МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «КубГУ»)

Факультет компьютерных технологий и прикладной математики
Кафедра прикладной математики

Допустить к защите
Заведующий кафедрой
д-р физ.-мат. наук, профессор
_____ М. Х. Уртенов
_____ 2022 г.

Руководитель ООП
д-р физ.-мат. наук, профессор
_____ М. Х. Уртенов
_____ 2022 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
(МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ)

РАЗРАБОТКА МУЗЫКАЛЬНОГО ПЛЕЕРА С ФУНКЦИЕЙ РАСПОЗНАВАНИЯ КОМПОЗИЦИЙ ДЛЯ ОС ANDROID

Работу выполнил _____ А.А. Хачатрян

Направление подготовки 01.04.02 Прикладная математика и информатика

Направленность (профиль) Математическое и информационное обеспечение экономической деятельности

Научный руководитель
канд. физ.-мат. наук, доц. _____ А.В. Письменский

Нормоконтролер
преподаватель _____ Е.С. Троценко

Краснодар
2022

РЕФЕРАТ

Выпускная квалификационная работа, 78 страниц, 57 рисунков, 18 источников.

РАЗРАБОТКА, UI/UX DESIGN, МУЗЫКАЛЬНЫЙ ПЛЕЕР, ИДЕНТИФИКАЦИЯ МУЗЫКАЛЬНЫХ КОМПОЗИЦИЙ, РАСПОЗНАВАНИЕ МУЗЫКИ, МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, ОС ANDROID

Целью данной работы является создание современного музыкального плеера с продуманным дизайном, распознающего музыку и позволяющего её бесплатно прослушать в популярных сервисах.

Для достижения поставленной цели необходимо знать язык программирования Java [1] или Kotlin, уметь разрабатывать приложения под ОС Android [2], уметь проектировать красивый и удобный дизайн, владеть графическим редактором Figma, а также решить следующие задачи:

– рассмотреть основные методы решения поставленной задачи;

– сделать обзор уже существующих приложений, позволяющих идентифицировать музыкальные композиции;

– разработать свою технологию распознавания музыки или воспользоваться сторонними сервисами, предоставляющими доступ к такому функционалу;

– придумать дизайн и спроектировать приложение;

– создать необходимые для дизайна графические объекты;

– разработать приложение, распознающее музыкальные композиции;

– добавить в приложение дополнительные возможности, которые сделают его более привлекательным для пользователей, а также обеспечат функциональность, востребованную или, вернее будет сказать, ожидаемую большинством современных пользователей;

– протестировать приложение.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Все мы любим слушать музыку. Она нас развлекает, отвлекает от чего-то неприятного, расслабляет, создаёт хорошее настроение. Она может вызвать слёзы, заставить задуматься. Также в нашей жизни может возникнуть ситуация, когда мы случайно услышали какую-либо музыкальную композицию и желаем узнать её название и исполнителя. Можно, конечно, попробовать найти песню по нескольким словам, которые нам запомнились или отрывку текста песни в интернете, но это далеко не самый эффективный и удобный способ. А ведь может оказаться так, что искомая музыкальная композиция не содержит в себе текста, а состоит только из мелодии. Как же быть?

Выход есть! И в данной работе будет рассмотрено решение данной проблемы при помощи мобильного приложения. Это наверно самый удобный, эффективный и популярный способ решения этой задачи в наше время. Ведь у подавляющего большинства населения мира (по крайней мере в развитых и развивающихся странах) есть смартфоны. А на смартфоны, соответственно, устанавливаются мобильные приложения. Смартфоны обычно у всех всегда с собой. А это значит, что в случае желания или необходимости послушать или распознать какую-либо музыкальную композицию удобнее и, что важно, быстрее всего это удастся сделать при помощи приложения на смартфоне.

В нашем случае будет разрабатываться приложение для ОС Android. Это самая популярная мобильная операционная система в мире. По данным Statcounter Global Stats [3] по состоянию на июнь 2022 года ОС Android используется на 72.27 % мобильных устройств (в России по состоянию на май 2022 года 76.40 % по данным Яндекс.Метрики [4]). Главный её конкурент iOS «владеет» 27.06 % всех смартфонов (в России по состоянию на май 2022 года 23.53 % по данным Яндекс.Метрики [4]).

Итак, целью данной работы является разработка музыкального плеера с грамотно продуманным дизайном, распознающего музыкальные композиции и позволяющего бесплатно прослушать их в популярных среди пользователей сервисах.

1 Актуальность темы

Конечно, уже существуют мобильные приложения, идентифицирующие музыкальные композиции. Но ни одно из них не идеально, и всего 2-4 приложения в целом успешны, но и они имеют некоторые существенные недостатки:

– почти во всех этих приложениях нельзя слушать найденную композицию целиком бесплатно (только 30 секунд);

– в некоторых из них (например, Shazam) прослушать песню можно только в определённых приложениях (в случае Shazam-а это Apple Music). То есть пользователю необходимо скачать ещё одно приложение, зарегистрироваться в нём и, как это принято в последние несколько лет, оформить платную подписку, чтобы слушать музыку. Это очень неудобно для многих пользователей и, что важно, не бесплатно;

– нет возможности скачать найденную музыку (насколько мне известно, ни в одном подобном приложении), по крайней мере бесплатно.

Разработка такого приложения актуальна в 2022 году – актуальность в текущей ситуации возрастает, т. к. постепенно появляются ограничения на пользование различными зарубежными IT-сервисами (Spotify, часть функциональности в Google Play, Google Pay, системы денежных переводов и т. д.). Кто знает, что ещё будет ограничено сегодня вечером… Лучше подготовиться заранее.

Кроме того, ещё есть к чему стремиться – разработка музыкального плеера с функцией распознавания композиций имеет перспективы развития, т. к. можно расширить функциональность, предоставляя пользователю более подробную информацию о музыке на устройстве пользователя и распознанных приложением музыкальной композиции, его исполнителе, перевод теста песни и др..

Таким образом, тема является актуальной на сегодняшний день и можно ещё многое придумать, чтобы улучшить приложения данной направленности и сделать их более полезными и удобными для пользователей.

2 Способы разработки приложения для идентификации музыки

2.1 Разработка собственной технологии распознавания музыки

Для этого нужно очень хорошо разбираться в анализе звуковой информации и высшей математике, а также, для создания более эффективной технологии распознавания, и в нейросетях. Также для осуществления такого замысла необходимо очень много времени и ресурсов.

2.2 Использование готовых сервисов

К счастью, сейчас уже есть сервисы, предоставляющие возможность распознавания музыки через свои API. Одним из таковых является сервис распознавания музыки от платформы ACRCloud. Именно им мы и воспользуемся.

ACRCloud – это платформа автоматического распознавания контента, основанная на технологии акустической подписи звука (создание акустического отпечатка композиции). В базе данных ACRCloud более 100 млн. треков, и эта база постоянно обновляется. Создатель платформы намеревался помочь СМИ, вещательным компаниям и разработчикам приложений идентифицировать, отслеживать и монетизировать контент на «втором экране».

Итак, для того, чтобы воспользоваться этим сервисом, нам необходимо:

– зарегистрироваться на сайте ACRCloud [5];

– согласно инструкции ACRCLoud [6] создать проект распознавания музыки и получить необходимые для доступа к сервису параметры “host” (сервер), “access_key”, “access_secret” (ключи доступа);

– согласно инструкции ACRCLoud [7] подключить библиотеку “acrcloud-universal-sdk-1.2.20.jar“ в проекте нашего приложения в Android Studio (официальная среда разработки Android приложений) и настроить

главную активность приложения (т. е., простыми словами, главный «экран» приложения, который отображается при запуске пользователем приложения. Именно на этом экране будет расположена кнопка распознавания, а следовательно, именно из главной активности приложения будет вызываться сервис идентификации музыкальных композиций от ACRCloud).

После того, как вышеперечисленные шаги будут выполнены, приложение будет способно идентифицировать музыку.

3 Обзор существующих приложений, распознающих музыку

3.1 Shazam

Shazam – бесплатный кроссплатформенный проект, позволяющий пользователю определить, что за песня играет в данный момент. Это наверно самое удачное приложение для распознавания музыки на данный момент.

Пользователь Shazam использует микрофон устройства для записи фрагмента музыки, которая играет где-либо. Затем программа сравнивает фрагмент с центральной базой данных и при успешном сопоставлении выдаёт информацию о треке. В настоящий момент сервис предоставляет информацию о более чем 11 млн треков [8]. Shazam хранит каталог аудио, опознанных при помощи программы, давая прямые ссылки на данные треки на YouTube и Apple Music, если таковые там есть. Shazam в данный момент принадлежит Apple.

Приложение очень хорошее и удобное, но у него всё же есть некоторые недостатки: слушать найденную песню можно только в Apple Music, что не бесплатно, требует наличия у пользователя ещё и этого приложения, регистрации в нём и подписки (в общем, очень неудобно).

3.2 SoundHound

SoundHound – второе по популярности мобильное приложение для идентификации музыкальных композиций. Оно даёт возможность прослушать трек бесплатно через плеер YouTube (вместе с видеоклипом).

Это один из ближайших конкурентов Shazam для использования на смартфоне. Приложение адаптировано для работы Android и iOS-устройств. Оно умеет мгновенно определять название и исполнителя композиции, предлагает ознакомиться с текстом песни и даёт прямые ссылки на её покупку.

Кстати, вы можете нажать оранжевую кнопку, максимально похоже напеть фрагмент песни в микрофон смартфона, а система попробует распознать её.

Но и тут не всё гладко:

– Наблюдались случаи неполадок при нажатии на кнопку «Play», чтобы прослушать песню. Иногда она почему-то не загружалась, а переключалась на следующую песню исполнителя (из плейлиста с песнями исполнителя распознанной музыки, предложенного SoundHound), и та тоже не всегда загружается;

– в бесплатной версии присутствует реклама, расположенная не всегда в самом удачном месте (один из баннеров расположен на главном экране, прямо под кнопкой распознавания, что портит внешний вид как минимум (хотя этот баннер иногда исчезал, так и не появившись)).

3.3 Другие приложения и сервисы

Также послушать и распознать музыку можно с помощью мобильных приложений Яндекс.Музыка, Beatfind, Deezer, голосовых помощников Siri и Google Assistant, SoundSearch (стандартный виджет, который доступен в контекстном меню на рабочем столе мобильного Android устройства), а также некоторых веб-сайтов.

У всех этих сервисов есть свои достоинства и недостатки, и, следовательно, они не идеальны и им есть над чем поработать, а у других разработчиков мобильных приложений есть возможность сделать свои приложения и составить конкуренцию уже имеющимся.

4 Проектирование и дизайн

Разработка производилась в IDE Android Studio, с помощью языка программирования Java.

Дизайн частично разрабатывался при помощи графического редактора Figma.

4.1 Знакомство с понятиями Android

Активность приложения (activity) – это, простыми словами, конкретный «экран» приложения, который отображается на дисплее смартфона. На этих экранах расположены элементы взаимодействия с пользователем – представления (view).

Фрагмент (fragment) – это, по сути, это подобие Activity, которое мы можем подключать в разные части приложения. Одно Activity может содержать несколько фрагментов. Они не могут существовать сами по себе, а только вместе с Activity.

Макет (layout) определяет визуальную структуру пользовательского интерфейса. Существует два способа объявить макет:

– объявление элементов пользовательского интерфейса в XML;

– создание экземпляров элементов интерфейса во время выполнения.

Разработчик может объявить в XML макеты по умолчанию, включая элементы экрана, которые будут отображаться в макетах, и их свойства. Затем можно добавить в приложение код, который позволяет изменять состояние объектов на экране (включая объявленные в XML) во время выполнения.

Адаптеры (adapter) – это, по сути, переходники, которые упрощают связывание данных с элементом управления. Адаптер отвечает за извлечение данных из набора данных и за создание на их основе объектов View.

Android предоставляет несколько способов для хранения пользовательских данных приложений. SQLite является одним из способов хранения данных пользователя. SQLite это очень легковесная база данных, которая содержится в ОС Android.

Для рассмотрения отдельных элементов, которые используются для редактирования макета, можно переключиться в режим Blueprint, чтобы ничего не отвлекало от работы. В Blueprint отображаются только контуры View компонентов (т. е. представлений, элементов интерфейса).

4.2 Описание структуры приложения

Приложение состоит из 8 активностей (и их макетов) и 6 фрагментов (и их макетов), 4 адаптеров, 1 класса модели данных, 1 класса-сервиса для воспроизведения музыки, 1 класса для контроля уведомления музыкального плеера, 1 интерфейса, содержащего действия контроля воспроизведения музыки, 1 класса типа Application, контролирующего глобальное состояние приложения, и использует базу данных SQLite для хранения истории распознаваний. Также нам понадобится 3 интерфейса для работы с API сервисов, которые были использованы для некоторых целей, о которых будет сказано далее.

4.2.1 Главная активность приложения

Главная активность приложения (MainActivity) – это та активность (простыми словами, экран), которая отображается при запуске приложения. В нашем случае эта активность содержит 3 вкладки (это фрагменты, которые содержат историю распознаваний, песни на устройстве пользователя, музыкальные альбомы на устройстве пользователя), кнопку запуска процесса распознавания (она отображается поверх всех трёх фрагментов), мини-плеер для

контроля воспроизведением музыки с главного экрана, кнопки поиска и меню на панели действий (вверху экрана).

Рассмотрим макет главной активности приложения на рисунке 1.

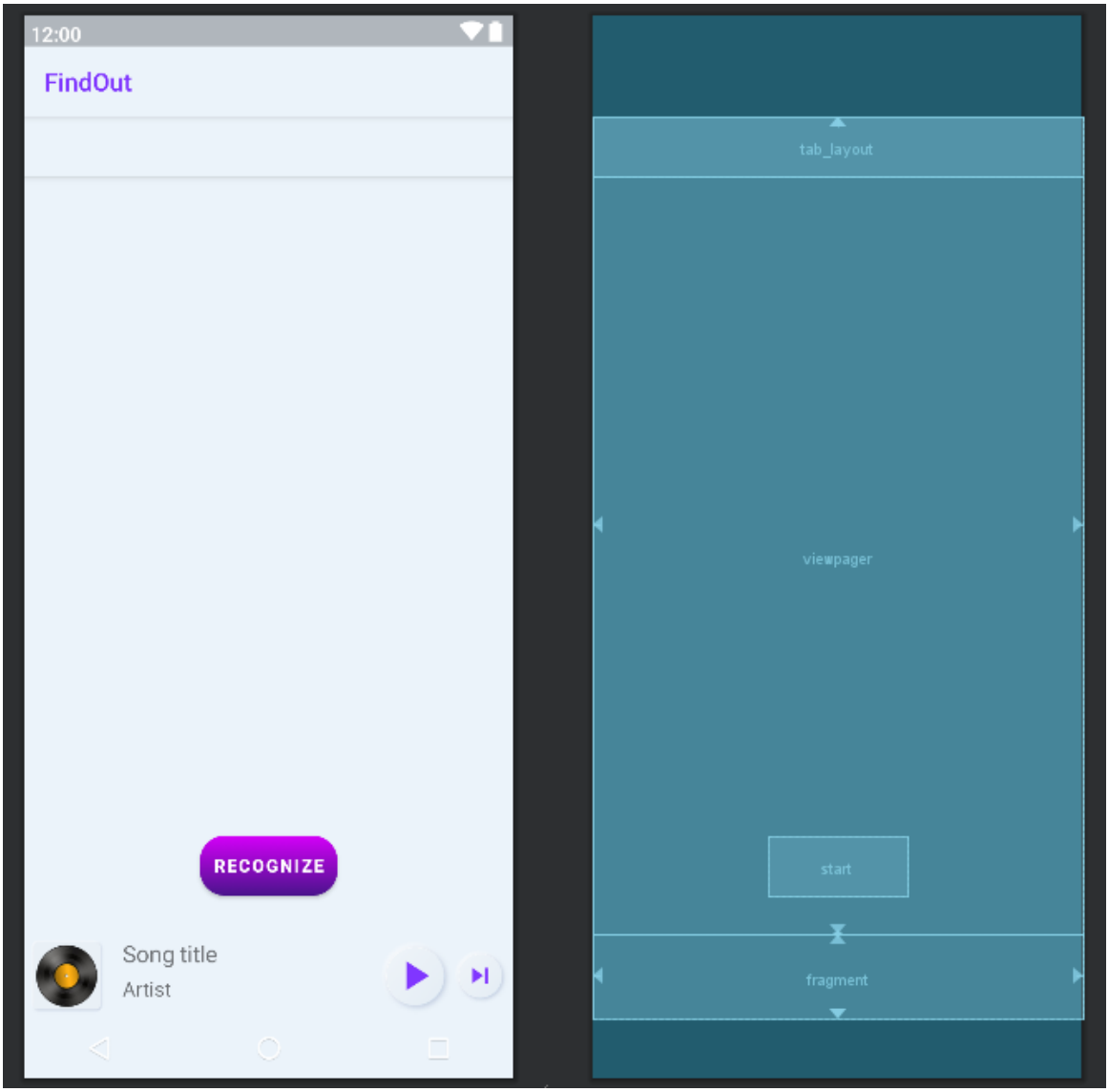


Рисунок 1 – Макет главной активности

Корневой элемент (корень иерархии представлений view – контейнер, в котором расположены все остальные view) этого макета это ConstraintLayout. Этот тип макетов очень удобен тем, что в этом случае элементы интерфейса связаны друг с другом и с границами экрана определённым образом, т. е. расположены друг от друга на заданном расстоянии (в пикселях), независимо от размера экрана самого устройства. Другими словами, макет будет выгля-

деть одинаково при любых размерах смартфона и при правильной настройке элементов, ни один из них не будет перегораживаться другим или выходить за пределы экрана (делаясь, тем самым, недоступным для пользователя).

Здесь в центре внизу расположена кнопка запуска распознавания. Она исчезает при прокрутке содержимого экрана вниз и появляется вновь при прокрутке содержимого экрана вверх. При нажатии на кнопку распознавания происходит анимация поочерёдной смены 4-х градиентных цветов (фиолетовый, оранжевый, зелёный и красный) [9].

4.2.2 Фрагмент с музыкой на устройстве

Этот фрагмент (SongsFragment), как уже упоминалось выше, включён в главную активность. Он отображается при запуске приложения. Здесь отображается музыка пользователя на устройстве.

При нажатии на композицию открывается музыкальный плеер и запускается воспроизведение выбранной музыки. Также пользователю предоставляется возможность удаления музыкального файла с устройства через мини-меню каждого музыкального элемента в списке. Раскрыть это меню можно нажав на три точки в правом краю музыкального элемента.

Ниже приведён макет фрагмента на рисунке 2.

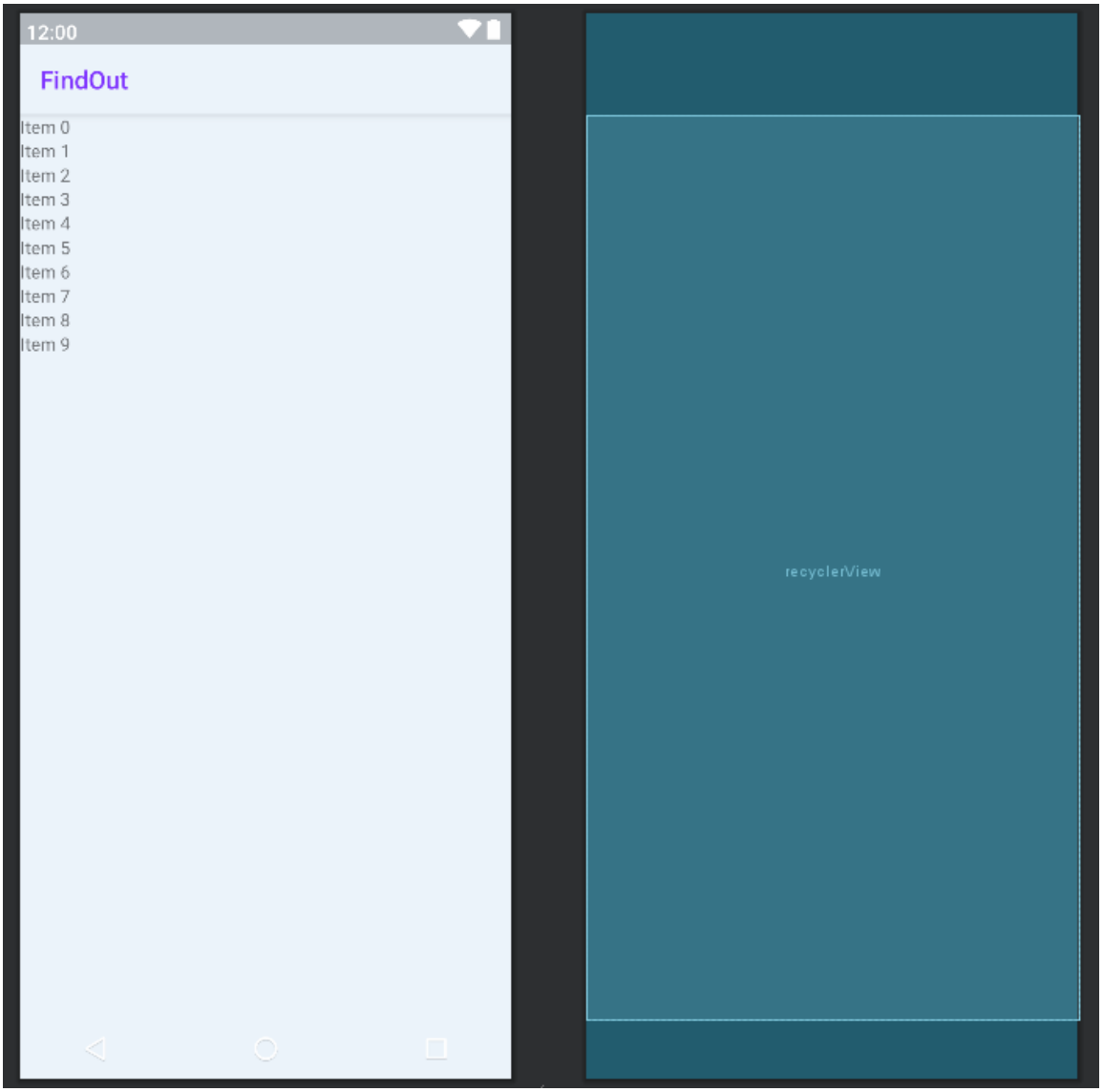


Рисунок 2 – Макет фрагмента с музыкой на устройстве

4.2.3 Фрагмент с историей распознавания

В этом фрагменте (RecognizedFragment) отображаются все распознанные пользователем музыкальные композиции. При идентификации композиции основная информация о ней (название песни, исполнитель (-ли) и ссылка на изображение обложки альбома (если она есть)) сохраняется в базу данных SQLite, и главная активность при запуске берёт оттуда эту информацию и передаёт в фрагмент с историей распознавания.

Информация о распознанных композициях отображается в виде списка карточек, на которых слева отображается изображение с обложкой альбома, к

которому относится композиция, а справа от этого изображения название композиции и её исполнитель (исполнители).

В дальнейшем, если приложение будет развиваться, разумно будет добавить на эти карточки и дополнительную информацию, такую как дата распознавания и место композиции в местном хит-параде. Также, развивая и улучшая приложение полезно будет добавить этим карточкам и новую функциональность, как например, удаление композиции из списка коллекции или добавление в избранное.

Ниже на рисунке 3 приведён макет фрагмента, отображающего историю распознавания.

