

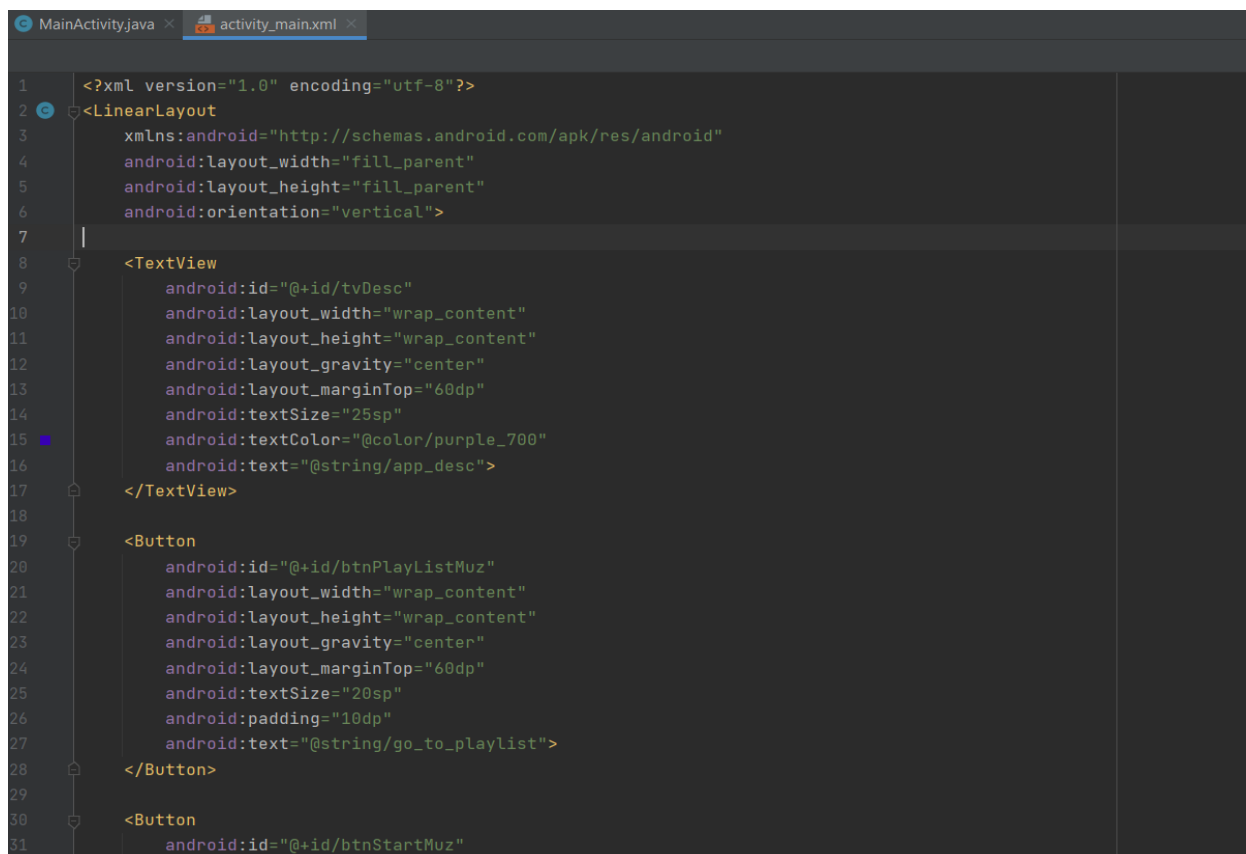
Отчёт по проделанной работе на семинаре по мобильной разработке (04.03.2022).

(выполнил Валяев Георгий Анатольевич)

В наше время в работе большинства многофункциональных мобильных приложений не менее важную и большую (местами рутинную) роль занимают сервисы (небольшая программа как компонент приложения, которая выполняет конкретный пул долгих задач и операций в фоновом режиме без отображения на интерфейсах пользователю).

В данной методичке мы на этот раз попытаемся поработать с инструментами работы музыкального плеера и сервисами, занимающимися прокруткой музыкальных композиций в фоновом режиме, разработав приложение-плеер с прослушиванием плейлиста песен и какой-то одной конкретной песни.

Начнём с файла визуала главной активности (activity_main.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:id="@+id/tvDesc"
10         android:layout_width="wrap_content"
11         android:layout_height="wrap_content"
12         android:layout_gravity="center"
13         android:layout_marginTop="60dp"
14         android:textSize="25sp"
15         android:textColor="@color/purple_700"
16         android:text="@string/app_desc">
17     </TextView>
18
19     <Button
20         android:id="@+id/btnPlayListMuz"
21         android:layout_width="wrap_content"
22         android:layout_height="wrap_content"
23         android:layout_gravity="center"
24         android:layout_marginTop="60dp"
25         android:textSize="20sp"
26         android:padding="10dp"
27         android:text="@string/go_to_playlist">
28     </Button>
29
30     <Button
31         android:id="@+id/btnStartMuz"
```

