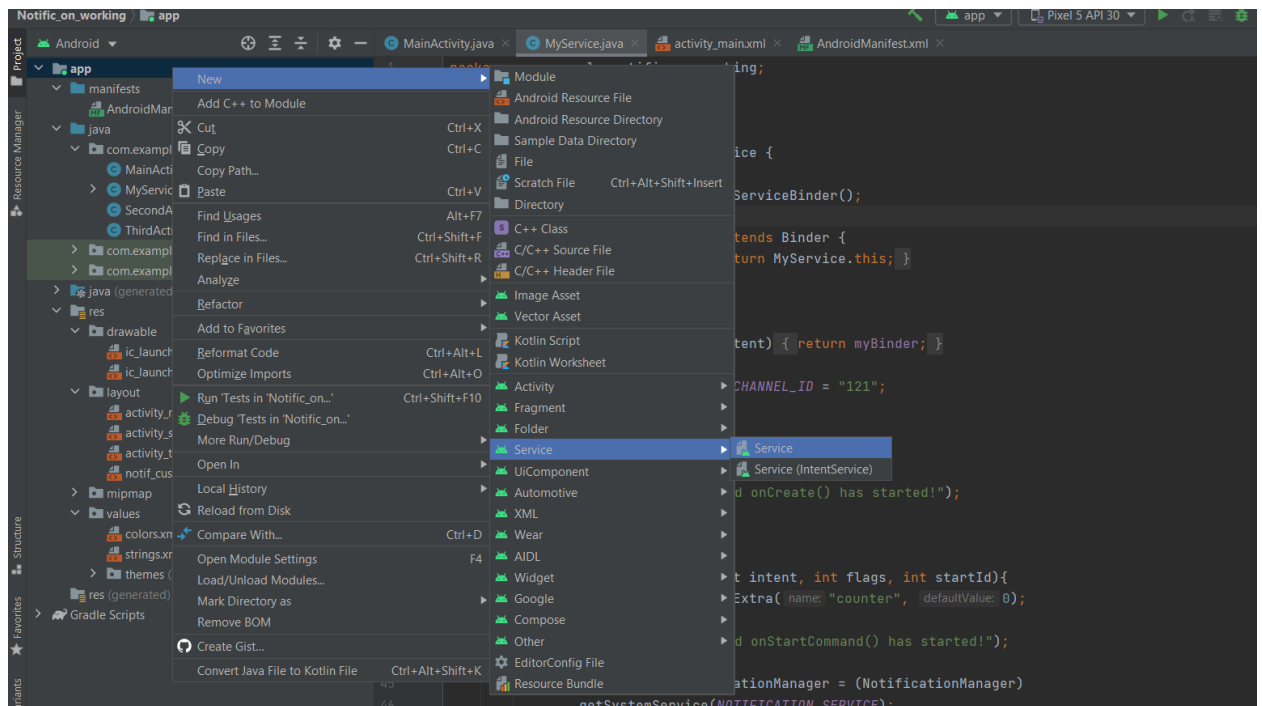
MainActivity.java x MyService.java x activity_main.xml x AndroidManifest.xml x
1  package com.example.notific_on_working;
2
3  import ...
9
10 public class MainActivity extends AppCompatActivity {
11
12     @Override
13     protected void onCreate(Bundle savedInstanceState) {
14         super.onCreate(savedInstanceState);
15         final int[] counter = {0};
16         setContentView(R.layout.activity_main);
17
18         Button btnStartServ = findViewById(R.id.btnStartFgService);
19         Button btnClicker = findViewById(R.id.btnClick);
20
21         Intent StartService = new Intent( packageContext: MainActivity.this, MyService.class);
22
23         btnStartServ.setOnClickListener(new View.OnClickListener() {
24             @Override
25             public void onClick(View view) {
26                 StartService.putExtra( name: "flag", value: 1);
27                 startForegroundService(StartService);
28             }
29         });
30
31         btnClicker.setOnClickListener(new View.OnClickListener() {
32             @Override
33             public void onClick(View view) {
34                 counter[0] += 1;
35                 StartService.putExtra( name: "counter", counter[0]);
36                 startForegroundService(StartService);
37             }
38         });
39     }

```

Там 40 строк кода (закрывающая главный класс скобка не поместилась)

Здесь мы имеем инициализацию переменных для подсчёта нажатий в рамках работы Кликера, интент для отправки сообщений в наш сервис при запуске сервиса переднего плана и обработчики нажатия для 2 кнопок (запуск сервиса переднего плана и отображение уведомлений, Кликер (увеличение количества нажатий) с отображением в уведомлениях).

Далее обратимся к файлу MyService.java, который мы создали следующим образом:



Таким образом, мы инициализируем сам сервис в манифесте приложения автоматически при создании, о чём ещё поговорим далее.

Коснёмся теперь функционала работы сервиса, который, во многом, является основным в работе нашего приложения:

```
MainActivity.java x MyService.java x activity_main.xml x AndroidManifest.xml x
1 package com.example.notific_on_working;
2
3 import ...