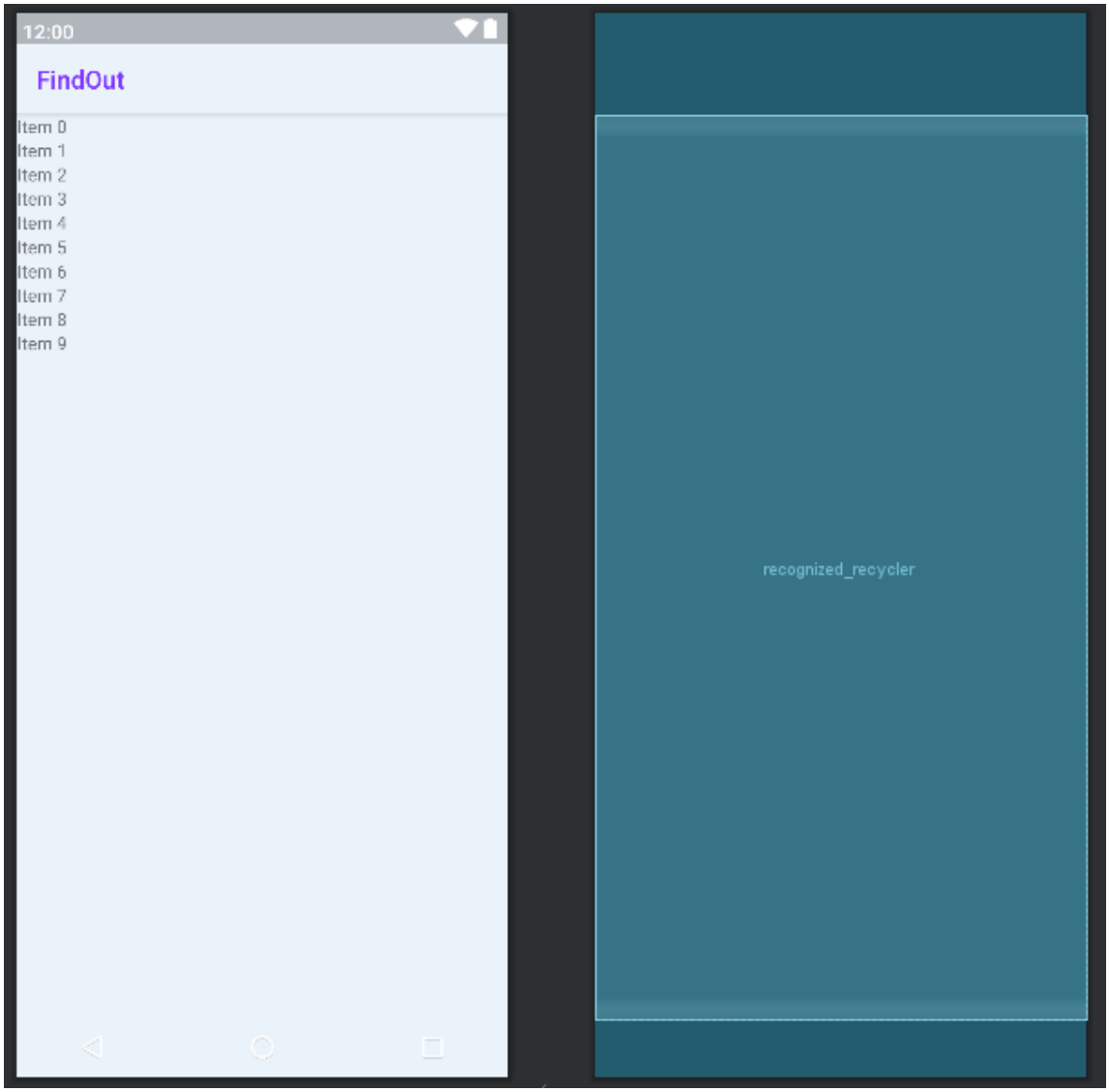


Рисунок 3 – Макет фрагмента с историей распознавания

Корневым элементом является LinearLayout – самый простой тип макета. В нём элементы отображаются по порядку их объявления в XML-файле в вертикальном или горизонтальном направлении.

Также здесь находится RecyclerView – с помощью этого компонента можно удобно вывести массив данных в виде «списка», не создавая вручную все количество View элементов, достаточно создать один фрагмент разметки в виде шаблона и передать ему массив данных через адаптер. Адаптер нужен для того, чтобы «взять данные», создать нужное количество View по нашему шаблону, заполнить их данными и вывести их в компоненте RecyclerView. В этом компоненте и отображается история распознаваний.

4.2.4 Фрагмент с музыкальными альбомами

В фрагменте AlbumFragment отображаются все музыкальные альбомы на устройстве пользователя. Они представлены в виде карточек, содержащих название и обложку альбома, и при нажатии на конкретный альбом открывается активность со всей музыкой из выбранного альбома.

При нажатии на композицию открывается музыкальный плеер и начинается воспроизведение выбранной композиции.

Ниже на рисунке 4 приведён макет фрагмента с музыкальными альбомами на устройстве пользователя.

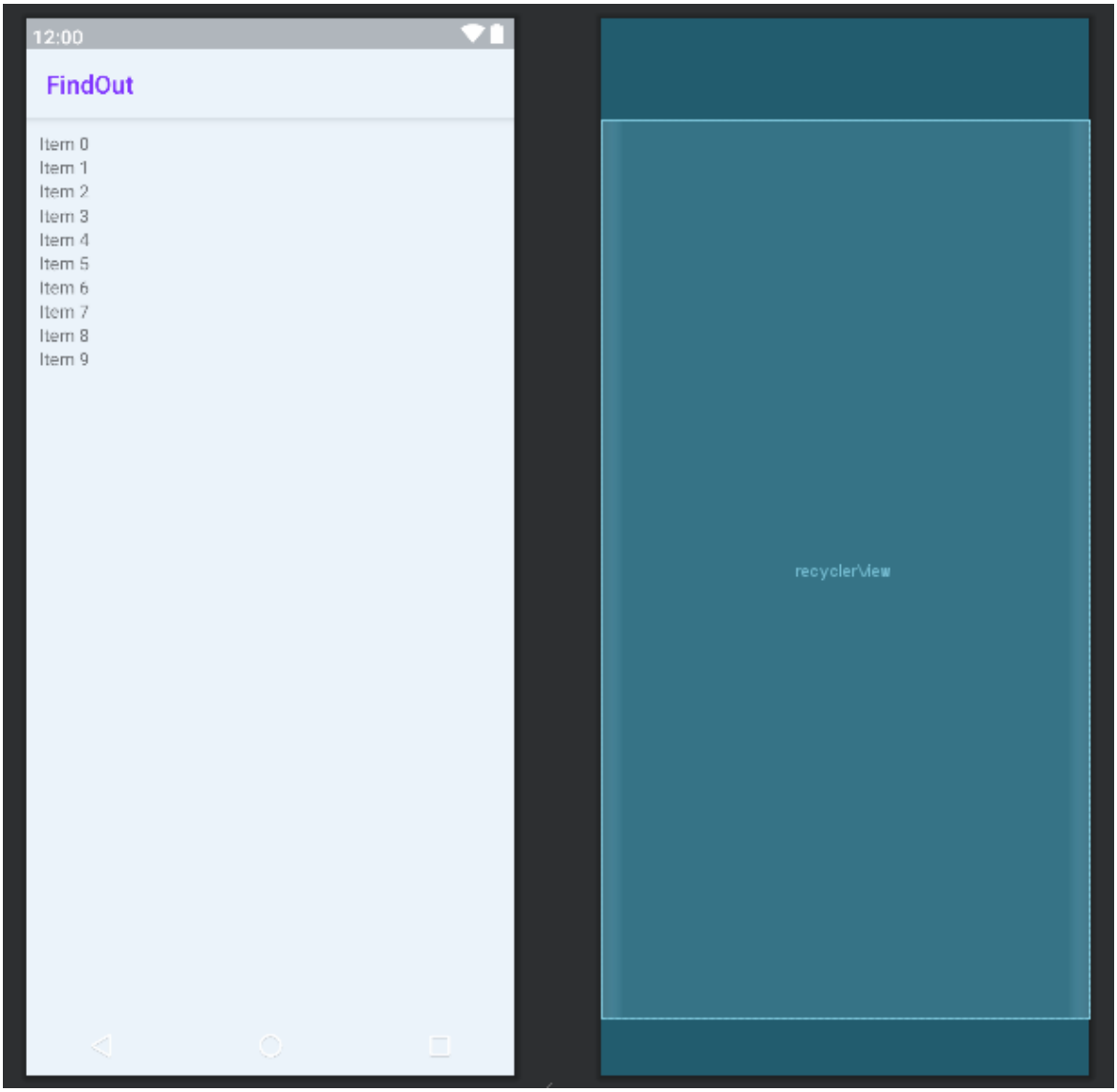


Рисунок 4 – Макет фрагмента с музыкальными альбомами

4.2.5 Активность музыкального альбома

Данная активность (AlbumDetailsActivity) отображает музыкальные композиции на устройстве пользователя, относящиеся к определённому музыкальному альбому, и обложку этого альбома.

При нажатии на композицию открывается музыкальный плеер и запускается воспроизведение выбранной музыки.

Вместо названия приложения на панель действий программно выводится название открытого альбома.

Ниже на рисунке 5 приведён макет этой активности.

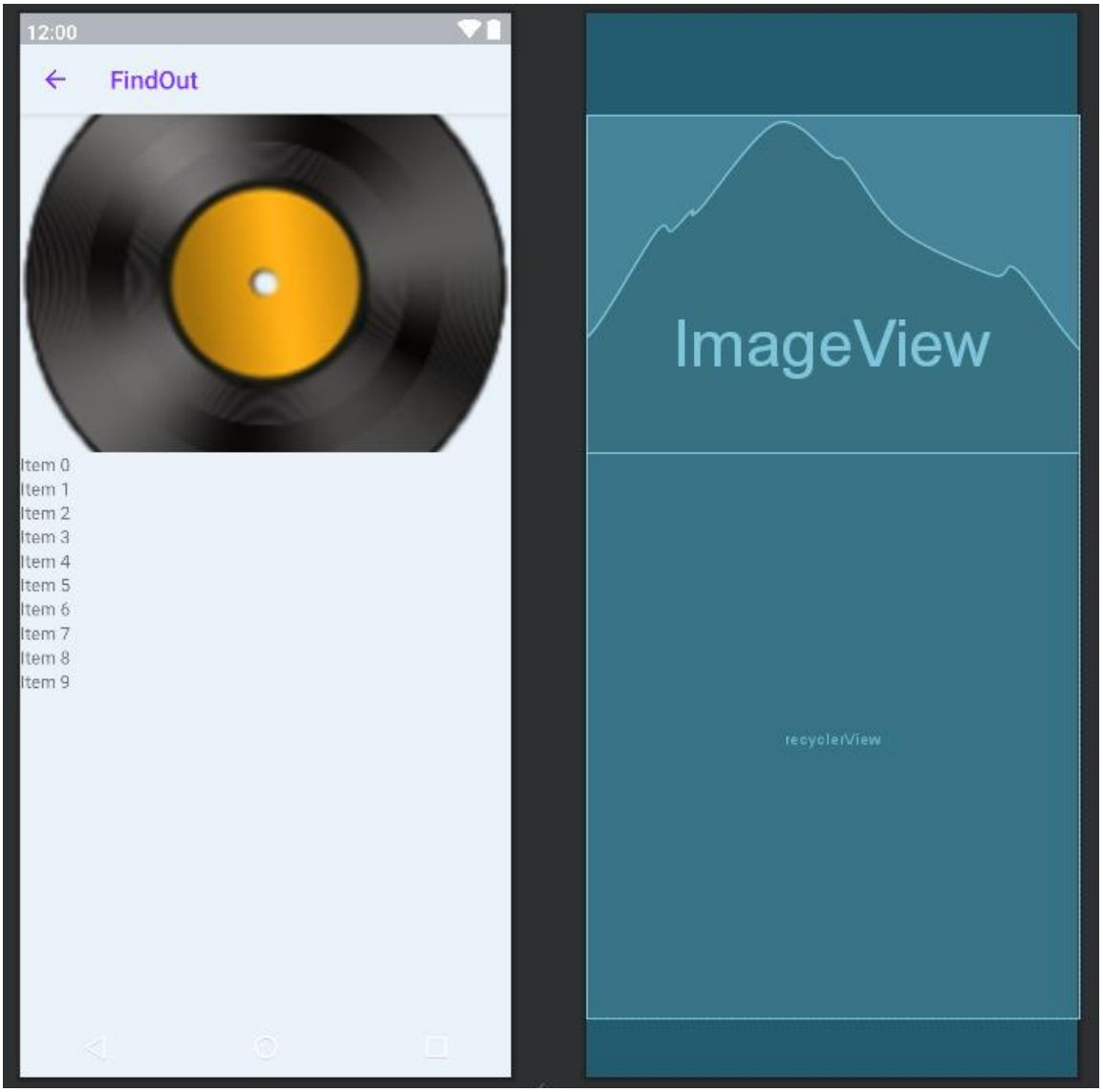


Рисунок 5 – Макет активности музыкального альбома

4.2.6 Активность неудавшегося распознавания

NothingFoundActivity – активность, которая отображается при неудавшемся распознавании. Рассмотрим её макет на рисунке 6.

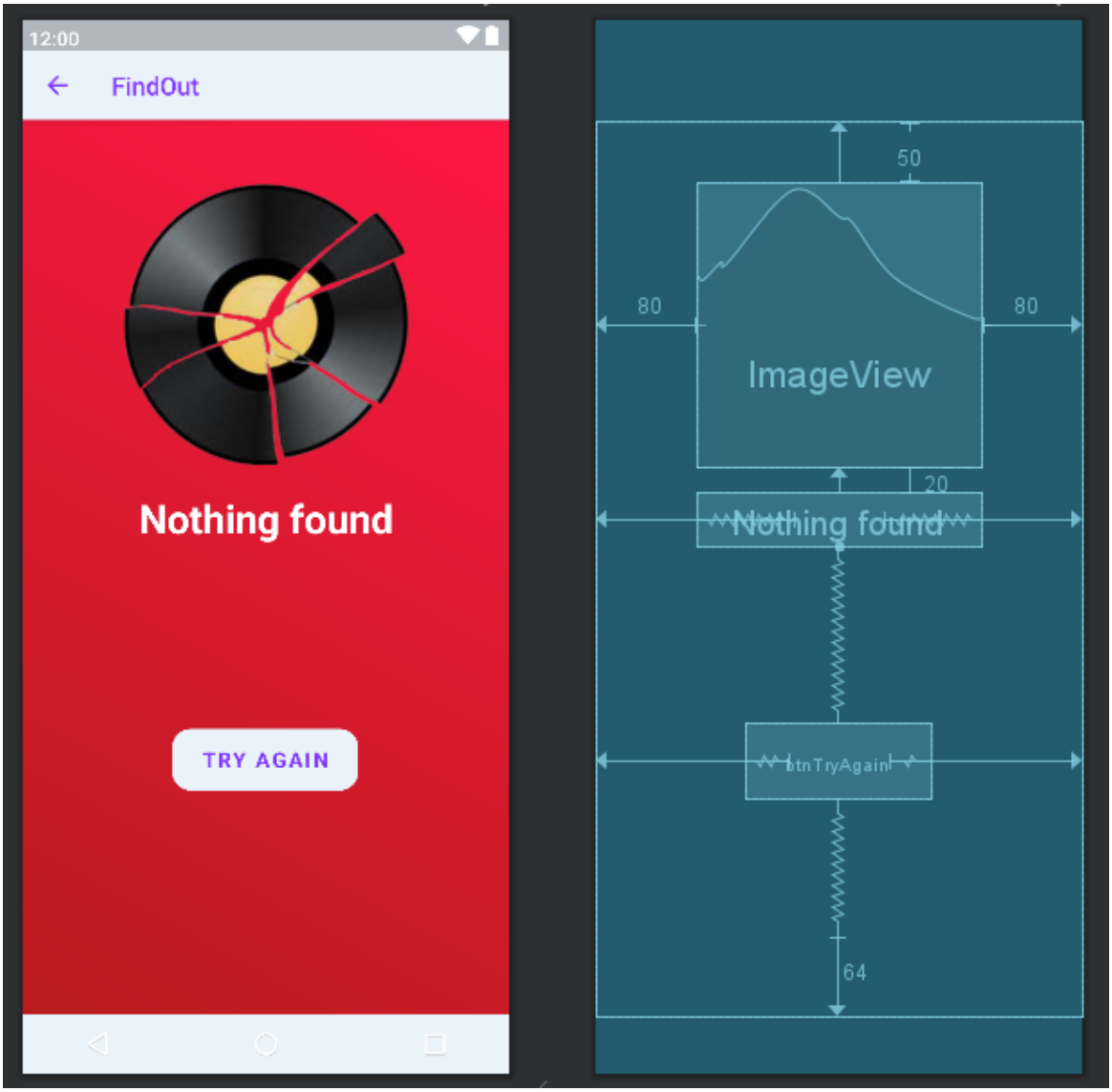


Рисунок 6 – Макет активности неудавшегося распознавания

Корневым элементом является ConstraintLayout. Здесь вместо названия активности отображается надпись «We’re so sorry :(», что в переводе с английского означает «нам очень жаль :(» (Данный текст, а также кнопка возврата назад устанавливаются на панель действий программно, во время выполнения).

Под панелью инструментов расположен компонент ImageView, с помощью которого отображается рисунок разбитого винилового диска (означающего проваленную попытку распознавания).

Под ImageView, расположен компонент TextView, в котором выводится надпись «Nothing found» (в пер. с англ. – «ничего не найдено»)

Под TextView расположена кнопка с надписью «TRY AGAIN» (с англ. – «попробовать снова»), нажав на которую приложение возвращает пользователя на главную активность и автоматически повторно запускается процесс распознавания.

4.2.7 Активность результатов распознавания

Активность результатов распознавания (RecognitionResultActivity) – активность, показывающая результаты распознавания (отображается в случае удачного распознавания).

Эта активность содержит в себе 2 фрагмента: SongFragment и MusicFragment. Первый из них отвечает за показ информации об идентифицированной композиции (название композиции, исполнитель, обложка альбома, название альбома, дата выхода, информацию о композиторах и авторах текста песни, лейбл, дату распознавания и слова песни), а второй – за предоставление пользователю возможностей прослушать музыкальную композицию.

Указанные фрагменты в активности отображаются раздельно (не одновременно). Пользователь видит либо SongFragment, либо MusicFragment.

Рассмотрим макет активности на рисунке 7.

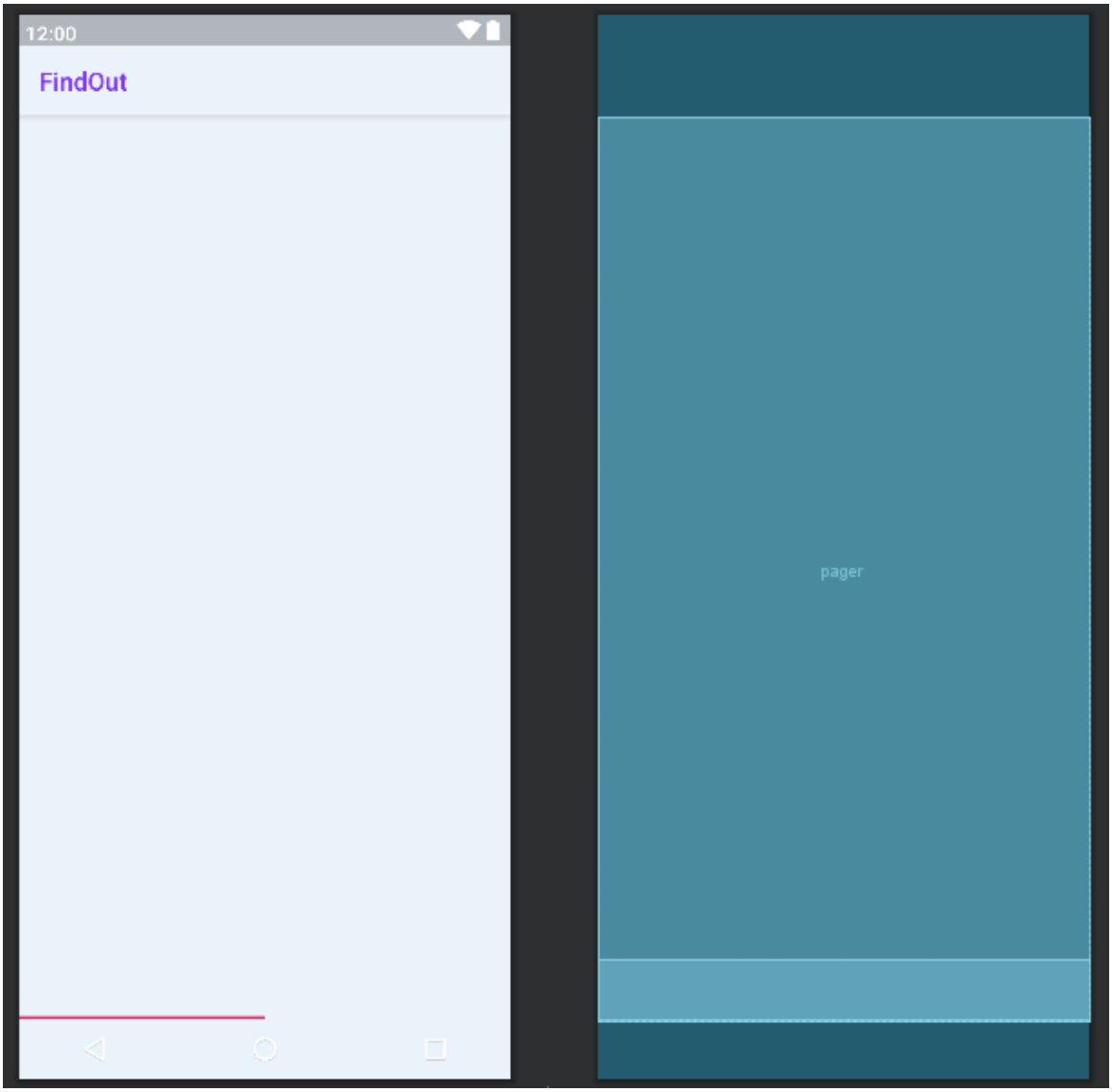


Рисунок 7 – Макет активности результата распознавания

В самом верху – ActionBar (панель действий), где программно отображаются слева кнопка «вверх» и справа – кнопка, имеющая форму экрана со значком «Play» внутри, позволяющая посмотреть видео на YouTube.

Большую часть макета занимает компонент ViewPager, который позволяет прокручивать экран влево-вправо жестом пальцами (свайп), то есть, в нашем случае мы переключаемся между SongFragment и MusicFragment.

В самом низу расположен компонент *TabLayout* – обеспечивает горизонтальное расположение для отображения вкладок, которые позволяют переключаться между различными фрагментами. В нашем случае 2 вкладки: «Music» – для MusicFragment, и «Song» – для SongFragment.

TabLayout расположен в BlurView от «8 Bit Lab», который используется для размытия заднего плана [10].

Корневым элементом является FrameLayout – обычно это пустое пространство на экране, в котором все дочерние элементы прикрепляются к верхнему левому углу экрана. В этой разметке нельзя определить различное местоположение для дочернего объекта. Последующие дочерние объекты View будут просто рисоваться поверх предыдущих компонентов, частично или полностью загораживая их, если находящийся сверху объект непрозрачен. Этот тип макета выбран не просто так. Он нужен для того, чтобы макеты фрагментов, которые расположены в ViewPager были растянуты на всё доступное пространство экрана устройства и были расположены под ActionBar-ом. Сам ActionBar частично прозрачен.

4.2.8 Фрагмент информации о композиции

Этот фрагмент, как уже упоминалось выше, включён в активность результатов распознавания (RecognitionResultActivity). Он отображается сразу после того, как музыка была распознана, а также когда пользователь переключается на вкладку «SONG».

В этом фрагменте отображается основная информация о найденной композиции, а именно: название композиции, исполнитель (исполнители), обложка альбома, в котором содержится данная композиция.

В этом фрагменте также пользователю даётся и дополнительная информация о распознанной песне (музыке), такая как название альбома, дата выхода, композитор (композиторы) и автор (авторы) текста, лейбл (звукозаписывающая компания), дата распознавания и слова песни (если они найдены).

Рассмотрим макет фрагмента на рисунке 8.

