

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University**

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
GRADUATION THESIS**

**Сравнительный анализ библиотек для воспроизведения аудио и видео контента для
операционной системы Android**

Обучающийся / Student Кадыров Амирджон Халимджонович

Факультет/институт/кластер/ Faculty/Institute/Cluster факультет программной инженерии и компьютерной техники

Группа/Group P34113

Направление подготовки/ Subject area 09.03.04 Программная инженерия

Образовательная программа / Educational program Системное и прикладное программное обеспечение 2018


Язык реализации ОП / Language of the educational program Русский

Статус ОП / Status of educational program

Квалификация/ Degree level Бакалавр

Руководитель ВКР/ Thesis supervisor Романова Асель, Университет ИТМО, факультет программной инженерии и компьютерной техники, старший преподаватель (квалификационная категория "старший преподаватель")

Обучающийся/Student

Документ подписан	
Кадыров Амирджон Халимджонович	
23.05.2022	

(эл. подпись/ signature)

Кадыров
Амирджон
Халимджонович

ч

(Фамилия И.О./ name
and surname)

Руководитель ВКР/
Thesis supervisor

Документ подписан	
Романова Асель	
23.05.2022	

(эл. подпись/ signature)

Романова Асель

(Фамилия И.О./ name
and surname)

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University**

**ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ /
OBJECTIVES FOR A GRADUATION THESIS**

Обучающийся / Student Кадыров Амирджон Халимджонович
Факультет/институт/кластер/ Faculty/Institute/Cluster факультет программной инженерии и компьютерной техники
Группа/Group P34113
Направление подготовки/ Subject area 09.03.04 Программная инженерия
Образовательная программа / Educational program Системное и прикладное программное обеспечение 2018
Язык реализации ОП / Language of the educational program Русский
Статус ОП / Status of educational program
Квалификация/ Degree level Бакалавр
Тема ВКР/ Thesis topic Сравнительный анализ библиотек для воспроизведения аудио и видео контента для операционной системы Android
Руководитель ВКР/ Thesis supervisor Романова Асель, Университет ИТМО, факультет программной инженерии и компьютерной техники, старший преподаватель (квалификационная категория "старший преподаватель")

Основные вопросы, подлежащие разработке / Key issues to be analyzed

Техническое задание и исходные данные к работе:

Провести сравнительный анализ плееров ОС Android для воспроизведения медиа контента при разработке мобильных приложений.

Содержание выпускной квалификационной работы (перечень подлежащих разработке вопросов):

1. Изучение ExoPlayer-а и MediaPlayer-а с использованием интернет-ресурсов
2. Определение критериев сравнения для плееров
3. Реализация работы каждого плеера с аудио и видео контентом
4. Сравнение плееров по основным критериям
5. Выбор наиболее подходящего плеера для работы с медиа контентом

Форма представления материалов ВКР / Format(s) of thesis materials:

Презентация по проделанной работе (в формате Microsoft PowerPoint)

Дата выдачи задания / Assignment issued on: 01.02.2022

Срок представления готовой ВКР / Deadline for final edition of the thesis 30.05.2022

Характеристика темы ВКР / Description of thesis subject (topic)

Название организации-партнера / Name of partner organization: -

Тема в области фундаментальных исследований / Subject of fundamental research: нет / not

Тема в области прикладных исследований / Subject of applied research: да / yes

СОГЛАСОВАНО / AGREED:


Руководитель ВКР/
Thesis supervisor

Документ подписан	
Романова Асель	
11.05.2022	

(эл. подпись)

Романова Асель


Задание принял к
исполнению/ Objectives
assumed BY

Документ подписан	
Кадыров Амирджон Халимджонович	
11.05.2022	

(эл. подпись)

Кадыров
Амирджон
Халимджонович

Руководитель ОП/ Head
of educational program

Документ подписан	
Дергачев Андрей Михайлович	
15.05.2022	

(эл. подпись)

Дергачев
Андрей
Михайлович

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University**

**АННОТАЦИЯ
ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ
SUMMARY OF A GRADUATION THESIS**

Обучающийся / Student Кадыров Амирджон Халимджонович
Факультет/институт/кластер/ Faculty/Institute/Cluster факультет программной инженерии и компьютерной техники
Группа/Group P34113
Направление подготовки/ Subject area 09.03.04 Программная инженерия
Образовательная программа / Educational program Системное и прикладное программное обеспечение 2018
Язык реализации ОП / Language of the educational program Русский
Статус ОП / Status of educational program
Квалификация/ Degree level Бакалавр
Тема ВКР/ Thesis topic Сравнительный анализ библиотек для воспроизведения аудио и видео контента для операционной системы Android
Руководитель ВКР/ Thesis supervisor Романова Асель, Университет ИТМО, факультет программной инженерии и компьютерной техники, старший преподаватель (квалификационная категория "старший преподаватель")

**ХАРАКТЕРИСТИКА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ
DESCRIPTION OF THE GRADUATION THESIS**

Цель исследования / Research goal

Проведение сравнительного анализа библиотек ОС Android для воспроизведения медиа-контента для сокращения времени выбора библиотеки при разработке мобильных приложений.


Задачи, решаемые в ВКР / Research tasks

1. Обзор библиотек ОС Android; 2. Определить критерии сравнения рассматриваемых библиотек; 3. Реализация работы библиотек с аудио и видео контентом; 4. Сравнения библиотек по основным критериям; 5. Выбор наиболее подходящей библиотеки для работы с медиа контентом

Краткая характеристика полученных результатов / Short summary of results/findings

1. Проведен обзор библиотек для воспроизведения медиафайлов 2. Описаны критерии сравнения 3. Разработано мобильное приложение для реализации работы библиотек 4. Получены результаты сравнения с помощью приложения по описанным критериям

Обучающийся/Student

Документ подписан	
Кадыров Амирджон	

Кадыров
Амирджон

Руководитель ВКР/
Thesis supervisor

Халимджонович	
23.05.2022	

(эл. подпись/ signature)

Халимджонович

(Фамилия И.О./ name and surname)

Документ подписан	
Романова Асель	
23.05.2022	

(эл. подпись/ signature)

Романова Асель

(Фамилия И.О./ name and surname)

СОДЕРЖАНИЕ

ТЕРМИНЫ И ОПРЕДЕЛЕНИЕ	7
ВВЕДЕНИЕ.....	8
1. ОПИСАНИЕ БИБЛИОТЕК ДЛЯ ОС ANDROID.....	10
1.1 MediaPlayer.....	10
1.2 ExoPlayer	14
2. ОПРЕДЕЛЕНИЕ КРИТЕРИЕВ ДЛЯ СРАВНЕНИЯ	19
2.1 Критерии сравнения для эффективной работы плеера	19
2.2 Критерии сравнения пользовательского интерфейса.	20
3. РЕАЛИЗАЦИЯ РАБОТЫ БИБЛИОТЕК	22
3.1 Проектирование и разработка приложения для сравнения работы библиотек	22
4. СРАВНИТЕЛЬНЫЙ АНАЛИЗ БИБЛИОТЕК	33
4.1 Результаты измерений времени первого воспроизведения	33
4.2 Сравнение поддерживаемых форматов.....	37
4.3 Сравнение реакций на изменение скорости интернета.....	40
4.4 Сравнение возможности кастомизации UI плеера	42
4.5 Возможности оптимизации библиотек.....	43
ЗАКЛЮЧЕНИЕ	46
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	47
ПРИЛОЖЕНИЕ А Реализация Navigation Drawer	49
ПРИЛОЖЕНИЕ Б Реализация компонентов RadioButton.....	51
ПРИЛОЖЕНИЕ В Реализация предпросмотра контента через бегунок прогресса.....	53

ТЕРМИНЫ И ОПРЕДЕЛЕНИЕ

API – набор готовых классов, функций, процедур, структур и констант.

URI – последовательность символов, идентифицирующая физический или абстрактный ресурс, который не обязательно должен быть доступен через сеть Интернет, причем, тип ресурса, к которому будет получен доступ, определяется контекстом и/или механизмом.

URL-адрес – местонахождение определенного Интернет-ресурса.

Gradle – система сборки, используемая в том числе и в Android.

Контент (медиа-контент) – это совокупность данных, информации, которая представлена с помощью аудио и видео эффектов.

HLS – коммуникационный протокол для потоковой передачи медиа на основе HTTP, разработанный компанией Apple как часть программного обеспечения QuickTime, Safari, OS X и iOS.

DASH – технология адаптивной потоковой передачи данных, предоставляющая возможность доставки потокового мультимедиа-контента через Интернет по протоколу HTTP.

ВВЕДЕНИЕ

При разработке приложений для ОС Android для работы с медиа контентом необходимо решить множество задач, к числу которых относится воспроизведение контента. Если начать разрабатывать весь программный код с нуля, то на это может уйти большое количество времени. В этом случае возникает необходимость рассмотреть вопросы, связанные с кодеками, кастомными компонентами, кэшированием, потоками, управлением сетевыми соединениями, обработкой исключений и многим другим. Поэтому наилучшим вариантом будет использование одной из существующих библиотек.

Большинство разработчиков не пишут собственную реализацию воспроизведения контента для приложений. Вместо этого они используют готовые решения, разработанные для той или иной платформы.

В мире ОС Android самыми популярными библиотеками являются MediaPlayer и ExoPlayer. Кроме того, в настоящее время всё большую популярность набирает Jetpack Player 3, но данный инструмент является оболочкой над ExoPlayer. У каждой из существующих библиотек есть свои преимущества и недостатки.

