МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт кибербезопасности и цифровых технологий

Кафедра КБ-14 «Цифровые технологии обработки данных»

ОТЧЕТ
по практической работе №4

Выполнил Бурмистров И.Г.
фамилия, имя, отчество

шифр 22Б0616 группа БСБО-07-22

Проверил _____ Изергин Д.А.
ученая степень, должность *фамилия, имя, отчество*

Москва 2025г.

В ходе данной работы были созданы модули «cryptoloader», «data_thread», «looper», «MusicPlayer», «serviceapp», «thread» и «workmanager» (см. Рисунок 1).

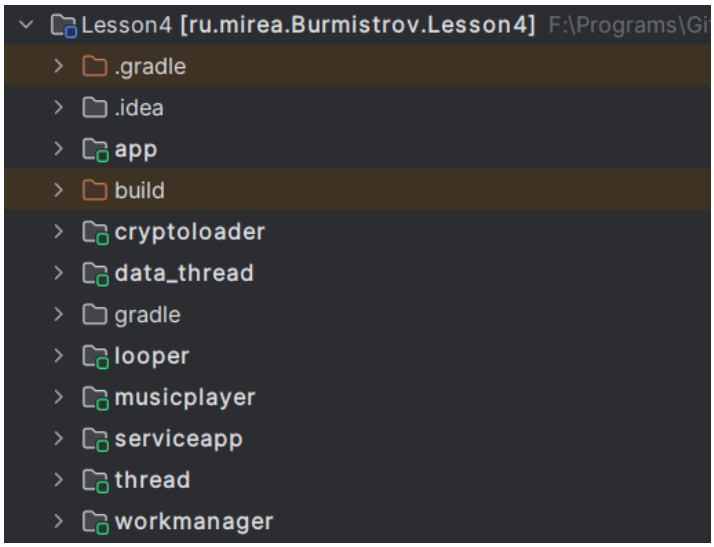

Рисунок 1. Модули проекта

В первом модуле «MusicPlayer» была создана активность, в которой с помощью изображений и текста и других компонентов было создано окно музыкального плеера и добавлена горизонтальная разметка (см. Рисунок 2).

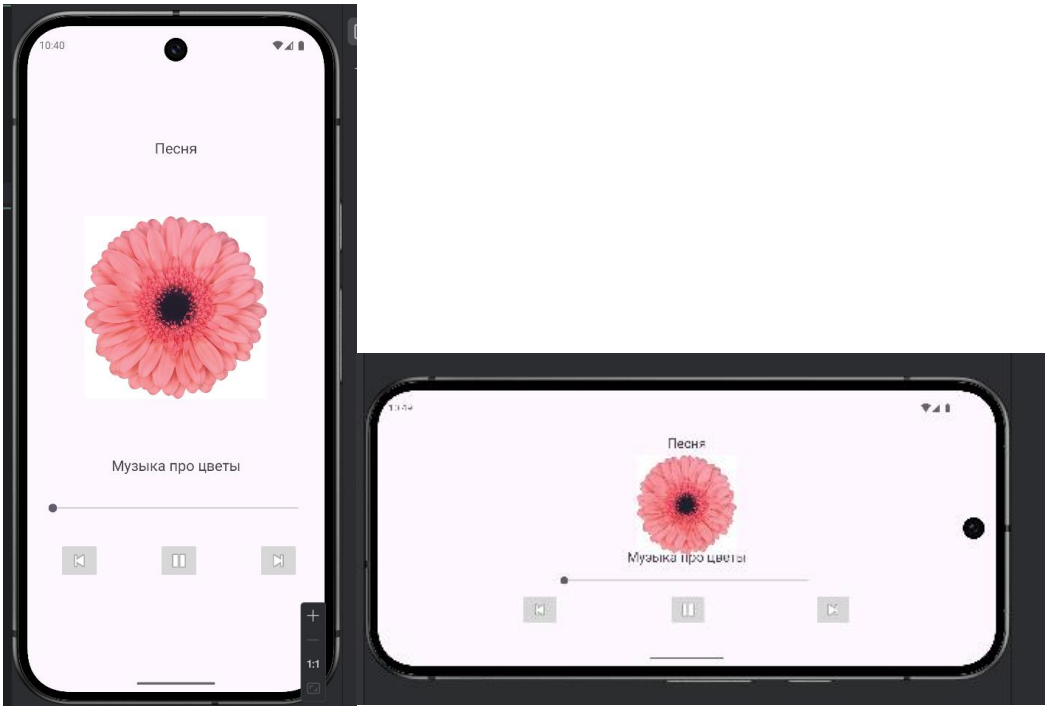

Рисунок 2. Пример активности

Далее в эту активность была добавлена адресация элементов ч помощью «binding» (см. Рисунок 3).

