1. **СЕРВИСЫ**

*Задание 1*

Добавим в проект в каталог res папку raw, для этого сделаем правый щелчок на каталоге res и в контекстном меню выберем New → Android Resource Directory. Укажем в качестве типа папки (Resource type) – raw. Теперь поместим в эту папку произвольный mp3-файл под именем music.mp3.

Добавим новый класс сервиса, для этого сделаем правый щелчок на каталоге java и в контекстном меню выберем New → Service → Service. Назовем его MediaService.

Для воспроизведения аудиофайла определите в классе MediaService код, предоставленный на рисунке 1.1.



Рисунок 1.1 – Содержимое файла MediaService.java

Добавим в файл activity\_main.xml две кнопки для управления сервисом XML разметка показана на рисунке 1.2.

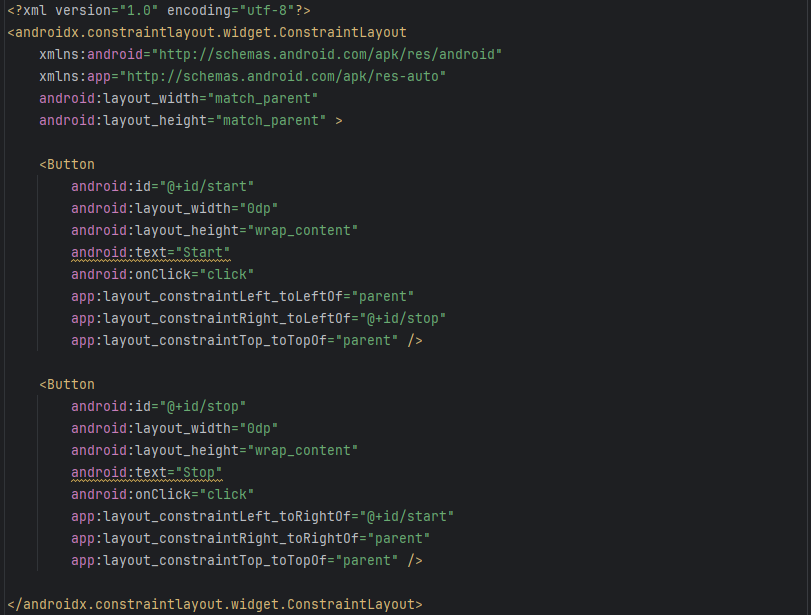


Рисунок 1.2 – Содержимое файла activity\_main.xml

Изменим код MainActivity.java рисунок 1.3.

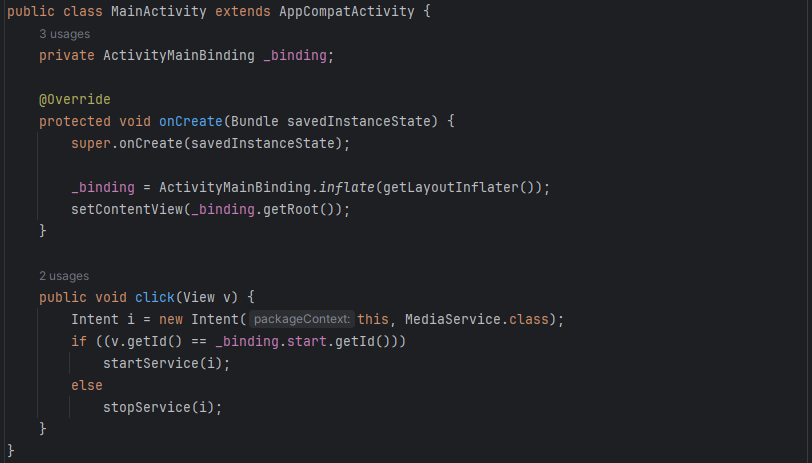


Рисунок 1.3 – Содержимое файла MainActivity.java

Запустим наше приложение чтобы убедиться в его работоспособности. Приложение, запущенное на эмуляторе предоставлено на рисунке 1.4.



