МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ

КАЛИНИНГРАДСКОЙ ОБЛАСТИ

ПРОФЕССИОНАЛЬНАЯ ОБРАЗОВАТЕЛЬНАЯ ОРГАНИЗАЦИЯ

«КОЛЛЕДЖ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И СТРОИТЕЛЬСТВА»

(ГБУ КО ПОО «КИТиС»)

|  |  |  |
| --- | --- | --- |
| Курсовой проект  допущен к защите  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (дата)  Зам.директора по УМР  \_\_\_\_\_\_\_\_\_\_\_\_\_Павленко Г.Я.  (подпись) |  | Курсовой проект  защищен с оценкой  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (оценка)  Руководитель работы  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) |

Пояснительная записка к курсовому проекту

по дисциплине: МДК 01.01 Разработка программных модулей

Тема Разработка приложения под ОС Android для создания электронной музыки

Специальность 09.02.07 «Информационные системы и программирование»

|  |  |
| --- | --- |
| Выполнил: | студент 3 курса,  группы ИСп 19-1  Толстов Владислав Андреевич  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |
| Руководитель: | Большакова-Стрекалова Анна Викторовна, преподаватель ГБУ КО ПОО КИТиС  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

Калининград

2024

Аннотация

Тема проекта: "Разработка приложения под ОС Android для создания электронной музыки ".

Данный курсовой проект посвящена созданию мобильного приложения, которое позволяет пользователям создавать музыку путём смешивания и редактирования аудиосэмплов. Приложение предоставляет возможность импорта аудиофайлов, их последующей обработки и комбинирования в единые музыкальные композиции. Основное внимание уделяется интуитивно понятному интерфейсу и доступности функций для аматоров, так и для опытных пользователей.

Исходные данные и средства реализации:

Язык программирования: Java.

Среда разработки: Android Studio.

Использованы технологии Android SDK и библиотека FFmpeg для обработки медиа.

Разработанное программное обеспечение и все материалы по курсовому проекту доступны на GitHub по ссылке: [insil3nc3/Cours3 (github.com)](https://github.com/insil3nc3/Cours3)

Эта работа может служить отправной точкой для дальнейших исследований и разработок в области мобильных приложений для создания музыки, предоставляя широкие возможности для творчества и экспериментов в области музыкальной композиции.

Содержание

[Введение 4](#_Toc165544921)

[1 Описание предметной области 4](#_Toc165544922)

[1.1 Аналоги разрабатываемого приложения 5](#_Toc165544923)

[1.1.2 FL Studio Mobile (Image-Line) 5](#_Toc165544924)

[1.2 Техническое задание 7](#_Toc165544925)

[1.3 Описание структуры приложения 15](#_Toc165544926)

[2 Описание разработки приложения 18](#_Toc165544927)

[2.1 Обоснование средств разработки 18](#_Toc165544928)

[2.2 Разработка интерфейса 19](#_Toc165544929)

[2.3 Разработка логики работы приложения. Схема взаимодействия компонентов проекта 21](#_Toc165544930)

[2.4 Описание переменных, компонентов, классов и подпрограмм 22](#_Toc165544931)

[3 Тестирование и установка приложения 37](#_Toc165544932)

[3.1 Сценарии тестирования: 37](#_Toc165544933)

[3.2 Предполагаемые ошибки: 40](#_Toc165544934)

[3.3 Установка приложения: 40](#_Toc165544935)

[Определения, сокращения и аббревиатуры 41](#_Toc165544936)

[Заключение 42](#_Toc165544937)

[Список использованной литературы 42](#_Toc165544938)

[Приложение А. Код приложения 43](#_Toc165544939)

Введение

Создание музыки с помощью цифровых технологий — это быстро развивающаяся область, которая привлекает множество музыкантов и продюсеров, желающих исследовать новые возможности для выражения своего творчества. С развитием мобильных технологий возникла потребность в создании инструментов, которые позволяют пользователям генерировать и обрабатывать музыкальные треки непосредственно на своих мобильных устройствах. Это открывает новые горизонты для творчества на ходу, делая процесс создания музыки более доступным и удобным.

Разрабатываемое мобильное приложение для создания музыки через сэмплирование и смешивание различных звуков позволяет пользователям создавать уникальные музыкальные композиции. Такой подход позволяет не только профессиональным музыкантам, но и аматорам без специальной музыкальной подготовки реализовывать свои творческие идеи.

Тема данной курсовой работы актуальна, так как музыкальная индустрия продолжает активно развиваться, и существует потребность в новых цифровых решениях, которые бы упрощали процесс создания музыки и делали его более интерактивным и доступным. Рассматриваемая проблема уже нашла отражение в ряде существующих приложений, однако потенциал для развития и улучшения еще велик, что подтверждается постоянным появлением новых продуктов в данной области. Это свидетельствует о значимости дальнейших исследований и разработок в сфере мобильных музыкальных приложений, способных предложить пользователю широкие функциональные возможности и интуитивно понятный интерфейс.

Цель данной работы — разработать мобильное приложение, которое не только соответствует текущим требованиям музыкантов и продюсеров, но и предоставляет новые инструменты для творчества, открывая новые границы в создании и обработке музыкального контента.

1. Описание предметной области
   1. Аналоги разрабатываемого приложения

1.1.2 FL Studio Mobile (Image-Line)

Описание: Приложение для создания и сохранения музыкальных



Рисунок 1 – логотип приложения

Функционал Fl Studio Mobile.

Главный экран приложения:

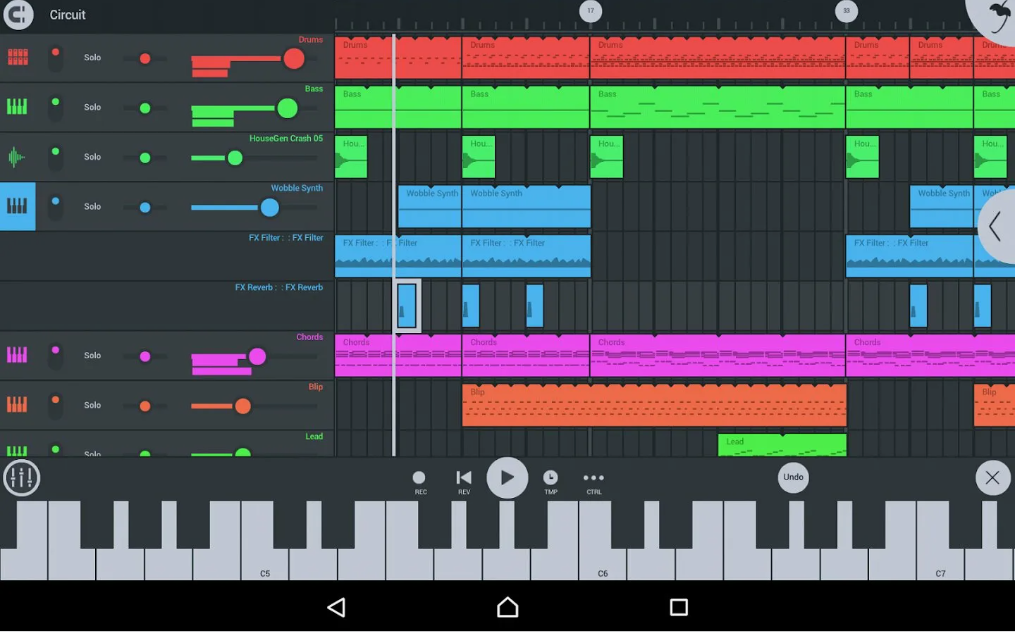


Рисунок 2 – главный экран приложения

На главном экране расположены поля для добавления инструментов, настройка для каждого инструмента. В центрально-правой части расположено окно, в которое добавляются блоки инструментов. Для того, чтобы прописать мелодию, снизу расположен piano roll – интерфейс пианино для удобного выбора нот инструменту.

Домашняя панель:

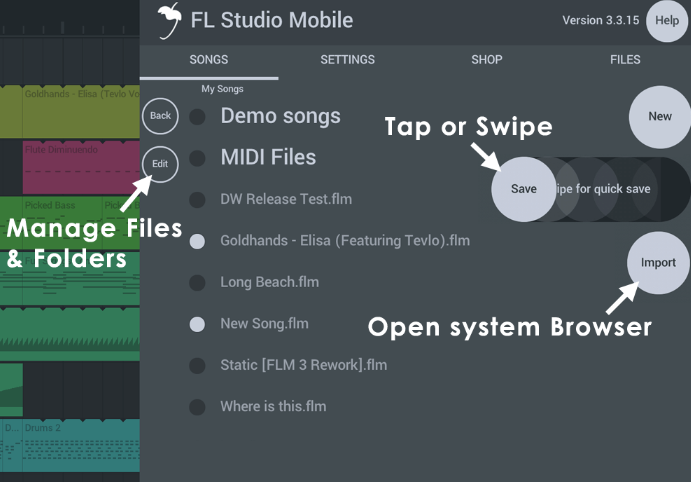


Рисунок 3 – домашняя панель

В домашней панели расположены кнопки для сохранения, импорта, создания нового проекта, импорта файла нот (MIDI file), окно открытия демо-проектов и кнопка помощи. Тут же находится файловый менеджер, настройки проекта и магазин файлов и плагинов.

Создание нового проекта:

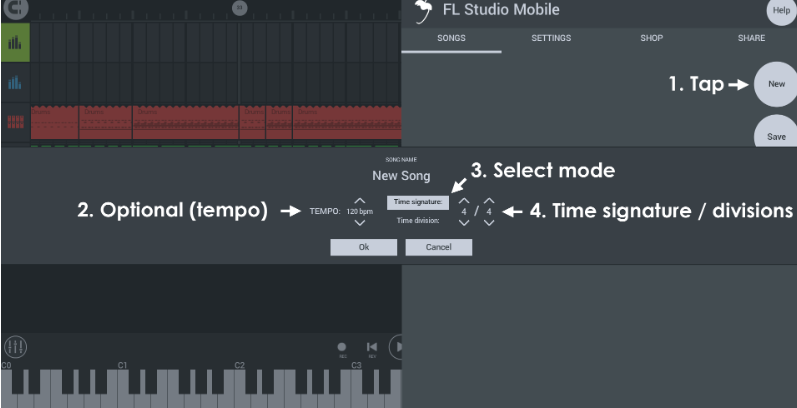


Рисунок 4 – создание нового проекта

Для создания проекта нужно указать темп проекта, имя, выбрать режим: Time signature или Time division.

Сохранение или отправка файла:

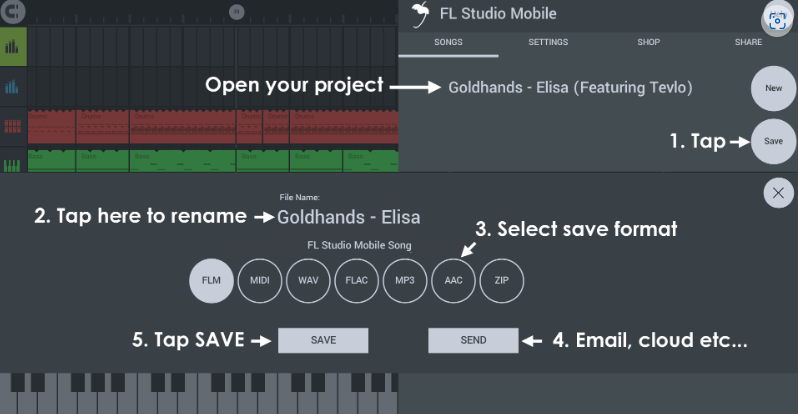


Рисунок 5 – сохранение или отправка файла

В окне сохранении или отправки проекта можно выбрать формат файла сохранения, есть возможность переименовать название и кнопки «save» для сохранения и «send» для отправки.

* 1. Техническое задание

1. Введение

1.1 Цель документа

Этот документ предназначен для определения и описания функциональных и нефункциональных требований к мобильному приложению "INLOOPER". Техническое задание будет служить руководством для разработчиков, проектных менеджеров и всех заинтересованных сторон, обеспечивая последовательное и целенаправленное развитие проекта.

1.2 Область применения приложения

"INLOOPER" - это мобильное приложение для создания и редактирования музыкальных лупов и сэмплов, предназначенное для музыкантов и продюсеров всех уровней. Приложение позволяет пользователям управлять музыкальными фрагментами, экспортировать их и делиться ими, поддерживая креативность и производительность вне зависимости от их местоположения.

2. Общие требования

2.1 Функциональные требования

2.1.1 Интерфейс пользователя:

Приложение должно предлагать простой и интуитивно понятный интерфейс, который позволяет пользователям без затруднений переходить между созданием, редактированием и управлением проектами.

2.1.2 Управление данными:

Должны быть реализованы функции для эффективного управления аудио файлами, включая загрузку, сохранение и экспорт данных.

2.2 Нефункциональные требования

2.2.1 Производительность:

Приложение должно обеспечивать высокую производительность, обрабатывать данные без значительных задержек, особенно при работе с аудио.

2.2.2 Совместимость:

Поддержка различных устройств на Android с версией 7.0 и выше, адаптация под разные размеры экранов.

3. Системные требования

3.1 Аппаратные требования

Приложение должно быть способно функционировать на любом современном Android устройстве, которое имеет минимум 2 ГБ оперативной памяти и 4-ядерный процессор для обеспечения стабильной работы без технических сбоев.