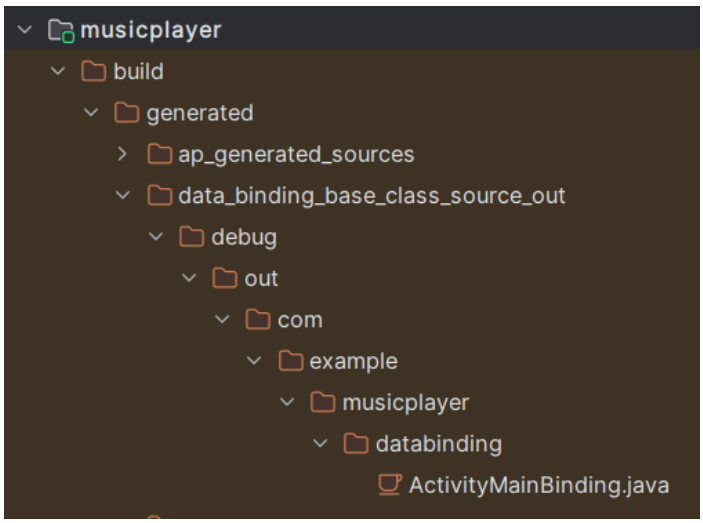

Рисунок 3. Сгенерированный файл Binding

Далее был создан второй модуль «thread». В нём при инициализации заменяется имя главного потока. Также реализован функционал подсчёта в фоновом потоке среднее количество пар в день за период одного месяца (см. Рисунок 4 и Листинг 1).

Рисунок 4. Пример работы приложения

```
public class MainActivity extends AppCompatActivity {
    private ActivityMainBinding binding;
    private Handler handler;
```

```
    private int counter;

    private int allLessons;

    private int allLessondays;
    private double result;

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        binding = ActivityMainBinding.inflate(getLayoutInflater());

        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;
        });
        setContentView(binding.getRoot());

        Thread mainthread = Thread.currentThread();
        binding.textView.setText("Имя текущего потока: " +
mainthread.getName());

        mainthread.setName("Мой номер группы: БСБО-07-22, мой номер по
списку: 4, мой любимый фильм: Мстители");
        binding.textView.append("\n Новое имя потока: " +
mainthread.getName());
        Log.d("Поток","Stack: " +
Arrays.toString(mainthread.getStackTrace()));

        binding.button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                new Thread(new Runnable() {
                    @Override
                    public void run() {

                        int numberthread = counter++;
                        Log.d("Поток",String.format("Запущен поток № %d
студентом группы № %s номер по списку № %d", numberthread, "БСБО-07-22", 4));
                        long endTime = System.currentTimeMillis() + 2 * 1000;
                        while(System.currentTimeMillis() < endTime){
                            synchronized (this){
                                try{
                                    allLessons =
Integer.valueOf(String.valueOf(binding.editTextPairs.getText()));
                                    allLessondays =
Integer.valueOf(String.valueOf(binding.editTextDays.getText()));
                                    wait(endTime -
System.currentTimeMillis());

                                    Log.d(MainActivity.class.getSimpleName(),
"Endtime: "+ endTime);
                                    result = (double) allLessons
/allLessondays;
                                    Log.d("Поток", ""+result);
                                    runOnUiThread(new Runnable() {
```

```
                                        @Override
                                        public void run() {

binding.textView2.setText("среднее кол-во пар за месяц: "+ result);
                                        }
                                    });

                                }catch (Exception e){
                                    throw new RuntimeException(e);
                                }
                            }
                        }
                    }

                }).start();

            }
        });

    }
}
```

Листинг 1. Запуск фоновых процессов

В модуле «data_thread» было определено в какой последовательности происходит запуск процессов. Спустя 2 секунды запускается 1-ый процесс, меняющий текст на Run1, затем спустя 1 секунду запускается 2-ой процесс, после чего 3-й, так как при помощи метода «postDelayed» процесс «Run3» был поставлен в очередь с задержкой в 2 секунды (см. Рисунок 5 и Листинг 2).