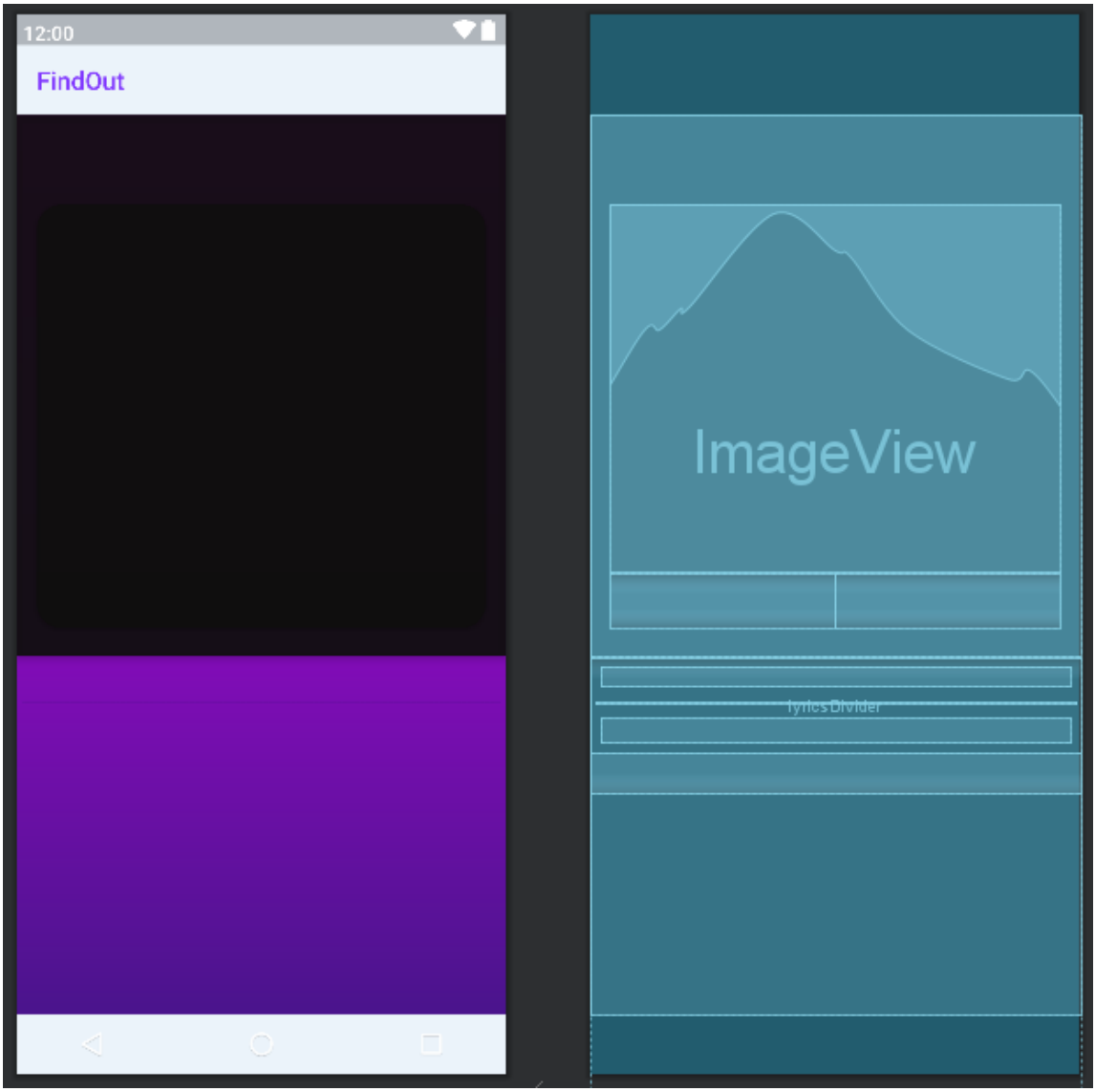


Рисунок 8 – Макет фрагмента информации о композиции

Корневым элементом является CircularRevealCoordinatorLayout – целью и философией этого контейнера является координация view элементов, которые находятся внутри него (одни View зависят от других). Он позволяет связывать (координировать) виджеты, помещённые в него (по сути, является продвинутым FrameLayout).

В самом верху расположен AppBarLayout – это вертикальный LinearLayout, в котором реализованы многие особенности концепции панели приложений для *Material Design*, а именно жесты прокрутки.

В AppBarLayout помещён CollapsingToolbarLayout – это оболочка для панели инструментов (Toolbar), которая реализует сворачивающуюся панель

приложения. Он предназначен для использования в качестве прямого потомка AppBarLayout. Дочерние представления могут быть прокручены в этом макете в режиме параллакса (эффект изменения видимого положения объекта относительно удалённого фона в зависимости от положения наблюдателя) – именно для этого и был выбран данный макет панели приложения.

В CollapsingToolbarLayout находится MaterialCardView – содержит контент (информацию) и, по желанию, действия по конкретной теме. В нашем случае здесь располагаются ImageView – для обложки альбома и LinearLayout, содержащий 2 TextView для отображения названия композиции и её исполнителя (-лей).

Под AppBarLayout находится NestedScrollView – используется для обеспечения прокрутки контента. В нём расположен LinearLayout, в котором помещены:

– TextView – для дополнительной информации о композиции (название альбома, дата выхода композиции, композиторы и авторы текста, лейбл и дата распознавания);

– Horizontal Divider – горизонтальный разделитель для разделения разделов дополнительной информации и текста песни. Он анимирован – 4 градиентных цвета (фиолетовый, оранжевый, зелёный и красный) поочерёдно сменяют друг друга;

– TextView – для заголовка «Lyrics»;

– и ещё один TextView – для самого текста песни.

4.2.9 Фрагмент прослушивания музыки

В этом фрагменте пользователю предоставляется возможность прослушать найденную композицию следующими способами:

– с помощью YouTube (с видеоклипом) – один из немногих, к сожалению, способов прослушать музыку бесплатно и без ограничений;

– с помощью музыки в «Вконтакте». Нажав на кнопку, осуществляется переход на мобильную версию сайта VK с помощью WebView (показ веб-страницы прямо в приложении, не выходя за его пределы). Прослушивание данным способом также бесплатно и пока, к счастью, неограниченно. Функционал мобильной версии урезан, но главная цель – прослушать музыку бесплатно – доступна.

Итак, рассмотрим макет фрагмента на рисунке 9.

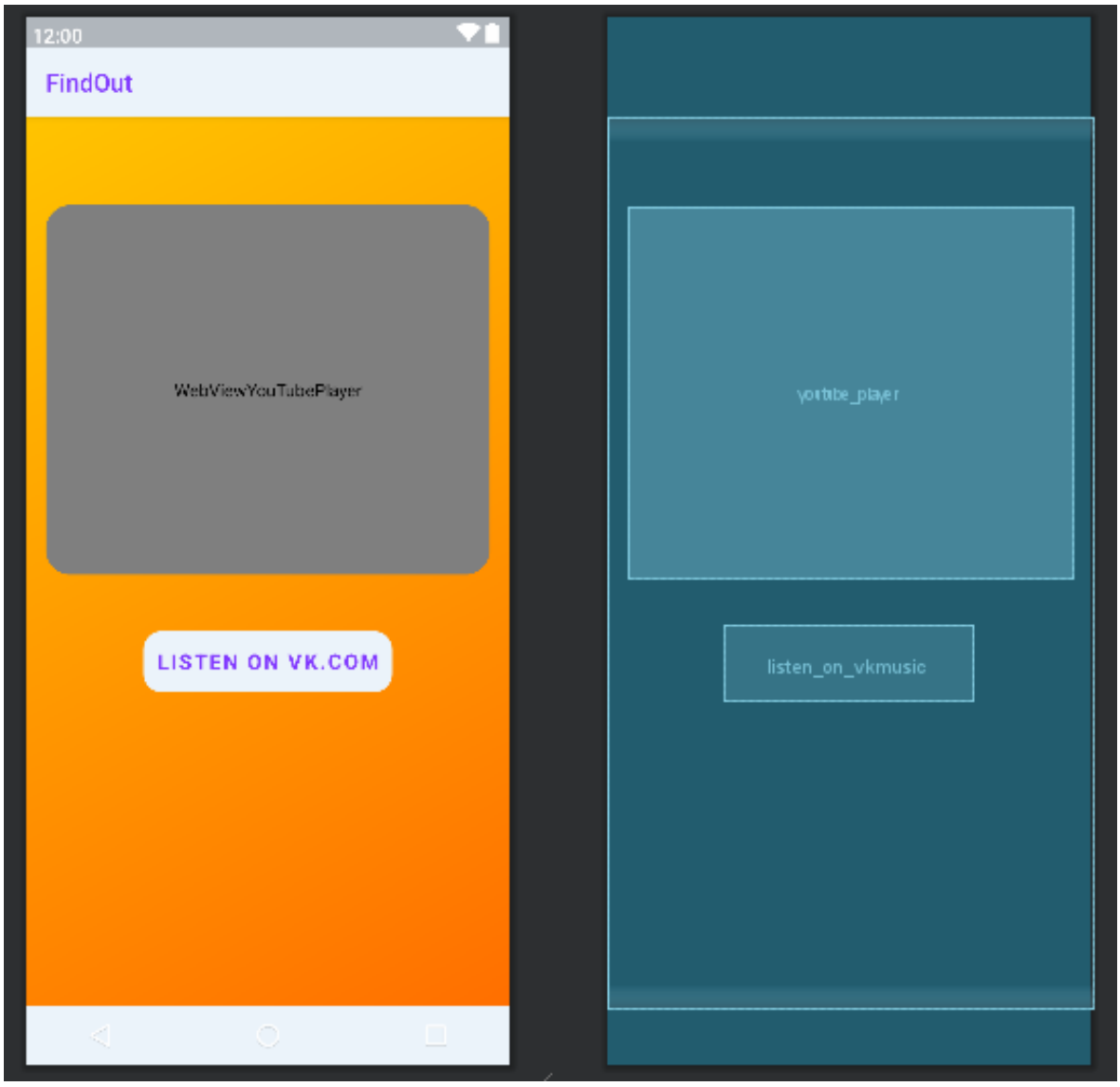


Рисунок 9 – Макет фрагмента прослушивания музыки

Корневой элемент – FrameLayout. В нём расположен LinearLayout, содержащий в себе всего 2 элемента: YouTubePlayerView и Button.

YouTubePlayerView – view из библиотеки *android-youtube-player*, представленной разработчиком Pierfrancesco Soffritti [11]. Используется для показа видео из YouTube.

Официальная библиотека Google для интеграции видео YouTube в приложения для Android – YouTube Android Player API. Но она довольно глючная (некоторым ошибкам старше нескольких лет) и долгое время не имела поддержки от Google. Это довольно ненадёжно и, следовательно, непригодно для использования в разработке.

Кнопка с надписью «LISTEN ON VK.COM» перенаправляет пользователя на соответствующую активность с WebView для прослушивания музыки в «Вконтакте».

4.2.10 Активности прослушивания музыки и просмотра видео

Макеты обоих активностей (WatchVideoActivity и VKMusicActivity) идентичны. Они содержат только WebView.

Рассмотрим их макеты на рисунках 10 и 11.

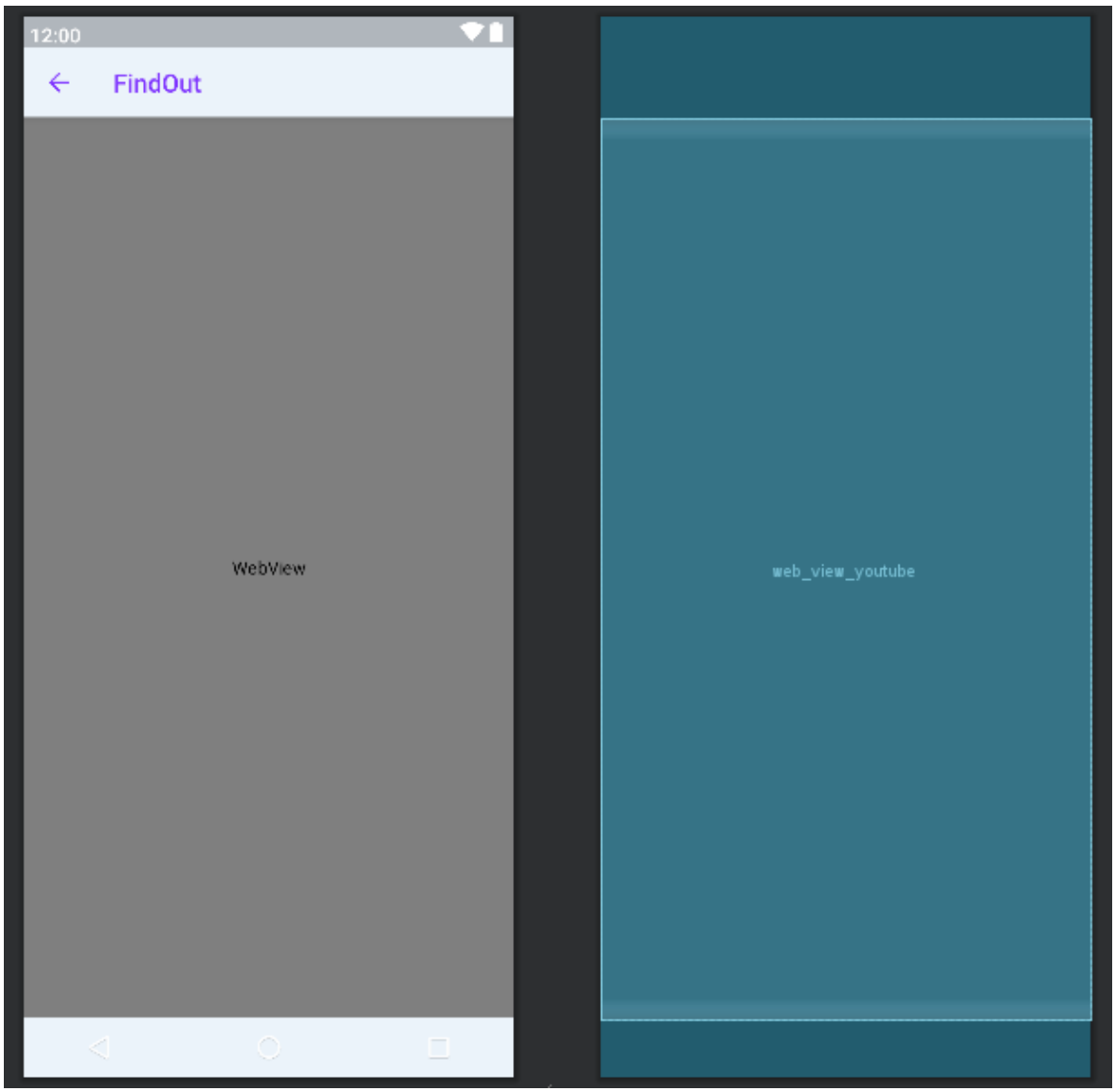


Рисунок 10 – Макет активности просмотра видео на YouTube

WebView активности WatchVideoActivity открывает сайт по адресу «https://www.youtube.com/watch?v=YOUTUBE_ID», где YOUTUBE_ID – идентификатор видеозаписи на YouTube.

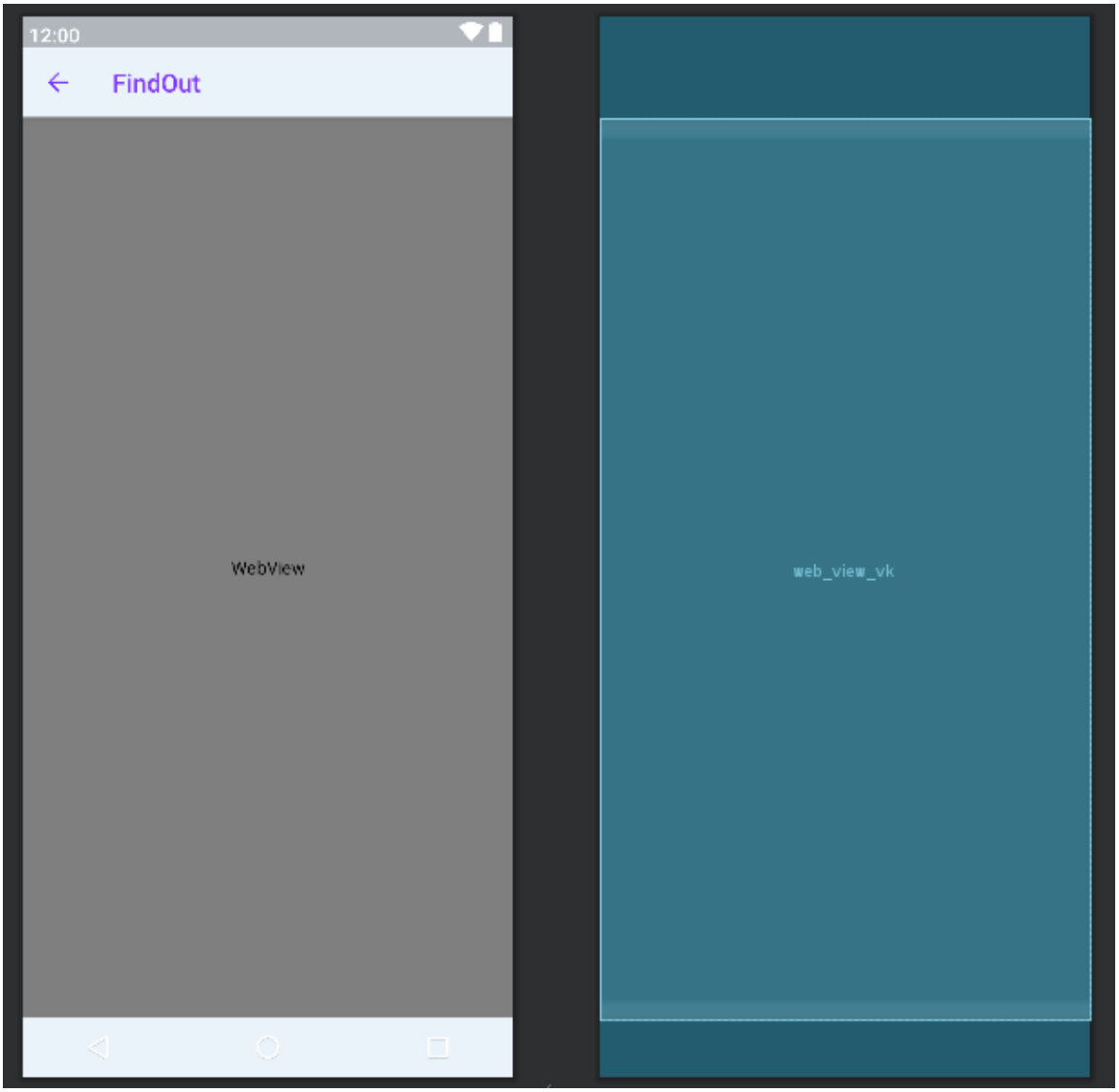


Рисунок 11 – Макет активности прослушивания музыки в «ВКонтакте»

WebView в случае VKMusicActivity открывает сайт по адресу «https://m.vk.com/audio?q=QUERY», где QUERY – содержит текст запроса, содержащего в себе имя (имена) исполнителя (-лей) и название композиции. В итоге откроется мобильная версия сайта «Вконтакте», автоматически откроется раздел с аудиозаписями с уже введённым в поиск запросом, а следовательно, будет найдена нужная музыкальная композиция.

4.2.11 Активность музыкального плеера

PlayerActivity – активность, обеспечивающая воспроизведение музыки.

Здесь отображается следующая информация и элементы контроля:

– обложка музыкального альбома,

– название и исполнитель (исполнители) песни,

– SeekBar (элемент интерфейса, отображающий звуковую дорожку, и позволяющий её контролировать),

– длительность композиции и текущее время воспроизведения,

– кнопки контроля воспроизведения музыки.

Ниже приведён макет активности музыкального плеера на рисунке 12.

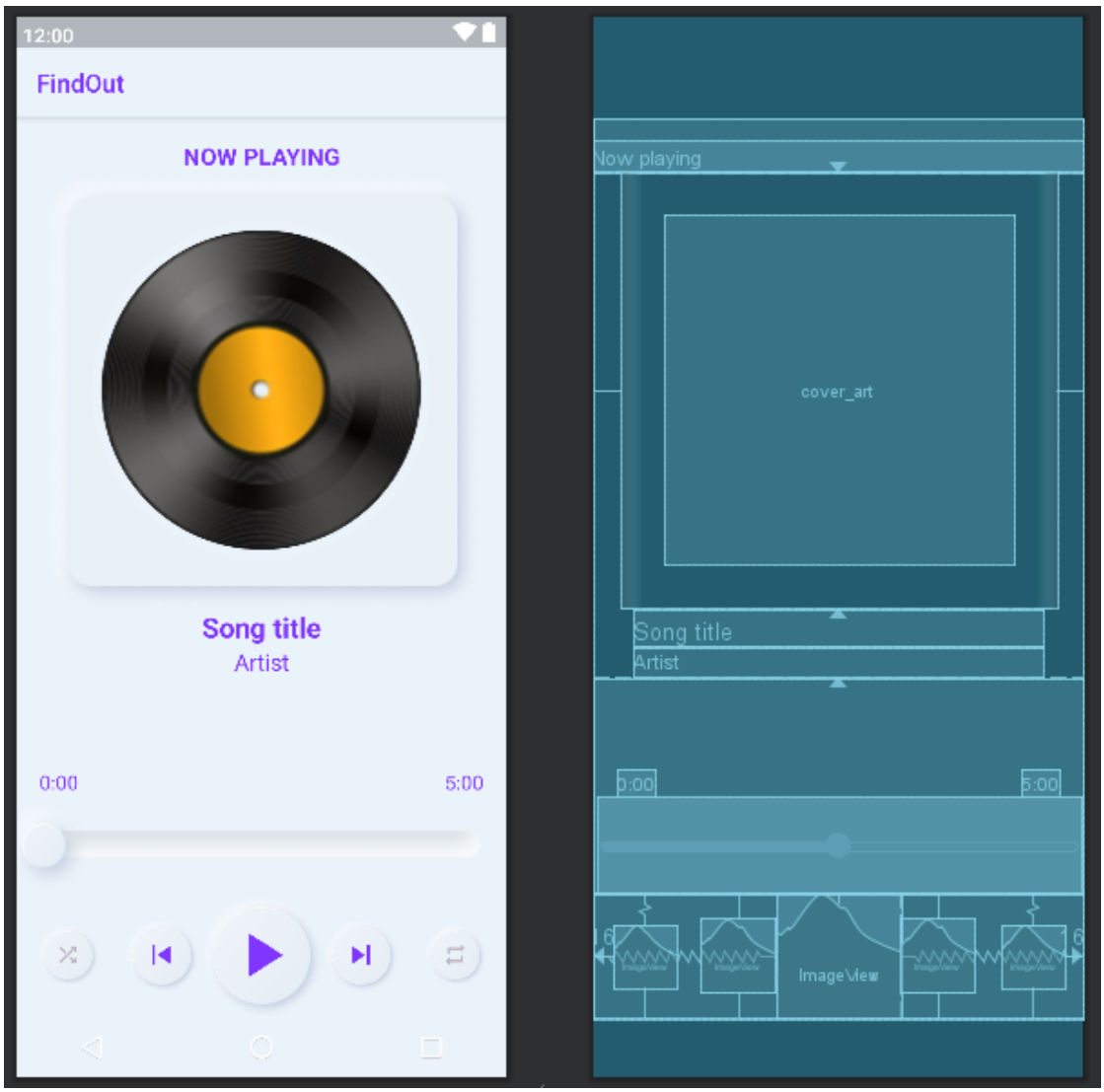


Рисунок 12 – Макет активности музыкального плеера

4.2.12 Фрагмент мини-плеера

NowPlayingBottomFragment отображается в самом низу главной активности приложения, чтобы позволить пользователю видеть и контролировать музыку, проигрываемую в текущий момент.

Здесь отображаются обложка музыкального альбома, название и исполнитель песни, и кнопки «Play»/«Pause» и «Next».

Ниже приведён макет фрагмента мини-плеера на рисунке 13.

