МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Кафедра информационных технологий**

**КУРСОВАЯ РАБОТА**

**ANDROID-ПРИЛОЖЕНИЕ ДЛЯ ВОСПРОИЗВЕДЕНИЯ ЗВУКА**

Работу выполнила \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Глазова К.С.  
 (подпись, дата)

Факультет Компьютерных технологий и прикладной математики, курс 3\_

Специальность/направление 02.03.02. «Фундаментальная математика и информационные технологии»

Научный руководитель,  
к.т.н, доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Т.А. Приходько  
 (подпись, дата)

Нормоконтролёр,  
к.физ-мат. н, доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е.В. Кособуцкая  
 (подпись, дата)

Краснодар 2016

СОДЕРЖАНИЕ

ВВЕДЕНИЕ ……………………………………………………………………… 3

1 Постановка задачи…………………………………………..………………..... 4

2 Анализ аналогичных программ………………………..…………………….... 5

3 Разработка программы …………………………………..……………………. 6

3.1 Работа с MediaPlayer ………………………………....….……………...… 7

3.1.1 Реализация MediaPlayer в программе ……………...……....……....... 10

3.2 AudioManager и аудиофокус ..…………………………….……………... 19

3.3 Загрузка файлов на устройство .………….………………...…………… 20

3.3.1 Реализация DownloadManager в программе ……….…….……...… 21

ЗАКЛЮЧЕНИЕ ……………………………………………….…………..…... 25

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ ………………..……..… 26

ВВЕДЕНИЕ

В наше время мобильные устройства развились до такой степени, что при выполнении большинства повседневных задач почти вытеснили компьютеры. А причина этому – компактность и собственно мобильность. Несмотря на то, что смартфоны до сих пор уступают компьютерам в мощности, их ресурсов все равно хватает на качественное воспроизведение аудио и видео файлов, обмен данными, запуск небольших по ресурсозатратности программ, работу с графикой и т.д. В основе всех мобильных устройств, как и компьютеров, лежит операционная система. Лидирующие позиции на сегодняшний день занимают платформы Android, iOS и WindowsPhone. Потребности пользователей растут с каждым днем, что побуждает программистов совершенствовать уже существующие программы или придумывать что-то новое, тем самым упрощая жизнь пользователей.

1 Постановка задачи

Существует множество приложений для воспроизведения аудиокниг в ОС Android, как и для воспроизведения музыки. Основные требования к подобным приложениям: учитывать порядок воспроизведения файлов, сохранение текущего времени воспроизведения после завершения работы программы, возможность ставить воспроизведение на паузу, перематывать вперед и назад. Большинство таких приложений используют воспроизведение on-line через интернет, что не всегда удобно для пользователей ввиду ограниченности интернет-трафика. Некоторые плееры аудиокниг позволяют скачать книгу на устройство заранее, а потом проигрывать в off-line режиме, но в этой системе есть один большой минус: скачивание аудиокниги происходит одним большим файлом, в то время как звукозаписывающие студии во время записи книг часто разделяют их на несколько файлов. Большой файл будет загружаться в разы дольше и потребует больше интернет-трафика.

Задача данной курсовой – разработка приложения для ОС Android, позволяющее воспроизводить аудиокниги в режиме on-line через интернет или в режиме off-line с мобильного устройства, предварительно скачав его. Отличие данного приложения от других подобных в том, что пользователь сможет скачивать отдельные файлы аудиокниги (если такое возможно).

2 Анализ аналогичных программ

Будем производить анализ на конкретных примерах:

Smart Audio Book [3] – данное приложение позволяет воспроизводить аудиокниги только в режиме off-line. Пользователь сам указывает путь к папке, из которой приложение будет воспроизводить файлы. Программа отличается от остальных большим набором функций: таймер выключения, таймер бездействия (воспроизведение останавливается, если устройство не трогали в течение определенного времени), возможность создавать закладки и главный плюс - запоминание места, на котором остановилось воспроизведение, чтобы при повторном запуске продолжить воспроизведение с этого места. Но у программы есть и минусы: отсутствует своя база книг, из-за чего нет возможности загрузить их через интернет, а также нет воспроизведения on-line.

