



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных
технологий

Отчет по практической работе №11

по дисциплине «Разработка мобильных приложений»

Выполнил:

Студент группы ИКБО-20-23

Комисарик М.А.

Проверил:

Старший преподаватель кафедры
МОСИТ

Шешуков Л.С.

Москва 2025 г.

СОДЕРЖАНИЕ

1 ТЕОРЕТИЧЕСКОЕ ВВЕДЕНИЕ	3
1.1 WebView	3
1.2 Работа с мультимедиа	6
1.3 Анимации в Android.....	7
1.3.1 Анимация вращения элемента	8
1.3.2 Анимация перемещения элемента по экрану	10
1.3.3 Анимация изменения размера элемента	12
1.4 Уведомления	13
2 ПРАКТИЧЕСКОЕ ЗАДАНИЕ	18
2.1 WebView	18
2.2 Воспроизведения аудио из интернета	20
2.3 Анимации	21
2.3.1 Разметка.....	21
2.3.2 Реализация.....	22
2.3.3 Тестирование	23
2.4 Уведомления	25
2.4.1 Разрешения.....	25
2.4.2 Разметка.....	25
2.4.3 Реализация.....	26
2.4.4 Тестирование	28
ЗАКЛЮЧЕНИЕ	30

1 ТЕОРЕТИЧЕСКОЕ ВВЕДЕНИЕ

1.1 WebView

WebView в Android — это компонент, который позволяет разработчикам показывать веб-страницы внутри приложения. Это своего рода встроенный браузер, управляемый при помощи класса WebView. Он используется для отображения веб-контента, такого как пользовательские страницы для авторизации, справочные материалы или даже некоторые виды интерактивного контента внутри приложения.

Основные функции и использование WebView:

- отображение веб-страниц: можно загружать HTML-страницы из интернета или загружать их как локальные ресурсы;
- интерактивность: пользователи могут взаимодействовать со страницами, как в обычном веб-браузере;
- JavaScript: WebView поддерживает выполнение JavaScript, что позволяет взаимодействовать с веб-страницей и обрабатывать пользовательский ввод;
- настройка и безопасность: разработчики могут настраивать поведение WebView, включая управление кэшированием, cookies, историей просмотров и т.д. Важно правильно настроить параметры безопасности, чтобы избежать уязвимостей.

Рассмотрим пример использования WebView.

Сначала идёт добавление WebView в XML макет Activity (Рисунок 1).

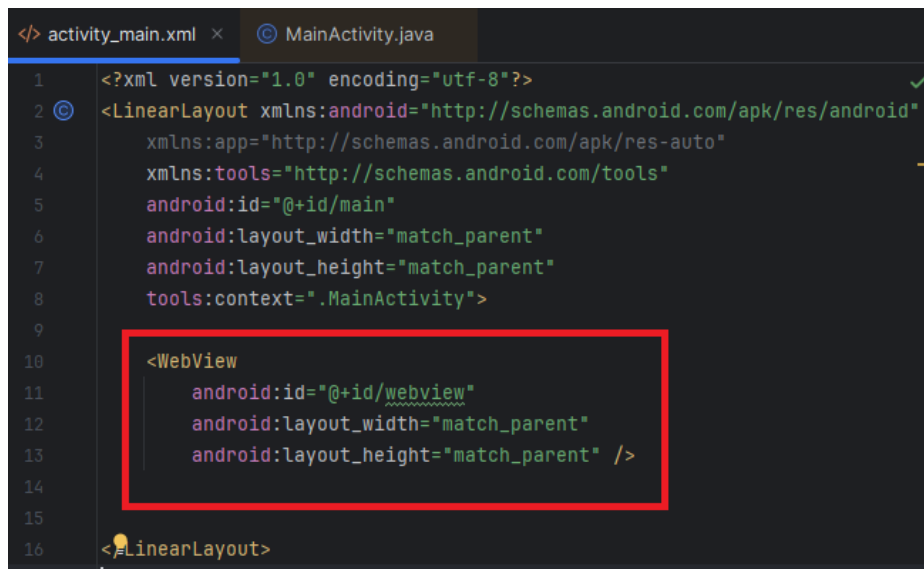


Рисунок 1 – Добавление WebWiev в XML макет Activity

Далее необходимо настроить WebView в коде Activity (Рисунок 2).

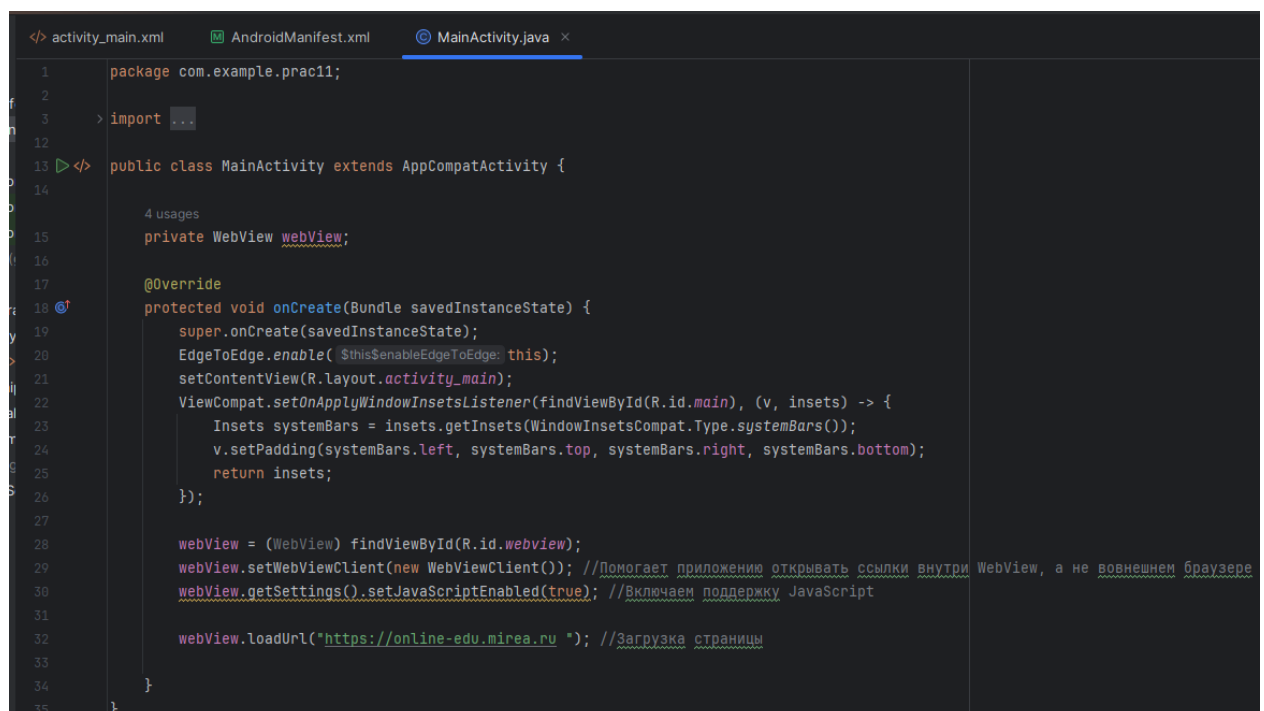
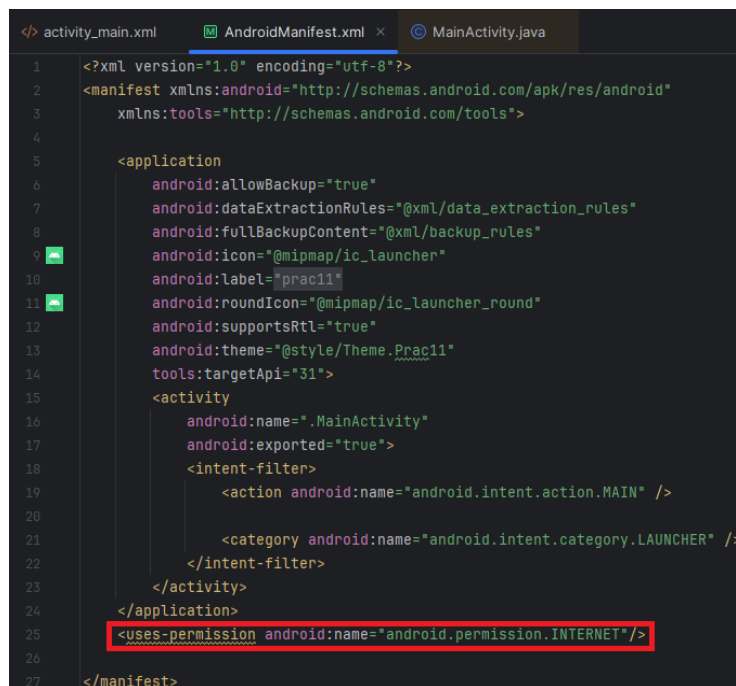


Рисунок 2 – Настройка WebView в коде Activity

Важные моменты:

- WebViewClient: этот класс помогает управлять загрузкой различных URL и следить за процессом загрузки веб-страницы;
- JavaScript: включение JavaScript необходимо для многих современных веб-страниц, которые зависят от клиентских скриптов для пользовательской интерактивности.

Для получения доступа к интернету из приложения, необходимо указать в файле манифеста AndroidManifest.xml соответствующее разрешение (Рисунок 3).