15
16 public class MyService extends Service {
17
18     final IBinder myBinder = new MyServiceBinder();
19
20     public class MyServiceBinder extends Binder {
21         MyService getService() { return MyService.this; }
22     }
23
24
25
26     @Override
27     public IBinder onBind(Intent intent) { return myBinder; }
28
29
30
31     public static final String NOT_CHANNEL_ID = "121";
32
33
34     @Override
35     public void onCreate(){
36         super.onCreate();
37         Log.d( tag: "SV", msg: "Method onCreate() has started!");
38     }
39
40     @Override
41     public int onStartCommand(Intent intent, int flags, int startId){
42         int counter = intent.getIntExtra( name: "counter", defaultValue: 0);
43
44         Log.d( tag: "SV", msg: "Method onStartCommand() has started!");
45
46         NotificationManager notificationManager = (NotificationManager)
47             getSystemService(NOTIFICATION_SERVICE);
```

```
MainActivity.java x MyService.java x activity_main.xml x AndroidManifest.xml x
46         getSystemService(NOTIFICATION_SERVICE);
47
48         notificationManager.createNotificationChannel(
49             new NotificationChannel
50                 (NOT_CHANNEL_ID, name: "SV", NotificationManager.IMPORTANCE_DEFAULT)
51         );
52
53
54         Intent secondActivity = new Intent( packageContext: this, SecondActiv.class);
55         Intent thirdActivity = new Intent( packageContext: this, ThirdActiv.class);
56
57         NotificationCompat.Builder notificationBuilder = new NotificationCompat.Builder(
58             context: this, NOT_CHANNEL_ID
59         );
60
61         PendingIntent secondPendingIntent = PendingIntent.getActivity(
62             context: this,
63             requestCode: 0,
64             secondActivity,
65             PendingIntent.FLAG_UPDATE_CURRENT
66         );
67
68         PendingIntent thirdPendingIntent = PendingIntent.getActivity(
69             context: this,
70             requestCode: 0,
71             thirdActivity,
72             PendingIntent.FLAG_UPDATE_CURRENT
73         );
74
75         RemoteViews mRemoteViews = new RemoteViews(getPackageName(), R.layout.notif_custom);
76         mRemoteViews.setTextViewText(R.id.title_notify, text: "TITLE");
77         mRemoteViews.setTextViewText(R.id.content_notify, text: "CONTENT");
78
```

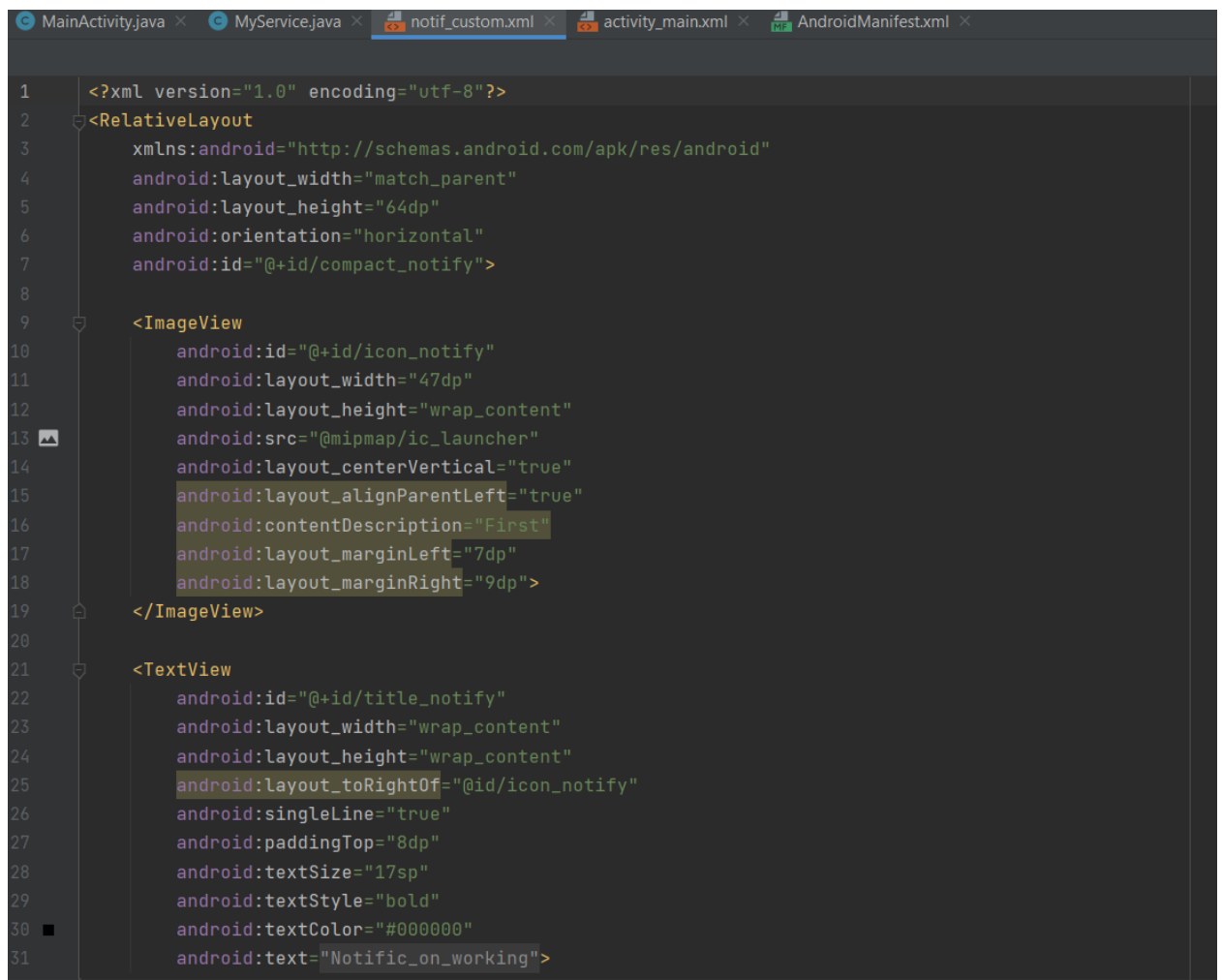
```
MainActivity.java x MyService.java x activity_main.xml x AndroidManifest.xml x
64         secondActivity,
65         PendingIntent.FLAG_UPDATE_CURRENT
66     );
67
68     PendingIntent thirdPendingIntent = PendingIntent.getActivity(