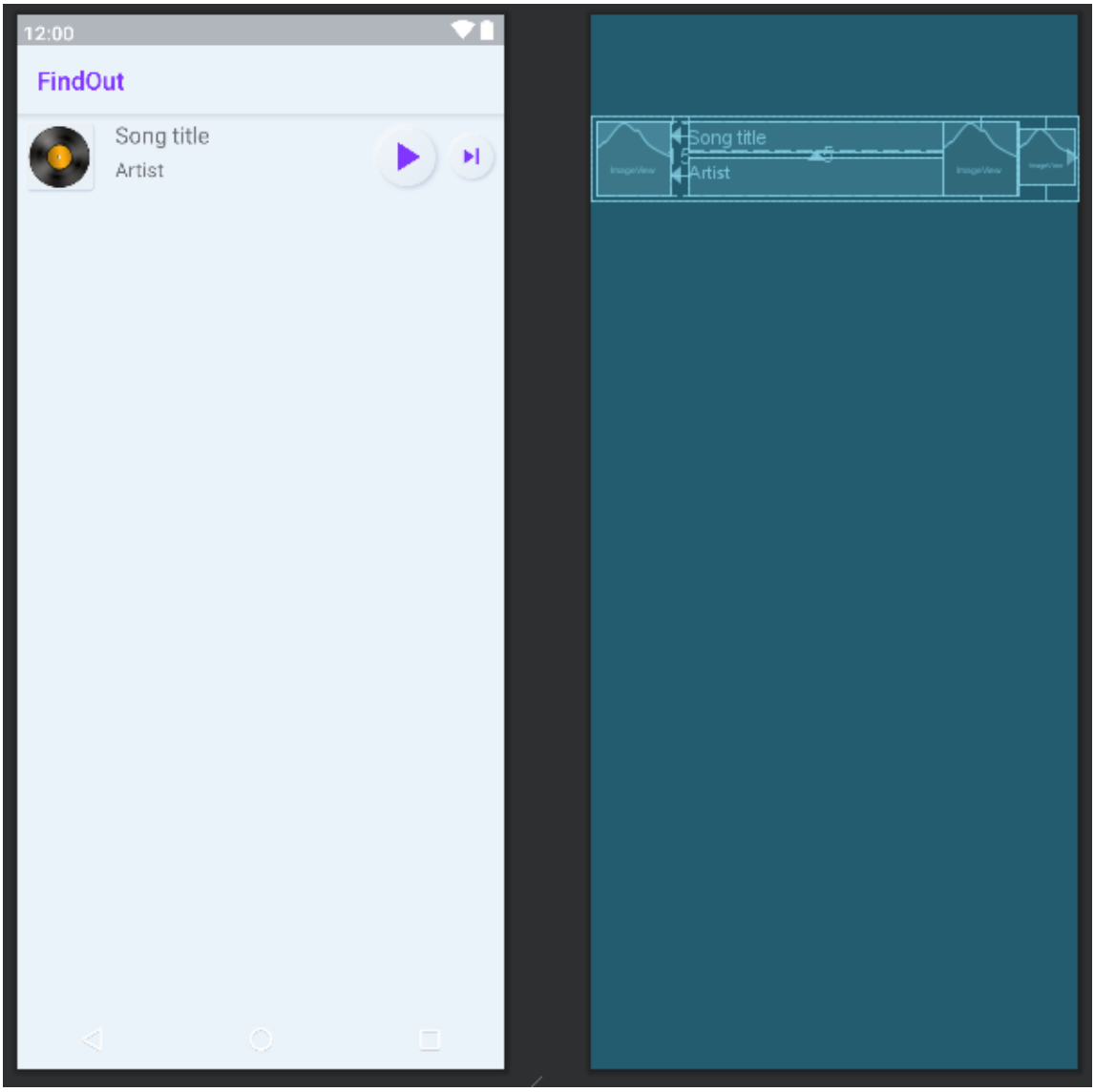


Рисунок 13 – Макет фрагмента мини-плеера

4.2.13 Активность настроек

В SettingsActivity отображаются пользовательские настройки приложения. На данный момент пользователю доступна лишь одна настройка – переключение между светлой и тёмной темой для приложения.

Ниже приведён макет активности настроек на рисунке 14.

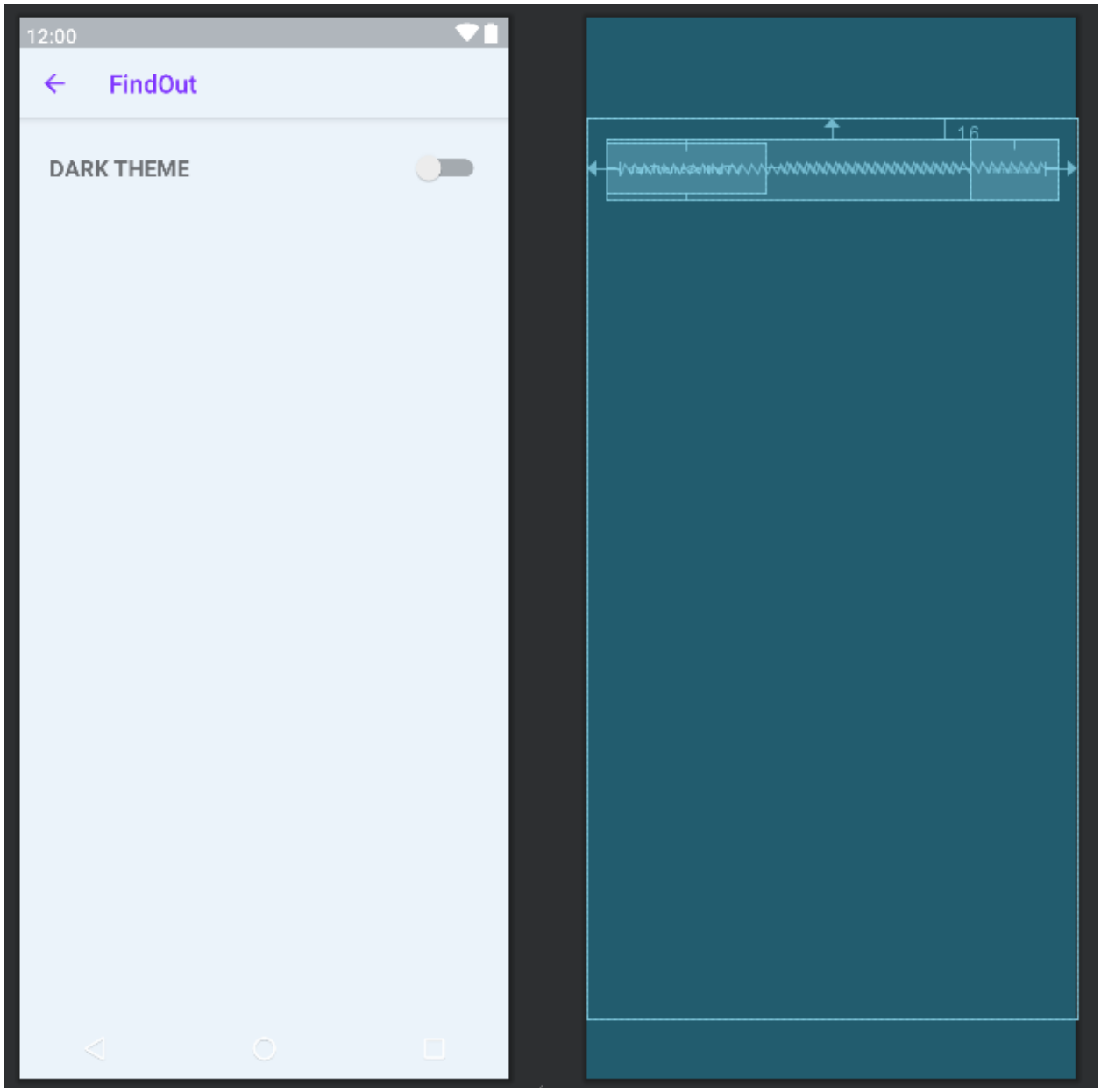


Рисунок 14 – Макет активности настроек

4.3 Схема взаимодействий в приложении

Ниже изображена общая схема взаимодействий между экранами приложения на рисунке 15.

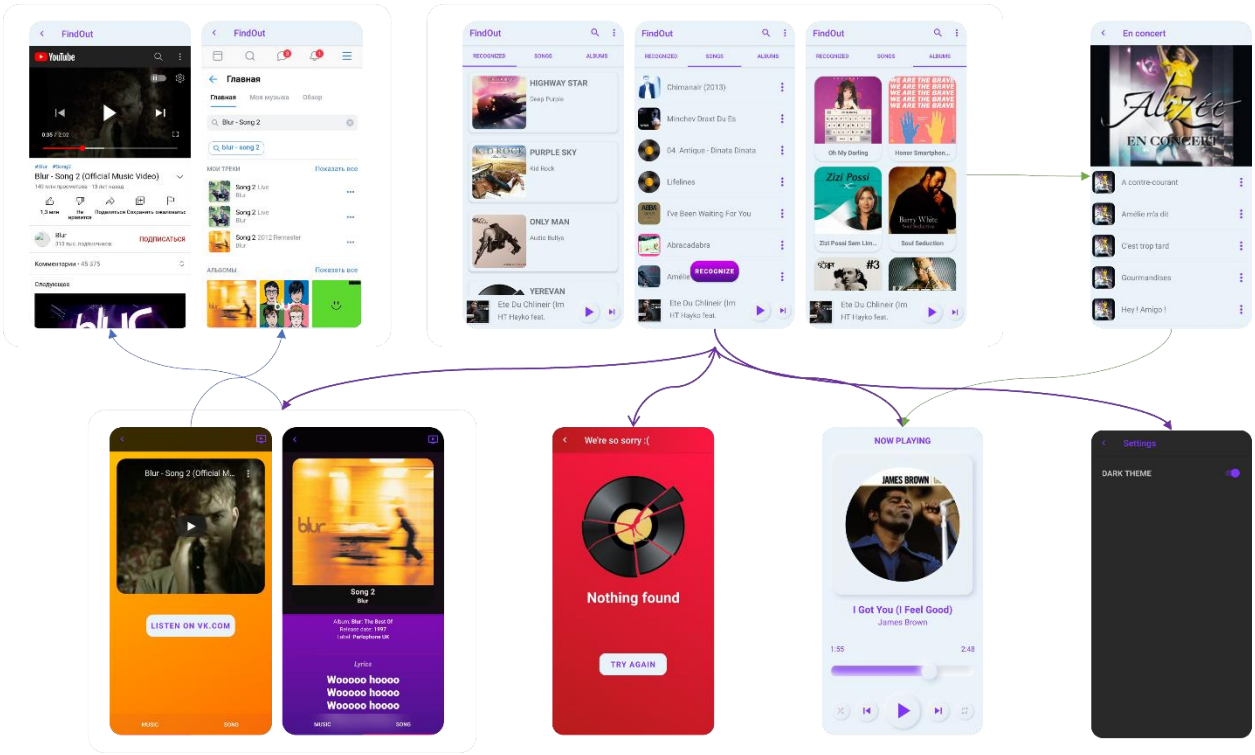


Рисунок 15 – Схема взаимодействий в приложении

4.4 Дизайн

Вопросу UI/UX дизайна пользователя уделялось особое внимание. Некоторые элементы дизайна разработаны при помощи дизайнерского инструмента Figma. Также приложение поддерживает 2 темы: светлую и тёмную.

Тёмная тема стала популярной в 2018 году после её релиза на macOS Mojave [12]. Её полюбили за удобство для глаз и возможность свежего взгляда на привычные интерфейсы.

Зачем нужна тёмная тема:

– экономия батарей аккумуляторов с OLED-дисплеями,

– комфорт для глаз,

– фокусировка на задачах,

– мода.

Ниже на рисунке 16 приведены элементы интерфейса, созданные в Figma, для светлой и тёмной темы.

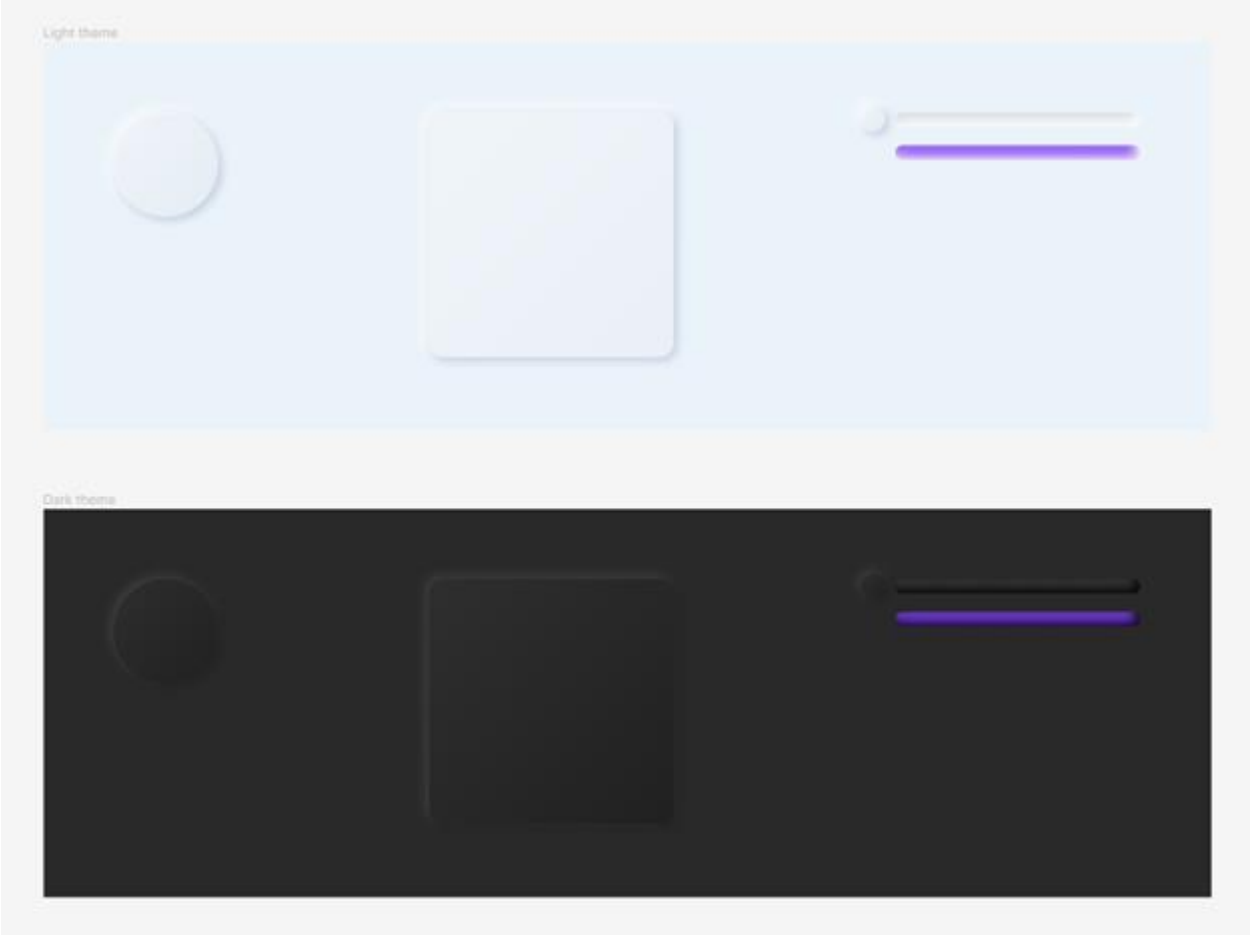


Рисунок 16 – Элементы интерфейса, созданные в Figma

Дизайн приложения частично выполнен в стиле “Neumorphism” (его ещё называют мягким UI) [13].

Особенности стиля:

– минимализм – без перегруженности, ничего лишнего в интерфейсе;

– плавные эффекты – интерфейс переходит из одного состояния в другое, и пользователь видит только переключение экранов;

– сочетание теней – кнопки и другие элементы управления создаются с помощью внутренних и внешних теней;

– красота и эффектность – напрямую влияют на то, скачает ли пользователь приложение или нет.

В интерфейсе преобладают закруглённые углы у различных её элементов. И это не только для красоты. Закруглённые углы кажутся менее броскими. Закруглённые углы легче обрабатывать. Исследования показали, что прямоугольники со скруглёнными углами меньше раздражают глаза, чем прямоугольники с острыми краями, потому что для их визуальной обработки требуется меньше когнитивных усилий [14].

Иконка приложения – это виниловый диск на фиолетовом фоне.

Фон активности неудавшегося распознавания красного цвета, фон фрагмента информации о композиции фиолетовый, а фон фрагмента прослушивания музыки – оранжевый. Все эти цвета градиентные.

5 Программирование

Далее описаны основные моменты в разработке данного приложения.

Все активности расширяют класс *AppCompatActivity* – это базовый класс для активностей, которые используют функциональные возможности панели приложения из библиотеки поддержки (простыми словами этот класс используется для того, чтобы внедрить новые "плюшки" андроида в старые версии).

5.1 Главная активность приложения

5.1.1 Распознавание композиций

Именно в MainActivity происходит запуск распознавания музыки. В разделе 2.2 было рассказано, как настроить эту активность, чтобы она могла использовать сервис распознавания платформы ACRCloud (инструкция ACRCLoud [7]).

Приведём основной код (на рисунках 17–20), который получился после выполнения инструкций ACRCloud [7].

```
public class MainActivity
        extends AppCompatActivity
        implements IACRCloudListener, IACRCloudRadioMetadataListener,
            SearchView.OnQueryTextListener {

    /**
     * ACRCloud fields
     */
    private static final String TAG = "MainActivity";
    private static final int TRY_AGAIN_REQUEST = 1;  // Код запроса на повторную попытку распознания

    private boolean mProcessing = false;
    private boolean initState = false;

    private String path = "";

    private ACRCloudConfig mConfig = null;
    private ACRCloudClient mClient = null;

```

Рисунок 17 – Код инициализации сервиса распознавания (часть 1)

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        recognizedFragmentRootLayout = findViewById(R.id.fragment_recognized_root_layout);

        path = Environment.getExternalStorageState() + "/acrcloud";
        Log.e(TAG, path);

        File file = new File(path);
        if(!file.exists()) {
            file.mkdirs();
        }

        findViewById(R.id.start).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (!mProcessing) {
                    start();
                } else {
                    cancel();
                }
            }
        });

        verifyPermissions();

```

Рисунок 18 – Код инициализации сервиса распознавания (часть 2)

```
157
158         this.mConfig = new ACRCloudConfig();
159         this.mConfig.acrcloudListener = this;
160         this.mConfig.context = this;
161
162         // Please create project in "http://console.acrcloud.cn/service/avr".
163         this.mConfig.host = "identify-eu-west-1.acrcloud.com";
164         this.mConfig.accessKey = "[redacted]";
165         this.mConfig.accessSecret = "[redacted]";
166
167         // auto recognize access key
168         this.mConfig.hostAuto = "";
169         this.mConfig.accessKeyAuto = "";
170         this.mConfig.accessSecretAuto = "";
171
172         this.mConfig.recorderConfig.rate = 8000;
173         this.mConfig.recorderConfig.channels = 1;
174
175         // If you do not need volume callback, you set it false.
176         this.mConfig.recorderConfig.isVolumeCallback = true;
177
178         this.mClient = new ACRCloudClient();
179         ACRCloudLogger.setLog(true);
180         this.initState = this.mClient.initWithConfig(this.mConfig);
181
```

Рисунок 19 – Код инициализации сервиса распознавания (часть 3)

```
377
378     private static final int REQUEST_EXTERNAL_STORAGE = 1;
379
380     private static String[] PERMISSIONS = {
381             Manifest.permission.ACCESS_NETWORK_STATE,
382             Manifest.permission.ACCESS_WIFI_STATE,
383             Manifest.permission.INTERNET,
384             Manifest.permission.RECORD_AUDIO,
385             Manifest.permission.WRITE_EXTERNAL_STORAGE
386     };
387
388     public void verifyPermissions() {
389         for (String perm : PERMISSIONS) {
390             int permission = ActivityCompat.checkSelfPermission( context: this, perm);
391
392             if (permission != PackageManager.PERMISSION_GRANTED) {
393                 ActivityCompat.requestPermissions( activity: this, PERMISSIONS, REQUEST_EXTERNAL_STORAGE);
394                 break;
395             }
396         }
397     }
398
```

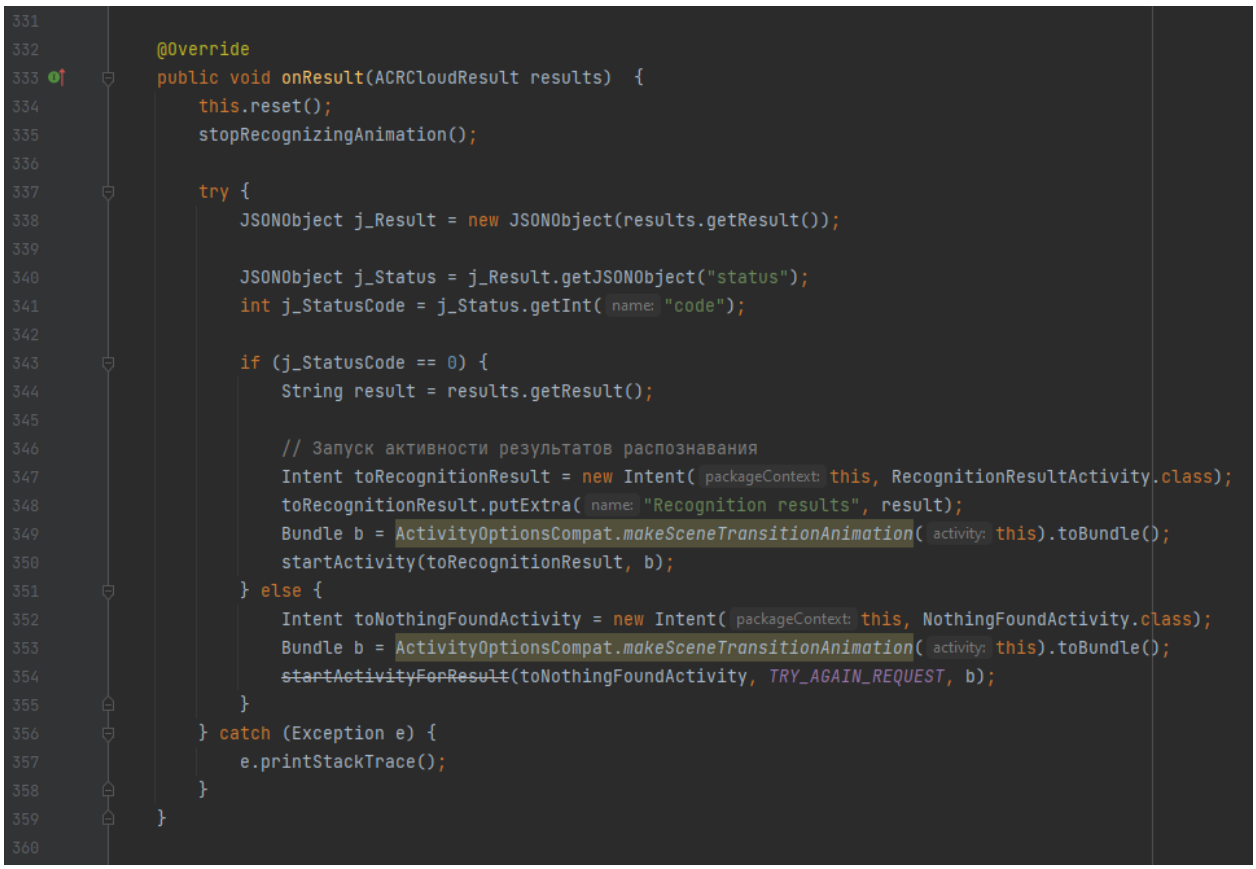
Рисунок 20 – Код инициализации сервиса распознавания (часть 4)

Активность обязательно должна реализовывать интерфейсы IACRCloudListener, IACRCloudRadioMetadataListener. Это интерфейсы из

библиотеки от ACRCloud. Именно они содержат методы, позволяющие распознавать музыку и управлять результатами этого распознавания.

После того как написан этот код, становится возможным идентификация музыки.

По окончании процесса распознавания вызывается метод onResult (код приведён на рисунке 21), в котором нужно указать дальнейшие инструкции, которые должно выполнить приложение. В нашем случае, прекращается анимация процесса распознавания, проверяется статус распознавания (удачное/неудачное) и, в случае удачи – осуществляется переход к активности результатов распознавания, с передачей ей результатов, полученных от сервера ACRCloud, для их извлечения из JSON формата в текстовый и отображения пользователю, а в случае неудачи – переход к активности неудавшегося распознавания, чтобы сообщить пользователю о неудаче и предложить повторить попытку, либо просто вернуться обратно на главную активность приложения.