3.2 Программные требования

3.2.1 Операционная система:

Поддержка Android OS 7.0 и выше для обеспечения широкой совместимости с различными устройствами.

3.2.2 Требования к разработке:

Программирование должно осуществляться в Android Studio с использованием языка Java, применение современных библиотек для работы с мультимедийными файлами.

4. Детальные требования

4.1 Функциональные требования по модулям

4.1.1 Создание и управление проектами

Создание нового проекта: Пользователи могут легко начинать новый проект, вводя его название в предоставленное поле.

Открытие существующих проектов: Предоставление списка всех ранее сохранённых проектов.

Управление проектами: Возможность удалять проекты прямо из интерфейса управления.

4.1.2 Редактирование аудио

Импорт аудиофайлов: Загрузка аудиофайлов в проект из локальной памяти устройства или через облачные сервисы.

Редактирование: Инструменты для редактирования громкости и регулировка панорамирования аудиофайлов.

Микширование треков: Инструменты для микширования нескольких аудиодорожек, включая регулировку уровней, баланса и других параметров микса.

4.1.3 Экспорт и сохранение

Сохранение проекта: Возможность сохранять текущее состояние проекта для последующей работы.

Экспорт аудио: Экспорт итогового микса в MP3 формат.

4.2 Ограничения и исключения

Ограничения на размер файлов: Максимальный размер файла для импорта ограничен 100 МБ для обеспечения стабильности работы приложения.

Количество треков в проекте: Максимальное количество треков в проекте ограничено до 50 для предотвращения перегрузки процессора устройства.

Временные ограничения: Максимальная длительность одного трека не должна превышать 10 минут.

Поддерживаемые форматы файлов: Приложение поддерживает только определённые форматы аудиофайлов, такие как WAV, MP3. Импорт файлов в других форматах требует их предварительной конвертации.

5. Требования к разработке

5.1 Тестирование

Модульное тестирование: Разработка и выполнение модульных тестов для каждого компонента приложения с использованием фреймворков, таких как JUnit, для обеспечения надежности функций.

Интеграционное тестирование: Проверка взаимодействия между компонентами приложения, чтобы убедиться, что они работают вместе как единое целое.

Приемочное тестирование: Проведение тестов, которые имитируют реальные сценарии использования приложения пользователями для проверки функциональности и удобства использования.

5.2 Документация

Техническая документация: Создание подробной документации для каждого аспекта приложения, включая архитектуру, используемые библиотеки, а также подробные описания всех функций и методов.

Пользовательская документация: Разработка инструкций для пользователей по использованию приложения.

6. Стадии и этапы разработки

6.1 Разработка требований и проектирование с 25.03.2024 по 01.04.2024.

6.2 Кодирование и реализация функционала с 02.04.2024 по 15.04.2024.

6.3 Тестирование и отладка с 16.04.2024 по 23.04.2024.

6.4 Подготовка к выпуску и сдача проекта 24.04.2024 - 25.04.2024.

7. Дополнительные требования

Совместимость версий: Убедиться, что все обновления совместимы с существующими версиями приложения и не вызовут потерю данных или сбои в работе.

8. Приложения

8.1 Макеты интерфейса

Визуальные прототипы: Предоставление детализированных макетов интерфейса, которые включают все экраны, диалоги и компоненты пользовательского интерфейса.