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/prac11"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportRtl="true"
        android:theme="@style/Theme.Prac11"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <uses-permission android:name="android.permission.INTERNET"/>
    </application>
</manifest>
```

Рисунок 3 – Добавление разрешения на выход в интернет

При запуске приложения отобразится выбранный сайт (Рисунок 4).

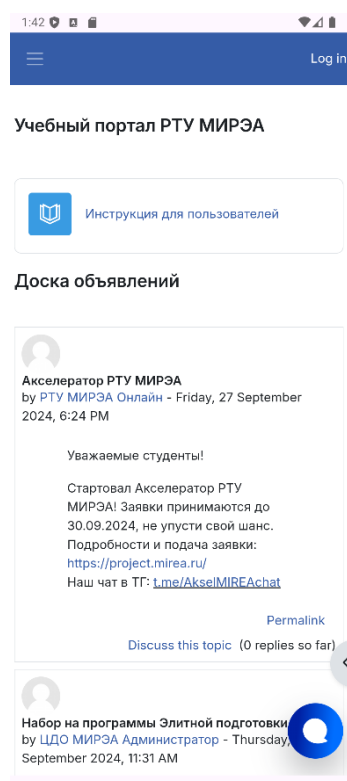
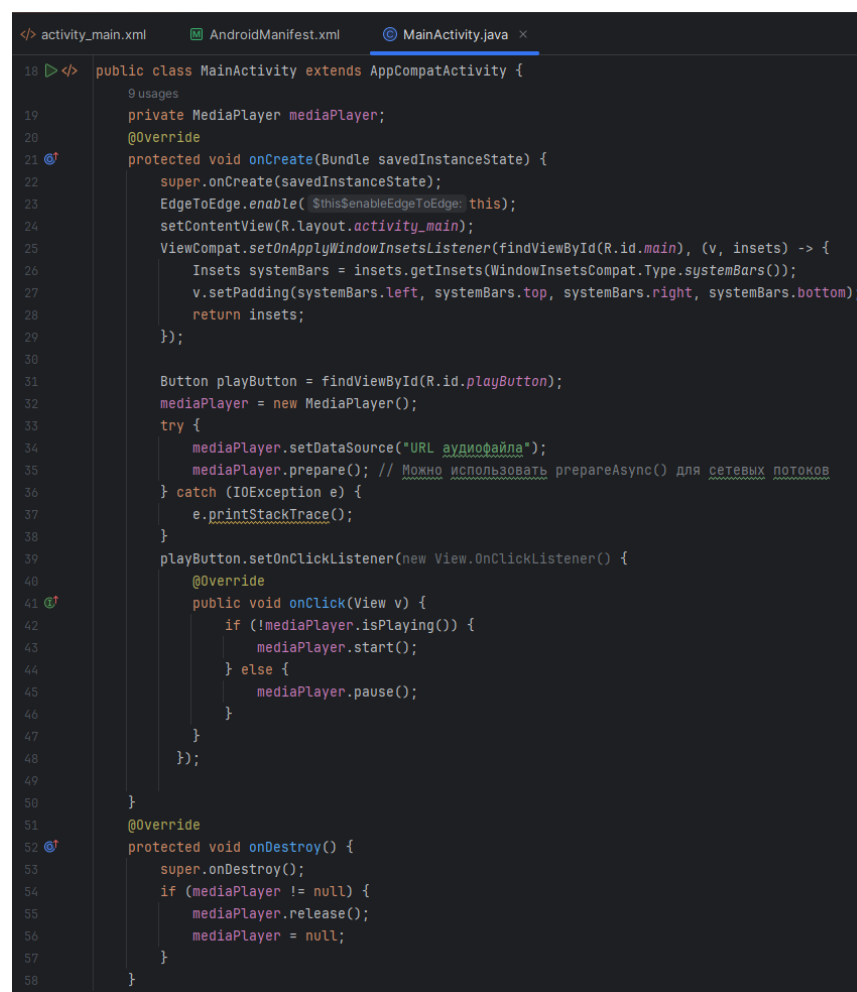


Рисунок 4 – Отображение выбранного сайта в WebView

1.2 Работа с мультимедиа

Работа с мультимедиа в Android — это важная часть разработки мобильных приложений, так как позволяет включать в приложения такие функции, как воспроизведение музыки, видео, захват изображений и видео с камеры, и обработка звука. Это улучшает интерактивность и функциональность приложения. Android предоставляет класс `MediaPlayer` для воспроизведения медиафайлов. Эти файлы могут быть локальными (хранятся на устройстве) или удаленными (доступны через интернет). Ранее был рассмотрен пример воспроизведения локального файла, теперь рассмотрим воспроизведение удаленного файла.

В код `activity` необходимо добавить логику создания и управления `MediaPlayer` (Рисунок 5).



```
<? activity_main.xml  AndroidManifest.xml  MainActivity.java x
18  <> public class MainActivity extends AppCompatActivity {
19      9 usages
20      private MediaPlayer mediaPlayer;
21      @Override
22      protected void onCreate(Bundle savedInstanceState) {
23          super.onCreate(savedInstanceState);
24          EdgeToEdge.enable( $this$enableEdgeToEdge: this);
25          setContentView(R.layout.activity_main);
26          ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
27              Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
28              v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
29              return insets;
30          });
31
32          Button playButton = findViewById(R.id.playButton);
33          mediaPlayer = new MediaPlayer();
34          try {
35              mediaPlayer.setDataSource("URL аудиофайла");
36              mediaPlayer.prepare(); // Можно использовать prepareAsync() для сетевых потоков
37          } catch (IOException e) {
38              e.printStackTrace();
39          }
40
41          playButton.setOnClickListener(new View.OnClickListener() {
42              @Override
43              public void onClick(View v) {
44                  if (!mediaPlayer.isPlaying()) {
45                      mediaPlayer.start();
46                  } else {
47                      mediaPlayer.pause();
48                  }
49              }
50          });
51
52      @Override
53      protected void onDestroy() {
54          super.onDestroy();
55          if (mediaPlayer != null) {
56              mediaPlayer.release();
57              mediaPlayer = null;
58          }
59      }
60  }
```

Рисунок 5 – Логика работы `MediaPlayer`

Этот пример демонстрирует базовый способ воспроизведения аудио с использованием MediaPlayer. Следует обратить внимание на обработку исключений и корректное освобождение ресурсов в методе onDestroy().

Работая с мультимедиа, важно помнить о потенциальных исключениях и о необходимости управления ресурсами, чтобы избежать утечек памяти и сохранить стабильность приложения.

1.3 Анимации в Android

Анимации в Android используются для улучшения пользовательского интерфейса путём добавления визуальных эффектов при взаимодействии с элементами приложения. Они могут помочь сделать приложение более динамичным и интересным, а также интуитивно понятным, подчеркивая изменения состояний, переходы между активностями или фрагментами и реакции на действия пользователя.

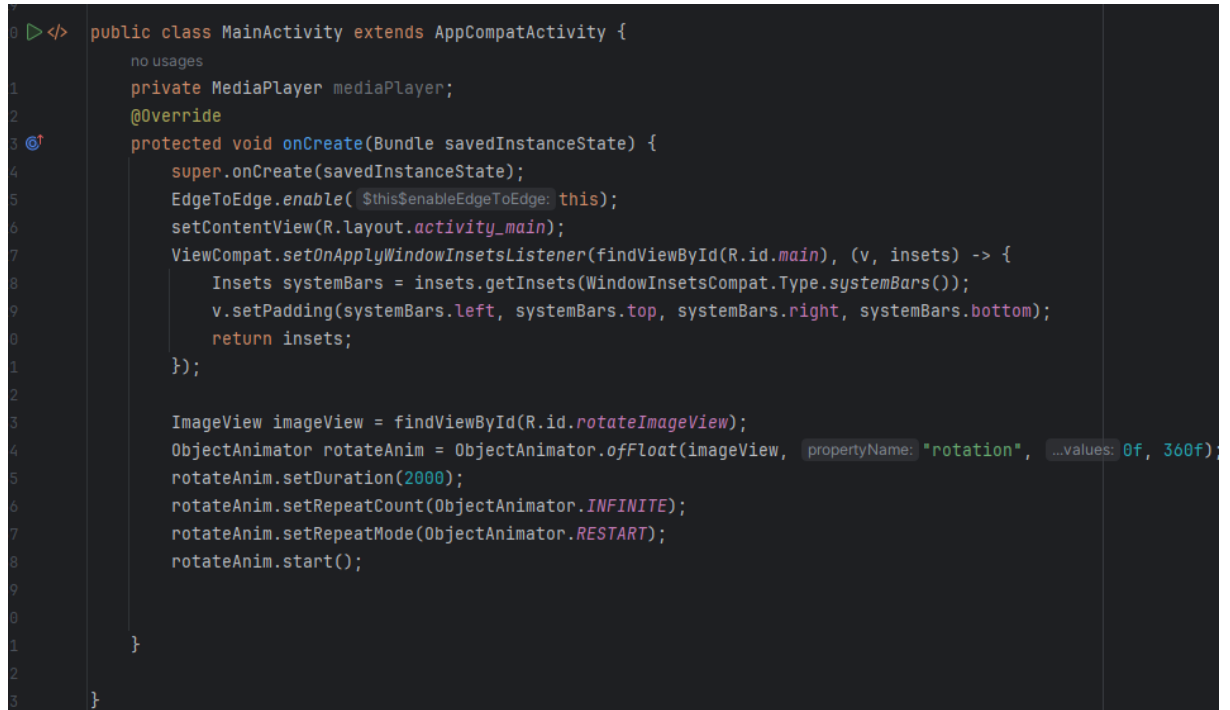
Типы анимации в Android:

- View анимации: это простые анимации, которые можно применять к элементам интерфейса (views), такие как повороты, масштабирование, смещения и прозрачность. Они описываются в XML и могут быть запущены в коде;
- анимация свойств (Property Animations): эти анимации предоставляют более сложный контроль, позволяя анимировать любое свойство объекта, такое как цвет фона, позиция или размер. Наиболее популярным классом для таких анимаций является ObjectAnimator;
- анимации переходов: используются для анимации переходов между активностями или фрагментами, позволяя создавать сложные анимации при переключении между экранами.

Рассмотрим несколько примеров анимации элементов.

1.3.1 Анимация вращения элемента

Для анимации вращения элемента необходимо воспользоваться таким классом, как `ObjectAnimator` (Рисунок 6).



