**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 мультимедиапроигрыватель) — компьютерная программа, предназначенная для воспроизведения файлов мультимедиа-контента.

Одна разновидность медиаплееров предназначена для воспроизведения только аудио- или же видеофайлов, и которые называются, соответственно, — 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 делятся 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 содержат 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 хранения аудиоинформации. MP3 является одним из самых распространённых и популярных форматов цифрового кодирования звуковой информации. Он широко используется в файлообменных сетях для оценочного скачивания музыкальных произведений. Формат может проигрываться практически во всех популярных операционных системах, на большинстве портативных аудиоплееров, а также поддерживается всеми современными моделями музыкальных центров и DVD-плееров.

Как и формат JPEG, **MP3** использует спектральные отсечения, согласно психоакустической модели. Звуковой сигнал разбивается на равные по продолжительности отрезки, каждый из которых после обработки упаковывается в свой фрейм (кадр). Разложение в спектр требует непрерывности 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 гармоник. 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 актуален, высокочастотные гармоники не отсекаются, как в JPEG, а только выборочно удаляются, чтобы уменьшить поток информации за счёт разрежения спектра. После спектральной «зачистки» применяются математические 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 каналов). 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, которые, в свою очередь, состоят из заголовка и блока данных. Такая последовательность фрагментов называется элементарным потоком. Фрагменты не являются независимыми элементами («резервуар байт»), и поэтому не могут быть извлечены произвольно. Блок данных MP3-файла содержит сжатую 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

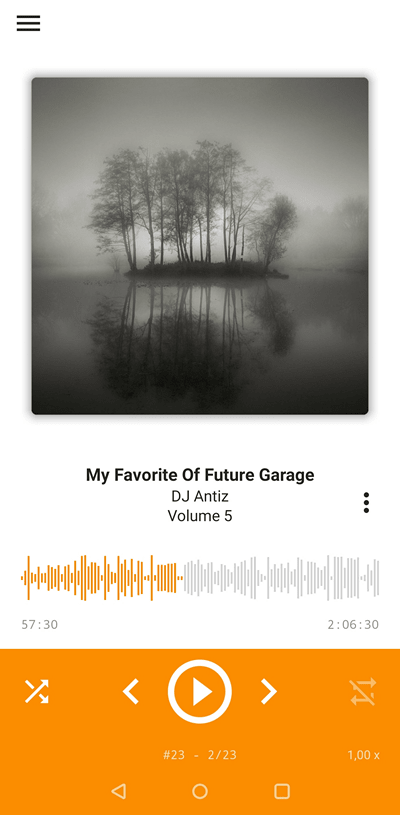


Рисунок 1. Аудиоплеер AIMP.

* Audify Music Player



Рисунок 2. Аудиоплеер Audify Music Player.

* BlackPlayer



Рисунок 3. Аудиоплеер BlackPlayer.

* jetAudio

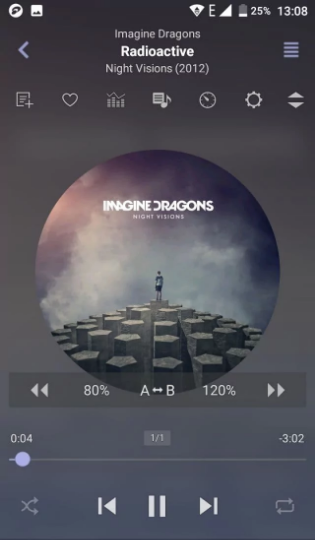


Рисунок 4. Аудиоплеер jetAudio.

**Проектная часть**

**Описание инструментов разработки программы**

**Язык программирования.**

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

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.

**Диаграмма прецедентов**

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



Рисунок 5. Диаграмма прецедентов приложения.

**Общая схема работы программы**

В данном разделе содержится набор возможных состояний приложения и общая схема работы приложения.



Рисунок 6. Схема работы программы.