Патефон [3] – приложение, полностью противоположное предыдущему. Есть своя база книг, реализована возможность скачивания файла на устройство и on-line воспроизведение. Возможности плеера ограничены, но также сохраняется время воспроизведения и присутствует возможность создавать закладки. Большим минусом загрузки на устройство является то, что книга скачивается одним большим файлом. Кроме того, нет возможности использовать приложение как плеер своих файлов с устройства.

Слушай LitRes [3] – также, как и у предыдущего приложения, имеется своя база книг и возможность on-line прослушивания, плеер с точно таким же функционалом. Но при этом программа не имеет функции загрузки файлов на устройство, из-за чего использование интернета становится обязательным.

3 Разработка программы

В ОС Android существует несколько классов для работы со звуком:

* SoundPool [1] – позволяет воспроизводить звук, управлять громкостью, скоростью и потоком воспроизведения; этот класс поддерживает одновременное воспроизведение нескольких аудиопотоков; SoundPool в основном подходит для озвучки нажатия кнопок или различных событий в играх: выстрелов, реплик, оповещений;
* AudioManager [1] – предназначен для управления громкостью звука, аудиопотоком, аудиофокусом и устройствами воспроизведения и записи звука (Bluetooth-гарнитурами, микрофонами, наушниками и т.д);
* MediaPlayer [1] - воспроизводит и управляет воспроизведением звуковых файлов из различных источников: интернета, устройства, APK-файла.

Для разработки программы я буду использовать классы MediaPlayer и AudioManager. Класс SoundPool не подходит для выполнения данной задачи.

3.1 Работа с MediaPlayer

Класс MediaPlayer [1] работает на основе состояний, которые управляются с помощью внутреннего конечного автомата – машины состояний. На рисунке 1 показано изменение состояния объекта MediaPlayer для элемента управления воспроизведением. Одиночная стрелка означает синхронные вызовы метода, а пунктирная - асинхронные и обратные вызовы.

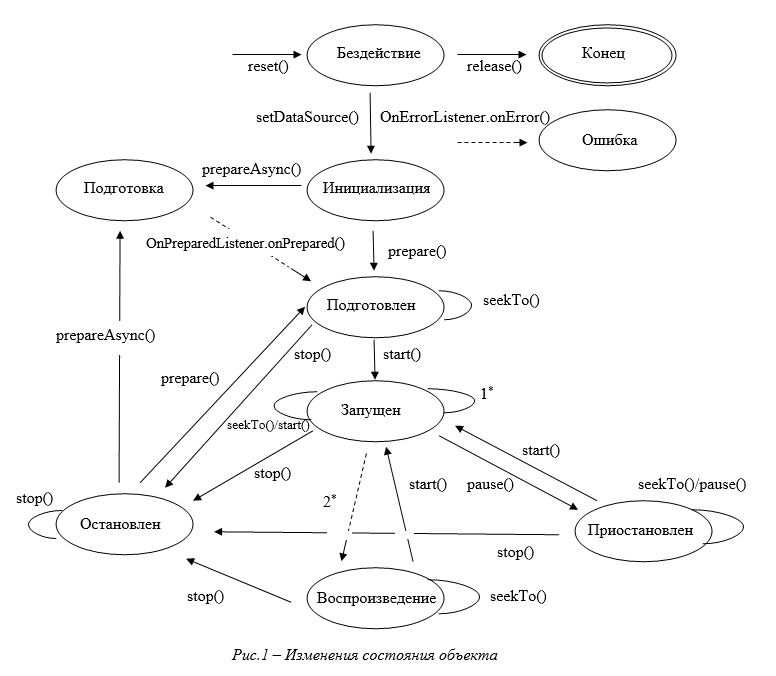


Рис.1 Жизненный цикл объекта MediaPlayer

1\* - Looping == true && воспроизведение завершается  
 2\* - Looping == false && Completion() вызывается для OnCompletionListener.