Рисунок 1.4 – Приложение запущенное на эмуляторе

1. **ДИАЛОГОВЫЕ ОКНА**

**DatePickerDialog и TimePickerDialog**

*Задание 2*

Определите разметку интерфейса в activity\_main.xml рисунок 2.1

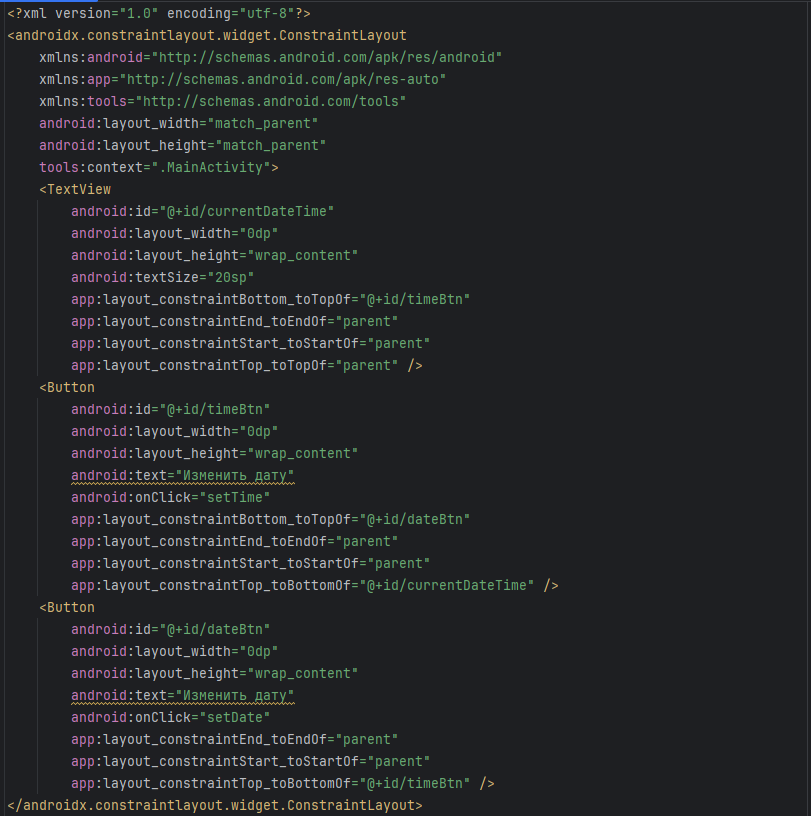


Рисунок 2.1 – Содержимое файла activity\_main.xml

Изменим код MainActivity рисунок 2.2.

Ключевым классом здесь является java.util.Calendar из стандартной библиотекиклассов Java. В методе setInitialDateTime() получаем из экземпляра этого класса количество миллисекунд dateAndTime.getTimeInMillis и с помощью форматирования выводим на текстовое поле.

Метод setDate(), вызываемый по нажатию на кнопку, отображает окно для выбора даты. При создании окна его объекту передается обработчик выбора даты DatePickerDialog.OnDateSetListener, который изменяет дату на текстовом поле

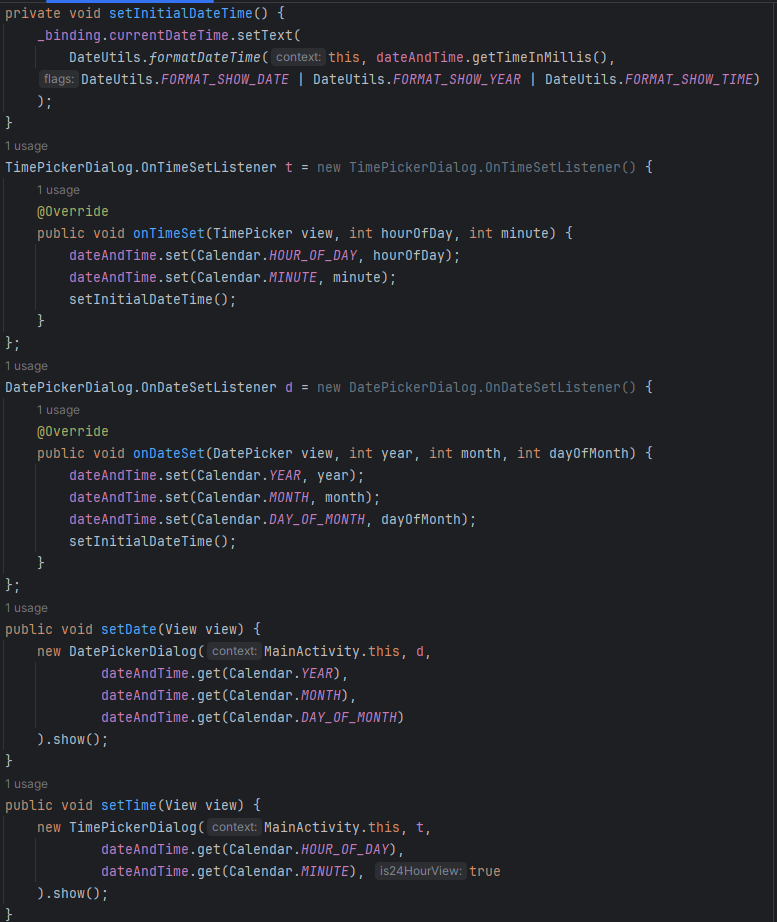


Рисунок 2.2 – Часть методов реализованных в MainActivity.java

Запустим наше приложение чтобы убедиться в его работоспособности. Приложение, запущенное на эмуляторе предоставлено на рисунке 2.3.



Рисунок 2.3 – Приложение запущенное на эмуляторе

**Кастомизация диалогового окна с AlertDialog**

*Задание 3*

Часть 1

В листинге 7.6 для создания диалогового окна в методе onCreateDialog() применяется класс AlertDialog.Builder. С помощью набора своих методов он позволяет настроить отображение диалогового окна:

* setTitle: устанавливает заголовок окна;
* setView: устанавливает разметку интерфейса окна;
* setIcon: устанавливает иконку окна;
* setPositiveButton: устанавливает кнопку подтверждения действия;
* setNeutralButton: устанавливает "нейтральную" кнопку, действие которой
* может отличаться от действий подтверждения или отмены;
* setNegativeButton: устанавливает кнопку отмены;
* setMessage: устанавливает текст диалогового окна;
* create: создает окно.

