



**Содержание**

[Введение 4](#_Toc74002613)

[1 Анализ задания и описание предметной области 5](#_Toc74002614)

[1.1 Постановка задач 5](#_Toc74002615)

[1.2 Функциональные требования 5](#_Toc74002616)

[1.3 Выбор средств реализации 6](#_Toc74002617)

[2 Проектирование приложения 7](#_Toc74002618)

[2.1 Проектирование интерфейса 7](#_Toc74002619)

[2.2 Проектирование классов 8](#_Toc74002620)

[2.3 Проектирование базы данных 10](#_Toc74002621)

[3 Реализация приложения 13](#_Toc74002622)

[3.1 Спецификация реализованных классов 13](#_Toc74002623)

[4 Тестирование 15](#_Toc74002624)

[4.1 Методика тестирования 15](#_Toc74002625)

[4.2 Разработанные тесты 15](#_Toc74002626)

[4.3 Результаты тестирования 19](#_Toc74002627)

[Заключение 39](#_Toc74002628)

[Список использованных источников 40](#_Toc74002629)

# Введение

Целью курсового проекта является закрепление теоретических знаний полученных в процессе изучения курса «Объектно-ориентированное программирование» и развитие практических навыков применения объектного подхода в ходе решения конкретной практической задачи – разработки приложения с использованием объектно-ориентированного подхода, а также развитие способностей к обоснованному принятию самостоятельных решений в ходе проектной деятельности и получение базового опыта разработки программного обеспечения.

В рамках курсового проектирования решается задача создания Android-приложения «EarFeeder» - музыкального плеера с возможностью создания и корректирования списков воспроизведения и добавления пользовательских интернет-радиоканалов. Вся информация о музыкальных композициях должна браться из памяти телефона и храниться в локальной базе данных.

1. Анализ задания и описание предметной области

## Постановка задач

В ходе разработки необходимо решить следующие задачи:

1. Выработать функциональные требования к приложению
2. Выполнить проектирование интерфейса
3. Осуществить проектирование классов
4. Произвести проектирование базы данных с помощью DAO классов
5. Разработать приложение, основываясь на результатах проделанной работы вышеперечисленных пунктов
6. Определить методику тестирования и проверить работу приложения
7. Резюмировать результаты курсового проектирования
   1. Функциональные требования

Приложение должно выполнять функции музыкального проигрывателя:

1. Работа с музыкальными треками.
   1. Сбор всех музыкальных файлов с устройства в один список треков.
   2. Возможность выбора трека для воспроизведения.
   3. Возможность прослушивания композиций в фоновом режиме.
   4. Возможность управлять (ставить на паузу, продолжать воспроизведение, переключать вперед и назад) композициями.
   5. Возможность добавлять композиции в «избранное» и отдельно прослушивать их.
   6. Возможность управления композициями путем перемотки, перемешивания и зацикливания.
2. Работа с интернет-радиоканалами.
   1. Просмотр списка сохраненных радиоканалов.
   2. Добавление нового интернет-радиоканала в список по ссылке.
   3. Удаление радио из списка.
   4. Выбор и прослушивание радио из списка.
3. Работа с очередями композиций.
   1. Создание плейлистов.
   2. Редактирование плейлистов.
   3. Удаление плейлистов.
   4. Перемешивание плейлистов.
   5. Сохранение плейлистов в локальную базу данных.
4. Пользовательские настройки приложения.
   1. Возможность выбора цветовой схемы (красный, зеленый, голубой).
   2. Возможность сбросить все сохраненные материалы в приложении (музыкальные композиции с устройства останутся на месте, в списке).
   3. Выбор средств реализации

В качестве минимального уровня Android SDK был выбран 21 уровень API, что позволит обеспечить возможность использования приложения примерно на 94.1% всех мобильных устройств.

Для реализации Android-приложения были выбраны следующие программные инструменты и библиотеки:

* + - 1. Android studio – IDE для разработки приложения под ОС Android
      2. Room [11]– ORM библиотека для создания и управления базы данных на основе SQLite и с помощью DAO классов
      3. Gauvark Audio visualizer [10]– библиотека для визуализации музыкальных композиций в виде всплывающих столбцов

# Проектирование приложения

* 1. Проектирование интерфейса

Требования к интерфейсу:

1. Возможность управлять композициями и интернет-радиоканалами внутри приложения и через меню уведомлений.
2. Возврат в приложение при нажатии на меню уведомлений.
3. Удобный переход между экранами через боковое меню навигации.
4. Общий нижний фрагмент приложения для управления композициями.
5. Возможность переходить назад по экранам без специальных функций телефона.

На основе анализа функциональных требований были выделены следующие экраны приложения (см. рис. 1):

1. Приветственный экран – всплывает на 2 секунды после включения приложения (если запуск первый или приложение долго не запускалось).
2. Главный экран выбора треков – на нем находится список всех найденных треков на устройстве, композиции можно запустить и прослушать. Так же отсюда можно попасть в любой экран приложения.
3. Экран выбора радио – позволяет запустить и прослушать интернет-радиоканалы, уже хранящиеся в приложении или добавленные пользователем.
4. Экран выбора плейлистов – позволяет просмотреть или создать плейлисты.
5. Экран создания плейлиста – инструмент для создания плейлистов из всех композиций на устройстве.
6. Экран просмотра плейлиста – показывает все композиции из выбранного плейлиста.
7. Экран редактирования плейлиста – возможность добавить в существующий плейлист другие композиции из оставшихся треков на устройстве.
8. Экран выбора любимых песен – отображает композиции, которые были отмечены пользователем, как «любимые».
9. Экран настроек приложения – позволяет выбрать цветовую схему приложения из предложенных трех (красный, зеленый, голубой) и очистить все данные, кроме списка песен с устройства.
10. Детальное окно играющего трека – возможность расширенного управлению текущей композицией: перемотка, перемешивание, повтор, переключение.

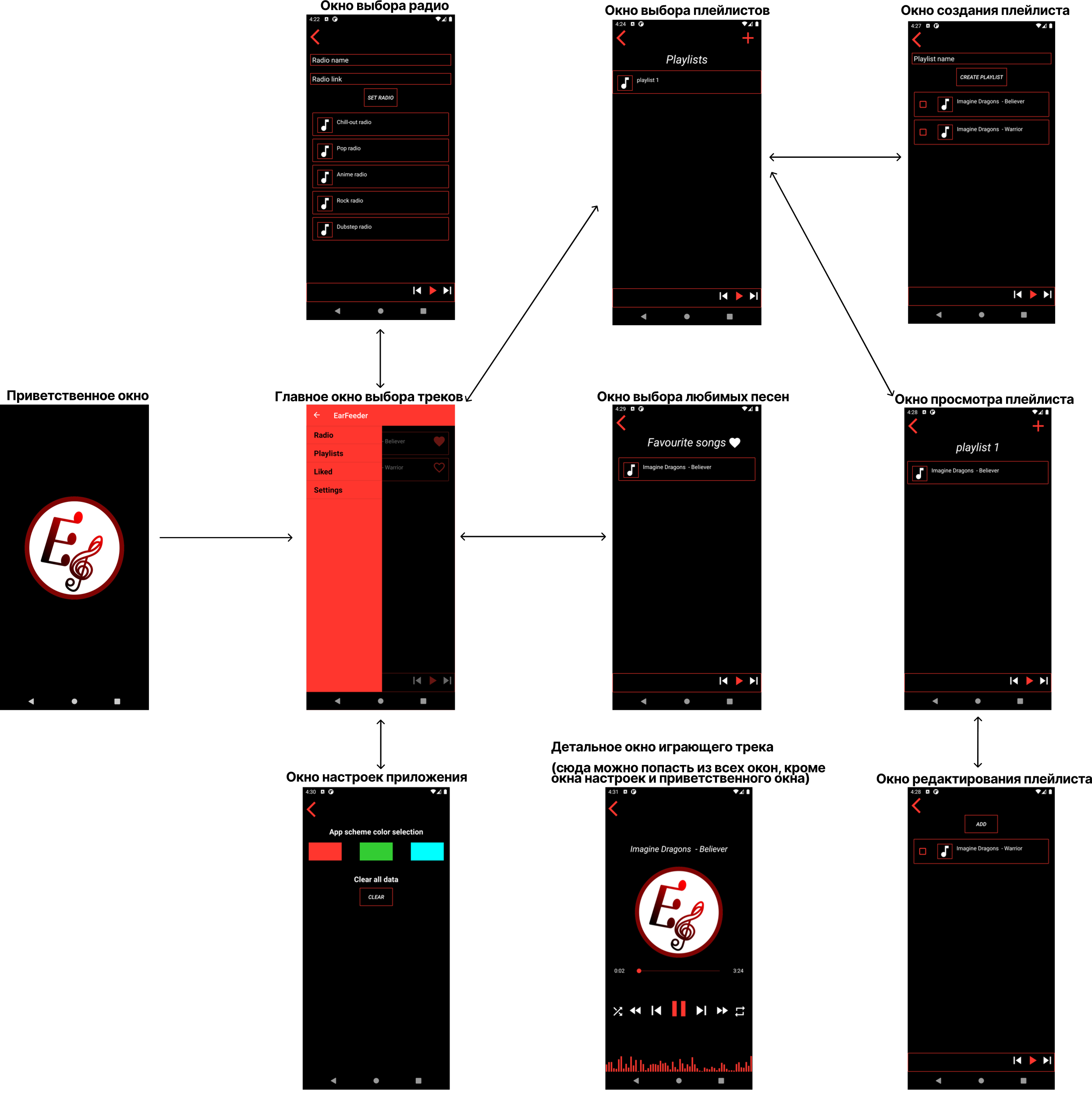


Рисунок 1 – Схема переходов между экранами приложения

* 1. Проектирование классов

Для реализации работы приложения с базой данных были определены следующие классы-сущности:

1. Track – музыкальная композиция, взятая из хранилища пользователя.
2. Radio – интернет-радиоканал
3. Playlist – список музыкальных композиций, собранных лично пользователем
4. TrackPlaylist – принадлежность музыкальных композиций к плейлистам

Были выделены следующие группы классов (см. рис. 2):

1. Database – на основе принадлежности к базе данных (сущности, запросы и абстрактный класс, соединяющий все классы-запросы)
2. Service – на основе наследование от класса Service
3. Player – отдельный класс для управления музыкальным плеером
4. App – запускающий класс, содержащий все важные переменные и методы, позволяющие приложению функционировать без лишних переменных, храня единственный экземпляр нужных значений

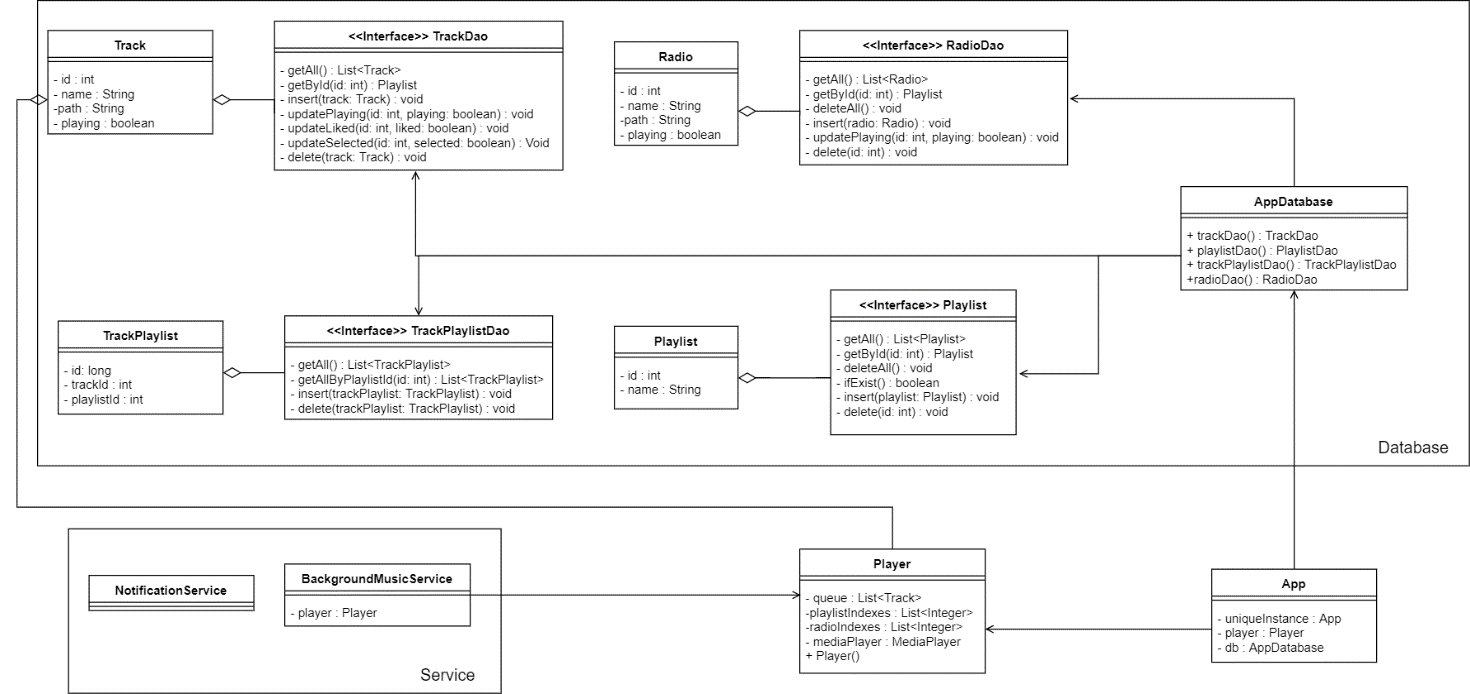


Рисунок 2 – Диаграмма классов UML, первый фрагмент

Для взаимодействия пользователя с приложением были определены следующие группы классов (см. рис. 3):

1. Activity – группа экранов приложения
2. Playable – интерфейс, реализующий взаимодействие между плеером и меню уведомлений

