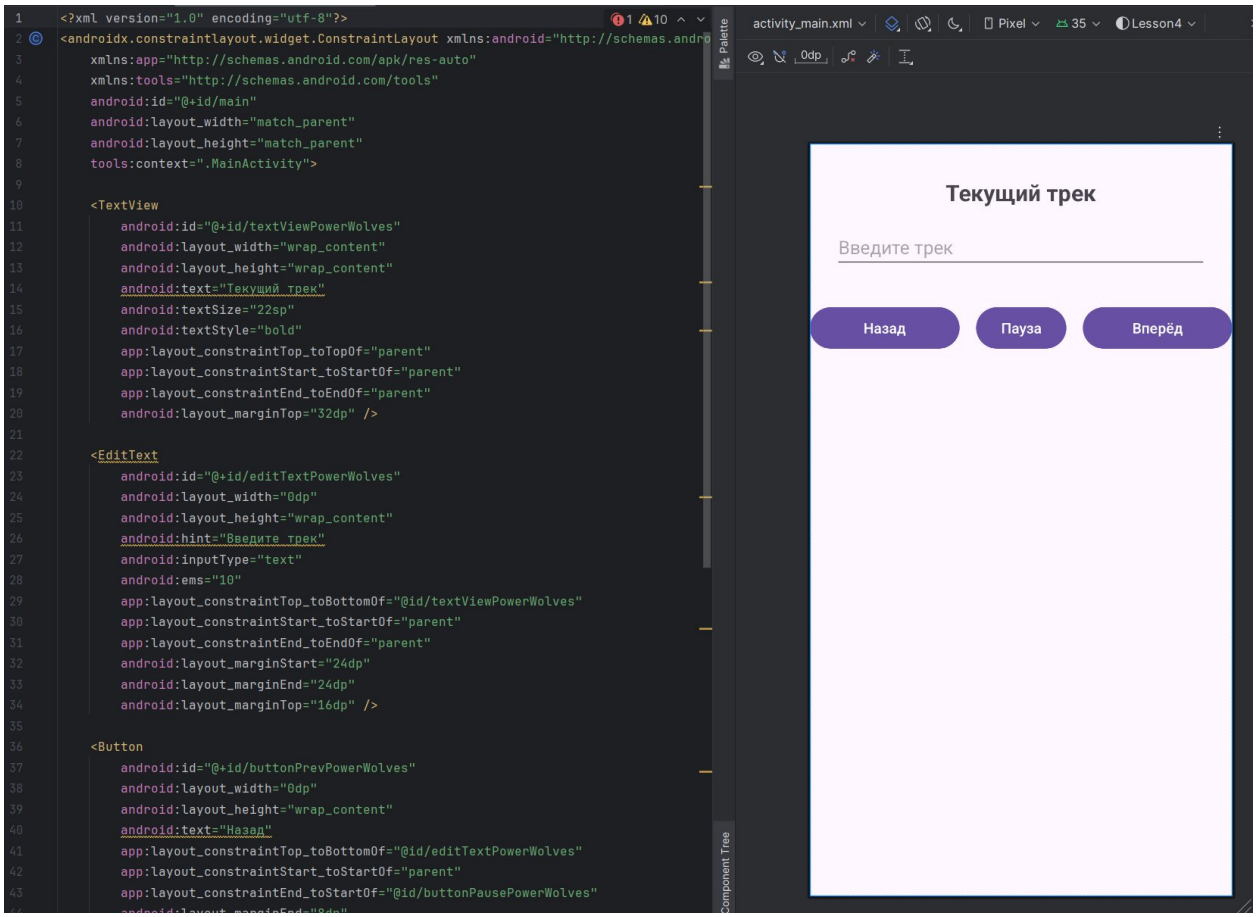
Практическая работа №4

Работа с потоками, сервисами, шифрованием и компонентами Android

Задание 1. Интерфейс музыкального плеера

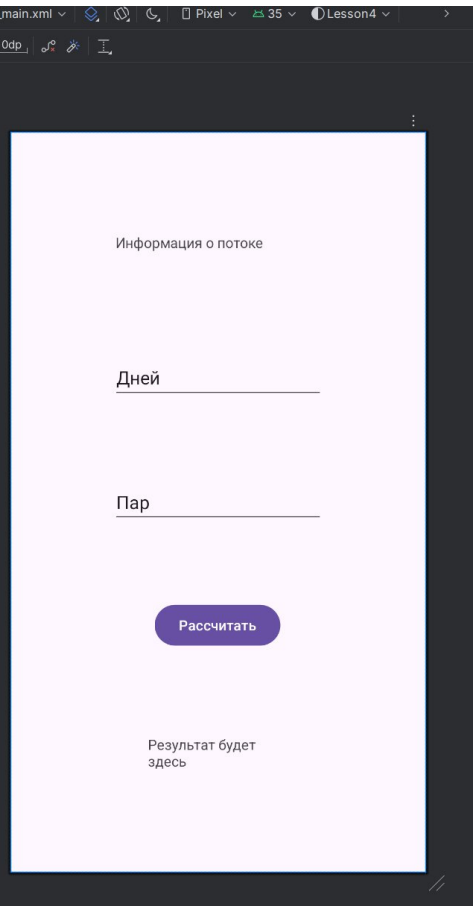
Создан экран музыкального плеера с применением ViewBinding. Макет адаптирован под горизонтальную и вертикальную ориентации. Разметка включает элементы управления: кнопки воспроизведения/паузы и названия текущей композиции. Название песни задаётся программно через binding.textView.text = "...".



Задание 2. Расчёт средней нагрузки в потоке

Создан модуль thread. Интерфейс состоит из трёх полей ввода: количество пар, учебных дней и кнопки запуска. По нажатию запускается фоновый поток, в котором рассчитывается среднее количество пар в день. Результат выводится в TextView.

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

    <TextView
        android:id="@+id/textViewResult"
        android:layout_width="139dp"
        android:layout_height="51dp"
        android:text="Результат будет здесь"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/buttonshow"
        app:layout_constraintVertical_bias="0.497" />

    <Button
        android:id="@+id/buttonshow"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="75dp"
        android:text="Рассчитать"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/editTextPairs" />

    <EditText
        android:id="@+id/editTextPairs"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="78dp"