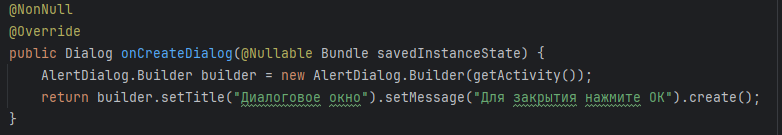


Рисунок 3.1 – Содержимое файла CustomDialogFragment.java

На рисунке 3.2 диалоговое окно просто выводит некоторое сообщение. Для вызова этого диалогового окна нам нужно определить кнопку

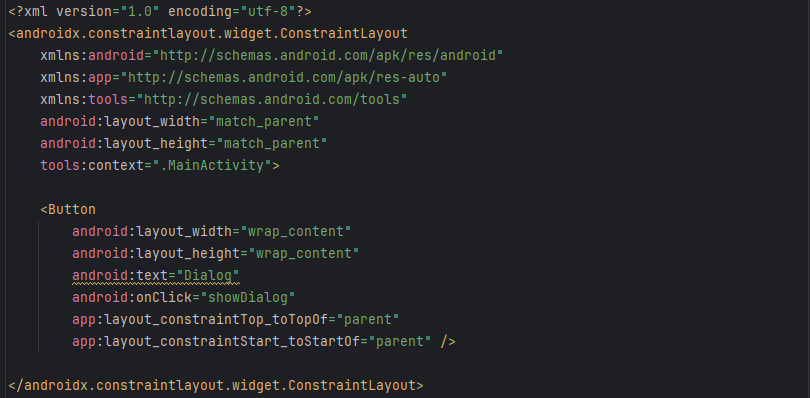


Рисунок 3.2 – Содержимое файла activity\_main.xml

В коде MainActivity определим обработчик нажатия кнопки, который будет запускать диалоговое окно рисунок 3.3.

Здесь для вызова диалогового окна создается объект фрагмента CustomDialogFragment, затем у него вызывается метод show(). В этот метод передается менеджер фрагментов FragmentManager и строка (произвольный тег).

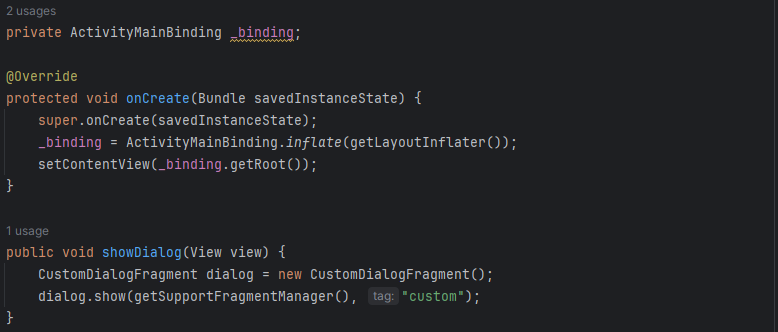


Рисунок 3.3 – Содержимое файла MainActivity.java

Запустим наше приложение чтобы убедиться в его работоспособности. Приложение, запущенное на эмуляторе предоставлено на рисунке 3.4.



Рисунок 3.4 – Приложение запущенное на эмуляторе

Часть 2

Кастомизируем диалоговое окно, для этого отредактируйте код метода onCreateDialog() в классе фрагмента рисунок 3.5.

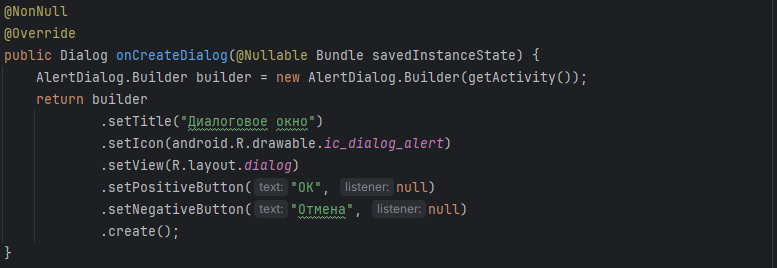


Рисунок 3.5 – Изменение метода в файле CustomDialogFragment.java

Метод setView() устанавливает в качестве интерфейса окна новый layout-ресурс dialog.xml. При этом кнопки и заголовок с иконкой не входят в разметку. Добавим в папку res/layout новый файл dialog.xml и сформируем его содержимое в соответствии с листингом 3.6.



Рисунок 3.6 – Содержимое файла dialog.xml

Запустим наше приложение чтобы убедиться в его работоспособности. Приложение, запущенное на эмуляторе предоставлено на рисунке 3.7.