69         context: this,
70         requestCode: 0,
71         thirdActivity,
72         PendingIntent.FLAG_UPDATE_CURRENT
73     );
74
75     RemoteViews mRemoteViews = new RemoteViews(getPackageName(), R.layout.notif_custom);
76     mRemoteViews.setTextViewText(R.id.title_notify, text: "TITLE");
77     mRemoteViews.setTextViewText(R.id.content_notify, text: "CONTENT");
78
79     Notification notification = notificationBuilder.setOngoing(true)
80         .setSmallIcon(R.mipmap.ic_launcher)
81         .setContentTitle("Title")
82         .setContentText("Clicks: " + counter)
83         .setContentIntent(secondPendingIntent)
84         .setPriority(NotificationManager.IMPORTANCE_MIN)
85         .setContent(mRemoteViews)
86         .setCategory(Notification.CATEGORY_SERVICE)
87         .addAction(R.drawable.ic_launcher_background, title: "Second activity", secondPendingIntent)
88         .addAction(R.drawable.ic_launcher_background, title: "Third activity", thirdPendingIntent)
89         .build();
90
91     startForeground(id: 1, notification);
92     return super.onStartCommand(intent, flags, startId);
93 }
94
95 }
```

Разберёмся по порядку:

- 1) Объявляем свой класс связки с клиентом и создаём его новый экземпляр, который используем в переопределённом методе `onBind()`.
- 2) Объявляем уникальный ID для канала отправки уведомлений.
- 3) В методе создания сервиса `onCreate()` отправляем в консоль ответ о выполнении процесса создания сервиса.
- 4) В методе `onStartCommand()` получаем данные от интента главной активности, отправляем в консоль строку о запуске метода, создаём менеджер уведомлений, присваивая ему канал с нашим уникальным id.
- 5) Создаём 2 интента (2 кнопки в уведомлении с переводом в 2 новые активности приложения).
- 6) Создаём первоначальный конструктор уведомления с уникальным id канала.
- 7) Определяем перевод на 2 активности при нажатии кнопок на активностях при помощи соответствующих интентов (`PendingIntent`).

- 8) Задаём визуальную структуру уведомления при помощи RemoteViews, в который передаём специально созданный layout с полем заголовка уведомления и телом уведомления (подробнее о нём ниже)
- 9) Завершаем построение уведомления передачей в методы наших готовых данных (иконка, текст, интент, приоритет, визуал, категория, кнопки с переводом в активности приложения). Командой .build() окончательно завершаем постройку готового уведомления приложения.
- 10) Напоследок, запускаем сервис переднего плана с нашим уведомлением и возвращаем в конце реализацию метода унаследованного класса onStartCommand().