```
@Override
public void onResult(ACRCloudResult results)  {
    this.reset();
    stopRecognizingAnimation();

    try {
        JSONObject j_Result = new JSONObject(results.getResult());

        JSONObject j_Status = j_Result.getJSONObject("status");
        int j_StatusCode = j_Status.getInt( name: "code");

        if (j_StatusCode == 0) {
            String result = results.getResult();

            // Запуск активности результатов распознавания
            Intent toRecognitionResult = new Intent( packageContext: this, RecognitionResultActivity.class);
            toRecognitionResult.putExtra( name: "Recognition results", result);
            Bundle b = ActivityOptionsCompat.makeSceneTransitionAnimation( activity: this).toBundle();
            startActivity(toRecognitionResult, b);
        } else {
            Intent toNothingFoundActivity = new Intent( packageContext: this, NothingFoundActivity.class);
            Bundle b = ActivityOptionsCompat.makeSceneTransitionAnimation( activity: this).toBundle();
            startActivityForResult(toNothingFoundActivity, TRY_AGAIN_REQUEST, b);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Рисунок 21 – Метод onResult

5.1.2 Фрагмент истории распознаваний

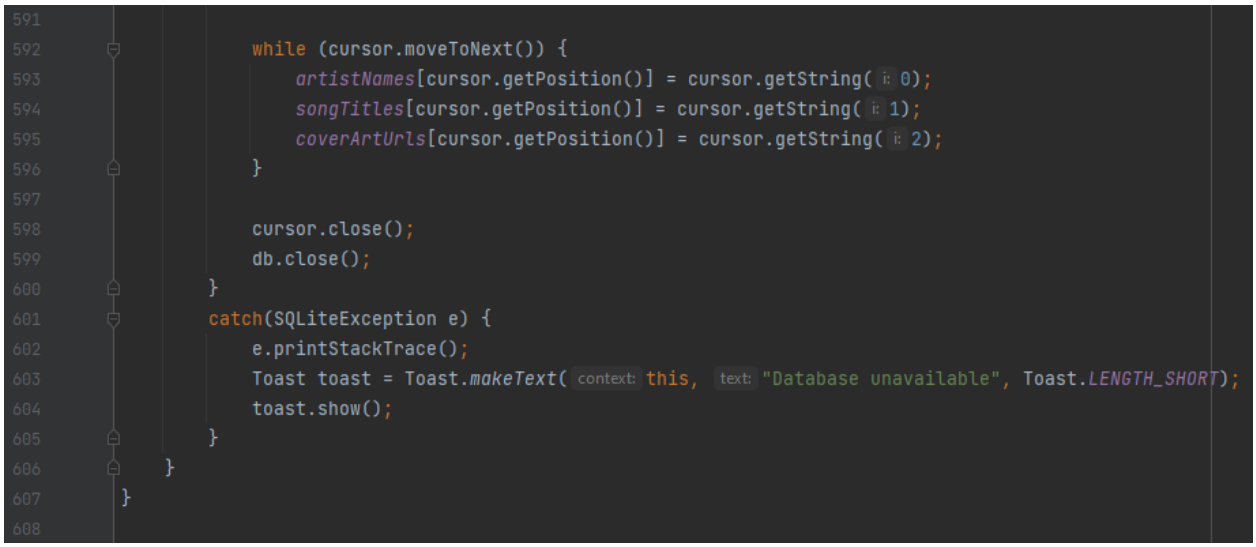
В этом фрагменте отображаются все распознанные приложением музыкальные композиции. Для этого нужно прочитать всю информацию из базы данных (в порядке убывания, чтобы в самом верху были самые свежие распознавания). Само чтение данных происходит в MainActivity, которая потом передаёт эти данные в RecognizedFragment.

Ниже приведён код чтения из БД на рисунке 22.

```
544
545     private void getRecognizedCollection() {
546
547         SQLiteOpenHelper findOutDatabaseHelper = new FindOutDatabaseHelper( context: this);
548
549         try {
550             SQLiteDatabase db = findOutDatabaseHelper.getReadableDatabase();
551
552             //Код чтения данных из базы
553             Cursor cursor = db.query( table: "COLLECTION",
554                     new String[] {"ARTISTS", "SONG_TITLE", "IMAGE_URL"},
555                     selection: null, selectionArgs: null, groupBy: null, having: null,
556                     orderBy: "_id DESC");
557
558             songTitles = new String[cursor.getCount()];
559             artistNames = new String[cursor.getCount()];
560             coverArtUrls = new String[cursor.getCount()];
561
562             if (cursor.moveToFirst())
563             {
564                 if (recognizedFragmentPlaceholder != null) {
565                     recognizedFragmentRootLayout.removeView(recognizedFragmentPlaceholder);
566                     findViewById(R.id.recognized_recycler).setVisibility(View.VISIBLE);
567                     recognizedFragmentPlaceholder = null;
568                 }
569
570                 artistNames[cursor.getPosition()] = cursor.getString( i: 0);
571                 songTitles[cursor.getPosition()] = cursor.getString( i: 1);
572                 coverArtUrls[cursor.getPosition()] = cursor.getString( i: 2);
```

Рисунок 22 – Чтение самой первой записи из БД

Далее, после первой записи, читаются остальные, если они есть. Код приведён на рисунке 23.



```
591
592             while (cursor.moveToNext()) {
593                 artistNames[cursor.getPosition()] = cursor.getString( i: 0);
594                 songTitles[cursor.getPosition()] = cursor.getString( i: 1);
595                 coverArtUrls[cursor.getPosition()] = cursor.getString( i: 2);
596             }
597
598             cursor.close();
599             db.close();
600         }
601         catch(SQLiteException e) {
602             e.printStackTrace();
603             Toast toast = Toast.makeText( context: this, text: "Database unavailable", Toast.LENGTH_SHORT);
604             toast.show();
605         }
606     }
607 }
608
```

Рисунок 23 – Чтение оставшихся записей из БД

Чтобы записи отображались в виде списка карточек, был создан специальный адаптер и применён к корневому элементу макета данного фрагмента. Код приведён на рисунке 24.

```
53
54          SongCardAdapter adapter = new SongCardAdapter(songTitles, artistNames, coverArtUrls);
55          recognizedRecycler.setAdapter(adapter);
56
57          LinearLayoutManager layoutManager = new LinearLayoutManager(getContext());
58          recognizedRecycler.setLayoutManager(layoutManager);
59
```

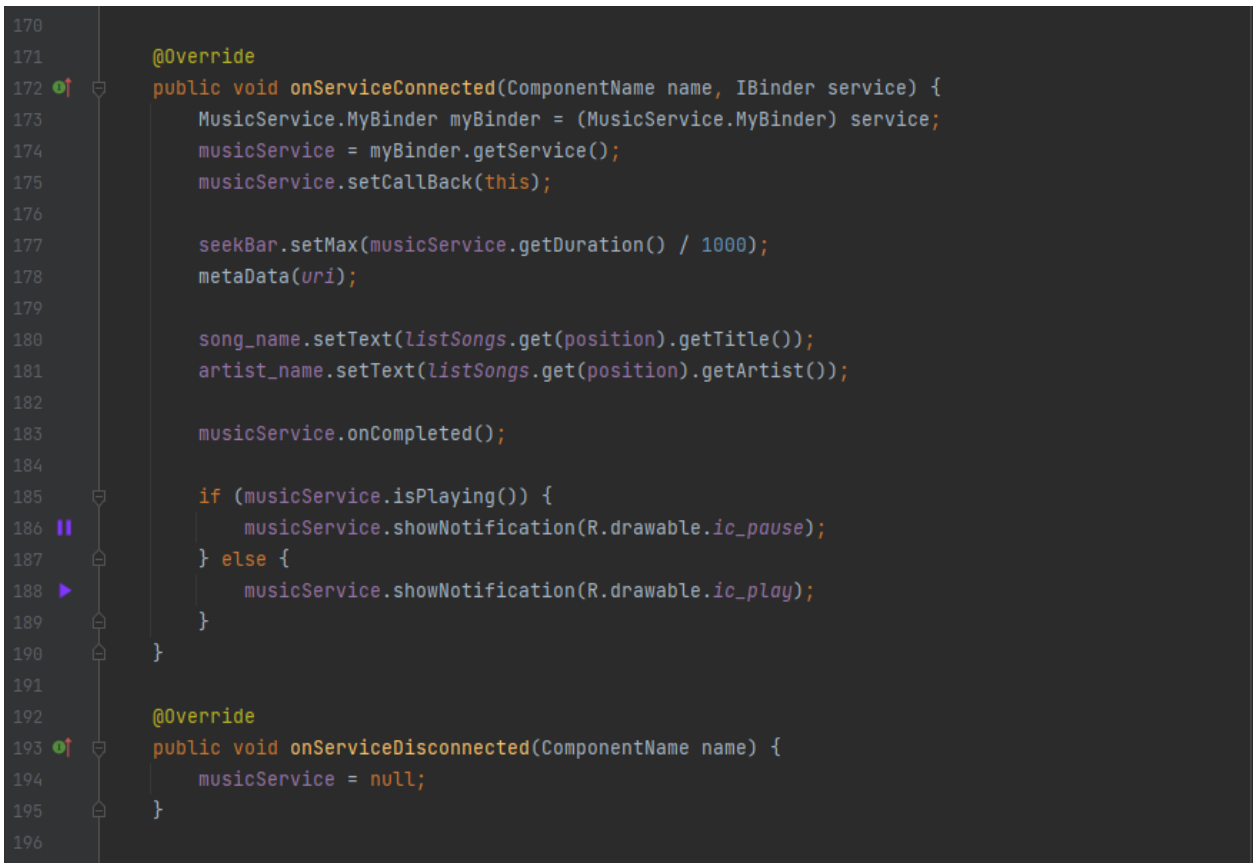
Рисунок 24 – Применение к макету фрагмента адаптера карточного списка

Этому адаптеру в конструкторе передаются массивы с названиями композиций, именами исполнителей и ссылками на изображения обложек альбомов.

5.2 Музыкальный плеер

Основное взаимодействие с музыкальным плеером происходит через активность PlayerActivity. Эта активность реализует интерфейс ServiceConnection. Он нужен для того, чтобы контролировать состояние музыкального сервиса MusicService.

На рисунке 25 приведён код методов интерфейса ServiceConnection, выполняемый при подключении и отключении сервиса.



```
170
171     @Override
172     public void onServiceConnected(ComponentName name, IBinder service) {
173         MusicService.MyBinder myBinder = (MusicService.MyBinder) service;
174         musicService = myBinder.getService();
175         musicService.setCallBack(this);
176
177         seekBar.setMax(musicService.getDuration() / 1000);
178         metaData(uri);
179
180         song_name.setText(listSongs.get(position).getTitle());
181         artist_name.setText(listSongs.get(position).getArtist());
182
183         musicService.onCompleted();
184
185         if (musicService.isPlaying()) {
186             musicService.showNotification(R.drawable.ic_pause);
187         } else {
188             musicService.showNotification(R.drawable.ic_play);
189         }
190     }
191
192     @Override
193     public void onServiceDisconnected(ComponentName name) {
194         musicService = null;
195     }
196
```

Рисунок 25 – Методы интерфейса ServiceConnection

Здесь подключается музыкальный сервис и устанавливаются состояния элементов интерфейса, созданных для взаимодействия пользователя с музыкальным плеером.

Сервис MusicService позволяет создать медиаплеер при помощи класса MediaPlayer (код приведён на рисунке 26) и управлять его состоянием при помощи методов класса MediaPlayer (код приведён на рисунке 27).

```
121
122     public void createMediaPlayer(int positionInner) {
123         position = positionInner;
124         uri = Uri.parse(musicFiles.get(position).getPath());
125
126         SharedPreferences.Editor editor = getSharedPreferences(MUSIC_LAST_PLAYED, MODE_PRIVATE)
127                 .edit();
128         editor.putString(MUSIC_FILE, uri.toString());
129         editor.putString(ARTIST_NAME, musicFiles.get(position).getArtist());
130         editor.putString(SONG_NAME, musicFiles.get(position).getTitle());
131         editor.apply();
132
133         mediaPlayer = MediaPlayer.create(getBaseContext(), uri);
134     }
135
```

Рисунок 26 – Создание медиаплеера

```
135
136     public void release() { mediaPlayer.release(); }
139
140     public void start() { mediaPlayer.start(); }
143
144     public void pause() { mediaPlayer.pause(); }
147
148     public void stop() { mediaPlayer.stop(); }
151
152     public boolean isPlaying() {
153         if (mediaPlayer != null) {
154             return mediaPlayer.isPlaying();
155         } else {
156             return false;
157         }
158     }
159
160     public int getDuration() { return mediaPlayer.getDuration(); }
163
164     public int getCurrentPosition() { return mediaPlayer.getCurrentPosition(); }
167
168     public void seekTo(int position) { mediaPlayer.seekTo(position); }
171
172     public void onCompleted() { mediaPlayer.setOnCompletionListener(this); }
175
```

Рисунок 27 – Часть методов управления состоянием плеера

5.3 Активность неудавшегося распознавания

В этой активности пользователю сообщается о том, что в результате попытки идентификации композиции ничего не найдено. Также здесь предлагается либо просто вернуться на главную активность с помощью кнопки

«Вверх», либо попробовать распознать музыку ещё раз с помощью кнопки «TRY AGAIN».

В случае если пользователь захочет попробовать снова, приложение вернётся на главную активность и автоматически повторно запустится процесс распознавания.

Рассмотрим код, который позволяет нам добиться повторного распознавания. Для начала необходимо запустить данную активность из главной методом startActivityForResult (это можно увидеть на рисунке выше). Данный метод применяется, когда возникает необходимость вызвать активность, выполнить в нём какое-либо действие и вернуться с результатом. Отличие от обычного *startActivity* (запускает указанную в интенте активность) в том, что запускающая активность (MainActivity) становится «родителем» для запускаемой активности (NothingFoundActivity). И когда NothingFoundActivity закрывается, вызывается метод onActivityResult (код приведён на рисунке 28) в MainActivity, тем самым давая нам знать, что закрылась активность, которую мы вызывали методом startActivityForResult.

В startActivityForResult в качестве параметров мы передаём Intent и requestCode (необходим для идентификации), а также дополнительные данные.

```
360
361     @Override
362     protected void onActivityResult(int requestCode, int resultCode, Intent data) {
363         super.onActivityResult(requestCode, resultCode, data);
364
365         if (requestCode == TRY_AGAIN_REQUEST) {
366             if (resultCode == RESULT_OK) {
367                 start();
368             }
369         }
370     }
371
```

Рисунок 28 – Метод onActivityResult

В onActivityResult мы видим следующие параметры:

– requestCode – тот же идентификатор, что и в методе startActivityForResult. Он используется для того, чтобы отличать друг от друга пришедшие результаты. По нему мы определяем, из какой активности пришёл результат;

– resultCode – код возврата (результата). Он определяет успешно ли прошёл вызов;

– data – интент, в котором возвращаются данные (если они есть).

В нашем случае requestCode должен равняться значению константы TRY_AGAIN_REQUEST (= 1), чтобы в случае необходимости повторно запустить процесс распознавания. И если это условие выполняется, то проверяется resultCode, который определяется действием пользователя в активности NothingFoundActivity (код на рисунке 29). Если он равен RESULT_OK, то вызывается метод start() для повторного запуска распознавания музыки.

```
44
45      public void TryAgain(View view) {
46          setResult(RESULT_OK);   // Успешное выполнение операции try again
47          finish();               // Вернуться к родительской активности, которая ждёт результат
48      }
49  }
50
```

Рисунок 29 – Обработчик кнопки «TRY AGAIN»

Когда пользователь нажмёт на кнопку повтора попытки распознать композицию, методом setResult параметру resultCode будет присвоено значение RESULT_OK. После этого методом finish() закрываем активность NothingFoundActivity.

5.4 Активность результатов распознавания

В этой активности показываются результаты распознавания. При удачном распознавании в методе onResult главной активности результаты поме-

щаются в интент, которым запускается данная активность (код приведён выше на рисунке 21). Далее в активности результатов распознавания (RecognitionResultActivity) эти результаты извлекаются из этого интента (код приведён на рисунке 30) для дальнейшей обработки.

```
68
69      @Override
70      protected void onCreate(Bundle savedInstanceState) {
71          supportRequestWindowFeature(Window.FEATURE_ACTION_BAR_OVERLAY);
72          super.onCreate(savedInstanceState);
73          setContentView(R.layout.activity_recognition_result);
74
75          recognitionResults = getIntent().getStringExtra( name: "Recognition results");
76          recognitionResultsParsing();    // Парсинг результатов распознавания
77
78          String queryForVK = songArtists + " - " + songTitle;
79
80          musicFragment = new MusicFragment(queryForVK);
81          songFragment =  new SongFragment(songTitle,
82                  songArtists,
83                  albumInfo,
84                  composersInfo,
85                  lyricistInfo,
86                  releaseDateInfo,
87                  labelInfo,
88                  timestampInfo
89          );
90
```

Рисунок 30 – Метод onCreate (часть 1)

В начале метода onCreate (он вызывается при создании всех активностей) извлекаются результаты распознавания из интента, которым была запущена данная активность, и вызывается метод recognitionResultsParsing() для обработки этих результатов. После этого в соответствующих переменных будут записаны нужные нам данные о распознанной композиции. Далее здесь создаётся переменная queryForVK – строка для поиска найденной композиции в каталоге музыки социальной сети «ВКонтакте». А после всего этого создаются объекты фрагментов SongFragment и MusicFragment с передачей им необходимых данных.

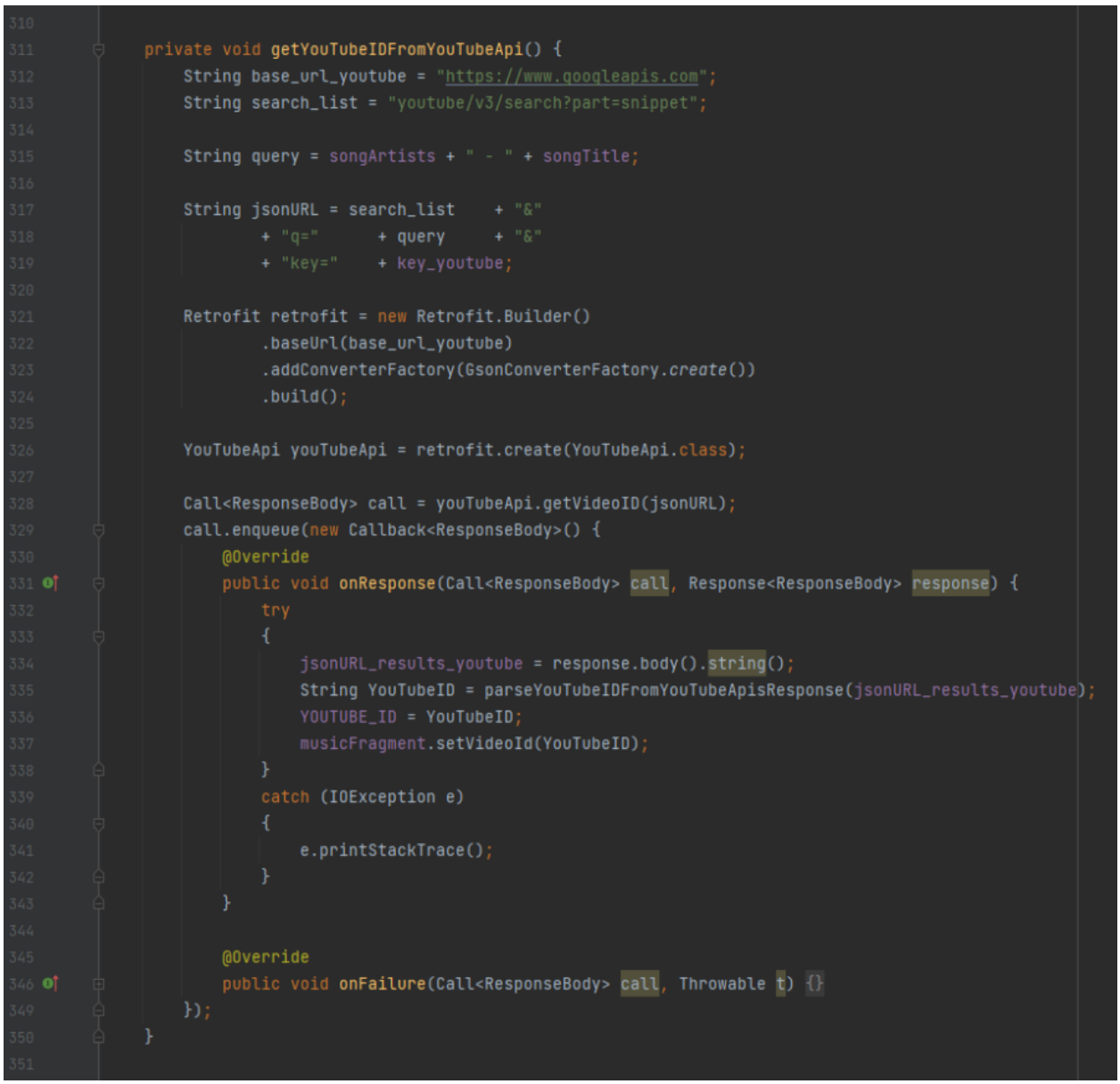
Ниже на рисунке 31 приведена часть метода recognitionResultsParsing(), где из результатов в JSON формате создаётся Java объект и из него извлека-

ются метаданные о найденной композиции. Из объекта этих метаданных в свою очередь извлекаются объекты или массивы объектов, содержащие информацию о названии композиции, исполнителе (-лях) и т. п., которая записывается в строковые переменные, которые и передаются фрагментам SongFragment и MusicFragment в методе onCreate.

В конце этого метода вызывается getYouTubeIDFromYouTubeApi() (код на рисунке 32) – метод для получения идентификатора видеозаписи для распознанной композиции на YouTube с помощью YouTube Data API.

```
211
212     private void recognitionResultsParsing() {
213         String title;
214         String artists = "";
215
216         try {
217             JSONObject j_Result = new JSONObject(recognitionResults);
218             JSONObject j_metadata = j_Result.getJSONObject("metadata");
219
220             if (j_metadata.has( name: "music")) {
221                 JSONArray j_musics = j_metadata.getJSONArray( name: "music");
222                 JSONObject j_song = (JSONObject) j_musics.get(0);
223
224                 // Получаем название песни
225                 title = j_song.getString( name: "title");
226                 songTitle = title;
227
228
229                 // Получаем исполнителя(ей)
230                 JSONArray j_artistsArray = j_song.getJSONArray( name: "artists");
231                 for (int j = 0; j < j_artistsArray.length(); j++) {
232                     JSONObject j_artist = (JSONObject) j_artistsArray.get(j);
233                     artists += j_artist.getString( name: "name");
234
235                     if (j != (j_artistsArray.length() - 1)) {
236                         artists += " feat. ";
237                     }
238                 }
239                 songArtists = artists;
240
```

Рисунок 31 – Часть метода recognitionResultsParsing



```
310
311     private void getYouTubeIDFromYouTubeApi() {
312         String base_url_youtube = "https://www.googleapis.com";
313         String search_list = "youtube/v3/search?part=snippet";
314
315         String query = songArtists + " - " + songTitle;
316
317         String jsonURL = search_list    + "&"
318                 + "q="      + query     + "&"
319                 + "key="    + key_youtube;
320
321         Retrofit retrofit = new Retrofit.Builder()
322                 .baseUrl(base_url_youtube)
323                 .addConverterFactory(GsonConverterFactory.create())
324                 .build();
325
326         YouTubeApi youTubeApi = retrofit.create(YouTubeApi.class);
327
328         Call<ResponseBody> call = youTubeApi.getVideoID(jsonURL);
329         call.enqueue(new Callback<ResponseBody>() {
330             @Override
331             public void onResponse(Call<ResponseBody> call, Response<ResponseBody> response) {
332                 try
333                 {
334                     jsonURL_results_youtube = response.body().string();
335                     String YouTubeID = parseYouTubeIDFromYouTubeApisResponse(jsonURL_results_youtube);
336                     YOUTUBE_ID = YouTubeID;
337                     musicFragment.setVideoId(YouTubeID);
338                 }
339                 catch (IOException e)
340                 {
341                     e.printStackTrace();
342                 }
343             }
344
345             @Override
346             public void onFailure(Call<ResponseBody> call, Throwable t) {}
349         });
350     }
351
```

Рисунок 32 – Метод получения ID видео на YouTube

Здесь была использована библиотека Retrofit 2 – типобезопасный HTTP-клиент для Android и Java. Он является незаменимым инструментом для работы с API в клиент-серверных приложениях.