```
MainActivity.java × activity_main.xml ×
25         android:textSize="20sp"
26         android:padding="10dp"
27         android:text="@string/go_to_playlist">
28     </Button>
29
30     <Button
31         android:id="@+id/btnStartMuz"
32         android:layout_width="wrap_content"
33         android:layout_height="wrap_content"
34         android:layout_gravity="center"
35         android:layout_marginTop="100dp"
36         android:textSize="20sp"
37         android:padding="10dp"
38         android:text="@string/start_listening">
39     </Button>
40
41     <Button
42         android:id="@+id/btnStopMuz"
43         android:layout_width="wrap_content"
44         android:layout_height="wrap_content"
45         android:layout_marginTop="50dp"
46         android:layout_gravity="center"
47         android:padding="10dp"
48         android:textSize="20sp"
49         android:text="@string/stop_listening">
50     </Button>
51
52 </LinearLayout>
```

Здесь мы описали составляющую главной активности приложения: текстовое окно с информацией о приложении, кнопка запуска полноценного плейлиста и кнопки запуска и остановки прослушивания одного трека.

MainActivity.java:

```

1  package com.example.music_player_many_songs;
2
3  import ...
4
5
6
7
8
9
10 public class MainActivity extends AppCompatActivity {
11
12     @Override
13     protected void onCreate(Bundle savedInstanceState) {
14         super.onCreate(savedInstanceState);
15         setContentView(R.layout.activity_main);
16
17         Button btnStart = findViewById(R.id.btnStartMuz);
18         Button btnStop = findViewById(R.id.btnStopMuz);
19         Button btnPlaylist = findViewById(R.id.btnPlayListMuz);
20
21         Intent playService = new Intent(packageContext: MainActivity.this, MyService.class);
22
23         btnStart.setOnClickListener(new View.OnClickListener() {
24             @Override
25             public void onClick(View view) { startService(playService); }
26         });
27
28
29         btnStop.setOnClickListener(new View.OnClickListener() {
30             @Override
31             public void onClick(View view) { stopService(playService); }
32         });
33
34
35         btnPlaylist.setOnClickListener(new View.OnClickListener() {
36             @Override
37             public void onClick(View view) {
38                 startActivity(new Intent(packageContext: MainActivity.this, Playlist.class));
39             }
40         });
41     }
42 }
43

```

В последней 44 строке закрывающая главный класс фигурная скобка.

В этом файле мы инициализируем объекты кнопок, созданных в XML-файле описания визуала активности, запускаем новый интент в отдельный сервис прослушивания одного трека и прорабатываем обработчиками нажатия события, возникающие при нажатии на определённую кнопку (запуск и остановка сервиса прослушивания песни, запуск активности плейлиста треков с передачей интента).

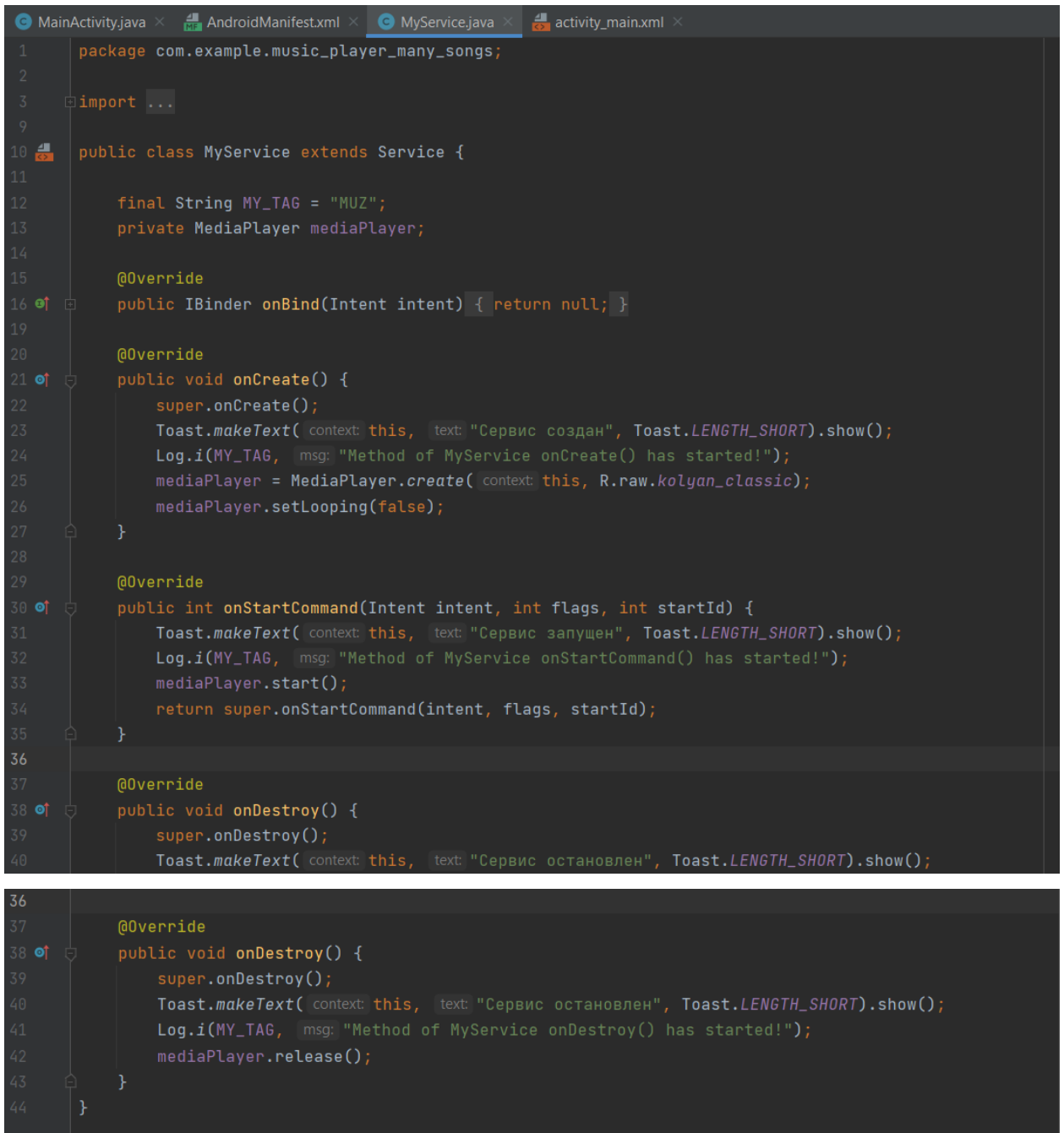
Создавая сервисы и активности, мы отражаем их в главном манифесте приложения:

