МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»

Кафедра телекоммуникационных систем и вычислительных средств (ТС и ВС)

РГР
по дисциплине
«Визуальное програмироавние»

по теме:
РАЗРАБОТКА МЕДИА-ПЛЕЕРА НА ANDROID С ИСПОЛЬЗОВАНИЕМ JETPACK COMPOSE

Студент:
Группа ИА-331 *С.Х. Иргит*

Предподаватель:
Старший преподаватель *Р.В. Ахпашев*

Новосибирск 2025 г.

СОДЕРЖАНИЕ

1 ВВЕДЕНИЕ

Современные мобильные устройства предоставляют широкие возможности для обработки мультимедийной информации. Аудио и видео являются важной частью пользовательского опыта, особенно в эпоху постоянной доступности потоковых и локальных медиа. Одной из базовых задач является воспроизведение аудиофайлов, что реализуется с помощью медиаплееров.

Актуальность темы обусловлена тем, что с каждым годом растёт количество пользователей Android, а спрос на качественные и функциональные мультимедийные приложения не снижается. Поэтому рассмотрение практической реализации аудиоплеера — это полезная задача как с точки зрения обучения, так и с точки зрения создания реального продукта.

В работе рассматривается создание простого медиаплеера с графическим интерфейсом на основе Jetpack Compose — современного инструмента для UI-разработки от Google. Приложение позволяет воспроизводить локальные аудиофайлы, переключаться между треками и управлять воспроизведением.

2 ПОСТАНОВКА ЗАДАЧИ

2.1 Цель работы

Целью данной работы является разработка мобильного приложения — медиа плеера , реализованного на языке Kotlin. Интерфейс создаётся с помощью Jetpack Compose [1].

2.2 Задачи

Реализовать основные функции медиа плеера: воспроизведение аудиофайлов, навигация по плейлисту, пауза, переход к следующему/предыдущему треку, отображение информации о треке.

3 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

3.0.1 Jetpack Compose

Jetpack Compose представляет собой декларативный подход к разработке пользовательских интерфейсов. В отличие от традиционного XML, в Compose UI описывается с помощью функций на Kotlin [2]. Это упрощает разработку и позволяет избежать избыточности кода. Например, элемент интерфейса может быть создан как простая функция:

```
@Composable
fun Greeting(name: String) {
    Text(text = "Hello $name!")
}
```

3.0.2 MediaPlayer API

Для воспроизведения аудио в Android используется класс `MediaPlayer`[3]. Он предоставляет следующие возможности:

- Загрузка локальных и удалённых аудиофайлов;
- Поддержка различных форматов (MP3, WAV, OGG и др.);
- Управление воспроизведением (пауза, стоп, перемотка);
- Работа в фоновом режиме.

Пример использования MediaPlayer:

```
val player = MediaPlayer()
player.setDataSource("path/to/file.mp3")
player.prepare()
player.start()
```

3.0.3 Система разрешений в Android

Начиная с Android 6.0 (API 23), система безопасности требует запроса разрешений во время выполнения (runtime permissions). Для доступа к фай-

лам, расположенным во внешнем хранилище, необходимо явно запрашивать разрешение у пользователя. Android разделяет разрешения на два типа: обычные (normal) и опасные (dangerous). Чтение медиафайлов относится ко второму типу.

С Android 13 (API 33) были введены новые отдельные разрешения, включая `READ_MEDIA_AUDIO`[4], позволяющее приложению читать только аудиофайлы. Это позволяет более точно контролировать доступ приложений к пользовательским данным.

В Jetpack Compose для запроса разрешения используется специальный механизм:

```
val launcher = rememberLauncherForActivityResult(
    ActivityResultContracts.RequestPermission()
) { isGranted ->
    if (isGranted) {
        // Разрешениеполучено
    }
}
```

3.0.4 Работа с состоянием в Compose

Jetpack Compose использует реактивную модель состояния. Это означает, что UI автоматически обновляется при изменении состояния. Для хранения состояния используется API `remember` и `mutableStateOf`:

```
var isPlaying by remember { mutableStateOf(false) }

Button(onClick = { isPlaying = !isPlaying }) {
    Text(if (isPlaying) "Pause" else "Play")
}
```

Состояния в Compose позволяют создавать динамические и отзывчивые интерфейсы без необходимости ручного обновления UI-элементов.

3.0.5 Обработка жизненного цикла и ресурсов

При работе с мультимедиа важно учитывать жизненный цикл активности. Объекты, такие как `MediaPlayer`, требуют ручного освобождения ресурсов во избежание утечек памяти. Compose предоставляет функцию `DisposableEffect`, которая позволяет управлять ресурсами:

```
DisposableEffect(Unit) {
    onDispose {
        player.release()
    }
}
```

Это особенно полезно для компонентов, которые не управляются фреймворком напрямую, как в случае с Java- или Android-классами.