Следовательно, возникает вопрос, какую библиотеку лучше всего использовать в промышленной разработке. *Проблема* является в выборке библиотеки при разработке приложения. Android разработчики тратят время на выбор библиотеки изучая их документацию.

Необходимость ответа на поставленный вопрос обуславливает актуальность выбранной темы выпускной квалификационной работы.

Целью выпускной квалификационной работы является проведение сравнительного анализа библиотек ОС Android для воспроизведения медиа контента при разработке мобильных приложений.

В соответствии с поставленной целью были определены следующие задачи:

- Обзор библиотек ОС Android;
- Определить критерии сравнения рассматриваемых библиотек;
- Реализация работы библиотек с аудио и видео контентом;
- Сравнения библиотек по основным критериям;
- Выбор наиболее подходящей библиотеки для работы с медиа-контентом

Полученные результаты с теоретической значимости проведенного исследования могут послужить:

- теоретической и практической базой для ознакомления с рассмотренными библиотеками.
- получением наиболее полного представления о библиотеках, исключая необходимость поиска нужной библиотеки

Практическая значимость данного исследования заключается в выборе библиотеки в зависимости от функциональных требований при разработке приложения.

1. ОПИСАНИЕ БИБЛИОТЕК ДЛЯ ОС ANDROID

Воспроизведение аудио и видео контента для ОС Android достигается путем использования библиотек MediaPlayer [1] или ExoPlayer [2]. ExoPlayer - проект с открытым исходным кодом, который не является частью платформы Android и распространяется отдельно от Android SDK. Можно было рассмотреть такие библиотеки какijkplayer или Vitamio, но они не поддерживаются уже несколько лет. Кроме того, MediaPlayer и ExoPlayer единственные библиотеки, которые рекомендованы компанией Google, поэтому в выпускной квалификационной работе будут рассмотрены только MediaPlayer и ExoPlayer.

1.1 MediaPlayer

MediaPlayer является частью мультимедийной платформы Android, которая воспроизводит аудио и видео из каталога ресурсов или из галереи устройства. Благодаря данной библиотеке можно воспроизвести музыку или видео с URL-адреса. При воспроизведении медиафайла MediaPlayerAPI [3], [4] проходит через несколько состояний (Рисунок 1):

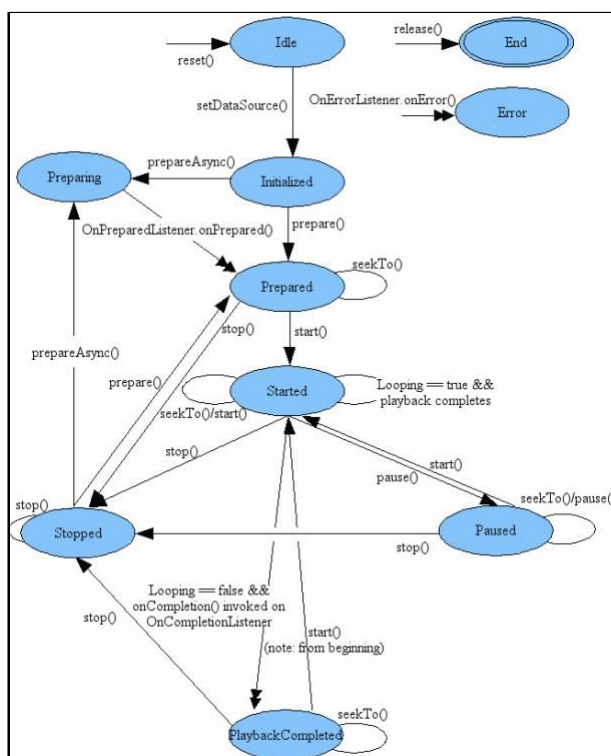


Рисунок 1 – Описание жизненного цикла MediaPlayer

1. Состояние простоя (Idle State): MediaPlayer находится в состоянии ожидания, когда впервые создается экземпляр. Данное состояние достигается после вызова метода `reset()`. На этом этапе нельзя воспроизводить, приостанавливать или останавливать контент. Если попытаться сделать это принудительно, приложение может завершиться с ошибкой.
2. Конечное состояние (End State): Вызов метода `release()` освобождает ресурсы и переводит их в конечное состояние. На данном этапе нельзя воспроизводить или приостанавливать воспроизведение контента.
3. Состояние ошибки (Error State): если попытаться воспроизвести, приостановить или остановить экземпляр MediaPlayer объекта, у которого не установлен источник данных, то возникает состояние ошибки. Однако можно поймать исключение, используя обратный вызов `MediaPlayer.OnErrorListener.onError()`.
4. Инициализированное состояние (Initialized State): достигается при установке источника данных. Для этого используется метод `setDataSource()`. Можно установить источник данных, когда MediaPlayer находится в состоянии ожидания, иначе он вызовет исключение `IllegalStateException`.
5. Подготовленное состояние (Prepared State): Перед воспроизведением любого мультимедиа из файла или URL-адреса, необходимо “подготовить” MediaPlayer, вызвав либо метод `prepare()`, либо `prepareAsync()`. После срабатывает обратный вызов `onPreparedListener()`.
6. Запущенное состояние (Started State): когда MediaPlayer находится в подготовленном состоянии, можно воспроизвести медиафайл, вызвав

метод `start()`. Во время воспроизведения музыки или видео `MediaPlayer` находится в запущенном состоянии.

7. Состояние паузы (Paused State): при приостановке воспроизведения контента, `MediaPlayer` находится в текущем состоянии. Для этого необходимо вызвать метод `pause()`.
8. Stopped State: `MediaPlayer` находится в данном состоянии, во время прекращения воспроизведение мультимедиа. Если `MediaPlayer` находится в данном состоянии и нужно снова воспроизвести медиафайл, необходимо снова подготовить его, вызвав `prepare()` или `prepareAsync()`.
9. Playback Completed: когда воспроизведение завершено, вызывается обратный вызов `onCompletion()`. Если `MediaPlayer` находится в текущем состоянии, можно вызвать `start()` и снова воспроизвести медиафайл.

Следует отметить, что `MediaPlayer` не является потокобезопасным. Создание экземпляров проигрывателя и все доступы к ним должны осуществляться в одном потоке. При регистрации обратных вызовов у потока должен быть `Looper`.

Воспроизведения локального аудио контента осуществляется путем вызова метода `start()` объекта `MediaPlayer` (Рисунок 2).

```
mediaPlayerAudio = MediaPlayer.create(context, R.raw.audio)
mediaPlayerAudio?.start()
```

Рисунок 2 - Воспроизведение локального аудио контента

В случае воспроизведения контента из Интернет-ресурсов или из хранилища устройства пользователя, нужно явно указывать путь до ресурса (Рисунок 3).

```

val myUri: Uri? = Uri.parse(audioUrl) // initialize Uri here
mediaPlayerAudio = MediaPlayer().apply {
    setAudioAttributes(
        AudioAttributes.Builder()
            .setContentType(AudioAttributes.CONTENT_TYPE_MUSIC)
            .setUsage(AudioAttributes.USAGE_MEDIA)
            .build()
    )
    setDataSource(context!!, myUri!!)
    prepare()
    start()
}

```

Рисунок 3 - Воспроизведение аудио контента из сети

Воспроизведение локального видео контента следует осуществлять двумя способами:

1. Использовать MediaPlayer и SurfaceView [6];
2. Использовать VideoView [5]

Если использовать SurfaceView нужно явно указать на элемент отображения MediaPlayer (Рисунок 4).

```

SurfaceView sv = (SurfaceView) findViewById(R.id.surfaceView1);
SurfaceHolder holder = sv.getHolder();
holder.addCallback(new Callback(){
    @Override
    public void surfaceChanged(SurfaceHolder holder, int format, int width, int height) { }

    @Override
    public void surfaceCreated(SurfaceHolder holder) {
        mp.setDisplay(holder);
        mp.start();
    }

    @Override
    public void surfaceDestroyed(SurfaceHolder holder) { }
});

```

Рисунок 4 - Привязка SurfaceView к MediaPlayer

VideoView является объединением MediaPlayer и SurfaceView. Для того чтобы воспроизвести видео контент через VideoView нужен Uri контента и объект класса MediaController (Представление, содержащее элементы управления для MediaPlayer. Обычно содержит такие кнопки, как «Воспроизведение/Пауза», «Перемотка назад», «Перемотка вперед» и бегунок прогресса (seekbar). MediaController самостоятельно делает синхронизацию элементов управления с состоянием MediaPlayer (Рисунок 5).

```
val uri: Uri = Uri.parse(videoPath)
videoViewMP.setVideoURI(uri)
val mediaController = MediaController(context)
videoViewMP.setMediaController(mediaController)
mediaController.setAnchorView(videoViewMP)
videoViewMP.start()
```

Рисунок 5 - Воспроизведение видео контента из сети

В переменной videoPath указывается местоположения видео контента. Это может быть путь до директории ресурсов или URL видео из Интернет-ресурса.