Жизненный цикл объекта MediaPlayer содержит несколько состояний. При создании нового объекта или вызове метода reset() MediaPlayer переходит в состояние бездействия, в котором воспроизведение невозможно.   
 Чтобы начать воспроизведение нужно пройти инициализацию, а для этого необходимо указать допустимый источник мультимедиа, вызвав метод setDataSourse().

Затем приложение вызывает метод prepare() или prepareAsync() для перехода в состояние готовности. Оба этих метода отвечают за получение данных, их буферизацию и декодирование файла, но метод prepare() значительно уступает методу prepareAsync() в скорости, из-за чего лучше использовать метод prepareAsync(), который выполняется в фоновом режиме и возвращается сразу же после завершения, отправляя обратный вызов OnPreparedListener.onPrepared() и приводит объект в состояние готовности.

После прохождения инициализации и проверки на готовность можно начинать воспроизведение и управлять им с помощью команд start() (начало воспроизведения) и seekTo() (перемотать воспроизведение на N миллисекунд вперед или назад).

После начала воспроизведения объект переходит в состояние «запущен». При желании можно поставить на паузу воспроизведение, вызвав метод pause(), после чего объект MediaPlayer перейдет в состояние «пауза», из которого можно выбраться, снова вызвав метод start(), что возобновит воспроизведение и переведет объект в состояние «запущен» или вызвать метод stop(), перейдя в состояние «остановлен». Также в состоянии паузы можно вызывать метод seekTo() для перемотки.

В случае завершения воспроизведения файла, можно возобновить воспроизведение, вызвав метод start() (в этом случае состояние изменится на «запущен») или же объект перейдет в состояние «остановлен», а затем в состояние «конец», с помощью метода release().

Обработчик OnErrorListener, метод обратного вызова OnErrorListener.onError() будет вызван при любых ошибках, чтобы можно было их обрабатывать надлежащим образом.

Класс MediaPlayer поддерживает воспроизведение медиафайлов форматов: WAV, AAC, MP3, WMA, AMR, OGG, MIDI. Чаще всего для аудиозаписей используется формат MP3.

3.1.1 Реализация MediaPlayer в программе

Функция, обеспечивающая воспроизведение в off-line режиме с устройства:

public void playClick(View view) {

if (!check\_play) {

mediaPlayer=MediaPlayer.create(this, Uri.parse(Environment.getExternalStorageDirectory().getPath()+ "/Music/2.mp3"));

mediaPlayer.setAudioStreamType(AudioManager.STREAM\_MUSIC);

mediaPlayer.start();

check\_play = true;

}

else {

if (mediaPlayer != null) {

if (mediaPlayer.isPlaying()) {

mediaPlayer.pause();

} else {

mediaPlayer.start();}

}

}

if (mediaPlayer != null) {

finishTime = mediaPlayer.getDuration();

startTime = mediaPlayer.getCurrentPosition();

if (oneTimeOnly == 0) {

seekbarOne.setMax((int) finishTime);

oneTimeOnly = 1;

}

timeText2.setText(String.format("%d:%d",

TimeUnit.MILLISECONDS.toMinutes((long) finishTime),

TimeUnit.MILLISECONDS.toSeconds((long) finishTime) - TimeUnit.MINUTES.toSeconds(TimeUnit.MILLISECONDS.toMinutes((long)

finishTime)))

);

timeText1.setText(String.format("%d:%d",

TimeUnit.MILLISECONDS.toMinutes((long) startTime),

TimeUnit.MILLISECONDS.toSeconds((long) startTime) -

TimeUnit.MINUTES.toSeconds(TimeUnit.MILLISECONDS.toMinutes((long)

startTime)))