Для начала определяется URL для запроса, далее создаётся объект Retrofit, которому передаётся базовая часть URL (https://www.googleapis.com), а также создаётся объект интерфейса YouTubeApi (код приведён на рисунке 33), необходимого для работы библиотеки Retrofit. Далее создаётся запрос к API YouTube и выполняется методом *enqueue*. При удачном ответе от серве-

ра извлекается результат в формате JSON, обрабатывается и записывается в переменную YOUTUBE_ID, после значение, содержащееся в ней, передаётся фрагменту прослушивания музыки MusicFragment. Также этот идентификатор передастся через интент активности просмотра видео на YouTube при нажатии на соответствующую кнопку на панели действий (ActionBar).

```
16
17  public interface YouTubeApi {
18      @GET
19      Call<ResponseBody> getVideoID(@Url String url);
20  }
21
```

Рисунок 33 – Интерфейс YouTube API

Далее, в методе onCreate (код на рисунке 34) создаётся SectionsPagerAdapter для отображения фрагментов в этой активности, и этот адаптер задаётся объекту ViewPager (он позволяет листать между фрагментами). Сам ViewPager связывается с вкладками, которые отображаются внизу экрана.

```
91
92          // Связывание SectionsPagerAdapter с ViewPager
93          SectionsPagerAdapter pagerAdapter = new SectionsPagerAdapter(
94                  getSupportFragmentManager(),
95                  BEHAVIOR_RESUME_ONLY_CURRENT_FRAGMENT
96          );
97          ViewPager pager = (ViewPager) findViewById(R.id.pager);
98          pager.setAdapter(pagerAdapter);
99          pager.setCurrentItem(1);        // Сначала показывается SongFragment
100
101         // Связывание ViewPager с TabLayout
102         TabLayout tabLayout = (TabLayout) findViewById(R.id.tabs);
103         tabLayout.setupWithViewPager(pager);
104
```

Рисунок 34 – Метод onCreate (часть 2)

SectionsPagerAdapter – это внутренний класс, расширяющий класс FragmentPagerAdapter. Ниже на рисунке 35 приведён код этого внутреннего класса. В нём переопределяются методы, необходимые для отображения

фрагментов SongFragment и MusicFragment, а также названий соответствующих им вкладок.

```
158
159     /**
160      * Адаптер для ViewPager
161      */
162     private class SectionsPagerAdapter extends FragmentPagerAdapter {
163
164         public SectionsPagerAdapter(@NonNull FragmentManager fm, int behavior) {
165             super(fm, behavior);
166         }
167
168
169         @Override
170 @       public Fragment getItem(int position) {   // Позиция определяет номер страницы (начиная с 0)
171
172             switch (position)
173             {
174                 case 0: return musicFragment;
175                 case 1: return songFragment;
176                 default: return null; // shouldn't happen
177             }
178         }
179
180
181         @Override
182         public int getCount() {
183             return 2;
184         }
185
186
187         private String[] tabsTitles = new String[]{"MUSIC", "SONG"};
188
189         @Override
190         public CharSequence getPageTitle(int position) {
191             return tabsTitles[position];
192         }
193     }
194
```

Рисунок 35 – Внутренний класс SectionsPagerAdapter

5.4.1 Фрагмент информации о композиции

В фрагменте SongFragment отображается информация о распознанной композиции, которая была получена от активности RecognitionResultActivity, а также часть этой информации сохраняется в базу данных для отображения в фрагменте истории распознаваний.

Также в этом фрагменте находится, получается и отображается обложка альбома (если она найдена), в которой состоит найденная композиция, и слова песни (если они найдены). Эти два действия выполняются с помощью методов getDiscogsData() и getLyrics().

В методе getDiscogsData() осуществляется поиск обложки альбома с помощью сервиса Discogs API [15]. Здесь данные также получаются с помощью библиотеки Retrofit 2 (код приведён на рисунке 36).

```
278
279     private void getDiscogsData() {
280         String base_url_discogs = "https://api.discogs.com";
281         String database_search = "/database/search?";
282
283         String query = songArtists + " - " + songTitle;
284         String track = songTitle;
285         String artist = songArtists;
286
287         String jsonURL = database_search
288                         + "q="      + query  + "&"
289                         + "artist"  + artist + "&"
290                         + "track="  + track  + "&"
291                         + "key="    + key_discogs + "&"
292                         + "secret=" + secret_discogs;
293
294
295         Retrofit retrofit = new Retrofit.Builder()
296                 .baseUrl(base_url_discogs)
297                 .addConverterFactory(GsonConverterFactory.create())
298                 .build();
299
300         DiscogsServiseApi discogsServiseApi = retrofit.create(DiscogsServiseApi.class);
301
```

Рисунок 36 – Метод getDiscogsData (часть 1)

Сначала осуществляется поиск нужной композиции в сервисе Discogs, далее из результатов выбирается самый первый (обычно он является наиболее достоверным), потом из него извлекается URL обложки альбома и с помощью библиотеки Picasso это изображение загружается прямо в ImageView, созданное для обложки альбома. Также найденное URL помещается в базу данных для отображения в истории распознанных композиций.

В случае отсутствия изображения на сервисе отображается картинка сломанного винилового диска, а в БД записывается значение «default», чтобы в истории распознаваний тоже отображать эту картинку.

Ниже на рисунке 37 приведён код, выполняющий описанные выше действия.

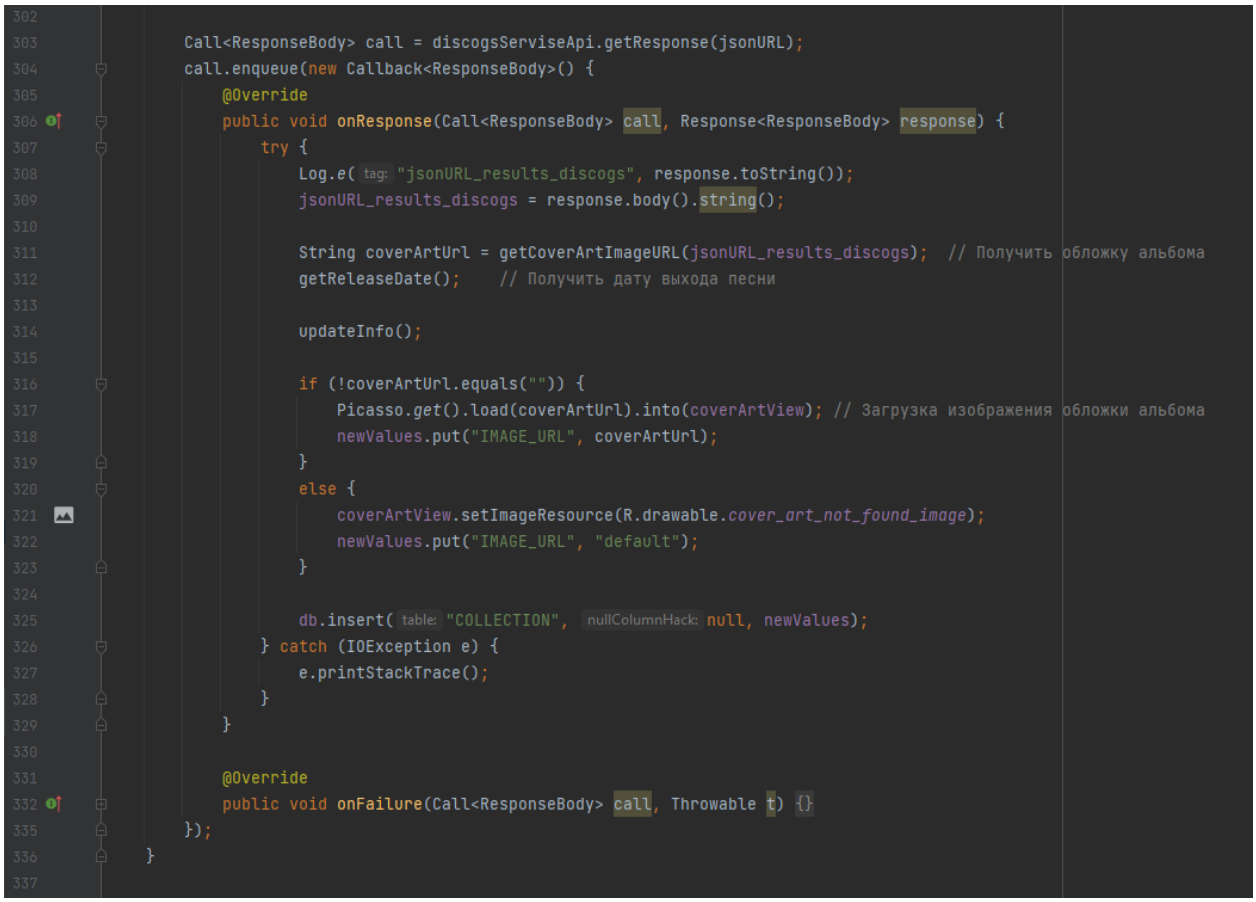


Рисунок 37 – Метод getDiscogsData (часть 2)

Метод getLyrics() получает текст песни от сервиса Musixmatch через предоставляемое им API [16]. Этот метод почти аналогичен методу getDiscogsData(): также поиск и получение информации происходит с помощью Retrofit 2, но результатом является текст песни (если он найден), который отображается в соответствующем TextView. Если текст не найден, этот TextView делается невидимым для пользователя (как будто его и не существовало).

Важно отметить, что текст песни в данном случае неполный, т. к. бесплатно от Musixmatch можно получить только 30 % текста песни. Если нужен весь текст, необходимо оплатить эту услугу.

Для более привлекательного отображения дополнительной информации о композиции для соответствующих этой информации переменных вместо типа String был применён тип SpannableString [17] – класс, который реализует интерфейс Spannable. Этот интерфейс предназначен для стилизации текста (сделать текст жирным, подчёркнутым, отобразить курсивом и т. п.).

5.4.2 Фрагмент прослушивания музыки

В MusicFragment присутствуют только YouTube плеер и кнопка перехода к активности прослушивания музыки в VK.

Плеер YouTube подключается очень просто. Достаточно вызвать метод getYouTubePlayerWhenReady класса YouTubePlayerView из библиотеки android-youtube-player [11]. В параметре этого метода – лямбда-выражение, в котором мы получаем сам плеер и загружаем видеозапись по её ID. Следующий код на рисунке 38 показывает все эти действия.

```
72
73      @Override
74      public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
75          super.onViewCreated(view, savedInstanceState);
76
77          youTubePlayerView = view.findViewById(R.id.youtube_player);
78          youTubePlayerView.getYouTubePlayerWhenReady(youTubePlayer -> {
79              this.youTubePlayer = youTubePlayer;
80              this.youTubePlayer.cueVideo(VIDEO_ID, v: 0.0f);
81          });
82      }
83
```

Рисунок 38 – Получение плеера YouTube и загрузка видео

5.5 Активности с WebView для YouTube и VK

Таких активностей всего 2: активность прослушивания музыки в «ВКонтакте» (VKMusicActivity) и активность просмотра видео на YouTube (WatchVideoActivity). Они устроены одинаково. Различаются только веб-адреса.

Рассмотрим активности с WebView на примере активности прослушивания музыки на сайте «ВКонтакте» (VKMusicActivity) (код на рисунке 39).

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_vk_music);

    QUERY = getIntent().getExtras().getString( key: "QUERY");

    ActionBar actionBar = getSupportActionBar();
    actionBar.setDisplayHomeAsUpEnabled(true);
    actionBar.setHomeAsUpIndicator(R.drawable.ic_chevron_left);

    myWebView = findViewById(R.id.web_view_vk);
    mWebChromeClient = new MyWebChromeClient();
    myWebView.setWebChromeClient(mWebChromeClient);
    myWebView.setWebViewClient(shouldOverrideUrlLoading(view, url) -> {
            return false;
    });

    WebSettings webSettings = myWebView.getSettings();
    webSettings.setJavaScriptEnabled(true);
    myWebView.loadUrl("https://m.vk.com/audio?q=" + QUERY);
}
```

Рисунок 39 – Метод onCreate активности VKMusicActivity

Чтобы использовать WebView для воспроизведения видео или прослушивания музыки вместо перехода на внешний веб-сайт, создаётся свой клиент WebView (MyWebChromeClient) для замены стандартного клиента WebView для представления WebView [18].

С помощью веб-настроек WebView включён Javascript и загружена строка с необходимым адресом в экземпляр веб-просмотра с помощью метода loadUrl() представления WebView.

MyWebChromeClient – внутренний класс расширяющий класс WebChromeClient. В нём определены настройки макета для WebView и некоторые другие настройки.

5.6 Пользовательские настройки приложения

Пользователю предоставлена возможность переключаться между светлой и тёмной темой приложения через активность настроек SettingsActivity.

При включении/отключении переключателя настройки темы соответствующая настройка сохраняется при помощи интерфейса SharedPreferences и применяется при помощи метода AppCompatDelegate.setDefaultNightMode() (код приведён на рисунке 40).

```
39
40      private void initSwitchListener() {
41          themeSwitch.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
42              @Override
43              public void onCheckedChanged(CompoundButton compoundButton, boolean checked) {
44                  if (checked) {
45                      settings.setCustomTheme(ApplicationClass.DARK_THEME);
46                  } else {
47                      settings.setCustomTheme(ApplicationClass.LIGHT_THEME);
48                  }
49
50                  SharedPreferences.Editor editor = getSharedPreferences(ApplicationClass.PREFERENCES, MODE_PRIVATE).edit();
51                  editor.putString(ApplicationClass.CUSTOM_THEME, settings.getCustomTheme());
52                  editor.apply();
53
54                  updateView();
55              }
56          });
57      }
58
59      private void updateView() {
60          if (settings.getCustomTheme().equals(ApplicationClass.LIGHT_THEME)) {
61              AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_NO);
62              themeSwitch.setChecked(false);
63          } else {
64              AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_YES);
65              themeSwitch.setChecked(true);
66          }
67      }
68  }
69
```

Рисунок 40 – Установка настройки темы приложения

Далее при запуске приложения в классе ApplicationClass типа Application, контролирующего глобальное состояние приложения, применяется та тема, которая была выбрана пользователем (код на рисунке 41).

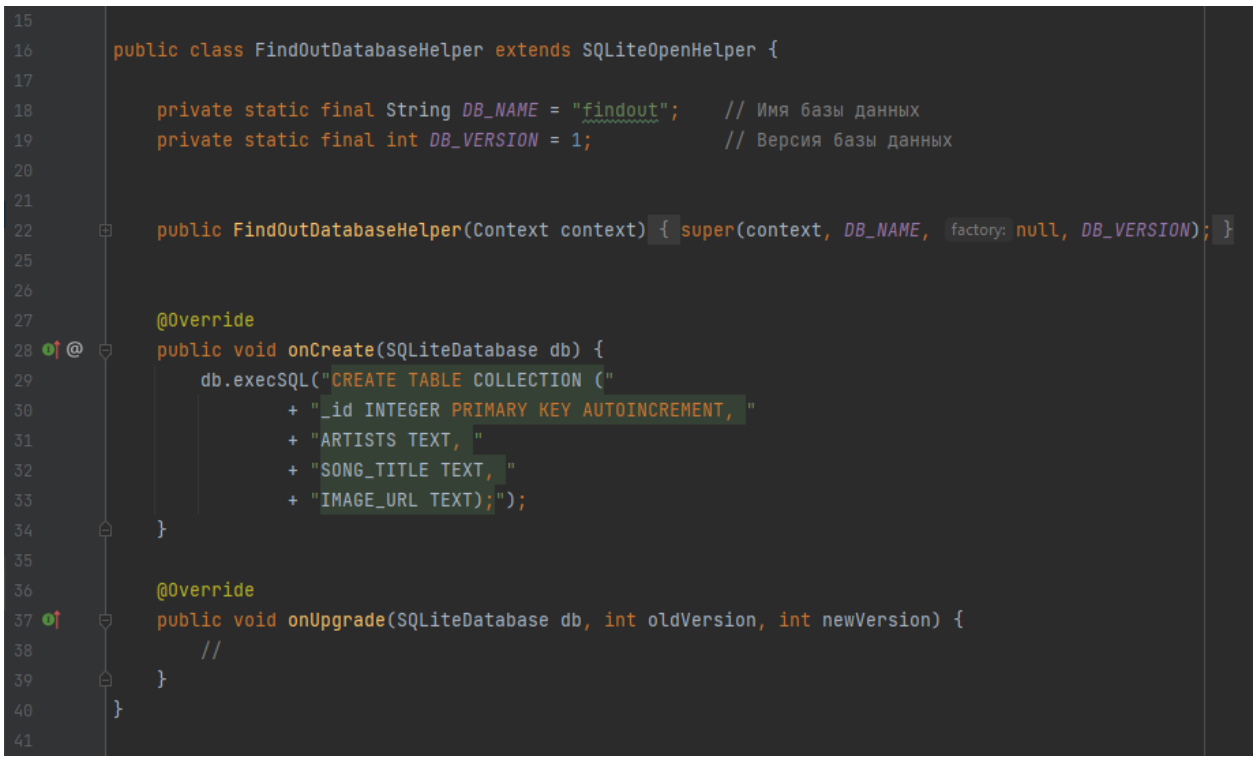
```
    public static final String PREFERENCES = "preferences";
    public static final String CUSTOM_THEME = "customTheme";
    public static final String LIGHT_THEME = "lightTheme";
    public static final String DARK_THEME = "darkTheme";

    private String customTheme;

    @Override
    public void onCreate() {
        super.onCreate();
        createNotificationChannel();

        SharedPreferences sharedPreferences = getSharedPreferences(ApplicationClass.PREFERENCES, MODE_PRIVATE);
        String theme = sharedPreferences.getString(ApplicationClass.CUSTOM_THEME, ApplicationClass.LIGHT_THEME);
        setCustomTheme(theme);
        if (customTheme.equals(ApplicationClass.LIGHT_THEME)) {
            AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_NO);
        } else {
            AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_YES);
        }
    }
```

Рисунок 41 – Применение настройки темы при запуске приложения

5.7 База данных

В приложении используется база данных SQLlite. Она представляет собой файл с расширением *.db*.

Ниже на рисунке 42 приведён код БД.

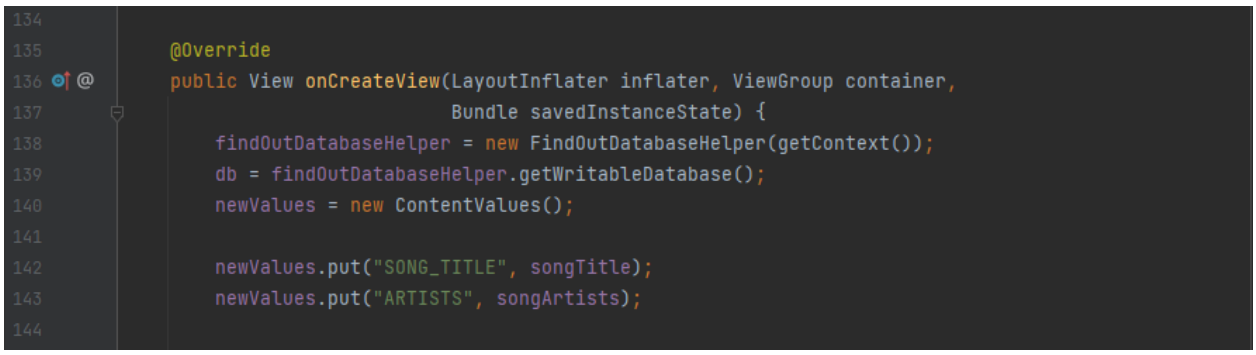


```
public class FindOutDatabaseHelper extends SQLiteOpenHelper {

    private static final String DB_NAME = "findout";    // Имя базы данных
    private static final int DB_VERSION = 1;            // Версия базы данных


    public FindOutDatabaseHelper(Context context) { super(context, DB_NAME, factory: null, DB_VERSION); }


    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("CREATE TABLE COLLECTION ("
                + "_id INTEGER PRIMARY KEY AUTOINCREMENT, "
                + "ARTISTS TEXT, "
                + "SONG_TITLE TEXT, "
                + "IMAGE_URL TEXT);");
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        //
    }
}
```

Рисунок 42 – Код базы данных SQLite

В методе onCreate (здесь он вызывается только 1 раз, когда БД только создаётся) создаётся единственная таблица с колонками _id, ARTISTS, SONG_TITLE и IMAGE_URL для хранения идентификатора записи, имён исполнителей, названий песен и ссылок на обложки альбомов (если они есть) соответственно.

Чтобы создать запись необходимо выполнить следующий код на рисунках 43 и 44 (это делается в фрагменте SongFragment).