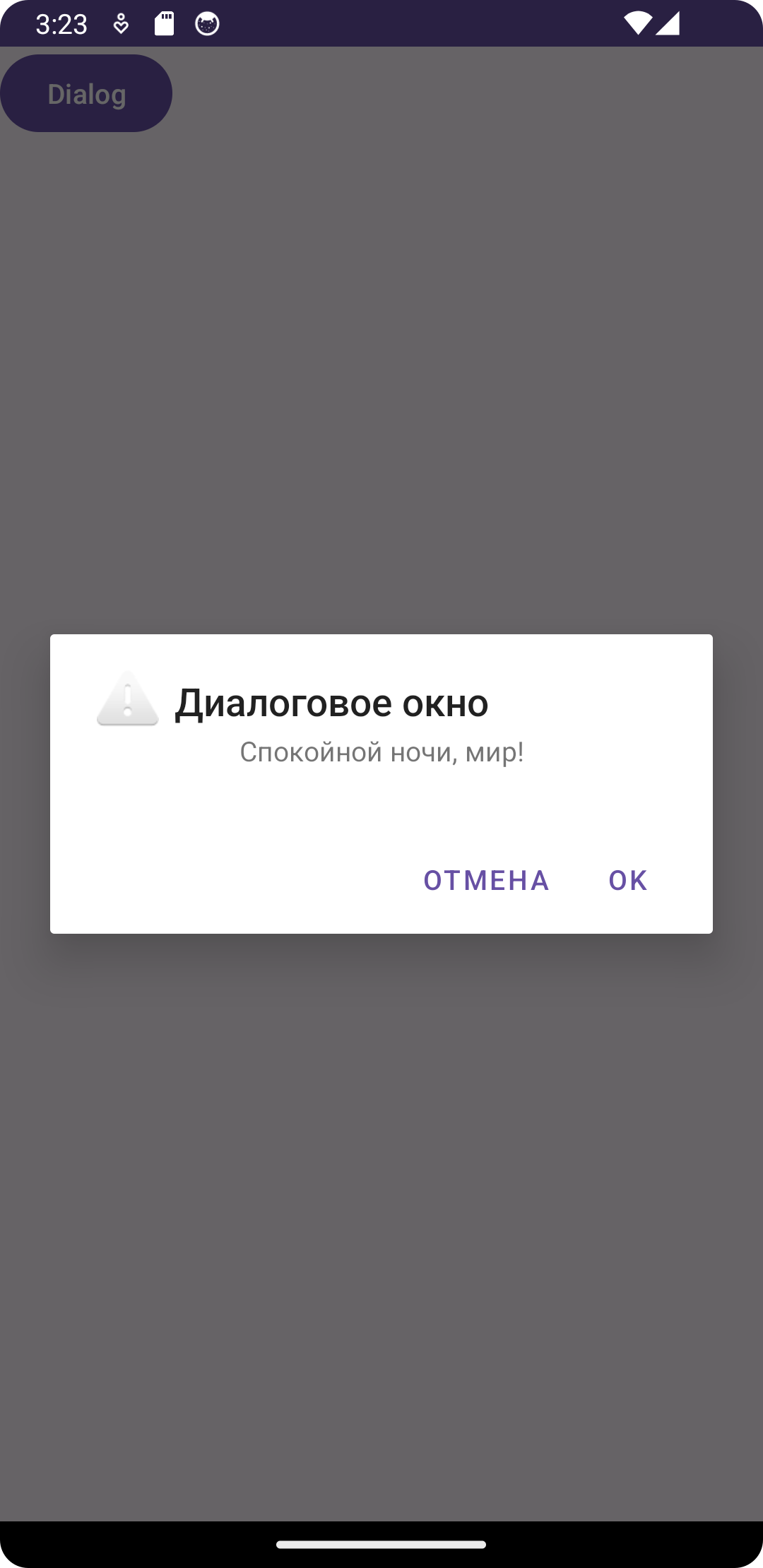


Рисунок 3.7 – Приложение запущенное на эмуляторе

1. **ВЗАИМОДЕЙСТВИЕ С ДИАЛОГОВЫМ ОКНОМ**

**Передача данных в вызываемое окно**

*Задание 4*

Определим в файле activity\_main.xml список ListView в соответствии с рисунком 4.1.

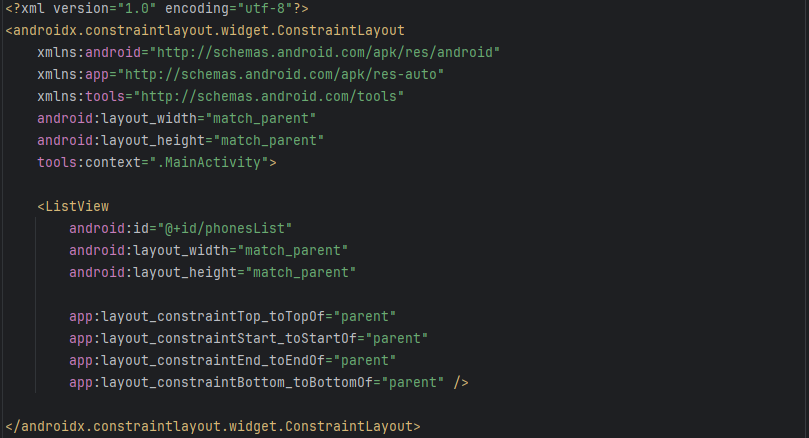


Рисунок 4.1 – Содержимое файла activity\_main.xml

В классе MainActivity определим для этого списка данные рисунок 4.2.