1.2 ExoPlayer

ExoPlayer - медиаплеер уровня приложения для Android, представляет собой альтернативу Android MediaPlayer API для воспроизведения аудио и видео как локально, так и через Интернет. Самый простой способ начать использовать ExoPlayer — добавить его в качестве зависимости в gradle в файл build.gradle модуля приложения (Рисунок 6).

```
implementation 'com.google.android.exoplayer:exoplayer:2.X.X'
```

Рисунок 6 - Добавление зависимости в build.gradle

В качестве альтернативы использования всей библиотеки, можно полагаться только на те библиотечные модули, которые действительно нужны. Например, добавить зависимости от модулей библиотеки Core, DASH и пользовательского интерфейса (Рисунок 7):

```
implementation 'com.google.android.exoplayer:exoplayer-core:2.X.X'  
implementation 'com.google.android.exoplayer:exoplayer-dash:2.X.X'  
implementation 'com.google.android.exoplayer:exoplayer-ui:2.X.X'
```

Рисунок 7 - Добавление зависимости отдельных модулей в build.gradle

Для того, чтобы создать экземпляр класса ExoPlayer, можно использовать ExoPlayer.Builder, который предоставляет ряд параметров для настройки. Приведенный ниже код является простейшим примером создания экземпляра (Рисунок 8).

```
ExoPlayer player = new ExoPlayer.Builder(context).build();
```

Рисунок 8 - Создание объекта ExoPlayer

Библиотека ExoPlayer предоставляет ряд готовых компонентов пользовательского интерфейса для воспроизведения мультимедиа. К ним относятся StyledPlayerView [7], который инкапсулирует логику StyledPlayerControlView, SubtitleView и Surface. StyledPlayerView может быть включен в XML-файл макета приложения. Привязать плеер к представлению можно сделать посредством вызова метода setPlayer() (Рисунок 9).

```
// Bind the player to the view.  
playerView.setPlayer(player);
```

Рисунок 9 - Привязка View к ExoPlayer

В ExoPlayer каждый элемент мультимедиа представлен объектом класса `MediaItem`. Чтобы воспроизвести медиа файл, нужно создать соответствующий объект `MediaItem`, добавить его в проигрыватель и вызвать метод `play()` для запуска воспроизведения (Рисунок 10).

```
// Build the media item.  
MediaItem mediaItem = MediaItem.fromUri(videoUri);  
// Set the media item to be played.  
player.setMediaItem(mediaItem);  
// Prepare the player.  
player.prepare();  
// Start the playback.  
player.play();
```

Рисунок 10 - Воспроизведение видео контента из сети с ExoPlayer

ExoPlayer напрямую поддерживает воспроизведения списка из медиа файлов, поэтому можно подготовить проигрыватель с несколькими элементами мультимедиа, которые будут воспроизводиться один за другим (Рисунок 11). Также список элементов воспроизведения можно обновлять во время воспроизведения без необходимости повторной подготовки плеера.


```
// Build the media items.  
MediaItem firstItem = MediaItem.fromUri(firstVideoUri);  
MediaItem secondItem = MediaItem.fromUri(secondVideoUri);  
// Add the media items to be played.  
player.addItem(firstItem);  
player.addItem(secondItem);  
// Prepare the player.  
player.prepare();  
// Start the playback.  
player.play();
```

Рисунок 11 - Добавление несколько медиаэлементов

Когда объект `ExoPlayer` подготовлен, то можно управлять воспроизведением, вызывая методы проигрывателя. Некоторые из наиболее часто используемых методов перечислены ниже:

- `play()` и `pause()` - запуск и приостановка воспроизведения.
- `seekTo()` - перемещение к заданной временной позиции контента
- `hasPrevious()`, `hasNext()`, `previous()`, `next()` - функции для навигации по плейлисту.
- `setRepeatMode()` - повтор воспроизведение контента.
- `setShuffleModeEnabled()` - управляет перемешиванием списка воспроизведения.
- `setPlaybackParameters()` - регулирует скорость воспроизведения и уровень звука.

Если проигрыватель привязан к `StyledPlayerView` или `StyledPlayerControlView`, то взаимодействие пользователя с данными компонентами приведет к вызову соответствующих методов проигрывателя.

Важно освободить ресурсы у проигрывателя, если они не используются системой, чтобы освободить ограниченные ресурсы, такие как декодеры

видео, для использования другими приложениями. Это можно сделать, вызвав метод `ExoPlayer.release()`.

`ExoPlayer` сообщает зарегистрированным экземплярам класса `Player.Listener` о таких событиях, как изменение состояния плеера или возникновение ошибки воспроизведения. Зарегистрировать слушателя для получения таких событий осуществляется посредством вызова метода `addListener` [8] (Рисунок 12).

```
// Add a listener to receive events from the player.  
player.addListener(listener);
```

Рисунок 12 - Добавление слушателя событий в `ExoPlayer`

Таким образом были рассмотрены две библиотеки у которых схожий функционал для работы с медиа контентом. Для дальнейших действий нужно определить критерии для сравнения данных библиотек между собой и разработать приложение.

2. ОПРЕДЕЛЕНИЕ КРИТЕРИЕВ ДЛЯ СРАВНЕНИЯ

Чтобы составить список характеристик для сравнения библиотек, следует исходить из пользовательского опыта (UX) использование плееров. Основными критериями являются: скорость работы библиотек, поддержка форматов, работа библиотек при разной скорости интернета и возможности оптимизации. Не менее важным критерием для сравнения является пользовательский интерфейс для плеера. Таким образом можно разделить критерии сравнения на две части: для эффективной работы плеера и для пользовательского интерфейса плеера.

2.1 Критерии сравнения для эффективной работы плеера

Как было описано выше, основными критериями эффективной работы плеера являются:

1. Быстрота плеера (время первого воспроизведения);
2. Поддержка форматов;
3. Реакция на изменение скорости интернета;
4. Возможность оптимизации

Для проверки быстроты работы плеера нужно рассчитать время, потраченное от начала инициализации плеера до воспроизведения контента. Нужно проверить разные типы медиа файлов (аудио и видео), учитывая расположения медиафайлов (локальный файл или из сети).

Для сравнения поддержки форматов следует изучить документацию библиотек и составить список поддерживаемых форматов для аудио и видео контента. Также нужно проверить поддержки протокола HLS [9] (HTTP Live Streaming) для каждой библиотеки и работу с таким протоколом.

Реакцию на изменение скорости интернета стоит проверять для работы библиотек с воспроизведением контента из сети. Нужно проверить размер буферизации контента (Рисунок 13), также возможность изменения размера

буфера для оптимизации времени первого воспроизведение контента и возможность изменения качества видео (Рисунок 14) во время воспроизведения.



Рисунок 13 - Белый индикатор указывающий размер буфера

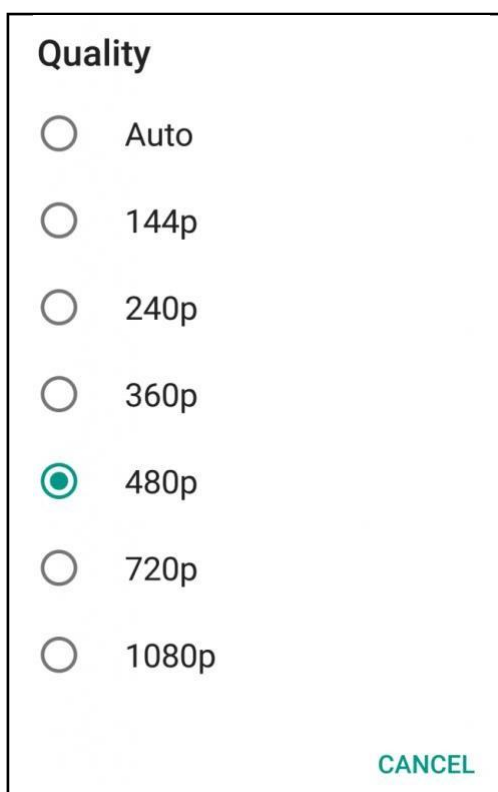


Рисунок 14 - Выбор качества видео

2.2 Критерии сравнения пользовательского интерфейса.

Основные требования плеера для пользователя, это его элементы управления, такие как пауза, воспроизведение, стоп, следующий, предыдущий и т.д. (Рисунок 15)

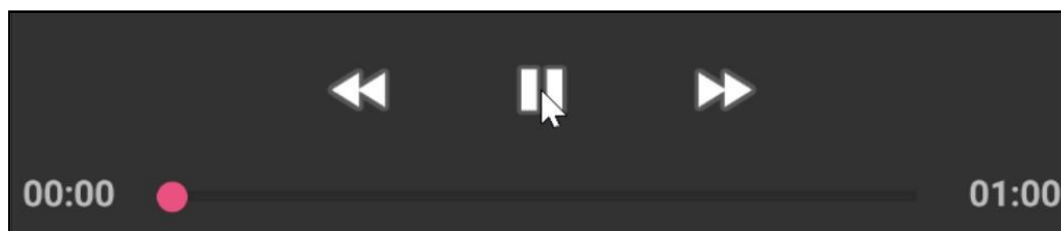


Рисунок 15 – Основные элементы управления плеера

Наличие дополнительных элементов управления (Рисунки 16-18), таких как перемотка на 10 секунд вперед или назад, предпросмотр контента через бегунок прогресса, кнопка блокировки от случайных нажатий на экран и т.д. улучшают пользовательский интерфейс плеера.