```

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

```

Далее мы переходим к созданию наших сервисов и начнём обзор с MyService.java:



```

1  package com.example.music_player_many_songs;
2
3  import ...
4
5
6
7
8
9
10 public class MyService extends Service {
11
12     final String MY_TAG = "MUZ";
13     private MediaPlayer mediaPlayer;
14
15     @Override
16     public IBinder onBind(Intent intent) { return null; }
17
18
19
20     @Override
21     public void onCreate() {
22         super.onCreate();
23         Toast.makeText( context: this, text: "Сервис создан", Toast.LENGTH_SHORT).show();
24         Log.i(MY_TAG, msg: "Method of MyService onCreate() has started!");
25         mediaPlayer = MediaPlayer.create( context: this, R.raw.kolyan_classic);
26         mediaPlayer.setLooping(false);
27     }
28
29
30     @Override
31     public int onStartCommand(Intent intent, int flags, int startId) {
32         Toast.makeText( context: this, text: "Сервис запущен", Toast.LENGTH_SHORT).show();
33         Log.i(MY_TAG, msg: "Method of MyService onStartCommand() has started!");
34         mediaPlayer.start();
35         return super.onStartCommand(intent, flags, startId);
36     }
37
38
39     @Override
40     public void onDestroy() {
41         super.onDestroy();
42         Toast.makeText( context: this, text: "Сервис остановлен", Toast.LENGTH_SHORT).show();
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

В этом сервисе мы прорабатываем функционал каждого метода его жизненного цикла, начав с инициализации тэга для будущего логирования событий в консоль и объекта музыкального плеера внутри класса.

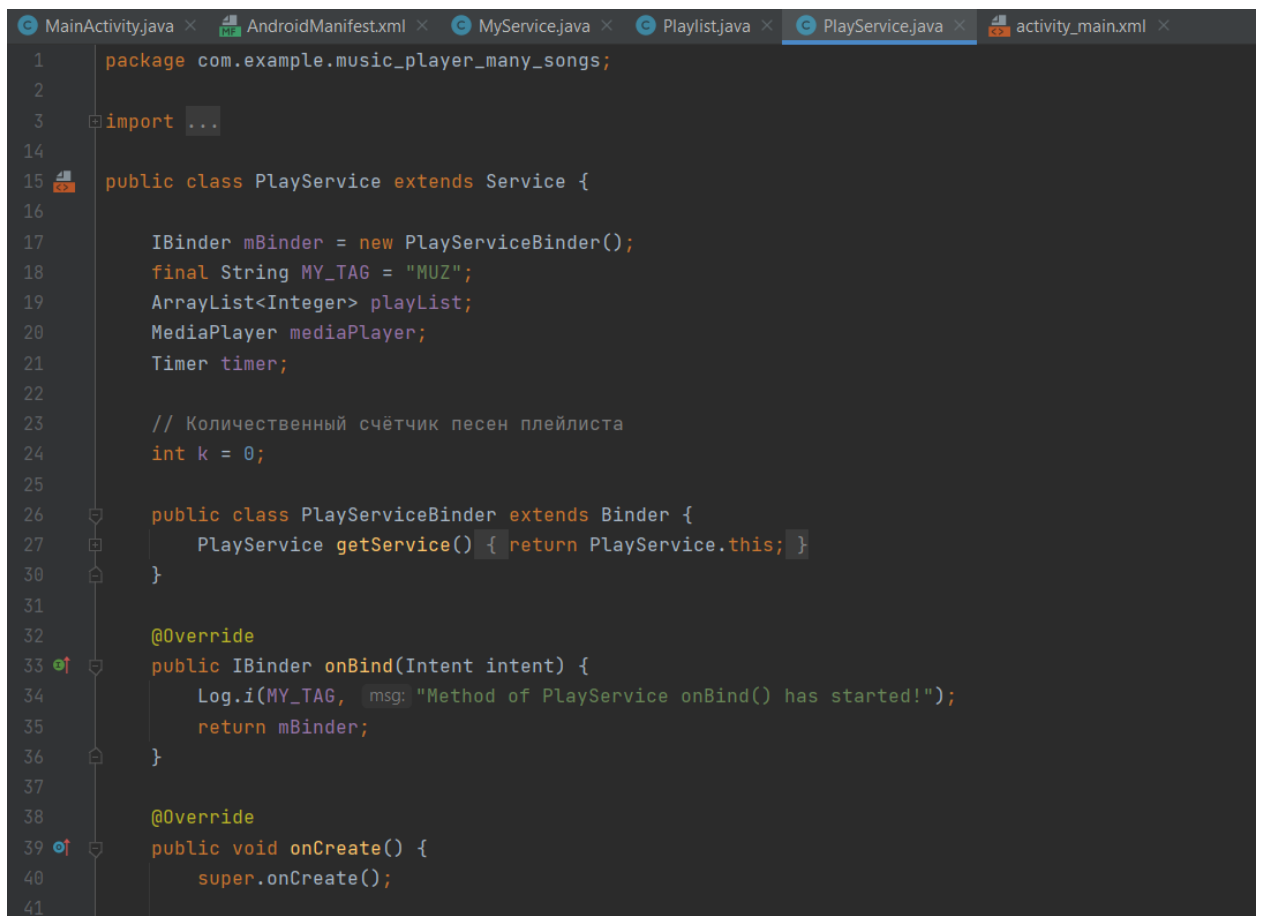
Метод `onBind()` нас интересует в приложении в меньшей степени, но он также является основным в жизненном цикле, поэтому пропишем в него интент и поставим пустое значение на возвращение.