Перейдём к специально созданному layout-у в рамках работы нашего сервиса с уведомлениями:



```
1  <?xml version="1.0" encoding="utf-8"?>
2  <RelativeLayout
3      xmlns:android="http://schemas.android.com/apk/res/android"
4      android:layout_width="match_parent"
5      android:layout_height="64dp"
6      android:orientation="horizontal"
7      android:id="@+id/compact_notify">
8
9      <ImageView
10         android:id="@+id/icon_notify"
11         android:layout_width="47dp"
12         android:layout_height="wrap_content"
13         android:src="@mipmap/ic_launcher"
14         android:layout_centerVertical="true"
15         android:layout_alignParentLeft="true"
16         android:contentDescription="First"
17         android:layout_marginLeft="7dp"
18         android:layout_marginRight="9dp">
19     </ImageView>
20
21     <TextView
22         android:id="@+id/title_notify"
23         android:layout_width="wrap_content"
24         android:layout_height="wrap_content"
25         android:layout_toRightOf="@id/icon_notify"
26         android:singleLine="true"
27         android:paddingTop="8dp"
28         android:textSize="17sp"
29         android:textStyle="bold"
30         android:textColor="#000000"
31         android:text="Notific_on_working">
```

```

29         android:textStyle="bold"
30         android:textColor="#000000"
31         android:text="Notific_on_working">
32     </TextView>
33
34     <TextView
35         android:id="@+id/content_notify"
36         android:layout_width="wrap_content"
37         android:layout_height="wrap_content"
38         android:layout_toRightOf="@id/icon_notify"
39         android:paddingBottom="9dp"
40         android:layout_alignParentBottom="true"
41         android:singleLine="true"
42         android:textSize="13sp"
43         android:textColor="#575757"
44         android:text="Content">
45     </TextView>
46
47 </RelativeLayout>

```

Здесь мы задаём иконку уведомления, заголовок и тело уведомления, прикрепляя их к правому (заголовок относительно иконки) и нижнему (тело уведомления относительно заголовка) углу уведомления.

Теперь затронем наши 2 активности, которые вызываются при нажатии кнопок на уведомлении в течение работы сервиса переднего плана:

В качестве простого примера не будем нагружать их каким-либо функционалом и составим только визуальную составляющую активности (текстовое окно с поздравлениями о переходе в определённую активность).

Файлы java абсолютно идентичны в этом случае:

```

1 package com.example.notific_on_working;
2
3 import ...
4
5
6
7 public class SecondActiv extends AppCompatActivity {
8
9     @Override
10    protected void onCreate(Bundle savedInstanceState) {
11        super.onCreate(savedInstanceState);
12        setContentView(R.layout.activity_second);
13    }
14 }

```



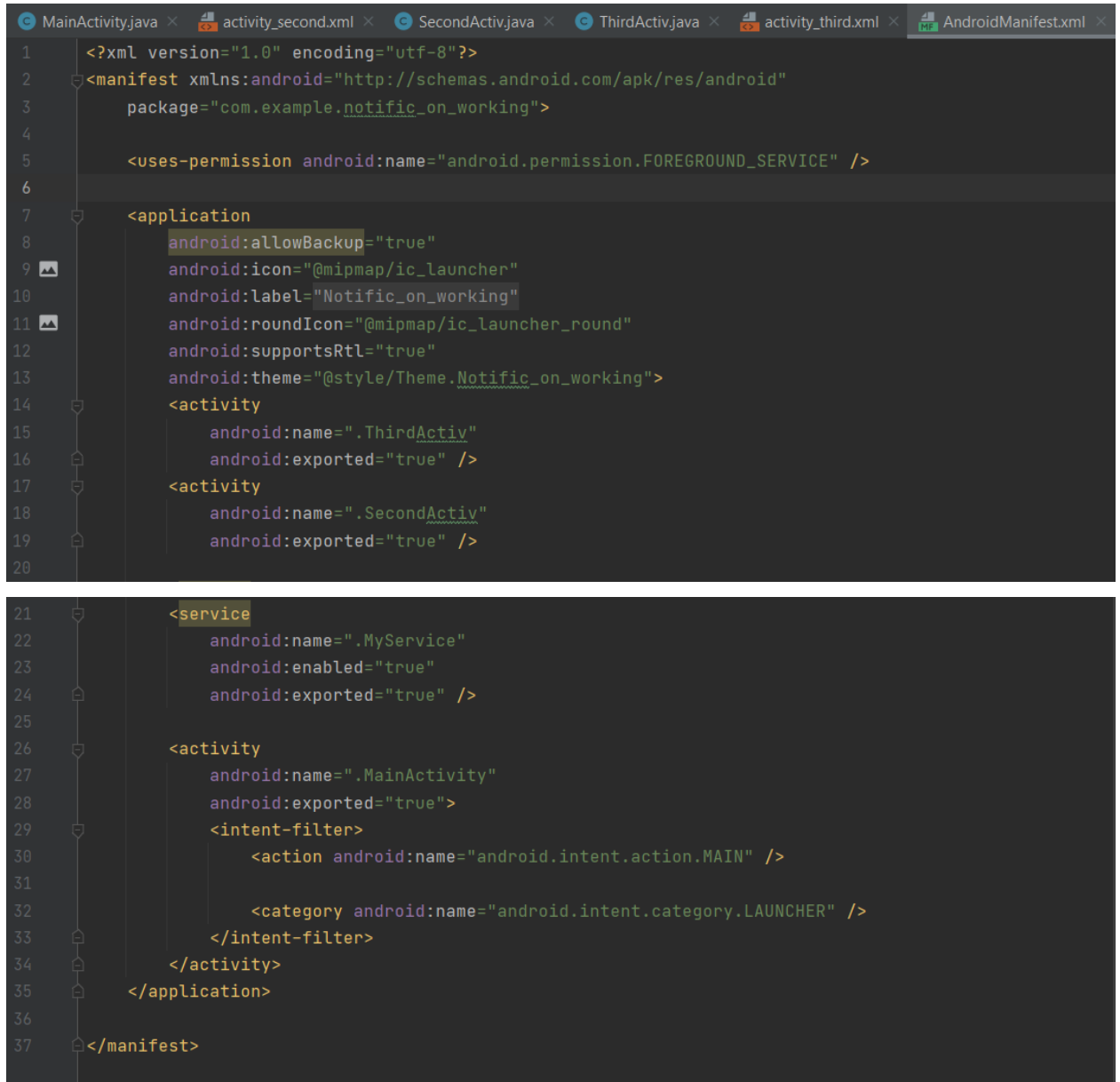
```
MainActivity.java × activity_second.xml × SecondActiv.java × ThirdActiv.java × activity_third.xml × AndroidManifest.xml ×
1 package com.example.notific_on_working;
2
3 import ...