**Диаграммы классов**

В данном разделе находятся диаграммы классов приложения.

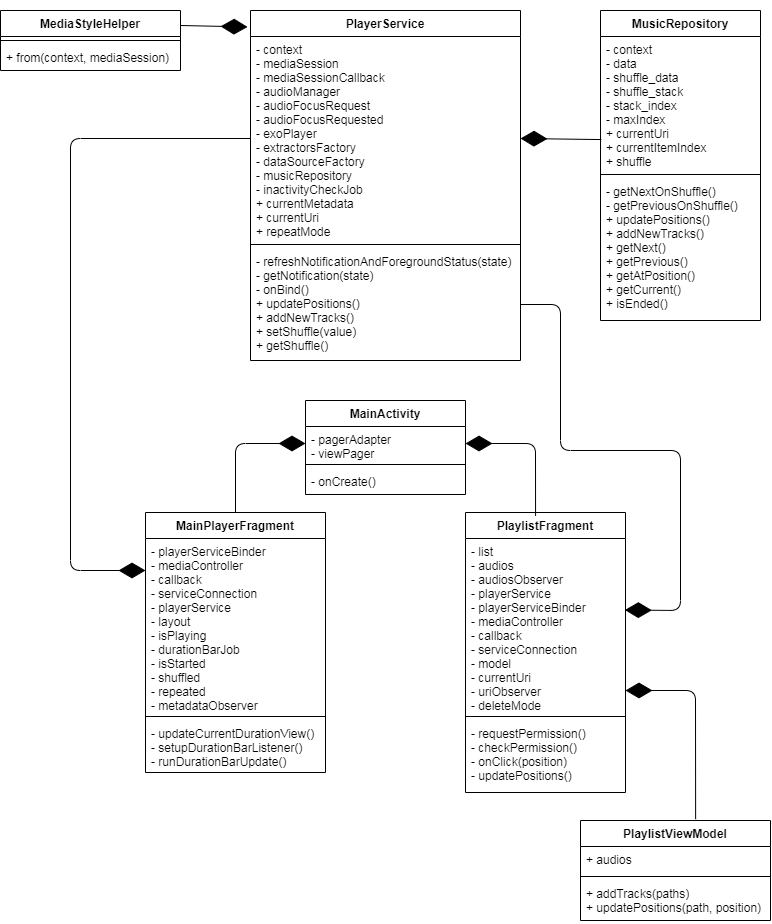


Рисунок 7. Диаграмма основных классов приложения

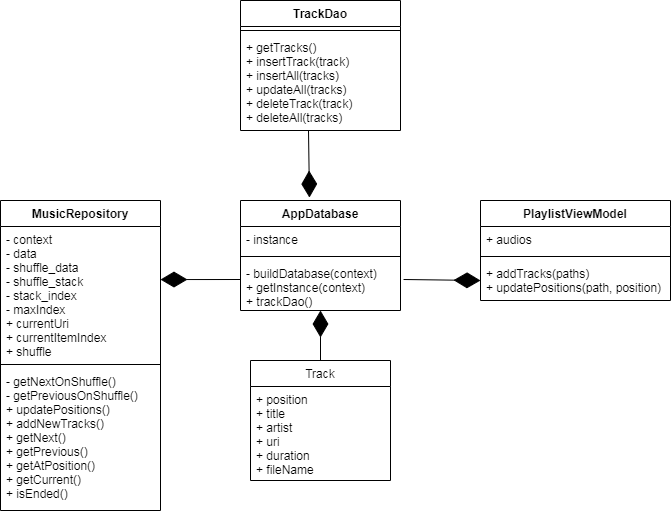
****

Рисунок 8. Диаграмма классов, связанных с БД.