В `onCreate()` мы сообщаем пользователю через всплывающее уведомление о начале работы сервиса, прологировав его в консоль. Далее создаём объект медиаплеера, передав в него файл одного трека в директории `raw` проекта (в нём хранятся все наши треки плейлиста), и отключаем его циклический повтор через метод `setLooping()`.

Аналогично в методе `onStartCommand()` мы запускаем наш сервис с проигрыванием трека, уведомив пользователя и записав событие в консоль.

При отключении сервиса в методе `onDestroy()` также запускаем соответствующее уведомление и запись в консоли об отключении сервиса, при этом также высвобождая производительные ресурсы плеера и завершая его работу через метод `.release()`.

Переходим к функционалу плейлиста (`PlayService.java`):



```
1 package com.example.music_player_many_songs;
2
3 import ...
4
14
15 public class PlayService extends Service {
16
17     IBinder mBinder = new PlayServiceBinder();
18     final String MY_TAG = "MUZ";
19     ArrayList<Integer> playList;
20     MediaPlayer mediaPlayer;
21     Timer timer;
22
23     // Количественный счётчик песен плейлиста
24     int k = 0;
25
26     public class PlayServiceBinder extends Binder {
27         PlayService getService() { return PlayService.this; }
28     }
29
30
31
32     @Override
33     public IBinder onBind(Intent intent) {
34         Log.i(MY_TAG, "Method of PlayService onBind() has started!");
35         return mBinder;
36     }
37
38     @Override
39     public void onCreate() {
40         super.onCreate();
41     }
```

```

38     @Override
39     public void onCreate() {
40         super.onCreate();
41
42         // Полноценный массив треков плейлиста
43         playList = new ArrayList<>();
44         playList.add(R.raw.tischiny_hochu);
45         playList.add(R.raw.call_3739);
46         playList.add(R.raw.call_zhivi);
47         playList.add(R.raw.federiko_fellini);
48         playList.add(R.raw.kolyan_classic);
49         playList.add(R.raw.rampampam_ring);
50         playList.add(R.raw.ringtone_babel);
51         playList.add(R.raw.tak_chochetsa_zhit);
52
53         mediaPlayer = MediaPlayer.create(context: this, playList.get(0));
54         mediaPlayer.start();
55         Log.i(MY_TAG, msg: "Method of PlayService onCreate() has started!");
56         Log.i(MY_TAG, msg: "PlayList has started!");
57
58         timer = new Timer();
59         if (playList.size() > 1) nextSong();
60     }

```

```

60     }
61
62     public void nextSong() {
63         timer.schedule(() -> {
64             mediaPlayer.reset();
65             mediaPlayer = MediaPlayer.create(context: PlayService.this, playList.get(++k));
66             mediaPlayer.start();
67             if (playList.size() > k + 1) {
68                 nextSong();
69             }
70         }, mediaPlayer.getDuration());
71         Log.i(MY_TAG, msg: "Method of PlayService nextSong() has started!");
72     }
73
74 }
75
76

