

Колледж космического машиностроения и технологий

**КУРСОВОЙ ПРОЕКТ**

**По МДК.01.02 «Прикладное программирование»**

**Тема: «Разработка музыкального андроид-плеера»**

Выполнил студент

Константинович А.

Группа П1-18

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Подпись)

\_\_\_\_\_\_\_\_\_\_\_ (Дата сдачи работы)

Принял преподаватель

Гусятинер Л.Б.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Оценка)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Подпись)

**Королёв 2020 г.**

# 

# Введение

Данный курсовой проект заключается в разработке приложения под операционную систему Android. Приложение представляет собой аудио проигрыватель, который воспроизводит аудиофайлы из памяти устройства. Разработка программы будет происходить на языке программирования Kotlin.

В первой части будут рассмотрены предметная область и существующие продукты по данной теме.

Во второй части будут описаны инструменты разработки программы и модули программы.

В третьей части описаны взаимодействие пользователя с программой и сообщения оператору.

В заключительной части будут приведены общие выводы по проекту.

# Теоретическая часть

## Описание предметной области

**Медиаплеер** (также проигрыватель мультимедиа или мультимедиа-проигрыватель) — компьютерная программа, предназначенная для вос-произведения файлов мультимедиа-контента.

Одна разновидность медиаплееров предназначена для воспроизведе-ния только аудио- или же видеофайлов, и которые называются, соответ-ственно, — аудиоплеер и видеоплеер. В основном все видеоплееры делятся на несколько типов: Простые (для работы на слабых ПК), Универсальные и Специализированные (специфические и профессиональные программы для решения уникальных задач). Примерно по такому же принципу делят-ся и аудиоплееры. Разработчики таких плееров стремятся сделать их как можно более удобными для воспроизведения соответствующих форматов.

Другая разновидность программ-медиаплееров поддерживают как аудио так и видео (включая множество медиаконтейнеров) и называется мультимедиа-центры.

Большинство современных операционных систем по умолчанию со-держат в своём составе медиаплееры: например,Windows — Windows Media Player, Mac OS X — QuickTime Player (для воспроизведения видео в формате QuickTime) и iTunes (для некоторых других форматов), Linux — Amarok, Rhythmbox или иные (в зависимости от дистрибутива).

Медиаплееры есть в телевизорах Smart TV и ресиверах цифрового телевидения.

Разрабатываемый проигрыватель поддерживает файлы в формате MP3.

**MP3** — это разработанный командой MPEG формат файла для хра-нения аудиоинформации. MP3 является одним из самых распространён-ных и популярных форматов цифрового кодирования звуковой информа-ции. Он широко используется в файлообменных сетях для оценочного ска-чивания музыкальных произведений. Формат может проигрываться прак-тически во всех популярных операционных системах, на большинстве пор-тативных аудиоплееров, а также поддерживается всеми современными мо-делями музыкальных центров и DVD-плееров.

Как и формат JPEG, MP3 использует спектральные отсечения, со-гласно психоакустической модели. Звуковой сигнал разбивается на равные по продолжительности отрезки, каждый из которых после обработки упа-ковывается в свой фрейм (кадр). Разложение в спектр требует непрерывно-сти входного сигнала, в связи с этим для расчётов используется также предыдущий и следующий фрейм. В звуковом сигнале есть гармоники с меньшей амплитудой и гармоники, лежащие вблизи более интенсивных — такие гармоники отсекаются, так как среднестатистическое человеческое ухо не всегда сможет определить присутствие либо отсутствие таких гар-моник. Такая особенность слуха называется эффектом маскировки. Также возможна замена двух и более близлежащих пиков одним усреднённым (что, как правило, и приводит к искажению звука). Критерий отсечения определяется требованием к выходному потоку. Поскольку весь спектр ак-туален, высокочастотные гармоники не отсекаются, как в JPEG, а только выборочно удаляются, чтобы уменьшить поток информации за счёт раз-режения спектра. После спектральной «зачистки» применяются математи-ческие методы сжатия и упаковка во фреймы. Каждый фрейм может иметь несколько контейнеров, что позволяет хранить информацию о нескольких потоках (левый и правый канал либо центральный канал и разница кана-лов). Степень сжатия можно варьировать, в том числе в пределах одного фрейма. Интервал возможных значений битрейта составляет 8—320 кбит/c.

MP3-файл состоит из нескольких фрагментов (фреймов) MP3, кото-рые, в свою очередь, состоят из заголовка и блока данных. Такая последо-вательность фрагментов называется элементарным потоком. Фрагменты не являются независимыми элементами («резервуар байт»), и поэтому не мо-гут быть извлечены произвольно. Блок данных MP3-файла содержит сжа-тую аудиоинформацию в виде частот и амплитуд.

## Описание существующих разработок

На данный момент существует множество уже разработанных приложений на данную тематику. Практически все из них предоставляют похожий функционал:

* Поддержка многих аудиоформатов;
* 8-ми полосный эквалайзер с возможностью автоматической подстройки под жанр трека;
* Автоматическое определение кодировки данных в тегах;
* Поддержка обложек альбома;
* Возможность выборочного добавления файлов / папок в плейлист;
* Возможность быстрого добавления всей музыки с SD-карт;
* Возможность повтора плейлиста / повтора трека / проигрывания плей-листа без повтора;
* Последовательное проигрывание треков / проигрывание в случайном порядке;
* Управление из «шторки» (для Android v3.0 и новее) и с гарнитуры;
* Воспроизведение музыки из файловых менеджеров.

Примеры приложений:

* AIMP

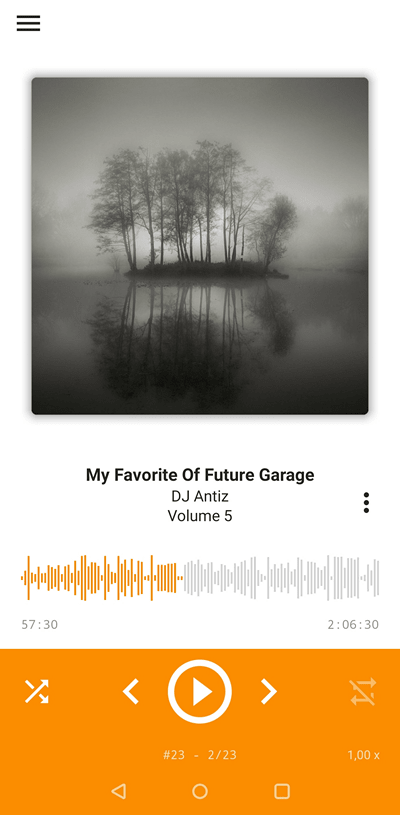


Рисунок 1. Аудиоплеер AIMP.