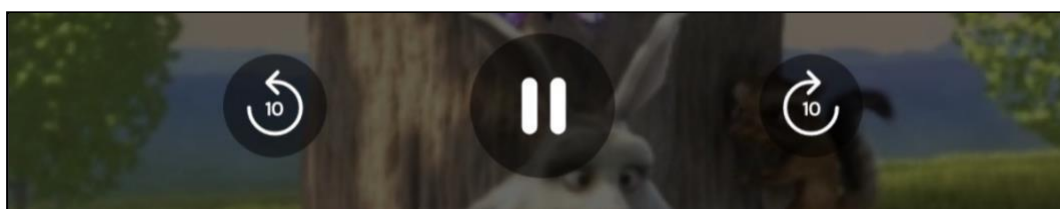


Рисунок 16 – Перемотка на 10 секунд



Рисунок 17 – Кнопка блокировки экрана

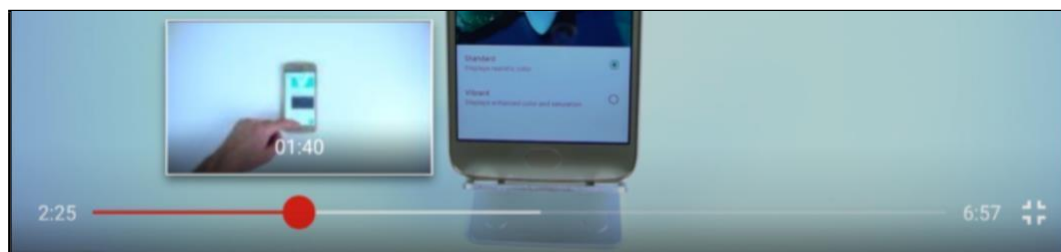


Рисунок 18 – Функция предпросмотра контента

3. РЕАЛИЗАЦИЯ РАБОТЫ БИБЛИОТЕК

Для проведения сравнительного анализа необходимо реализовать работу библиотек.

3.1 Проектирование и разработка приложения для сравнения работы библиотек

Было разработано мобильное приложение для ОС Android на языке программирования Kotlin.

Требования к разрабатываемому приложению:

- Воспроизведение медиафайла из локального хранилища или из сети;
- Выбор параметров для сравнения:
 - параметр выборки библиотеки;
 - параметр выборки типа медиафайла;
 - параметр выборки местоположения медиафайла;
 - параметр выборки внешнего вида плеера
- Вывод результатов измерений

Use-case модель приложения на Рисунке 19

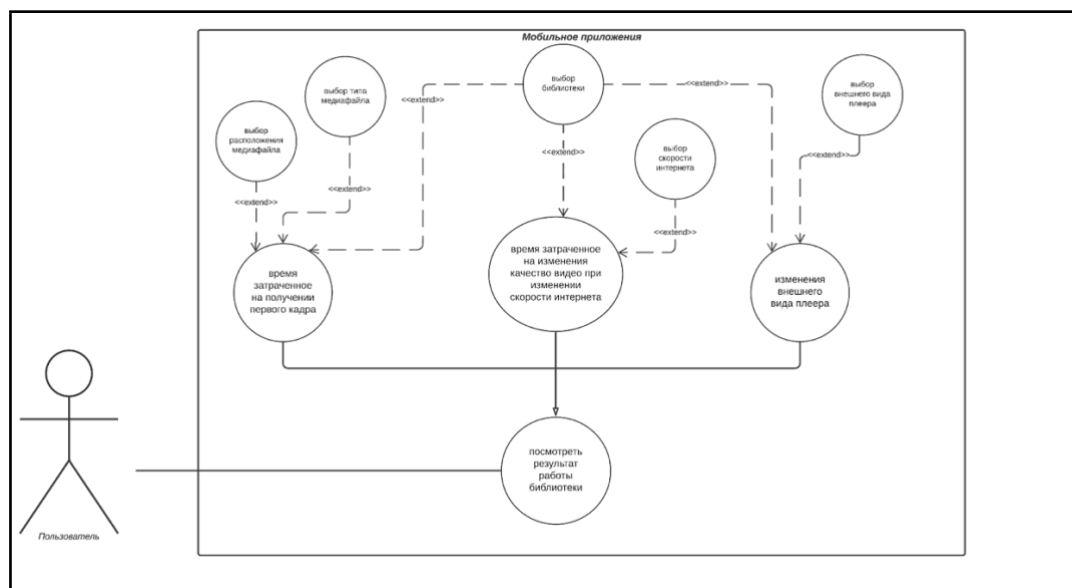


Рисунок 19 – Use case модель приложения

Для разработки макета приложения был использован онлайн сервис Figma. Было создано 3 макета страниц (экранов) для работы библиотек:

- Экран для сравнения времени первого воспроизведения;
- Экран для сравнения поведения библиотек на изменение скорости интернета;
- Экран для сравнения возможности кастомизации UI плеера

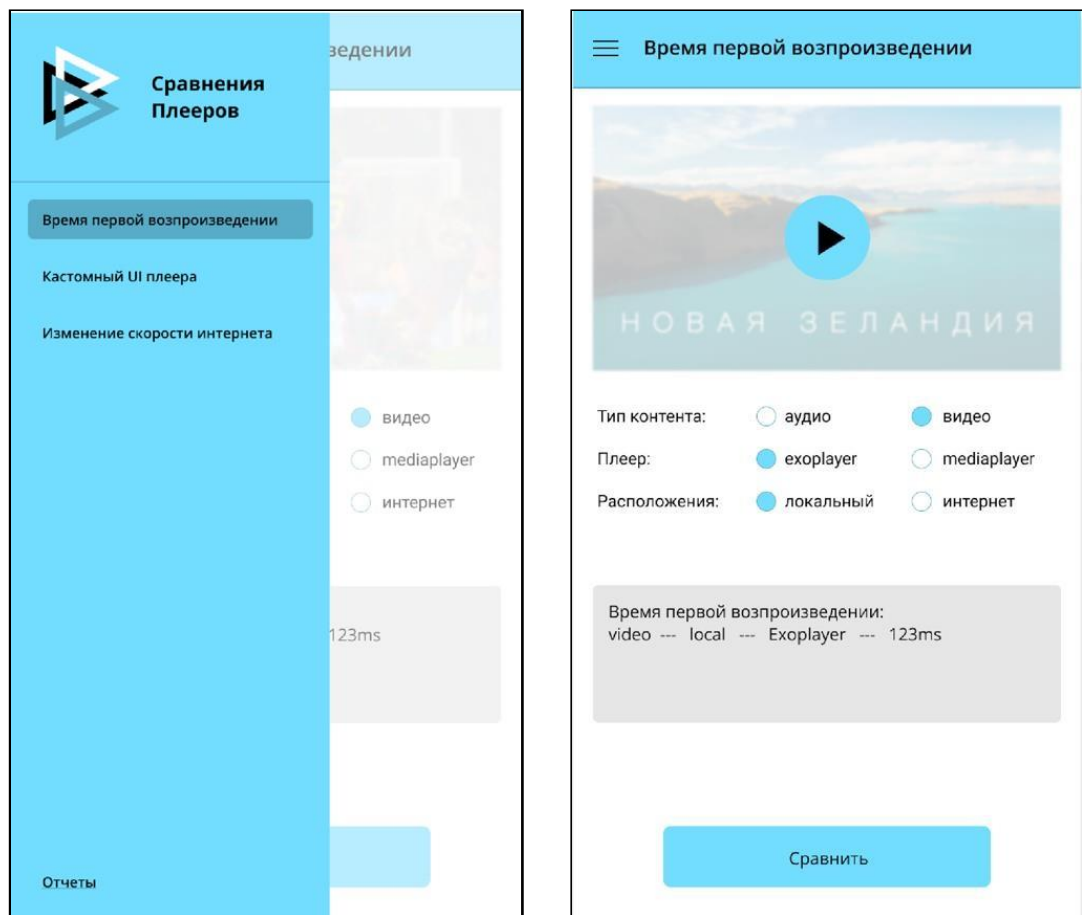


Рисунок 20 - Макет первого экрана

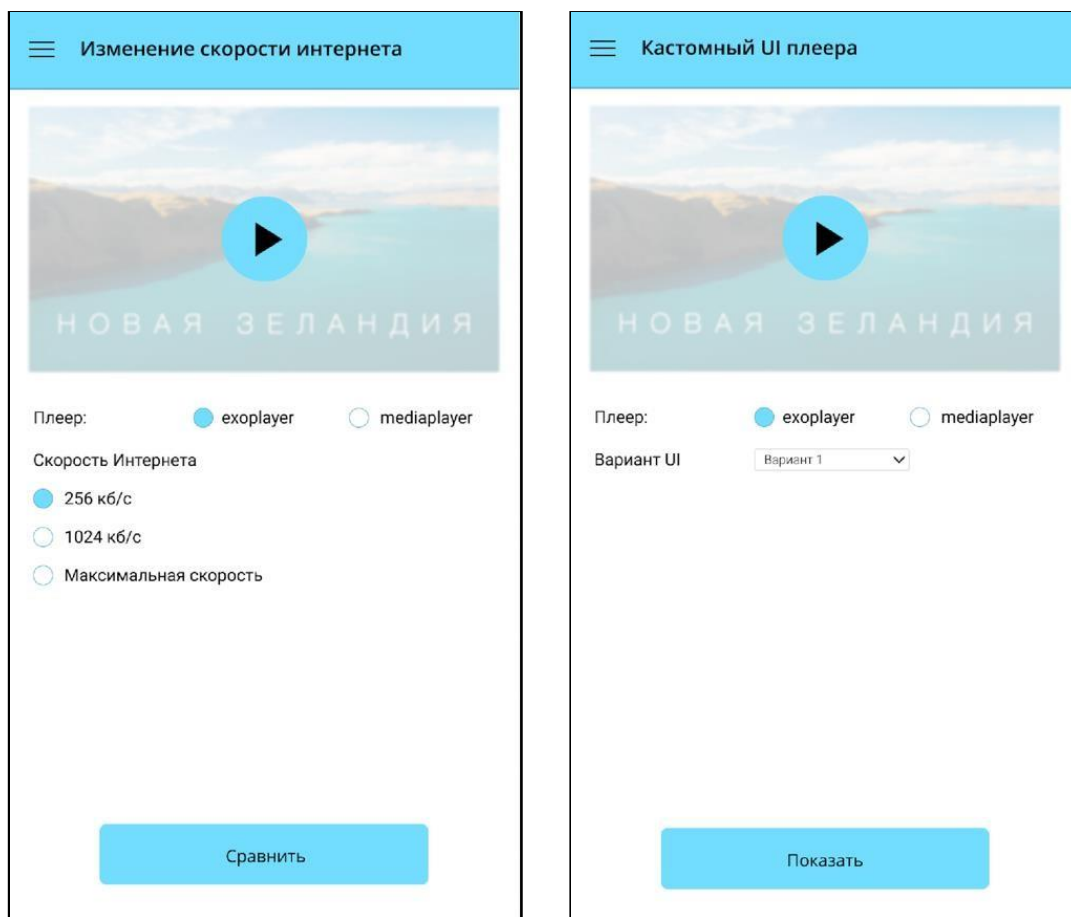


Рисунок 21 - Макет второго и третьего экрана

Для навигации между экранами был использован компонент Navigation Drawer из Android SDK. Реализация Navigation Drawer в XML-файле представлена в Приложении А.