        android:ems="10"
        android:inputType="text"
        android:text="Пар"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/editTextDays" />

    <EditText
```




```java
public class MainActivity extends AppCompatActivity
    private ActivityMainBinding binding;
    no usages
    private int counter = 0;
    3 usages
    private TextView infoTextView;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        infoTextView = findViewById(R.id.textView2);
        Thread mainThread = Thread.currentThread();

        infoTextView.setText("Имя текущего потока: " + mainThread.getName());
        mainThread.setName("МОЙ НОМЕР ГРУППЫ: 03, НОМЕР ПО СПИСКУ: 23, МОЙ ЛЮБИМЫЙ ФИЛЬМ: Интерстеллар");
        infoTextView.append("\n Новое имя потока: " + mainThread.getName());

        Log.d(MainActivity.class.getSimpleName(), msg: "Стек: " + Arrays.toString(mainThread.getStackTrace()));

        binding.buttonshow.setOnClickListener(new View.OnClickListener()
        {
            @Override
            public void onClick(View v)
            {
                String pairsString = binding.editTextDays.getText().toString();
                String daysString = binding.editTextPairs.getText().toString();
                int Pairs = Integer.parseInt(pairsString);
                int Days = Integer.parseInt(daysString);
                new AveragePairsPerDay().execute(Pairs, Days);
            }
        });