3.0.6 Архитектурные подходы: MVVM

Несмотря на то, что демонстрационное приложение реализовано в одном файле, для более масштабных проектов рекомендуется использовать архитектурный паттерн MVVM (Model-View-ViewModel). В нём:

- **Model** — содержит бизнес-логику и доступ к данным (например, загрузка аудиофайлов);
- **ViewModel** — содержит состояние UI и бизнес-логику, обрабатывает события;
- **View** — Jetpack Compose-компоненты, подписанные на данные ViewModel.

Jetpack предоставляет библиотеку `androidx.lifecycle.viewmodel.compose` которая позволяет интегрировать ViewModel в Compose-приложения.

3.0.7 Преимущества использования Jetpack Compose

Jetpack Compose обладает рядом преимуществ по сравнению с традиционным подходом через XML:

- Более простая и лаконичная запись интерфейсов;

– Возможность полной декларативности и реактивности;
– Отличная поддержка LiveData и Flow;
– Интеграция с архитектурными компонентами Android;
– Улучшенная совместимость с Kotlin.

4 ПРАКТИЧЕСКАЯ ЧАСТЬ

4.1 Анализ и структура реализации музыкального плеера

Проект состоит из одной основной активности и одного основного компонента для интерфейса.

4.1.1 Структура проекта

В проекте определена одна активность — `MediaPlayerActivity`, которая служит точкой входа и отвечает за запуск пользовательского интерфейса с помощью функции `setContent`. Внутри вызывается основная компонуемая функция `MusicLibraryUI`, реализующая весь интерфейс и логику плеера.

4.1.2 Функция интерфейса MusicLibraryUI

Функция `MusicLibraryUI` выполняет ключевые задачи приложения:

- Запрашивает у пользователя разрешение на чтение медиафайлов;
- Загружает аудиофайлы из папки `Music` во внешнем хранилище;
- Отображает список доступных треков с помощью компонента `LazyColumn`;
- Управляет воспроизведением треков с помощью `MediaPlayer`;
- Следит за состоянием плеера и обновляет интерфейс (позиция, длительность и т.д.).

4.1.3 Запрос разрешений

Для корректной работы с файловой системой необходимо запросить соответствующие разрешения у пользователя. В зависимости от версии Android используются:

- `READ_MEDIA_AUDIO` — начиная с Android 13 (API 33);
- `READ_EXTERNAL_STORAGE` — для Android 12 и ниже.

Разрешения запрашиваются через механизм `rememberLauncherForActivityR` из Compose API.

4.1.4 Загрузка аудиофайлов

Файлы с расширениями `.mp3`, `.wav`, `.ogg` загружаются с помощью функции `loadAudioFiles`, которая ищет их в стандартной папке `Environment.DIRECTORY_MUSIC`.

4.1.5 Управление воспроизведением

Воспроизведение реализовано с помощью Android API `MediaPlayer`. Функция `playTrack(index)` отвечает за выбор и воспроизведение трека. Также предусмотрены следующие элементы управления:

- Переключение на следующий и предыдущий трек;
- Кнопка воспроизведения и паузы;
- Отображение текущей позиции воспроизведения и полной длительности трека.

Состояние обновляется в режиме реального времени с помощью корутин и реактивных состояний Compose.

4.1.6 Пользовательский интерфейс

Интерфейс построен на компонентах Jetpack Compose: `Column`, `LazyColumn`, `ListItem`, `IconButton`. При выборе трека пользователь видит его название, а также может управлять воспроизведением через удобный набор иконок. Приложение динамически обновляет отображаемую информацию в зависимости от выбранного трека и текущего состояния плеера.

5 ЗАКЛЮЧЕНИЕ

Разработка медиаплеера на Android позволила на практике освоить ключевые концепции платформы, включая работу с разрешениями, пользовательским интерфейсом и воспроизведением медиа. Применение Jetpack Compose упростило реализацию UI, сделав код более читаемым и гибким. Созданное приложение может стать базой для более сложного медиаплеера, поддерживающего плейлисты, обложки, работу в фоне и другие функции. Работа имеет практическую ценность как учебный проект, демонстрирующий современные подходы к мобильной разработке.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Android Developers*. Jetpack Compose Documentation. — 2025. — URL: https://developer.android.com/jetpack/compose ; Accessed: 2025-05-23.
2. *JetBrains*. Kotlin Language Documentation. — 2025. — URL: https://kotlinlang.org/docs/reference/ ; Accessed: 2025-05-23.
3. *Android Developers*. MediaPlayer overview. — 2025. — URL: https://developer.android.com/guide/topics/media/mediaplayer ; Accessed: 2025-05-23.
4. *Android Developers*. Request app permissions. — 2025. — URL: https://developer.android.com/training/permissions/requesting ; Accessed: 2025-05-23.