);

seekbarOne.setProgress((int) startTime);

hand.postDelayed(songTime, 100);

}

}

@Override

protected void onDestroy() {

super.onDestroy();

check\_play = false;

}

Для воспроизведения функция использует файл, путь к которому указывается в программе. Класс AudioManager используется для создания аудиопотока. При первом нажатии на кнопку воспроизведения функция производит инициализацию, а при повторном нажатии останавливает или возобновляет воспроизведение, не проводя повторную инициализацию. Кроме того, функция следит за временем воспроизведения текущего файла, что позволяет пользователю видеть сколько всего времени понадобится для воспроизведения файла и сколько уже было воспроизведено.

Также за отображение шкалы времени используется функция:

private Runnable songTime = new Runnable() {

public void run() {

nowTime = mediaPlayer.getCurrentPosition();

timeText1.setText(String.format("%d:%d",

TimeUnit.MILLISECONDS.toMinutes((long) nowTime),

TimeUnit.MILLISECONDS.toSeconds((long) nowTime) -

TimeUnit.MINUTES.toSeconds(TimeUnit.MILLISECONDS.

toMinutes((long) nowTime)))

);

seekbarOne.setProgress((int) nowTime);

hand.postDelayed(this, 100);

}

};

Следующие функции добавляют необходимый функционал проигрывателю – перемотку воспроизведения: на 10 секунд вперед и назад и на 1 минуту вперед и назад. Пример перемотки на 10 секунд:

private double nowTime = 0;

private double startTime = 0;

private double finishTime = 0;

private double tenTime = 10000;

public void rightTenClick(View view) {

nowTime = mediaPlayer.getCurrentPosition();

finishTime = mediaPlayer.getDuration();

int temp=(int) nowTime;

if((temp+ tenTime)<=finishTime) {

nowTime += tenTime;

mediaPlayer.seekTo((int) nowTime);

}

else

{

mediaPlayer.seekTo((int) finishTime-5000);

}

}

public void leftTenClick(View view){

nowTime = mediaPlayer.getCurrentPosition();

finishTime=mediaPlayer.getDuration();

int temp = (int) nowTime;

if((temp - tenTime)>0)

{

nowTime -= tenTime;

mediaPlayer.seekTo((int) nowTime);

}

else

{

mediaPlayer.seekTo((int) startTime);

}

}

Функции предусматривают ошибки, которые могут быть вызваны перемоткой вперед или назад на время, большее, чем 0 секунд или конечное время воспроизведения. В ситуации перемотки назад время будет устанавливаться на 0, а для перемотки вперед – за 5 секунд до окончания воспроизведения файла.

Функция on-line воспроизведения:

public void playClick(View view) {

if (!check\_play) {

releaseMP();

mediaPlayer = new MediaPlayer();

mediaPlayer.setAudioStreamType(AudioManager.STREAM\_MUSIC);

try {

mediaPlayer.setDataSource(url);

mediaPlayer.setOnPreparedListener(this);

mediaPlayer.prepareAsync();

check\_play = true;

} catch (IOException e) {

e.printStackTrace();

}

}

else {

if (mediaPlayer != null) {

if (mediaPlayer.isPlaying()) {

mediaPlayer.pause();

} else {

mediaPlayer.start();

}

}

}

if (mediaPlayer != null) {

finishTime = mediaPlayer.getDuration();

startTime = mediaPlayer.getCurrentPosition();

if (oneTimeOnly == 0) {

seekbarOne.setMax((int) finishTime);

oneTimeOnly = 1;

}

timeText2.setText(String.format("%d:%d",

TimeUnit.MILLISECONDS.toMinutes((long) finishTime),

TimeUnit.MILLISECONDS.toSeconds((long) finishTime) -

TimeUnit.MINUTES.toSeconds(TimeUnit.MILLISECONDS.toMinutes((long)

finishTime)))