        EdgeToEdge.enable( $this$enableEdgeToEdge: this);
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), ( View v, WindowInsetsCompat insets) ->
        {
            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
            return insets;
        });
    }
```

Задание 3. Последовательность потоков и задержек

Модуль data_thread демонстрирует различия между методами runOnUiThread, post и postDelayed. На экране отображается последовательность вызова заданий:

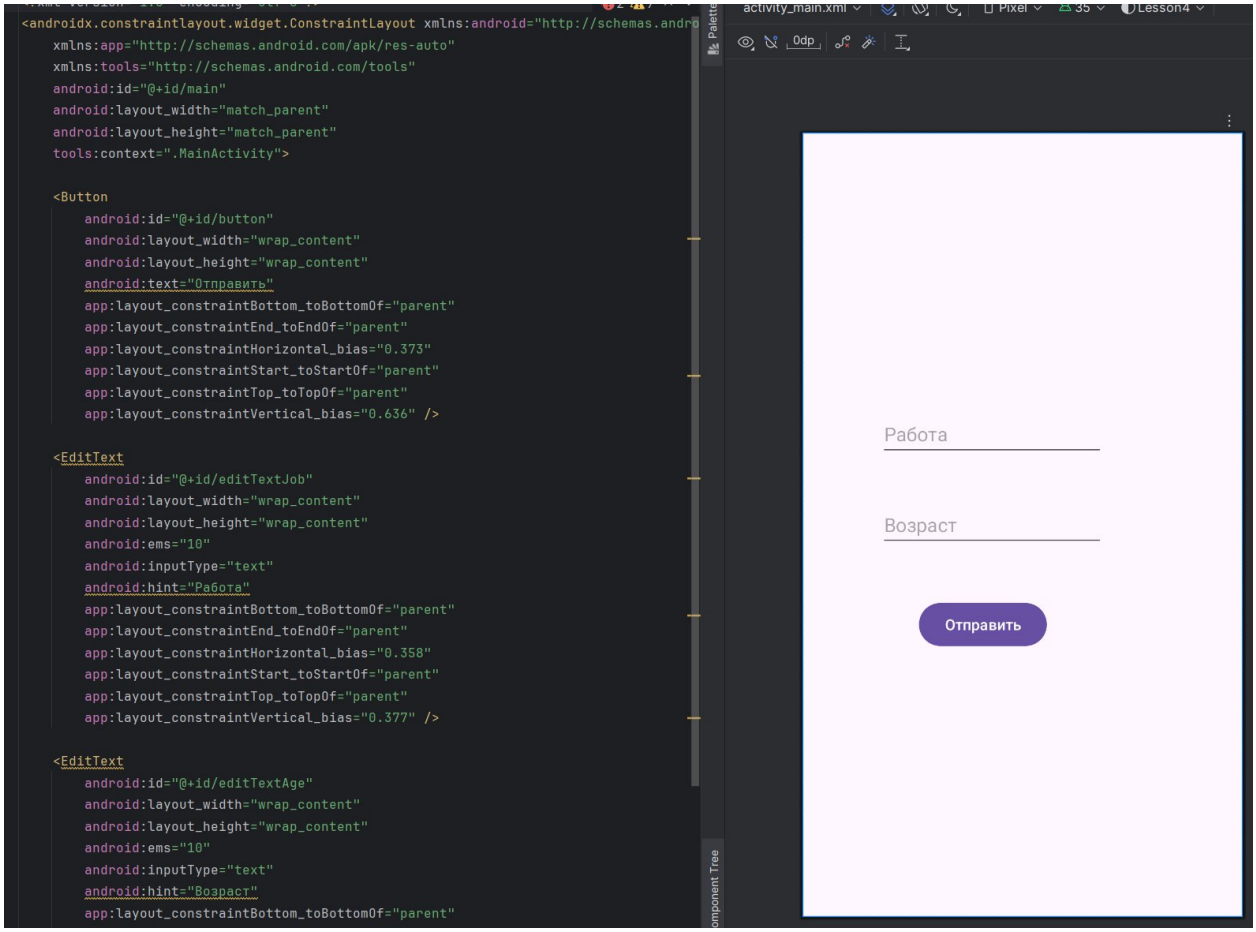
- runn1 после 2 секунд,
- затем runn2 и runn3 с разной задержкой.

Результаты отображаются в TextView, свойства которого настроены на отображение нескольких строк.