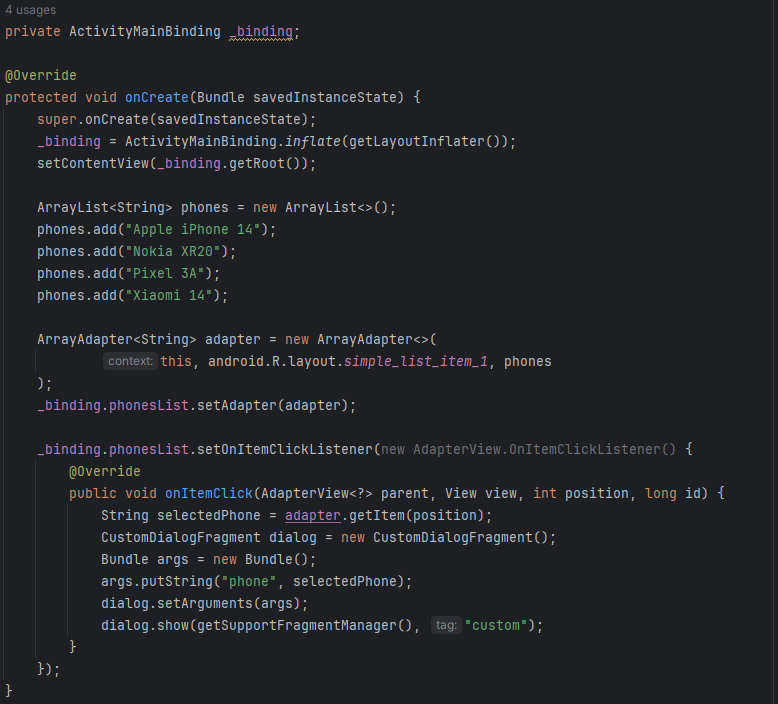


Рисунок 4.2 – Содержимое файла MainActivity.java

Определим класс фрагмента CustomDialogFragment в соответствии с рисунком 4.3.

В этом классе с помощью метода getArguments() получаем переданный в MainActivity объект Bundle. И так как была передана строка, то для её извлечения применяется метод getString()

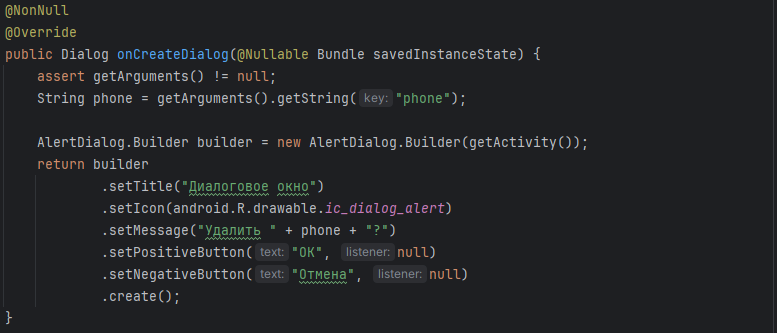


Рисунок 4.3 – Содержимое файла CustomDialogFragment.java

Запустим наше приложение чтобы убедиться в его работоспособности. Приложение, запущенное на эмуляторе предоставлено на рисунке 4.4.