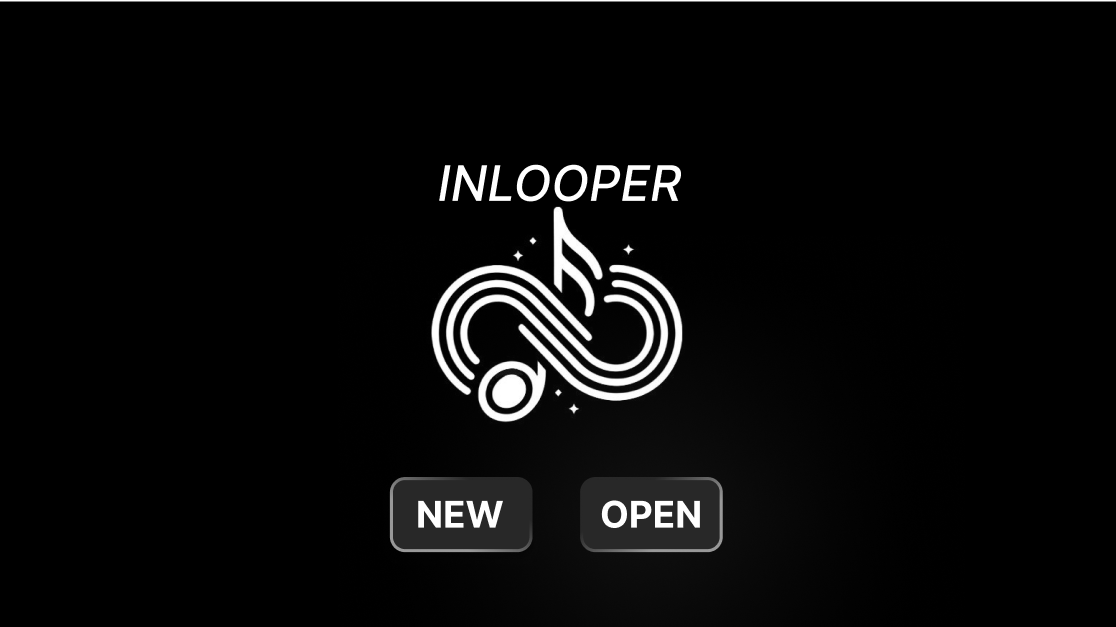


Рисунок 6 – Стартовый экран

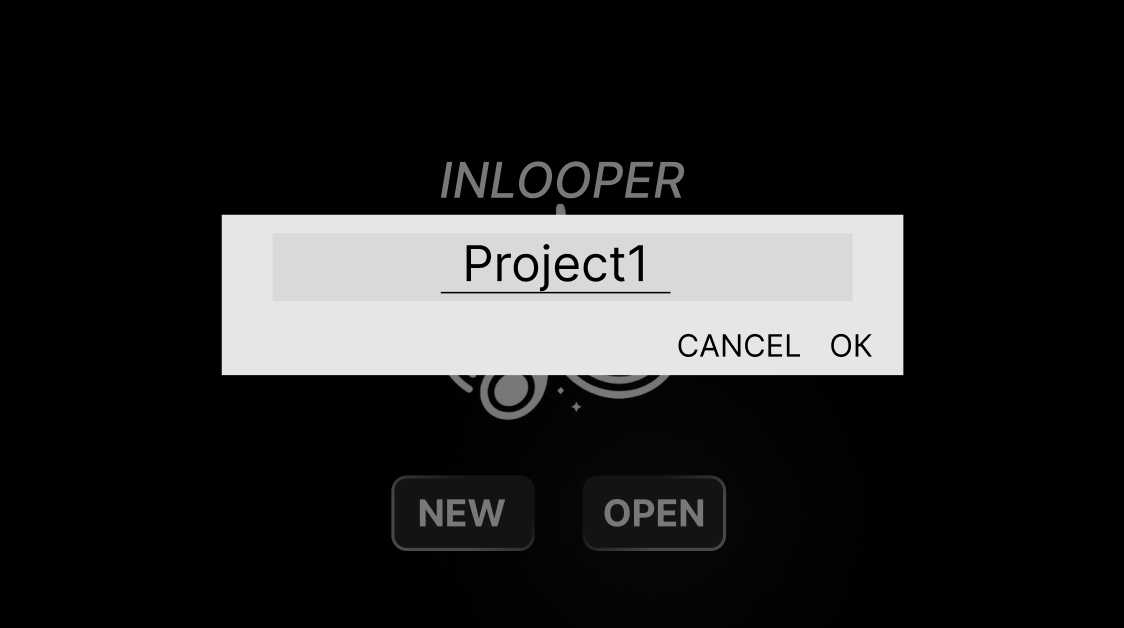


Рисунок 7 – Создание нового проекта и редактирование его названия

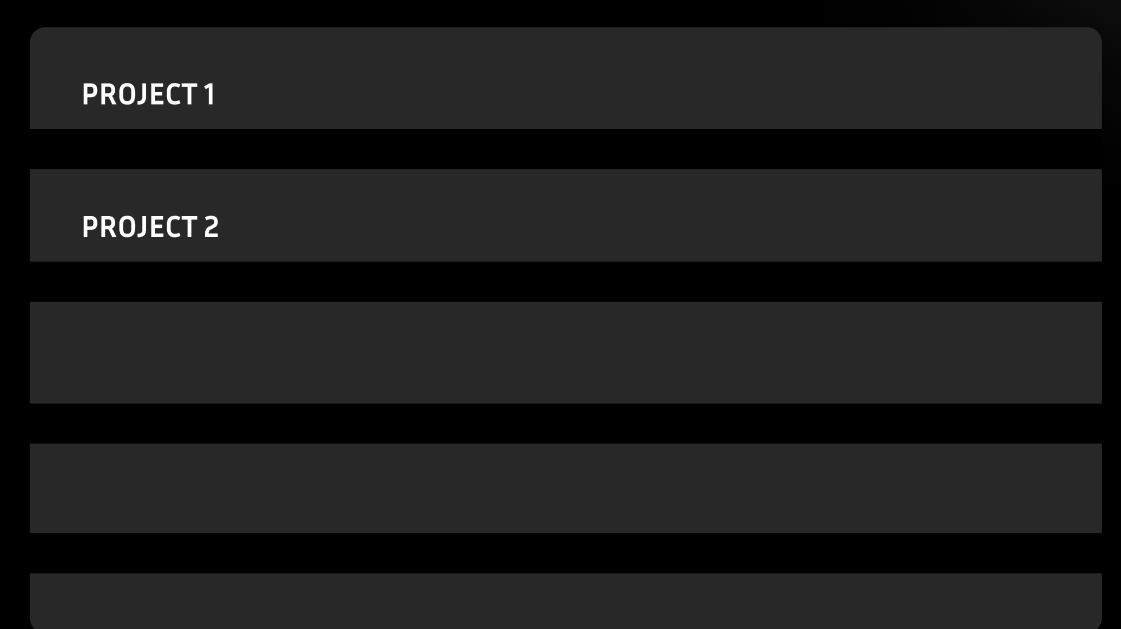


Рисунок 8 – Открытие хранилища проектов для взаимодействия с ними

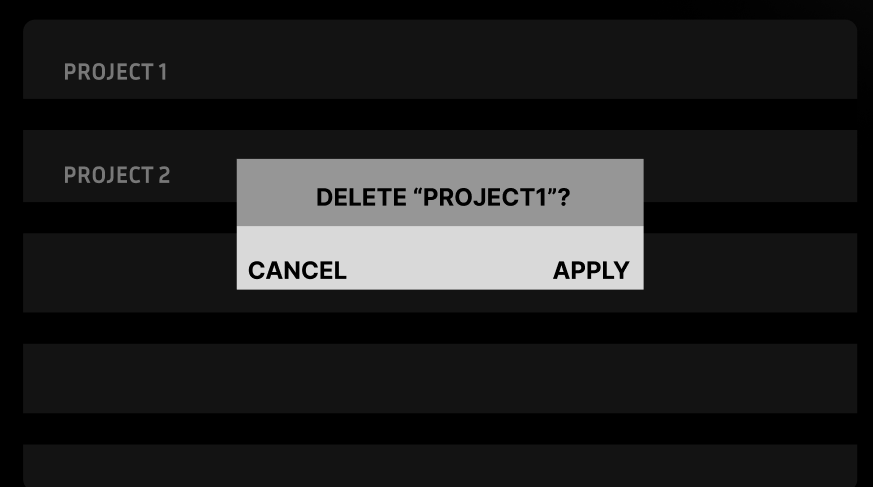


Рисунок 9 – Удаление проекта

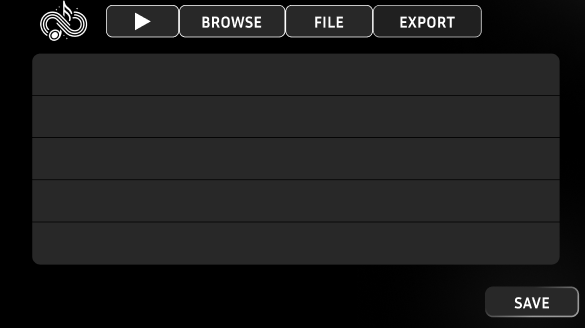


Рисунок 10 – Главное меню

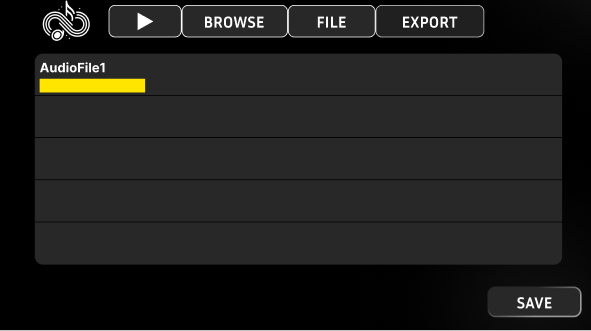


Рисунок 11 – Главное меню с добавленным аудиофайлом

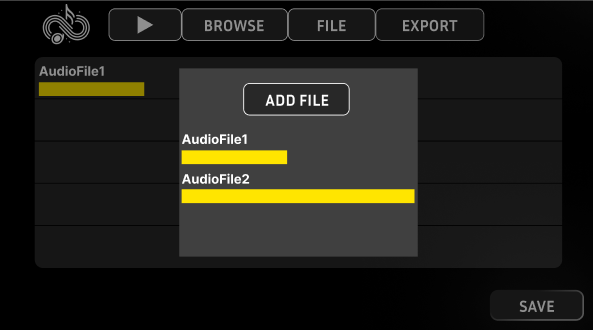


Рисунок 12 – Меню добавления аудиофайлов в проект

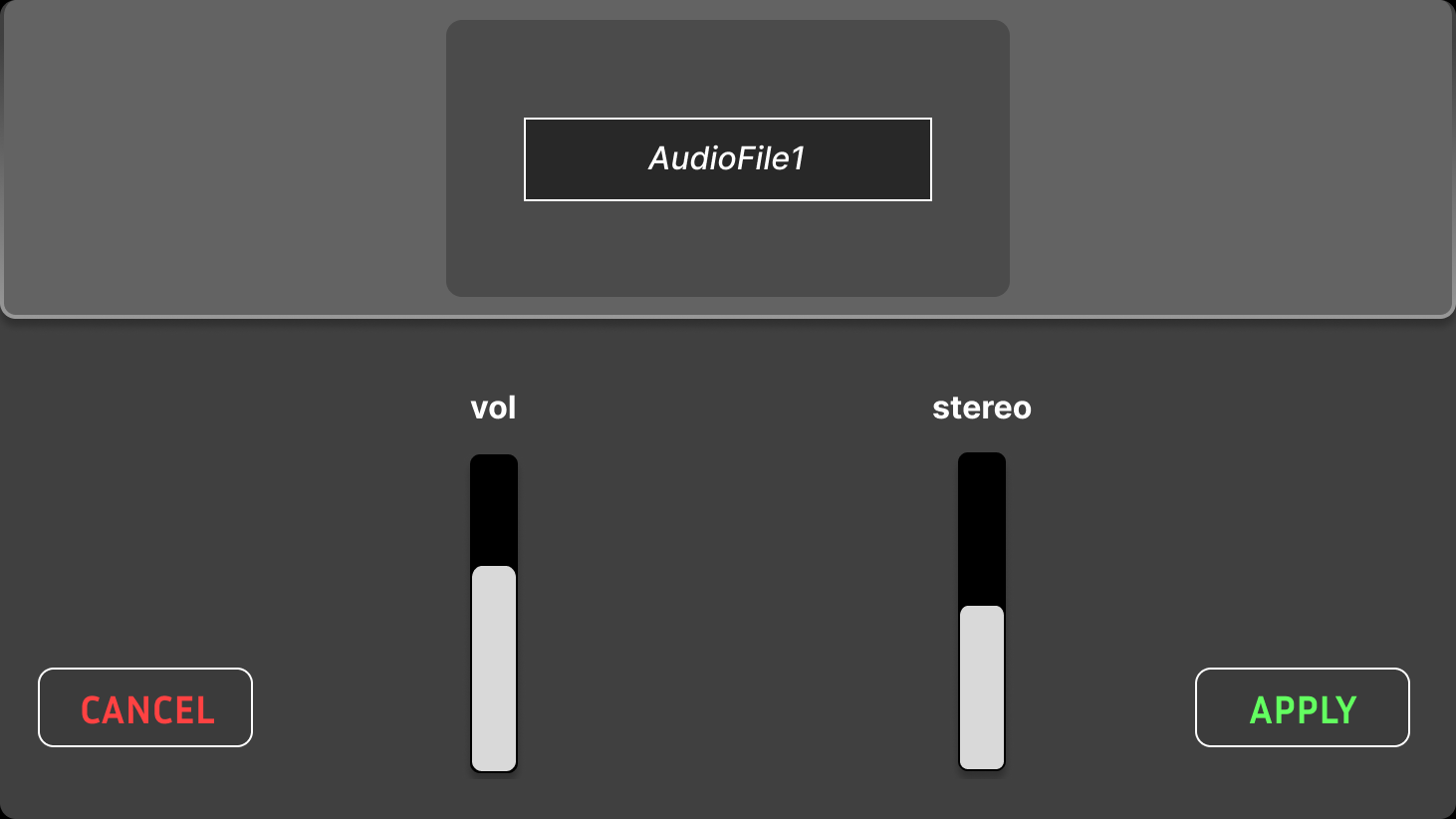


Рисунок 13 – редактирование аудиофайла

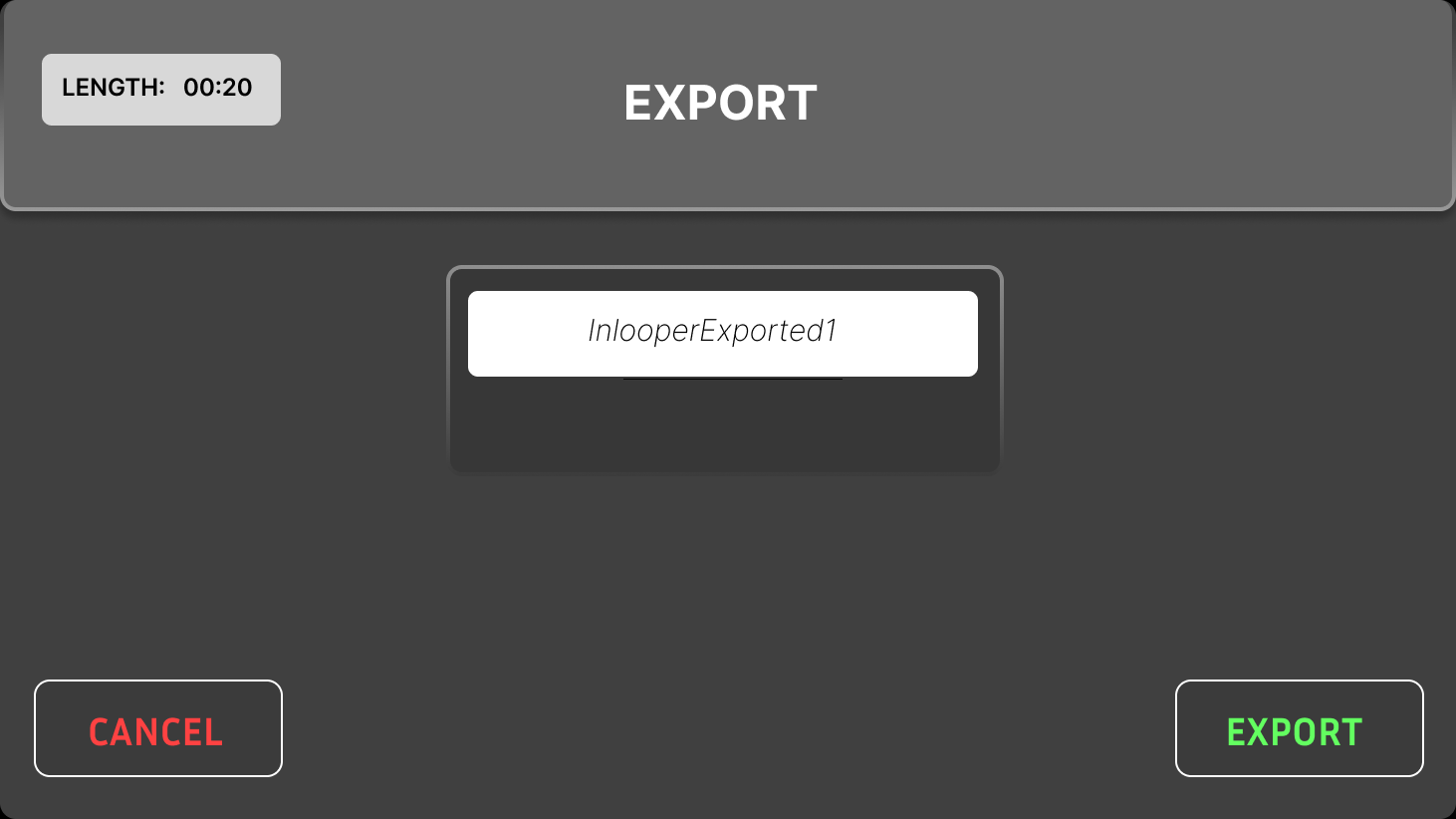


Рисунок 14 – Меню экспортирования итогового результата в MP3

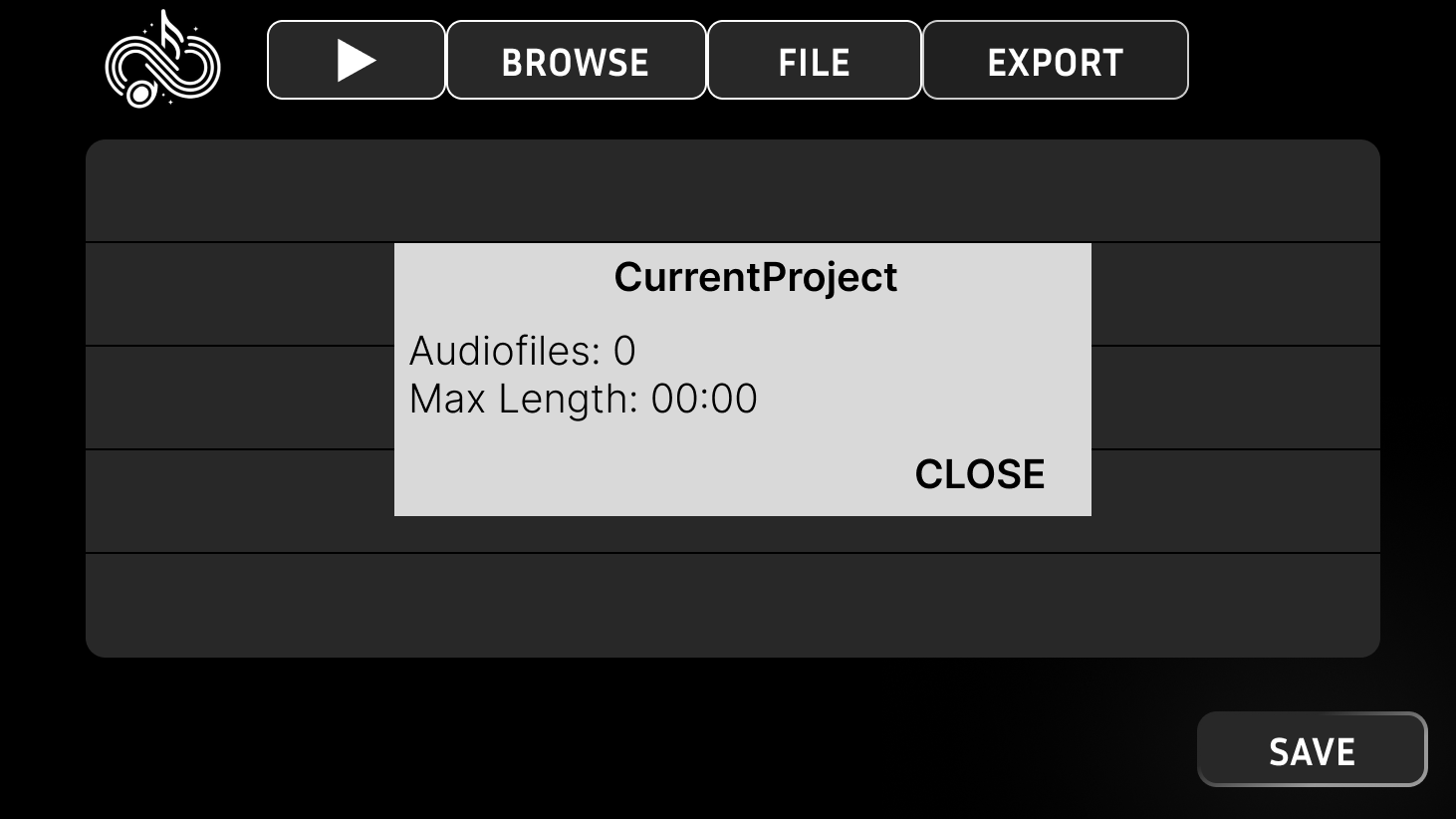


Рисунок 15 – меню информации о проекте.

8.3 Руководства пользователя

Руководство пользователя: Подробное описание функций приложения, советы по использованию и рекомендации по решению типичных проблем.

9. Технико-экономические показатели

9.1 Предполагаемая экономическая эффективность

Сокращение времени на создание музыкальных композиций благодаря мобильному доступу и упрощенным инструментам редактирования.

Увеличение дохода музыкантов за счет возможности быстрее выпускать и распространять свои работы.

9.2 Предполагаемые затраты на разработку

Бюджет на проект выделен не был.

Расходы, включающие зарплату разработчиков, стоимость тестирования и верификации, а также маркетинг и продвижение приложения, не были затрачены.

9.3 Плановые сроки начала и окончания работ по проекту

Разработка начнется 25.03.2024 и завершится 25.04.2024, после чего начнутся стадии тестирования и подготовки к выпуску.

1.3 Описание структуры приложения

Общая структура

Приложение INLOOPER организовано вокруг нескольких ключевых классов, которые взаимодействуют для обеспечения функциональности создания, редактирования и управления аудиопроектами. Ниже представлено более подробное описание каждого из классов:

1. MainActivity:

Основной экран приложения, который служит точкой входа для пользователя. Здесь происходит навигация ко всем основным функциям приложения, таким как создание и редактирование проектов.

1. StartActivity:

Стартовый экран приложения, который может содержать логику инициализации приложения или выбора проекта.