```
public class MainActivity extends AppCompatActivity
{
    7 usages
    private ActivityMainBinding binding;
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());
        final Runnable runn1 = new Runnable()
        {
            public void run() { binding.tvInfo.setText("runn1"); }
        };
        final Runnable runn2 = new Runnable()
        {
            public void run() { binding.tvInfo.setText("runn2"); }
        };
        final Runnable runn3 = new Runnable()
        {
            public void run() { binding.tvInfo.setText("runn3"); }
        };
        Thread t = new Thread(new Runnable()
        {
            public void run()
            {
                try
                {
                    TimeUnit.SECONDS.sleep( timeout: 2);
                    runOnUiThread(runn1);
                    TimeUnit.SECONDS.sleep( timeout: 1);
                    binding.tvInfo.postDelayed(runn3,  delayMillis: 2000);
                    binding.tvInfo.post(runn2);
                }
                catch (InterruptedException e)
                {
                    e.printStackTrace();
                }
            }
        });
    }
}
```

Задание 4. Работа с Looper и задержками

Создан модуль looper. Пользователь вводит возраст и профессию, нажимает кнопку. Далее создаётся задержка, соответствующая возрасту (в секундах), после чего данные выводятся в лог через Log.d. Используются классы Handler и Looper.

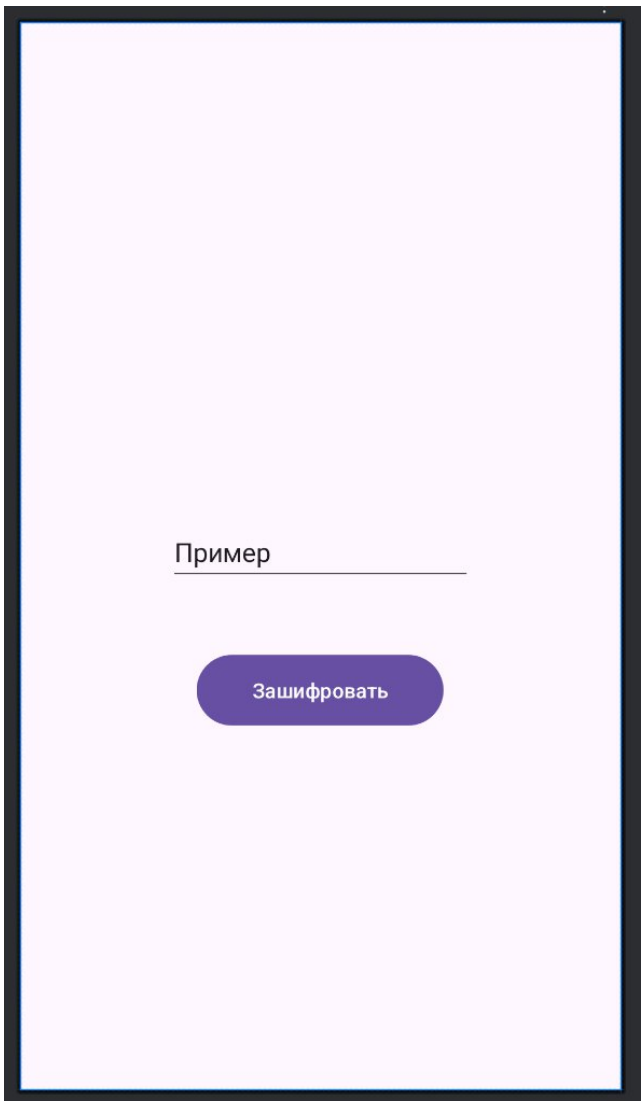