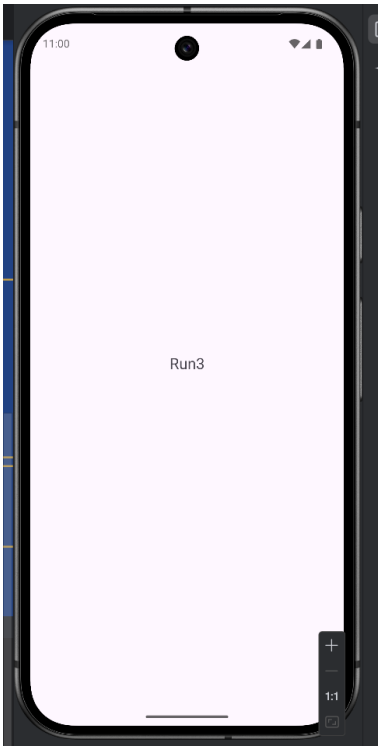

Рисунок 5. Пример работы

```
public class MainActivity extends AppCompatActivity {
    private ActivityMainBinding binding;
```

```
    @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;
        });

        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        final Runnable runn1 = new Runnable() {
            @Override
            public void run() {
                binding.TextView.setText("Run1");
            }
        };

        final Runnable runn2 = new Runnable() {
            @Override
            public void run() {
                binding.TextView.setText("Run2");
            }
        };

        final Runnable runn3 = new Runnable() {
            @Override
            public void run() {
                binding.TextView.setText("Run3");
            }
        };

        Thread t = new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    TimeUnit.SECONDS.sleep(2);
                    runOnUiThread(runn1);
                    TimeUnit.SECONDS.sleep(1);
                    binding.TextView.postDelayed(runn3, 2000);
                    binding.TextView.post(runn2);
                }catch (InterruptedException e){
                    e.printStackTrace();
                }
            }
        });
        t.start();
    }
}
```

Листинг 2. Код класса модуля «data_thread»

Далее был создан модуль «looper», в котором вводится возраст и работа. После нажатия на кнопку происходит задержка, равная возрасту, после которой выводится в логах сообщение с возрастом и работой (см. Рисунок 6 и Листинг 3).

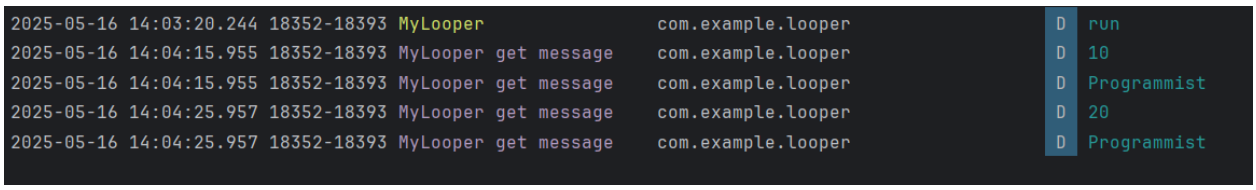

Рисунок 6. Пример вывода данных

```
public MyLooper(Handler mainThreadHandler){
    mainHandler = mainThreadHandler;
}
public void run(){
    Log.d("MyLooper","run");
    Looper.prepare();
    mHandler = new Handler(Looper.myLooper()){
        public void handleMessage(Message msg){
            Integer age = msg.getData().getInt("AGE");
            Log.d("MyLooper get message", String.valueOf(age));
            String Job = msg.getData().getString("JOB");
            Log.d("MyLooper get message", Job);

            Message message = new Message();
            Bundle bundle = new Bundle();
            try{
                Thread.sleep(age * 1000);
            } catch (InterruptedException e){
                e.printStackTrace();
            }

            bundle.putString("result",String.format("Возраст: %d ||| Работа:
%s", age, Job));
            message.setData(bundle);

            mainHandler.sendMessage(message);
        }
    };
    Looper.loop();

}
```

Листинг 3. Класс «MyLooper»

В следующем модуле «cryptoloader» было реализовано шифрование и дешифрование сообщения в фоне при помощи «loader». При нажатии на кнопку создаётся «loader», в который передаётся ключ и зашифрованное сообщение. Затем он дешифрует сообщение и появляется сообщение (см. Рисунок 7 и Листинг 4 и 5).