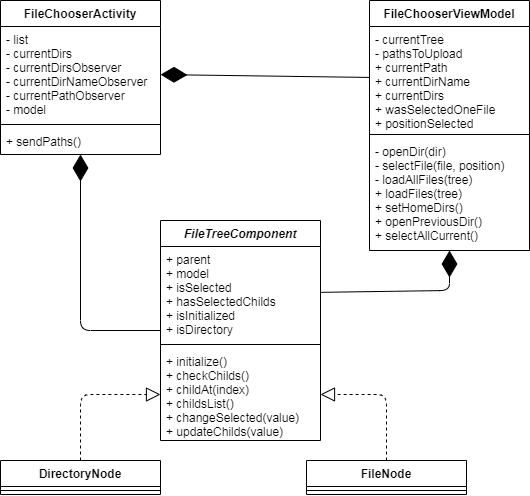


Рисунок 9. Диаграмма классов экрана добавления файлов.

// 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!!)

}

}

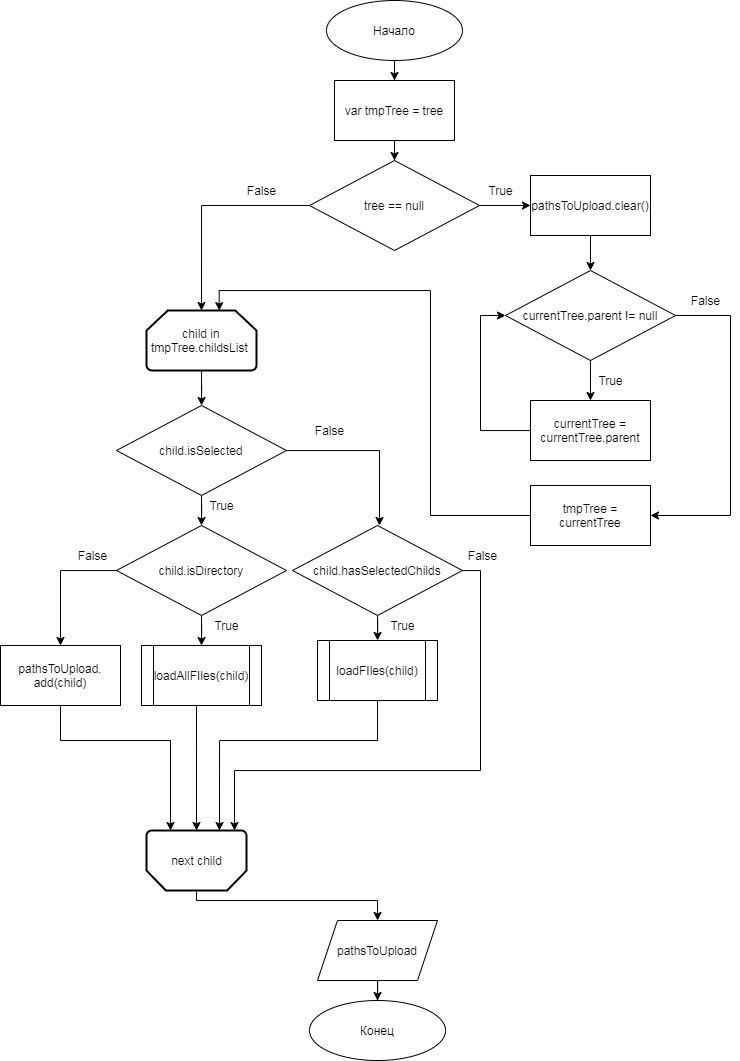


Рисунок 10. Блок схема функции loadFIles.

**Описание модулей**

В моей программе реализованы следующие модули:

* Экран управления плеером и текущим треком
* Экран управления плейлистом
* Экран добавления файлов в плейлист
* Сервис для проигрывания музыки
* Репозиторий получения данных из БД для сервиса

**Экран управления плеером и текущим треком** – на данном экране отображается информация о текущей проигрываемой песне, а также находятся кнопки для управления плеером. 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.

**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.

**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).

**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)

**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.

Картинка.

**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.

Картинка.

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

Картинка.