1. SplashActivity:

Экран-заставка приложения, отображается при запуске и предназначен для инициализации ресурсов или системных проверок перед переходом к основному контенту.

1. ProjectActivity:

Экран для управления и редактирования конкретного проекта. Пользователи могут добавлять, удалять и изменять аудиофайлы в рамках проекта.

1. EditAudioActivity:

Экран для детального редактирования аудиофайлов в проекте. Здесь можно изменять параметры звука, такие как громкость, стереопозиция и другие эффекты.

1. ExportActivity:

Экран для экспорта аудиофайлов в формате, доступном для воспроизведения на других устройствах или платформах.

1. AudioModel и ParcelableAudioModel:

Классы, представляющие аудиофайл, его свойства и методы для работы с аудио. ParcelableAudioModel предоставляет функциональность для передачи данных между компонентами Android.

1. Project:

Класс, представляющий музыкальный проект, содержит список аудиофайлов и методы для управления проектом.

1. AudioFileAdapter и ProjectAdapter:

Адаптеры для управления списками аудиофайлов и проектов в пользовательском интерфейсе. Обеспечивают связь данных с RecyclerView для отображения списка элементов.

1. FileUtils и AudioUtils:

Утилитные классы, содержащие статические методы для работы с файлами и аудио соответственно. Это включает в себя чтение, запись файлов, а также обработку аудиофайлов.

1. ProjectSaveAdapter:

Класс для адаптации объекта проекта при сохранении и загрузке. Используется для сериализации и десериализации данных проекта.

Взаимодействие компонентов

MainActivity служит центральным узлом для навигации по различным экранам (StartActivity, ProjectActivity, EditAudioActivity и т.д.).

ProjectActivity использует ProjectAdapter для отображения списка аудиофайлов, а каждый аудиофайл редактируется через EditAudioActivity.

AudioFileAdapter и ProjectAdapter обрабатывают действия пользователя, такие как выбор элемента списка и его редактирование.

AudioUtils и FileUtils вызываются различными компонентами приложения для выполнения задач по обработке и сохранению файлов.

ExportActivity используется для финальной обработки и экспорта проекта.

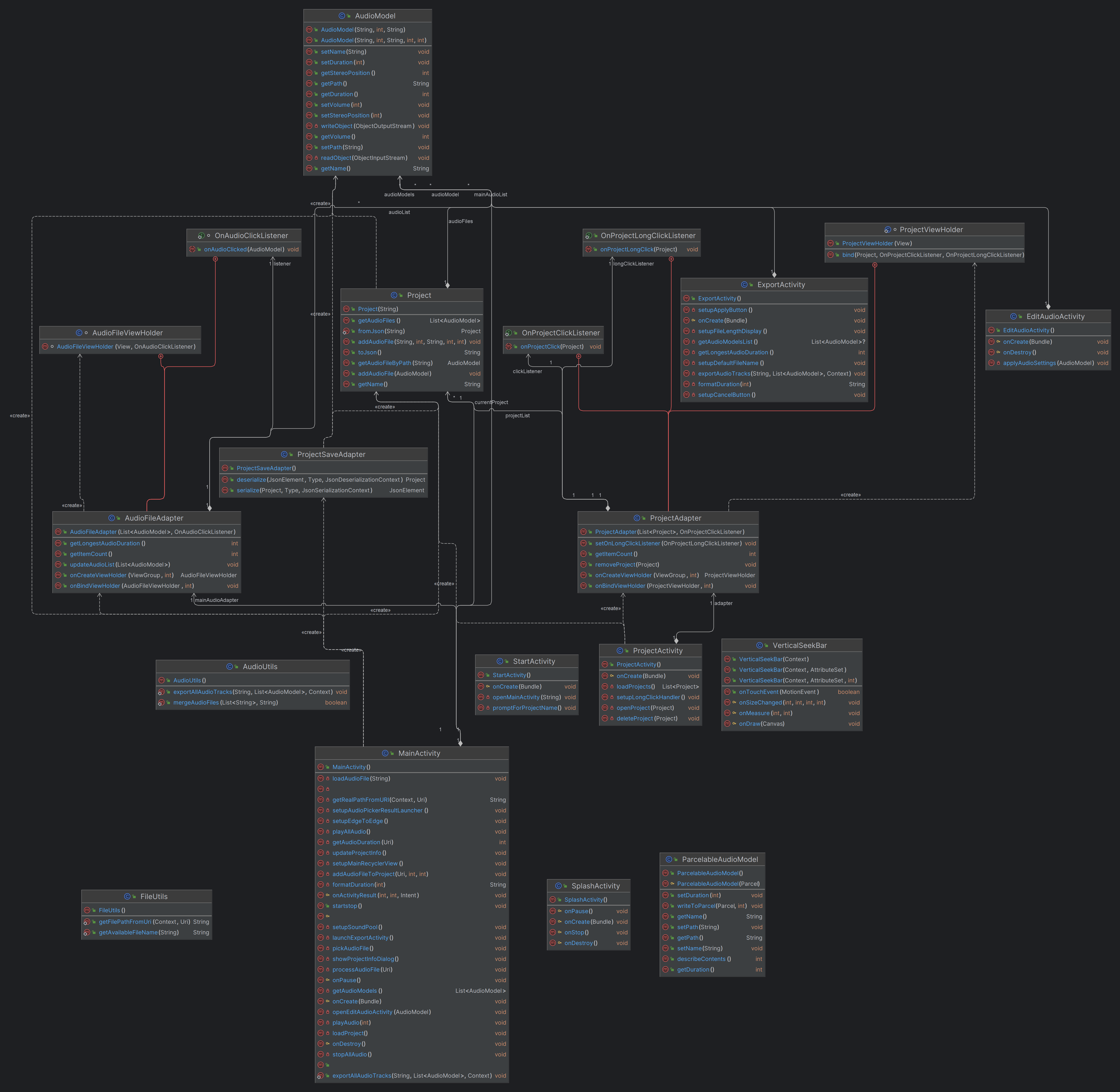


Рисунок 1 – UML диаграмма классов приложения

1. Описание разработки приложения

2.1 Обоснование средств разработки

Среда разработки: Android Studio

Android Studio – это официальная среда разработки для создания мобильных приложений на платформе Android, разработанная Google. Она предоставляет богатый набор инструментов, специально ориентированных на разработку приложений для Android.

Плюсы:

Официальная IDE для Android: Android Studio – основной инструмент, рекомендуемый Google для разработки Android-приложений, обеспечивая полную совместимость с платформой.

Встроенный эмулятор: Предоставляет мощный эмулятор Android, позволяющий быстро тестировать приложения на разных устройствах и версиях Android.

Интеграция с инструментами Google: Поддерживает интеграцию с сервисами Google, такими как Firebase, Google Cloud Platform и Google Play Services.

Множество инструментов разработки: Включает редактор кода, отладчик, профайлер производительности, анализатор памяти и другие инструменты, необходимые для качественной разработки.

Активное сообщество разработчиков: Большое сообщество разработчиков Android обеспечивает доступ к многочисленным ресурсам, плагинам и расширениям, которые могут ускорить процесс разработки.

Минусы:

Высокие системные требования: Android Studio может потреблять значительные ресурсы, особенно при запуске эмулятора и работе с большими проектами.

Может быть сложна для новичков: Из-за большого количества функций и инструментов, начальное обучение работе с Android Studio может занять время.

Иногда может быть нестабильной: Пользователи могут сталкиваться с задержками и сбоями, особенно при интенсивной работе.

Язык программирования: Java

Java – это широко используемый объектно-ориентированный язык программирования, известный своей надежностью и универсальностью. Он часто используется для разработки мобильных приложений, веб-сервисов и корпоративных систем.

Плюсы:

Надежность и стабильность: Java известна своим стабильным выполнением и обширной экосистемой библиотек и фреймворков.

Широкое применение: Java применяется в различных областях, включая разработку мобильных, веб- и корпоративных приложений.

Платформенная независимость: Java предоставляет возможность писать код, который будет работать на различных платформах благодаря Java Virtual Machine (JVM).

Большое сообщество: Широкое сообщество разработчиков обеспечивает доступ к множеству ресурсов, библиотек и фреймворков, что облегчает разработку.

Минусы:

Более медленное выполнение по сравнению с некоторыми другими языками: Java может быть менее эффективной по производительности, чем языки, компилируемые в нативный код.

Требует опыта для оптимизации: Оптимизация производительности в Java может потребовать специальных знаний и опыта.

Большие объемы памяти: Java-приложения могут требовать больше ресурсов, особенно для больших проектов или высокопроизводительных приложений.

2.2 Разработка интерфейса

Разработка пользовательского интерфейса является ключевым аспектом при создании приложения для создания музыки. Пользовательский интерфейс должен быть интуитивно понятным, удобным в использовании и привлекательным для пользователей, обеспечивая легкий доступ ко всем функциям приложения.

Анализ требований пользователя: Прежде чем приступить к разработке интерфейса, необходимо провести анализ потребностей и ожиданий пользователей. Это позволит определить основные функциональные требования к интерфейсу и оптимальную организацию его элементов.

Дизайн пользовательского интерфейса: На этапе проектирования пользовательского интерфейса создается макет приложения, который включает в себя расположение элементов управления, цветовую схему, типографику и другие детали. Дизайн должен быть согласован с общим стилем приложения и отражать его концепцию.

Использование Material Design: Для разработки пользовательского интерфейса рекомендуется использовать принципы Material Design. Material Design обеспечивает единообразие и согласованность интерфейса приложения, делая его более понятным и удобным для пользователей. Это также обеспечивает совместимость с различными устройствами и версиями Android.

Интерактивность и отзывчивость: Пользовательский интерфейс должен быть интерактивным и отзывчивым, обеспечивая мгновенное выполнение действий пользователя и быструю навигацию между различными экранами и функциями приложения. Это достигается за счет оптимизации процесса обработки пользовательских действий и использования анимаций для подчеркивания взаимодействия.

Тестирование пользовательского интерфейса: После разработки пользовательского интерфейса необходимо провести тестирование его на различных устройствах и разрешениях экрана. Это позволяет выявить и исправить возможные проблемы с отображением и взаимодействием интерфейса до выпуска приложения в продакшн.

Макеты экранов были разработаны с использованием графического инструмента Figma (См. список использованных источников).

2.3 Разработка логики работы приложения. Схема взаимодействия компонентов проекта

Процесс работы приложения основан на взаимодействии различных компонентов. Схема взаимодействия компонентов:

1. StartActivity:

При запуске приложения пользователь попадает сначала в стартовое окно.

Здесь пользователю предлагается выбрать создание нового проекта или открытие существующего.

Создание нового проекта:

Пользователь вводит имя проекта и переходит на MainActivity.

1. MainActivity:

В главном окне пользователь может добавлять аудиофайлы в основное рабочее пространство, используя кнопку "browse".

Может воспроизводить и останавливать воспроизведение аудиофайлов кнопкой "play" (все аудио играют одновременно).

Может просмотреть информацию о количестве аудиофайлов в проекте и максимальной длительности.

Может нажать кнопку "export" и перейти в ExportActivity для экспорта проекта.

Может нажать на "save" для сохранения проекта.

Может нажать на аудиофайл, чтобы перейти в EditAudioActivity.

1. ExportActivity:

Отображает итоговую длину файла и предоставляет возможность ввести имя файла для экспорта.

Содержит кнопку "apply" для экспорта файла и кнопку "cancel" для отмены.

1. EditAudioActivity:

Предоставляет возможность изменить отображаемое имя в проекте, изменить громкость и стереопозицию ползунком.

Содержит кнопки "apply" и "cancel".

1. Открытие проекта:

При нажатии на кнопку "открыть проект" в стартовом окне открывается список проектов.

Здесь пользователю показывается список сохраненных проектов, на которые можно нажать, чтобы открыть, и можно зажать для удаления.