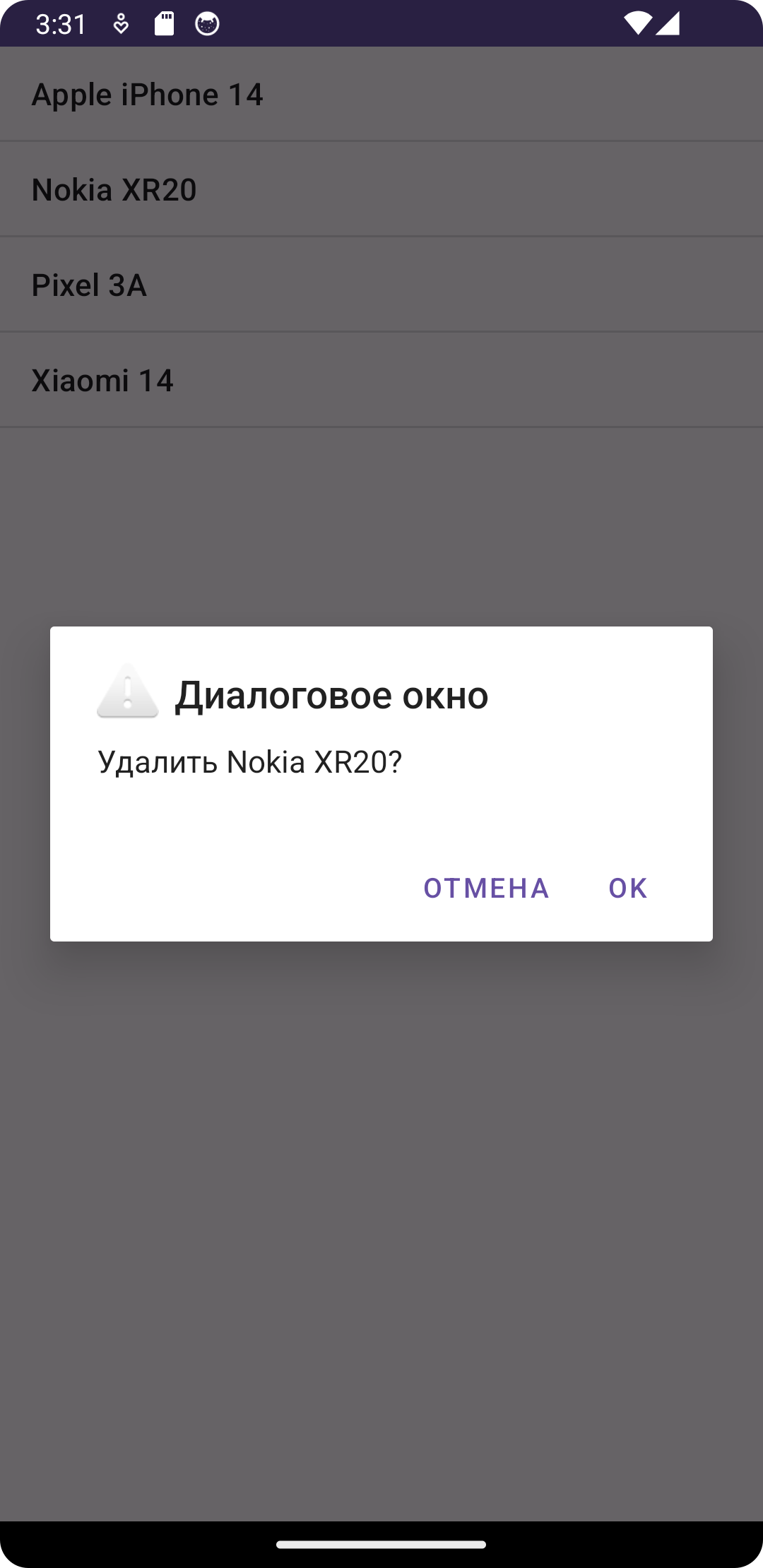


Рисунок 4.4 – Приложение запущенное на эмуляторе

**Взаимодействие диалогового окна с Activity**

*Задание 5*

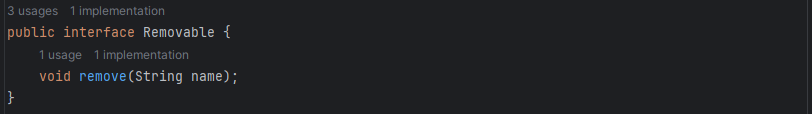
Реализуем в проекте из предыдущего примера новый интерфейс Removable в соответствии с рисунком 5.1, для этого при добавлении нового Java-класса в диалоговом окне выберите Interface.

Рисунок 5.1 – Содержимое файла Removable.java

Далее реализуем этот интерфейс в коде MainActivity, отредактировав его в соответствии с рисунком 5.2.



Рисунок 5.2 – Содержимое файла MainActivity.java

Далее добавим в определение фрагмента CustomDialogFragment логику взаимодействия диалогового окна с Activity в соответствии с рисунком 5.3.



Рисунок 5.3 – Содержимое файла CustomDialogFragment.java

Запустим наше приложение чтобы убедиться в его работоспособности. Приложение, запущенное на эмуляторе предоставлено на рисунке 5.4.