);

timeText1.setText(String.format("%d:%d",

TimeUnit.MILLISECONDS.toMinutes((long) startTime),

TimeUnit.MILLISECONDS.toSeconds((long) startTime) -

TimeUnit.MINUTES.toSeconds(TimeUnit.MILLISECONDS.toMinutes((long)

startTime)))

);

seekbarOne.setProgress((int) startTime);

hand.postDelayed(songTime, 100);

}

}

public void onBackPressed()

{

mediaPlayer.stop();

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

startActivity(intent);

}

private void releaseMP() {

if (mediaPlayer != null) {

try {

mediaPlayer.release();

mediaPlayer = null;

} catch (Exception e) {

e.printStackTrace();

}

}

}

public void onPrepared(MediaPlayer mp) {

mp.start();

}

@Override

protected void onDestroy() {

super.onDestroy();

check\_play = false;

releaseMP();

}

Функция воспроизводит файл по ссылке, указанной в коде в строке «url». Сам файл хранится в интернет-хранилище. Аналогично с функцией off-line воспроизведения, первое нажатие на кнопку воспроизведения запускает процедуру инициализации и проверки на готовность, а повторное возобновляет или останавливает воспроизведение. Для подготовки к воспроизведению используется метод prepareAsync() вместо prepare(), так как он позволяет быстрее перейти к воспроизведению, не дожидаясь полной загрузки файла.