2.4 Описание переменных, компонентов, классов и подпрограмм

Таблица 1 – Основные переменные класса MainActivity

|  |  |  |
| --- | --- | --- |
| Переменная | Тип | Описание |
| soundPool | SoundPool | Используется для управления аудиоресурсами, позволяет воспроизводить аудиофайлы. |
| MAX\_PLAYERS | int | Константа, определяющая максимальное количество аудиоплееров. |
| activeMediaPlayers | List<MediaPlayer> | Список активных аудиоплееров для управления воспроизведением звуков. |
| players | SimpleExoPlayer[] | Массив плееров для воспроизведения медиафайлов. |
| progressAnimator | ValueAnimator | Аниматор для управления анимациями прогресса в UI. |
| requestPermissionLauncher | ActivityResultLauncher<String> | Обработчик запроса разрешений в runtime. |
| mediaPlayer | MediaPlayer | Медиаплеер для воспроизведения аудиофайлов. |
| soundMap | Map<Integer, Integer> | Карта для управления звуковыми ID и связанными с ними ресурсами. |
| Переменная | Тип | Описание |
| mainAudioList | List<AudioModel> | Основной список аудиофайлов, с которыми работает пользователь. |
| isPlaying | boolean | Флаг, указывающий, воспроизводится ли аудио в данный момент. |
| buttonPlayStop | Button | Кнопка для управления воспроизведением: начать/остановить. |
| EDIT\_AUDIO\_REQUEST\_CODE | int | Код запроса для интента редактирования аудио. |
| totalAudioFiles | int | Общее количество аудиофайлов в проекте |
| longestDuration | int | Длительность самого длинного аудиофайла в проекте в секундах |
| projectName | String | Название текущего проекта. |
| temporaryAudioList | List<AudioModel> | Временный список аудиофайлов для операций, таких как добавление новых файлов в проект. |
| mainAudioAdapter | AudioFileAdapter | Адаптер для основного списка аудиофайлов. |
| mainRecyclerView | RecyclerView | RecyclerView для отображения основного списка аудиофайлов. |
| temporaryAdapter | AudioFileAdapter | Адаптер для временного списка аудиофайлов. |
| player | SimpleExoPlayer | Экземпляр плеера ExoPlayer для воспроизведения медиа. |
| mediaItems | List<MediaItem> | Список медиаэлементов для воспроизведения в плеере. |
| recyclerView | RecyclerView | RecyclerView для других UI элементов, например, в диалоговых окнах. |
| currentProject | Project | Текущий загруженный проект. |
| audioList | List<AudioModel> | Список аудиомоделей, используемых в текущей активности. |
| exportButton | Button | Кнопка для экспорта проекта. |
| audioPickerResultLauncher | ActivityResultLauncher<Intent> | Лаунчер для выбора аудиофайлов. |

Таблица 2 – Основные методы MainActivity.

|  |  |
| --- | --- |
| Метод | Описание |
| onCreate | Инициализация активности, установка UI, загрузка данных и прочие начальные настройки. |
| getAudioModels | Возвращает список аудиомоделей, который потенциально загружается из некоторого источника (пример показывает пустой список). |
| setupSoundPool | Настройка SoundPool для управления аудиоресурсами приложения. |
| startstop | Управление воспроизведением аудио: начать или остановить в зависимости от текущего состояния. |
| loadAudioFile | Загружает аудиофайл в SoundPool по указанному пути файла. |
| onPause | Обработчик жизненного цикла Activity, вызывается при паузе Activity. |
| onStop | Обработчик жизненного цикла Activity, вызывается при остановке Activity. |
| onDestroy | Обработчик жизненного цикла Activity, вызывается при уничтожении Activity. Освобождает ресурсы. |
| updateProjectInfo | Обновляет информацию о проекте, например, общее количество и продолжительность аудиофайлов. |
| showProjectInfoDialog | Отображает диалоговое окно с информацией о текущем проекте. |
| formatDuration | Форматирует продолжительность времени из секунд в часы, минуты и секунды. |
| loadProject | Загружает проект из файла, восстанавливая его состояние. |
| setupMainRecyclerView | Настройка RecyclerView для отображения списка аудиофайлов. |
| openEditAudioActivity | Открывает активность для редактирования аудиофайла. |
| onActivityResult | Обрабатывает результаты возвращенные из других Activity, например, из активности редактирования аудио. |
| showBrowseDialog | Показывает диалоговое окно для выбора аудиофайлов. |
| saveProject | Сохраняет текущий проект в файл. |
| pickAudioFile | Запускает выбор аудиофайлов через системный интерфейс. |
| getAudioDuration | Возвращает продолжительность аудиофайла в секундах. |
| setupEdgeToEdge | Настраивает окно для отображения в полноэкранном режиме без ограничений. |
| playAudio | Воспроизводит аудио по ID, используя SoundPool. |
| playAllAudio | Воспроизводит все аудиофайлы в списке. |
| launchExportActivity | Запускает активность для экспорта аудио. |
| stopAllAudio | Останавливает воспроизведение всех аудиофайлов и освобождает ресурсы. |
| setupAudioPickerResultLauncher | Настраивает лаунчер для результатов выбора аудиофайлов. |
| addAudioFileToProject | Добавляет аудиофайл в проект, устанавливая громкость и стереопозицию. |
| processAudioFile | Обрабатывает добавленный аудиофайл, обновляя информацию в проекте. |
| exportAllAudioTracks | Экспортирует все аудиотреки в один файл, используя FFmpeg. |
| getRealPathFromURI | Возвращает реальный путь к файлу по его URI. |

Таблица 3 – Основные переменные StartActivity

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип | Описание |
| savedInstanceState | Bundle | Содержит информацию о состоянии, передаваемую методу onCreate при воссоздании активности после ее уничтожения. |
| openButton | Button | Ссылка на элемент кнопки с ID buttonOpen в разметке активности. |
| btnNew | Button | Ссылка на элемент кнопки с ID buttonNew в разметке активности. |
| intent (в обработчике события нажатия openButton) | Intent | Объект класса Intent, используемый для перехода к ProjectActivity. |
| builder (в promptForProjectName) | AlertDialog.Builder | Используется для создания диалогового окна с запросом имени проекта. |
| input (в promptForProjectName в обработчике события нажатия кнопки "OK") | EditText | Ссылка на текстовое поле ввода в диалоговом окне, где пользователь вводит имя проекта. |
| projectName (в promptForProjectName в обработчике события нажатия кнопки "OK") | String | Хранит имя проекта, введенное пользователем в диалоговом окне. |
| intent (в openMainActivity) | Intent | Еще один объект класса Intent, используемый для перехода к MainActivity. |

Таблица 4 – Основные методы класса StartActivity

|  |  |
| --- | --- |
| Название метода | Описание |
| onCreate(Bundle savedInstanceState) | Метод жизненного цикла, вызываемый при первом создании активности. Он отвечает за инициализацию состояния и разметки активности. |
| promptForProjectName() | Этот метод создает диалоговое окно, запрашивающее у пользователя ввод имени проекта. Он обрабатывает пользовательский ввод и проверяет, указано ли имя проекта. |
| openMainActivity(String projectName) | Этот метод создает новый Intent для перехода к MainActivity и передает указанное имя проекта в качестве дополнительного значения. |

Таблица 5 – Основные переменные класса ProjectActivity

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип | Описание |
| recyclerViewProjects | RecyclerView | Ссылка на компонент списка проектов. |
| adapter | ProjectAdapter | Адаптер, связывающий данные о проектах с RecyclerView. |

Таблица 6 – Основные методы класса ProjectActivity

|  |  |
| --- | --- |
| Название метода | Описание |
| onCreate(Bundle savedInstanceState) | Метод жизненного цикла, вызываемый при первом создании активности. Он отвечает за инициализацию разметки, списка проектов и обработку нажатий. |
| setupLongClickHandler() | Устанавливает обработчик долгого нажатия на элемент списка проектов для удаления. |
| deleteProject(Project project) | Удаляет указанный проект из списка и внутреннего хранилища приложения. |
| loadProjects() | Загружает список проектов из внутреннего хранилища приложения. |
| openProject(Project project) | Открывает выбранный проект, переходя к MainActivity и передавая его название. |

Таблица 7 – Основные переменные класса ExportActivity

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип | Описание |
| fileLength | TextView | Ссылка на текстовый элемент, отображающий общую продолжительность аудиофайлов. |
| exportFileName | EditText | Ссылка на поле ввода для имени экспортируемого файла. |
| cancelButton | ImageButton | Ссылка на кнопку отмены экспорта. |
| applyButton | ImageButton | Ссылка на кнопку применения настроек экспорта. |
| audioModels | List<AudioModel> | Список объектов AudioModel, содержащих информацию об аудиофайлах для экспорта (получается из Intent). |
| REQUEST\_STORAGE\_PERMISSION | int | Код запроса разрешения на доступ к хранилищу. |

Таблица 8 – Основные методы класса ExportActivity

|  |  |
| --- | --- |
| Название метода | Описание |
| onCreate(Bundle savedInstanceState) | Метод жизненного цикла, вызываемый при первом создании активности. Он отвечает за инициализацию разметки, получение списка аудиофайлов, отображение общей продолжительности и настройку кнопок. |
| getAudioModelsList() | Извлекает список объектов AudioModel из переданного Intent. |
| setupFileLengthDisplay() | Вычисляет и отображает общую продолжительность аудиофайлов в списке audioModels. |
| getLongestAudioDuration() | Находит аудиофайл с наибольшей продолжительностью из списка audioModels. |
| formatDuration(int totalSecs) | Форматирует продолжительность в секундах в строку вида "чч:мм:сс". |
| setupDefaultFileName() | Устанавливает уникальное имя файла экспорта по умолчанию. |
| setupCancelButton() | Определяет действие кнопки отмены - закрытие активности. |
| setupApplyButton() | Определяет действие кнопки применения - запуск экспорта аудиофайлов. |
| exportAudioTracks(String fileName, List<AudioModel> audioList, Context context) (commented out) | Псевдоним метода (вероятно, не реализован в данном классе). Предполагается, что он выполняет объединение аудиодорожек, применение настроек и экспорт финального файла. |

Таблица 9 – Основные переменные класса AudioUtils

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип | Описание |
| models | List<AudioModel> | Список объектов AudioModel, содержащих информацию об аудиофайлах для экспорта. |
| musicDir | File | Ссылка на директорию для сохранения экспортируемого файла. |
| outputFile | File | Ссылка на файл, куда будет сохранен объединенный аудиофайл. |
| filePath | String | Абсолютный путь к выходному файлу. |
| inputs | StringBuilder | Строка-построитель для формирования списка входных файлов в команде FFmpeg. |
| filterComplex | StringBuilder | Строка-построитель для фильтра filter\_complex в команде FFmpeg, отвечающего за регулировку громкости и объединение аудиодорожек. |

Таблица 10 – Основные методы класса AudioUtils

|  |  |
| --- | --- |
| Название метода | Описание |
| exportAllAudioTracks(String fileName, List<AudioModel> models, Context context) | Экспортирует все аудиофайлы из списка models в один объединенный файл fileName.mp3 с применением настроек громкости. Использует библиотеку Mobile FFmpeg для выполнения команд FFmpeg. |
| mergeAudioFiles(List<String> audioPaths, String outputFile) (commented out) | Псевдоним метода (вероятно, не реализован в данном классе). Предполагается, что он выполняет программное объединение аудиофайлов из списка audioPaths в выходной файл outputFile на более низком уровне, используя классы MediaExtractor, MediaCodec, и MediaMuxer напрямую. |

Таблица 11 – Основные переменные класса FileUtils

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип | Описание |
| directory | File | Ссылка на директорию приложения во внешнем хранилище. |
| fileName | String | Базовое имя файла для поиска уникального названия. |
| file | File | Объект файла с проверяемым именем. |
| count | int | Счетчик для создания уникального имени файла. |
| inputStream | InputStream | Поток ввода для чтения данных из Uri. |
| outputStream | OutputStream | Поток вывода для записи временного файла. |
| filePath | String | Абсолютный путь к скопированному временному файлу. |

Таблица 12 – Основные методы класса FileUtils

|  |  |
| --- | --- |
| Название метода | Описание |
| getAvailableFileName(String baseName) | Генерирует уникальное имя файла на основе baseName в директории приложения внешнего хранилища. |
| getFilePathFromUri(Context context, Uri uri) | Копирует файл по Uri во временный файл во внутреннем хранилище приложения и возвращает его абсолютный путь. |

Таблица 13 – Основные переменные AudioFileAdapter

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип | Описание |
| audioList | List<AudioModel> | Список объектов AudioModel, содержащих информацию об аудиофайлах. |
| listener | OnAudioClickListener | Ссылка на слушателя, реагирующего на нажатие элемента списка. |

Таблица 14 – Основные методы AudioFileAdapter

|  |  |
| --- | --- |
| Название метода | Описание |
| AudioFileAdapter(List<AudioModel> audioList, OnAudioClickListener listener) (constructor) | Инициализирует адаптер списком аудиофайлов audioList и слушателем listener. |
| onCreateViewHolder(@NonNull ViewGroup parent, int viewType) | Создает новый холдер элемента списка AudioFileViewHolder при необходимости отобразить новый элемент. |
| onBindViewHolder(@NonNull AudioFileViewHolder holder, int position) | Заполняет данными холдер элемента списка holder на указанной позиции position. |
| getItemCount() | Возвращает количество элементов в списке. |
| AudioFileViewHolder(View itemView, OnAudioClickListener listener) (constructor) | Инициализирует холдер элемента списка AudioFileViewHolder. |
| getLongestAudioDuration() (not in inner class) | Не относится к внутреннему классу Находит аудиофайл с наибольшей продолжительностью из списка audioList. |
| updateAudioList(List<AudioModel> newAudioList) | Обновляет список аудиофайлов audioList новыми данными newAudioList и уведомляет прикрепленный адаптер об изменении данных. |

Таблица 15 – Основные переменные AudioModel

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип | Описание |
| name | String | Название аудиофайла. |
| duration | int | Длительность аудиофайла в секундах. |
| path | String | Путь к аудиофайлу на устройстве. |
| volume | int | Громкость аудиофайла (вероятно, в процентах). |
| stereoPosition | int | Позиционирование в стереопанораме (значение не совсем понятно из данного кода). |
| serialVersionUID | long | Уникальный идентификатор версии для сериализации объекта AudioModel. |