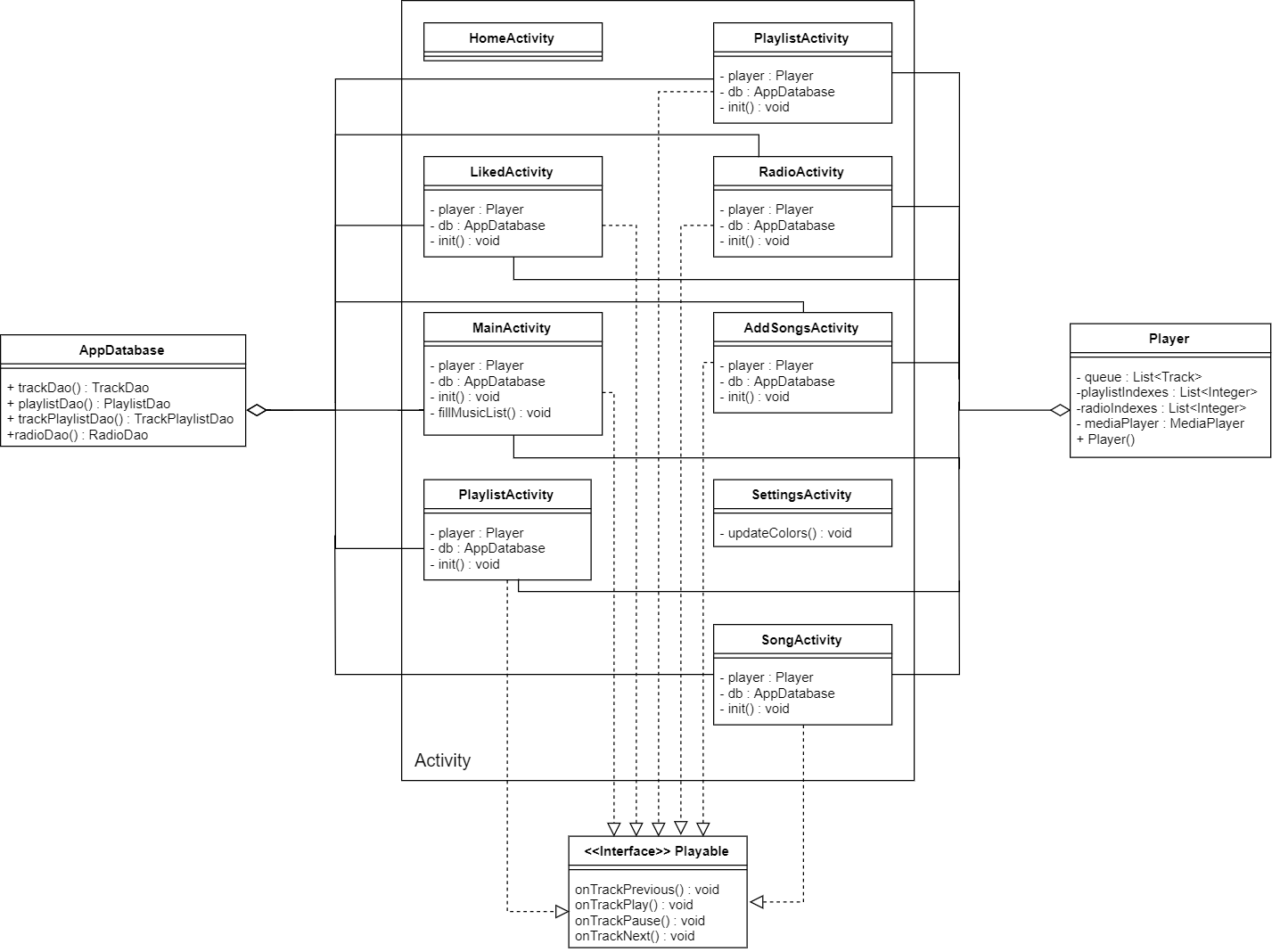


Рисунок 3 – Диаграмма классов UML, второй фрагмент

* 1. Проектирование базы данных

Для работы с базой данных на основе упомянутых сущностей в проектировании классов (см. рис. 2) были созданы следующие таблицы (см. рис. 4).

Первичным ключом во всех четырех таблицах является их уникальный идентификатор id, по которому таблицы Track, Playlist и TrackPlaylist могут «общаться» друг с другом.

В таблице TrackPlaylist хранятся id треков и плейлистов, к которым они принадлежат. В то время, как таблица Radio существует сама по себе, хранит только записи о интернет-радиоканалах и не взаимодействует ни с какой другой таблицей.

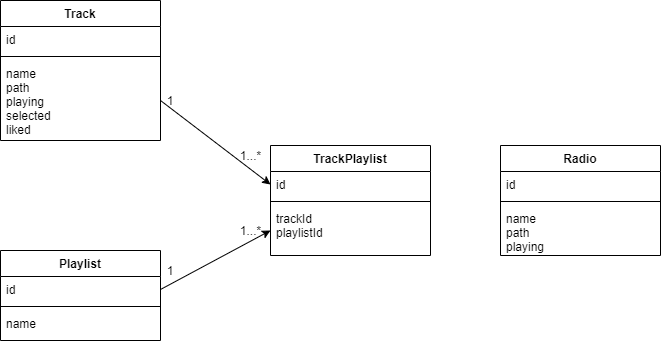


Рисунок 4 - Диаграмма классов базы данных

Таблица 1 – Описание таблиц БД

|  |  |  |  |
| --- | --- | --- | --- |
| Имя таблицы | Имя атрибута | Тип | Описание |
| Track | id | int | Идентификатор трека |
| name | string | Название трека |
| path | string | Путь до трека |
| playing | boolean | Играет ли трек сейчас |
| selected | boolean | Выбран ли он в меню выбора треков |
| liked | boolean | Находится ли трек в «любимых» |
| Playlist | id | int | Идентификатор плейлиста |
| name | string | Название плейлиста |
| TrackPlaylist | id | long | Идентификатор записи |
| trackId | int | Идентификатор трека |
| playlistId | int | Идентификатор плейлиста, которому принадлежит трек |
| Radio | id | int | Идентификатор радио |
| name | string | Название радио |
| path | string | Ссылка на радио |
| playing | boolean | Играет ли радио сейчас |

Реализованные SQL-запросы к базе данных в DAO классах:

TrackDao:

1. SELECT \* FROM Track – возвращает все треки
2. SELECT \* FROM Track WHERE id = :id – возвращает трек по его идентификационному номеру
3. UPDATE Track SET playing = :playing WHERE id = :id – обновляет статус «играет» у трека в базе данных
4. UPDATE Track SET liked = :liked WHERE id = :id – обновляет статус «понравившийся» у трека в базе данных
5. UPDATE Track SET selected = :selected WHERE id = :id – обновляет статус «выбран» у трека в базе данных
6. INSERT Track – вставляет трек в базу данных
7. DELETE Track – удаляет трек из базы данных

PlaylistDao:

1. SELECT \* FROM Playlist – возвращает все плейлисты
2. SELECT \* FROM Playlist WHERE id = :id – возвращает плейлист по его идентификационному номеру
3. DELETE FROM Playlist – удаляет все плейлисты в базе данных
4. SELECT EXISTS(SELECT \* FROM Playlist) – проверяет, существует ли хоть один плейлист
5. DELETE FROM Playlist WHERE id LIKE :id – удаляет плейлист по его идентификационному номеру из базы данных
6. INSERT Playlist – вставляет плейлист в базу данных

TrackPlaylistDao:

1. SELECT \* FROM TrackPlaylist - возвращает все записи
2. SELECT \* FROM TrackPlaylist WHERE playlistId = :id - возвращает все записи, где идентификатор плейлиста равен заданному
3. DELETE FROM TrackPlaylist – удаляет все записи TrackPlaylist из базы данных
4. INSERT TrackPlaylist – вставляет запись в базу данных
5. DELETE TrackPlaylist – удаляет запись из базы данных

RadioDao:

1. SELECT \* FROM Radio - возвращает все интернет-радиоканалы
2. SELECT \* FROM Radio WHERE id = :id - возвращает интернет-радиоканал по его идентификационному номеру
3. DELETE FROM Radio – удаляет все интернет-радиоканалы из базы данных
4. UPDATE Radio SET playing = :playing WHERE id = :id – обновляет статус «играет» у интернет-радиоканала в базе данных
5. DELETE FROM Radio WHERE id LIKE :id – удаляет интернет-радиоканал по его идентификационному номеру из базы данных
6. INSERT Radio – вставляет интернет-радиоканал в базу данных

# Реализация приложения

* 1. Спецификация реализованных классов

Пакет Activity:

1. HomeActivity – приветственный экран с логотипом приложения
2. LikedActivity –экран с любимыми песнями пользователя
3. MainActivity – главный экран со всеми треками с устройства
4. PlaylistActivity –экран с просмотром и возможностью добавления плейлистов
5. PlaylistViewActivity – экран с просмотром песен в выбранном плейлисте
6. RadioActivity – экран с возможностью прослушивания и добавления радио
7. AddSongsActivity – экран с возможность добавления песен в выбранный плейлист
8. CreatingPlaylistActivity – экран с созданием плейлиста
9. SettingsActivity – экран с настройками приложения
10. SongActivity – экран текущей песни с расширенными возможностями управления

Пакет Adapter:

1. LikeTrackAdapter – редактирование показа любимых песен в списке
2. PlaylistAdapter – редактирование показа плейлистов в списке
3. TrackAdapter - редактирование показа песен в списке
4. RadioAdapter - редактирование показа радио в списке
5. TrackAdapterSelect – редактирования списка песен с возможностью выбора (checkbox)

ORM (Object-Relational Mapping) – техника, создающая виртуальную объектную БД и добавляющая дополнительный слой абстракции на работу с ней [7].

DAO (Data Access Object) - интерфейс, отвечающий за передачу запросов в БД и обработку полученных от нее ответов [8].

Пакет Database:

1. Playlist – сущность «плейлист»
2. PlaylistDao – запросы к сущности «плейлист»
3. Radio – сущность «радио»
4. RadioDao – запросы к сущности «радио»
5. Track – сущность «трек»
6. TrackDao – запросы к сущности «трек»
7. TrackPlaylist – таблица связей «треков» и «плейлистов»
8. AppDatabase – набор абстрактных ссылок на классы запросов

Пакет Service:

1. BackgroundMusicService – сервис, который организует воспроизведение музыки
2. NotificationService – сервис, который позволяет отображать уведомления
3. NotificationActionService – сервис, который организует отправку действий, которые происходят на уведомлении

Пакет Notification:

1. CreateNotification – создание уведомления
2. Playable – интерфейс, позволяющий управлять уведомлением

Пакет Fragment:

1. BottomFragment – нижняя панель приложения для управления треками
2. ClearAllFragment – кнопка очистки данных приложения
3. ColorSelectionFragment – панель управления цветовой схемой приложения

Singleton – паттерн, позволяющий иметь лишь единственный экземпляр некоторого класса через предоставление глобального доступа к этому экземпляру [9].

LoadingDialog – управление панелью загрузки

RecyclerItemListener – переопределение событий нажатия на RecyclerView

AppColor – организация управления цветовой схемой приложения

Player – все переменные и методы, нужные для воспроизведения музыки

App – запускающий класс для управления и хранения главных состояний приложения, основан на паттерне Singleton

# Тестирование

## Методика тестирования

В качестве метода тестирования было выбрано приемочное тестирование.

Целью данного метода является определение готовности продукта, что достигается путем прохода тестовых сценариев и случаев, которые построены на основе спецификации требований к разрабатываемому ПО.

## Разработанные тесты

Таблица 2 – Тесты в главном меню MainActivity

|  |  |  |
| --- | --- | --- |
| № | Совершенное действие | Ожидаемый результат |
| 1 | Запуск трека в главном меню. | Начинает играть нажатая песня, появляется знак того, что эта песня играет, на меню снизу и меню уведомлений в выпадающем верхнем меню телефона знак «запустить» поменяется на знак «пауза». |
| 2 | Нажатие на кнопку «нравится» в форме сердца на песне. | Если песня уже была в «понравившихся», то сердце кнопка станет без заполнения, иначе – заполнится выбранным цветом. Трек добавится в меню «понравившиеся». |
| 3 | Нажатие на кнопку «следующий» на нижней панели приложения. | Песня переключится на следующую в списке, название на панели снизу поменяется. Поменяется иконка играющего трека. |
| 4 | Нажатие на кнопку «следующий» на нижней панели приложения, когда дальше по списку нет песен. | Ничего не произойдет. |
| 5 | Нажатие на кнопку «предыдущий» на нижней панели приложения. | Песня переключится на предыдущую в списке, название на панели снизу поменяется. Поменяется иконка играющего трека. |
| 6 | Нажатие на кнопку «предыдущий» на нижней панели приложения, когда перед текущим треком нет ничего. | Ничего не произойдет. |
| 7 | Нажатие на кнопку «пуск» / «пауза» на нижней панели приложения. | Песня начнет играть/остановится и поменяется изображение на кнопке на нижней панели приложения. |

Таблица 2 - Продолжение

|  |  |  |
| --- | --- | --- |
| 8 | Нажатие на кнопку «пуск» / «пауза», «следующий» или «предыдущий» на нижней панели приложения, когда в начале работы приложения еще не запущен никакой трек. | Ничего не произойдет. |
| 9 | Пункт 3, но нажатие происходит на меню уведомлений. | См пункт 3. |
| 10 | Пункт 4, но нажатие происходит на меню уведомлений. | См пункт 4. |
| 11 | Пункт 5, но нажатие происходит на меню уведомлений. | См пункт 5. |
| 12 | Пункт 6, но нажатие происходит на меню уведомлений. | См пункт 6. |
| 13 | Пункт 7, но нажатие происходит на меню уведомлений. | См пункт 7. |
| 14 | Пункт 8, но нажатие происходит на меню уведомлений. | См пункт 8. |
| 15 | Нажатие на название песни на нижней панели приложения. | Открытие activity с расширенным функционалом управления текущей песни. |
| 16 | Нажатие на название песни на нижней панели приложения, когда в начале работы приложения еще не запущен никакой трек или играет радио. | Ничего не произойдет. |
| 17 | Нажатие на названия других activity в боковом выдвигающемся меню. | Будет произведет переход на соответствующие activity. |

Таблица 3 – Тесты на SongActivity

|  |  |  |
| --- | --- | --- |
| № | Совершенное действие | Ожидаемый результат |
| 18 | Перемещение тайм-кода песни (Seekbar). | Песня окажется на тайм-коде, куда было произведено нажатие. |
| 19 | Перемещение тайм-кода песни во время паузы. | Песня начнет играть с установленного тайм-кода. |
| 20 | Нажатие кнопки «назад». | Действие вернется на главное меню. |
| 21 | Нажатие кнопки «перемотать назад». | Вернет песню на 10 секунд назад. |
| 22 | Нажатие кнопки «перемотать назад», когда от начала песни прошло меньше 10 секунд. | Песня вернется на нулевую секунду. |

Таблица 3 - Продолжение

|  |  |  |
| --- | --- | --- |
| 23 | Нажатие кнопки «перемотать вперед». | Перемотает песню на 10 секунд вперед. |
| 24 | Нажатие кнопки «перемотать вперед», когда до конца песни осталось меньше 10 секунд. | Начнет играть следующая песня. Если дальше нет песен, то воспроизведение окончится. |
| 25 | Нажатие кнопки «предыдущий». | Начнется играть предыдущая песня по списку. |
| 26 | Нажатие кнопки «предыдущий», когда нет предшествующих песен. | Текущая песня возобновится. |
| 27 | Нажатие кнопки «следующий». | Начнется играть следующая песня по списку. |
| 28 | Нажатие кнопки «следующий», когда нет последующих песен. | Воспроизведение окончится. |
| 29 | Нажатие кнопки «повторить». | Если еще не была нажата – песня будет играть на повторе, иначе – песни будут играть по порядку списка. Изменится значок (не закрашенный / закрашенный). |
| 30 | Нажатие кнопки «перемешать». | Песни перемешаются в случайном порядке, приоритет отдается любимым песням. Сразу начнется воспроизведение с первой песни из списка перемешанных. Изменится значок (не закрашенный / закрашенный). |

Таблица 4 – Тесты на Settings

|  |  |  |
| --- | --- | --- |
| № | Совершенное действие | Ожидаемый результат |
| 31 | Нажатие на кнопки цветов. | Переключение цветовой схемы в цвет кнопки. |
| 32 | Нажатие на кнопку «очистить». | Очистятся все плейлисты, любимые песни и радио. |