* Audify Music Player



Рисунок 2. Аудиоплеер Audify Music Player.

* BlackPlayer



Рисунок 3. Аудиоплеер BlackPlayer.

* jetAudio

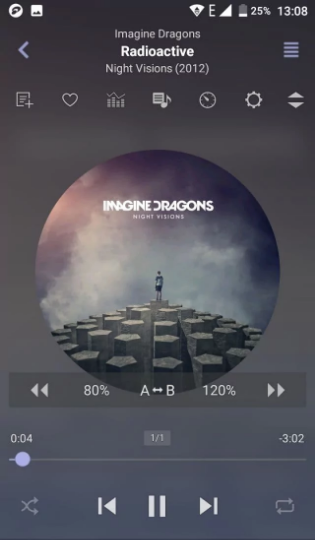


Рисунок 4. Аудиоплеер jetAudio.

# Проектная часть

## Диаграмма прецедентов

В данном разделе содержится диаграмма прецедентов для приложения, на которой показаны возможные функциональные отношения. (рис. 5)



Рисунок 5. Диаграмма прецедентов приложения.

## Выбор инструментов

### Язык программирования

В разработке приложений на Android используются в основном либо Java, либо Kotlin. Данный курсовой проект написан на Kotlin.

**Kotlin** - язык программирования, разработанный компанией JetBrains, работающий на платформе Java. Он использует JDK, как и сама Java, но имеет другой синтаксис. Синтаксис языка использует элементы из Паскаля, TypeScript, Haxe, PL/SQL, F#, Go и Scala, C++, Java, C#, Rust и D. При объявлении переменных и параметров типы данных указываются после названия (разделитель - двоеточие). Точка с запятой как разделитель операторов так же необязательна. Кроме объектно-ориентированного подхода, Kotlin также поддерживает процедурный стиль с использованием функций. Как и в Си, C++ и D, точка входа в программу – функция main, принимающая массив параметров командной строки. Также поддерживается вывод типов.

В сравнении класса Java с эквивалентным классом Kotlin демонстрирует лаконичность кода Kotlin. Для той же операции, что выполняется в классе Java, класс Kotlin требует меньше кода. Одно из основных различий между Java и Kotlin заключается в том, что в последнем нет условий для проверяемых исключений (checked exception). Следовательно, нет необходимости отлавливать или объявлять какие-либо исключения. Несомненным преимуществом является полная совместимость с Java, в том числе и обратная. Все библиотеки для Java будут работать на Kotlin и наоборот. Также Kotlin позволяет разработчикам расширять класс новыми функциями с помощью функций расширения. Эти функции недоступны в Java. В отличие от Java, в Kotlin все типы по умолчанию являются не-nullable. Если разработчики попытаются присвоить или вернуть значение null в коде Kotlin, во время компиляции произойдет сбой.

Kotlin помог разработчикам писать программы с меньшим количеством кода. Помимо всего того, что есть в Java, он добавляют вещи из мира функционального программирования. Это значительно облегчает написание кода - делает его короче и выразительнее.

### Среда разработки программного обеспечения

**Android Studio** — интегрированная среда разработки (IDE) для работы с платформой Android.

Android Studio, основанная на программном обеспечении IntelliJ IDEA от компании JetBrains, — официальное средство разработки Android приложений. Данная среда разработки доступна для Windows, macOS и GNU/Linux. 17 мая 2017, на ежегодной конференции Google I/O, Google анонсировал поддержку языка Kotlin, используемого в Android Studio, как официального языка программирования для платформы Android в дополнение к Java и С++.

### База данных

В разработке приложений на Android в основном в качестве базы данных используется **SQLite**. Причины использования:

1. **Минимальные затраты ресурсов.** Для работы большинства систем управления базами данных необходим специальный процесс сервера базы данных. SQLite обходится без сервера: база данных SQLite представляет собой обычный файл. Когда БД не используется, она не расходует процессорное время. Это особенно важно на мобильных устройствах, чтобы избежать разрядки аккумулятора.
2. **Оптимизация для одного пользователя.** С базой данных взаимодействует только наше приложение. Поэтому можно обойтись без идентификации с именем пользователя и паролем.
3. **Надежность и быстрота.** Базы данных SQLite поддерживают транзакции баз данных. Кроме того, операции чтения и записи данных реализуются на оптимизированном коде С.

## Проектирование сценария

В данном разделе приведен сценарий использования программы пользователем (Рисунок 6).



Рисунок 6. Сценарий использования приложения

При запуске приложения открывается экран управления текущим треком. По умолчанию текущего трека нет, поэтому при запуске плеера запустится первая песня из плейлиста. Также пользователь может выбрать песню из экрана управления плейлистом. Чтобы добавить новые песни следует перейти в экран добавления новых песен.

## Диаграммы классов

В данном разделе находятся диаграммы классов приложения. (Рисунки 7, 8, 9)