```

```

76
77     public void playNextSong() {
78         if (mediaPlayer.isPlaying()) {
79             mediaPlayer.stop();
80         }
81         timer.cancel();
82         mediaPlayer.reset();
83         if (playList.size() > k + 1) {
84             Toast.makeText(PlayService.this, getApplicationContext(),
85                 text: "Next трек по кнопке",
86                 Toast.LENGTH_SHORT).show();
87             mediaPlayer = MediaPlayer.create(context: PlayService.this, playList.get(++k));
88             mediaPlayer.start();
89             timer = new Timer();
90             Log.i(MY_TAG, msg: "Method of PlayService playNextSong() has started!");
91             nextSong();
92         } else {
93             Toast.makeText(PlayService.this, getApplicationContext(),
94                 text: "Треки закончились!",
95                 Toast.LENGTH_SHORT).show();
96             k -= 1;
97             Log.i(MY_TAG, msg: "Method of PlayService playNextSong() has stopped!");
98         }
99     }

```

```

92         } else {
93             Toast.makeText(PlayService.this.getApplicationContext(),
94                 text: "Треки закончились!",
95                 Toast.LENGTH_SHORT).show();
96             k -= 1;
97             Log.i(MY_TAG, msg: "Method of PlayService playNextSong() has stopped!");
98         }
99     }
100
101     @Override
102     public void onDestroy() {
103         Log.i(MY_TAG, msg: "Method of PlayService onDestroy() has started!");
104         if (mediaPlayer.isPlaying()) {
105             mediaPlayer.stop();
106         }
107         timer.cancel();
108         super.onDestroy();
109     }
110 }

```

Во многом механика работы сервиса схожа с нашим небольшим сервисом, однако есть тонкости в функциональности плейлиста:

В процессе инициализации объекта нашего класса, наследуемого свойства класса Binder, связанного элемента сервиса, вызываемого методом onBind() мы также указываем объекты медиаплеера, таймера и массива из объектов треков нашего плейлиста. Также не забудем о теге для логирования событий плейлиста и счётчике количества песен.

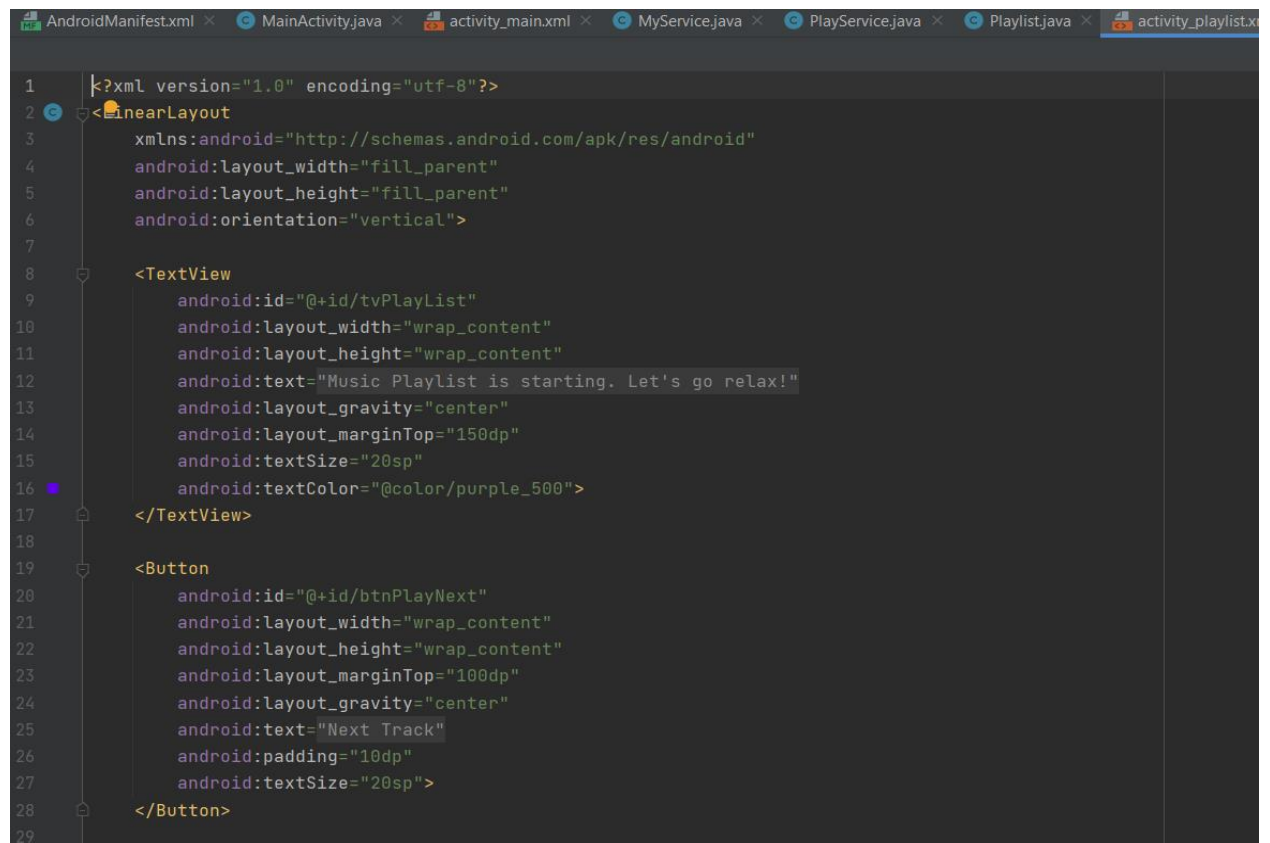
В методе onCreate(), запуская сервис, мы добавляем все наши песни в массив, создаём объект медиаплеера и запускаем его совместно со включением таймера. В конце для переключения на новый трек прописываем условие объёма загруженности объекта плейлиста.