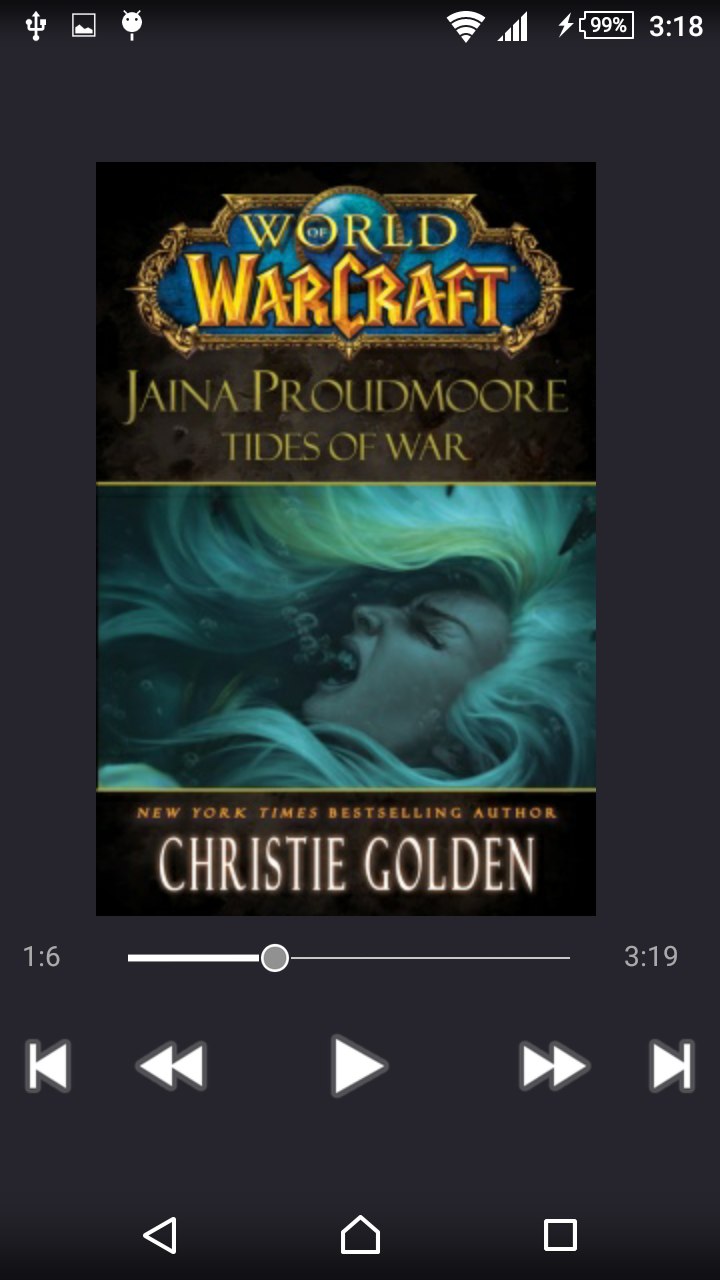


Рис. 2 – Внешний вид плеера

3.2 AudioManager и аудиофокус

Аудиофокус [5] позволяет задавать аудиопотокам приоритет и, в зависимости от этого приоритета, управлять громкостью или же вообще останавливать воспроизведение определенных звуков.

Фокус запрашивается с помощью метода requestAudioFocus. Типы фокусов:

* AUDIOFOCUS\_GAIN – долгое воспроизведение звука. Приложение должно остановить воспроизведение.
* AUDIOFOCUS\_GAIN\_TRANSIENT – среднее по длине воспроизведение. Приложение должно приостановить не на долго свое воспроизведение.
* AUDIOFOCUS\_GAIN\_TRANSIENT\_MAY\_DUCK – короткое воспроизведение звука. Приложение может не останавливать воспроизведение, достаточно будет уменьшить громкость.

В данном приложении аудиофокус может быть реализован примерно таким образом: потоку аудиокниги выдается высокий приоритет, если во время воспроизведения приходит оповещение от другого приложения, то приоритет плеера понижается относительно приоритета оповещения. Программа это видит и останавливает воспроизведение до тех пор, пока снова не получит приоритет. Уменьшение громкости воспроизведения во время коротких оповещений не подходит для данной задачи, так как пользователь может не расслышать слова из книги.

3.3 Загрузка файлов на устройство

Существует множество классов для загрузки данных через интернет в ОС Android. Вот некоторые из них:

* Thread [4] – подходит для операций, требующих установки приоритета выполнения или для операций, активно расходующих ресурсы CPU. Позволяет выполнять одну и ту же операцию множество раз, через какой-либо интервал времени, а также выполнять несколько фоновых потоков параллельно.
* AsyncTask [4] – подходит для простых операций с файловой системой, загрузки небольшого количества данных и для операций, на выполнение которых нужно потратить несколько секунд. В отличие от предыдущего класса, в AsyncTask невозможно задать приоритет потоку и для всех операций экземпляров класса отводится только один поток.
* Service [4] – не является отдельным процессом или потоком, но имеет свой жизненный цикл. Жизненный цикл сервисов, как правило, дольше, чем Activity. Стартовав однажды, сервис будет жив, пока у него не закончится работа, после чего он самостоятельно остановится. Этот класс подходит для выполнения длинных по времени операций.
* DownloadManager [4] - удобно использовать для загрузки больших файлов, которые могут представлять интерес пользователю отдельно от приложения. Он возьмет на себя установление соединения, реагирование на ошибки, возобновление загрузки, создание уведомлений в Notification bar и саму загрузку файлов в фоновом потоке.

В данном приложении я буду использовать DownloadManager.

3.3.1 Реализация DownloadManager в программе

Для загрузки с помощью DownloadManager достаточно указать ссылку на скачивание файла, путь к папке, в которую будет идти загрузка, тип файла и его будущее имя. Также необходимо реализовать методы остановки и возобновления загрузки. При желании можно создать название и комментарий для отображения загрузки в окне уведомлений.

public class downloadActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_download);

Boolean down\_check = getIntent().getExtras().getBoolean("Boolean");

if(down\_check) {

downloadManager = (DownloadManager) getSystemService(DOWNLOAD\_SERVICE);

Request request = new Request(Uri.parse(Song\_URL));

request.setTitle("Война и мир");

request.setDescription("3");

request.setMimeType("application/mp3");

request.setDestinationInExternalPublicDir(Environment.DIRECTORY\_DOWNLOADS, "3.mp3");

request.setNotificationVisibility(DownloadManager.Request.VISIBILITY\_VISIBLE\_NOTIFY\_COMPLETED);

downloadManager.enqueue(request);