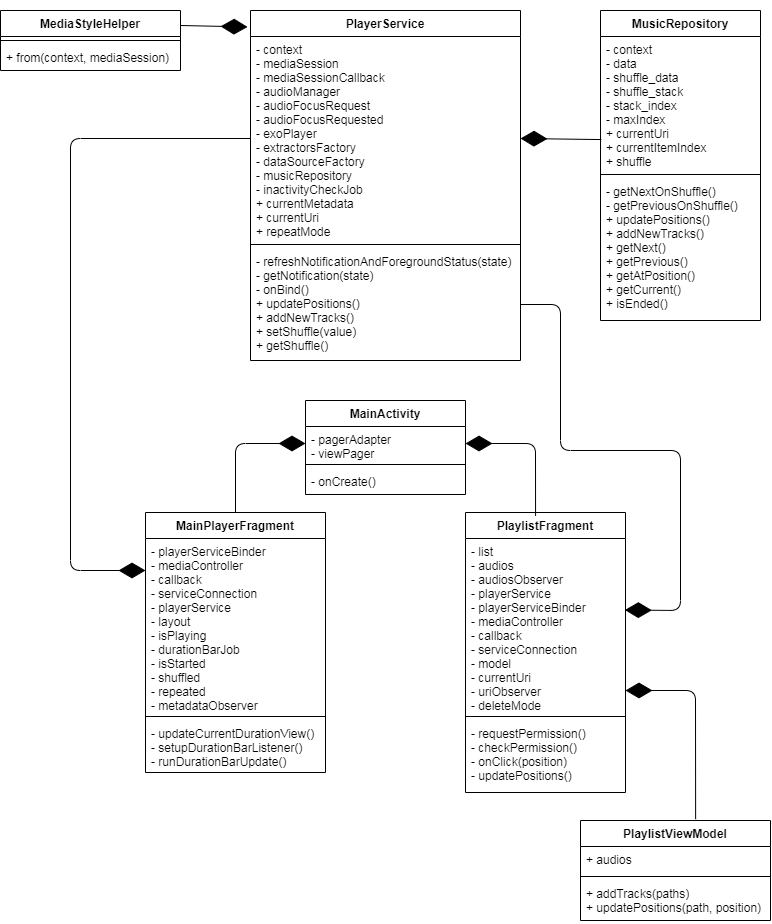


Рисунок 7. Диаграмма основных классов приложения

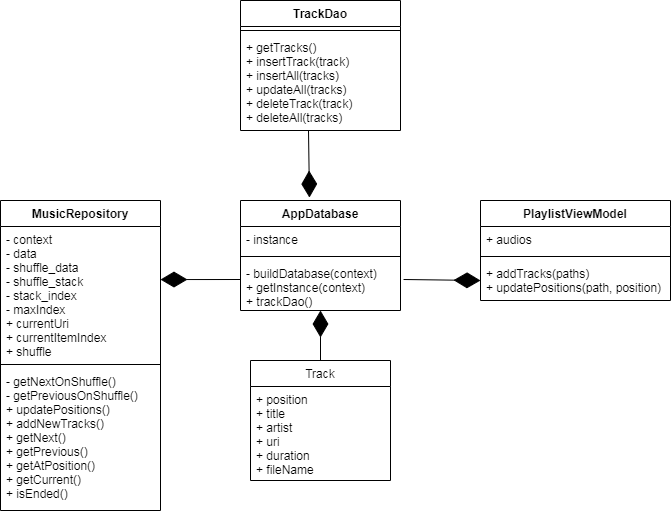
****

Рисунок 8. Диаграмма классов, связанных с БД.

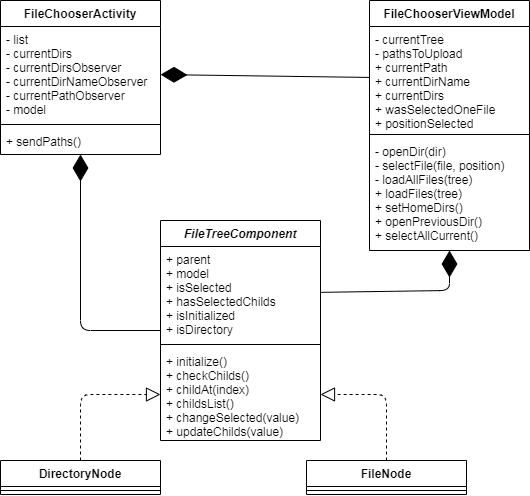


Рисунок 9. Диаграмма классов экрана добавления файлов.

## Описание главного модуля.

Главный модуль состоит из класса MainActivity. Он выполняется при запуске приложения. В данном модуле создаются основные экраны приложения и отображаются пользователю.

**Листинг 1. Главный модуль.**

package com.kmem.myplayer.ui.activities  
  
import android.content.Intent  
import android.os.Bundle  
import android.view.View  
import androidx.appcompat.app.AppCompatActivity  
import androidx.fragment.app.Fragment  
import androidx.fragment.app.FragmentManager  
import androidx.fragment.app.FragmentPagerAdapter  
import androidx.viewpager.widget.ViewPager  
import com.kmem.myplayer.ui.fragments.MainPlayerFragment  
import com.kmem.myplayer.ui.fragments.PlaylistFragment  
import com.kmem.myplayer.R  
import com.kmem.myplayer.service.PlayerService  
  
*/\*\*  
 \* Главная активность приложения. Она запускается при открытии приложения.  
 \* Отвечает за запуск остальных экранов приложения.  
 \*/*class MainActivity : AppCompatActivity() {  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_main*)  
 val pagerAdapter = SectionsPagerAdapter(*supportFragmentManager*)  
 val viewPager = findViewById<View>(R.id.*pager*) as ViewPager  
 viewPager.*adapter* = pagerAdapter

// запуск сервиса  
 startService(Intent(*baseContext*, PlayerService::class.*java*))  
 }  
  
 private inner class SectionsPagerAdapter(fm: FragmentManager?) :  
 FragmentPagerAdapter(fm!!, *BEHAVIOR\_RESUME\_ONLY\_CURRENT\_FRAGMENT*) {  
 override fun getCount(): Int {  
 return 2  
 }  
  
 override fun getItem(position: Int): Fragment {  
 when (position) { // Создание отсальных экранов  
 0 -> return MainPlayerFragment()  
 1 -> return PlaylistFragment()  
 }  
 return MainPlayerFragment()  
 }  
  
 override fun getPageTitle(position: Int): CharSequence? {  
 when (position) {  
 0 -> return "main"  
 1 -> return "Playlist"  
 }  
 return null  
 }  
 }  
}

## Описание спецификаций к модулям

Всего разработано 5 модулей:

* Экран управления плеером и текущим треком (MainPlayerFragment)
* Экран управления плейлистом (PlaylistFragment)
* Экран добавления файлов в плейлист (FileChooserActivity)
* Сервис для проигрывания музыки (PlayerService)
* Репозиторий получения данных из БД для сервиса (MusicRepository)

Взаимодействие между модулями можно определить по диаграммам классов из раздела 2.4.

## Описание модулей

**Экран управления плеером и текущим треком** – на данном экране отображается информация о текущей проигрываемой песне, а также находятся кнопки для управления плеером (Приложение 1). Сведения о текущем треке, которые находятся на экране: обложка, автор, название, текущая длительность. Для управления плеером в нижней части экрана находятся следующие кнопки:

* Управление перемешиванием
* Управление повторением
* Предыдущая песня
* Следующая песня
* Воспроизведение/пауза текущей песни

**Экран управления плейлистом** – содержит информацию о проигрываемом списке песен. Позволяет добавлять и удалять песни. При нажатии на элемент списка (песню) запускает проигрывание песни. В нижней части экрана содержится панель управления с двумя кнопками: кнопка добавления и кнопка удаления. При нажатии на «плюс» (кнопка добавления) открывается новый экран для выбора песен из памяти устройства. При нажатии на «минус» (кнопка удаления) возле каждой песни появится флажок для выбора данной песни (Приложение 2).