Таблица 5 – Тесты на LikedActivity

|  |  |  |
| --- | --- | --- |
| № | Совершенное действие | Ожидаемый результат |
| 33 | Нажатие на трек. | См пункт 1. |
| 34 | Нажатие на кнопку «назад». | См пункт 20. |

Таблица 6 – Тесты на PlaylistActivity

|  |  |  |
| --- | --- | --- |
| № | Совершенное действие | Ожидаемый результат |
| 36 | Нажатие на плейлист. | Откроется выбранный плейлист в новом activity. |
| 37 | Нажатие на кнопку «назад». | См пункт 20. |

Таблица 6 - Продолжение

|  |  |  |
| --- | --- | --- |
| 38 | Нажатие на кнопку «добавить». | Откроется CreatePlaylistActivity, где можно будет создать плейлист. |

Таблица 7 – Тесты на CreatePlaylistActivity

|  |  |  |
| --- | --- | --- |
| № | Совершенное действие | Ожидаемый результат |
| 39 | Нажатие на трек. | См пункт 1. |
| 40 | Нажатие на checkbox у трека. | Checkbox нажмется, либо отожмется. |
| 41 | Нажатие на кнопку «создать плейлист». | Все выбранные (checkbox’ом) треки образуют новый плейлист на PlaylistActivity. |
| 42 | Нажатие на кнопку «назад». | Действие вернется на PlaylistActivity. |

Таблица 8 – Тесты на PlaylistViewActivity

|  |  |  |
| --- | --- | --- |
| № | Совершенное действие | Ожидаемый результат |
| 43 | Нажатие на трек. | См пункт 1. |
| 44 | Нажатие на кнопку «назад». | Действие вернется на PlaylistActivity. |
| 45 | Нажатие на кнопку «добавить». | Откроется AddSongsActivity, где можно будет добавить треки в данный плейлист. |

Таблица 9 – Тесты на AddSongsActivity

|  |  |  |
| --- | --- | --- |
| № | Совершенное действие | Ожидаемый результат |
| 46 | Создание activity. | Появятся все песни, не принадлежащие выбранному плейлисту. |
| 47 | Нажатие на трек. | См пункт 1. |
| 48 | Нажатие на checkbox у трека. | Checkbox нажмется, либо отожмется. |
| 49 | Нажатие на кнопку «добавить». | Выбранные треки добавятся в выбранный плейлист. |
| 50 | Нажатие на кнопку «назад». | Действие вернется на PlaylistViewActivity. |

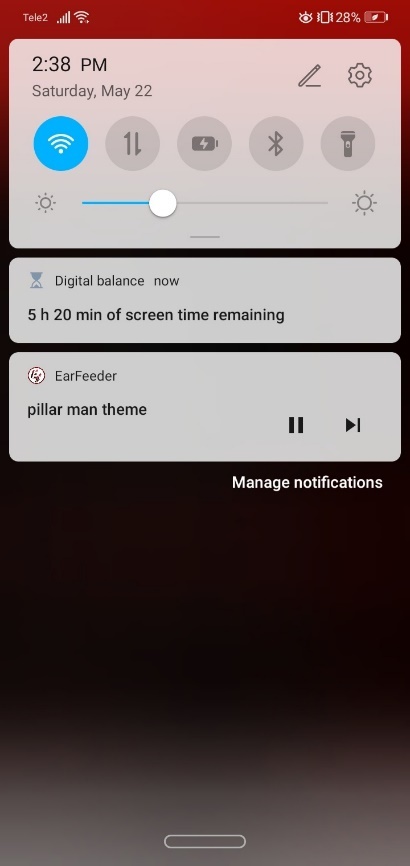
Таблица 10 – Тесты на RadioAcitivty

|  |  |  |
| --- | --- | --- |
| № | Совершенное действие | Ожидаемый результат |
| 51 | Нажатие на радио. | На экране появится колесо загрузки, которое нельзя прервать. Радио начнет воспроизводиться. |
| 52 | Нажатие на радио, когда нет интернета. | Всплывет сообщение «Нет интернет подключения». |
| 53 | Нажатие на кнопку «назад». | См пункт 20. |
| 54 | Нажатие на кнопку «добавить». | На экране появится колесо загрузки, которое нельзя прервать. Радио добавится в список и начнет воспроизводиться. |

Таблица 10 - Продолжение

|  |  |  |
| --- | --- | --- |
| 55 | Нажатие на кнопку «добавить», когда нет интернета. | Всплывет сообщение «Нет интернет подключения». |
| 56 | Нажатие на кнопку «добавить», когда ссылка неверная. | Всплывет сообщение «Неверная ссылка». |
| № | Совершенное действие | Ожидаемый результат |
| 57 | Нажатие на кнопку «старт» / «пауза» на нижней панели приложения, когда играет радио. | Появится загрузка, пока не включится радио. Поменяется иконка воспроизведения. |
| 58 | Нажатие на кнопку «следующий» на нижней панели приложения, когда играет радио. | Появится загрузка, пока радио не переключится на последующее. |
| 59 | Нажатие на кнопку «следующий» на нижней панели приложения, когда дальше нет радио. | Ничего не произойдет. |
| 60 | Нажатие на кнопку «предыдущий» на нижней панели приложения, когда играет радио. | Появится загрузка, пока радио не переключится на предыдущее. |
| 61 | Нажатие на кнопку «предыдущий» на нижней панели приложения, когда перед текущим больше нет радио. | Ничего не произойдет. |

## Результаты тестирования



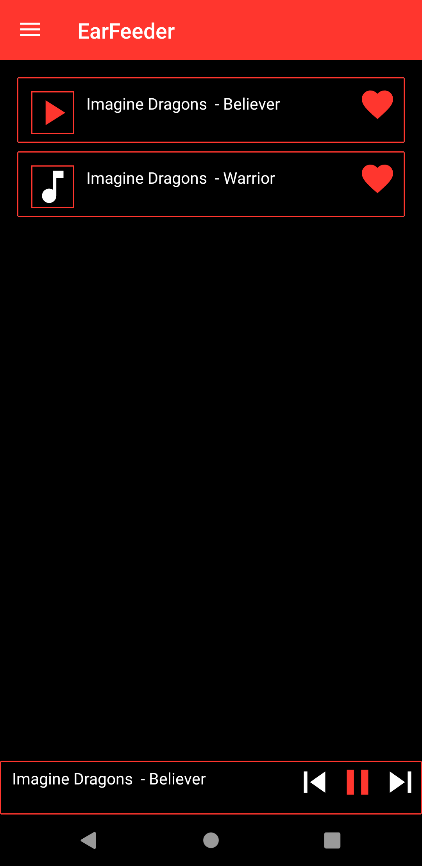


Рисунок 5 - результат теста 1

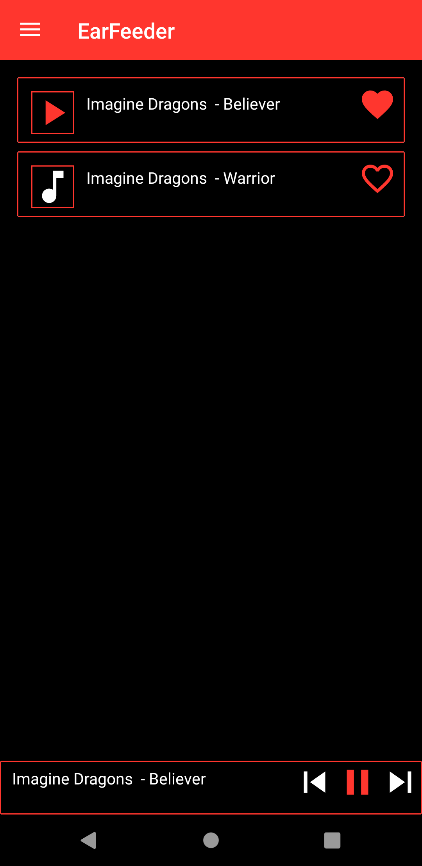
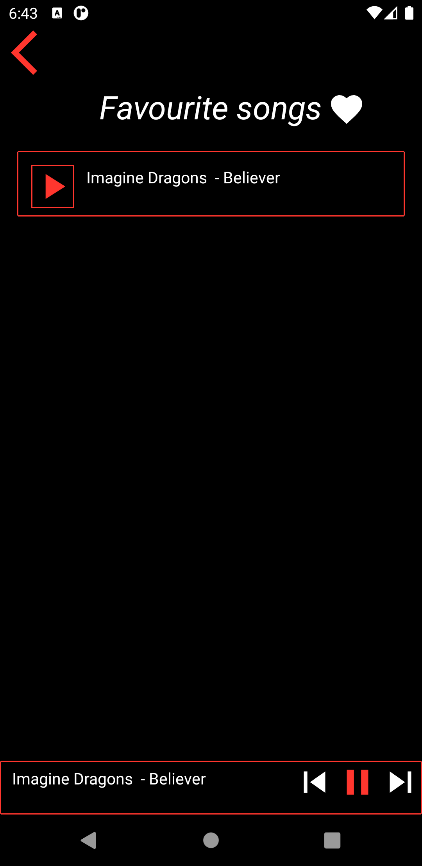


Рисунок 6 – результат теста 2

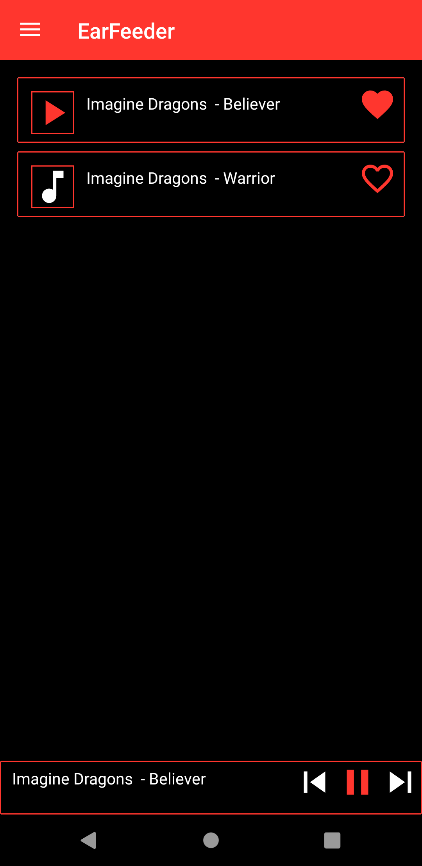
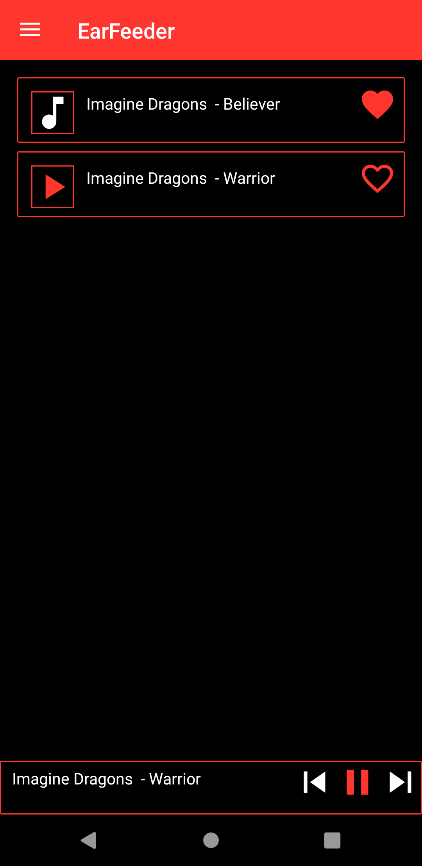


Рисунок 7 – а) результат теста 3; б) результат теста 5

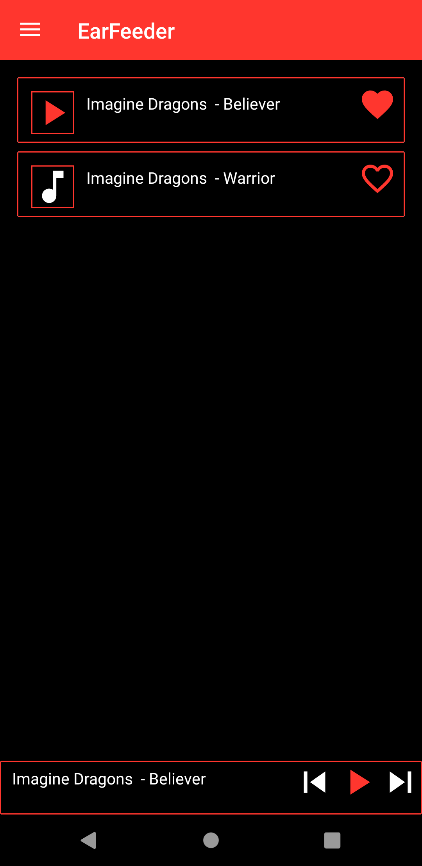
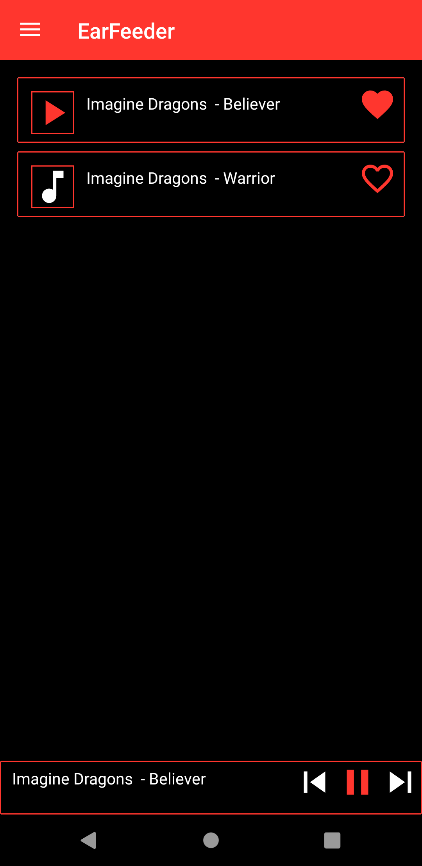