```
0  </> public class MainActivity extends AppCompatActivity {  
1      no usages  
2      private MediaPlayer mediaPlayer;  
3      @Override  
4      protected void onCreate(Bundle savedInstanceState) {  
5          super.onCreate(savedInstanceState);  
6          EdgeToEdge.enable(this);  
7          setContentView(R.layout.activity_main);  
8          ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {  
9              Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());  
10             v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);  
11             return insets;  
12         });  
13  
14         ImageView imageView = findViewById(R.id.rotateImageView);  
15         ObjectAnimator rotateAnim = ObjectAnimator.ofFloat(imageView, "rotation", 0f, 360f);  
16         rotateAnim.setDuration(2000);  
17         rotateAnim.setRepeatCount(ObjectAnimator.INFINITE);  
18         rotateAnim.setRepeatMode(ObjectAnimator.RESTART);  
19         rotateAnim.start();  
20     }  
21 }  
22 }
```

Рисунок 6 – Логика анимации вращения элемента

`ObjectAnimator` – это подкласс `ValueAnimator`, который позволяет нам устанавливать целевой объект и свойство объекта для анимации. Это простой способ изменить свойства вида с указанной продолжительностью. Мы можем указать конечную позицию и продолжительность анимации.

Методы:

- `ofFloat()`: создает и возвращает `ObjectAnimation`, который анимирует координаты;
- `setDuration()`: устанавливает продолжительность анимации;
- `getRepeatCount()` – определяет, сколько раз должна повторяться анимация;
- `getRepeatMode()` – определяет, что должна делать эта анимация, когда она дойдет до конца.

При запуске приложения выбранный ImageView начнёт вращаться по часовой стрелке (Рисунки 7-8).



Рисунок 7 – Отображение вращения ImageView, часть 1

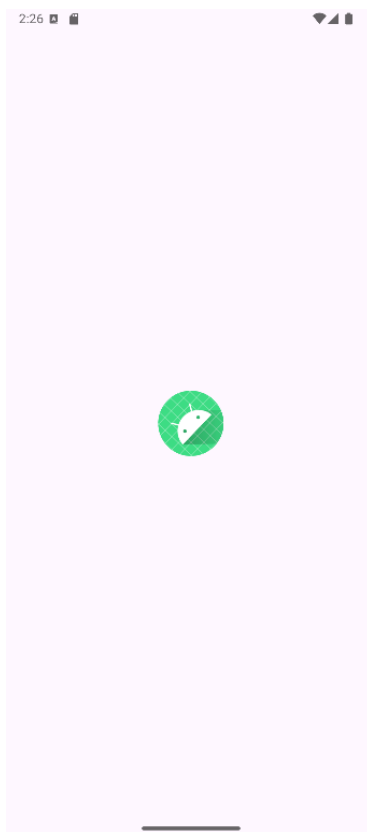
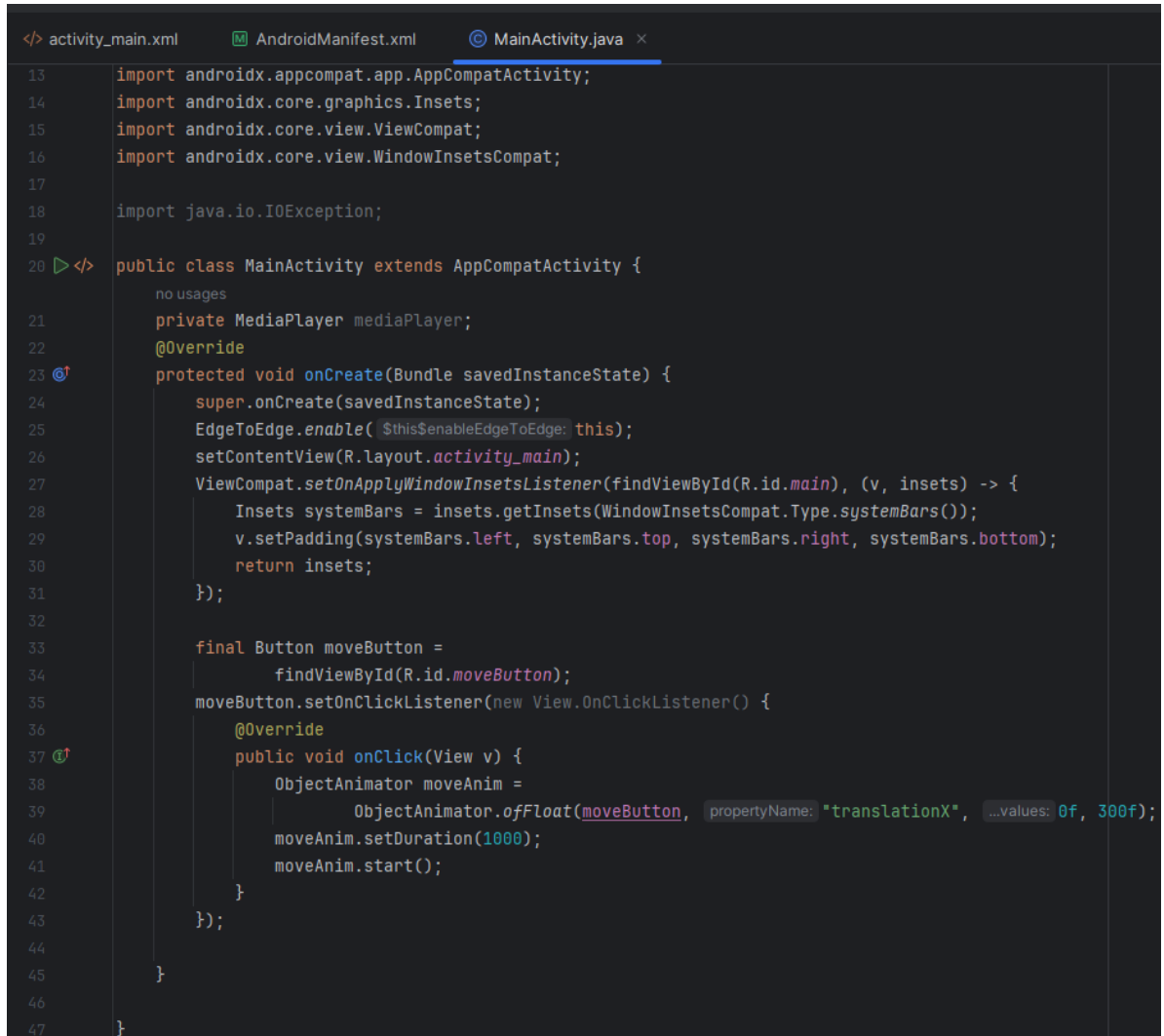


Рисунок 8 – Отображение вращения ImageView, часть 2

1.3.2 Анимация перемещения элемента по экрану

В данном случае также будет использоваться `ObjectAnimator` с методами, описанными в пункте 1.3.1 (Рисунок 9).