Прописывая метод nextSong(), мы прорабатываем алгоритм переключения плейлиста и медиаплеера на следующий трек и логирования события в консоли: приостановка работы плеера, запуск новой песни и увеличение счётчика количества треков.

В методе `playNextSong()` мы реализовываем принудительное переключение на следующий трек в плейлисте. При работе плеера мы его отключаем временно через условие `isPlaying()`, после чего отключаем таймер и сбрасываем объект плеера в начальное состояние перед проверкой индекса прослушиваемого перед нажатием кнопки трека: если трек не был последним, то мы уведомляем пользователя о включении следующего трека и записываем событие в консоль, создаём и запускаем следующий трек в объекте медиаплеера совместно с таймером, также проходя алгоритм переключения на следующий трек. В обратном условии уведомляем пользователя о том, что треки закончились, записываем событие в консоль и специально уменьшаем счётчик количества песен на 1 в случае, если пользователь намеренно пытается прослушать следующий трек и хочет повторить прослушивание последнего трека.

В методе `onDestroy()` отключаем полностью сервис (медиаплеер совместно с таймером) и записываем это событие в консоли.

Рассмотрим теперь визуал нашей активности плейлиста:



```
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:id="@+id/tvPlayList"
10        android:layout_width="wrap_content"
11        android:layout_height="wrap_content"
12        android:text="Music Playlist is starting. Let's go relax!"
13        android:layout_gravity="center"
14        android:layout_marginTop="150dp"
15        android:textSize="20sp"
16        android:textColor="@color/purple_500">
17    </TextView>
18
19    <Button
20        android:id="@+id/btnPlayNext"
21        android:layout_width="wrap_content"
22        android:layout_height="wrap_content"
23        android:layout_marginTop="100dp"
24        android:layout_gravity="center"
25        android:text="Next Track"
26        android:padding="10dp"
27        android:textSize="20sp">
28    </Button>
29
```

```

28     </Button>
29
30     <Button
31         android:id="@+id/btnUnBind"
32         android:layout_width="wrap_content"
33         android:layout_height="wrap_content"
34         android:layout_gravity="center"
35         android:layout_marginTop="50dp"
36         android:text="Unbind Service"
37         android:textSize="20sp"
38         android:padding="12dp">
39     </Button>
40
41 </LinearLayout>

```

Его структура вполне схожа с главной активностью: текстовое окно с информацией о самой активности, кнопки переключения на следующий трек и отключения связки сервиса с нашей активностью (отключения сервиса плейлиста).

Рассмотрим последний главный и основной файл нашего приложения – файл Playlist.java:

```

AndroidManifest.xml x MainActivity.java x activity_main.xml x MyService.java x PlayService.java x activity_playlist.xml x Playlist.java x
1  package com.example.music_player_many_songs;
2
3  import ...
4
13
14  public class Playlist extends AppCompatActivity {
15
16      PlayService playService;
17      boolean serviceConnected;
18
19      @Override
20      protected void onCreate(Bundle savedInstanceState) {
21          super.onCreate(savedInstanceState);
22          setContentView(R.layout.activity_playlist);
23
24          Button btnPlayNext = findViewById(R.id.btnPlayNext);
25          Button btnUnbind = findViewById(R.id.btnUnBind);
26
27          Intent playServiceIntent = new Intent( packageContext: this, PlayService.class);
28
29          ServiceConnection mConnection = new ServiceConnection() {
30              @Override
31              public void onServiceConnected(ComponentName className, IBinder service) {
32                  PlayServiceBinder binder = (PlayServiceBinder) service;
33                  playService = binder.getService();
34                  serviceConnected = true;
35              }
36
37              @Override
38              public void onServiceDisconnected(ComponentName componentName) {
39                  serviceConnected = false;
40              }
41          };
42

```

```

36
37         @Override
38         public void onServiceDisconnected(ComponentName componentName) {
39             serviceConnected = false;
40         }
41     };
42
43     bindService(playServiceIntent, mConnection, BIND_AUTO_CREATE);
44
45     // Включаем следующий трек
46     btnPlayNext.setOnClickListener(new View.OnClickListener() {
47         @Override
48         public void onClick(View view) { playService.playNextSong(); }
49     });
50
51     btnUnbind.setOnClickListener(new View.OnClickListener() {
52         @Override
53         public void onClick(View view) {
54             if (serviceConnected) {
55                 unbindService(mConnection);
56                 serviceConnected = false;
57             }
58         }
59     });
60
61 }
62
63 }