Рисунок 8– результат теста 7

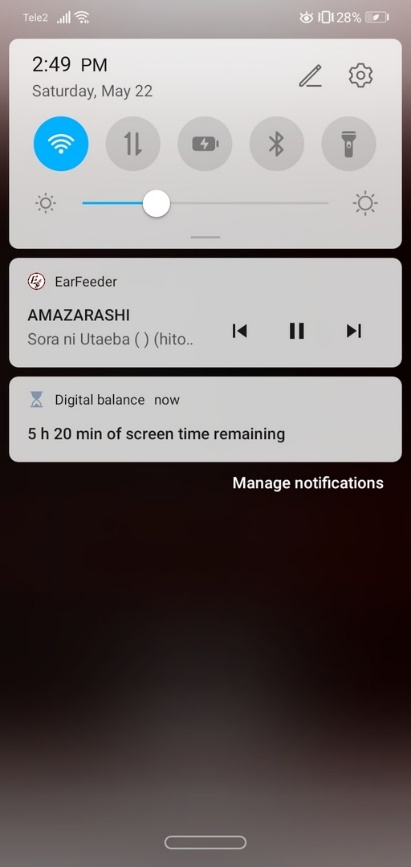
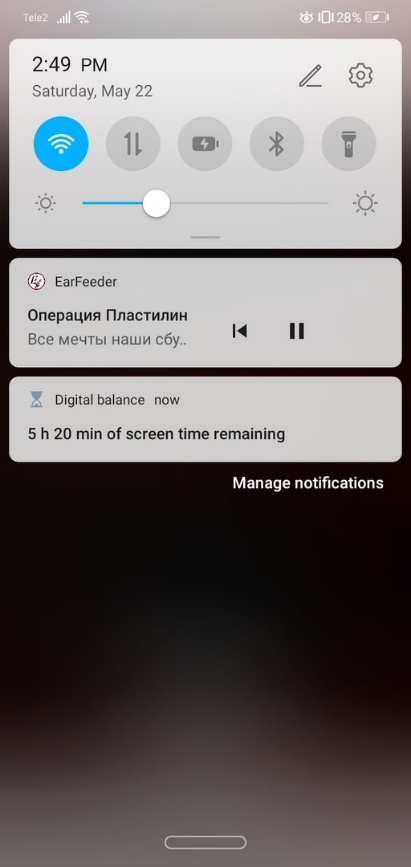


Рисунок 9 – а) результат теста 9; б) результат теста 10

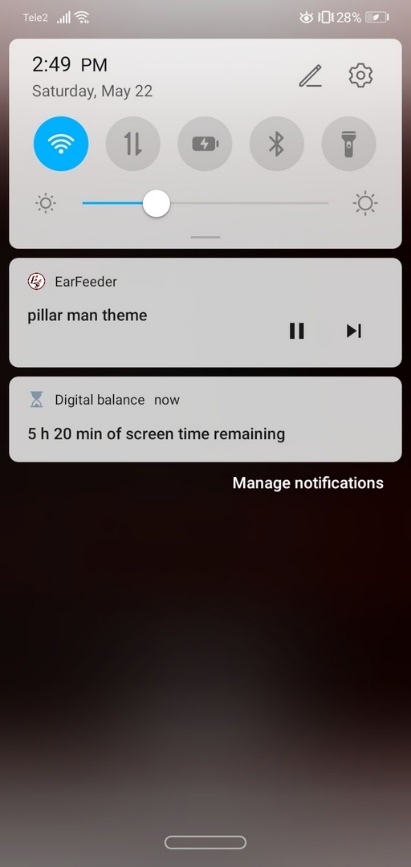
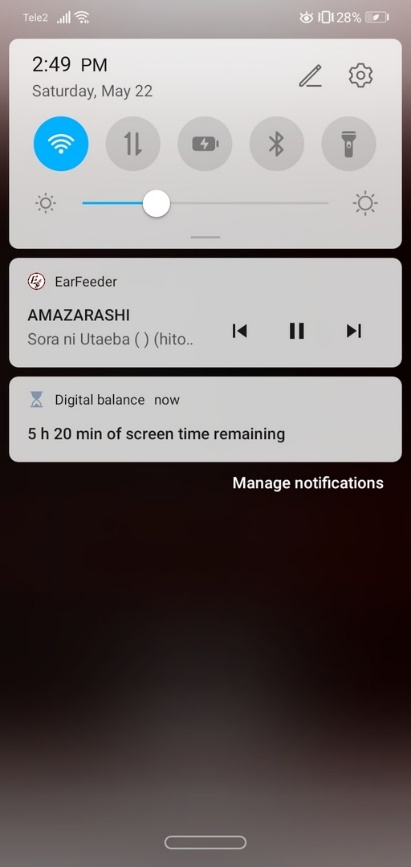


Рисунок 10 – результат тестов 11 и 12

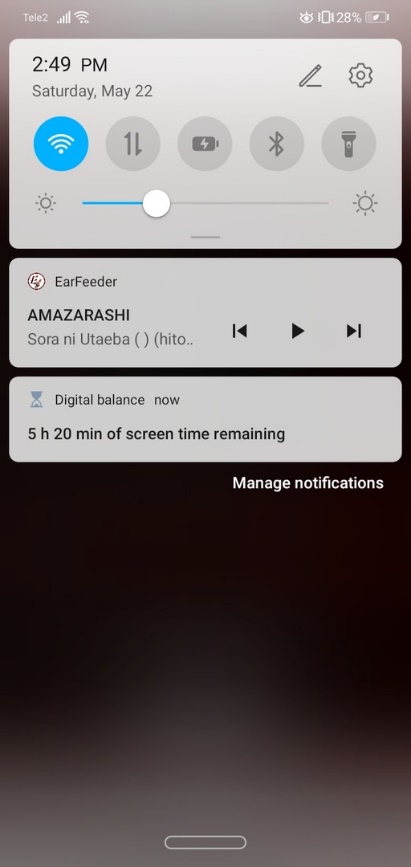
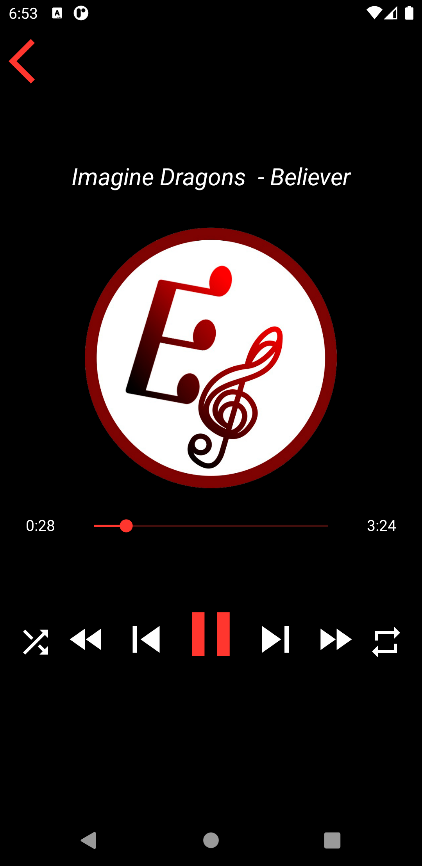


Рисунок 11 – а) результат теста 13; б) результат теста 15

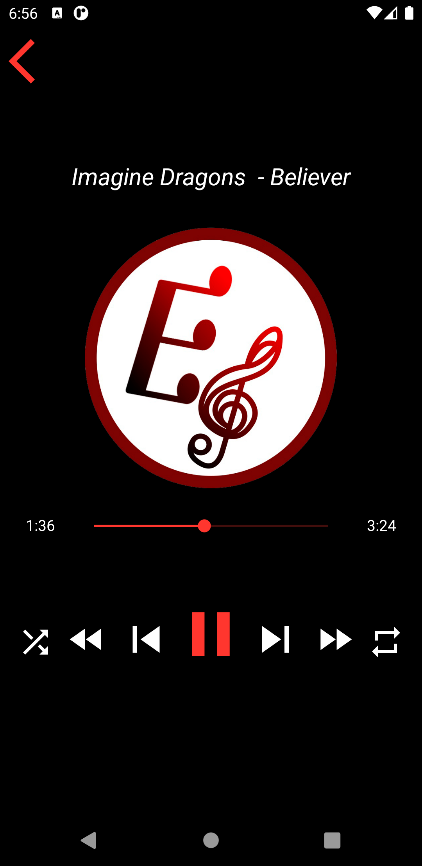


Рисунок 12 – а) результат теста 17; б) результат тестов 18 и 19

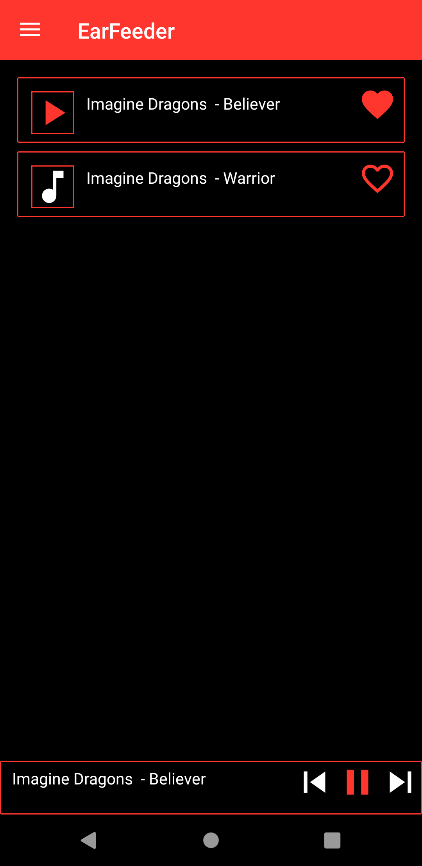


Рисунок 13 – результат теста 20

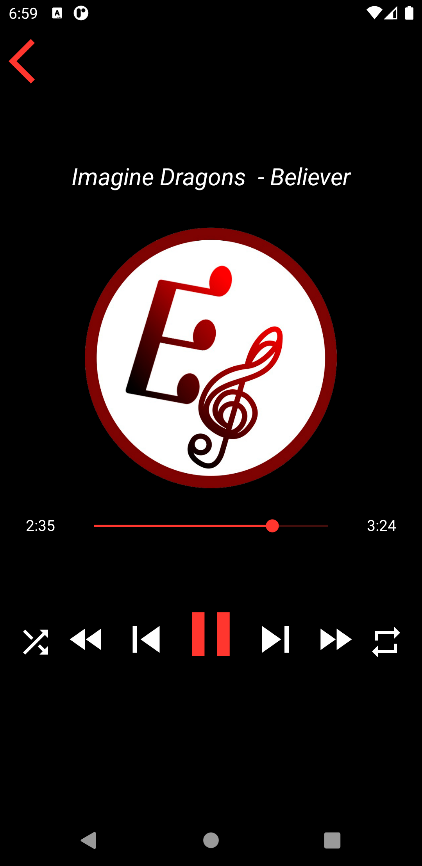




Рисунок 14 – результаты теста 21

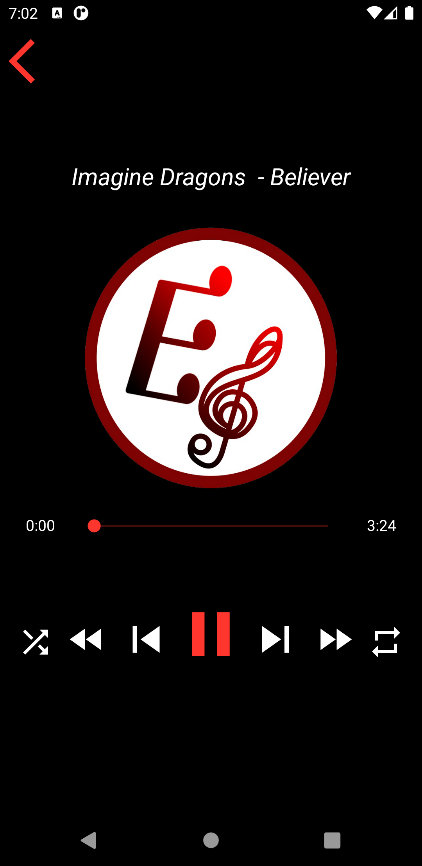
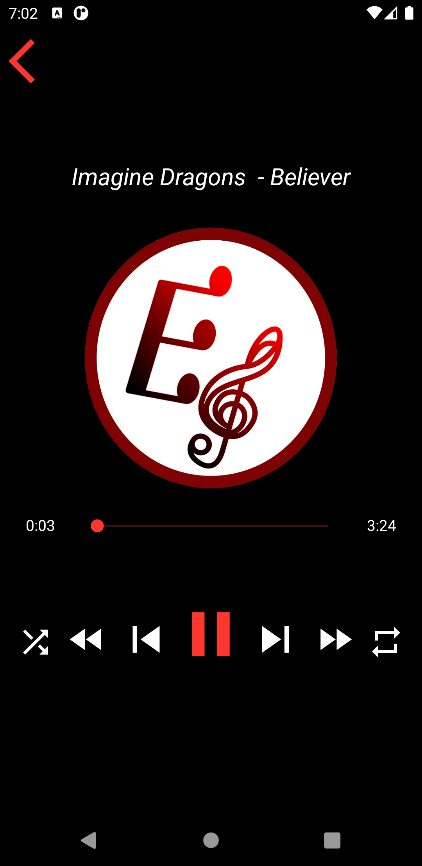


Рисунок 15 – результаты теста 22

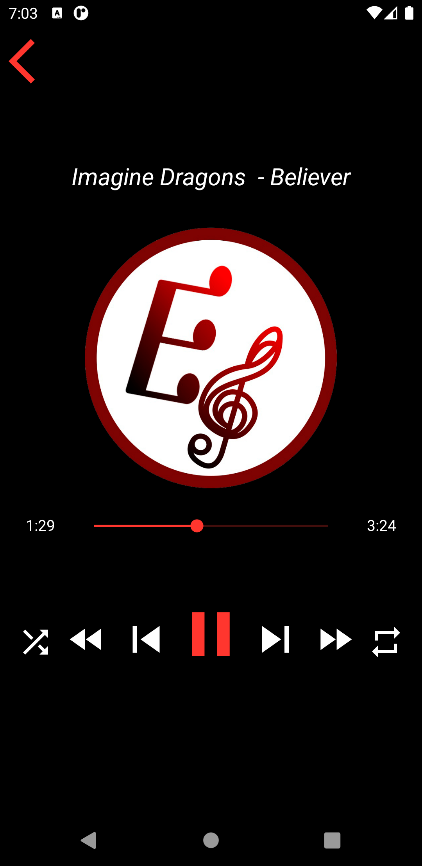
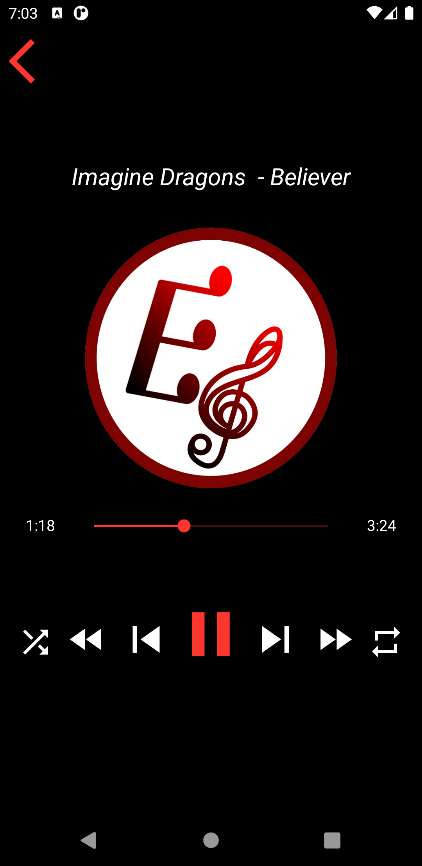