Таблица 16 – Основные методы AudioModel

|  |  |
| --- | --- |
| Название метода | Описание |
| AudioModel(String name, int duration, String path) (constructor) | Конструктор класса AudioModel, инициализирующий объект базовыми значениями громкости (70%) и стереопозиционирования (50%). |
| AudioModel(String name, int duration, String path, int volume, int stereoPosition) (constructor) | Конструктор класса AudioModel, инициализирующий объект всеми параметрами. |
| getName() | Геттер для поля name. |
| setName(String name) | Сеттер для поля name. |
| getDuration() | Геттер для поля duration. |
| setDuration(int duration) | Сеттер для поля duration. |
| getPath() | Геттер для поля path. |
| setPath(String path) | Сеттер для поля path. |
| getVolume() | Геттер для поля volume. |
| setVolume(int volume) | Сеттер для поля volume. |
| getStereoPosition() | Геттер для поля stereoPosition. |
| setStereoPosition(int stereoPosition) | Сеттер для поля stereoPosition. |
| writeObject(ObjectOutputStream out) (private) | Переопределенный метод сериализации объекта AudioModel. |
| readObject(ObjectInputStream in) (private) | Переопределенный метод десериализации объекта AudioModel. |

Таблица 17 – переменные EditAudioActivity

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип | Описание |
| audioModel | AudioModel | Объект, содержащий информацию об редактируемом аудиофайле. |
| nameEditText | EditText | Поле ввода для изменения названия аудиофайла. |
| volumeSeekBar | SeekBar | Ползунок для регулировки громкости аудиофайла. |
| stereoSeekBar | SeekBar | Ползунок для регулировки стереопозиционирования аудиофайла. |
| mediaPlayer | MediaPlayer | Медиаплеер для проигрывания редактируемого аудиофайла. |

Таблица 18 – методы editAudioActivity

|  |  |
| --- | --- |
| Название метода | Описание |
| onCreate(Bundle savedInstanceState) | Вызывается при создании активности. Инициализирует переменные на основе полученных данных Intent. |
| applyAudioSettings(AudioModel audioModel) | Применяет настройки громкости и стереопозиционирования из объекта audioModel к mediaPlayer. |
| onDestroy() | Вызывается при уничтожении активности. Освобождает ресурсы mediaPlayer. |

Таблица 19 – переменные ProjectAdapter

|  |  |  |
| --- | --- | --- |
| Переменная | Тип | Описание |
| projectList | List<Project> | Список объектов Project, содержащий информацию о проектах. |
| clickListener | OnProjectClickListener | Ссылка на слушателя интерфейса, реагирующего на короткое нажатие элемента проекта. |
| longClickListener | OnProjectLongClickListener | Ссылка на слушателя интерфейса, реагирующего на долгое нажатие элемента проекта (может быть null). |

Таблица 20 – методы ProjectAdapter

|  |  |
| --- | --- |
| Метод | Описание |
| ProjectAdapter(List<Project> projectList, OnProjectClickListener clickListener) (конструктор) | Инициализирует адаптер списком проектов projectList и слушателем clickListener для коротких нажатий. |
| setOnLongClickListener(OnProjectLongClickListener longClickListener) | Устанавливает слушателя longClickListener для обработки долгих нажатий на элементы проекта. |
| removeProject(Project project) | Удаляет проект project из списка projectList и уведомляет адаптер об изменениях. |
| onCreateViewHolder(@NonNull ViewGroup parent, int viewType) | Создает новый холдер элемента списка ProjectViewHolder для отображения элемента проекта. |
| onBindViewHolder(@NonNull ProjectViewHolder holder, int position) | Заполняет данными холдер элемента списка holder на указанной позиции position. |
| getItemCount() | Возвращает количество элементов (проектов) в списке. |

Таблица 21 – Внутренний класс ProjectViewHolder

|  |  |
| --- | --- |
| Метод | Описание |
| ProjectViewHolder(View itemView) (конструктор) | Инициализирует внутренний класс ProjectViewHolder. |
| bind(Project project, OnProjectClickListener clickListener, OnProjectLongClickListener longClickListener) | Связывает данные проекта project с текстовым элементом textView во вью холдере, устанавливает слушателей clickListener и longClickListener для обработки нажатий. |

Таблица 22 – переменные Project

|  |  |  |
| --- | --- | --- |
| Переменная | Тип | Описание |
| name | String | Название проекта. |
| audioFiles | List<AudioModel> | Список объектов AudioModel, содержащий информацию об аудиофайлах, входящих в проект. |
| creationDate | String | Дата создания проекта (строковое представление). |
| lastModifiedDate | String | Дата последнего изменения проекта (строковое представление). |

Таблица 23 – методы Project

|  |  |
| --- | --- |
| Метод | Описание |
| Project(String name) (конструктор) | Создает новый проект с указанным названием name. Инициализирует пустой список аудиофайлов audioFiles и устанавливает текущую дату для creationDate. |
| addAudioFile(AudioModel audio) | Добавляет существующий объект AudioModel в список аудиофайлов проекта audioFiles. |
| addAudioFile(String name, int duration, String path, int volume, int stereoPosition) | Создает новый объект AudioModel с указанными параметрами и добавляет его в список аудиофайлов проекта audioFiles. |
| getName() | Возвращает название проекта name. |
| getAudioFiles() | Возвращает неизменяемый список объектов AudioModel из проекта audioFiles. |
| toJson() | Сериализует объект проекта Project в строку JSON-формата. |
| fromJson(String json) (статический) | Десериализует строку JSON-формата в объект проекта Project. |
| getAudioFileByPath(String path) | Ищет аудиофайл по его пути path в списке audioFiles. Возвращает найденный объект AudioModel или null, если файл не найден. |

Таблица 24 – методы ProjectSaveAdapter

|  |  |
| --- | --- |
| Метод | Описание |
| serialize(Project src, Type typeOfSrc, JsonSerializationContext context) | Сериализует объект Project (src) в элемент JSON (JsonElement). |
| - src: Объект проекта для сериализации. | Создает объект JSON, содержащий: |
| - typeOfSrc: Тип объекта src. | - "name": Название проекта. |
| - context: Контекст сериализации Gson. | - "audioFiles": Массив JSON-объектов, каждый из которых представляет аудиофайл. |
|  | - "name": Название аудиофайла. |
|  | - "duration": Длительность аудиофайла в секундах. |
|  | - "path": Путь к аудиофайлу. |
|  | - "volume": Громкость аудиофайла (вероятно, в процентах). |
|  | - "stereoPosition": Позиционирование в стереопанораме (значение не совсем понятно из данного кода). |
| deserialize(JsonElement json, Type typeOfT, JsonDeserializationContext context) | Десериализует элемент JSON (json) в объект Project. |
| - json: Элемент JSON, содержащий данные проекта. | Парсит JSON-элемент и создает новый объект Project. |
| - typeOfT: Ожидаемый тип объекта (должен быть Project). | - Извлекает имя проекта из JSON-объекта. |
| - context: Контекст десериализации Gson. | - Извлекает массив JSON-объектов, представляющих аудиофайлы. |
|  | - Для каждого JSON-объекта аудиофайла: |
|  | - Создает новый объект AudioModel с данными из JSON-объекта. |
|  | - Добавляет AudioModel в список аудиофайлов проекта. |
|  | Возвращает созданный объект Project. |

Таблица 25 - Таблица метода onCreate(Bundle savedInstanceState)

|  |  |
| --- | --- |
| Шаг | Описание |
| 1 | Скрытие заголовка окна: requestWindowFeature(Window.FEATURE\_NO\_TITLE) |
| 2 | Скрытие статус бара: |
|  | - getWindow().setFlags(...) |
|  | - getWindow().getDecorView().setSystemUiVisibility(...) |
| 3 | Установка разметки активности: setContentView(R.layout.activity\_splash) |
| 4 | Задержка 3 секунды и переход к StartActivity: |
|  | - new Handler().postDelayed(...) |
|  | - Intent intent = new Intent(SplashActivity.this, StartActivity.class) |
|  | - startActivity(intent) |
|  | - finish() |

1. Тестирование и установка приложения

3.1 Сценарии тестирования:

Вход в приложение:

1. Запуск приложения:
   1. Запустите приложение INLOOPER.
   2. Убедитесь, что приложение корректно запустилось и отображается экран главного меню.
2. Переход к созданию музыки:
   1. Нажмите кнопку "NEW" в главном меню.
   2. Убедитесь, что происходит переход на экран создания музыки.
3. Выход из приложения:
   1. Выйдите навигационными кнопками из главного меню.
   2. Убедитесь, что приложение закрывается корректно.

2. Отображение экрана создания музыки:

1. Элементы экрана:
   1. Убедитесь, что на экране создания музыки отображаются все необходимые элементы:
      1. Панель инструментов (с кнопками управления)
      2. Окно для отображения трека
   2. Проверьте, что все элементы корректно отображаются (размер, цвета, шрифты).
2. Взаимодействие с элементами:
   1. Проверьте, что кнопки управления на панели инструментов реагируют на нажатия:
      1. Воспроизведение/пауза
      2. Добавить аудиофайл
      3. Посмотреть информацию о проекте
      4. Экспортировать проект
      5. Сохранить проект
   2. Проверьте, что функциональность кнопок работает корректно:
      1. При воспроизведении/паузе трек корректно стартует/останавливается
      2. При добавлении аудиофайла открывается проводник
      3. В диалоговом окне информация соответствует действительности
      4. При нажатии на кнопку экспорта открывается окно для настройки итогового файла
      5. При нажатии на кнопку сохранения проект сохраняется
3. Добавление аудиофайлов:
   1. Нажмите кнопку "Browse" на панели инструментов.
   2. Убедитесь, что открывается список доступных аудиофайлов.
   3. Выберите инструмент из списка или добавьте новый.
   4. Убедитесь, что выбранный инструмент добавляется в панель для управления параметрами аудиофайлов.
4. Управление параметрами аудиофайлов:
   1. Выберите инструмент на панели управления.
   2. Проверьте, что отображаются параметры выбранного инструмента.
   3. Измените параметры аудиофайла (например, громкость, панорама).
   4. Убедитесь, что изменения параметров аудиофайла отражаются в звучании.
5. Сохранение и загрузка трека:
   1. Нажмите кнопку "Save" на панели инструментов
   2. Убедитесь, что проект корректно сохраняется
   3. Нажмите на сохраненный проект в списке проектов
   4. Выберите сохраненный ранее проект
   5. Убедитесь, что проект корректно загружается и воспроизводится.

3.2 Предполагаемые ошибки:

1. Ошибка воспроизведения:
   1. При воспроизведении трека могут возникать заикания или прерывания.
2. Ошибка сохранения:
   1. При сохранении трека могут возникать ошибки (например, недостаточно места на диске).
3. Ошибка загрузки:
   1. При загрузке трека могут возникать ошибки (например, файл поврежден).

3.3 Установка приложения:

Для установки приложения INLOOPER на компьютер:

1. Загрузите установочный файл:
   1. Перейдите в GithHub разработчика INLOOPER (См. аннотацию).
   2. Загрузите установочный файл.
2. Запустите установщик:
   1. Кликните по установочному файлу.
   2. Дождитесь конца загрузки.
3. Завершение установки:
   1. Нажмите кнопку "установить".
   2. Дождитесь завершения установки.
   3. Запустите INLOOPER с помощью иконки или из каталога приложений.

Определения, сокращения и аббревиатуры

FFmpeg: Это свободная и открытая кросс-платформенная библиотека программ, которая позволяет записывать, конвертировать и потоково передавать аудио и видео. FFmpeg является ключевым инструментом в многих приложениях обработки медиафайлов и используется для поддержки различных кодеков, форматов и протоколов в обработке медиа. Она включает в себя libavcodec (библиотека кодеков), libavformat (библиотека форматов контейнеров), libavutil (вспомогательная утилита) и другие компоненты.

Аматар: Предположительно опечатка, возможно имелось в виду "аматор" или "аматер", что означает непрофессионал, любитель. В контексте музыкального приложения это может относиться к пользователям, которые не являются профессиональными музыкантами, но интересуются созданием музыки как хобби.

Продакшн (Production): В контексте разработки программного обеспечения, термин "продакшн" относится к фазе, в которой программное обеспечение или приложение уже полностью разработано, протестировано и готово к использованию конечными пользователями. Это окончательная версия продукта, предназначенная для эксплуатации в реальных условиях. В других контекстах, таких как кино или телевидение, продакшн означает производство — процесс создания фильмов, видео или других медиапродуктов.

Android Studio: Интегрированная среда разработки для Android-приложений, используемая для кодирования, тестирования и отладки.

Java: Основной язык программирования, используемый для разработки приложений Android.

(Loops) — короткие, повторяющиеся музыкальные фрагменты.

Сэмплы (Samples) — звуковые фрагменты, используемые для создания или модификации музыкальных треков.

Заключение

Разработка мобильного приложения для создания и редактирования музыкальных композиций через сэмплирование представляет собой актуальную задачу, способствующую расширению инструментария как профессиональных музыкантов, так и аматоров. В рамках данной курсовой работы было успешно разработано приложение "INLOOPER", предназначенное для работы с аудиофайлами: их воспроизведения, редактирования, смешивания и создания новых звуковых треков.

В процессе разработки были изучены и успешно применены современные технологии и инструменты, включая Android Studio, Java, библиотеку FFmpeg для обработки аудио, а также элементы Material Design для оформления пользовательского интерфейса. Это позволило создать удобное и функциональное приложение, соответствующее современным стандартам мобильных приложений.

Основное внимание в проекте было уделено пользовательскому интерфейсу, который позволяет легко управлять процессом создания музыки, а также функциональности по манипуляции с аудиофайлами. Разработанный функционал включает загрузку, редактирование, смешивание и сохранение аудиофайлов, что делает приложение максимально полезным для целевой аудитории.

Тестирование приложения показало его высокую стабильность и производительность. Возможности расширения функционала и оптимизации приложения остаются актуальными направлениями для дальнейшего развития проекта.