**Экран добавления файлов** – внутренний обозреватель файлов. Позволяет проходить по папкам и выбирать файлы для добавления. Возле каждого файла и директории находится флажок для выбора. В верхней части экрана находится панель управления с кнопками для перехода к домашней и предыдущей директориям. Для загрузки файлов в нижней части экрана есть кнопка для завершения выбора файлов. (Приложение 3)

Рассмотрим функцию загрузки выбранных файлов, а также её блок-схему:

**Листинг 2. Функция загрузки выбранных файлов.**

// Загружает файлы, выбранные пользоваетелем

fun loadFiles(tree: FileChooserActivity.FileTreeComponent? = null) : ArrayList<String> {

var tmpTree = tree

if(tree == null) {

pathsToUpload.clear() // Список с выбранными путями

// Получаем главное дерево

while(currentTree?.parent != null)

currentTree = currentTree?.parent

tmpTree = currentTree

}

// Проходим по всем потомкам

for (child in tmpTree?.childsList()!!) {

if (child.isSelected) {

if (child.isDirectory)

loadAllFiles(child)

else

pathsToUpload.add(child.model?.file?.absolutePath!!)

} else if (child.hasSelectedChilds) {

loadFiles(child)

}

}

return pathsToUpload

}

// Загружает все файлы из директории

private fun loadAllFiles(tree: FileChooserActivity.FileTreeComponent?) {

for (child in tree?.childsList()!!) {

if (child.isDirectory)

loadAllFiles(child)

else

pathsToUpload.add(child.model?.file?.absolutePath!!)

}

}

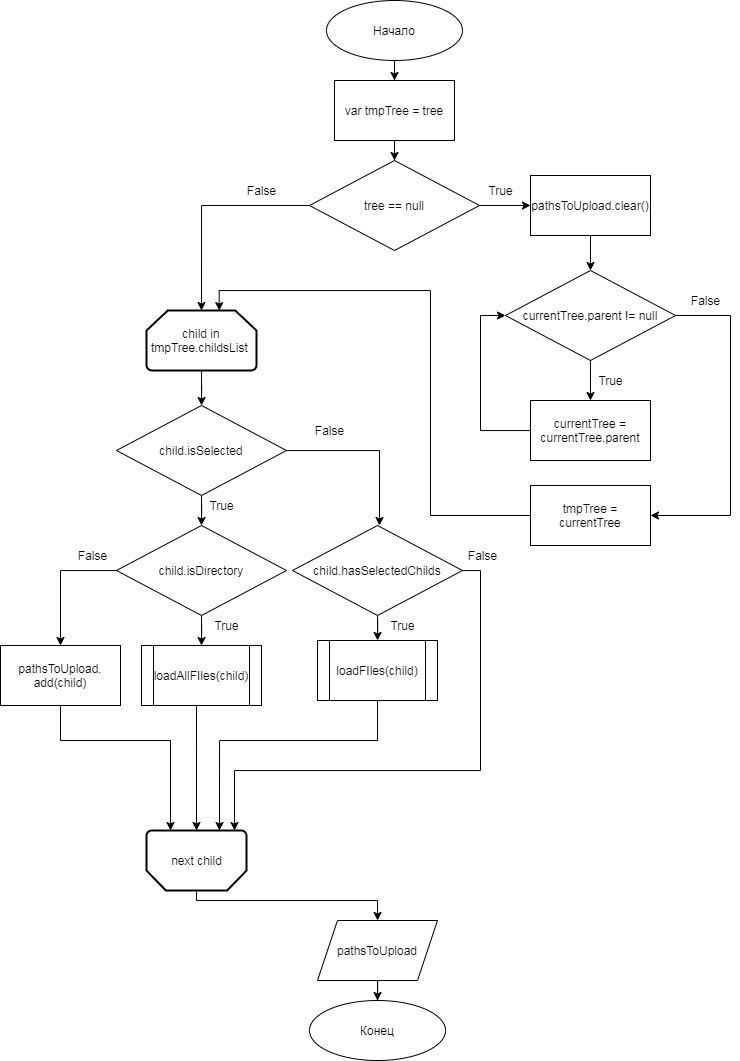


Рисунок 10. Блок схема функции loadFIles.

**Сервис для проигрывания музыки –** модуль, который непосредственно играет музыку и управляет плеером. Остальные модули получают от него информацию и через него взаимодействуют с плеером. Сервис также обрабатывает события от системы (например, поставить музыку на паузу при звонке). (Приложение 4)

**Репозиторий получения данных из БД для сервиса –** модуль, который отвечает за получение песен из БД. Сервис обращается к репозиторию для получения текущей/следующей/предыдущей песни. Также сервис может настраивать репозиторий. (например, перемешать песни). (Приложение 5)

## Описание тестовых наборов модулей

В некоторых модулях производится проверка данных на корректность, чтобы сообщить пользователю о проблеме и предотвратить аварийное завершение программы.

**Тест 1. Проверка на пустой плейлист.**

Если запустить плеер с пустым плейлистом, то в нижней части экрана появится соответствующее сообщение. (Рисунок 11)

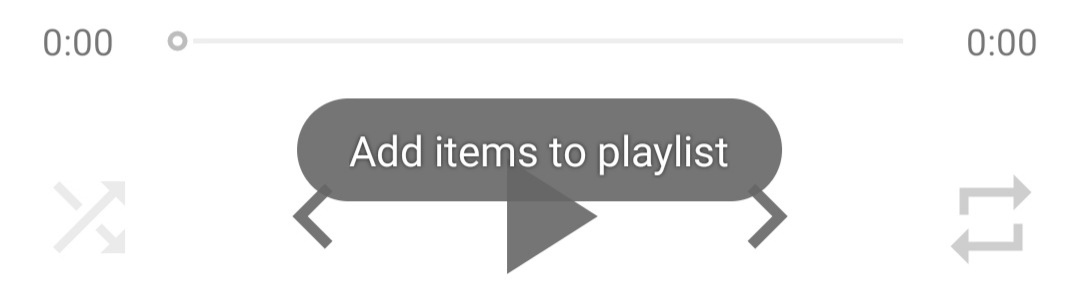


Рисунок 11. Ошибка при пустой плейлисте

**Тест 2. Проверка разрешения доступа к файлам.**

При запуске экрана выбора файлов, если пользователь до этого не давал разрешение доступа к файлам, приложение запросит соответствующее разрешение. (Рисунок 12)