Рисунок 16 – результаты теста 23

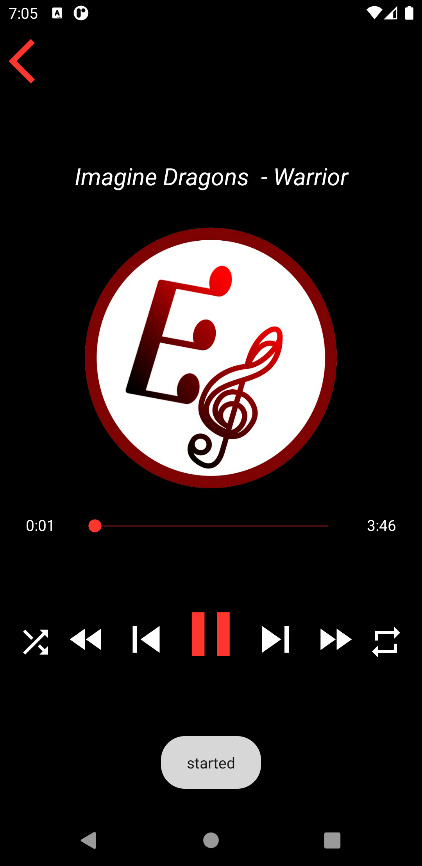
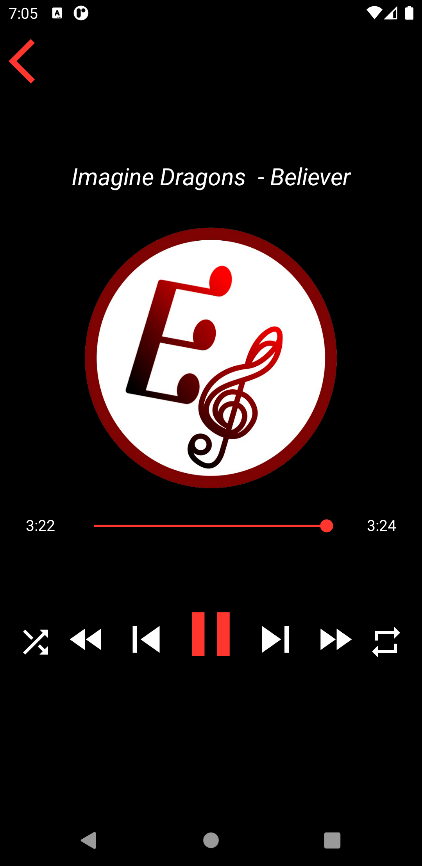


Рисунок 17 – результаты теста 24

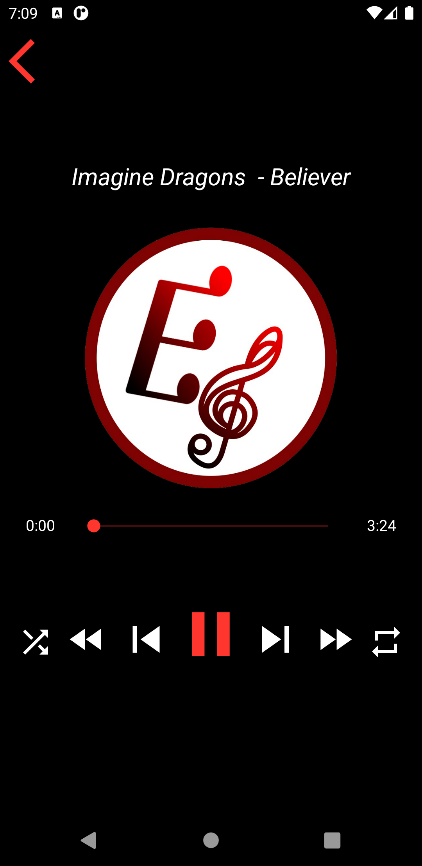
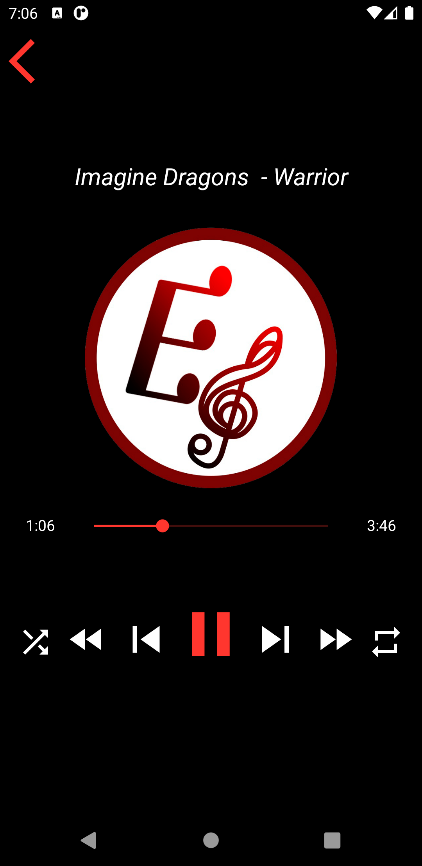


Рисунок 18 – результаты теста 25

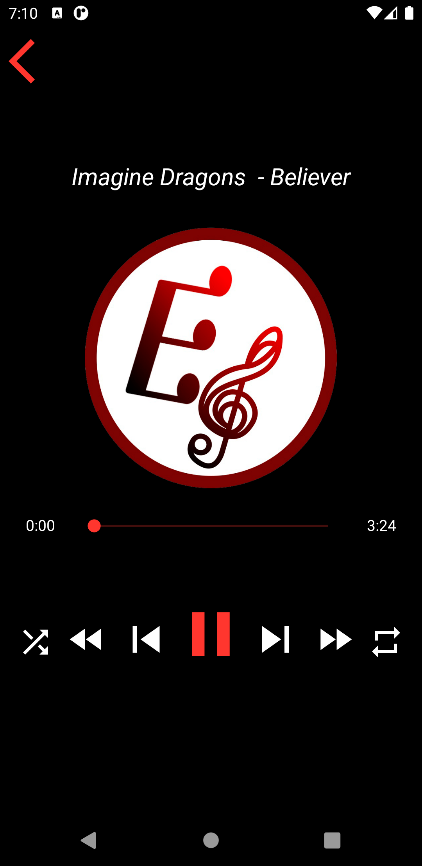
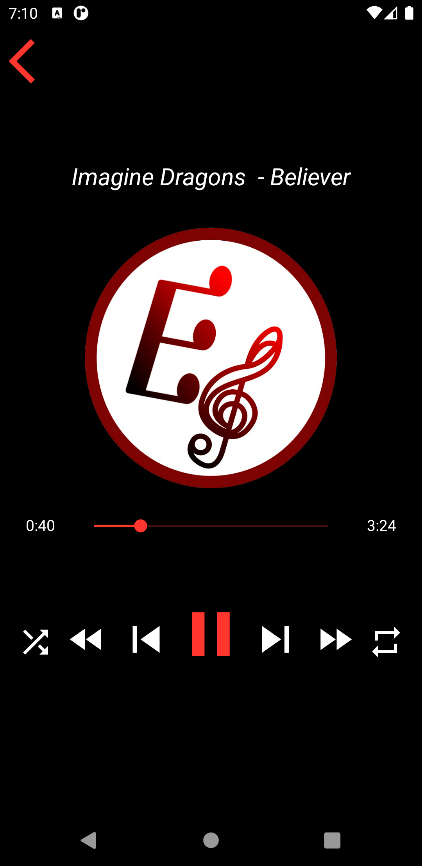


Рисунок 19 – результаты теста 26

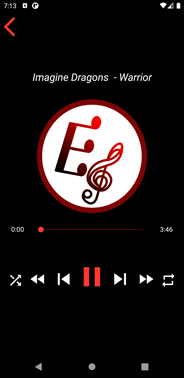
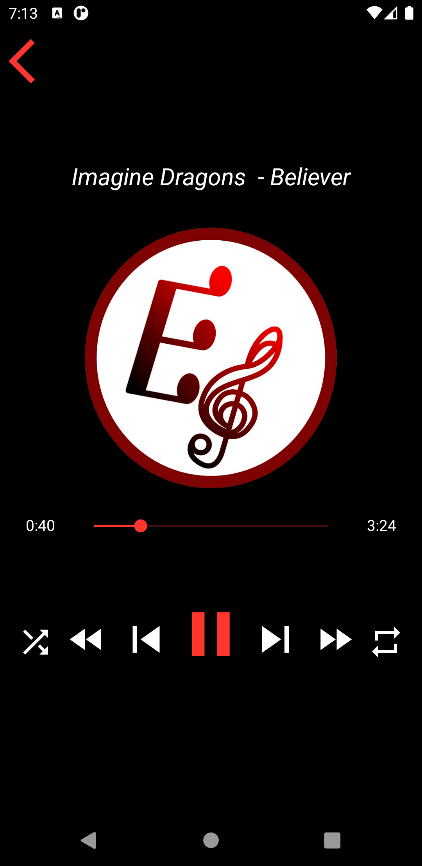


Рисунок 20 – результаты теста 27

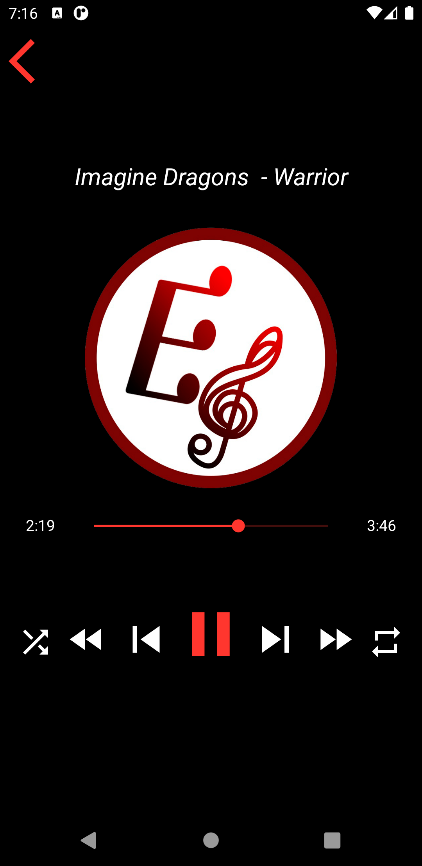


Рисунок 21 – результаты теста 28

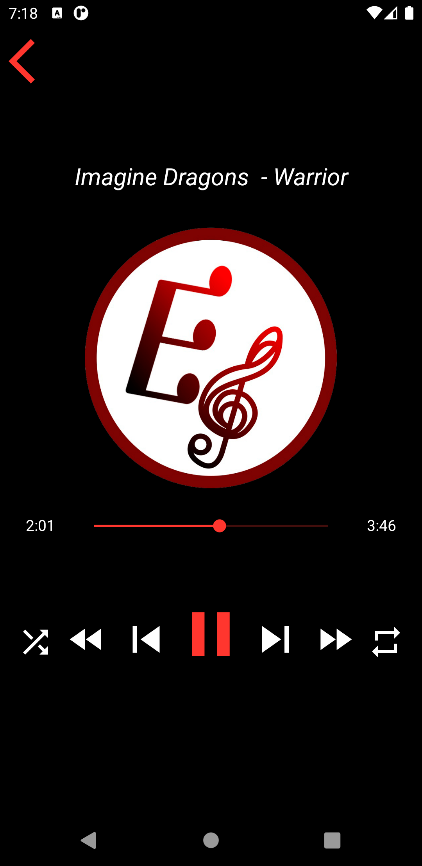
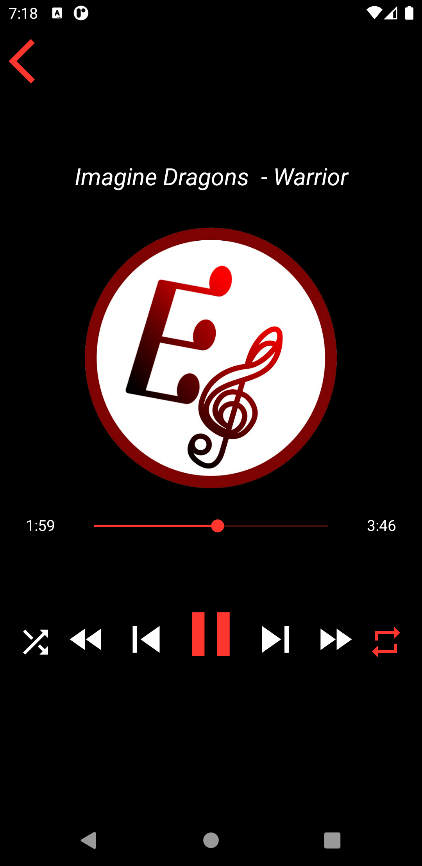


Рисунок 22 – результаты теста 29

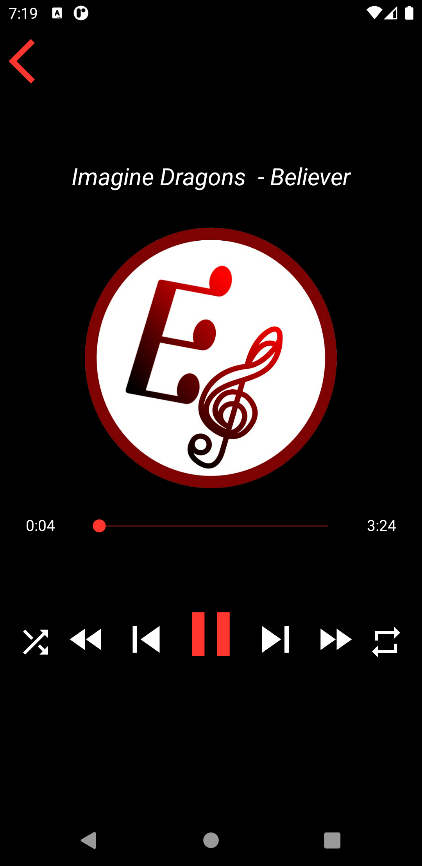
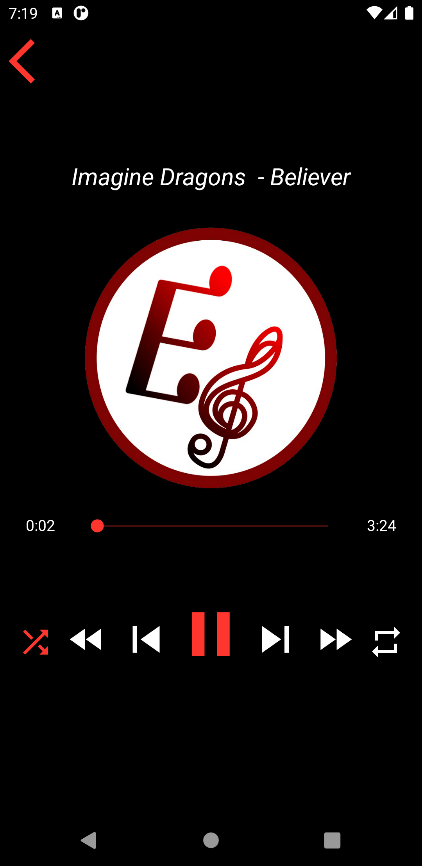