```
Мой возраст 21, работаю в сбере
Task execute. This is result: The number of letters in the word Мой возраст 21, работаю в сбере is 31
app_time_stats: avg=96.94ms min=3.63ms max=499.91ms count=10
```

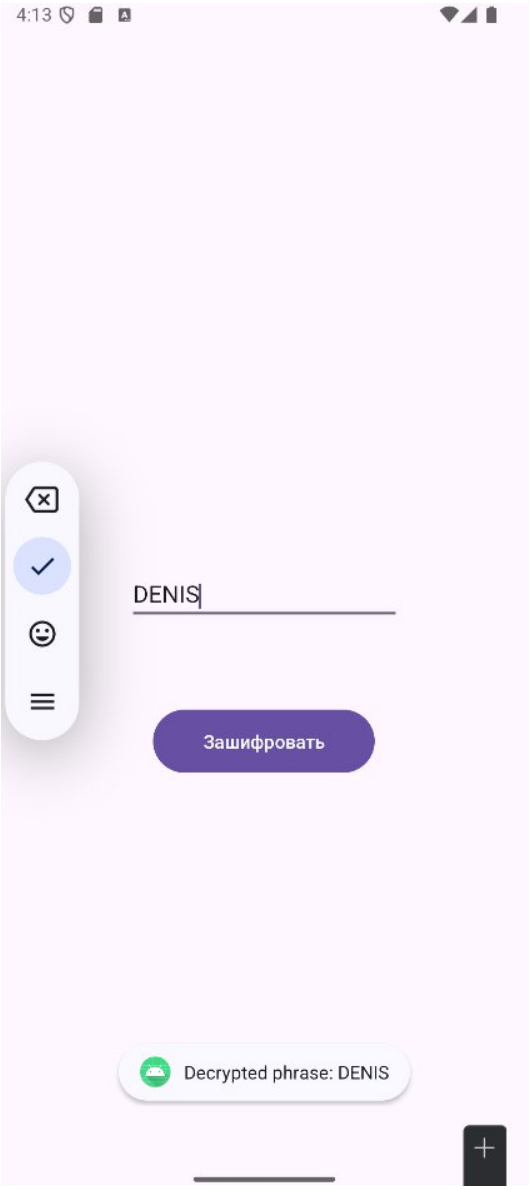
Задание 5–6. Шифрование и дешифровка

Создан модуль CryptoLoader. Добавлены поля ввода и кнопка. Пользователь вводит фразу, которая шифруется с помощью AES, передаётся в Loader, где дешифруется и отправляется обратно. Результат отображается через Toast.

```java
public class MyLoader extends AsyncTaskLoader<String>
    2 usages
    private byte[] encryptedData;
    2 usages
    private byte[] keyBytes;

    2 usages
    public static final String ARG_ENCRYPTED = "encrypted";
    2 usages
    public static final String ARG_KEY = "key";

    1 usage
    public MyLoader(@NonNull Context context, Bundle args)
    {
        super(context);
        if (args != null)
        {
            encryptedData = args.getByteArray(ARG_ENCRYPTED);
            keyBytes = args.getByteArray(ARG_KEY);
        }
    }
    3 usages
    @Override
    protected void onStartLoading()
    {
        super.onStartLoading();
        forceLoad();
    }
    12 usages
    @Override
    public String loadInBackground()
    {
        try
        {
            SecretKey secretKey = new SecretKeySpec(keyBytes,  algorithm: "AES");
            return decryptMsg(encryptedData, secretKey);
        }
        catch (Exception e)
        {
            Log.e( tag: "MyLoader",  msg: "Decryption error", e);
            return null;
        }
    }
```


Пример
Зашифровать

```
1 usage
private SecretKey generateKey() {
    try {
        SecureRandom sr = SecureRandom.getInstance( algorithm: "SHA1PRNG");
        sr.setSeed("any data used as random seed".getBytes());

        KeyGenerator kg = KeyGenerator.getInstance( algorithm: "AES");
        kg.init( keysize: 256, sr);

        return kg.generateKey();
    } catch (NoSuchAlgorithmException e) {
        throw new RuntimeException(e);
    }
}