```
<? activity_main.xml  AndroidManifest.xml  MainActivity.java x
13  import androidx.appcompat.app.AppCompatActivity;
14  import androidx.core.graphics.Insets;
15  import androidx.core.view.ViewCompat;
16  import androidx.core.view.WindowInsetsCompat;
17
18  import java.io.IOException;
19
20  public class MainActivity extends AppCompatActivity {
21      no usages
22      private MediaPlayer mediaPlayer;
23      @Override
24      protected void onCreate(Bundle savedInstanceState) {
25          super.onCreate(savedInstanceState);
26          EdgeToEdge.enable(this);
27          setContentView(R.layout.activity_main);
28          ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
29              Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
30              v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
31              return insets;
32          });
33
34      final Button moveButton =
35          findViewById(R.id.moveButton);
36      moveButton.setOnClickListener(new View.OnClickListener() {
37          @Override
38          public void onClick(View v) {
39              ObjectAnimator moveAnim =
40                  ObjectAnimator.ofFloat(moveButton, "translationX", 0f, 300f);
41              moveAnim.setDuration(1000);
42              moveAnim.start();
43          }
44      });
45  }
46
47  }
```

Рисунок 9 – Логика анимации перемещения элемента по экрану

При нажатии на кнопку она переместится вправо на заранее прописанное расстояние (Рисунки 10-11).

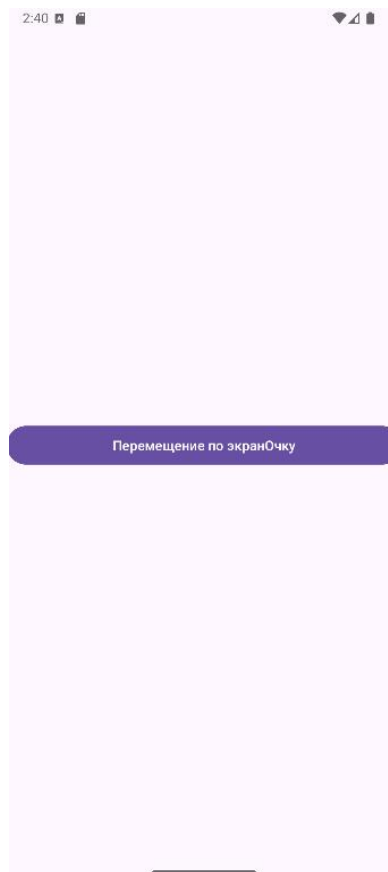


Рисунок 10 – Отображение анимации перемещения элемента, часть 1

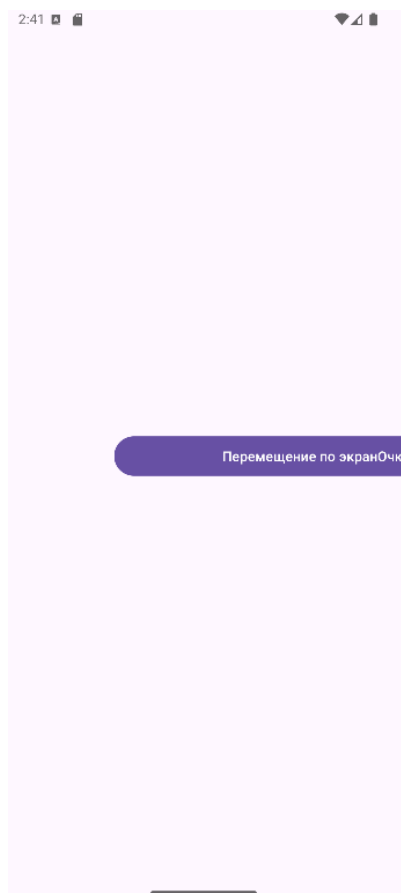


Рисунок 11 – Отображения анимации перемещения элемента, часть 2

1.3.3 Анимация изменения размера элемента

И последняя на рассмотрение анимация – анимация изменения размера элемента (Рисунки 12-14).

```
21 ></> public class MainActivity extends AppCompatActivity {  
22     no usages  
23     private MediaPlayer mediaPlayer;  
24     @Override  
25     protected void onCreate(Bundle savedInstanceState) {  
26         super.onCreate(savedInstanceState);  
27         EdgeToEdge.enable(this);  
28         setContentView(R.layout.activity_main);  
29         ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {  
30             Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());  
31             v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);  
32             return insets;  
33         });  
34  
35         TextView scaleText = findViewById(R.id.scaleTextView);  
36         scaleText.setOnClickListener(new View.OnClickListener() {  
37             @Override  
38             public void onClick(View v) {  
39                 ObjectAnimator scaleX =  
40                     ObjectAnimator.ofFloat(scaleText, "scaleX", 1f, 2f);  
41                 ObjectAnimator scaleY = ObjectAnimator.ofFloat(scaleText, "scaleY", 1f, 2f);  
42                 scaleX.setDuration(1000);  
43                 scaleY.setDuration(1000);  
44                 scaleX.start();  
45                 scaleY.start();  
46             }  
47         });  
48     }  
49 }  
50  
51 }
```

Рисунок 12 – Логика анимации изменения размера элемента

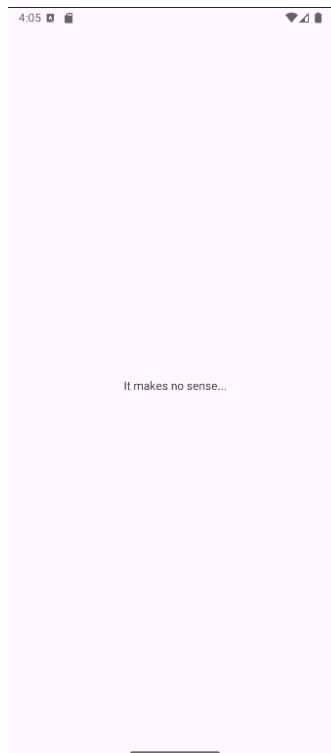


Рисунок 13 – Отображение анимации изменения размера элемента, часть 1



Рисунок 14 – Отображение анимации изменения размера элемента, часть 2

Эти примеры демонстрируют различные способы использования анимации для улучшения визуального восприятия и интерактивности Android-приложений.

Анимацию можно создавать как отдельный ресурс в папке под названием «anim» для хранения ресурсов анимации.

1.4 Уведомления

Уведомления в Android представляют собой сообщения, которые можно отображать вне интерфейса вашего приложения для информирования пользователей о различных событиях, даже когда приложение не используется активно. Это мощный инструмент для повышения взаимодействия и поддержания актуальности приложения в глазах пользователей.

Уведомления в Android управляются через NotificationManager, который является системной службой, ответственной за доставку уведомлений. Для создания уведомления используется Notification.Builder класс, который

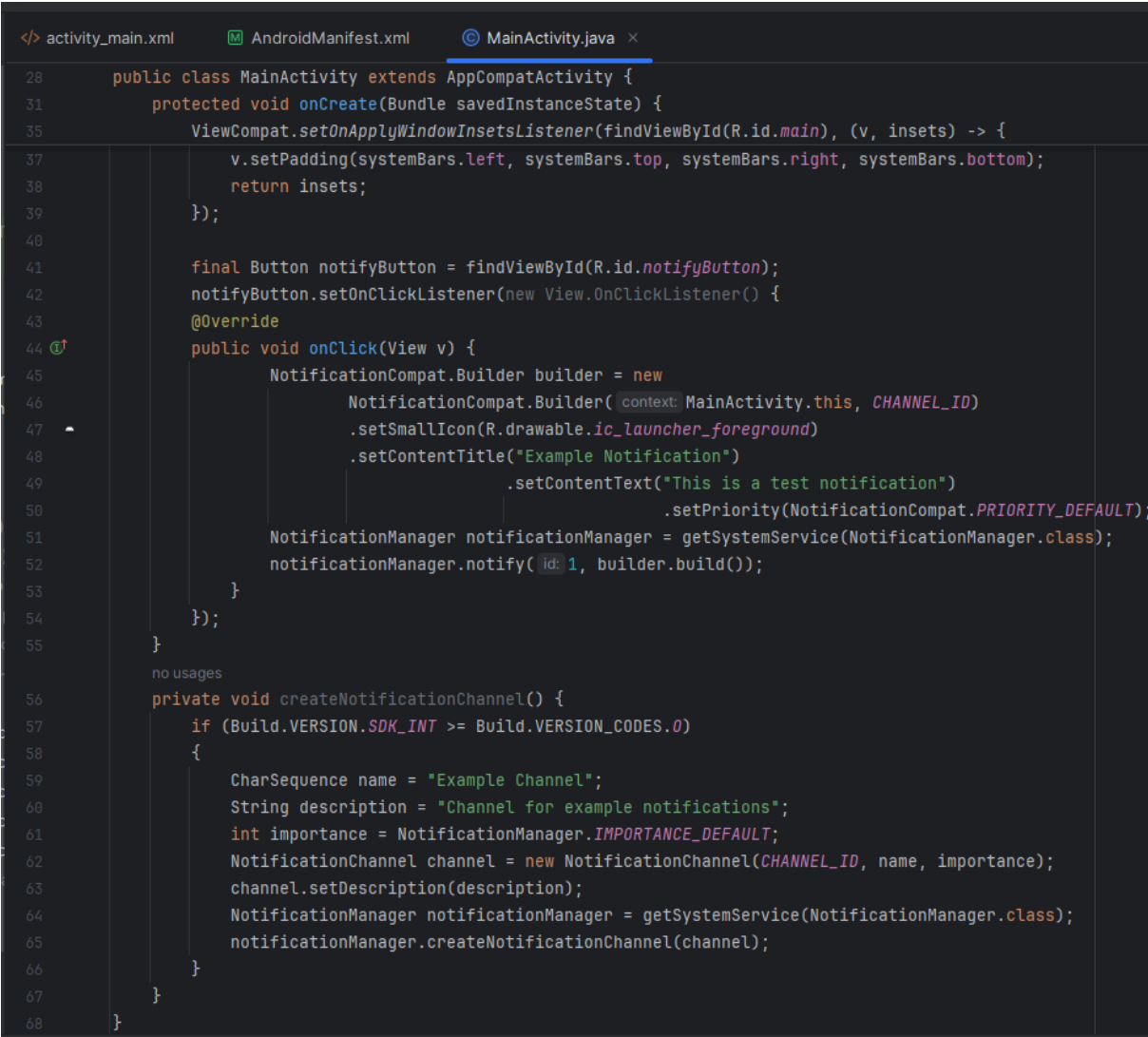
позволяет настроить заголовок, текст, иконку и другие параметры уведомления. С Android Oreo (API уровень 26) введено понятие каналов уведомлений (notification channels), которое требует определения категории уведомлений, что позволяет пользователям управлять настройками уведомлений для различных типов контента.

Для отправки уведомлений на Android версии 13+ необходимо в файле манифеста прописать соответствующее разрешение (Рисунок 15).

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

Рисунок 15 – Разрешение приложения на отправку уведомлений в манифесте

Далее создадим код для показа простых уведомлений (Рисунок 16).