Рисунок 23 – результаты теста 30

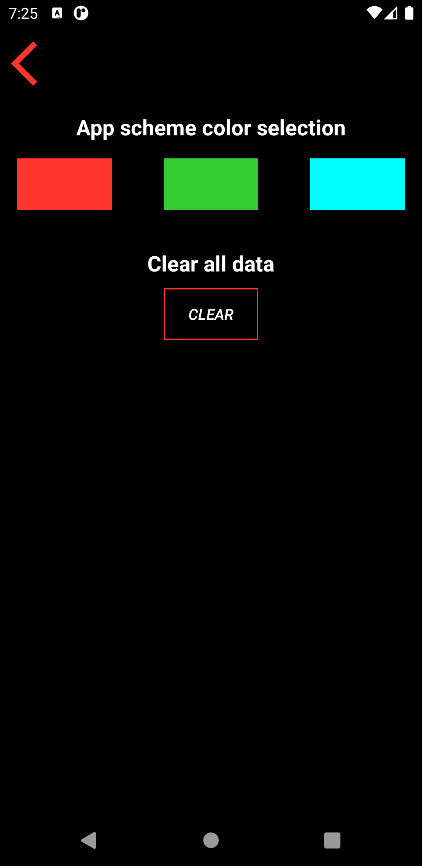
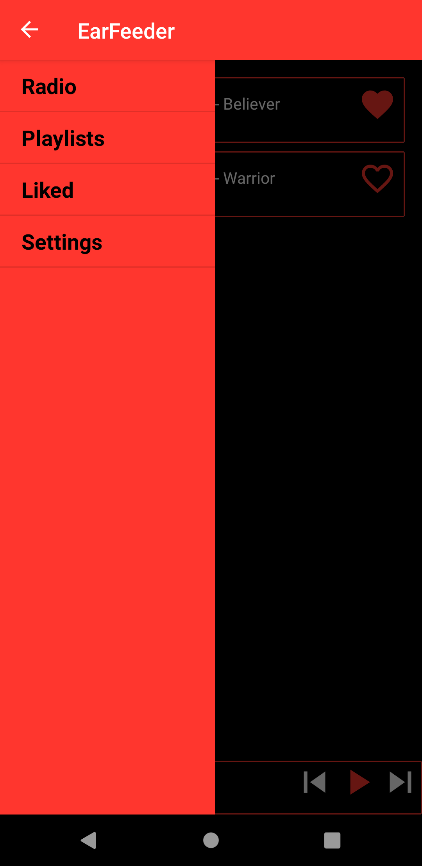


Рисунок 24 – результаты теста 31



Рисунок 25 – результаты теста 31

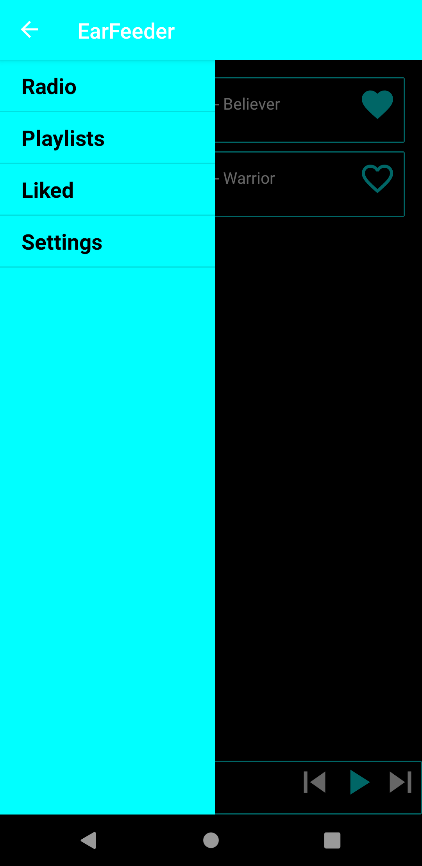
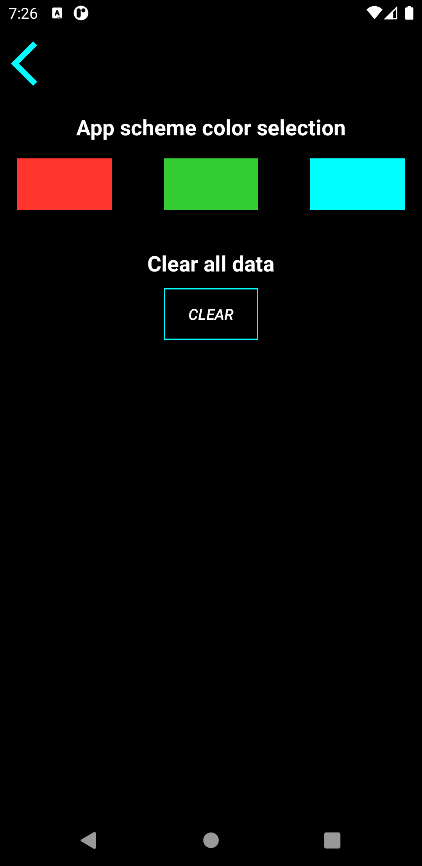


Рисунок 26 – результаты теста 31

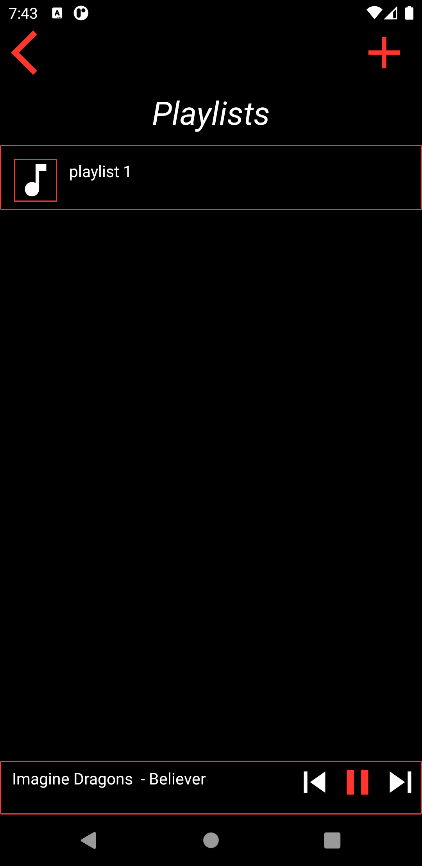
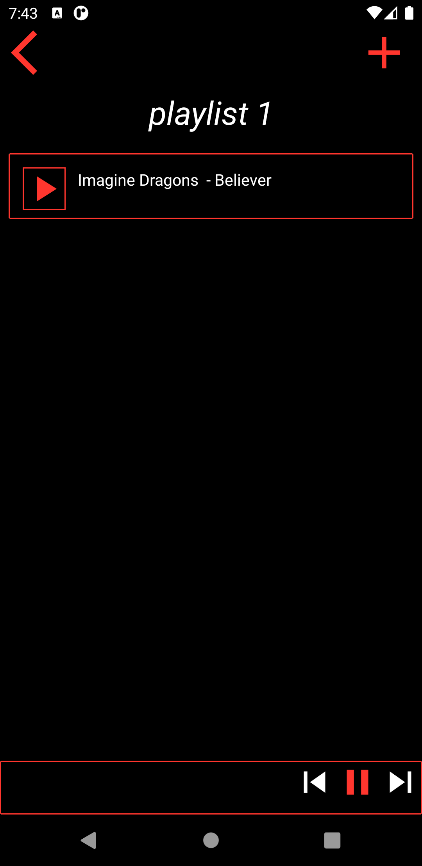


Рисунок 27 – результаты теста 36



Рисунок 28 – результат теста 37

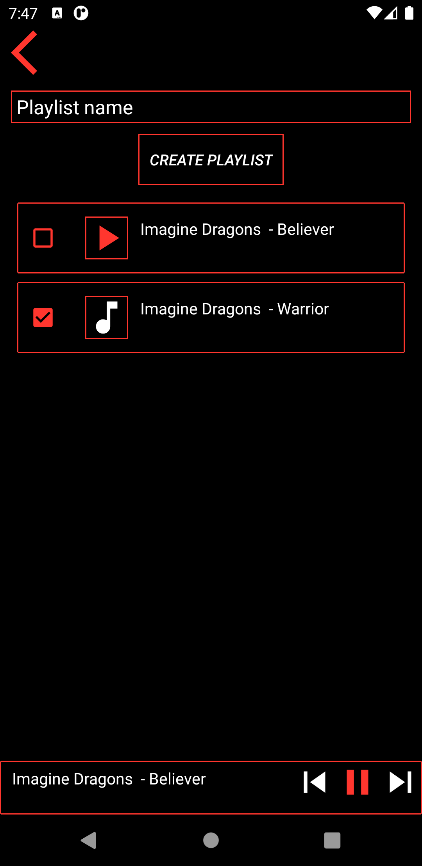


Рисунок 29 – результат теста 40

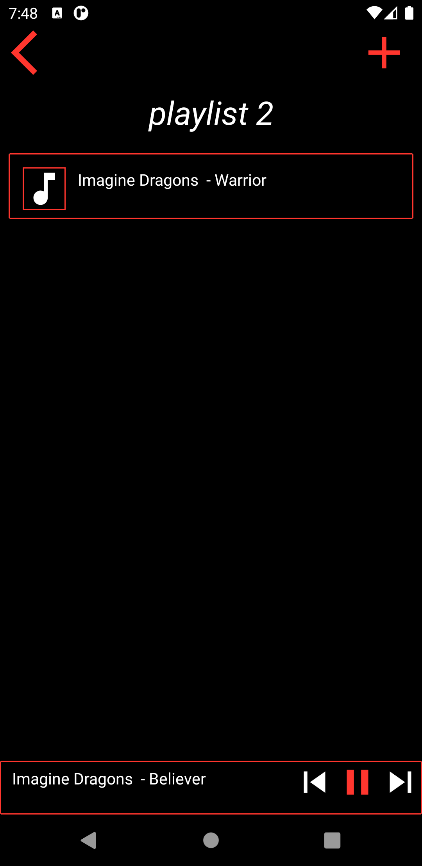
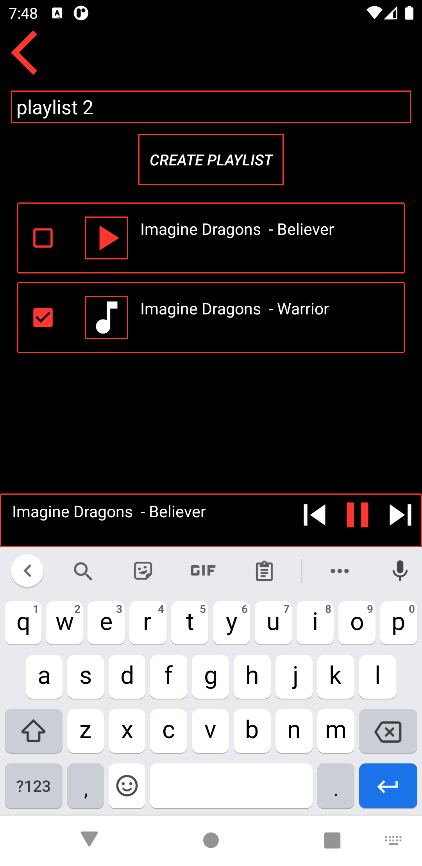


Рисунок 30 – результат теста 41

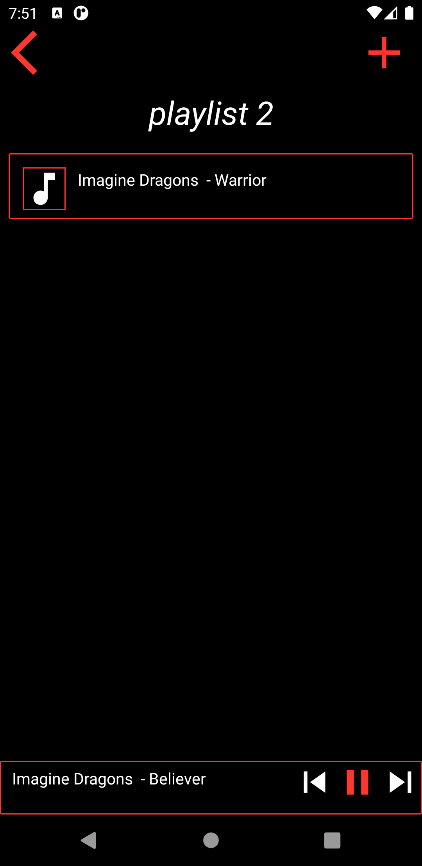
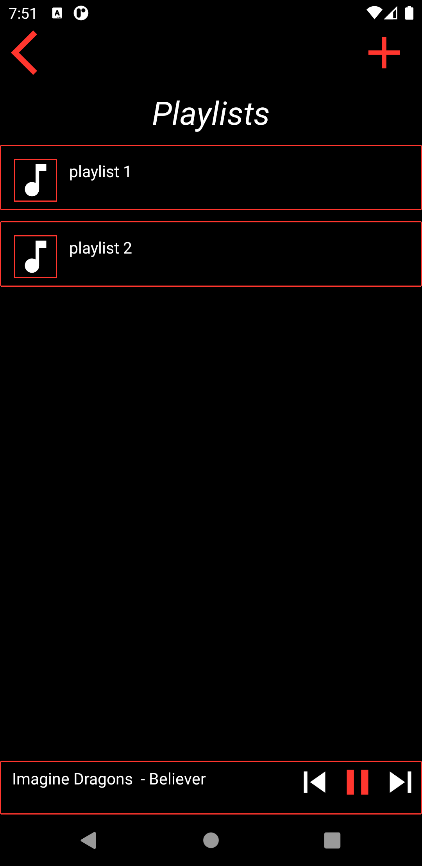


Рисунок 31 – результат теста 44

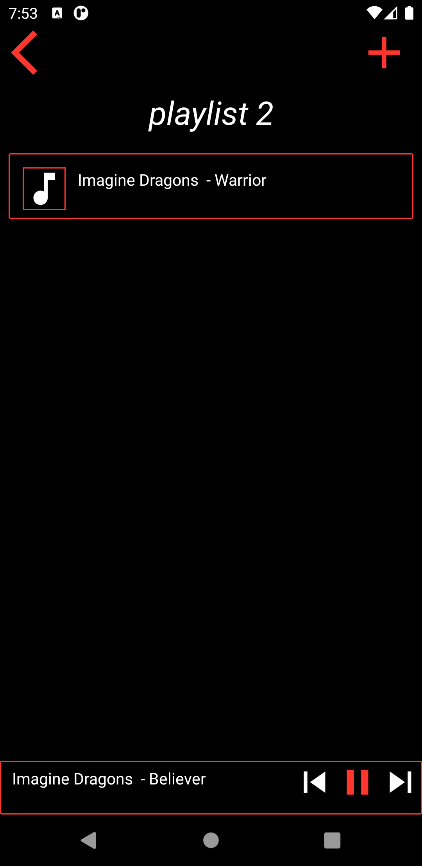
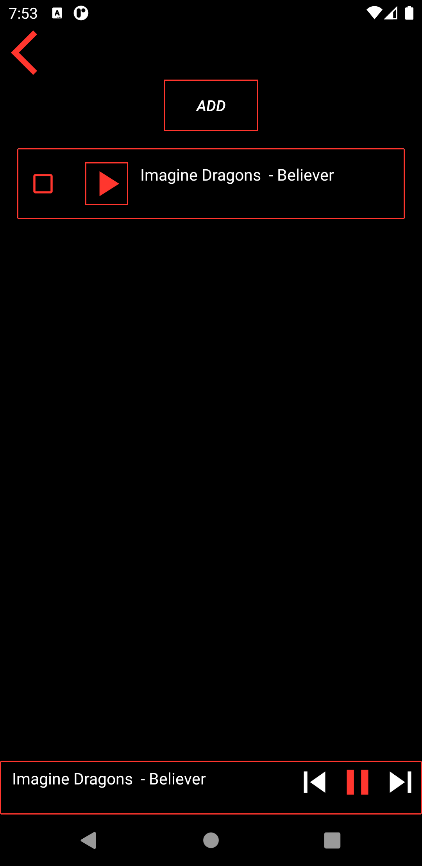