4
5
6 public class ThirdActiv extends AppCompatActivity {
7
8
9     @Override
10    protected void onCreate(Bundle savedInstanceState) {
11        super.onCreate(savedInstanceState);
12        setContentView(R.layout.activity_third);
13    }
14 }
```

В XML-файлах добавим просто текстовое описание номера активности с расположением по центру:

```
MainActivity.java × activity_second.xml × activity_third.xml × AndroidManifest.xml ×
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     android:layout_width="fill_parent"
5     android:layout_height="fill_parent"
6     android:orientation="vertical">
7
8     <TextView
9         android:layout_width="match_parent"
10        android:layout_height="0dp"
11        android:textAlignment="center"
12        android:textSize="30sp"
13        android:textColor="@color/purple_700"
14        android:layout_weight="0.5"
15        android:layout_marginHorizontal="50dp"
16        android:layout_marginVertical="100dp"
17        android:text="It's a 2-nd Activity. You're welcome!">
18    </TextView>
19
20 </LinearLayout>
```

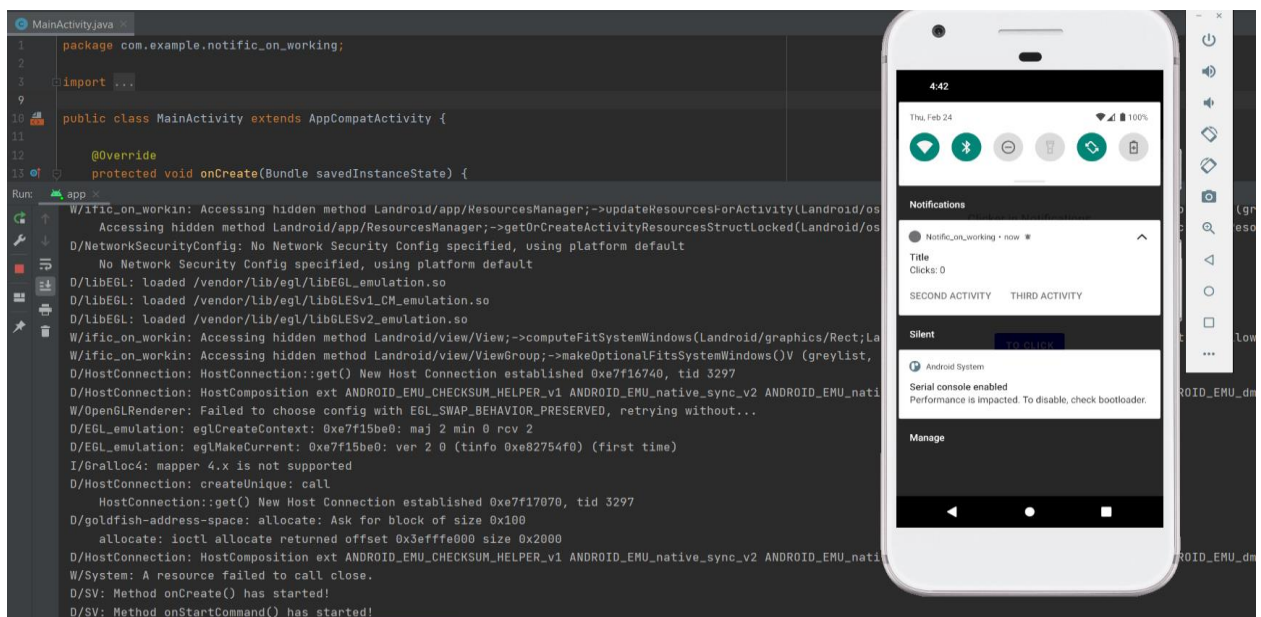
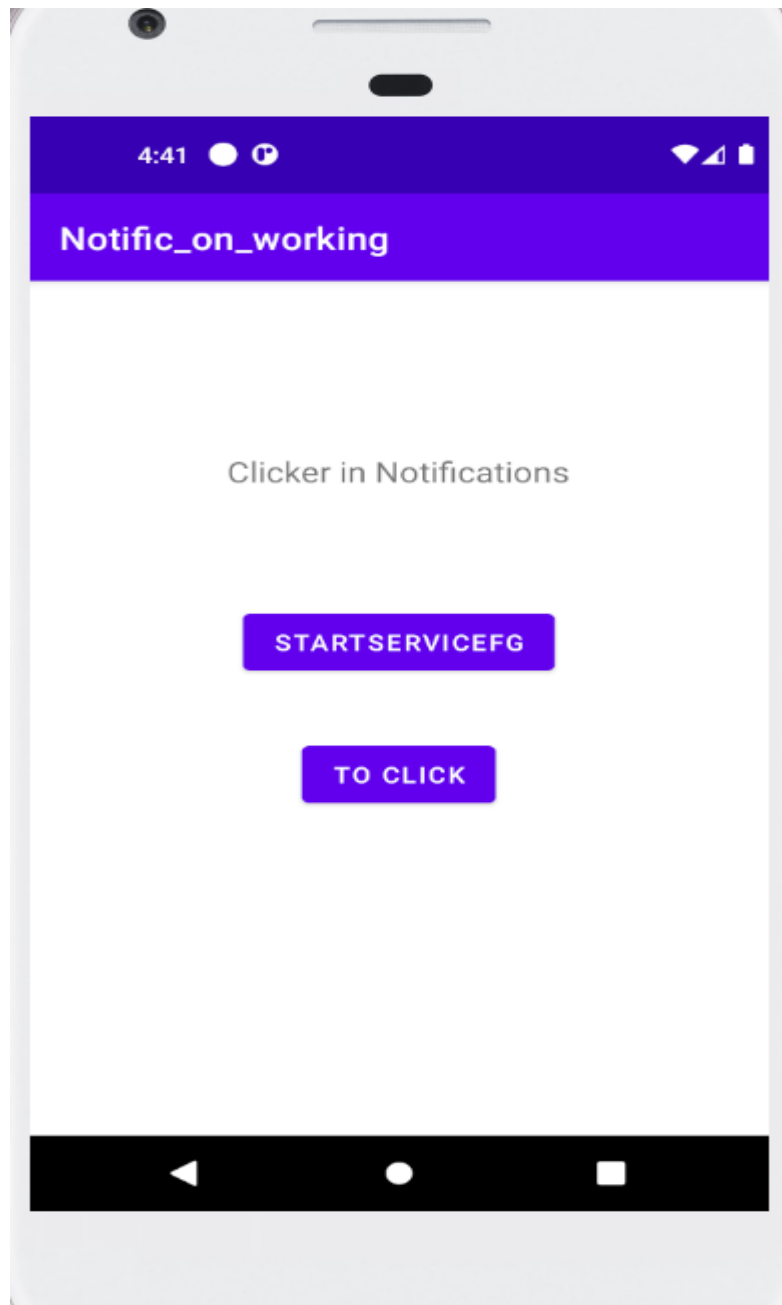
```
MainActivity.java × activity_second.xml × activity_third.xml × AndroidManifest.xml ×
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     android:layout_width="fill_parent"