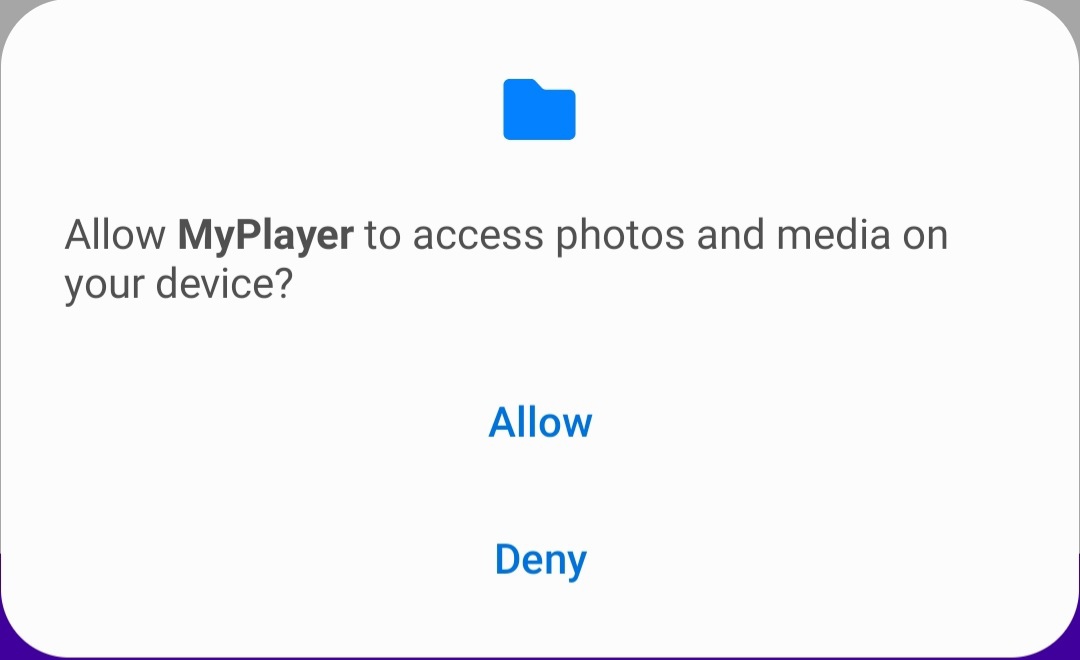


Рисунок 12. Запрос разрешения к файлам

Если пользователь отказал в доступе, то появится сообщение о необходимости разрешения для добавления файлов в плейлист. (Рисунок 13)

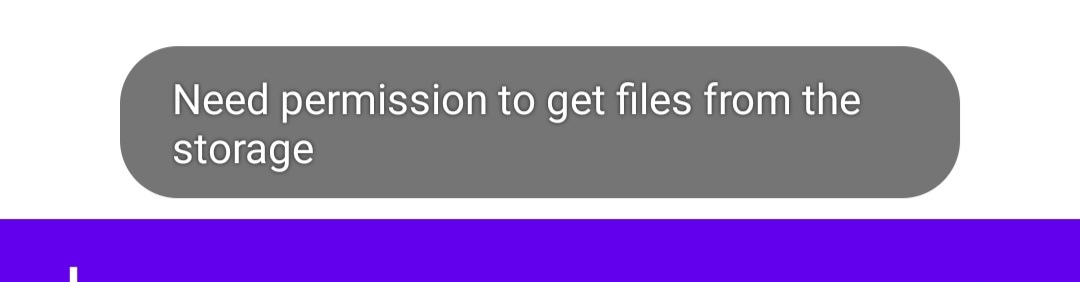


Рисунок 13. Ошибка получения разрешения

## Описание применения средств отладки

В ходе написания курсового проекта при попытке запустить программу были получены ошибки (Рисунок 14):

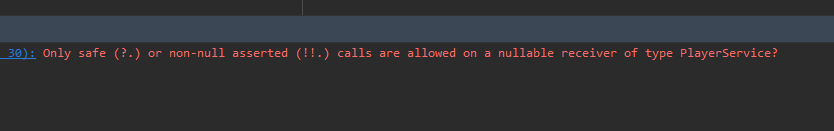


Рисунок 14. Сообщение об ошибке

При проверке кода были исправлены найденные ошибки, в результате при запуске программы ошибок не было (Рисунок 15):

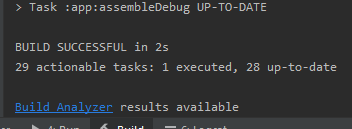


Рисунок 15. Успешная сборка

## Анализ оптимальности использования памяти и быстродействия

При анализе оптимальности использования памяти были обнаружены неиспользуемые файлы, которые увеличивали размер приложения (Рисунок 16):

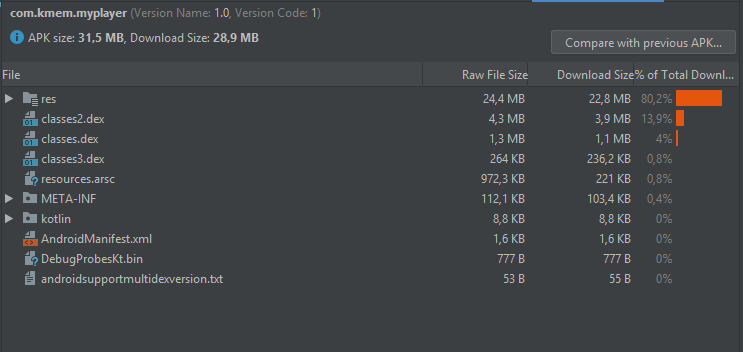


Рисунок 16. Размер приложения до оптимизации

После удаления ненужных файлов размер приложения уменьшился в 4 раза (Рисунок 17):

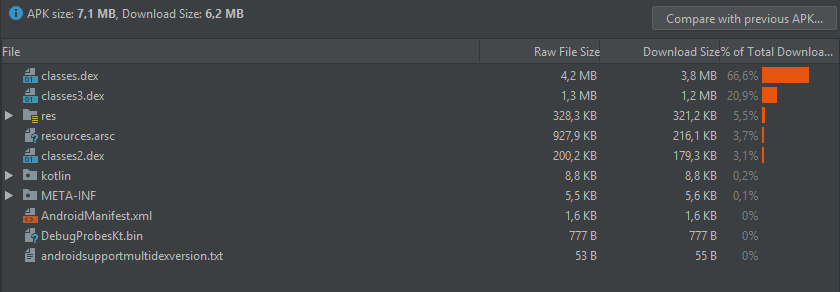


Рисунок 17. Размер после оптимизации

Также в класс DirectoryNode был добавлен метод для отложенной инициализации. Таким образом при создании дерева загружались не все узлы, а только необходимые.