Рисунок 32– результат тестов 45 и 46

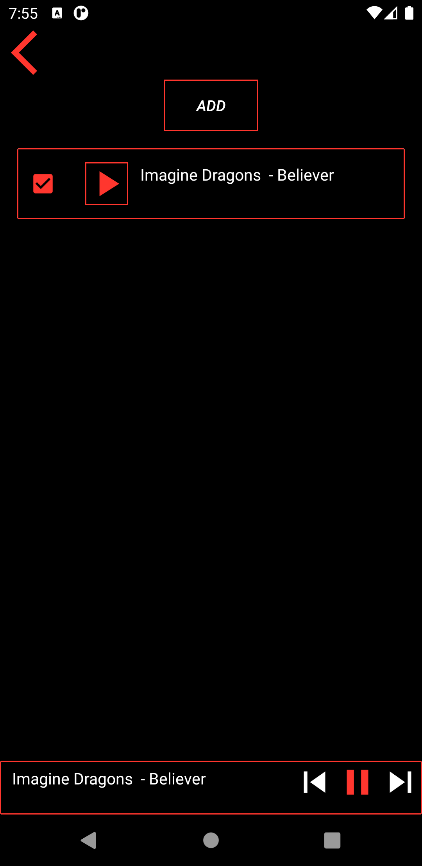
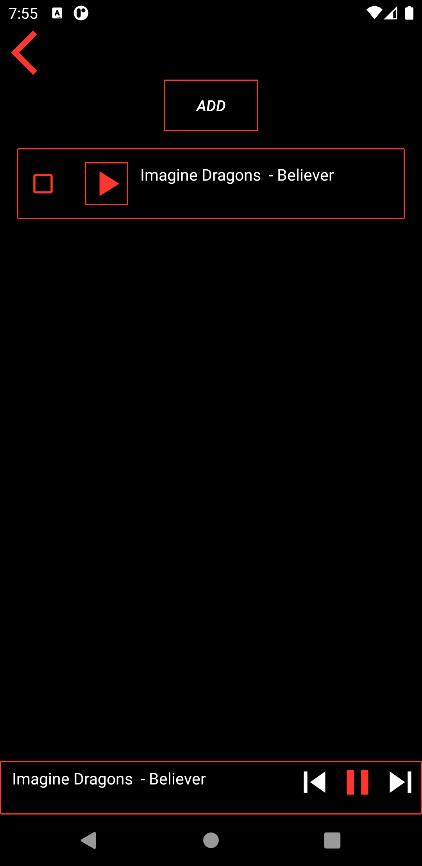


Рисунок 33 – результаты теста 48

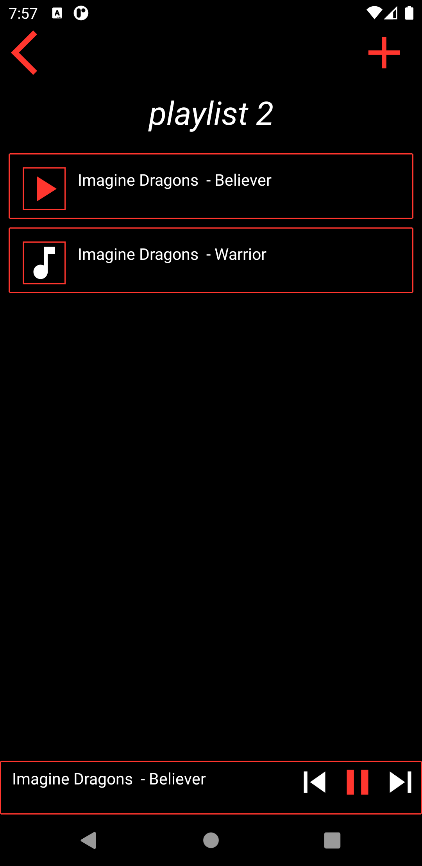
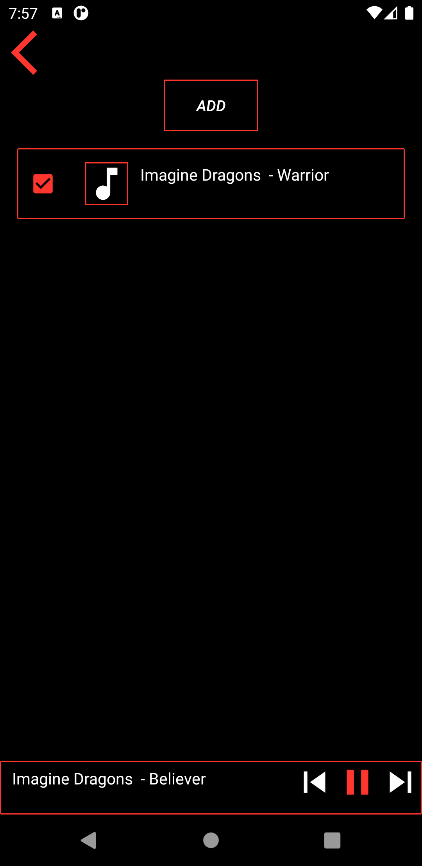


Рисунок 34 – результат теста 49

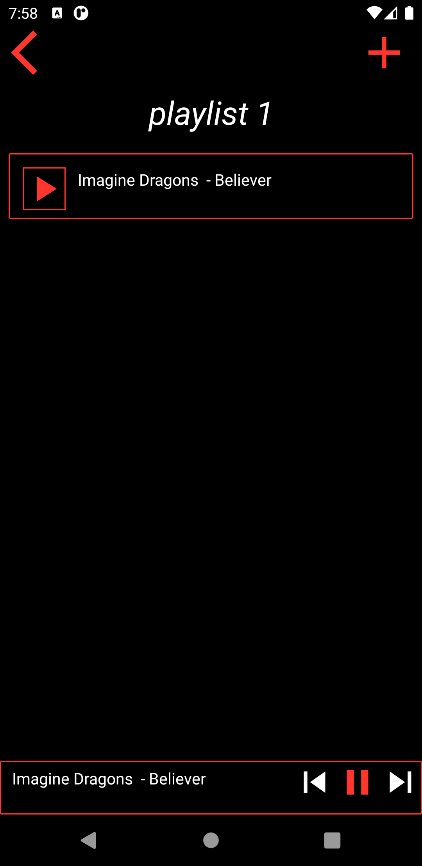
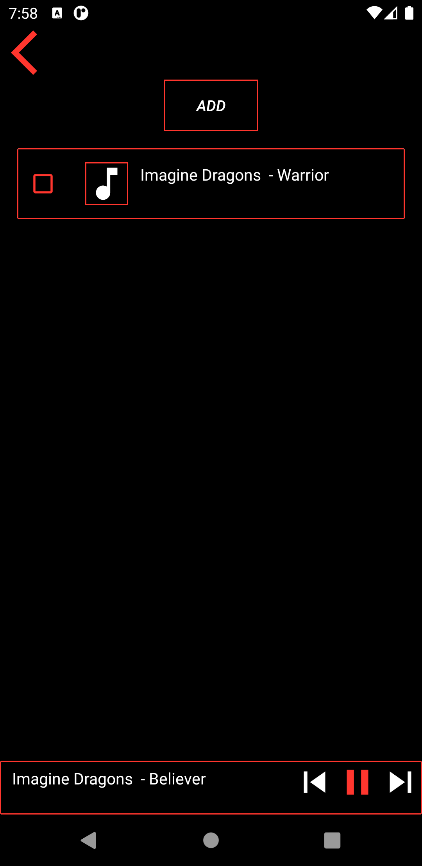


Рисунок 35 – результат теста 50

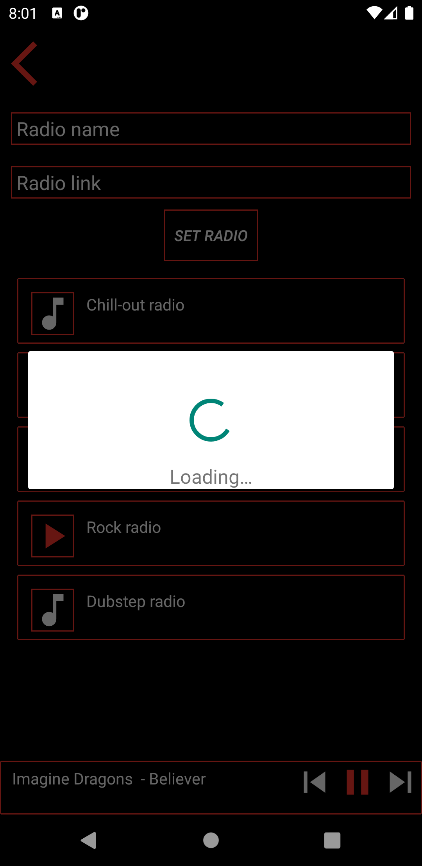
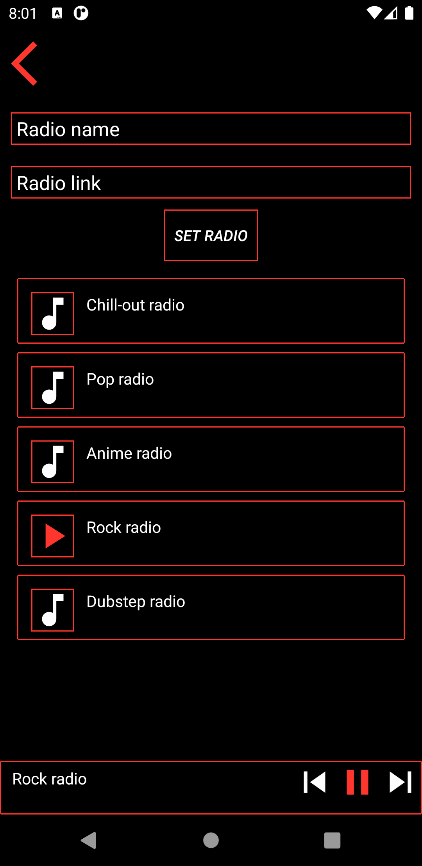


Рисунок 36 – результат теста 51

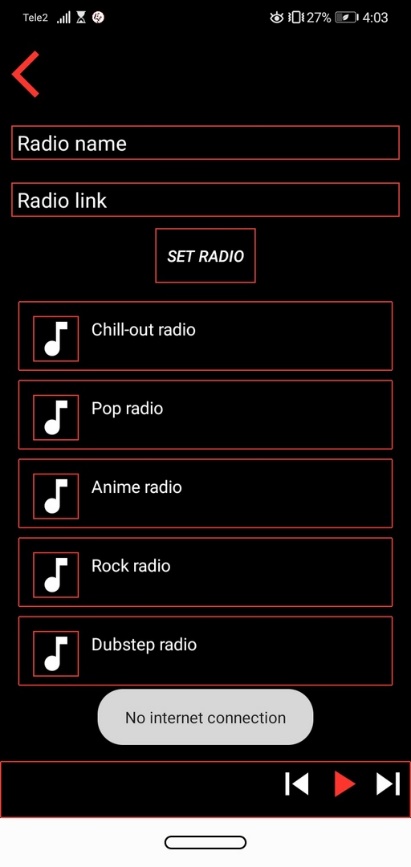


Рисунок 37 – результат теста 52

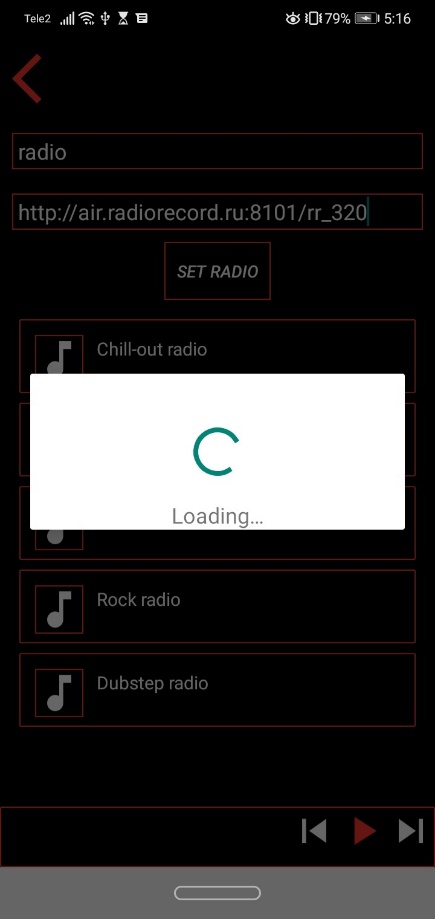
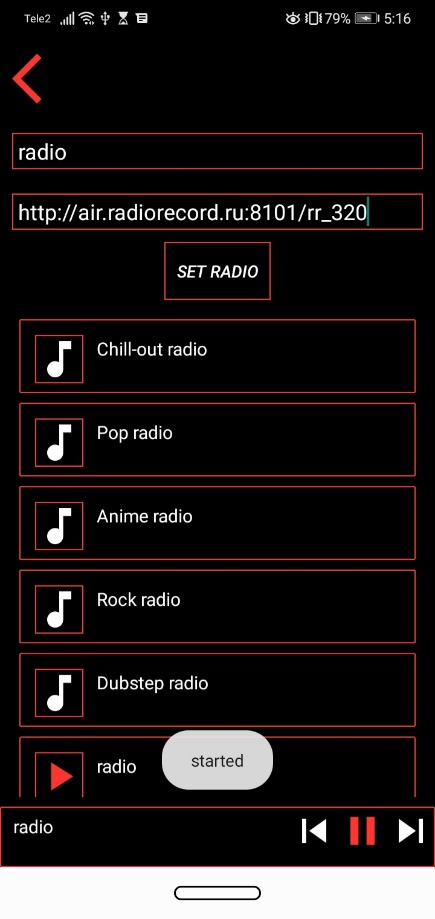