5     android:layout_height="fill_parent"
6     android:orientation="vertical">
7
8     <TextView
9         android:layout_width="match_parent"
10        android:layout_height="0dp"
11        android:layout_weight="0.5"
12        android:textAlignment="center"
13        android:textColor="@color/bagp_200"
14        android:layout_marginHorizontal="50dp"
15        android:layout_marginVertical="100dp"
16        android:textSize="30sp"
17        android:text="Welcome to Third Activity in this App! Congratulations.">
18    </TextView>
19
20 </LinearLayout>
```

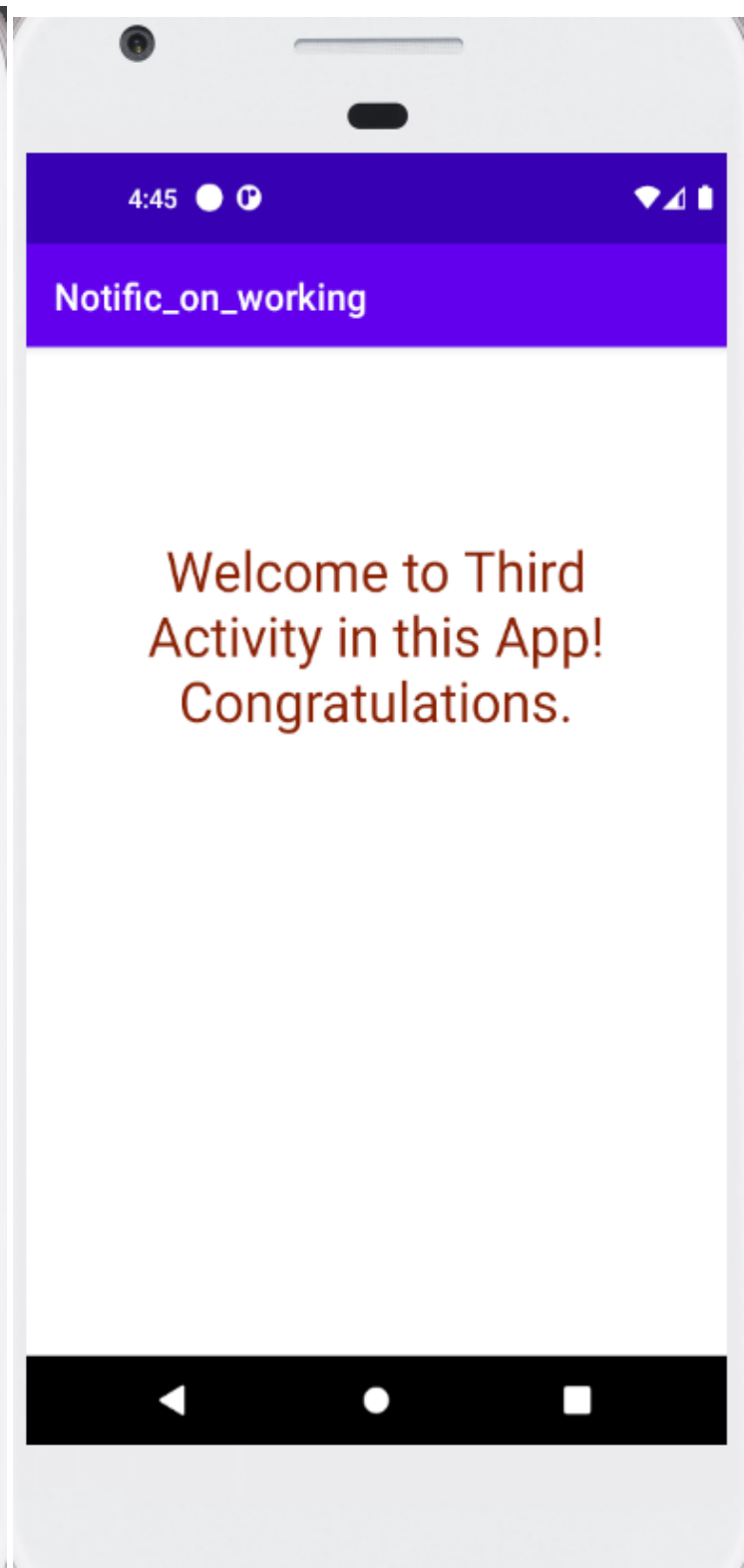
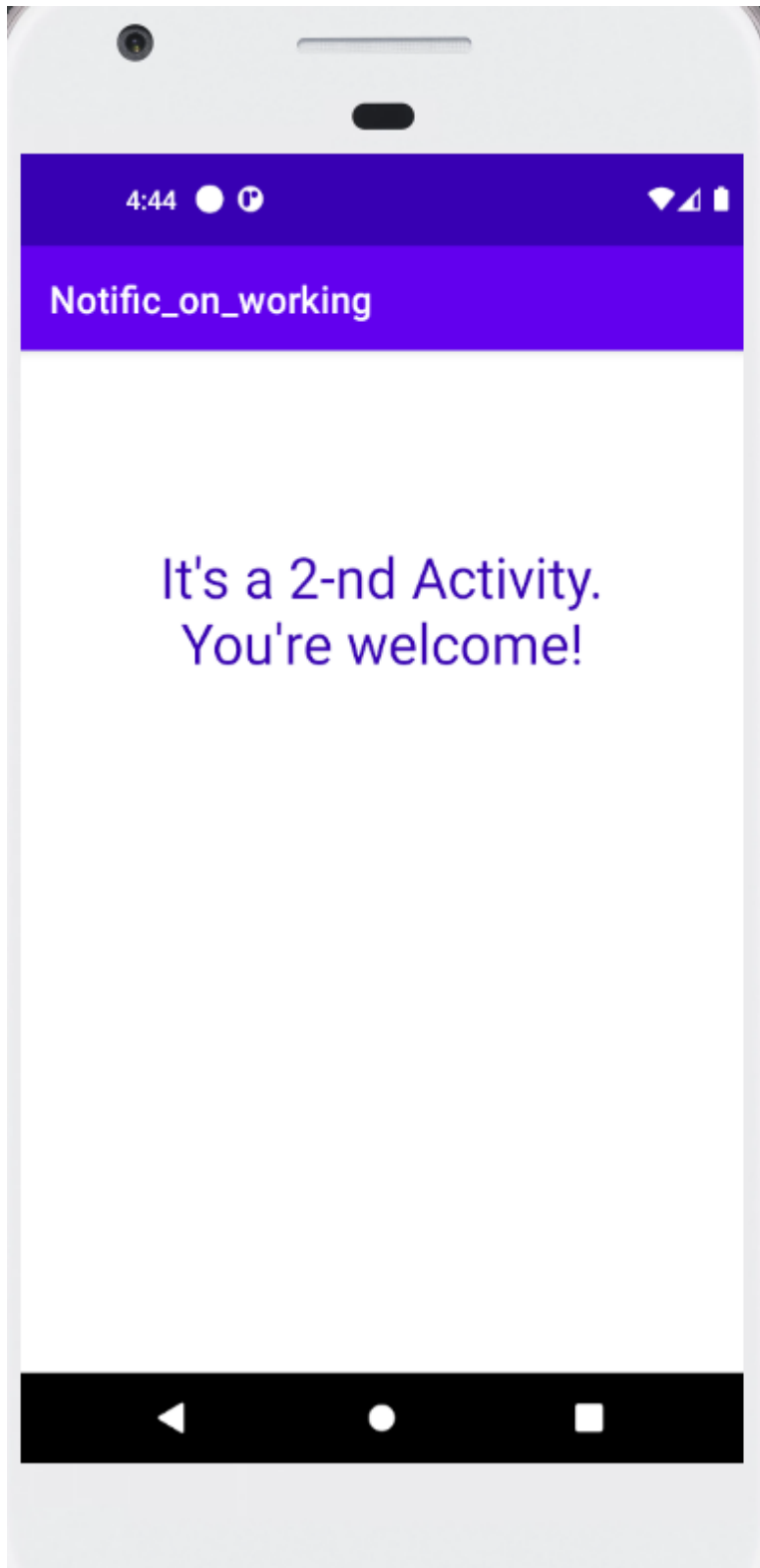
Коснёмся главного файла – манифест приложения, в котором важно и нужно указать 2 важные детали: настроить разрешение на работу сервиса переднего плана в начале и XML-тег самого сервиса с названием java-класса функционала сервиса:



```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="com.example.notific_on_working">
4
5      <uses-permission android:name="android.permission.FOREGROUND_SERVICE" />
6
7      <application
8          android:allowBackup="true"
9          android:icon="@mipmap/ic_launcher"
10         android:label="Notific_on_working"
11         android:roundIcon="@mipmap/ic_launcher_round"