Результаты данной работы могут быть использованы в дальнейших исследованиях и проектах, связанных с разработкой мобильных приложений для обработки и создания музыки. Перспективы развития проекта включают интеграцию с облачными сервисами для хранения аудиоданных, использование искусственного интеллекта для автоматизации создания музыки и расширение количества поддерживаемых аудиоформатов.

В заключение, проект "INLOOPER" успешно достиг поставленных целей и задач, демонстрируя эффективность выбранных методов и технологий в области мобильной разработки. Это делает его значимым вкладом в область музыкальных технологий и мобильной разработки.

Начало формы

Список использованной литературы

1. Оформление курсовой работы по ГОСТу 2024 + образец [Электронный ресурс]. – режим доступа: [Общие требования к курсовой работе](https://retext.ai/ru/blog/oformlenie-kursovoj-rabotyi-po-gost)
2. Разработка мобильных приложений для платформы Android [Электронный ресурс]. – режим доступа: [20130403.pdf (uniyar.ac.ru)](http://www.lib.uniyar.ac.ru/edocs/iuni/20130403.pdf)
3. ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению [Электронный ресурс]. – режим доступа: [ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению](https://www.prj-exp.ru/gost/gost_19-201-78.php)
4. Интеграционное тестирование: что это? Виды, примеры. [Электронный ресурс]. – режим доступа: [Интеграционное тестирование](https://logrocon.ru/news/intgration_testing)
5. FFmpeg documentation [Электронный ресурс]. – режим доступа: [Documentation (ffmpeg.org)](https://ffmpeg.org/documentation.html)
6. Учебник: программирование на java [Электронный ресурс]. – режим доступа: [Учебник: программирование на Java - Бесплатное онлайн обучение программированию на языке Java (java9.ru)](https://java9.ru/)

Приложение А. Код приложения

package com.example.mbhere;

import static com.arthenica.mobileffmpeg.Config.RETURN\_CODE\_SUCCESS;

import static com.example.mbhere.AudioUtils.exportAllAudioTracks;

import android.Manifest;

import android.animation.ValueAnimator;

import android.app.Activity;

import android.app.AlertDialog;

import android.app.Dialog;

import android.content.Context;

import android.content.DialogInterface;

import android.content.Intent;

import android.content.pm.PackageManager;

// import android.media.AudioAttributes;

import android.database.Cursor;

import android.media.AudioManager;

import com.arthenica.mobileffmpeg.FFmpeg;

import android.media.SoundPool;

import android.net.Uri;

import android.os.Build;

import android.os.Bundle;

import android.os.Parcelable;

import android.provider.MediaStore;

import android.util.Log;

import android.view.View;

import android.view.Window;

import android.view.WindowManager;

import android.widget.Button;

import android.widget.ImageButton;

import android.widget.Toast;

import com.google.android.exoplayer2.C;

import com.google.android.exoplayer2.ExoPlayer;

import com.google.android.exoplayer2.MediaItem;

import com.google.android.exoplayer2.SimpleExoPlayer;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.InputStreamReader;

import java.io.ObjectOutputStream;

import java.io.IOException;

import android.media.MediaPlayer;

import android.net.Uri;

import androidx.activity.result.ActivityResultLauncher;

import androidx.activity.result.contract.ActivityResultContracts;

import androidx.annotation.NonNull;

import androidx.annotation.Nullable;

import androidx.appcompat.app.AppCompatActivity;

import androidx.core.app.ActivityCompat;

import androidx.core.content.ContextCompat;

import androidx.recyclerview.widget.LinearLayoutManager;

import androidx.recyclerview.widget.RecyclerView;

import java.io.OutputStreamWriter;

import java.io.Serializable;

import java.nio.charset.StandardCharsets;

import java.util.ArrayList;

import java.util.Collections;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import android.media.MediaMetadataRetriever;

import com.google.android.exoplayer2.C;

import com.google.android.exoplayer2.audio.AudioAttributes;

import com.google.android.exoplayer2.SimpleExoPlayer;

import com.google.gson.Gson;

import com.google.gson.GsonBuilder;

import com.google.gson.stream.JsonReader;

import com.google.gson.stream.JsonToken;

public class MainActivity extends AppCompatActivity {

// Пул звуковых эффектов для воспроизведения аудиофайлов.

private SoundPool soundPool;

// Максимальное количество одновременно воспроизводимых аудиоплееров.

private static final int MAX\_PLAYERS = 10;

// Список активных аудио плееров для управления воспроизведением.

private List<MediaPlayer> activeMediaPlayers = new ArrayList<>();

// Массив плееров ExoPlayer, каждый из которых может управлять своим воспроизведением.

private SimpleExoPlayer[] players = new SimpleExoPlayer[MAX\_PLAYERS];

// Аниматор для анимации прогресса воспроизведения или других визуальных эффектов.

private ValueAnimator progressAnimator;

// Запускатель активностей с запросом разрешений от пользователя.

private ActivityResultLauncher<String> requestPermissionLauncher;

// Объект MediaPlayer для управления медиа воспроизведением.

private MediaPlayer mediaPlayer;

// Словарь для хранения маппинга звуковых ID и их характеристик.

private Map<Integer, Integer> soundMap = new HashMap<>();

// Список моделей аудиофайлов, используемых в текущем проекте.

private List<AudioModel> mainAudioList = new ArrayList<>();

// Флаг воспроизведения, указывающий, идет ли воспроизведение в данный момент.

private boolean isPlaying = false;

// Кнопка для управления воспроизведением (пуск/пауза).

private Button buttonPlayStop;

// Код запроса для редактирования аудио, используется в активности редактирования.

private static final int EDIT\_AUDIO\_REQUEST\_CODE = 1001;

// Примеры данных для демонстрации, которые должны быть заменены реальными данными.

private int totalAudioFiles = 5; // Пример, замените на реальные данные

private int longestDuration = 120; // Пример, в секундах, замените на реальные данные

// Название текущего проекта, дата создания и последнего изменения.

private String projectName;

private String creationDate;

private String lastModifiedDate;

// Визуальный компонент для отображения прогресса воспроизведения.

private View verticalProgress;

// Длительность анимации прогресса.

private int progressAnimatorDuration = 5000; // В миллисекундах

// Временный список аудиомоделей для работы с данными в диалоговых окнах и т.д.

private List<AudioModel> temporaryAudioList = new ArrayList<>();

// Адаптеры для работы с аудиофайлами в RecyclerView.

private AudioFileAdapter mainAudioAdapter, temporaryAdapter;

// Основной плеер ExoPlayer для воспроизведения медиа.

private SimpleExoPlayer player;

// Список медиаэлементов для воспроизведения.

private List<MediaItem> mediaItems = new ArrayList<>();

// RecyclerView для отображения списка проектов или аудиофайлов.

private RecyclerView recyclerView, mainRecyclerView;

// Текущий загруженный проект.

private Project currentProject;

// Список аудиомоделей, используемых в проекте.

private List<AudioModel> audioList;

// Кнопка для экспорта аудио.

private Button exportButton;

// Запускатель результатов для выбора аудиофайлов.

private ActivityResultLauncher<Intent> audioPickerResultLauncher;

// Метод вызывается при создании активности, здесь инициализируются компоненты и загружаются данные.

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

requestWindowFeature(Window.FEATURE\_NO\_TITLE);

getWindow().setFlags(WindowManager.LayoutParams.FLAG\_FULLSCREEN,

WindowManager.LayoutParams.FLAG\_FULLSCREEN);

getWindow().getDecorView().setSystemUiVisibility(View.SYSTEM\_UI\_FLAG\_HIDE\_NAVIGATION);

setContentView(R.layout.activity\_main);

setupEdgeToEdge();

setupMainRecyclerView();

setupSoundPool();

loadProject();

player = new SimpleExoPlayer.Builder(this).build();

// Инициализация лаунчера для запроса разрешения

requestPermissionLauncher = registerForActivityResult(new ActivityResultContracts.RequestPermission(), isGranted -> {

if (isGranted) {

// Разрешение было предоставлено, выполните необходимые действия

showBrowseDialog();

} else {

// Разрешение не было предоставлено, выполните соответствующие действия

Toast.makeText(MainActivity.this, "Permission Denied", Toast.LENGTH\_SHORT).show();

}

});

mediaPlayer = new MediaPlayer();

Intent intent = getIntent();

if (intent != null && intent.hasExtra("audioList")) {

List<AudioModel> audioModels = (List<AudioModel>) intent.getSerializableExtra("audioList");

Log.d("ExportActivity", "Received audioModels list: " + audioModels.size() + " items");

} else {

Log.d("ExportActivity", "No audioModels data received");

}

player.setMediaItems(mediaItems);

player.play();

String projectName = getIntent().getStringExtra("project\_name");

if (projectName != null) {

currentProject = new Project(projectName);

} else {

// Обрабатываем случай, когда имя проекта не передано (если такое возможно в вашем приложении)

Log.e("MainActivity", "Не было передано имя проекта");

finish(); // Закрываем Activity, если имя проекта не было передано

}

setupAudioPickerResultLauncher();

Button settingsButton = findViewById(R.id.buttonFileSettings);

settingsButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

showProjectInfoDialog(); // Показываем диалог с информацией о проекте

}

});

Button saveButton = findViewById(R.id.save\_button);

saveButton.setOnClickListener(v -> {

// Предположим, что у вас есть текущий объект проекта

if (currentProject != null) {

saveProject(currentProject);

} else {

Toast.makeText(this, "Проект не загружен", Toast.LENGTH\_SHORT).show();

}

});

audioList = getAudioModels(); // Здесь вы получаете ваш список аудиомоделей

exportButton = findViewById(R.id.buttonExport);

exportButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

launchExportActivity();

}

});

Button buttonBrowse = findViewById(R.id.buttonBrowse);

buttonBrowse.setOnClickListener(v -> {

if (ContextCompat.checkSelfPermission(this, Manifest.permission.READ\_EXTERNAL\_STORAGE) == PackageManager.PERMISSION\_GRANTED) {

showBrowseDialog();

} else {

requestPermissionLauncher.launch(Manifest.permission.READ\_EXTERNAL\_STORAGE);

}

});

audioPickerResultLauncher = registerForActivityResult(new ActivityResultContracts.StartActivityForResult(), result -> {

if (result.getResultCode() == Activity.RESULT\_OK && result.getData() != null) {

Uri audioUri = result.getData().getData();

// Запрос постоянного разрешения на доступ к файлу

final int takeFlags = Intent.FLAG\_GRANT\_READ\_URI\_PERMISSION | Intent.FLAG\_GRANT\_WRITE\_URI\_PERMISSION;

try {

getContentResolver().takePersistableUriPermission(audioUri, takeFlags);

} catch (SecurityException e) {

Log.e("MainActivity", "Security Exception: Failed to take persistable URI permission", e);

}

processAudioFile(audioUri);

}

});

requestPermissionLauncher = registerForActivityResult(new ActivityResultContracts.RequestPermission(), isGranted -> {

if (isGranted) {

showBrowseDialog();

} else {

Toast.makeText(MainActivity.this, "Permission Denied", Toast.LENGTH\_SHORT).show();

}

});

}

private List<AudioModel> getAudioModels() {

// Здесь ваш код для получения списка аудиомоделей

return new ArrayList<>(); // Пример

}

// Инициализация пула звуков для эффектов.

private void setupSoundPool() {

if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.LOLLIPOP) {

soundPool = new SoundPool.Builder().setMaxStreams(10).build();

} else {

soundPool = new SoundPool(10, AudioManager.STREAM\_MUSIC, 0);

}

}

// Настройка и запуск/остановка воспроизведения аудио по нажатию кнопки.

public void startstop() {

buttonPlayStop = findViewById(R.id.buttonPlayStop);

buttonPlayStop.setOnClickListener(v -> {

if (!isPlaying) {

playAllAudio();

buttonPlayStop.setText("Stop");

} else {

stopAllAudio();

buttonPlayStop.setText("Play");

}

});

}

// Загрузка аудиофайла в пул звуков.

private void loadAudioFile(String filePath) {

if (soundPool != null) {

File file = new File(filePath);

if (file.exists()) {

int soundId = soundPool.load(file.getAbsolutePath(), 1);

// Здесь вы можете делать что-то с идентификатором звука, если это нужно

// Например, сохранять его в Map для последующего использования

} else {

Log.e("SoundPool", "File not found: " + filePath);

}

}

}

// Обработка паузы активности.

@Override

protected void onPause() {

super.onPause();

Log.d("Lifecycle", "onPause called");

}

// Обработка остановки активности, остановка всех плееров.

@Override

protected void onStop() {

super.onStop();

for (SimpleExoPlayer player : players) {

if (player != null) {

player.stop();

}

}

}

// Обработка уничтожения активности, освобождение ресурсов.