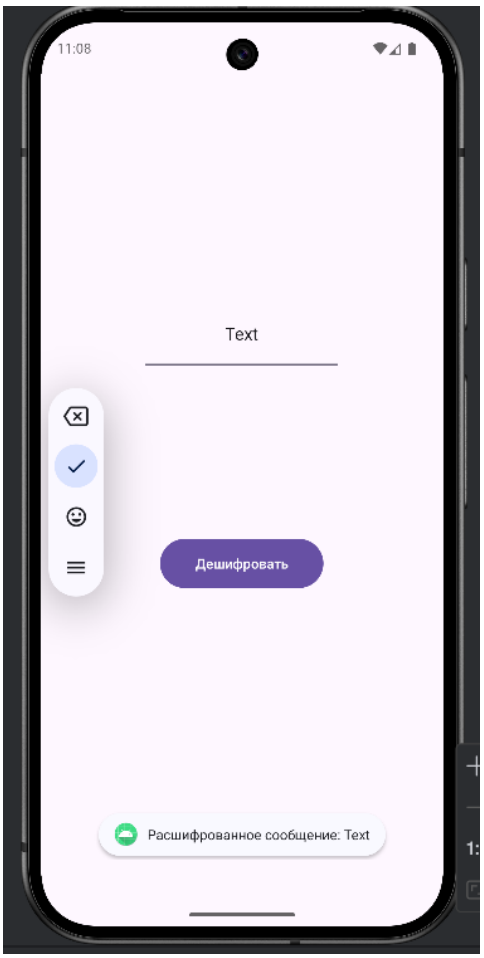

Рисунок 7. Пример работы приложения

```
public class MainActivity extends AppCompatActivity implements
LoaderManager.LoaderCallbacks<String> {
    private final int LoaderID = 1234;
    private ActivityMainBinding binding;
    @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;
        });
        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());
    }

    public void onClick(View view){
        Bundle bundle = new Bundle();
        SecretKey key = generateKey();
        bundle.putByteArray(MyLoader.ARG_WORD,
encryptMsg(binding.editTextText.getText().toString(), key));
        bundle.putByteArray("Key", key.getEncoded());
        LoaderManager.getInstance(this).restartLoader(LoaderID, bundle,
this);
```

```
    }


    @Override
    public void onLoaderReset(@NonNull Loader<String> loader){
        Log.d("Loader", "OnLoaderReset");
    }

    @NonNull
    @Override
    public Loader<String> onCreateLoader(int id, @Nullable Bundle bundle){
        if(id== LoaderID){
            Toast.makeText(this, "onCreateLoader: " + id,
Toast.LENGTH_SHORT).show();
            return new MyLoader(this, bundle);
        }
        throw new InvalidParameterException("Invalid loader id");

    }

    @Override
    public void onLoadFinished(@NonNull Loader<String> loader, String s) {
        if(loader.getId()==LoaderID){
            Log.d("Loader", "On loaderFinished: " + s);
            Toast.makeText(this, "Расшифрованное сообщение: " + s,
Toast.LENGTH_SHORT).show();
        }

    }

    public static SecretKey generateKey() {
        try{
            SecureRandom random = SecureRandom.getInstance("SHA1PRNG");
            random.setSeed("Metal Gear Solid".getBytes());
            KeyGenerator keyGenerator = KeyGenerator.getInstance("AES");
            keyGenerator.init(256, random);
            return keyGenerator.generateKey();

        } catch (NoSuchAlgorithmException e) {
            throw new RuntimeException(e);
        }
    }
    public static byte[] encryptMsg(String message, SecretKey key) {
        Cipher cipher = null;
        try {
            cipher = Cipher.getInstance("AES");
            cipher.init(Cipher.ENCRYPT_MODE, key);
            return cipher.doFinal(message.getBytes());
        } catch (NoSuchAlgorithmException | NoSuchPaddingException |
InvalidKeyException |
                 BadPaddingException | IllegalBlockSizeException e) {
            throw new RuntimeException(e);
        }
    }
}
```