12         android:supportsRtl="true"
13         android:theme="@style/Theme.Notific_on_working">
14         <activity
15             android:name=".ThirdActiv"
16             android:exported="true" />
17         <activity
18             android:name=".SecondActiv"
19             android:exported="true" />
20
21         <service
22             android:name=".MyService"
23             android:enabled="true"
24             android:exported="true" />
25
26         <activity
27             android:name=".MainActivity"
28             android:exported="true">
29             <intent-filter>
30                 <action android:name="android.intent.action.MAIN" />
31
32                 <category android:name="android.intent.category.LAUNCHER" />
33             </intent-filter>
34         </activity>
35     </application>
36
37 </manifest>
```

Именно такую структуру проекта мы имеем в конце разработки приложения. Теперь остаётся увидеть результат работы итогового продукта:





Контрольные вопросы

1. Зачем нужны уведомления?
2. Как работают сервисы переднего плана?
3. Что такое отложенные намерения?

- 1) Уведомления предназначены по своему функционалу для оповещения пользователя о каком-то важном событии (например, завершилось долгожданное обновление), которое случилось в приложении, (в бизнесе) для привлечения внимания потенциальной ЦА к участию в уникальных предложениях, скидках, акциях и других мероприятиях в рамках пиар-кампании. Таким образом, они выступают главным и активным каналом коммуникации между самим приложением и его пользователями, главная особенность которого – показ или сбор информации без нахождения в самом приложении.
- 2) Сервисы переднего плана представляют из себя высшие по приоритету сервисы, выполняющие операции, которые видит пользователь. Они отображают также уведомление в строке состояния, чтобы пользователи были в курсе того, что ваше приложение выполняет задачу на переднем плане и потребляет системные ресурсы. Уведомление не может быть отклонено до тех пор, пока служба не будет остановлена или удалена с переднего плана.
- 3) Отложенные намерения – это описание как самого намерения, так и определённого действия, которое важно и нужно выполнить именно с помощью намерения. Экземпляры класса `PendingIntent` создаются с помощью `getActivity(Context, int, Intent, int)`, `getActivities(Context, int, Intent[], int)`, `getBroadcast(Context, int, Intent, int)` и `getService(Context, int, Intent, int)`; возвращаемый объект может быть передан другим приложениям, чтобы они могли выполнить описанное вами действие от вашего имени позже.