}

else {

Intent intent2 = new Intent(this, OfPlayerActivity.class);

intent2.putExtra("Boolean", false);

startActivity(intent2);

}

}

private static final String Song\_URL = "https://content.screencast.com/users/glazowa.karina/folders/Default/media/13ecec3e-4678-4f84-92a2-fd9746556181/3.mp3";

DownloadManager downloadManager;

@Override

protected void onResume() {

super.onResume();

registerReceiver(receiver, new IntentFilter(DownloadManager.ACTION\_DOWNLOAD\_COMPLETE));

registerReceiver(receiver, new IntentFilter(DownloadManager.ACTION\_NOTIFICATION\_CLICKED));

@Override

protected void onPause() {

super.onPause();

unregisterReceiver(receiver);

}

BroadcastReceiver receiver = new BroadcastReceiver() {

@Override

public void onReceive(Context context, Intent intent) {

String action = intent.getAction();

if (DownloadManager.ACTION\_DOWNLOAD\_COMPLETE.equals(action)){

long downloadId = intent.getLongExtra(DownloadManager.EXTRA\_DOWNLOAD\_ID, 0);

DownloadManager.Query query = new DownloadManager.Query();

query.setFilterById(downloadId);

} else if (DownloadManager.ACTION\_NOTIFICATION\_CLICKED.equals(action)){

long[] ids = intent.getLongArrayExtra(DownloadManager.EXTRA\_NOTIFICATION\_CLICK\_DOWNLOAD\_IDS);

DownloadManager.Query query = new DownloadManager.Query();

query.setFilterById(ids);

}

} };

Файл загружается в папку Downloads в корне устройства, откуда потом и будет воспроизводиться. Тип файла указывается в программе как mp3-файл. Весь процесс загрузки пользователь может наблюдать в строке уведомлений Android. После завершения загрузки пользователь должен нажать кнопку «Play», чтобы перейти на экран плеера и начать воспроизведение. Воспроизведение начнется сразу. если файл уже был загружен на устройство.

ЗАКЛЮЧЕНИЕ

На данном этапе разработки в приложении реализовано меню выбора книг и отдельных файлов, on-line и off-line воспроизведение, а также скачивание файлов на устройство. У плеера имеются функции паузы и два вида перемотки. Также есть полоса загрузки, позволяющая следить за временем воспроизведения.

В процессе разработки были проанализированы способы решения аналогичных задач и изучены…

Все поставленные задачи выполнены и основные функции приложение уже способно выполнять. В дальнейшем планируется усовершенствование программы. Например, приложение должно предупреждать о размере скачиваемого файла, что позволит пользователю самостоятельно контролировать расход интернет-трафика. Кроме того, приложение будет проверять сеть на устойчивость и, если результат неудовлетворительный, предлагать пользователю воспроизвести другие файлы с устройства или подождать пока загрузится часть текущего файла. Также планируется провести сравнительную оценку авторского приложения с другими на основе критериев: скорость скачивания, устойчивость и удобство работы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. URL: https://developer.android.com/ (дата обращения: 01.12.2016).
2. Давыдов С.В., Ефимов А.А. IntelliJ IDEA. Профессиональное программирование на Java. – СПб.: БХВ-Петербург, 2005. – 800с.: ил.
3. URL: https://play.google.com/ (дата обращения: 23.11.2016).
4. Отзывчивое Android-приложение или 1001 способ загрузить картинку URL: https://habrahabr.ru/company/eastbanctech/blog/192998/ (дата обращения: 17.12.2016).
5. Audio Focus URL: https://startandroid.ru/ru/uroki/vse-uroki-spiskom/240-urok-128-media-audio-focus.html (дата обращения: 17.12.2016).