В приложении не учитывается возможность хранения пользовательских данных и обработка состояния при повороте и закрытие экрана, поэтому была использована простая MVC архитектура. В качестве среды разработки - Android Studio. В роли системы сборки - Gradle. Приложение было запущено на реальном мобильном устройстве. Основные характеристики устройства:

- тактовая частота процессора - 2300 МГц
- количество ядер процессора - 8
- объем оперативной памяти - 6 ГБ
- частота оперативной памяти - 1866 МГц
- объем встроенной памяти - 64 ГБ

Для того чтобы воспроизвести медиа контент, необходимо выбрать тип контента (аудио или видео), выбрать библиотеку (MediaPlayer или EchoPlayer) и определить, откуда брать контент (из локального хранилища или из сети).

Данная функциональность была реализована посредством компонентов RadioButton, которая используется в составе контейнера RadioGroup. Реализация компонентов RadioButton в XML-файле для выбора типа контента, библиотеки и местоположения контента представлена в Приложении Б. Таким образом, в каждом контейнере RadioGroup есть несколько переключателей, и только один из них может быть выбран.

После того, как все необходимые параметры выбраны, нужно воспользоваться кнопкой «Сравнить». Компонент Button реализован в XML-файле (Листинг 1).

```
<Button
    android:id="@+id/btn_done"
    android:text="Сравнить"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
/>
```

Листинг 1 – Реализация компонента Button

Чтобы обрабатывать нажатие кнопки, необходимо присвоить ей обработчик с помощью `setOnClickListener` (Листинг 2). Следует отметить, что был добавлен View Binding для того, чтобы каждый раз не вызывать метод `findViewById(R.id.*)`:

```
val btn: Button = binding.btnDone
btn.setOnClickListener {
    if (binding.radioAudio.isChecked) {
        binding.audioPlace.visibility = View.VISIBLE
    }
}
```

```

        playAudio(getPlayer(), getLocation())
    } else {
        binding.videoPlace.visibility = View.VISIBLE
        playVideo(getPlayer(), getLocation())
    }
}

```

Листинг 2 – Настройка обработчика `setOnClickListener`

Проверку выбранного элемента из контейнера `radioGroup`, можно осуществлять с помощью свойства `isChecked`. Используя свойство `isChecked` были описаны методы для получения библиотеки (плеера), локации контента и типа контента (Листинги 3-5):

```

private fun getPlayer(): Int {
    return if (binding.radioMp.isChecked) {
        MEDIA_PLAYER
    } else {
        EXO_PLAYER
    }
}

```

Листинг 3 – Метод для получения типа плеера

```

private fun getLocation(): Int {
    return if (binding.radioInet.isChecked) {
        INTERNET
    } else {
        LOCAL
    }
}

```

Листинг 4 – Метод для получения локации контента

```

private fun getContent(): Int {
    return if (binding.radioAudio.isChecked) {
        AUDIO
    }
}

```

```

        } else {
            VIDEO
        }
    }
}

```

Листинг 5 – Метод для получения типа контента

В зависимости от выбранной библиотеки и типа контента, будет меняться содержание экрана, а именно, компонент, отвечающий за воспроизведение контента. Для управления видимостью элементов использовалось свойство `visibility` класса `View`. Значение `VISIBLE` означает, что элемент видим, а `GONE` – элемент не является видимым и не занимает место в разметке (Листинг 6)

```

binding.*.visibility = View.GONE
binding.*.visibility = View.VISIBLE

```

Листинг 6 – Использование свойства `visibility`

Логика скрывания и отображения компонентов указано в Листинге 2. Когда выбран `MediaPlayer` используется компонент `VideoView`, а в случае `ExoPlayer` `PlayerView` из пакета `com.google.android.exoplayer2.ui`. Для вывода информации пользователю был добавлен `TextView` поверх кнопки (Листинг 7)

```

<TextView
    android:id="@+id/res_txt"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:textColor="@color/white"
/>

```

Листинг 7 – Реализация компонента `TextView`

Для воспроизведения медиа контента были написаны две функции, для воспроизведения аудио контента и видео контента. Логика вызова функций

описана в Листинге 2. Каждый раз перед тем как вызвать данные методы, вызывается `stopPlayer()`, метод для очистки и деинициализации объектов плеера (Листинг 8).

```
private fun stopPlaying() {
    //stopping MediaPlayer
    videoViewMP.stopPlayback()
    videoViewMP.setVideoURI(null)
    videoViewMP.setMediaController(null)

    //stopping ExoPlayer
    playerEP.stop()
    videoViewEP.player = null

    //stop MP audio
    mediaPlayerAudio.stop()
    mediaPlayerAudio = null
}
```

Листинг 8 – Реализация метода `stopPlayer`

Для вывода информации пользователю с помощью компонента `TextView`, был написан метод `updateUi` (Листинг 9). Метод принимает единственный параметр — это время первого воспроизведения контента в формате `String` и вывод информации в отформатированном виде.

```
private fun updateUi(msg: String){
    val textView = binding.resTxt
    val content = if (getContent() == VIDEO) "Видео" else
    "Аудио"
    val location = if (getLocation() == INTERNET) "Сеть" else
    "Локально"
    val player = if (getPlayer() == MEDIA_PLAYER) "MediaPlayer"
    else "ExoPlayer"
```

```

        textView.text = "Вид контента: $content\n Локация контента:
        $location\n Используется плеер: $player\n Время первого
        запуска: $msg ms"
    }

```

Листинг 9 - Реализация метода updateUi

Для того чтобы подсчитать время первого воспроизведения плеера, нужно вычислить разницу времени от начала инициализации плеера до начала воспроизведения контента. Чтобы понять, когда контент начал воспроизведение, нужно произвести поиск в документации каждой библиотеки. В случае MediaPlayer, нужно настроить обработчик `setOnPreparedListener` (Листинг 10)

```

videoViewMP.setOnPreparedListener {
    updateUi((System.currentTimeMillis()-startTime))
}

```

Листинг 10 – Настройка обработчика `setOnPreparedListener`

В случае `ExoPlayer`, нужно настроить обработчик `onPlayerStateChanged` (Листинг 11)

```

playerEP.addListener(object : Player.Listener{
    override fun onPlayerStateChanged(playWhenReady: Boolean,
    playbackState: Int) {
        if (playWhenReady && playbackState == Player.STATE_READY)
            // call update UI
    }
}

```

Листинг 11 - Настройка обработчика `onPlayerStateChanged`

После изучения документации было выяснено, что и `MediaPlayer`, и `ExoPlayer` могут работать с протоколом HLS, и при изменении скорости интернета плеер автоматически меняет качество видео. Для того чтобы

получить информацию о том, что плеер изменил качество видео, в случае MediaPlayer нужно настроить обработчик `setOnVideoSizeChangeListener` в обработчике `setOnPreparedListener` (Листинг 12)

```
videoViewMP.setOnPreparedListener {  
    it.setOnVideoSizeChangeListener {  
        x, p1, p2 ->  
        updateUi("width = $p1, high = $p2, pxWHR = x")  
    }  
}
```

Листинг 12 - Настройка обработчика `setOnVideoSizeChangeListener`

В случае `ExoPlayer`, нужно настроить обработчик `onVideoSizeChanged` (Листинг 13)

```
playerEP?.addVideoListener(object : VideoListener {  
    override fun onVideoSizeChanged(width: Int, height: Int, _,  
    x: Float ) {  
        updateUi("width = $width, high = $height, pxWHR = $x")  
    }  
}))
```

Листинг 13 - Настройка обработчика `onVideoSizeChanged`

`MediaPlayer` не предоставляет функции предпросмотра контента через бегунок прогресса. С помощью библиотеки `ExoPlayer` можно реализовать данный функционал. Пример реализации представлен в Приложении В. Также можно воспользоваться одним из популярных готовых решений таких как `rubensousa/PreviewSeekBar` (Листинги 14)

```
<com.github.rubensousa.previewseekbar.exoplayer.PreviewTimeBar  
    android:id="@+id/previewBar"  
    android:layout_width="match_parent"  
    android:layout_height="14dp"
```

```

app:previewAnimationEnabled="true"
app:previewFrameLayout="@id/previewFrameLayout"
/>

```

Листинг 14 - Реализация компонента PreviewSeekBar

Для кастомизации UI плеера ExoPlayer предоставляет [11] ряд готовых атрибутов для компонентов представлений StyledPlayerControlView, PlayerView и т.д. (Листинг 15)

```

app:fastforward_increment="5000"
app:rewind_increment="5000"
app:show_timeout="2000"
...