Листинг 4. Класс активности

```
public class MyLoader extends AsyncTaskLoader<String> {
    private String firstName;
    public static final String ARG_WORD = "word";
    public static final String ARG_CIPHER = "cipher";
    public static final String ARG_KEY = "Key";

    private Bundle args;


    public MyLoader(Context context, Bundle args) {
        super(context);
        if(args != null)
            this.firstName = args.getString(ARG_WORD);
        this.args = args;
    }
    @Override
    protected void onStartLoading() {
        super.onStartLoading();
        forceLoad();
    }
    @Override
    public String loadInBackground() {

        byte[] cryptText = args.getByteArray(ARG_WORD);
        byte[] key = args.getByteArray(ARG_KEY);
        SecretKey OriginKey = new SecretKeySpec(key, 0, key.length, "AES");
        return decryptMsg(cryptText, OriginKey);
    }

    public static String decryptMsg(byte[] cipherText, SecretKey secretKey) {
        try{
            Cipher cipher = Cipher.getInstance("AES");
            cipher.init(Cipher.DECRYPT_MODE, secretKey);
            return new String(cipher.doFinal(cipherText));

        }catch (NoSuchAlgorithmException | NoSuchPaddingException |
IllegalBlockSizeException |
                BadPaddingException | InvalidKeyException e){
            throw new RuntimeException(e);
        }
    }
}
```

Листинг 5. Класс «MyLoader»

В модуле «ServiceApp» была реализована функция воспроизведения музыки через фоновую службу. При нажатии на кнопку появляется сообщение и начинает играть музыка, которая продолжает играть после сворачивания приложения. При нажатии на кнопку стоп служба останавливается (см. Рисунок 8 и Листинг 6).

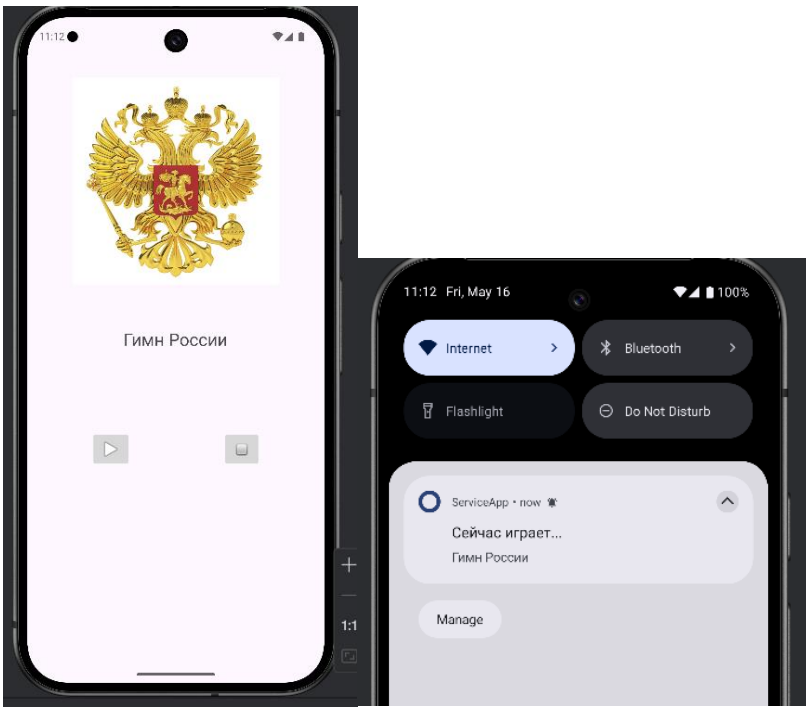

Рисунок 8. Пример работы приложения

```
public class PlayerService extends Service {
    private MediaPlayer mediaPlayer;
    public static final String CHANNEL_ID = "ForegroundServiceChannel";

    @Override
    public IBinder onBind(Intent intent) {
        // TODO: Return the communication channel to the service.
        throw new UnsupportedOperationException("Not yet implemented");
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        mediaPlayer.start();
        mediaPlayer.setOnCompletionListener(new
MediaPlayer.OnCompletionListener() {
            @Override
            public void onCompletion(MediaPlayer mp) {
                stopForeground(true);
            }
        });
        return super.onStartCommand(intent, flags, startId);
    }



    @SuppressLint("ForegroundServiceType")
    @Override
    public void onCreate() {
        super.onCreate();
        NotificationCompat.Builder builder = new
NotificationCompat.Builder(this, CHANNEL_ID)
                .setContentText("Playing Anthem of Russia")
                .setSmallIcon(R.mipmap.ic_launcher)
                .setPriority(NotificationCompat.PRIORITY_HIGH)
                .setStyle(new NotificationCompat.BigTextStyle().bigText("Гимн
России"))
                .setContentTitle("Сейчас играет...");
        int importance = NotificationManager.IMPORTANCE_DEFAULT;
```

```
        NotificationChannel channel = null;
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            channel = new NotificationChannel(CHANNEL_ID, "Burmistrov IG
Notification", importance);
        }
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            channel.setDescription("MIREA Channel");
        }
        NotificationManagerCompat notificationManager =
NotificationManagerCompat.from(this);
        assert channel != null;
        notificationManager.createNotificationChannel(channel);
        startForeground(1, builder.build());

        mediaPlayer = MediaPlayer.create(this, R.raw.gimn_rossii_hor);
        mediaPlayer.setLooping(false);



    }
    @Override
    public void onDestroy() {
        stopForeground(true);
        mediaPlayer.stop();

    }
}
```