@Override

protected void onDestroy() {

super.onDestroy();

stopAllAudio();

for (MediaPlayer mp : activeMediaPlayers) {

if (mp != null) {

mp.release();

}

}

activeMediaPlayers.clear();

}

// Обновление информации о проекте.

private void updateProjectInfo() {

totalAudioFiles = mainAudioList.size();

longestDuration = mainAudioList.stream()

.mapToInt(AudioModel::getDuration)

.max()

.orElse(0);

}

// Отображение диалога с информацией о проекте.

private void showProjectInfoDialog() {

updateProjectInfo(); // Обновляем информацию перед показом

AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);

builder.setTitle(projectName);

builder.setMessage("\t\t\tТЕКУЩИЙ ПРОЕКТ\n\n" + "Количество аудиофайлов: " + totalAudioFiles +

"\nИтоговая длина: " + formatDuration(longestDuration));

builder.setPositiveButton("Закрыть", new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

dialog.dismiss();

}

});

AlertDialog dialog = builder.create();

dialog.show();

}

// Форматирование длительности в часы, минуты и секунды.

private String formatDuration(int totalSecs) {

int hours = totalSecs / 3600;

int minutes = (totalSecs % 3600) / 60;

int seconds = totalSecs % 60;

return String.format("%02d:%02d:%02d", hours, minutes, seconds);

}

// Загрузка данных проекта.

private void loadProject() {

Log.d("MainActivity", "Loading project...");

String projectName = getIntent().getStringExtra("project\_name");

Log.d("MainActivity", "Loading project: " + projectName);

if (projectName != null) {

File projectFile = new File(getFilesDir(), "com.example.mbhere/" + projectName);

if (projectFile.exists()) {

try (FileReader reader = new FileReader(projectFile)) {

Gson gson = new GsonBuilder()

.registerTypeAdapter(Project.class, new ProjectSaveAdapter())

.create();

currentProject = gson.fromJson(reader, Project.class);

if (currentProject != null && currentProject.getAudioFiles() != null) {

mainAudioList.clear();

mainAudioList.addAll(currentProject.getAudioFiles());

mainAudioAdapter.notifyDataSetChanged();

// Применяем значения громкости и стереопозиции к каждому аудиофайлу

for (AudioModel audioModel : currentProject.getAudioFiles()) {

for (int i = 0; i < mainAudioList.size(); i++) {

if (mainAudioList.get(i).getPath().equals(audioModel.getPath())) {

mainAudioList.set(i, audioModel);

mainAudioAdapter.notifyItemChanged(i);

// Логирование изменённых данных

Log.d("MainActivity", "Audio updated: " + audioModel.getName() +

", Volume: " + audioModel.getVolume() +

", Stereo: " + audioModel.getStereoPosition());

break;

}

}

}

}

} catch (IOException e) {

Toast.makeText(this, "Не удалось загрузить проект: " + e.getMessage(), Toast.LENGTH\_SHORT).show();

}

} else {

}

} else {

Toast.makeText(this, "Название проекта не передано", Toast.LENGTH\_SHORT).show();

finish();

}

}

// Настройка RecyclerView для отображения списка аудиофайлов.

private void setupMainRecyclerView() {

startstop();

mainRecyclerView = findViewById(R.id.recycler\_view);

mainRecyclerView.setLayoutManager(new LinearLayoutManager(this));

mainAudioAdapter = new AudioFileAdapter(mainAudioList, this::openEditAudioActivity);

mainRecyclerView.setAdapter(mainAudioAdapter);

}

// Открытие активности для редактирования аудио.

private void openEditAudioActivity(AudioModel audioModel) {

// Создание интента для открытия EditAudioActivity и передача данных

Intent intent = new Intent(this, EditAudioActivity.class);

intent.putExtra("audioModel", audioModel);

startActivityForResult(intent, EDIT\_AUDIO\_REQUEST\_CODE);

}

private boolean isDataModified = false;

// Проверка на изменения данных после возвращения из другой активности.

@Override

protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {

super.onActivityResult(requestCode, resultCode, data);

if (requestCode == EDIT\_AUDIO\_REQUEST\_CODE && resultCode == RESULT\_OK && data != null) {

AudioModel updatedAudioModel = (AudioModel) data.getSerializableExtra("updatedAudioModel");

for (int i = 0; i < mainAudioList.size(); i++) {

if (mainAudioList.get(i).getPath().equals(updatedAudioModel.getPath())) {

mainAudioList.set(i, updatedAudioModel);

mainAudioAdapter.notifyItemChanged(i);

isDataModified = true; // Устанавливаем флаг

break;

}

}

}

}

// Отображение диалога для просмотра и выбора аудиофайлов.

private void showBrowseDialog() {

Dialog browseDialog = new Dialog(this);

browseDialog.setContentView(R.layout.dialog\_browse);

RecyclerView recyclerView = browseDialog.findViewById(R.id.recycler\_view\_browse);

recyclerView.setLayoutManager(new LinearLayoutManager(this));

temporaryAdapter = new AudioFileAdapter(temporaryAudioList, audioModel -> {

mainAudioList.add(0, audioModel); // Добавляем в начало списка

mainAudioAdapter.notifyItemInserted(0);

mainRecyclerView.scrollToPosition(0); // Прокручиваем к началу

currentProject.addAudioFile(audioModel); // Добавляем аудиофайл в объект проекта

browseDialog.dismiss(); // Закрываем диалог

});

recyclerView.setAdapter(temporaryAdapter);

ImageButton addButton = browseDialog.findViewById(R.id.buttonAddFile);

addButton.setOnClickListener(v -> pickAudioFile());

browseDialog.show();

}

// Сохранение проекта в локальное хранилище.

public void saveProject(Project project) {

Gson agson = new Gson();

for (AudioModel audio : project.getAudioFiles()) {

Log.d("SaveProject", "Audio: " + audio.getName() + ", Volume: " + audio.getVolume() + ", Stereo Position: " + audio.getStereoPosition());

}

String cjson = agson.toJson(project);

Log.d("SaveProject", "Serialized JSON: " + cjson);

File directory = new File(getFilesDir(), "com.example.mbhere");

if (!directory.exists()) {

directory.mkdirs();

}

File file = new File(directory, project.getName() + ".json");

try (FileWriter writer = new FileWriter(file)) {

Gson gson = new GsonBuilder()

.registerTypeAdapter(Project.class, new ProjectSaveAdapter())

.create();

// Логируем параметры каждого аудиофайла перед сериализацией

for (AudioModel audio : project.getAudioFiles()) {

Log.d("saveProject", "Аудиофайл: " + audio.getName());

Log.d("saveProject", "Путь: " + audio.getPath());

Log.d("saveProject", "Громкость: " + audio.getVolume());

Log.d("saveProject", "Стереопозиция: " + audio.getStereoPosition());

}

// Логируем содержимое объекта Project перед сериализацией

Log.d("saveProject", "Содержимое объекта Project перед сериализацией: " + project.toString());

String json = gson.toJson(project);

// Логируем сериализованную JSON-строку

Log.d("saveProject", "Сериализованная JSON-строка: " + json);

writer.write(json);

Toast.makeText(this, "Проект сохранён: " + file.getAbsolutePath(), Toast.LENGTH\_LONG).show();

} catch (IOException e) {

Toast.makeText(this, "Ошибка сохранения проекта: " + e.getMessage(), Toast.LENGTH\_SHORT).show();

}

}

// Выбор аудиофайла через системный интерфейс.

private void pickAudioFile() {

Intent intent = new Intent(Intent.ACTION\_OPEN\_DOCUMENT);

intent.addCategory(Intent.CATEGORY\_OPENABLE);

intent.setType("audio/\*");

intent.setFlags(Intent.FLAG\_GRANT\_READ\_URI\_PERMISSION

| Intent.FLAG\_GRANT\_WRITE\_URI\_PERMISSION

| Intent.FLAG\_GRANT\_PERSISTABLE\_URI\_PERMISSION); // Добавление флага для постоянного доступа

audioPickerResultLauncher.launch(intent);

}

// Получение длительности аудиофайла из его URI.

private int getAudioDuration(Uri uri) {

MediaMetadataRetriever retriever = new MediaMetadataRetriever();

retriever.setDataSource(this, uri);

String durationStr = retriever.extractMetadata(MediaMetadataRetriever.METADATA\_KEY\_DURATION);

return Integer.parseInt(durationStr) / 1000;

}

// Настройка безрамочного дизайна для активности.

private void setupEdgeToEdge() {

Window window = getWindow();

window.setFlags(WindowManager.LayoutParams.FLAG\_LAYOUT\_NO\_LIMITS, WindowManager.LayoutParams.FLAG\_LAYOUT\_NO\_LIMITS);

}

// Воспроизведение аудио по ID из SoundPool.

private void playAudio(int soundId) {

if (soundPool != null) {

soundPool.play(soundId, 1.0f, 1.0f, 1, 0, 1.0f);

}

}

// Воспроизведение всех аудиофайлов из списка.

private void playAllAudio() {

Log.d("PlayAudio", "Starting playback of all audio files");

if (mainAudioList.isEmpty()) {

Log.d("PlayAudio", "No audio files to play");

return;

}

for (AudioModel audio : mainAudioList) {

try {

MediaPlayer mp = new MediaPlayer();

Uri audioUri = Uri.parse(audio.getPath());

Log.d("PlayAudio", "Preparing to play: " + audioUri);

android.media.AudioAttributes audioAttributes = new android.media.AudioAttributes.Builder()

.setContentType(android.media.AudioAttributes.CONTENT\_TYPE\_MUSIC)

.setUsage(android.media.AudioAttributes.USAGE\_MEDIA)

.build();

mp.setAudioAttributes(audioAttributes);

mp.setDataSource(getApplicationContext(), audioUri);

mp.prepareAsync();

mp.setOnPreparedListener(new MediaPlayer.OnPreparedListener() {

@Override

public void onPrepared(MediaPlayer mp) {

// Применяем индивидуальные настройки громкости и стереопозиции

float leftVolume = (1.0f - (float) audio.getStereoPosition() / 100f) \* ((float) audio.getVolume() / 100f);

float rightVolume = ((float) audio.getStereoPosition() / 100f) \* ((float) audio.getVolume() / 100f);

mp.setVolume(leftVolume, rightVolume);

mp.start();

}

});

mp.setOnCompletionListener(mediaPlayer -> {

mediaPlayer.release();

});

activeMediaPlayers.add(mp);

} catch (IOException e) {

Log.e("PlayAudio", "Error playing audio file: " + audio.getPath(), e);

}

}

isPlaying = true;

}

// Запуск активности экспорта аудио.

private void launchExportActivity() {

Intent exportIntent = new Intent(this, ExportActivity.class);

exportIntent.putExtra("audioList", (Serializable) mainAudioList); // Assuming audioFiles is your list of AudioModel

startActivity(exportIntent);

}

// Остановка всех аудиофайлов и освобождение ресурсов.

private void stopAllAudio() {

for (MediaPlayer mp : activeMediaPlayers) {

if (mp != null && mp.isPlaying()) {

mp.stop();

}

if (mp != null) {

mp.release();

}

}

activeMediaPlayers.clear();

isPlaying = false;

}

// Настройка лаунчера для выбора аудиофайлов.

private void setupAudioPickerResultLauncher() {

audioPickerResultLauncher = registerForActivityResult(new ActivityResultContracts.StartActivityForResult(), result -> {

if (result.getResultCode() == RESULT\_OK && result.getData() != null) {

Uri audioUri = result.getData().getData();

// Запрашиваем постоянные разрешения для URI

try {

// Получаем флаги, которые приложение может установить на этом URI

int takeFlags = result.getData().getFlags();

takeFlags &= (Intent.FLAG\_GRANT\_READ\_URI\_PERMISSION | Intent.FLAG\_GRANT\_WRITE\_URI\_PERMISSION);

getContentResolver().takePersistableUriPermission(audioUri, takeFlags);

// Добавляем аргументы volume и stereoPosition

int defaultVolume = 70;

int defaultStereoPosition = 50;

addAudioFileToProject(audioUri, defaultVolume, defaultStereoPosition);

} catch (SecurityException se) {

Log.e("AudioPicker", "Security Exception when taking persistable permissions", se);

} catch (Exception e) {

Log.e("AudioPicker", "Error processing audio file", e);

}

} else {

Log.d("AudioPicker", "Result NOT OK or Data is null");

}

});

}

// Добавление аудиофайла в проект с заданными настройками громкости и стереопозиции.

private void addAudioFileToProject(Uri audioUri, int volume, int stereoPosition) {

try {

int durationInSeconds = getAudioDuration(audioUri);

String audioPath = audioUri.toString();

String fileName = audioUri.getLastPathSegment() != null ? audioUri.getLastPathSegment() : "New Audio";

// Использование переданных значений громкости и стереопозиции

AudioModel newAudio = new AudioModel(fileName, durationInSeconds, audioPath, volume, stereoPosition);

temporaryAudioList.add(newAudio);

temporaryAdapter.notifyDataSetChanged();

mainAudioList.add(newAudio);

currentProject.addAudioFile(newAudio);

saveProject(currentProject);

} catch (SecurityException se) {

Log.e("AudioPicker", "Security exception accessing the URI", se);

} catch (Exception e) {

Log.e("PlayAudio", "Error processing audio file: " + audioUri, e);

}

}

// Обработка выбранного аудиофайла и его интеграция в проект.

private void processAudioFile(Uri audioUri) {

try {

int durationInSeconds = getAudioDuration(audioUri);

String audioPath = audioUri.toString();

String fileName = audioUri.getLastPathSegment() != null ? audioUri.getLastPathSegment() : "New Audio";

AudioModel newAudio = new AudioModel(fileName, durationInSeconds, audioPath);

temporaryAudioList.add(newAudio);

temporaryAdapter.notifyDataSetChanged();

mainAudioList.add(newAudio);

currentProject.addAudioFile(newAudio);

saveProject(currentProject);

} catch (Exception e) {

Log.e("AudioPicker", "Error processing audio file", e);

}

}

}