**Листинг 3. Функция инициализации DirectoryNode.**

override fun initialize() {

if (isInitialized)

return

var node: FileTreeComponent // temporary variable

model?.file?.listFiles()?.forEach {

if (isAcceptable(it))

childModels.add(FileModel(it.name, it))

}

for (childModel in childModels) {

if (childModel.file.isDirectory) {

node = DirectoryNode(this, childModel)

} else {

node = FileNode(this, childModel)

}

node.isSelected = this.isSelected

childs.add(node)

}

isInitialized = true

}

# Эксплуатационная часть

## Руководство оператора

### Назначение программы

Разработанное приложение позволяет прослушивать аудио файлы в формате MP3 с памяти устройства Android. Добавляемые файлы сохраняются в БД приложения и данный список песен можно изменять.

### Условия выполнения программы

Операционная система: Android 4.4 и выше.

Оперативная память: 512 Мб.

Доступного места на диске: 50 Мб.

### Выполнение программы

При запуске программы откроется экран с информацией о текущем треке:

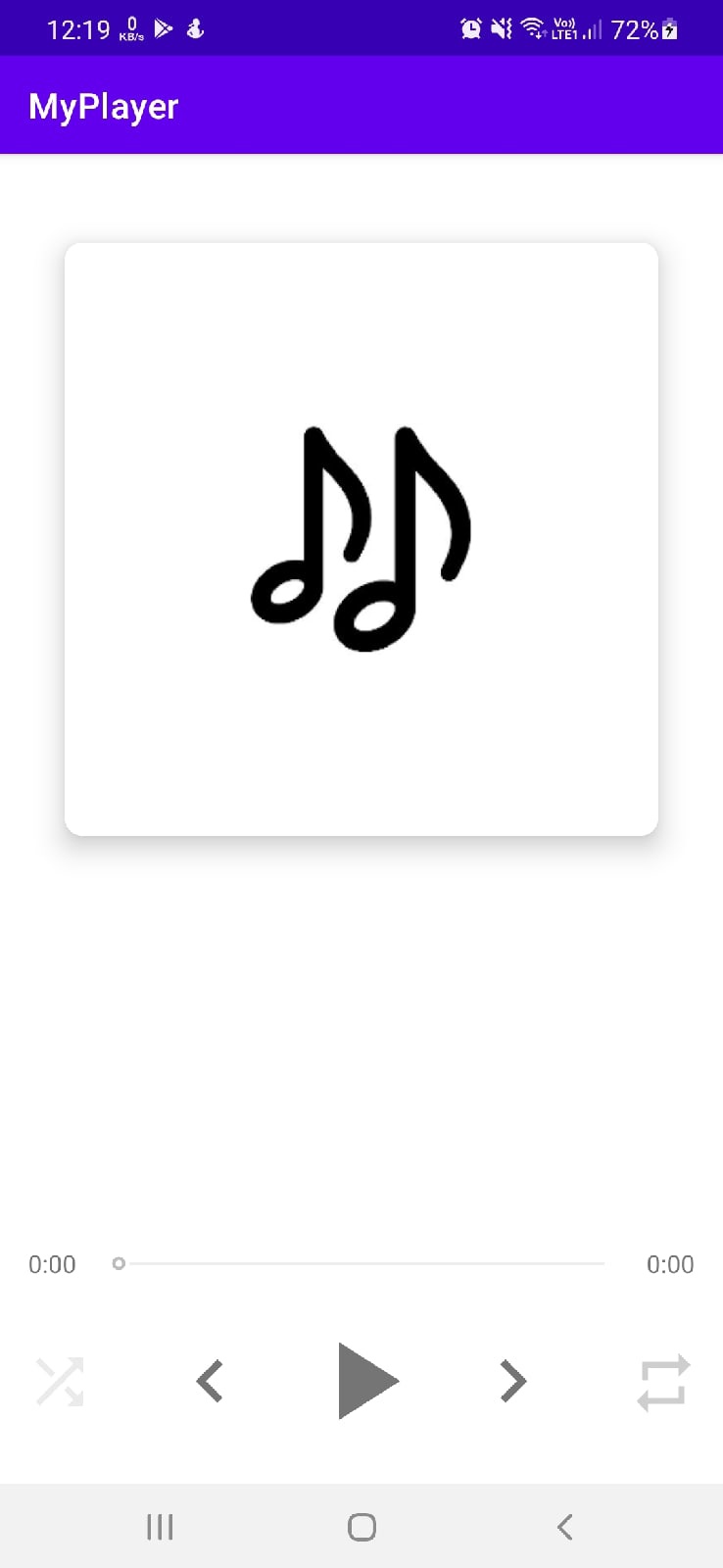


Рисунок 18. Запуск программы

Чтобы запустить плеер нужно добавить песни в плейлист. Для этого нужно провести влево для открытия экрана с плейлистом:



Рисунок 19. Экран с пустым плейлистом

Далее, чтобы добавить песни, нужно нажать на кнопку добавления в левом нижнем углу. При нажатии приложение запросит доступ к памяти и откроет экран добавления файлов:



Рисунок 20. Экран добавления файлов

При нажатии на директорию откроются файлы из этой дирктории. Чтобы выбрать файлы или директории рядом с ними находятся флажки:

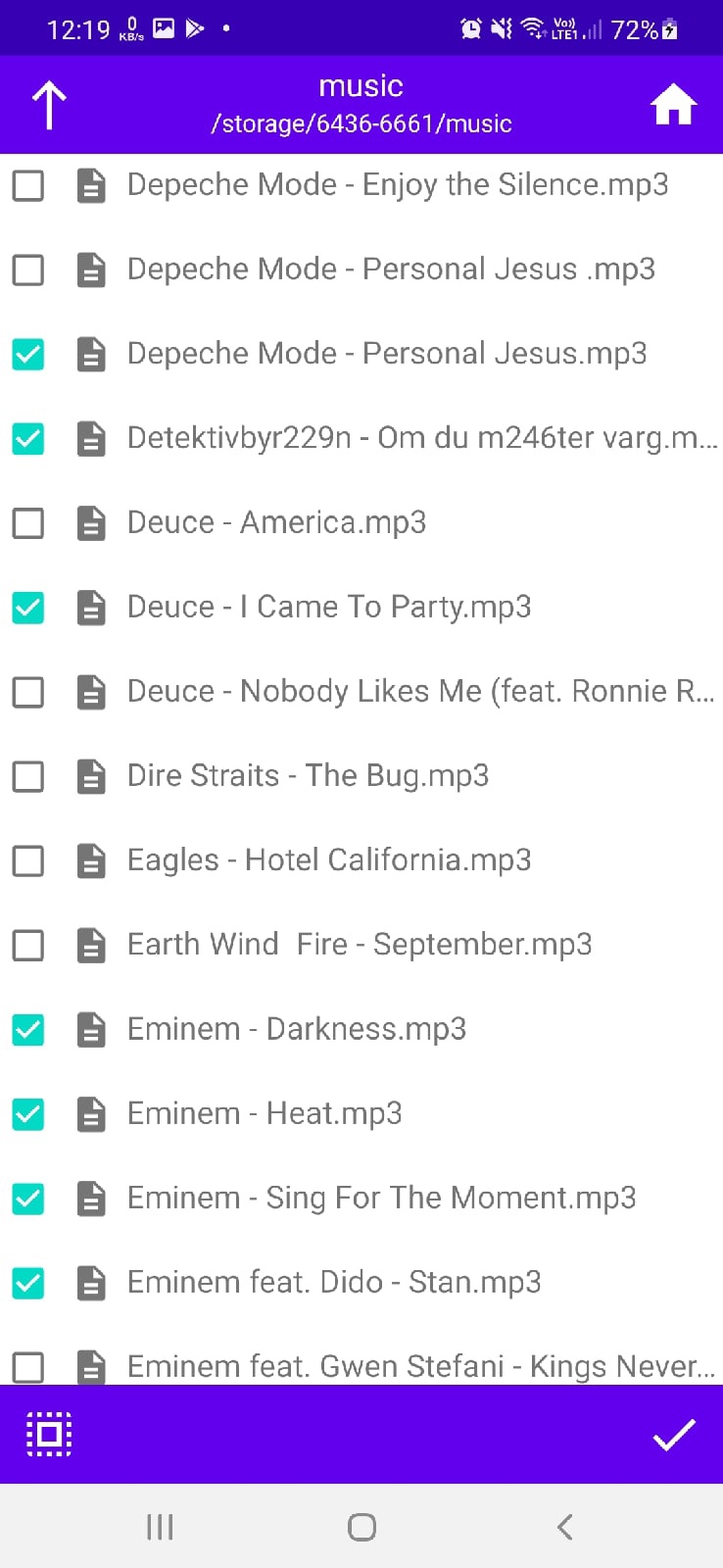


Рисунок 21. Выбор файлов

После выбора нужных файлов нужно нажать на кнопку подтверждения в правом нижнем углу. Откроется экран плейлиста с выбранными песнями:

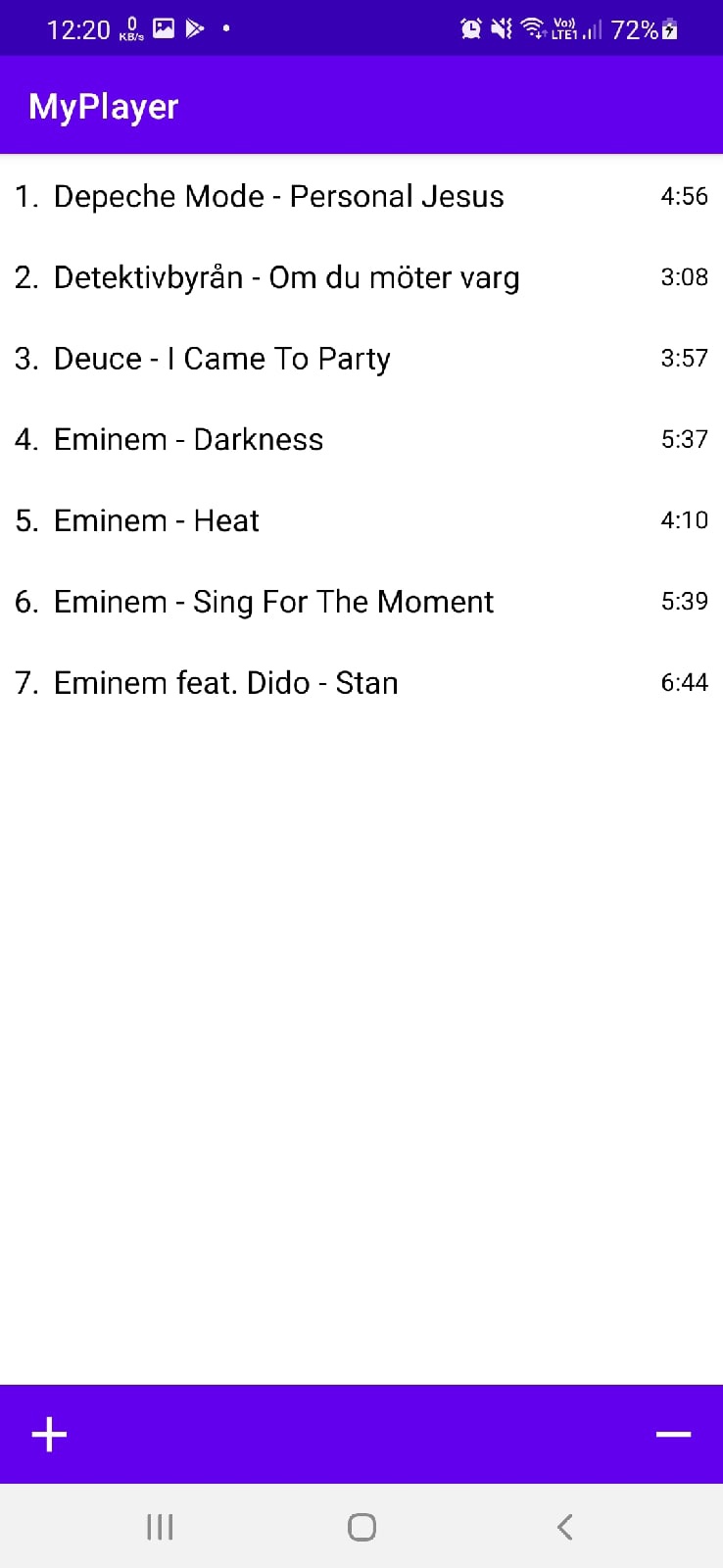


Рисунок 22. Плейлист с добавленными песнями

Чтобы начать проигрывание трека нужно нажать на элемент в списке песен. Он станет выделенным и экран текущего трека изменится:

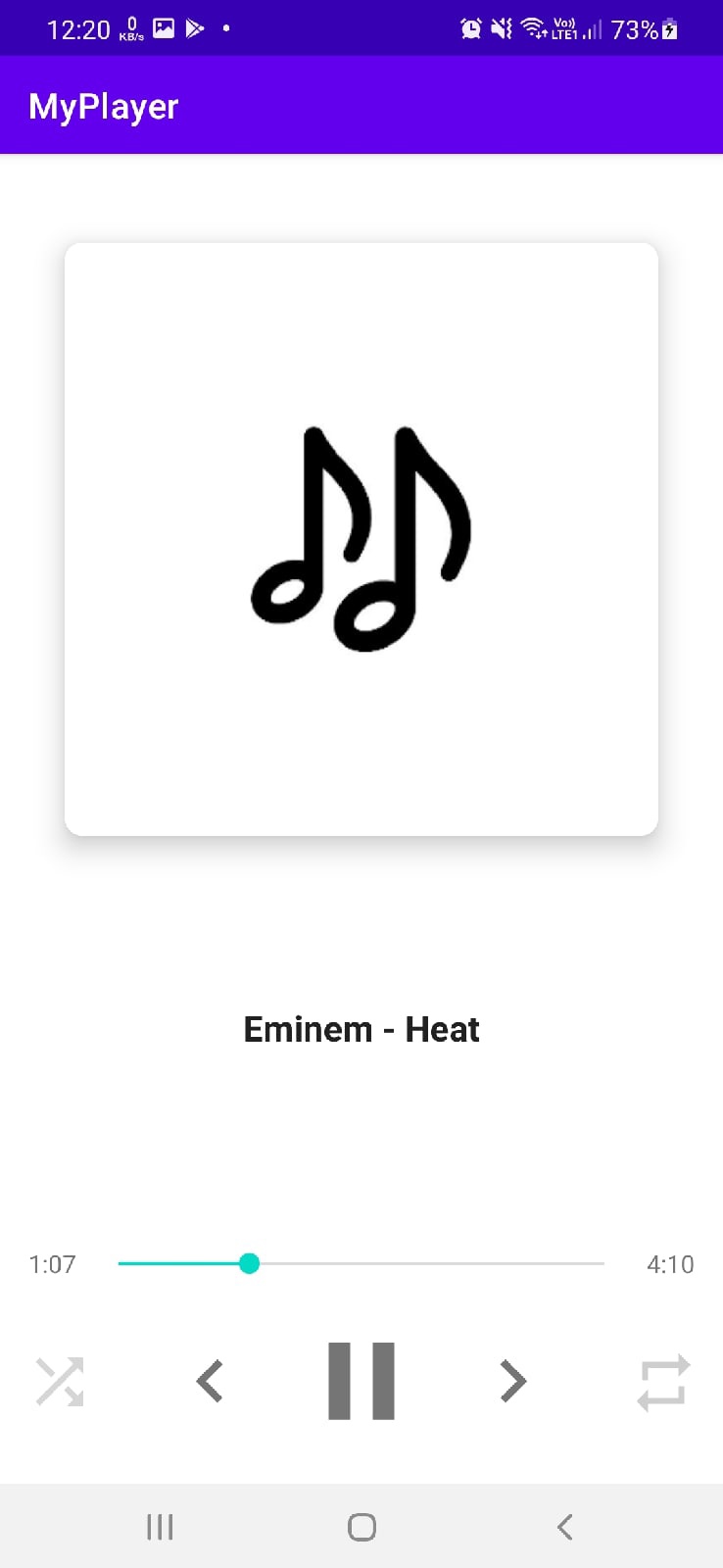


Рисунок 23. Текущая песня на главном экране

Чтобы **поменять позицию** трека в плейлисте нужно зажать на соответствующий элемент в списке и перетащить его на новую позицию.

Кнопка в правом нижнем углу на экране плейлиста позволяет **удалить песни** из плейлиста. При нажатии появятся флажки для выбора файлов, которые нужно удалить.

Для **перемешивания** песен или **повтора** песни используются соответствующие кнопки в нижней части экрана с текущей песней.

### Сообщение оператору

При попытке запуска плеера может появиться сообщение «Add items to playlist» (Рисунок 24):

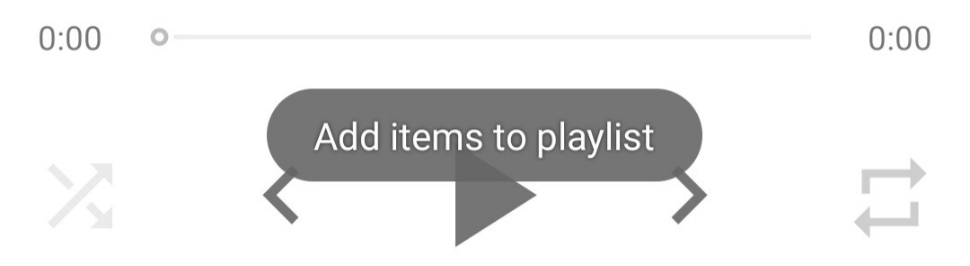


Рисунок 24. Сообщение об ошибке

**Причина:** отсутствие песен в плейлисте.

**Действия программы:** программа не запускает плеер.

**Действия оператора**: добавить песни в плейлист.

## To-Do лист

* Улучшить дизайн программы
* Добавить возможность создания нескольких плейлистов
* Увеличить функционал программы

# Заключение

Стандартный проигрыватель системы Android не очень удобен и имеет малый функционал. По этой причине в результате выполнения курсового проекта был создан музыкальный проиграватель для системы Android. Пока его функционал и возможности далеки от совершенства, но его уже можно использовать в качестве плеера для прослушивания песен.

# Список литературы и интернет ресурсов

1. Kotlin

<https://kotlinlang.ru/>

1. Android

<https://developer.android.com/>

1. Медиапроигрыватель

<https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B4%D0%B8%D0%B0%D0%BF%D1%80%D0%BE%D0%B8%D0%B3%D1%80%D1%8B%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C>

1. Head First Android Development: A Brain-Friendly Guide. Гриффитс Дэвид, Гриффитс Дон.
2. Kotlin for Android Developers А. Лейва
3. MP3

<https://ru.wikipedia.org/wiki/MP3>

1. AIMP

<https://play.google.com/store/apps/details?id=com.aimp.player&hl=ru&gl=US>

1. Audify Music Player

<https://play.google.com/store/apps/details?id=com.musicplayer.playermusic&hl=ru&gl=US>

1. BlackPlayer

<https://play.google.com/store/apps/details?id=com.musicplayer.blackplayerfree&hl=ru&gl=US>

1. jetAudio

<https://play.google.com/store/apps/details?id=com.jetappfactory.jetaudio&hl=ru&gl=US>