Листинг 6. Класс «PlayerService»

В следующем модуле «WorkManager» был добавлен критерий запуска «worker». Если устройство не подключено к wi-fi и к зарядке, то фоновый процесс не будет запущен (см. Рисунок 9 и Листинг 7).

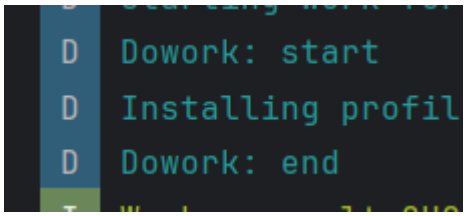

Рисунок 9. Пример выполнения процесса

```
public class MainActivity extends AppCompatActivity {

    private ActivityMainBinding binding;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_main);

        binding = ActivityMainBinding.inflate(getLayoutInflater());
        View view = binding.getRoot();
        setContentView(view);


        Constraints constraints = new Constraints.Builder()
                .setRequiredNetworkType(NetworkType.UNMETERED)
                        .setRequiresCharging(true)
                                .build();
```

```
        WorkRequest uploadWorkRequestt = new
OneTimeWorkRequest.Builder(UploadWorker.class)
                .setConstraints(constraints)
                        .build();

        WorkManager
                .getInstance(this)
                .enqueue(uploadWorkRequestt);

        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;
        });
    }
}
```

Листинг 7. Процесс с ограничением запуска

В проект «MireaProject» (в папке Lesson3) был добавлен ещё один фрагмент, который позволяет включать музыку через сервис, как это было реализовано в модуле «ServiceApp» (см. Рисунок 10 и Листинг 8).

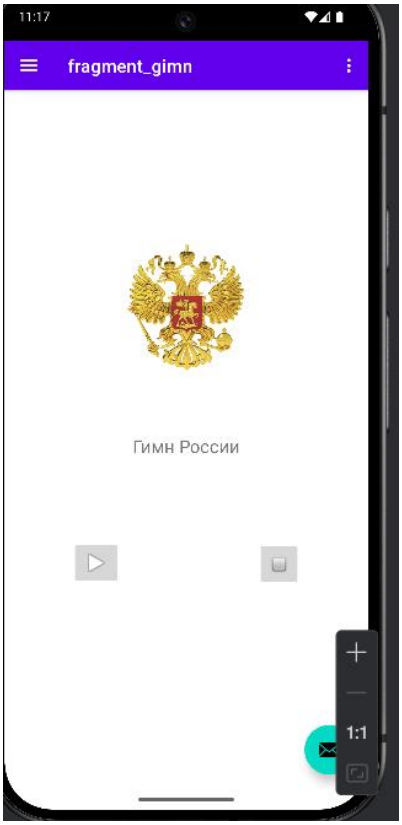

Рисунок 10. Музыкальный плеер

```
public class Gimn extends Fragment {
    private FragmentGimnBinding binding;

    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup
container, Bundle savedInstanceState) {
        binding = FragmentGimnBinding.inflate(inflater, container, false);
```

```
        return binding.getRoot();
    }

    @Override
    public void onViewCreated(@NonNull View view, Bundle savedInstanceState)
{
        super.onViewCreated(view, savedInstanceState);

        binding.imageButton.setOnClickListener(v -> {
            Intent serviceIntent = new Intent(requireContext(),
PlayerService.class);
            ContextCompat.startForegroundService(requireContext(),
serviceIntent);
        });

        binding.imageButton2.setOnClickListener(v -> {
            requireContext().stopService(new Intent(requireContext(),
PlayerService.class));
        });
    }

    @Override
    public void onDestroyView() {
        super.onDestroyView();
        binding = null;
    }
}
```

Листинг 8. Код фрагмента