```
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        findOutDatabaseHelper = new FindOutDatabaseHelper(getContext());
        db = findOutDatabaseHelper.getWritableDatabase();
        newValues = new ContentValues();

        newValues.put("SONG_TITLE", songTitle);
        newValues.put("ARTISTS", songArtists);

```

Рисунок 43 – Создание записи в БД (часть 1)

Рисунок 44 – Создание записи в БД (часть 2)

6 Тестирование

Ниже на рисунках 45–50 приведены экраны приложения в светлой и тёмной темах, кроме экранов, относящихся к процессу распознавания музыки (они будут приведены отдельно).

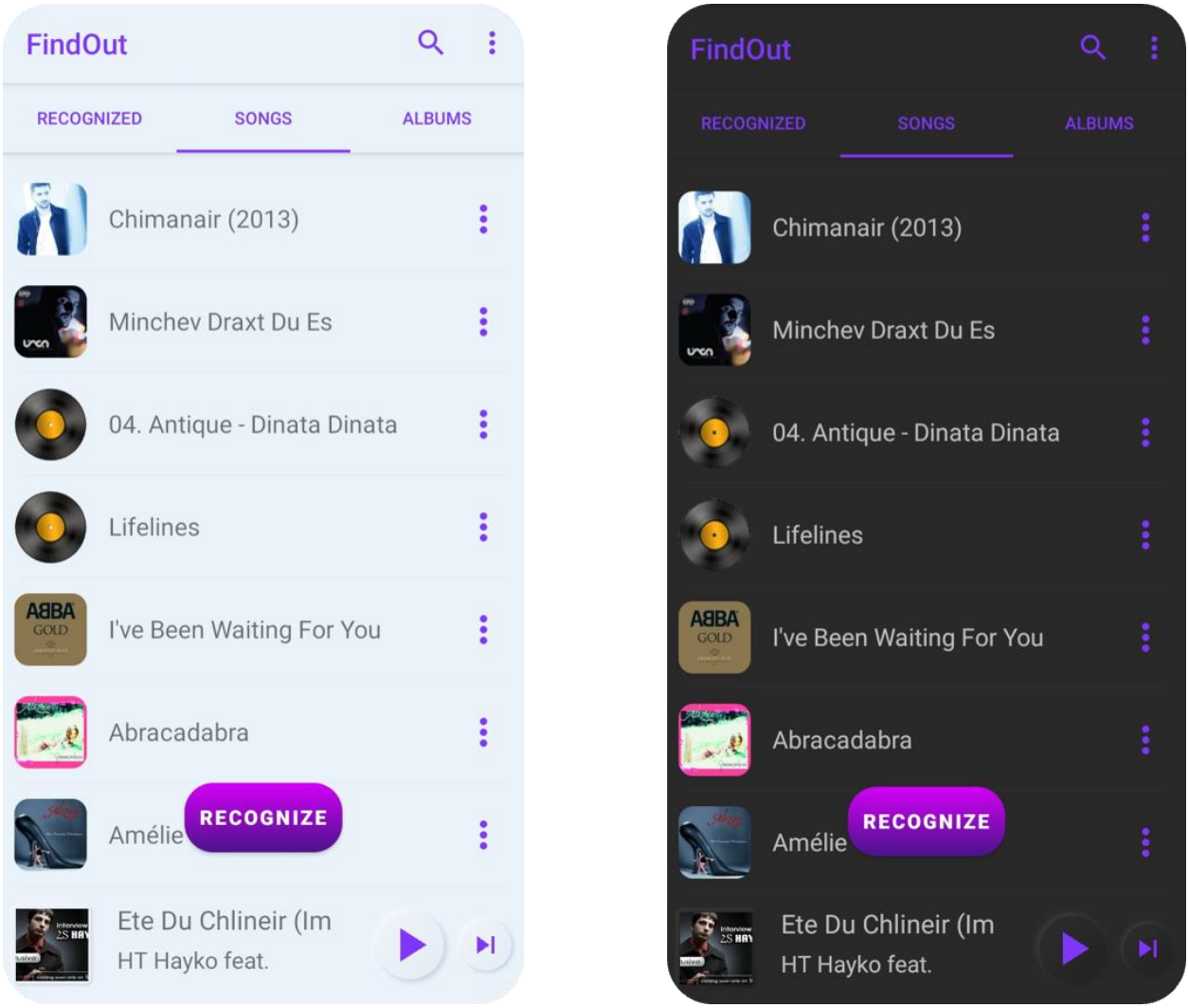


Рисунок 45 – Музыка на устройстве

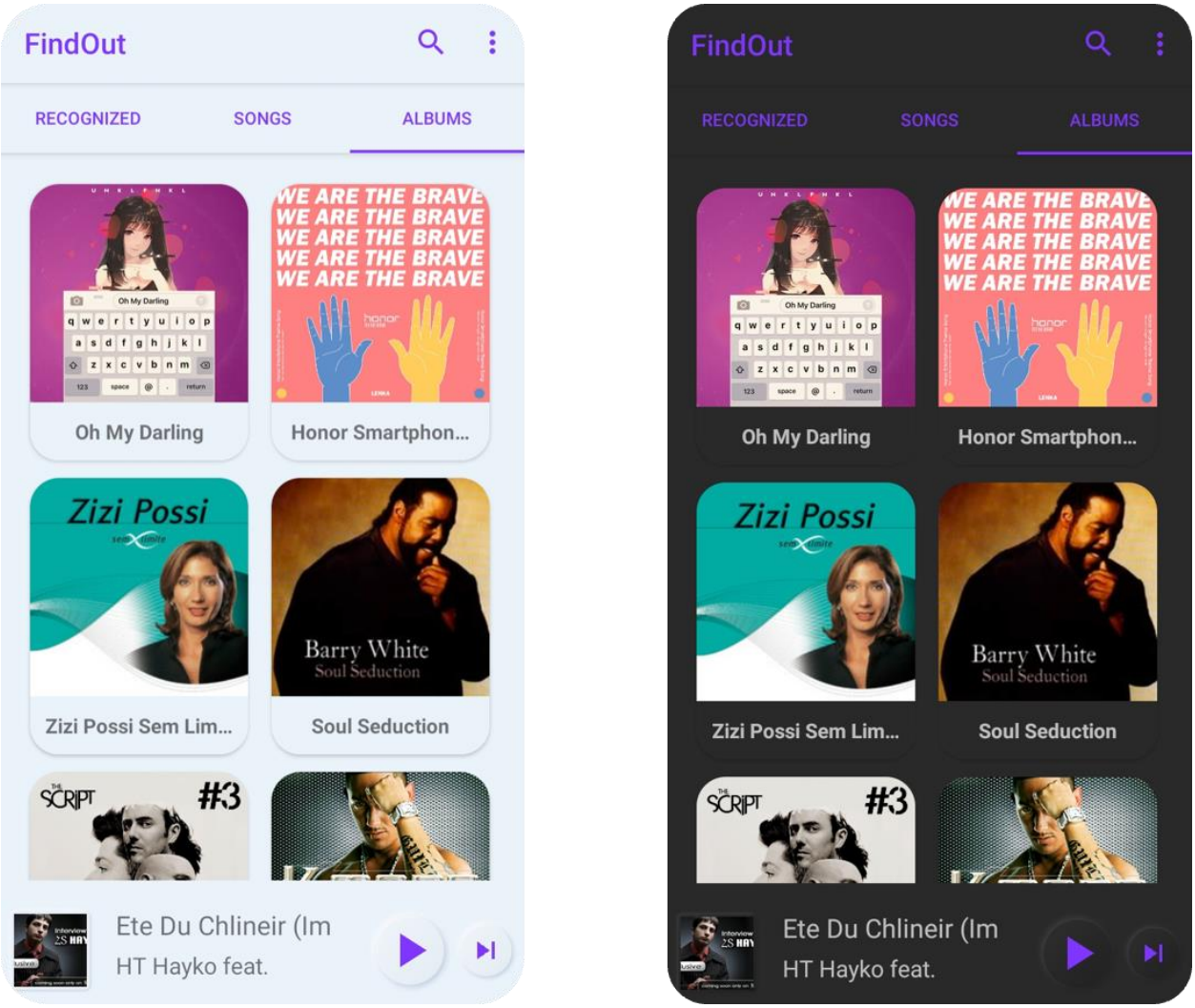


Рисунок 46 – Музыкальные альбомы

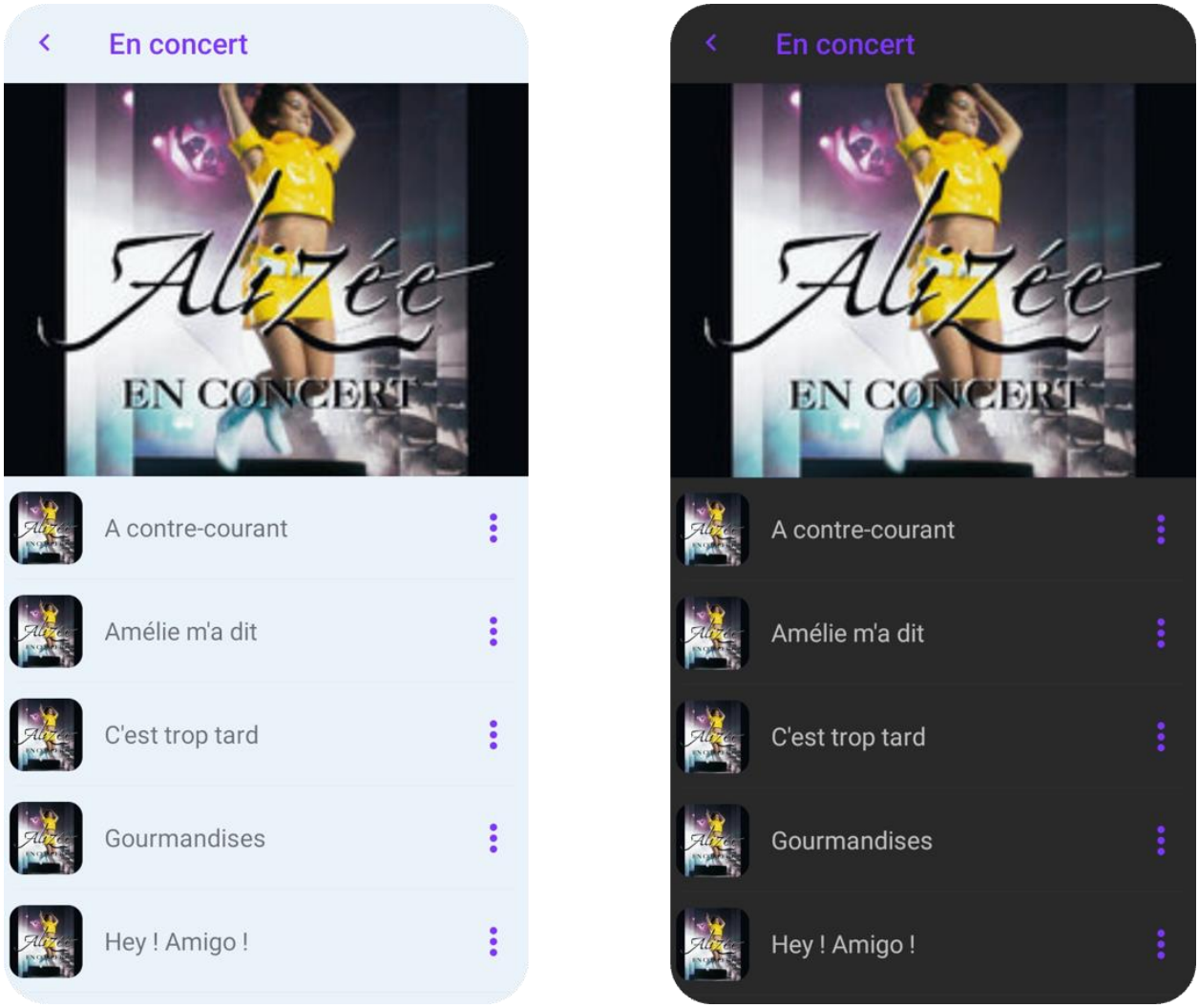


Рисунок 47 – Музыкальный альбом

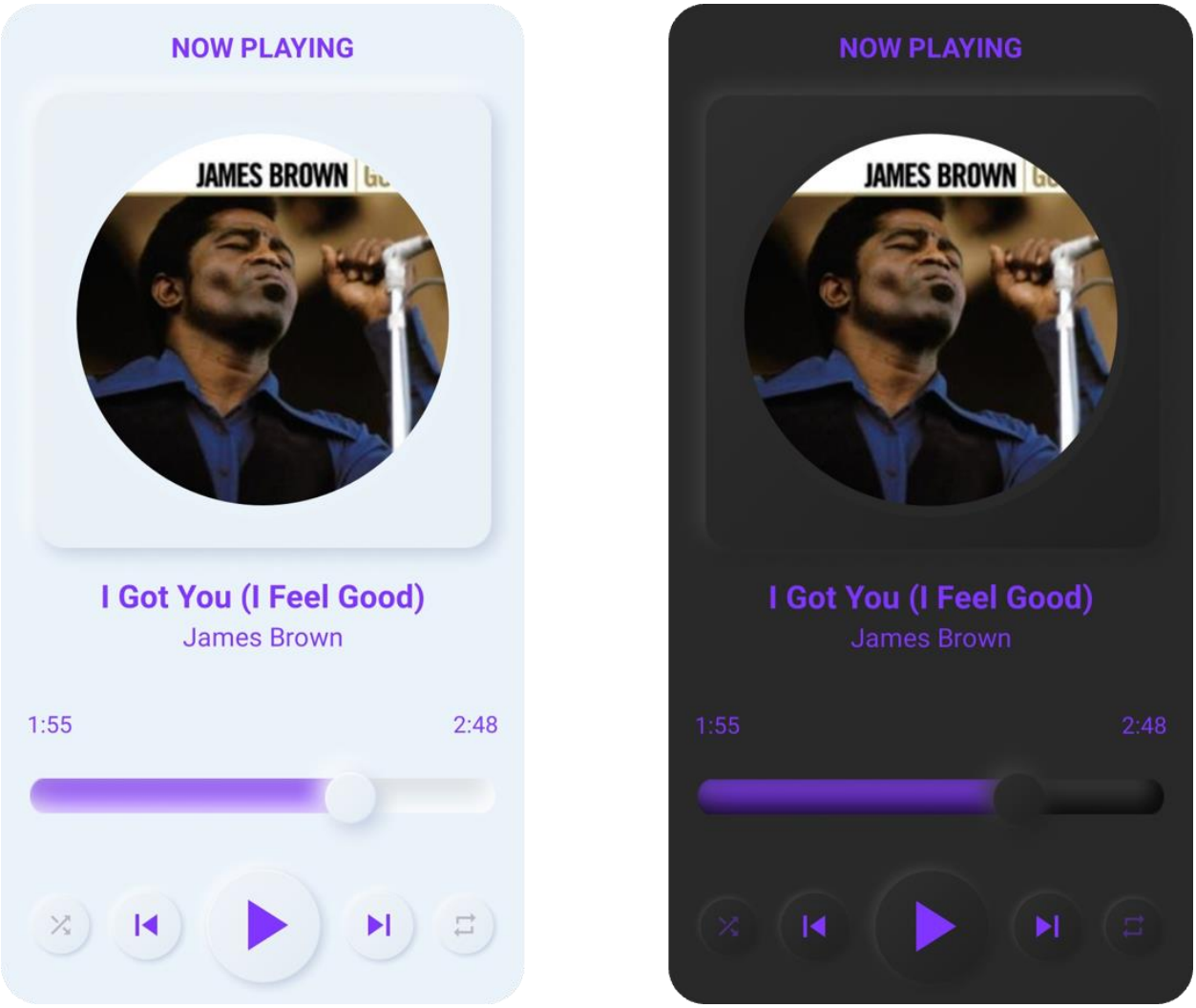


Рисунок 48 – Музыкальный плеер

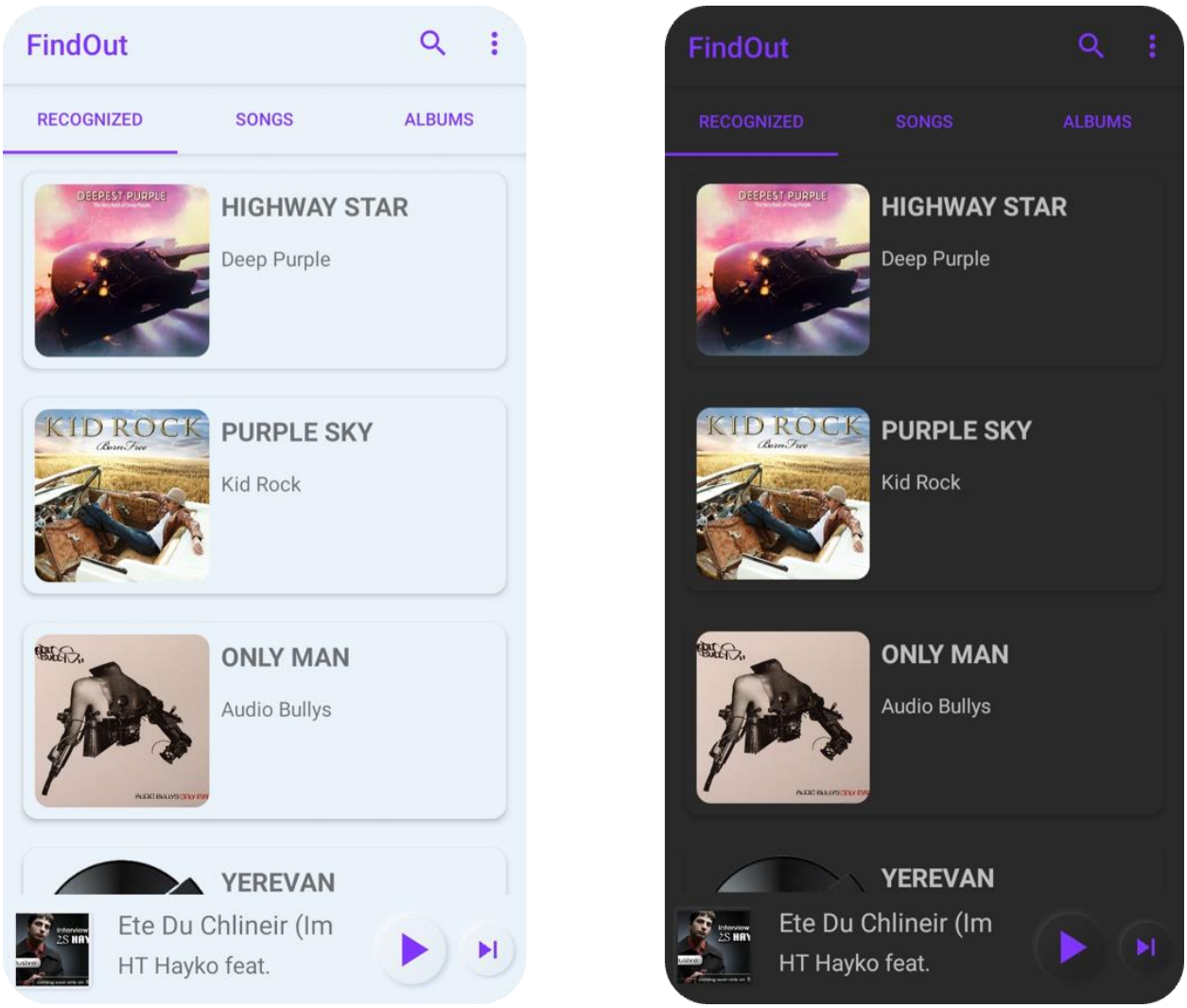


Рисунок 49 – История распознаваний

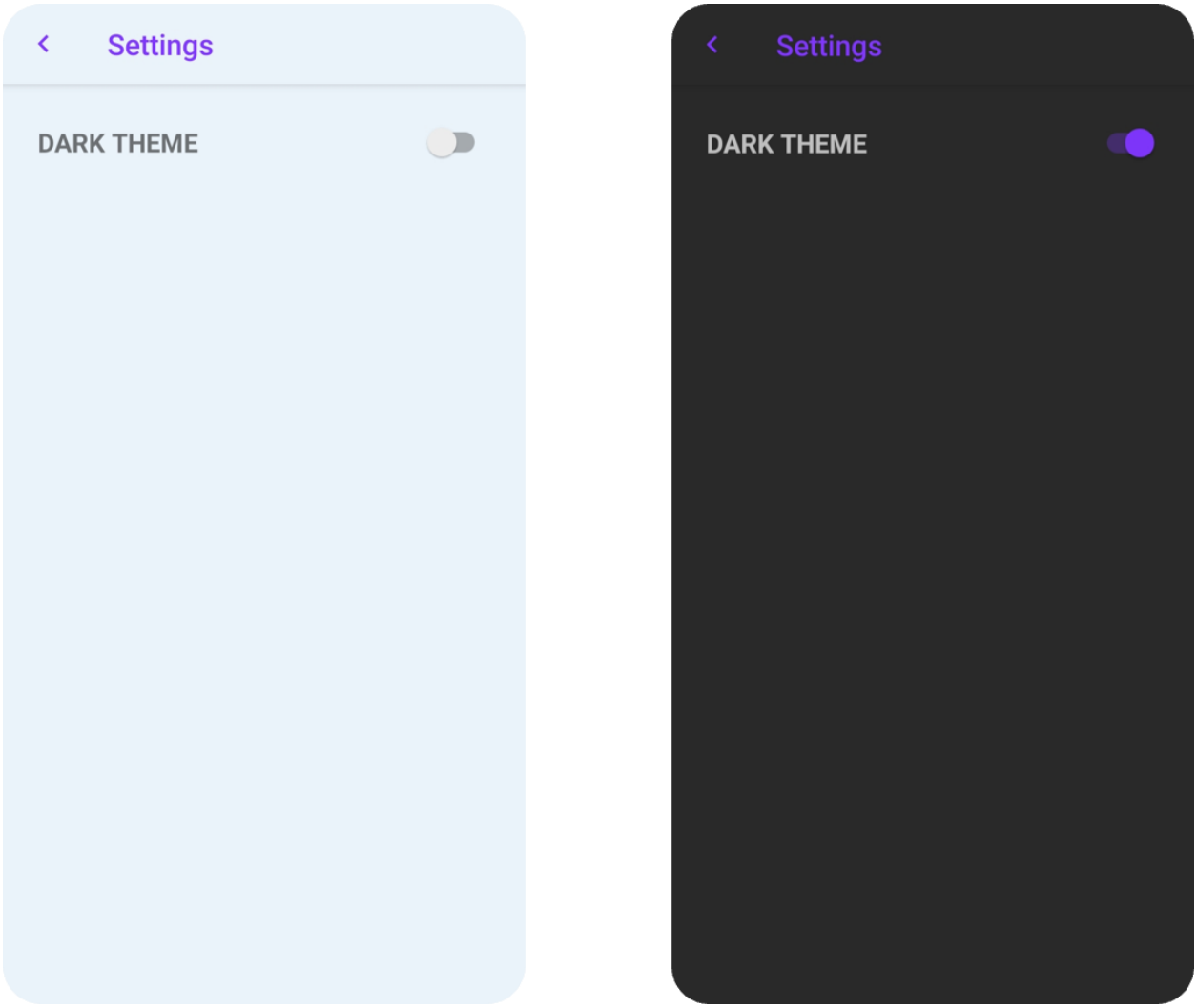


Рисунок 50 – Настройки

Протестируем распознавание в приложении, дав ему послушать песню под названием «Song 2» группы «Blur». Результат приведён на рисунке 51.

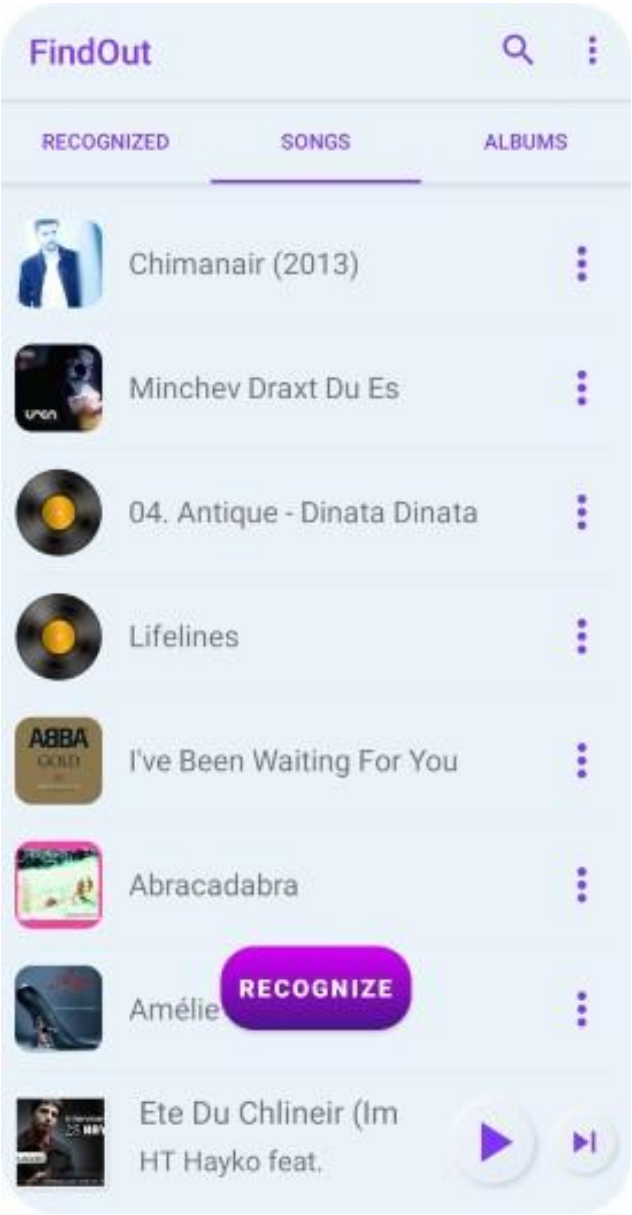


Рисунок 51 – Приложение слушает

Приложение успешно идентифицировало композицию. На рисунке 52 показан результат распознавания.

Рисунок 52 – Результат распознавания (часть 1)

Ниже на рисунке 53 показана вкладка, в которой можно выбрать способ прослушивания найденной песни.

Рисунок 53 – Вкладка выбора способа прослушивания композиции

Также можно прямо на вкладке «MUSIC» прослушать песню через YouTube, но только с видео, т. к. иначе это было бы против правил использования API YouTube.

На рисунке 54 показан результат нажатия на кнопку прослушивания музыки в VK.

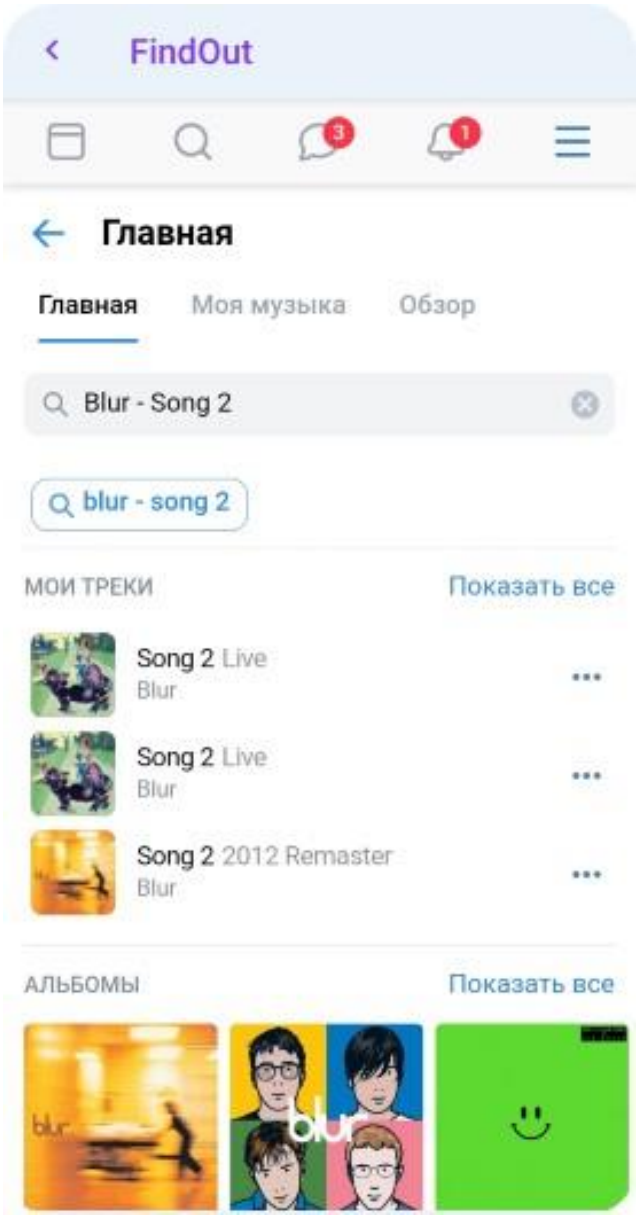


Рисунок 54 – Прослушивание музыки в ВК

Далее на рисунке 55 показан результат нажатия на кнопку перехода на YouTube (не выходя за пределы приложения), которая расположена на панели действий.

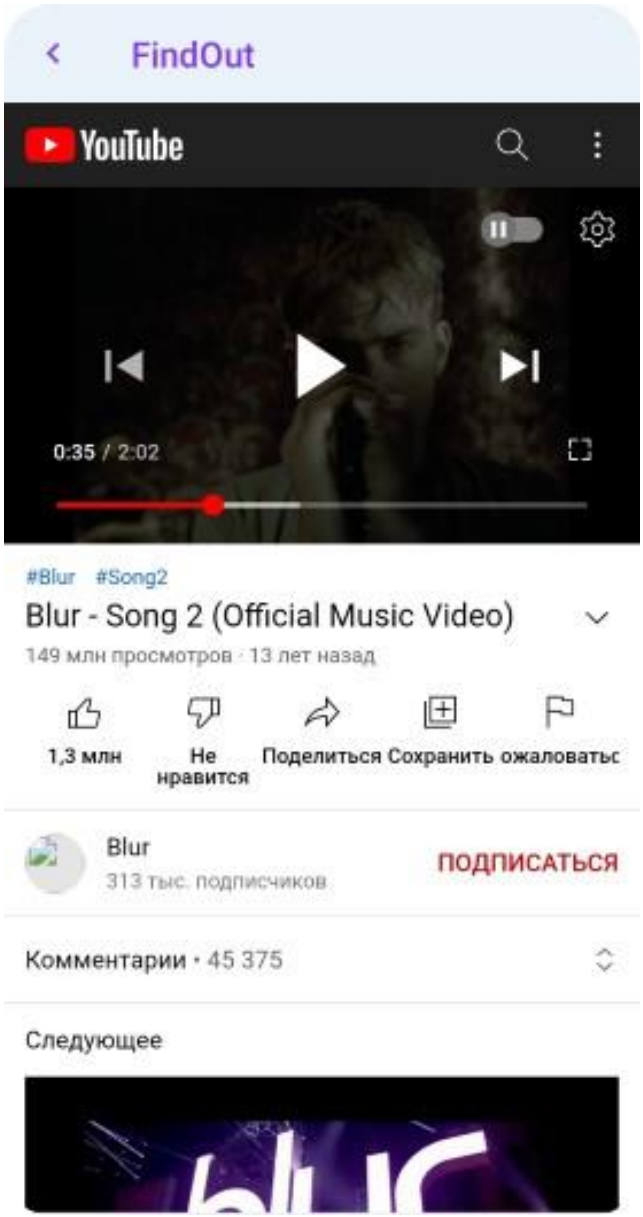


Рисунок 55 – Просмотр видеоклипа на YouTube

Далее заглянем в историю распознаваний. Как видим на рисунке 56, распознанная песня уже там.

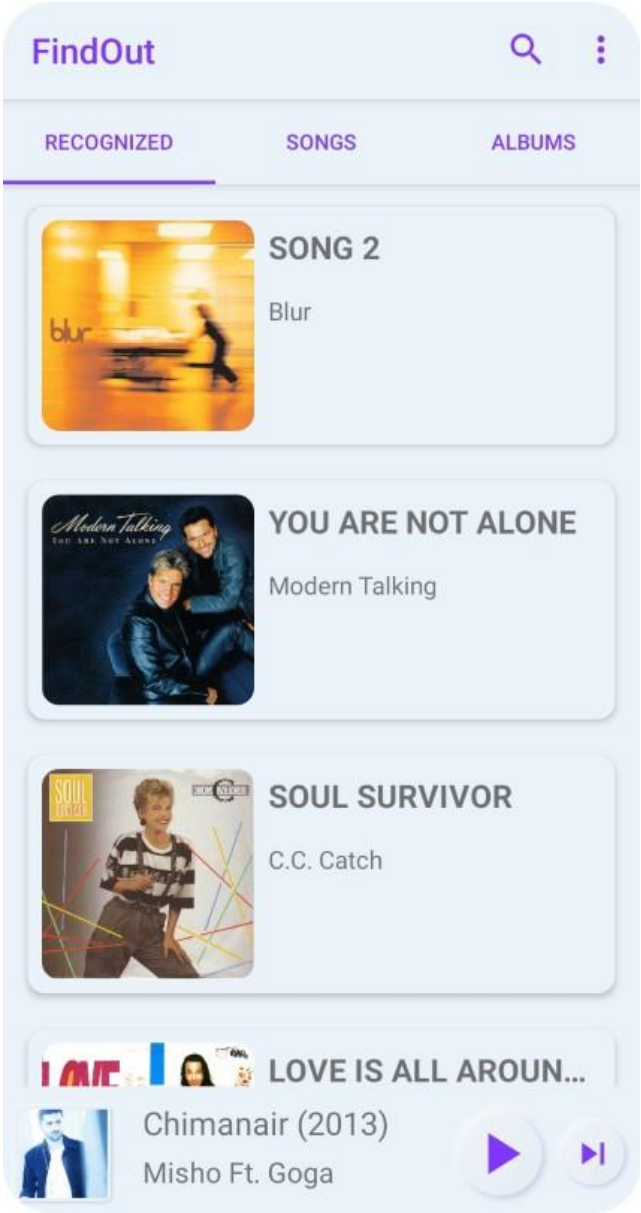


Рисунок 56 – Коллекция распознанных композиций

И наконец посмотрим, что же будет, если вдруг композиция не распознана. Результат приведён на рисунке 57.

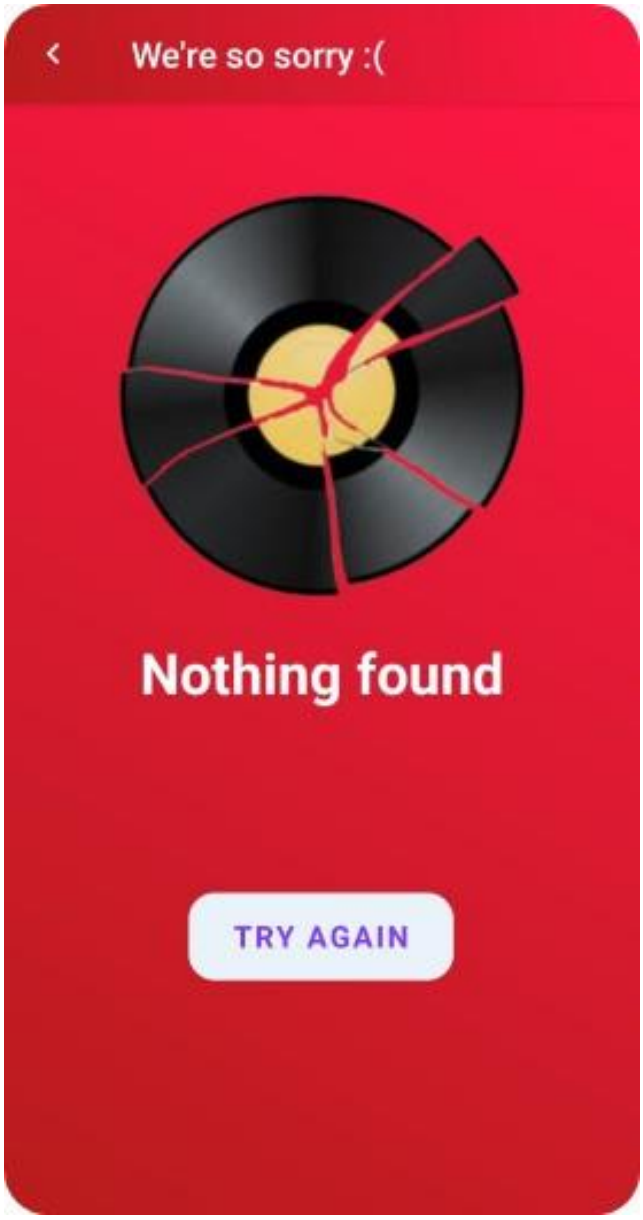


Рисунок 57 – Экран нераспознанной композиции

Также хотелось бы отметить, что приложение также распознаёт классическую и народную музыку (если конечно она есть в базе музыкальных композиций).

ЗАКЛЮЧЕНИЕ

В результате выполнения данной работы спроектирован и разработан современный музыкальный плеер с продуманным дизайном, распознающий музыку и позволяющий её бесплатно прослушать в популярных сервисах, а также дано описание процесса разработки. Все задачи, поставленные для достижения этой цели, решены, а именно:

– рассмотрены основные методы решения поставленной задачи;

– сделан обзор существующих приложений, позволяющих идентифицировать музыкальные композиции;

– для идентификации композиций был использован сторонний сервис – сервис распознавания музыки от платформы ACRCloud;

– придуман дизайн приложения, и оно было спроектировано;

– созданы необходимые для дизайна графические объекты;

– написано приложение, распознающее музыкальные композиции;

– в приложение добавлены дополнительные возможности, которые делают его более привлекательным для пользователей, а также обеспечивают функциональность, ожидаемую большинством современных пользователей (просмотр видео, прослушивание найденной музыки, отображение дополнительной информации о композиции, предоставление текста песни, сохранение истории распознаваний);

– приложение протестировано и работает хорошо.

Данная тема всё ещё актуальна несмотря на то, что подобные приложения уже существуют, и некоторые из них очень даже удачны. Но и у них есть свои недостатки.

И в конце концов, всегда можно сделать лучше, чем уже есть. Ведь совершенных программных продуктов не существует.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Шилдт Г. Java 8: руководство для начинающих, 6-е изд.: Пер. с англ. / Г. Шилдт. – М.: ООО "И. Д. Вильямс", 2015. – 720 с.

2. Гриффитс Д. Head First. Программирование для Android. 2-е изд. / Дэвид Гриффитс, Дон Гриффитс – СПб.: Питер, 2018. – 912 с.

3. Mobile Operating System Market Share Worldwide. // (Engl.). – URL: https://gs.statcounter.com/os-market-share/mobile/worldwide/ [11 June 2022].

4. Мобильные ОС в России. // (Рус.). – URL: https://radar.yandex.ru/mobile?selected_rows=F1A6ay%252CUk4F3H%252CMQuG4L%252CYhb4p1 [11 июня 2022].

5. Audio Recognition Services for Business. // (Engl.). – URL: https://www.acrcloud.com/ [11 June 2022].

6. ACRCloud Docs, Tutorials, Recognize Music. // (Engl.). – URL: https://docs.acrcloud.com/docs/acrcloud/tutorials/identify-music-by-sound/ [03 February 2020].

7. ACRCloud Docs, Demos, Android. // (Engl.). – URL: https://docs.acrcloud.com/docs/acrcloud/demos/android-demo/ [03 May 2020].

8. Как искать исполнителей понравившейся музыки: Shazam и не только. OnLime Блог // (Рус.). – URL: https://blog.onlime.ru/2018/04/02/kak_iskat_ispolniteley_ponravivsheysya_muziki_shazam_i_ne_tolko/ [01 июня 2020].

9. Animated Gradient Background in Android. // (Engl.). – URL: http://www.androidtutorialshub.com/animated-gradient-background-in-android/ [31 May 2020].

10. Dynamic iOS-like blur of underlying Views for Android. // (Engl.). – URL: https://github.com/Dimezis/BlurView [31 May 2020].

11. YouTube Player library for Android and Chromecast, stable and customizable. // (Engl.). – URL: https://github.com/PierfrancescoSoffritti/android-youtube-player [29 May 2020].

12. Тёмная тема: приёмы, которые помогут сделать качественный интерфейс // (Рус.). – URL: https://skillbox.ru/media/design/tyemnaya-tema-v-interfeise/ [13 июня 2022].

13. Неоморфизм в дизайне интерфейсов: что это такое и как правильно использовать // (Рус.). – URL: https://idbi.ru/blogs/blog/neomorfizm-v-dizayne-interfeysov [31 мая 2022].

14. Почему мы скругляем углы // (Рус.). – URL: https://ux.pub/dinozavrix/pochiemu-my-skrughliaiem-ughly-7ik [31 мая 2022].

15. Discogs API Documentation. SEARCH. // (Engl.). – URL: https://www.discogs.com/developers/#page:database,header:database-search [01 March 2020].

16. Musixmatch lyrics API Documentation. // (Engl.). – URL: https://developer.musixmatch.com/documentation [15 March 2020].

17. How to write part of a string bold in Android programmatically? // (Engl.). – URL: https://stackoverflow.com/questions/50118563/how-to-write-part-of-a-string-bold-in-android-programmatically/50118867 [28 May 2020].

18. How to play YouTube video inside Android WebView using Video Url. // (Engl.). – URL: https://inducesmile.com/android/how-to-play-youtube-video-inside-android-webview-using-video-url/#inline_content [01 April 2020].

АНТИПЛАГИАТ
ОБНАРУЖЕНИЕ ЗАИМСТВОВАНИЙ

СПРАВКА

о результатах проверки текстового документа
на наличие заимствований

Кубанский Государственный университет

ПРОВЕРКА ВЫПОЛНЕНА В СИСТЕМЕ АНТИПЛАГИАТ.ВУЗ

Автор работы: Хачатрян Арутюн Арамович
Самоцитирование рассчитано для: Хачатрян Арутюн Арамович
Название работы: РАЗРАБОТКА МУЗЫКАЛЬНОГО ПЛЕЕРА С ФУНКЦИЕЙ РАСПОЗНАВАНИЯ КОМПОЗИЦИЙ ДЛЯ ОС ANDROID
Тип работы: Магистерская диссертация