1 usage
public byte[] encryptMsg(String message, SecretKey secret) {
    try {
        Cipher cipher = Cipher.getInstance( transformation: "AES");
        cipher.init(Cipher.ENCRYPT_MODE, secret);
        return cipher.doFinal(message.getBytes());
    } catch (Exception e) {
        throw new RuntimeException(e);
    }
}
```

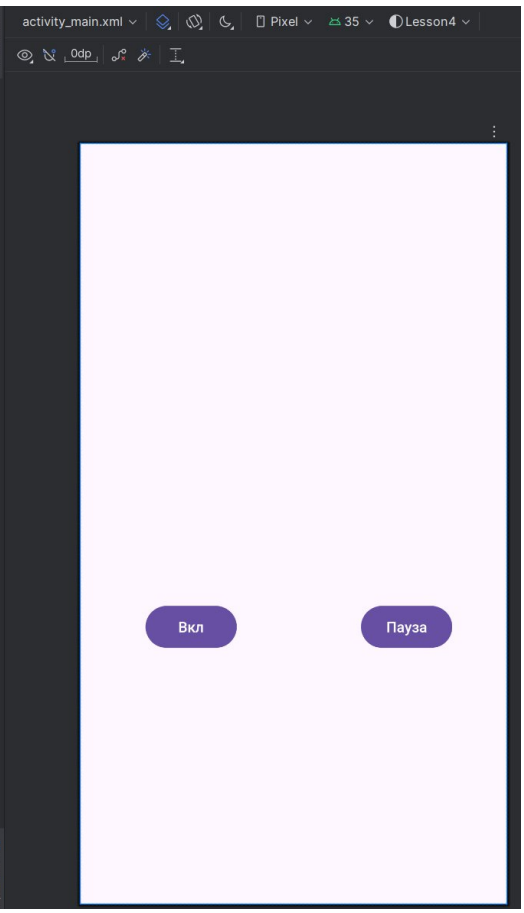


Используется AsyncTaskLoader, где переопределены методы onStartLoading и loadInBackground. Весь процесс шифрования и дешифровки реализован с использованием стандартных алгоритмов и SecretKeySpec.

Задание 7–8. Работа с сервисами и воспроизведение музыки

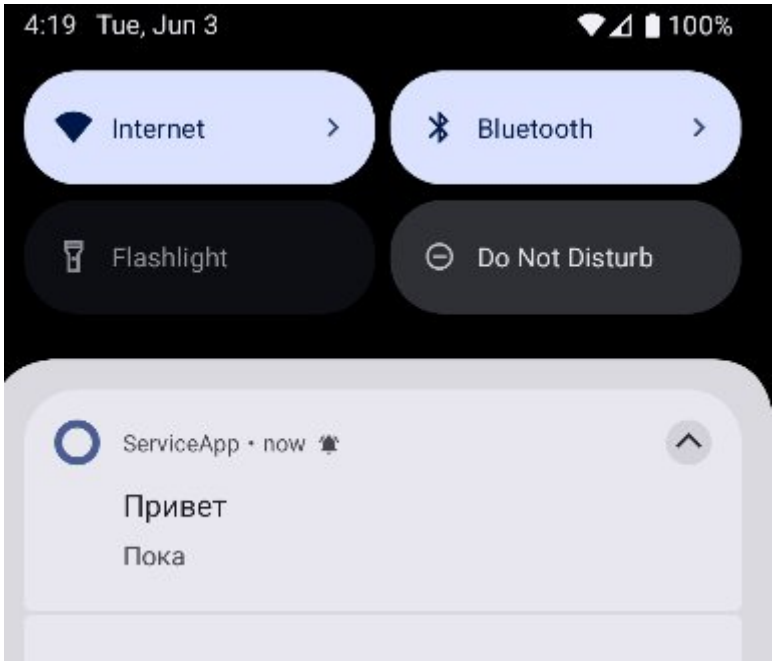
Модуль ServiceApp демонстрирует реализацию простого Service. В AndroidManifest.xml зарегистрирован сервис.

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

    <Button
        android:id="@+id/previous"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Вкл"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.196"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.645" />

    <Button
        android:id="@+id/next"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Пауза"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.838"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.645" />
</androidx.constraintlayout.widget.ConstraintLayout>
```




```java
public class PlayerService extends Service
    public static final String CHANNEL_ID = "ForegroundServiceChannel";

    @Override
    public IBinder onBind(Intent intent)
    {
        throw new UnsupportedOperationException("Привязка не реализована");
    }

    @SuppressLint("ForegroundServiceType")
    @Override
    public void onCreate()
    {
        super.onCreate();

        NotificationChannel channel = new NotificationChannel(
                CHANNEL_ID,
                name: "Уведомления Шамко",
                NotificationManager.IMPORTANCE_DEFAULT
        );
        channel.setDescription("Канал уведомлений Шамко");

        NotificationManagerCompat.from( context: this).createNotificationChannel(channel);