Рисунок 38– результат теста 54

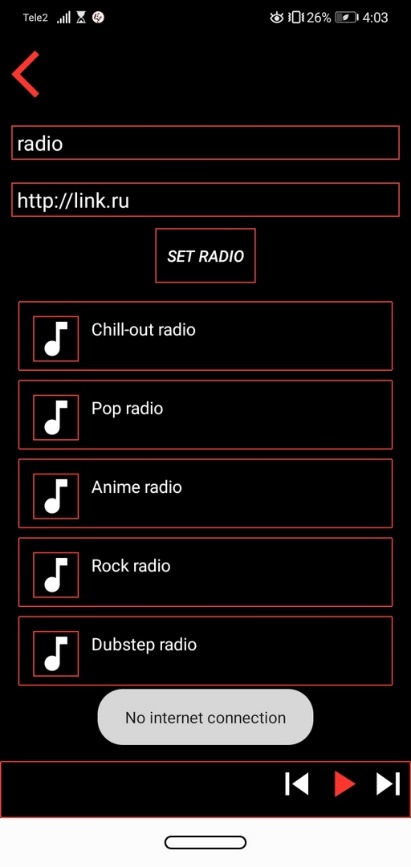


Рисунок 39 – результат теста 55

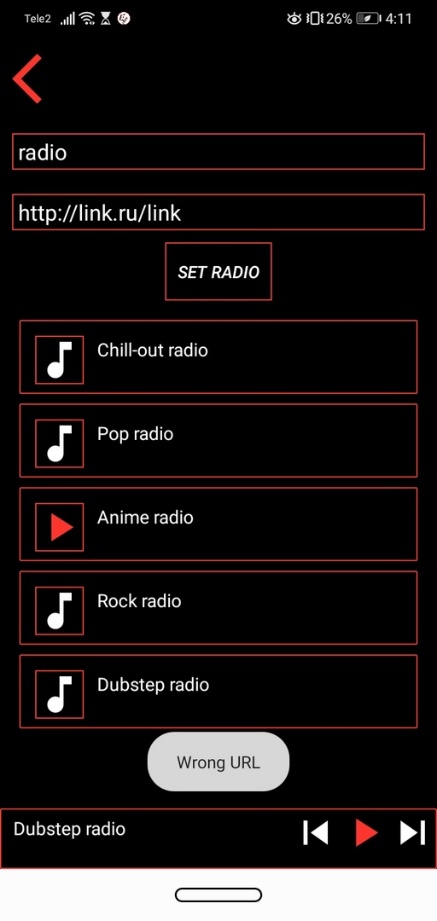


Рисунок 40 – результат теста 56

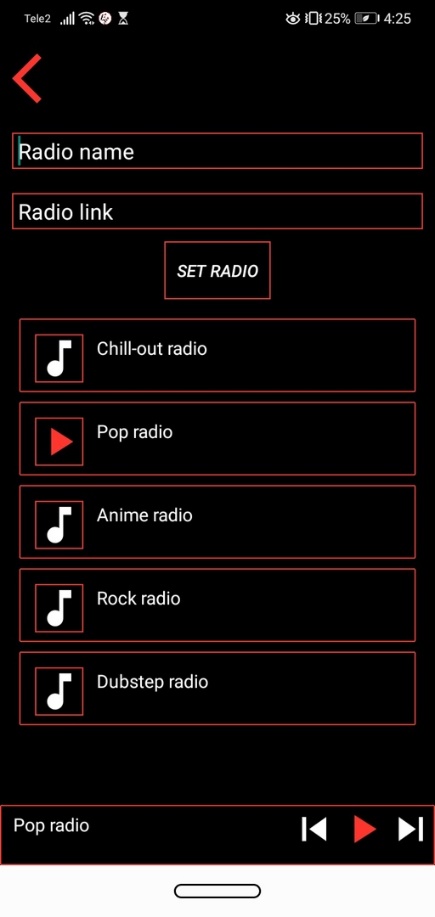
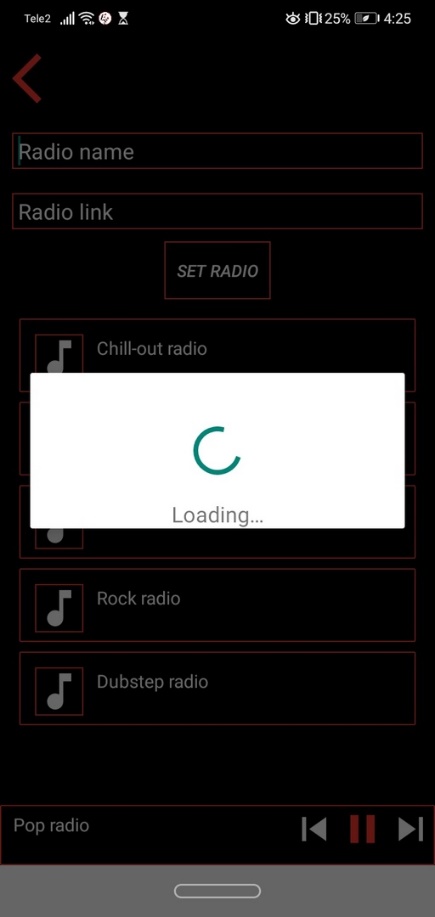
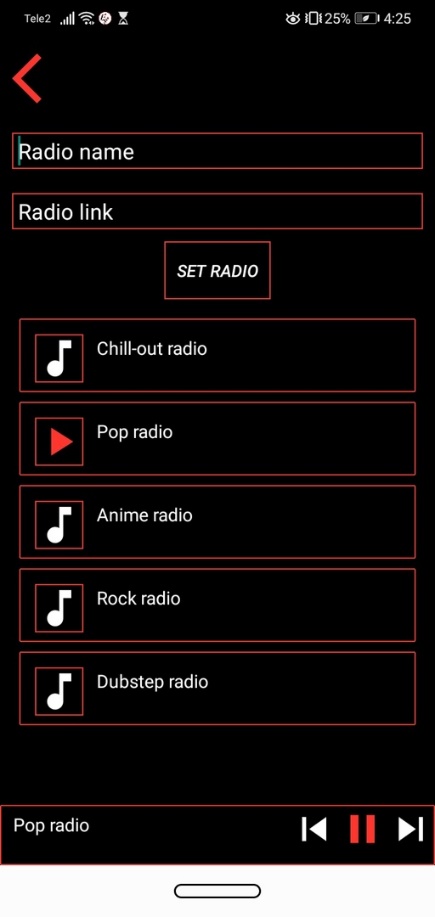
  

Рисунок 41– результат теста 57

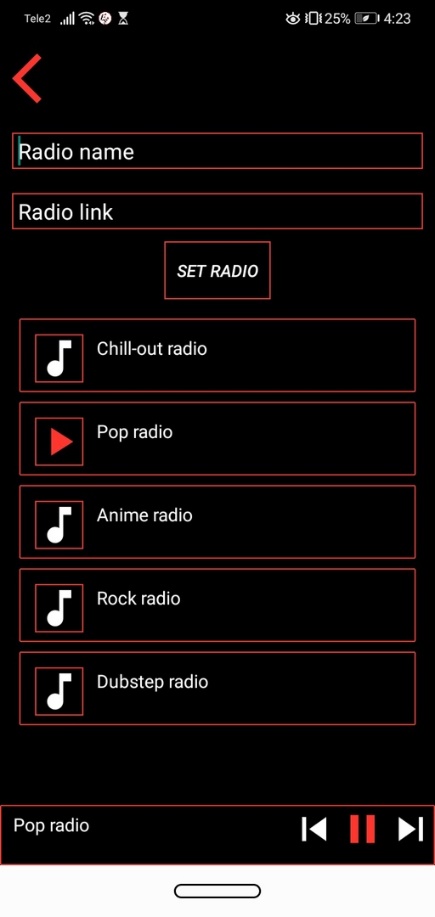
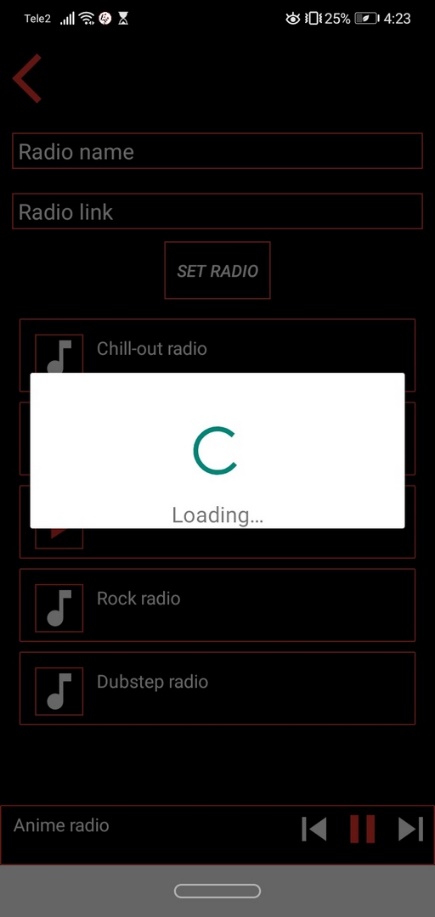
  

Рисунок 42– результат теста 58

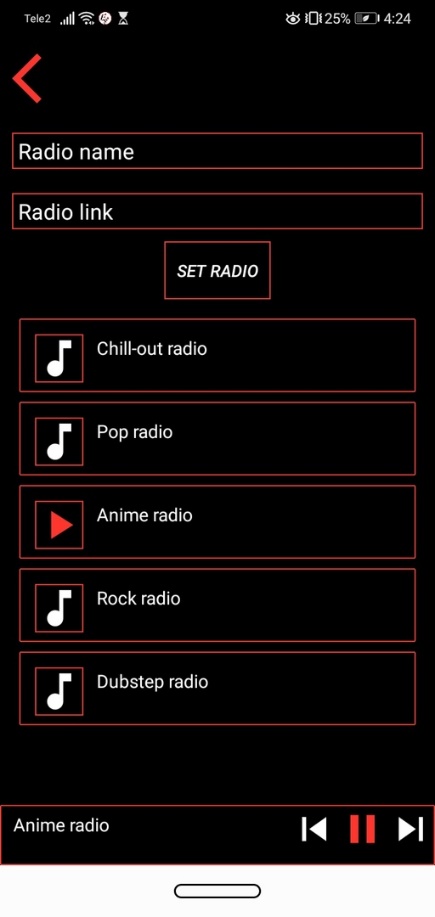
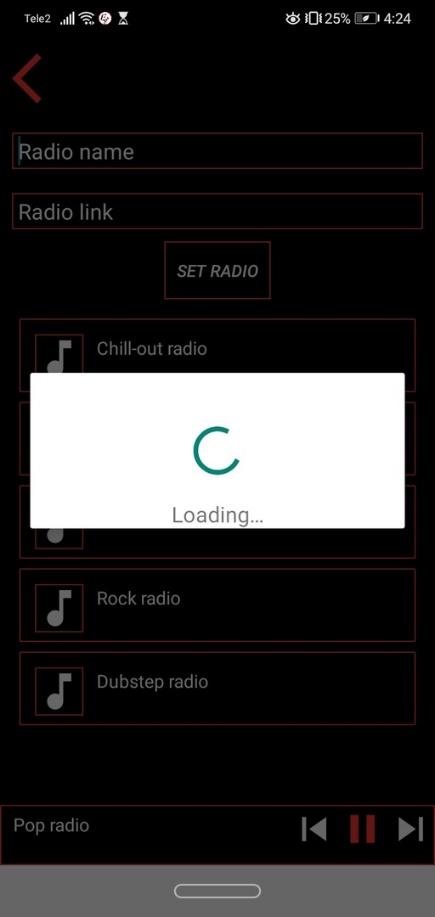
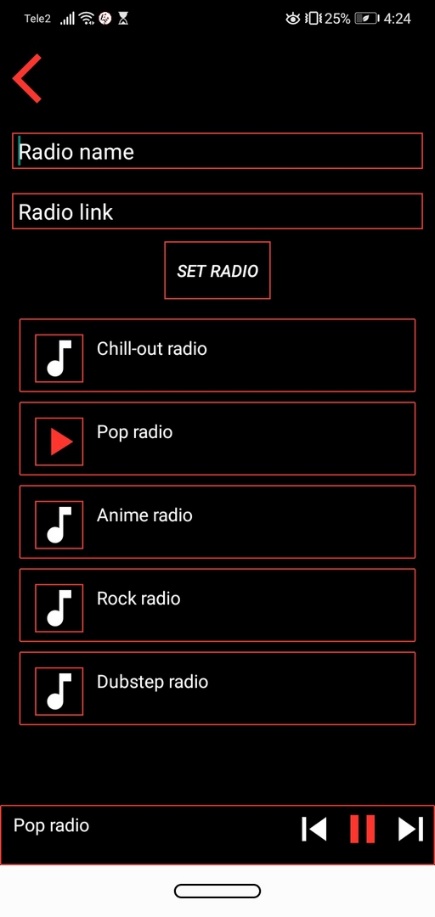
  

Рисунок 43 – результат теста 60

# Заключение

В процессе разработки данного приложения были получены и закреплены следующие навыки:

- проектирования и применения объектного подхода в реализации системы;

- проектирования пользовательского интерфейса и компоновке элементов взаимодействия с пользователем;

- создания и проектирования локальных баз данных с помощью СУБД SQLite;

- программирования под среду Android (в том числе: работа с внутренними файлами телефона, потоки, сервисы, фрагменты, уведомления и http-запросы);

- применения необходимых видов паттернов программирования (singleton, DAO, observer).

В ходе дипломного проектирования были решены следующие задачи:

1. Выработаны функциональные требования приложения.
2. Спроектированы интерфейс, классы и локальная база данных.
3. Разработано приложение согласно реализованным требованиям и проектированию.
4. Приложение было протестировано с помощью приемочного тестирования.
5. Составлена документация по всей проделанной работе.

Исходный код приложения опубликован в публичном репозитории на сервисе GitHub [12].

Цель курсового проекта достигнута.

Список использованных источников

1. Head first. Паттерны программирования. Эрик Фримен, Элизабет Робсон [2020] (дата обращения: 15.04.2021)
2. Сборник ссылок для радио [электронный ресурс] // URL: <https://mc-terra.ru/index.php?topic=7771.0> (дата обращения: 12.04.2021)
3. Start Android [электронный ресурс] // учебник по андроид для начинающих и продвинутых, URL: <https://startandroid.ru/ru/> (дата обращения: 04.04.2021)
4. Java. Экспресс-курс [электронный ресурс] // Сайт Александра Климова [сайт], URL: http :// developer.alexanderklimov . ru /android / java / java. php (дата обращения: 04.04.2021)
5. API Specification for the Java 7 SE. [официальный сайт] URL: http :// docs. oracle. com / javase /7/ docs /api / (дата обращения: 05.04.2021)
6. Отношения классов — от UML к коду [Электронный ресурс] // Хабрахабр: [сайт]. [2006]. URL: http://habrahabr.ru/post/150041/ (дата обращения: 20.05.2021)
7. Что такое ORM [Электронный ресурс]. [2017]. URL: <https://myrusakov.ru/what-is-it-orm-php.html> (дата обращения: 27.05.2021)
8. Data Access Object (DAO) [Электронный ресурс] // Хабрахабр: [сайт]. [2015]. URL: <https://habr.com/ru/post/262243/> (дата обращения: 27.05.2021)
9. Одиночка (Singleton) | Паттерны [Электронный ресурс] // Метанит: [сайт]. [2018]. URL: <https://metanit.com/sharp/patterns/2.3.php> (дата обращения: 27.05.2021)
10. Библиотека для визуализации музыкальных волн [Электронный ресурс] // GitHub: [сайт]. URL: <https://github.com/gauravk95/audio-visualizer-android> (Дата обращения: 08.04.2021)
11. Библиотека Room для работы с СУБД SQLite [Электронный ресурс]. URL: <https://developer.android.google.cn/jetpack/androidx/releases/room?hl=en>

(Дата обращения: 05.05.2021)

1. Репозиторий с исходных кодом проекта [Электронный ресурс] // GitHub: [сайт]. URL: <https://github.com/MAL1S/MusicPlayer.git>