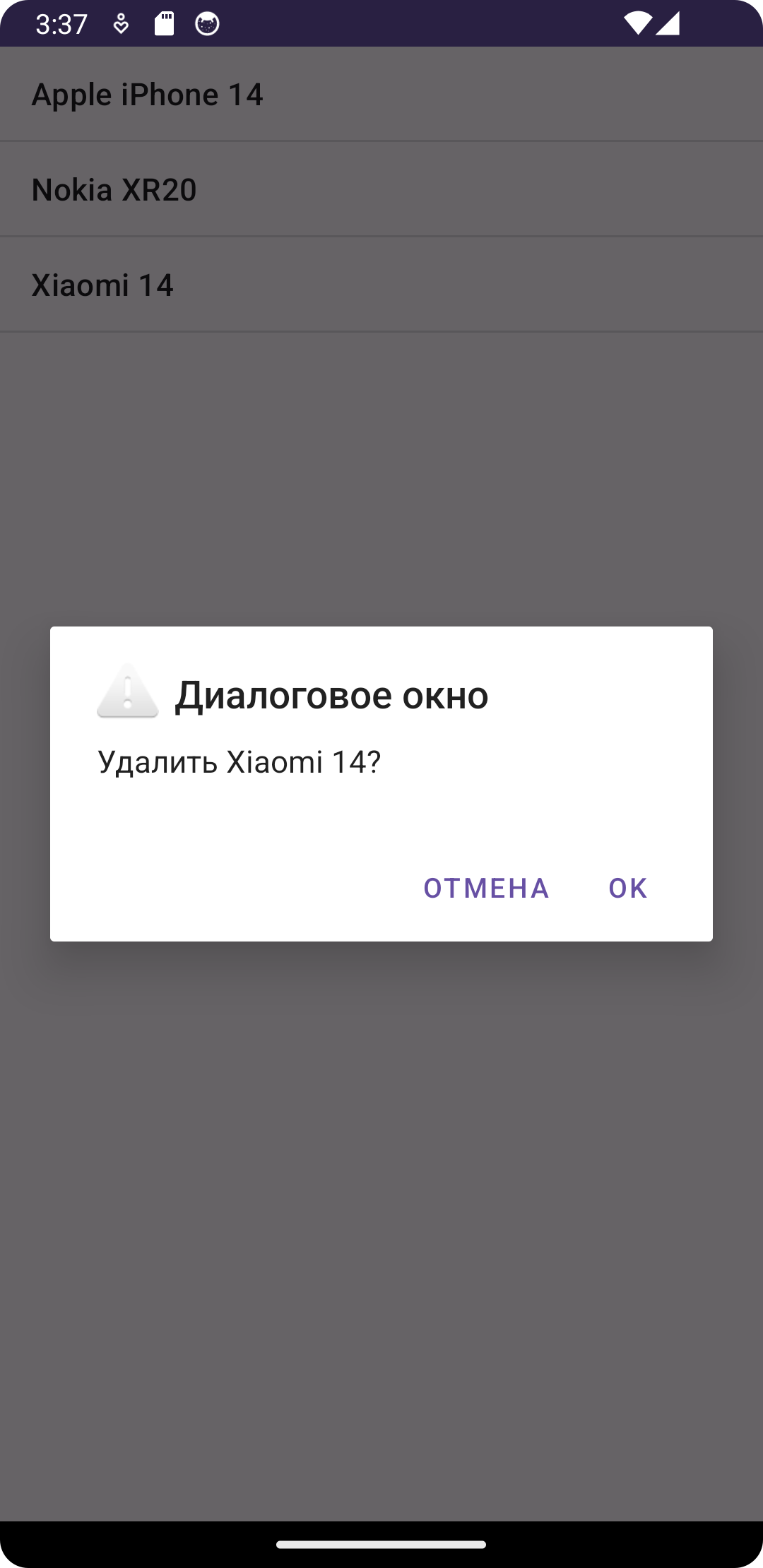


Рисунок 5.4 – Приложение запущенное на эмуляторе

1. **ПРАКТИЧЕСКИЕ ЗАДАНИЯ**

*Задание 6*

Реализуем переключатель для включения звукового сопровождения на Activity рисунок 6.1

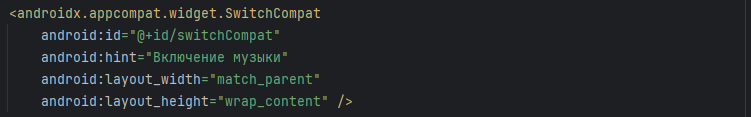


Рисунок 6.1 – Код реализующий переключатель

Реализуем код, который будет включать музыку и воспроизводить ее при запуске Activity. Также реализуем код, который будет отвечать за переключатель рисунок 6.2.

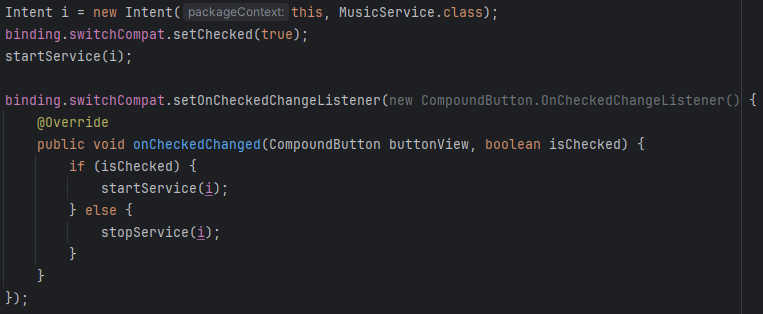


Рисунок 6.2 – Код отвечающий за музыкальное сопровождение

Добавим класс сервиса и назовем его MediaService. Для воспроизведения аудиофайла определим код в данном классе, предоставленном на рисунке 6.3.