```
28 public class MainActivity extends AppCompatActivity {
31     protected void onCreate(Bundle savedInstanceState) {
35         ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
37             v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
38             return insets;
39         });
40
41         final Button notifyButton = findViewById(R.id.notifyButton);
42         notifyButton.setOnClickListener(new View.OnClickListener() {
43             @Override
44             public void onClick(View v) {
45                 NotificationCompat.Builder builder = new
46                     NotificationCompat.Builder(context, MainActivity.this, CHANNEL_ID)
47                     .setSmallIcon(R.drawable.ic_launcher_foreground)
48                     .setContentTitle("Example Notification")
49                     .setContentText("This is a test notification")
50                     .setPriority(NotificationCompat.PRIORITY_DEFAULT);
51                 NotificationManager notificationManager = getSystemService(NotificationManager.class);
52                 notificationManager.notify(1, builder.build());
53             }
54         });
55     }
56     no usages
57     private void createNotificationChannel() {
58         if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O)
59         {
60             CharSequence name = "Example Channel";
61             String description = "Channel for example notifications";
62             int importance = NotificationManager.IMPORTANCE_DEFAULT;
63             NotificationChannel channel = new NotificationChannel(CHANNEL_ID, name, importance);
64             channel.setDescription(description);
65             NotificationManager notificationManager = getSystemService(NotificationManager.class);
66             notificationManager.createNotificationChannel(channel);
67         }
68     }
}
```

Рисунок 16 – Логика отправки уведомлений пользователю

В этом примере создается канал уведомлений, что необходимо для API 26 и выше. Затем, при нажатии на кнопку, создается уведомление с использованием NotificationCompat.Builder, которое отображает текст и иконку (Рисунок 17). Уведомление отправляется через NotificationManager.

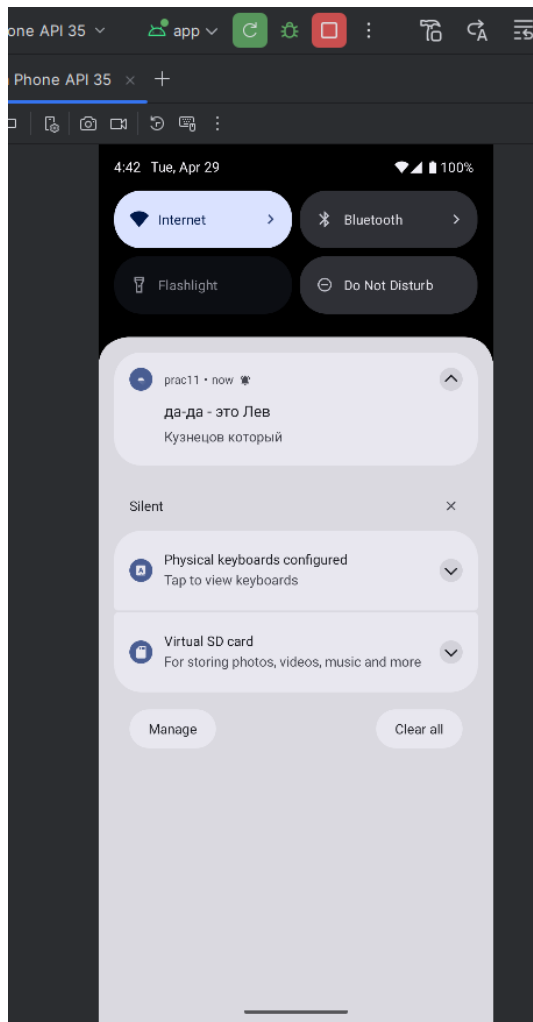


Рисунок 17 – Отображение отправленного уведомления

Используемая переменная CHANNEL_ID должна быть задана глобально в классе private static final String CHANNEL_ID = "example_channel".

Отложенные уведомления в Android позволяют отправить уведомление через определённый период времени, что может быть полезно для напоминаний, задач по расписанию или для других сценариев, где требуется не немедленное, а запланированное уведомление. Для реализации отложенных уведомлений можно использовать AlarmManager, который позволяет запланировать выполнение кода в определённое время (Рисунки 18-19).

```

33
34 <> public class MainActivity extends AppCompatActivity {
    2 usages
35     public static final String CHANNEL_ID = "delayed_channel";
36     @Override
37     protected void onCreate(Bundle savedInstanceState) {
38         super.onCreate(savedInstanceState);
39         EdgeToEdge.enable( $this$enableEdgeToEdge: this);
40         setContentView(R.layout.activity_main);
41         ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
42             Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
43             v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
44             return insets;
45         });
46         final Button delayedNotifyButton = findViewById(R.id.delayedNotifyButton);
47         delayedNotifyButton.setOnClickListener(new View.OnClickListener() {
48             @Override
49             public void onClick(View v) {
50                 scheduleNotification( delay: 2000); // 2 секунда
51             }
52         });
53     }
    no usages
54     private void createNotificationChannel() {
55         if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O)
56         {
57             CharSequence name = "Walter?!";
58             String description = "Channel for delayed example notifications";
59             int importance = NotificationManager.IMPORTANCE_DEFAULT;
60             NotificationChannel channel = new NotificationChannel(CHANNEL_ID, name, importance);
61             channel.setDescription(description);
62             NotificationManager notificationManager = getSystemService(NotificationManager.class);
63             notificationManager.createNotificationChannel(channel);
64         }
65     }

```

Рисунок 18 – Описание логики отправки сообщений с задержкой, часть 1

```

34 public class MainActivity extends AppCompatActivity {
65     }
    1 usage
66     private void scheduleNotification(long delay) {
67         Intent notificationIntent = new Intent( packageContext: this, AlarmReceiver.class);
68         PendingIntent pendingIntent = PendingIntent.getBroadcast( context: this, requestCode: 0, notificationIntent,
69             flags: PendingIntent.FLAG_UPDATE_CURRENT | PendingIntent.FLAG_IMMUTABLE);
70         AlarmManager alarmManager = (AlarmManager) getSystemService(ALARM_SERVICE);
71         long futureInMillis = System.currentTimeMillis() + delay;
72         try{
73             alarmManager.setExact(AlarmManager.RTC_WAKEUP, futureInMillis, pendingIntent);
74         } catch (SecurityException e){
75         }
76     }
77
78
79 }

```

Рисунок 19 – Описание логики отправки сообщений с задержкой, часть 2

Также для корректной работы приложения необходимо описать вспомогательный класс – AlarmReceiver (Рисунок 20).