        NotificationCompat.Builder builder = new NotificationCompat.Builder( context: this, CHANNEL_ID)
                .setSmallIcon(R.mipmap.ic_launcher_round)
                .setContentTitle("Привет")
                .setContentText("Воспроизведение...")
                .setPriority(NotificationCompat.PRIORITY_HIGH)
                .setStyle(new NotificationCompat.BigTextStyle().bigText( cs: "Лока"));

        startForeground( id: 1, builder.build());

        mediaPlayer = MediaPlayer.create( context: this, R.raw.est_probitie);
        mediaPlayer.setLooping(false);
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId)
    {
        if (mediaPlayer != null && !mediaPlayer.isPlaying())
        {
            mediaPlayer.start();
```

```java
public class MainActivity extends AppCompatActivity
    private final int PermissionCode = 200;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        checkPermissions();
        setupButtonListeners();
    }

    1 usage
    private void checkPermissions()
    {
        if (ContextCompat.checkSelfPermission( context: this, POST_NOTIFICATIONS) == PackageManager.PERMISSION_GRANTED)
        {
            Log.d(MainActivity.class.getSimpleName(), msg: "Доступ разрешён");
        }
        else
        {
            Log.d(MainActivity.class.getSimpleName(), msg: "Доступ запрещён!");
            ActivityCompat.requestPermissions(
                    activity: this,
                    new String[]{POST_NOTIFICATIONS, FOREGROUND_SERVICE},
                    PermissionCode
            );
        }
    }

    1 usage
    private void setupButtonListeners()
    {
        binding.previous.setOnClickListener( View v -> {
            Intent serviceIntent = new Intent( packageContext: MainActivity.this, PlayerService.class);
            ContextCompat.startForegroundService( context: MainActivity.this, serviceIntent);
        });

        binding.next.setOnClickListener( View v -> {
            stopService(new Intent( packageContext: MainActivity.this, PlayerService.class));
        });
    }
}
```