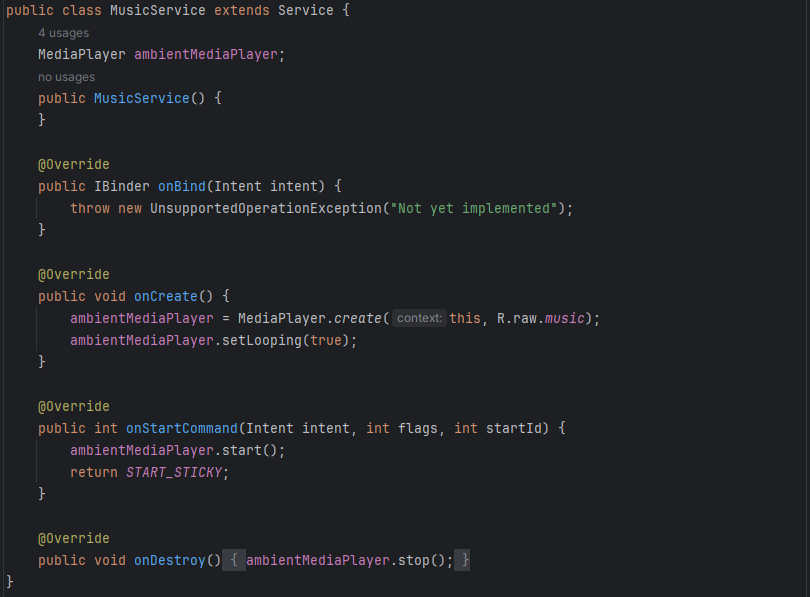


Рисунок 6.3 – Содержимое файла MusicService.java

Запустим наше приложение чтобы убедиться в его работоспособности. Приложение, запущенное на эмуляторе предоставлено на рисунке 6.4.

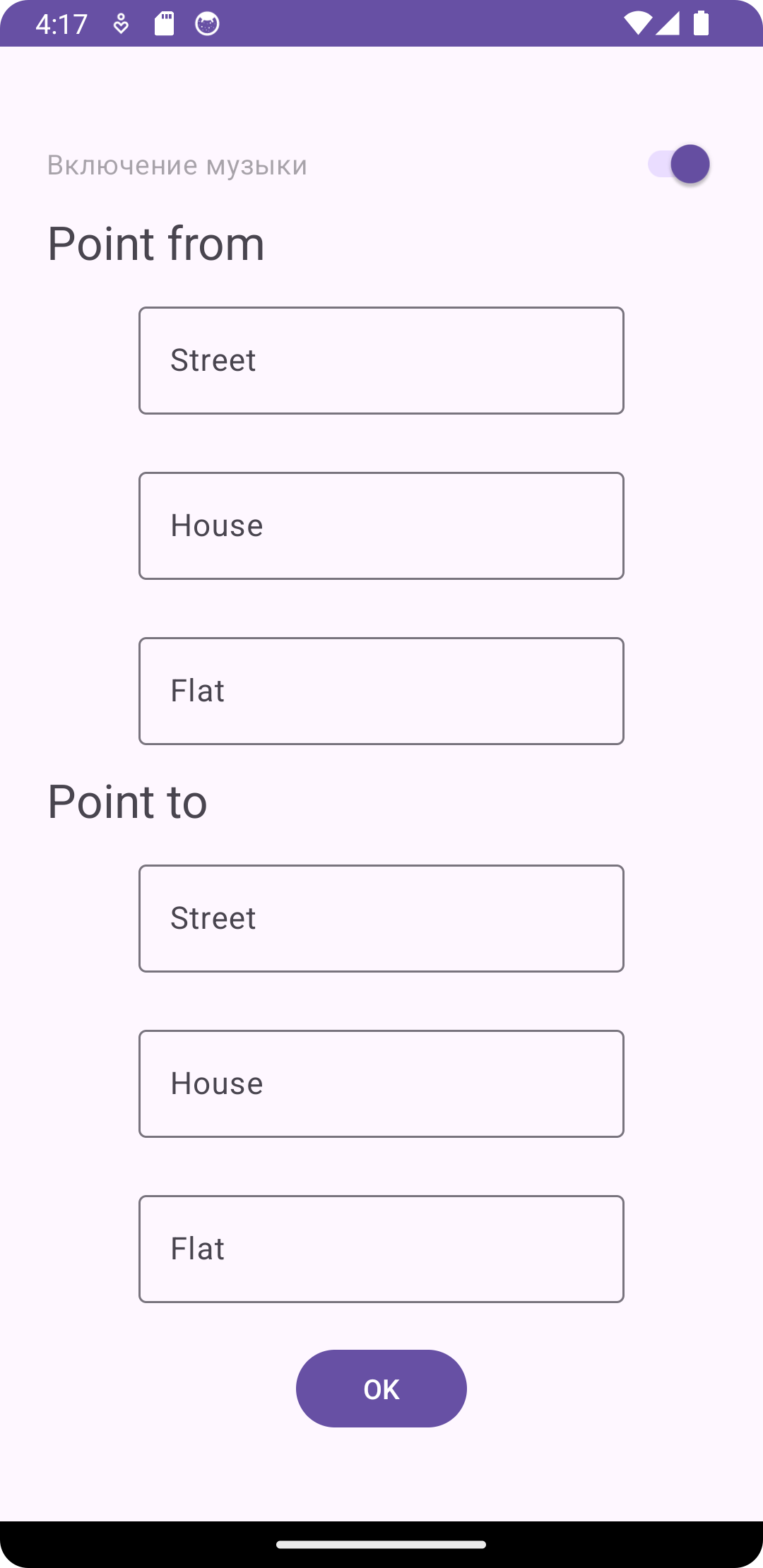


Рисунок 6.4 – Приложение запущенное на эмуляторе

*Задание 7*

*Задание 8*

*Задание 9*