```

Листинг 15 – Примеры параметров StyledPlayerControlView

MediaPlayer предоставляет малое количество атрибутов, но если готовые решения не удовлетворяет требованиям к UI разрабатываемого приложения, то можно использовать атрибуты для кастомного дизайна:

- app:controller_layout_id - Exoplayer (Рисунок 22)
- верстать собственные элементы управления поверх VideoView - MediaPlayer

```

<com.google.android.exoplayer2.ui.PlayerView
    android:id="@+id/video_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center"
    app:auto_show="true"
    app:controller_layout_id="@layout/custom_exo
    app:fastforward_increment="10000"
    app:repeat_toggle_modes="none"
    app:resize_mode="fixed_width"
    app:rewind_increment="10000"
    app:surface_type="surface_view"
    app:use_controller="true" />

```

Рисунок 22 - Реализация компонента PlayerView

В итоге было разработано Android-приложение на языке программирования Kotlin, позволяющее воспроизвести медиа контент используя MediaPlayer, либо ExoPlayer. В приложении проверяется работа библиотек по основным критериям сравнения, что позволит оценивать и сравнивать их между собой.

4. СРАВНИТЕЛЬНЫЙ АНАЛИЗ БИБЛИОТЕК

При разработке мобильных приложений, в которых требуется воспроизвести медиа контент, важно учитывать время, необходимое для их первого воспроизведения. Измерить и сравнить время можно с помощью разработанного приложения в данной выпускной квалификационной работе. Причём с помощью данного приложения проверяется, как библиотеки реагируют на изменение скорости интернета (изменение качества видео), и возможность изменения параметров библиотеки в целях повышения его эффективности. Также проверяется возможность кастомизации UI плеера (улучшение пользовательского интерфейса)

4.1 Результаты измерений времени первого воспроизведения

Проводилось измерение времени первого воспроизведения для аудио и видео контента с использованием MediaPlayer и ExoPlayer (Таблица 1-2). Также воспроизведение контента через сеть (Таблица 3-4). Время в таблицах указано в миллисекундах.

Таблица 1 - Время первого воспроизведения контента в MediaPlayer

Попытка	Вид контента	
	аудио	видео
1	51	93
2	67	100
3	70	107
4	57	96
5	60	101
6	54	87
7	56	92

8	51	87
9	61	93
10	66	100
Среднее значение	59	96

Таблица 2 - Время первого воспроизведения контента в EchoPlayer

Попытка	Вид контента	
	аудио	видео
1	193	314
2	185	290
3	169	279
4	232	261
5	182	247
6	137	251
7	167	291
8	163	264
9	170	260
10	181	231
Ср. значение	180	269

Таблица 3 - Время первого воспроизведения контента из сети в MediaPlayer

Попытка	Вид контента	
	аудио	видео
1	261	1250
2	265	1017

3	279	1111
4	288	1050
5	387	1105
6	348	1069
7	319	1080
8	265	1042
9	272	1069
10	259	1016
Ср. значение	294	1080

Таблица 4 - Время первого воспроизведения контента из сета в EchoPlayer

Попытка	Вид контента	
	аудио	видео
1	248	622
2	230	748
3	235	662
4	255	835
5	233	620
6	260	832
7	247	786
8	280	648
9	202	652
10	199	763
Ср. значение	239	717

На основе проведенного исследования, сделан анализ полученных результатов. Для наглядности, все данные были представлены в виде графиков и гистограмм (Рисунки 23-24)

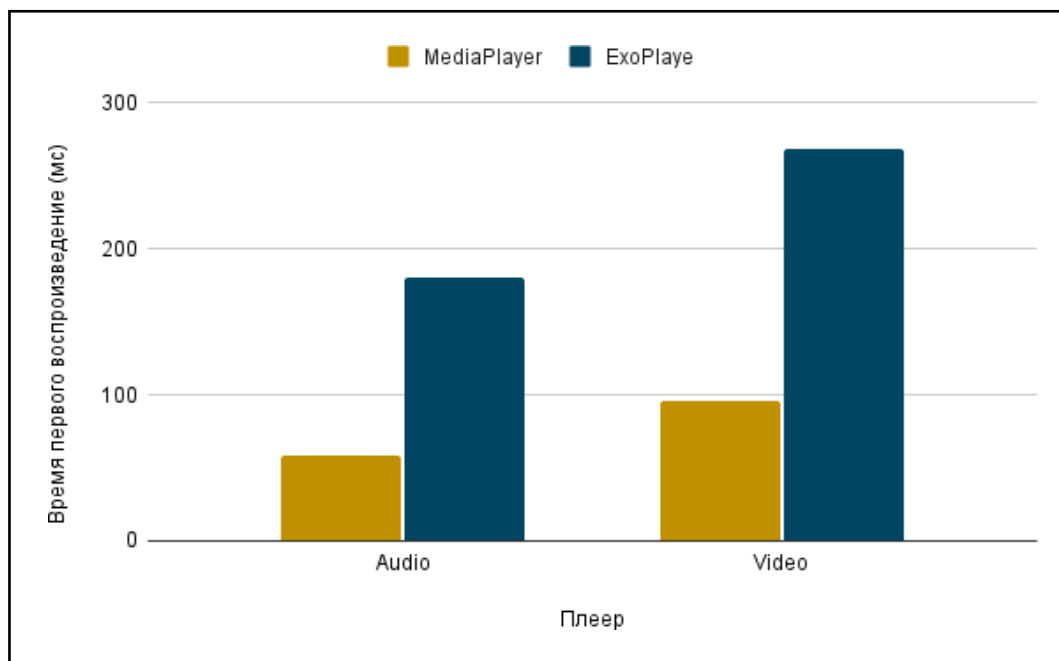


Рисунок 23 – Время первого воспроизведения контента из локального хранилища

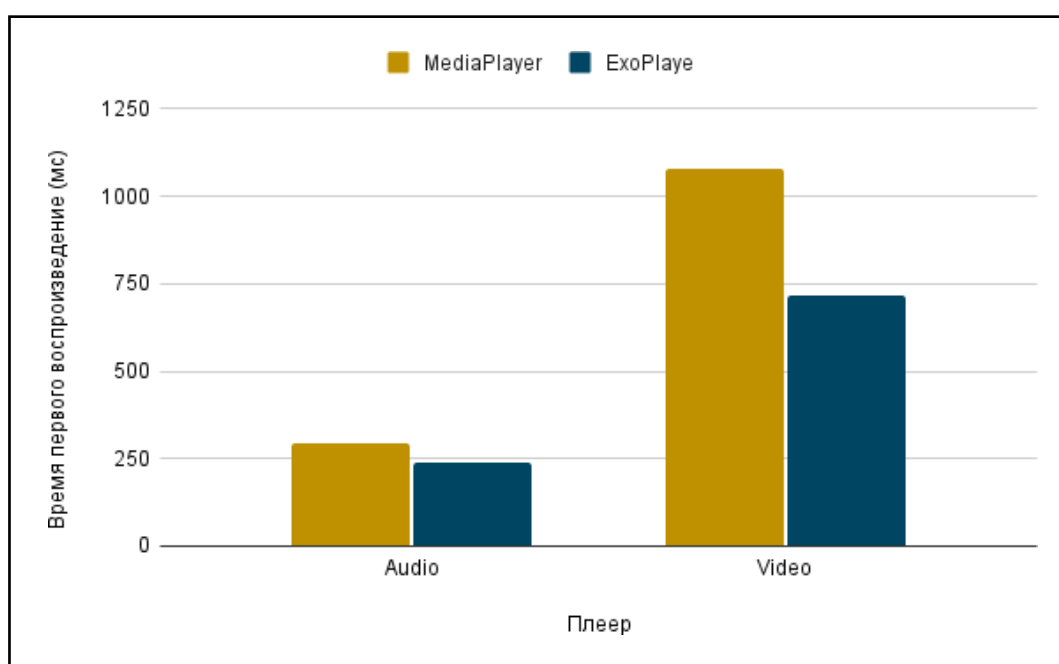


Рисунок 24 – Время первого воспроизведения контента из сети

Как видно из рисунка 23, при воспроизведении контента из локального хранилища лучший результат показала библиотека MediaPlayer. EhoPlayer также быстро воспроизводит контент, но разница в миллисекундах заметна – 121 мс в случае аудио, 173 мс в случае воспроизведения видео.

На рисунке 24 видно, что при воспроизведении контента через сеть лучший результат у библиотеки EhoPlayer (239 мс в случае аудио, 717 мс в случае видео). MediaPlayer заметно медленно осуществляет воспроизведение контента из сети. EhoPlayer в среднем, быстрее на 55 мс в случае аудио и на 363 мс в случае видео. Кроме того, MediaPlayer не дает возможности изменить размер буфера при буферизации медиафайлов из сети, а у EhoPlayer есть данный функционал. За счет изменения размера буфера можно еще оптимизировать работу EhoPlayer [10].

Таким образом, в зависимости от условий воспроизведения контента, библиотеки показывают разные результаты. В случае воспроизведения контента из локального хранилища лучше всего подходит MediaPlayer, потому что вызывается напрямую нативные функции для воспроизведения. В случае, если медиафайл воспроизводится из сети, то EhoPlayer подходит лучше всего, и с помощью данной библиотеки удобно создавать онлайн стриминговые сервисы и онлайн кинотеатры.