```

1 package com.example.prac11;
2
3 import android.content.BroadcastReceiver;
4 import android.content.Context;
5 import android.content.Intent;
6 import android.app.NotificationManager;
7 import androidx.core.app.NotificationCompat;
8
9
10 </> public class AlarmReceiever extends BroadcastReceiver {
11     @Override
12     public void onReceive(Context context, Intent intent) {
13         NotificationCompat.Builder builder = new
14             NotificationCompat.Builder(context, MainActivity.CHANNEL_ID)
15             .setSmallIcon(R.drawable.ic_launcher_foreground)
16             .setContentTitle("ННААААЙС! (delayed message)")
17             .setContentText("ФИШКИ РАБОТАЮТ!")
18             .setPriority(NotificationCompat.PRIORITY_DEFAULT);
19         NotificationManager notificationManager = (NotificationManager) context.getSystemService(Context.NOTIFICATION_SERVICE);
20         notificationManager.notify(1, builder.build());
21     }
22 }

```

Рисунок 20 – Описание класса AlarmReceiever

При нажатии на кнопку на экране через 2 секунды появляется оповещение (Рисунок 21).

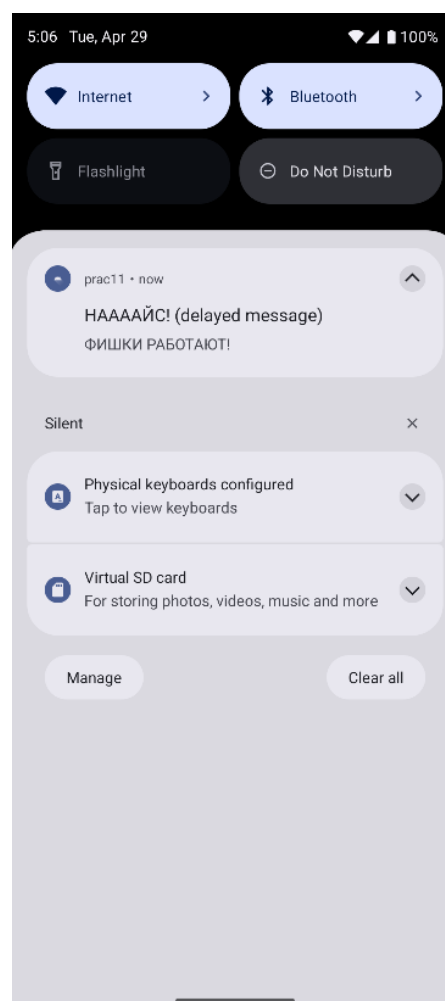


Рисунок 21 – Отображение оповещения с задержкой

2 ПРАКТИЧЕСКОЕ ЗАДАНИЕ

2.1 WebView

Сначала предоставим доступ к интернету в файле манифеста (Рисунок 22).

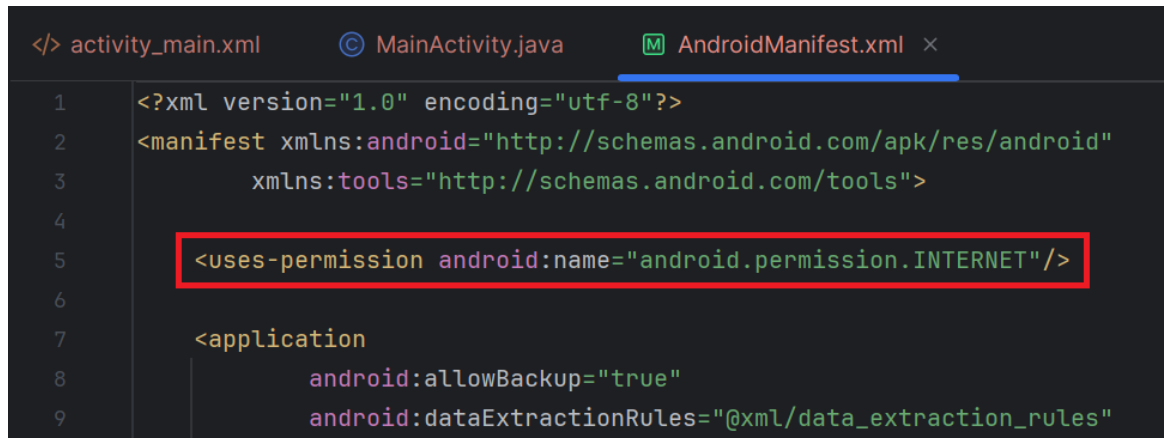


Рисунок 22 – Разрешение доступа в интернет в файле AndroidManifest.xml

Добавим элемент WebView в файл разметки activity_main.xml (Рисунок 23).

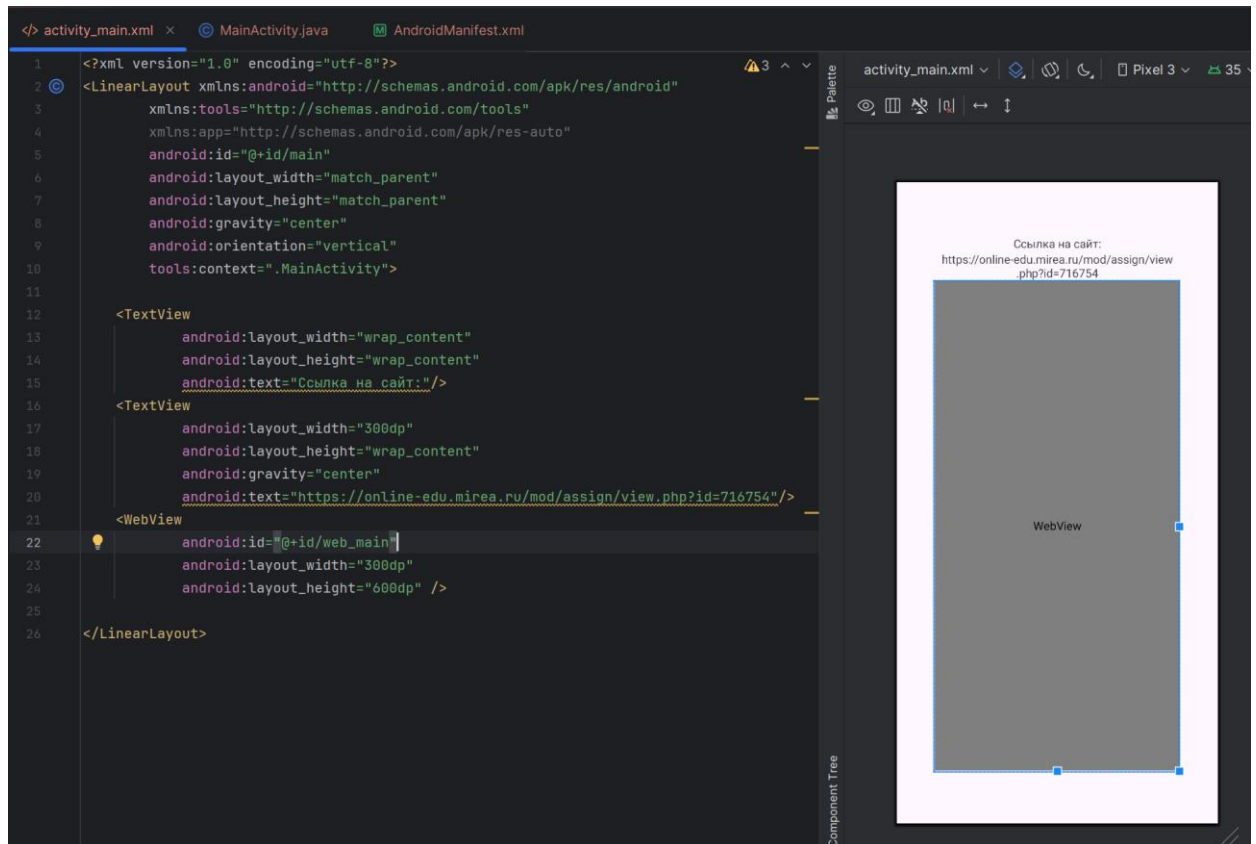


Рисунок 23 – Элемент WebView в файле разметки activity_main.xml

В классе MainActivity остается только предоставить окно отображения и возможность использовать JavaScript, а также ссылку на выбранную страницу для созданного WebView (Рисунок 24).

```
@Override
protected void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    EdgeToEdge.enable(this);
    setContentView(R.layout.activity_main);
    ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) ->
    {
        Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
        v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
        return insets;
    });

    WebView webView = findViewById(R.id.web_main);
    webView.setWebViewClient(new WebViewClient());
    webView.getSettings().setJavaScriptEnabled(true);

    webView.loadUrl("https://online-edu.mirea.ru/mod/assign/view.php?id=716754");
}
```

Рисунок 24 – Описание метода onCreate() класса MainActivity

Протестируем отображение страницы веб-сайта (Рисунок 25).

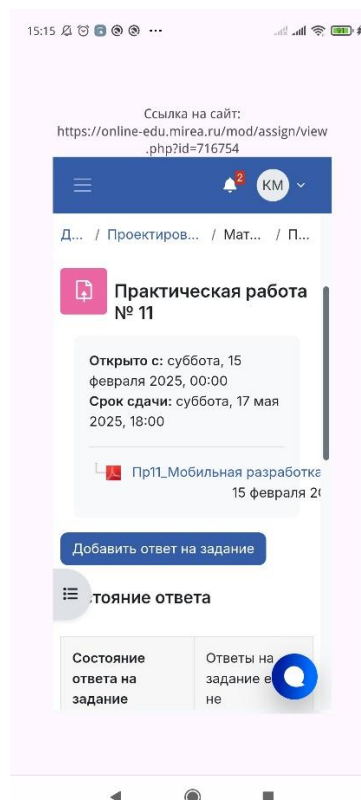


Рисунок 25 – Отображение сайта СДО с помощью WebView

2.2 Воспроизведения аудио из интернета

Создадим новую Activity SecondActivity.

Добавим в файл разметки activity_second.xml кнопку для воспроизведения аудио (Рисунок 26).

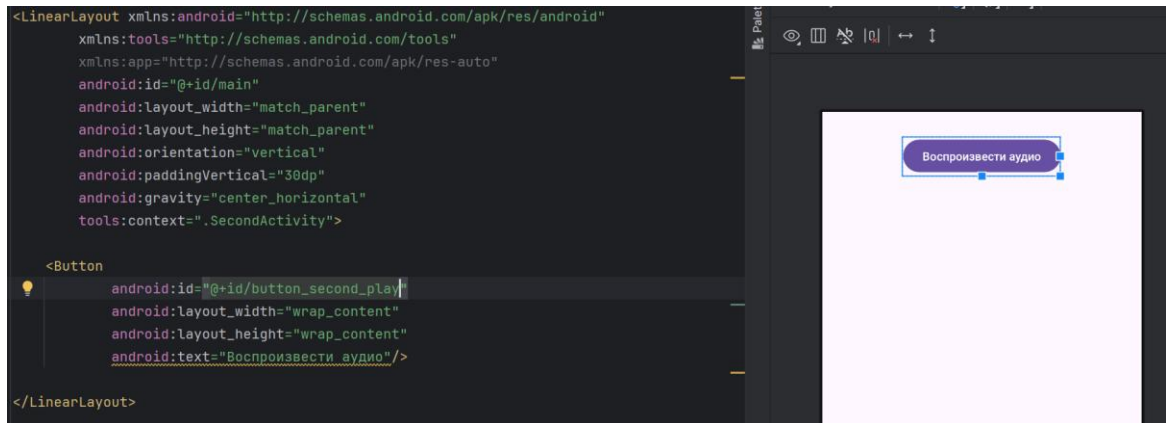


Рисунок 26 – Файл разметки activity_second.xml

В классе SecondActivity добавим реализацию воспроизведения аудио с помощью класса MediaPlayer (Рисунок 27).