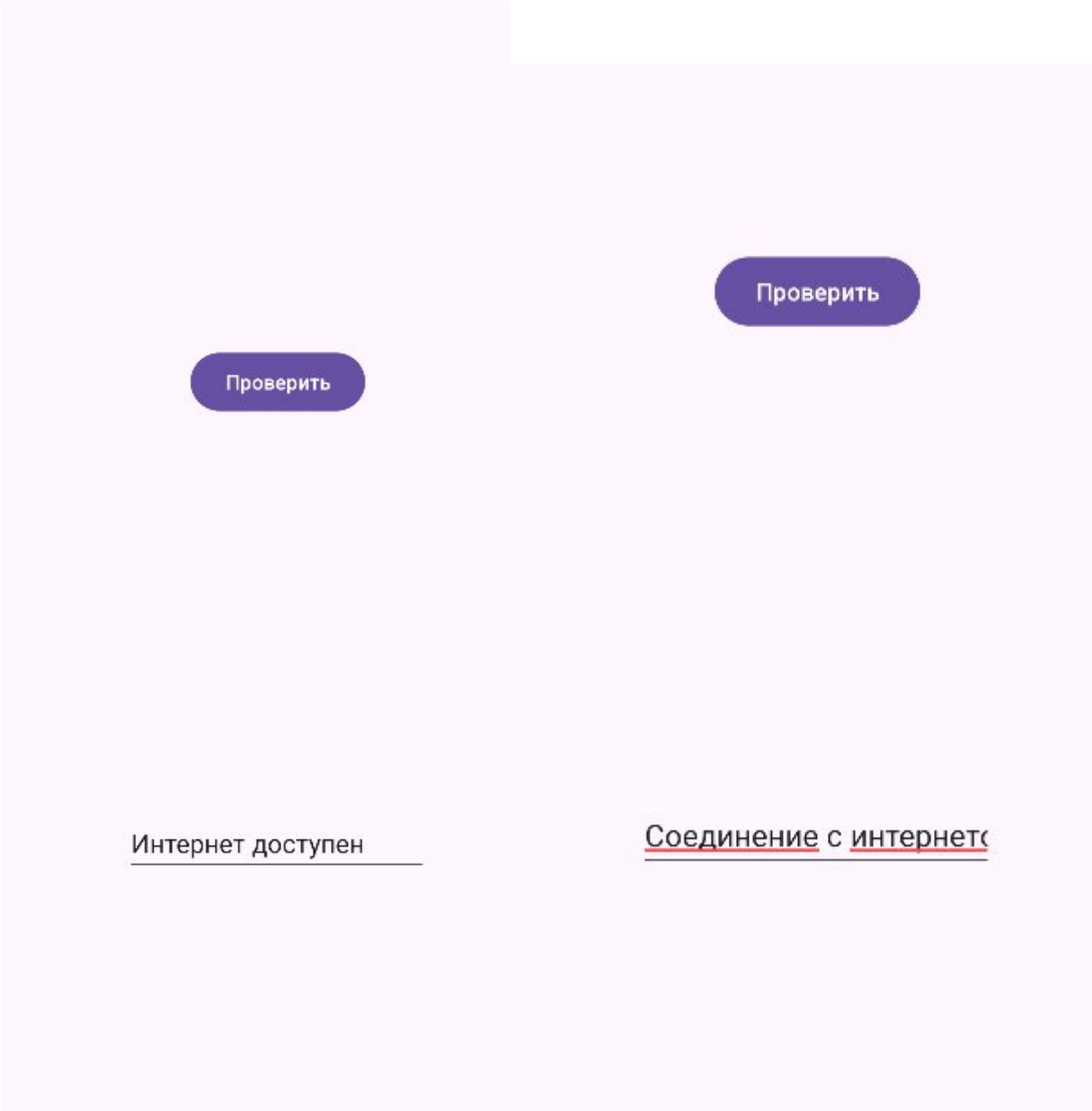
В модуле реализован проигрыватель MP3-файлов. Добавлен аудиофайл в ресурсы. Интерфейс содержит две кнопки: запуск и остановка

воспроизведения. Используется MediaPlayer, логика прописана вручную. Дизайн проигрывателя оформлен в индивидуальном стиле.

Задание 9. Использование WorkManager

Создан модуль WorkManager. Реализован фоновой Worker с условием запуска при наличии интернета. В MainActivity инициализируется задача, проверка доступа к сети происходит через Constraints.

```
@Override
protected void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    editText = findViewById(R.id.editTextText);
    button = findViewById(R.id.button);

    button.setOnClickListener(new View.OnClickListener()
    {
        @Override
        public void onClick(View v)
        {
            if (isNetworkAvailable())
            {
                editText.setText("Интернет доступен");
                startUploadWork();
            }
            else
            {
                editText.setText("Соединение с интернетом отсутствует");
            }
        }
    });
}

1 usage
private void startUploadWork()
{
```



Контрольное задание

В проекте MireaProject добавлен фрагмент, реализующий выполнение фоновой задачи. Применён Worker для запуска задачи. Условие выполнения — наличие подключения к интернету. Задание выполняется в фоне и отображает результат в пользовательском интерфейсе.