4.2 Сравнение поддерживаемых форматов

Перед сравнением поддерживаемых библиотекой форматов, важно отметить, что «форматы мультимедиа» разделяются на несколько уровней. От самого низкого до самого высокого:

- Формат отдельных образцов (сэмплов) мультимедиа (например, видео или аудио кадр). Например, типичный видеофайл будет храниться по крайней мере в двух примерных форматах; один для видео (например, H.264) и один для аудио (например, AAC).

- Формат контейнера, в котором хранятся образцы мультимедиа и связанные с ними метаданные. Медиафайл имеет один контейнерный формат (например, MP4), который обычно обозначается расширением файла. Для некоторых аудио форматов (например, MP3), форматы сэмпла и контейнера могут быть одинаковыми.
- Технологии адаптивной потоковой передачи, такие как DASH, SmoothStreaming и HLS. Это не форматы мультимедиа, однако все же необходимо определить, какой уровень поддержки предоставляют рассматриваемые библиотеки

MediaPlayer поддерживает большое количество медиа-форматов, таких как 3GPP, MP3, MPEG-4, MPEG-TS, AMR и т.д. EchoPlayer помимо тех форматов, которые поддерживаются в MediaPlayer, также поддерживает пару технологий адаптивной потоковой передачи контента, а именно DASH и SmoothStreaming. Хотя и обе библиотеки поддерживают HLS, в некоторых случаях оптимально использовать DASH или SmoothStreaming [12] (Таблица 5)

Таблица 5 – Сравнение HLS, DASH и SmoothStreaming

Поддерживается	HLS	DASH	Smooth Streaming
Развертывание на обычных HTTP-серверах	да	да	
Скрытые титры / субтитры	да	да	да
Несколько аудиоканалов (например, языки, комментарии и т. д.)	да	да	да
Гибкая защита контента с помощью общего шифрования (DRM)	да	да	да

Официальный международный стандарт (например, ISO/IEC MPEG)		да	
Эффективная вставка рекламы		да	
Быстрое переключение каналов		да	да
Параллельная поддержка нескольких CDN протоколов		да	
Поддержка HTML5		да	
Поддержка в HbbTV (версия 1.5)		да	
Поддержка HEVC (UHD/4K)		да	
Независимость от видеокодеков		да	
Независимость от аудиокодеков		да	
Сегменты формата базовых медиафайлов ISO		да	да
Сегменты транспортного потока MPEG-2	да	да	
Расширения формата сегмента за пределами MPEG		да	
Поддержка мультимплексированного (аудио + видео) контента	да	да	
Поддержка немультимплексированного (отдельного аудио, видео) контента	да	да	да

На данный момент, одно из лучших технологий адаптивной потоковой передачи контента является DASH. К сожалению, MediaPlayer не поддерживает DASH, ExoPlayer поддерживает.

4.3 Сравнение реакций на изменение скорости интернета

Проводилось измерение времени на изменение качества видео при увеличении и уменьшении трафика интернета на мобильном устройстве (Таблица 6)

Таблица 6 – Время и качество воспроизведения видео в MediaPlayer

Попытка	Начало воспроизведение		Изменение скорости интернета		
	разрешение видео	затраченное время	разрешение видео до	разрешение видео после	затраченное время
1	950x540	1385	950x540	640x360	2412
2	950x540	1571	950x540	640x360	2487
3	950x540	1201	950x540	640x360	2876
4	950x540	1267	640x360	950x540	3056
5	950x540	1290	640x360	950x540	3615
6	950x540	1193	640x360	950x540	3478
7	950x540	1187	640x360	950x540	4012
8	950x540	1204	950x540	640x360	1996
9	640x360	1289	950x540	640x360	2319
10	950x540	1264	950x540	640x360	2507
Ср. значение	950x540	1285	950x540	640x360	2433
			640x360	950x540	3540

Таблица 7 – Время и качество воспроизведения видео в EchoPlayer

Попытка	Начало воспроизведение	Изменение скорости интернета
---------	------------------------	------------------------------

	разрешение видео	затраченное время	разрешение видео до	разрешение видео после	затраченное время
1	950x540	912	950x540	640x360	1901
2	950x540	1095	950x540	640x360	1765
3	950x540	816	950x540	640x360	2002
4	950x540	614	640x360	950x540	2831
5	950x540	663	640x360	950x540	2615
6	950x540	812	640x360	950x540	2478
7	950x540	860	640x360	950x540	3012
8	950x540	901	950x540	640x360	1512
9	950x540	1009	950x540	640x360	1480
10	950x540	761	950x540	640x360	1526
Ср. значение	950x540	844	950x540	640x360	1698
			640x360	950x540	2437

На основе проведенного исследования сделан анализ полученных результатов. Для наглядности все данные были представлены в гистограмме (Рисунок 25)

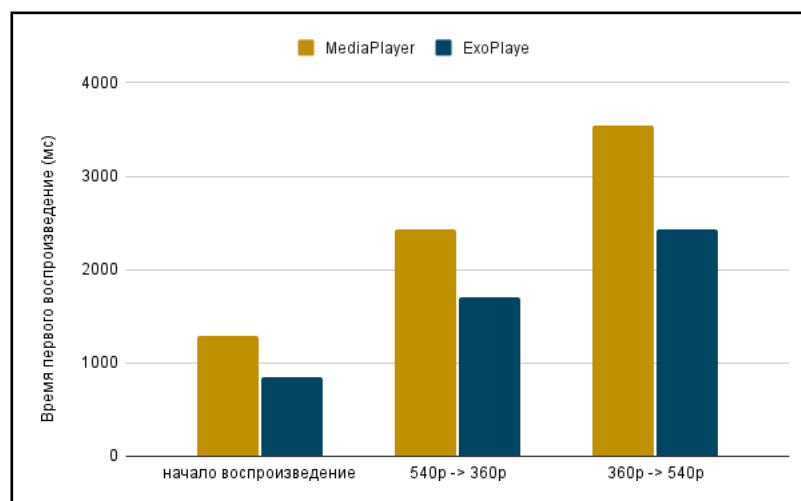


Рисунок 25 – Время затраченное на изменение качество видео

Как видно из рисунка 25, при воспроизведении контента с протоколом HLS, лучший результат показала библиотека ExoPlayer. MediaPlayer также быстро реагирует на изменение скорости интернета, но разница в миллисекундах заметна – 735 мс при уменьшении скорости интернета и 1103 мс при увеличении скорости интернета. У ExoPlayer хорошие показатели на время старта воспроизведения (на 441 мс быстрее MediaPlayer-a).

Можно сделать вывод, что и MediaPlayer, и ExoPlayer хорошо реагируют на изменение скорости интернета, но учитывая то, что ExoPlayer поддерживает DASH протокол, стоит сделать акцент на ExoPlayer

4.4 Сравнение возможности кастомизации UI плеера

Если необходимо найти минимальный изменения UI плеера, то обе библиотеки отлично справятся с данной задачей, но если есть сложные элементы управления, то лучше использовать ExoPlayer, так как данная библиотека предоставляет огромное количество возможностей для изменения элементов управления плеера, таких как предпросмотр через бегунок прогресса. Результаты сравнения по данным критериям указаны в Таблице 8

Таблица 8 - Сравнение возможности кастомизации UI плеера

Возможность	Плеер	
	MediaPlayer	ExoPlayer
Блокировка экрана	нет готового атрибута, легко реализуется	нет готового атрибута, легко реализуется

Перемотка на 10 секунд	нет готового атрибута, легко реализуется	есть готовый атрибут
Предпросмотр через бегунок прогресса	не реализуется	нет готового атрибута, легко реализуется
Перемотка быстрым нажатием	нет готового атрибута, легко реализуется	есть готовый атрибут
Кнопка изменение качество	нет готового атрибута, легко реализуется	нет готового атрибута, легко реализуется
Элемент управление громкостью	нет готового атрибута, легко реализуется	Есть готовый атрибут
Поворот экрана	нет готового атрибута, легко реализуется	нет готового атрибута, легко реализуется

4.5 Возможности оптимизации библиотек

1. Изначальное качество: (На примере протокола HLS):

При воспроизведении контента через сеть явным образом можно указать минимальное качества, чтобы быстро загрузить первый кадр. Изменение начального качества можно реализовать с помощью обеих библиотек. а 441 мс быстрее MediaPlayer-a).

2. Ускорить перемотку:

Когда пользователь перематывает видео в конкретное место, мы можем попасть не в опорный кадр, а между ними. Соответственно, всё что было с предшествующего опорного до него нужно загрузить. Можно откидывать на ближайший опорный кадр, тем самым сделав плеер еще быстрее (Рисунок 26). В случае двухчасового видео, пользователь не увидит разницы. Ускорение перемотки реализуется только с помощью библиотеки ExoPlayer.