```
@Override
protected void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    EdgeToEdge.enable(this);
    setContentView(R.layout.activity_second);
    ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) ->
    {
        Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
        v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
        return insets;
    });
    player = new MediaPlayer();
    try {
        player.setDataSource("https://www.myinstants.com/media/sounds/perduliatsia.mp3");
        player.prepare();
    }
    catch (IOException e) {
        Toast.makeText(context, this, text: "Не удалось получить звук", Toast.LENGTH_SHORT).show();
    }

    Button buttonPlay = findViewById(R.id.button_second_play);
    buttonPlay.setOnClickListener(v -> {
        if (player.isPlaying()) player.pause();
        else player.start();
    });
}

@Override
protected void onDestroy()
{
    super.onDestroy();
    player.release();
}
```

Рисунок 27 – Описание класса SecondActivity

Откроем приложение и нажмем на кнопку (Рисунок 28).



Рисунок 28 – Отображение SecondActivity

При нажатии кнопки проигрывается звук, а при повторном нажатии звук останавливается на паузу или продолжает проигрывание.

2.3 Анимации

Создадим новую Activity ThirdActivity.

В данной работе будут реализованы анимации вращения, растяжения и изменения цвета.

2.3.1 Разметка

Добавим в файл разметки ActivityThird три кнопки для каждого вида анимации (Рисунок 29).

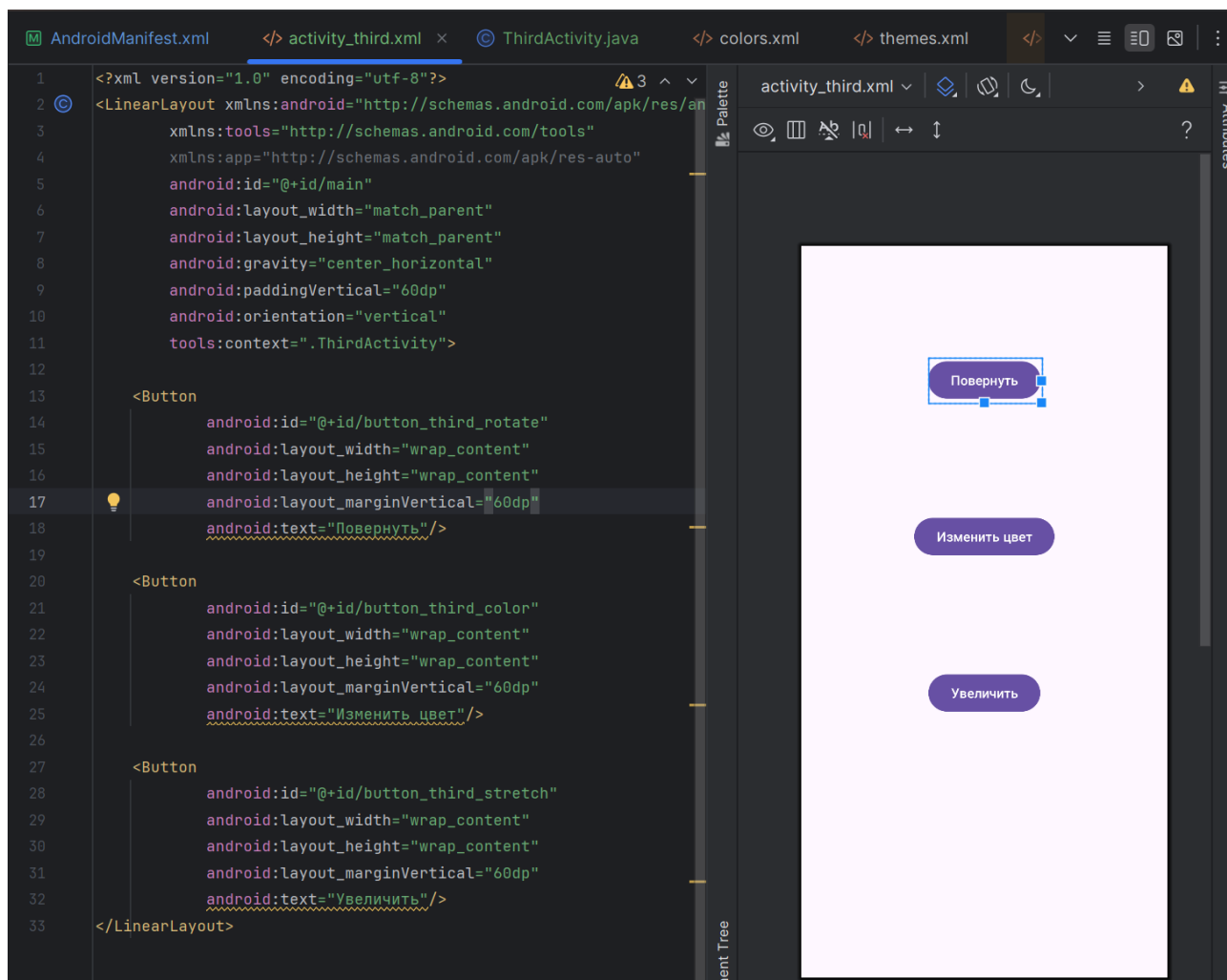


Рисунок 29 – Файл разметки activity_third.xml

2.3.2 Реализация

В методе onCreate() класса ThirdActivity опишем функционал кнопок с помощью класса ObjectAnimator.

Кнопкам была дана продолжительность в 2, 1 и 2 секунды соответственно.

Для кнопки растяжения был использован класс AnimatorSet, чтобы использовать анимации растяжения по горизонтали и по вертикали одновременно.

Для кнопки смены цвета был задан параметр Evaluator как ArgbEvaluator для плавной смены цвета.

Описание метода onCreate() представлено на рисунке 30.

```

@Override
protected void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    EdgeToEdge.enable( $this$enableEdgeToEdge: this);
    setContentView(R.layout.activity_third);
    ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) ->
    {
        Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
        v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
        return insets;
    });

    Button buttonRotate = findViewById(R.id.button_third_rotate);
    buttonRotate.setOnClickListener(v ->
    {
        ObjectAnimator
            .ofFloat(v, propertyName: "rotation", ...values: 0, 120)
            .setDuration(2000)
            .start();
    });
    Button buttonStretch = findViewById(R.id.button_third_stretch);
    buttonStretch.setOnClickListener(v ->
    {
        ObjectAnimator scaleX = ObjectAnimator.ofFloat(v, propertyName: "scaleX", ...values: 1, 2);
        ObjectAnimator scaleY = ObjectAnimator.ofFloat(v, propertyName: "scaleY", ...values: 1, 3);
        AnimatorSet animations = new AnimatorSet();
        animations.playTogether(scaleX, scaleY);
        animations.setDuration(1000).start();
    });
    Button buttonColor = findViewById(R.id.button_third_color);
    buttonColor.setOnClickListener(v ->
    {
        ObjectAnimator colorAnimator = ObjectAnimator
            .ofInt(v, propertyName: "backgroundColor", v.getSolidColor(), Color.GREEN);
        colorAnimator.setEvaluator(new ArgbEvaluator());
        colorAnimator.setDuration(2000).start();
    });
}

```

Рисунок 30 – Описание метода onCreate() класса ThirdActivity

2.3.3 Тестирование

Откроем приложение (Рисунок 31).

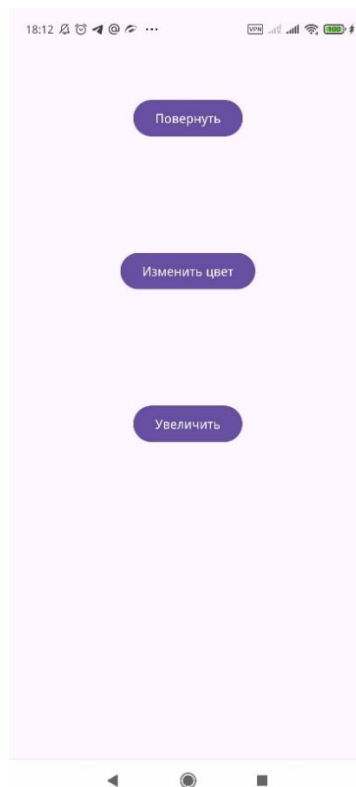


Рисунок 31 – Отображение ThirdActivity при запуске

Нажмем на каждую из кнопок. В результате проигрались соответствующие анимации (Рисунок 32).

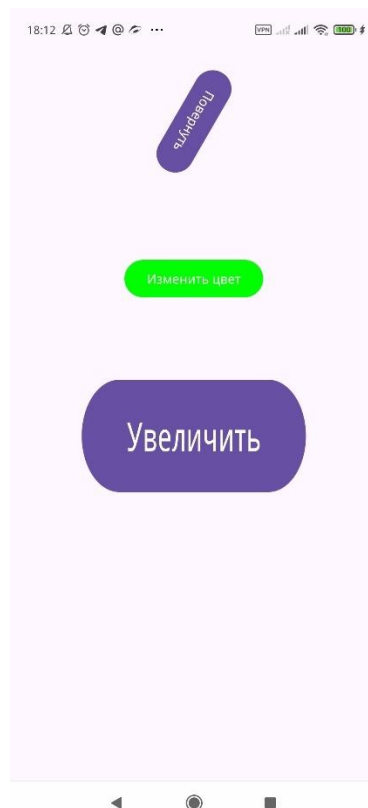


Рисунок 32 – Отображение ThirdActivity после нажатия на каждую из кнопок

2.4 Уведомления

Создадим новую Activity FourthActivity.

2.4.1 Разрешения

Для создания уведомлений нужно предоставить разрешение POST_NOTIFICATIONS, а чтобы сделать это с точной задержкой нужно предоставить разрешение SCHEDULE_EXACT_ALARM (Рисунок 33).

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.POST_NOTIFICATIONS"/>
<uses-permission android:name="android.permission.SCHEDULE_EXACT_ALARM" />
```

Рисунок 33 – Предоставление необходимых разрешений

2.4.2 Разметка

Заполним файл разметки activity_fourth.xml двумя кнопками, для создания моментального уведомления и с задержкой (Рисунок 34).