```

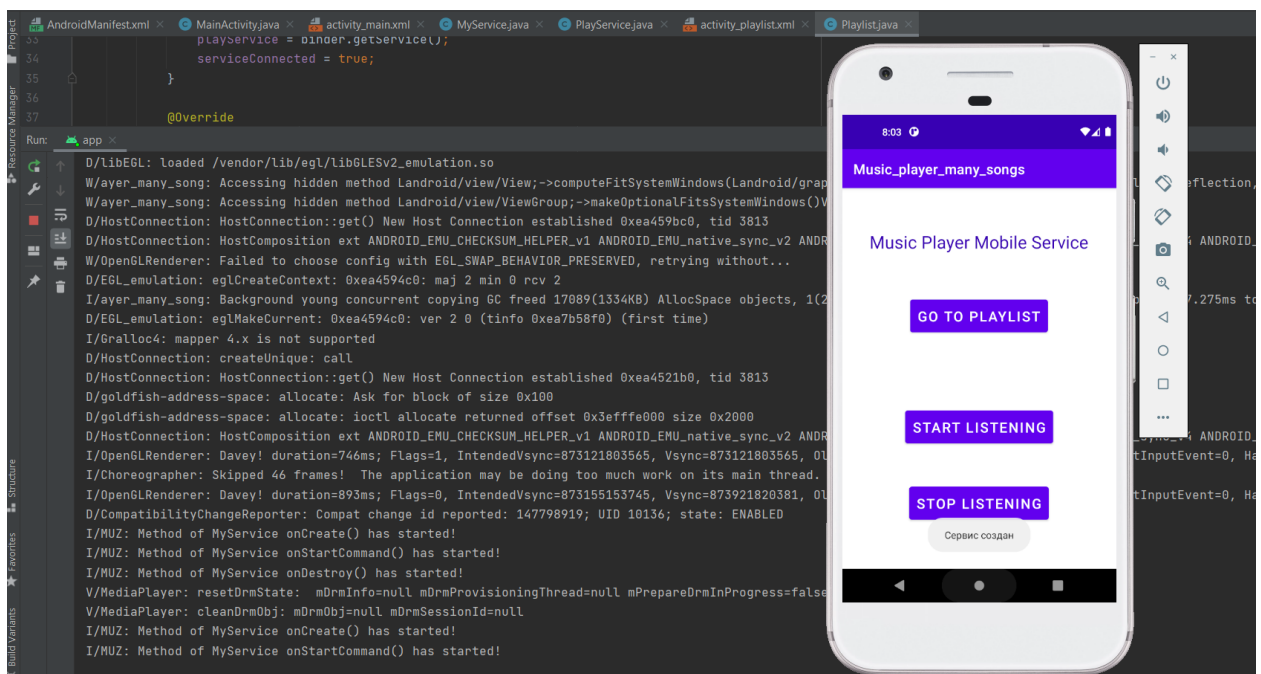
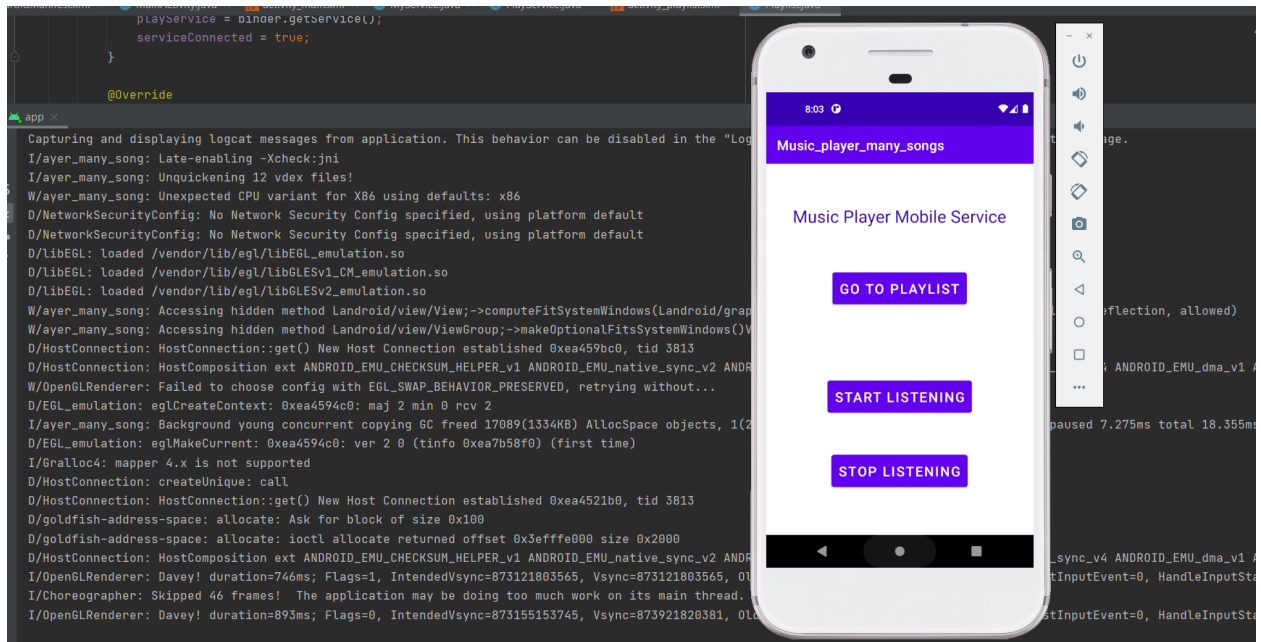
В этой активности мы работаем с обработчиками событий наших кнопок по переключению на следующий трек и отключению соединения с сервисом, однако этот функционал мы реализуем через экземпляр объекта `ServiceConnection()`, в котором прописываем 2 дополнительных метода по включению и отключению соединения с нашим сервисом по работе плейлиста треков (логическая переменная `serviceConnected` играет здесь особо важную роль регулировщика задач).