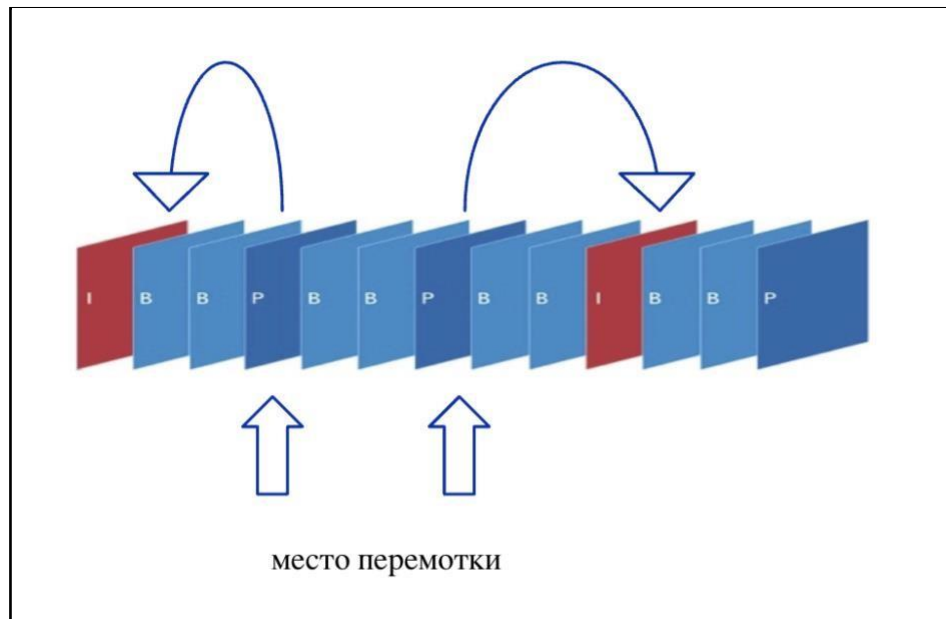


Рисунок 26 – Перемещение на ближайший кадр при перемотке

3. Запуск видео с какого-то момента:

Если видео запускается не с начала и нужно перематывать в определенную позицию, как сделано на примере приложения Кинопоиск, нужно правильно “подготовить” плеер (Рисунок 27).

```
...
player.prepare(mediaSource);
player.seekTo(resumePositionMs);
player.setPlayWhenReady(true);

...
player.seekTo(resumePositionMs);
player.prepare(mediaSource, /* resetPosition= */ false, ...);
player.setPlayWhenReady(true);
```

Рисунок 27 – Настройка плеера для оптимизации запуска видео с
определенного момента

4. Не отпускать кодек:

Если нужно сразу воспроизвести другое видео после текущего, как
сделано на примере приложения Instagram, не нужно отпускать кодек
(Рисунок 28). Данный функционал реализует только библиотека ExoPlayer.

```
void swapVideo(Uri uri)
{
player.stop();
    mediaSource = buildMediaSource(uri);
    player.prepare(mediaSource);
}
```

Рисунок 28 – Настройка плеера для оптимизации запуска видео друг за
другом

ЗАКЛЮЧЕНИЕ

Целью данной выпускной квалификационной работы являлось сравнение библиотек MediaPlayer и ExoPlayer, чтобы сократить время разработчиков на изучение документации. Библиотеки были изучены с использованием Интернет-ресурсов, что позволило провести краткий обзор каждой из них и обозначить наиболее ключевые моменты. Были определены основные критерии для сравнения библиотек и разработано мобильное приложение, в котором было измерено время первого воспроизведения, реакция на изменение скорости интернета и возможности кастомизации UI плееров.

Проанализировав с помощью данного приложения работу рассматриваемых библиотек, были получены результаты их сравнения и сделаны следующие выводы:

- MediaPlayer следует использовать, если нужно воспроизвести медиа контент из локального хранилища устройства или работы аудио контента в фоновом режиме. Например, для приложений музыкального плеера, приложения-подкаст, видеоплеер и т.д.

- ExoPlayer лучше использовать, если нужно воспроизвести медиа контент из сети или добавить собственные элементы управления плеером. При разработке стриминговых приложений, онлайн кинотеатров или социальных сетей следует использовать ExoPlayer.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. MediaPlayer [Электронный ресурс]. – URL: <https://developer.android.com/reference/android/media/MediaPlayer> (дата обращения 10.04.2022)
2. ExoPlayer [Электронный ресурс]. – URL: <https://exoplayer.dev/> (дата обращения 10.04.2022)
3. MediaPlayer - Android SDK - M.I.T. [Электронный ресурс]. – <https://stuff.mit.edu/afs/sipb/project/android/docs/reference/android/media/MediaPlayer.html> (дата обращения 14.04.2022)
4. Android - MediaPlayer - CoderLessons [Электронный ресурс]. – 2019 – URL: https://coderlessons.com/tutorials/mobilenaia_razrabotka/uchitsia_android/android-mediaplayer (дата обращения 15.04.2022)
5. Руководство Android VideoView [Электронный ресурс]. – URL: <https://betacode.net/10487/android-videoview> (дата обращения 15.04.2022)
6. Android: VideoView [Электронный ресурс]. – URL: <http://developer.alexanderklimov.ru/android/views/surfaceview.php> (дата обращения 21.04.2022)
7. ExoPlayer – StyledPlayerView [Электронный ресурс]. – URL: <https://exoplayer.dev/doc/reference/com/google/android/exoplayer2/ui/StyledPlayerView.html> (дата обращения 21.04.2022)
8. ExoPlayer – Events [Электронный ресурс]. – URL: <https://exoplayer.dev/listening-to-player-events.html> (дата обращения 21.04.2022)
9. Introduction to HTTP Live Streaming: HLS on Android and More [Электронный ресурс]. – URL: <https://www.toptal.com/apple/introduction-to-http-live-streaming-hls> (дата обращения 26.04.2022)
10. Get 4 times better re-buffering with drip-feeding technique in ExoPlayer on Android [Электронный ресурс]. – URL: <https://medium.com/@filipluch/how->

to-improve-buffering-by-4-times-with-drip-feeding-technique-in-exoplayer-on-android-b59eb0c4d9cc (дата обращения 26.04.2022)

11. Customize ExoPlayer Overlay to look like Youtube Player [Электронный ресурс]. – URL: <https://levelup.gitconnected.com/customize-exoplayer-overlay-look-like-youtube-player-14fdd6d4583d> (дата обращения 26.04.2022)
12. HLS vs. MPEG-DASH: A Live Streaming Protocol Comparison for 2021 [Электронный ресурс]. – URL: <https://www.dacast.com/blog/mpeg-dash-vs-hls-what-you-should-know/> (дата обращения 29.04.2022)

ПРИЛОЖЕНИЕ А Реализация Navigation Drawer

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.drawerlayout.widget.DrawerLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:openDrawer="start">

    <include
        android:id="@+id/app_bar_main"
        layout="@layout/app_bar_main"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <com.google.android.material.navigation.NavigationView
        android:id="@+id/nav_view"
        android:layout_width="wrap_content"
        android:background="@color/main"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:fitsSystemWindows="true"
        app:itemTextColor="@color/black"
        app:headerLayout="@layout/nav_header_main"
        app:itemBackground="@drawable/menu_item_bg"
        app:itemTextAppearance="@style/TextMenu"
        app:menu="@menu/activity_main_drawer" />
</androidx.drawerlayout.widget.DrawerLayout>
```

app_bar_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
```

```

tools:context=".MainActivity">

<com.google.android.material.appbar.AppBarLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:theme="@style/Theme.PlayerComparison.AppBarOverlay">

    <androidx.appcompat.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        app:titleTextColor="@color/black"
        app:titleTextAppearance="@style/TextToolBar"
        android:theme="@style/ToolBar"
        android:layout_height="?attr/actionBarSize"
        android:background="?attr/colorPrimary"
        app:popupTheme="@style/Theme.PlayerComparison.PopupOverlay"
    />

</com.google.android.material.appbar.AppBarLayout>

<include layout="@layout/content_main" />

</androidx.coordinatorlayout.widget.CoordinatorLayout>

```

ПРИЛОЖЕНИЕ Б Реализация компонентов RadioButton

```
<RadioGroup
    android:id="@+id/group_content"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@id/media_place">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Тип контента:"/>

    <RadioButton
        android:id="@+id/radio_audio"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="аудио"
        />

    <RadioButton
        android:id="@+id/radio_video"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="видео"
        android:checked="true"/>
</RadioGroup>

<RadioGroup
    android:id="@+id/group_player"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@id/group_content">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Плеер:"/>
```

```

<RadioButton
    android:id="@+id/radio_exo"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="exoplayer"/>

<RadioButton
    android:id="@+id/radio_mp"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:checked="true"
    android:text="mediaplayer"/>
</RadioGroup>

<RadioGroup
    android:id="@+id/group_location"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@id/group_player">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Расположения:"/>

    <RadioButton
        android:id="@+id/radio_local"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:buttonTint="@color/main"
        android:text="локальный"/>

    <RadioButton
        android:id="@+id/radio_inet"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="интернет"/>
</RadioGroup>

```

ПРИЛОЖЕНИЕ В Реализация предпросмотра контента через бегунок прогресса.

```
private const val MAX_LINES = 6
private const val MAX_COLUMNS = 10
private const val THUMBNAILS_EACH = 1000 // milliseconds
private const val ONE_MINUTE = 60000 // one minute in millisecond

class GlideThumbnailTransformation(position: Long) :
    BitmapTransformation() {

    private val x: Int
    private val y: Int

    init {
        val square = position.rem(ONE_MINUTE).toInt()/THUMBNAILS_EACH
        y = square / MAX_COLUMNS
        x = square % MAX_COLUMNS
    }

    override fun transform(pool: BitmapPool, toTransform: Bitmap,
        outWidth: Int, outHeight: Int): Bitmap {
        val width = toTransform.width / MAX_COLUMNS
        val height = toTransform.height / MAX_LINES
        return Bitmap.createBitmap(toTransform, x * width, y * height,
            width, height)
    }

    override fun updateDiskCacheKey(messageDigest: MessageDigest) {
        val data: ByteArray =
            ByteBuffer.allocate(8).putInt(x).putInt(y).array()
        messageDigest.update(data)
    }

    override fun hashCode(): Int {
        return (x.toString() + y.toString()).hashCode()
    }

    override fun equals(other: Any?): Boolean {
        if (other !is GlideThumbnailTransformation) {
            return false
        }
        return other.x == x && other.y == y
    }
}
```