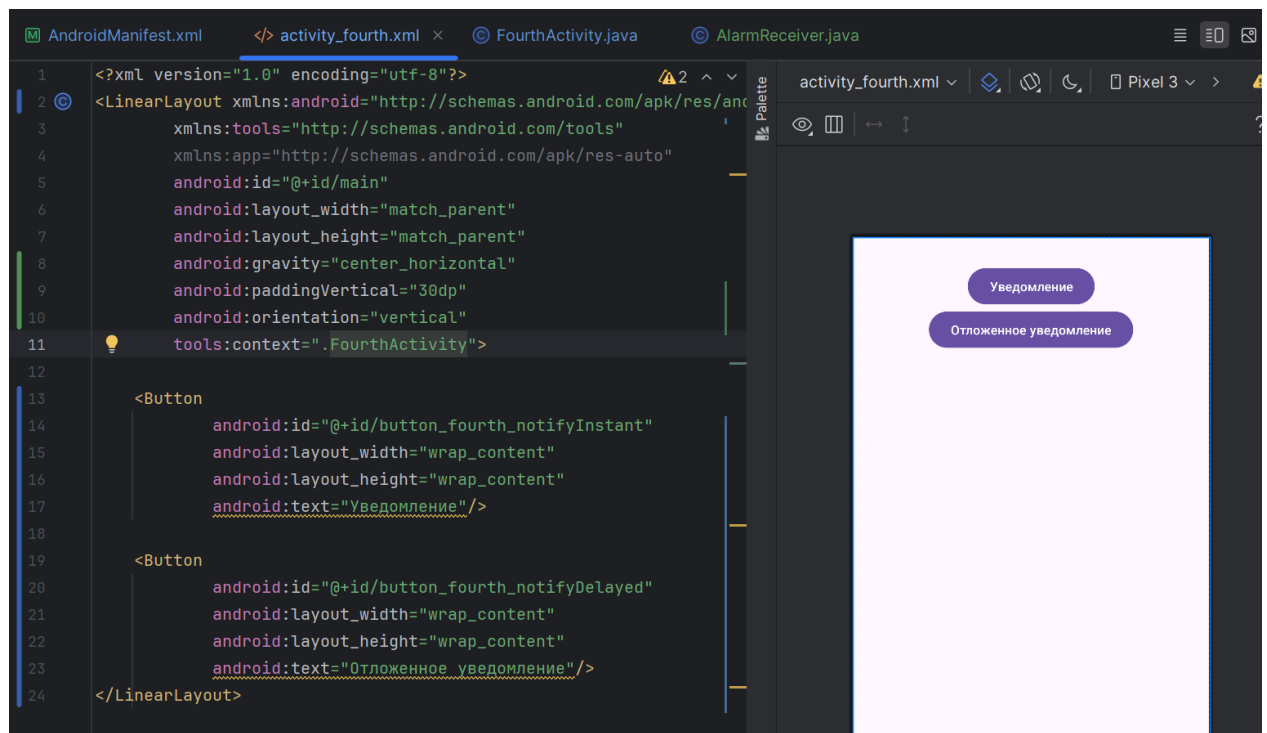


Рисунок 34 – Файл разметки activity_fourth.xml

2.4.3 Реализация

В методе onCreate() класса FourthActivity реализуем создание уведомлений двумя способами.

При нажатии на кнопку «Уведомление» создается обычное уведомление без задержки с применением сервиса NotificationManager. При создании задаются текст и иконка уведомления, после чего вызывается метод notify() класса NotificationManager (Рисунок 35).

```
public class FourthActivity extends AppCompatActivity
{
    public static final String CHANNEL_ID = "egg";

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable( $this$enableEdgeToEdge: this);
        setContentView(R.layout.activity_fourth);
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) ->
        {
            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
            return insets;
        });

        createNotificationChannel();
        Button buttonNotifyInstant = findViewById(R.id.button_fourth_notifyInstant);
        buttonNotifyInstant.setOnClickListener(v ->
        {
            NotificationCompat.Builder builder = new
                NotificationCompat.Builder(v.getContext(), CHANNEL_ID)
                .setSmallIcon(R.drawable.ic_egg)
                .setContentTitle("Моментальное уведомление")
                .setContentText("Это было быстро...");
            getSystemService(NotificationManager.class).notify( id: 1, builder.build());
        });
    }
}
```

Рисунок 35 – Описание метода onCreate() класса FourthActivity, часть 1

Далее в методе с помощью сервиса AlarmManager создается уведомление с задержкой в 1 секунду (Рисунок 36).

```

Button buttonNotifyDelayed = findViewById(R.id.button_fourth_notifyDelayed);
buttonNotifyDelayed.setOnClickListener(v ->
{
    Intent intent = new Intent(v.getContext(), AlarmReceiver.class);
    PendingIntent delayedIntent = PendingIntent.getBroadcast(v.getContext(), requestCode: 0,
        intent, flags: PendingIntent.FLAG_UPDATE_CURRENT | PendingIntent.FLAG_IMMUTABLE);
    try
    {
        AlarmManager alarmManager = (AlarmManager) getSystemService(ALARM_SERVICE);
        alarmManager.setExact(AlarmManager.RTC_WAKEUP,
            triggerAtMillis: System.currentTimeMillis() + 1000, delayedIntent);
    }
    catch (SecurityException e)
    {
        Toast.makeText(context: this, text: "Не удалось показать уведомление",
            Toast.LENGTH_SHORT).show();
    }
});
}

```

Рисунок 36 – Описание метода onCreate() класса FourthActivity, часть 2

Перед созданием уведомлений необходимо также создать канал уведомлений, что происходит в методе createNotificationChannel() класса FourthActivity() (Рисунок 37).

```

private void createNotificationChannel()
{
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O)
    {
        NotificationChannel channel = new NotificationChannel(CHANNEL_ID,
            name: "Яйцо", NotificationManager.IMPORTANCE_HIGH);
        getSystemService(NotificationManager.class).createNotificationChannel(channel);
    }
}

```

Рисунок 37 – Описание метода createNotificationChannel класса FourthActivity

Для создания уведомления с применением сервиса AlarmManager также пришлось создать класс AlarmReceiver, наследующий от BroadcastReceiver, в котором происходит создание уведомления (Рисунок 38).

```

public class AlarmReceiver extends BroadcastReceiver
{
    @Override
    public void onReceive(Context context, Intent intent)
    {
        NotificationCompat.Builder builder = new
            NotificationCompat.Builder(context, FourthActivity.CHANNEL_ID)
            .setSmallIcon(R.drawable.ic_egg)
            .setContentTitle("Уведомление с задержкой")
            .setContentText("Немножечко подождали");
        context.getSystemService(NotificationManager.class).notify(id: 2, builder.build());
    }
}

```

Рисунок 38 – Описание класса AlarmReceiver

После создания данного класса необходимо также указать его в теге `<receiver>` внутри тега `<application>` манифест файла (Рисунок 39).

```
tools:targetApi="31">  
<receiver android:enabled="true" android:name=".AlarmReceiver"/>  
<activity
```

Рисунок 39 – Тег "receiver" в файле AndroidManifest.xml

2.4.4 Тестирование

Сначала откроем приложение (Рисунок 40).



Рисунок 40 – Начальный экран FourthActivity

Нажмем на кнопку «Уведомление», после чего уведомление сразу появляется на экране (Рисунок 41).

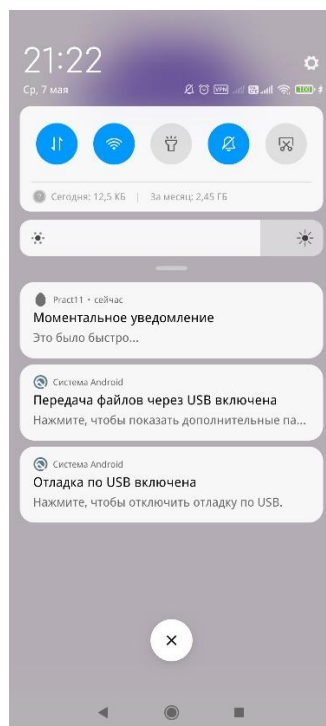


Рисунок 41 – Моментальное уведомление

После этого нажмем кнопку «Отложенное уведомление» и выйдем из приложения. Примерно через секунду приходит второе уведомление (Рисунок 42).



Рисунок 42 – Уведомление с задержкой

Тестирование приложения прошло успешно.

ЗАКЛЮЧЕНИЕ

В ходе работы были получены знания по работе с WebView, мультимедиа, анимациями и уведомлениями в Android Studio. Полученные знания были закреплены путём отображения страницы и воспроизведения аудио из интернета, воспроизведения нескольких анимаций различных элементов интерфейса и показа моментальных и отложенных уведомлений.