Подразделение: ФКТиПМ, кафедра прикладной математики

РЕЗУЛЬТАТЫ

■ ОТЧЕТ О ПРОВЕРКЕ КОРРЕКТИРОВАЛСЯ: НИЖЕ ПРЕДСТАВЛЕНЫ РЕЗУЛЬТАТЫ ПРОВЕРКИ ДО КОРРЕКТИРОВКИ

ЗАИМСТВОВАНИЯ	5.87%	ЗАИМСТВОВАНИЯ	5.87%
ОРИГИНАЛЬНОСТЬ	88.14%	ОРИГИНАЛЬНОСТЬ	88.14%
ЦИТИРОВАНИЯ	5.99%	ЦИТИРОВАНИЯ	5.99%
САМОЦИТИРОВАНИЯ	0%	САМОЦИТИРОВАНИЯ	0%

ДАТА ПОСЛЕДНЕЙ ПРОВЕРКИ: 14.06.2022

ДАТА И ВРЕМЯ КОРРЕКТИРОВКИ: 14.06.2022 09:05

Модули поиска: ИПС Адилет; Библиография; Сводная коллекция ЭБС; Интернет Плюс; Сводная коллекция РГБ; Цитирование; Переводные заимствования (RuEn); Переводные заимствования по eLIBRARY.RU (EnRu); Переводные заимствования по Интернету (EnRu); Переводные заимствования издательства Wiley (RuEn); eLIBRARY.RU; СПС ГАРАНТ; Модуль поиска "КубГУ"; Медицина; Диссертации НББ; Перефразирования по eLIBRARY.RU; Перефразирования по Интернету; Перефразирования по коллекции издательства Wiley; Патенты СССР, РФ, СНГ; СМИ России и СНГ; Шаблонные фразы; Кольцо вузов; Издательство Wiley; Переводные заимствования

Работу проверил: Троценко Екатерина Сергеевна
ФИО проверяющего

Дата подписи: 14.06.2022

Подпись проверяющего

Чтобы убедиться в подлинности справки, используйте QR-код, который содержит ссылку на отчет.

Ответ на вопрос, является ли обнаруженное заимствование корректным, система оставляет на усмотрение проверяющего. Предоставленная информация не подлежит использованию в коммерческих целях.

ООО «Меркури Девелопмент Самара»,
Начальник отдела QA,
Фахреев Равиль
«14» июня 2022 г.

Рецензия
на выпускную квалификационную работу
Хачатрян Арутюна Арамовича
по специальности
«Математическое и информационное обеспечение экономической деятельности»

Выпускная квалификационная работа Хачатрян А. А. выполнена на актуальную на сегодняшний день тему, поскольку разработка современного и удобного музыкального приложения востребована на рынке мобильных приложений, а также такое приложение популярно среди пользователей. Автором спроектирован и разработан современный музыкальный плеер, позволяющий распознавать музыкальные композиции и прослушивать их в популярных сервисах.

Хачатрян А. А. обработал большое количество материала, на высоком уровне проведено исследование предметной области. Материал в выпускной квалификационной работе логически структурирован, написан научным стилем изложения, чётко, по делу, и без лишних текстов.

В первой главе выпускной квалификационной работы автором грамотно обоснована актуальность темы работы.

Во второй главе описаны способы разработки необходимого функционала.

В третьей главе представлен обзор существующих решений в выбранной области приложений.

В четвёртой и пятой главах грамотно описаны структура приложения и процесс его разработки.

В шестой главе продемонстрировано тестирование разработанного мобильного приложения.

Автор выпускной квалификационной работы показал отличную способность формулировать собственную точку зрения по рассматриваемой проблеме. Сформулированные в работе выводы достаточно обоснованы и могут быть использованы в практической деятельности.

В ходе подготовки и защиты ВКР Хачатрян А. А. показал высокий уровень сформированности необходимых компетенций.

ВКР не содержит существенных недостатков.

Выпускная квалификационная работа соответствует требованиям, предъявляемым к выпускным квалификационным работам, и может быть рекомендована к защите на заседании государственной аттестационной комиссии. Работа заслуживает отличной оценки.

Начальник отдела QA, Фахреев Равиль _____

КубГУ, ФКТиПМ,
зав. кафедры КАДИИ,
доктор тех. наук
Коваленко Анна Владимировна
«14» июня 2022 г.

Рецензия
на выпускную квалификационную работу
Хачатрян Арутюна Арамовича
по специальности
«Математическое и информационное обеспечение экономической деятельности»

Выпускная квалификационная работа Хачатрян А. А. выполнена на актуальную на сегодняшний день тему, поскольку разработка современного и удобного музыкального приложения востребована на рынке мобильных приложений, а также такое приложение популярно среди пользователей. Автором спроектирован и разработан современный музыкальный плеер, позволяющий распознавать музыкальные композиции и прослушивать их в популярных сервисах.

Хачатрян А. А. обработал большое количество материала, на высоком уровне проведено исследование предметной области. Материал в выпускной квалификационной работе логически структурирован, написан научным стилем изложения, чётко, по делу, и без лишних текстов.

В первой главе выпускной квалификационной работы автором грамотно обоснована актуальность темы работы.

Во второй главе описаны способы разработки необходимого функционала.

В третьей главе представлен обзор существующих решений в выбранной области приложений.

В четвёртой и пятой главах грамотно описаны структура приложения и процесс его разработки.

В шестой главе продемонстрировано тестирование разработанного мобильного приложения.

Автор выпускной квалификационной работы показал отличную способность формулировать собственную точку зрения по рассматриваемой проблеме. Сформулированные в работе выводы достаточно обоснованы и могут быть использованы в практической деятельности.

В ходе подготовки и защиты ВКР Хачатрян А. А. показал высокий уровень сформированности необходимых компетенций.

ВКР не содержит существенных недостатков.

Выпускная квалификационная работа соответствует требованиям, предъявляемым к выпускным квалификационным работам, и может быть рекомендована к защите на заседании государственной аттестационной комиссии. Работа заслуживает отличной оценки.

Зав. кафедры КАДИИ, доктор тех. наук Коваленко А. В. _____

Отзыв
на выпускную квалификационную работу магистра Хачатряна А.А.
направление 01.04.02 Прикладная математика и информатика
тема «Разработка музыкального плеера с функцией распознавания композиций для ОС Android»

Тема выпускной квалификационной работы актуальна для большинства людей, имеющих смартфоны и нечуждых музыке, поскольку автоматизация идентификации музыкальных произведений при помощи мобильного приложения позволяет существенно упростить и ускорить этот процесс.

Содержание работы полностью соответствует цели работы и поставленным задачам. В первой главе обосновывается актуальность темы. Во второй проводится анализ предметной области, рассматриваются способы разработки приложения для идентификации музыкальных композиций. В третьей – проводится сравнительный анализ существующих решений. Видно, что автор изучил и проанализировал значительный объем литературы и Интернет-источников. Четвертая глава посвящена вопросам проектирования приложения, обоснован выбор инструментальных средств разработки, описаны структура и дизайн приложения. В пятой главе описана программная реализация (этап кодирования) и база данных. Шестая глава посвящена тестированию приложения.

Положительной чертой работы является квалифицированность и обоснованность принятых автором решений, а также ее итог – мобильное приложение для ОС Android, предназначенное для идентификации музыкальных композиций с удобным интерфейсом, потенциально имеющее важное прикладное значение для любителей музыки.

Работа выполнена студентом самостоятельно, на высоком профессиональном уровне. Все поставленные задачи решены в полном объеме. Результаты работы изложены технически грамотно, оформлены в соответствии с требованиями ГОСТ.

Оцениваю выпускную квалификационную работу Хачатряна А.А. на «отлично».

Научный руководитель,
канд. физ.-мат. наук,
доцент кафедры прикладной математики А.В. Письменский