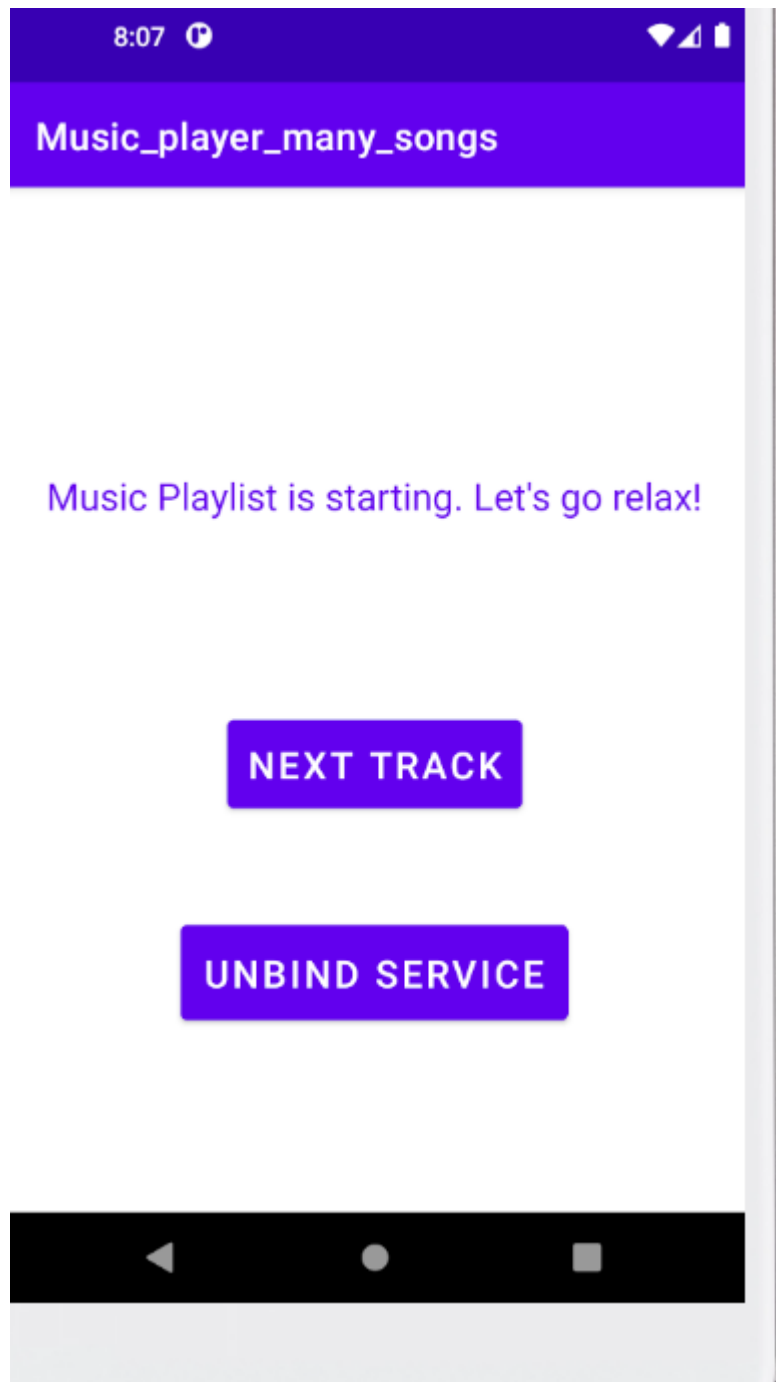
При включении соединения мы присваиваем новому объекту соединения binder объект класса `PlayServiceBinder` `service` как принимающий соединения клиент сервиса плейлиста, с которым мы устанавливаем соединение и определяем логической переменной состояние `true`.

Через метод `bindService` мы производим процесс подключения и начала работы сервиса, исходя из значения передаваемого параметра `mConnection`.

И уже при отключении соединения через кнопку мы переопределяем состояние подключения к сервису через метод `unbindService()` и значение `false` у переменной `serviceConnected`.

Пора смотреть получившийся результат:





```
clearDRMObj: DRMObj=null DRMSessionId=null
D/CompatibilityChangeReporter: Compat change id reported: 147798919; UID 10136; state: ENABLED
I/MUZ: Method of PlayService playNextSong() has started!
I/MUZ: Method of PlayService nextSong() has started!
I/MUZ: Method of PlayService onDestroy() has started!
```