Министерство образования Новосибирской области ГБПОУ НСО «Новосибирский авиационный технический колледж имени Б.С. Галущака»

**РАЗРАБОТКА МОБИЛЬНОГО АУДИО-ПЛЕЕРА**

Пояснительная записка к курсовому проекту

ПМ.01 Разработка модулей программного обеспечения для компьютерных систем

МДК.01.03. Разработка мобильных приложений

НАТКиГ.211400.043.000ПЗ

Разработал:

студент группы ПР - 315

Малов Н.В.

2022

# СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc28775)

[1 Исследовательский раздел 5](#_Toc24743)

[1.1 Описание предметной области 5](#_Toc31018)

[1.2 Образ клиента 6](#_Toc22348)

[1.3 Сценарии 6](#_Toc23449)

[1.4 Сбор и анализ прототипов 8](#_Toc29063)

[2 ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ 14](#_Toc2665)

[2.1 UI/UX дизайн проекта 14](#_Toc9778)

[2.2 Выбор технологии, языка и среды программирования 19](#_Toc31801)

[3 РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ 20](#_Toc10224)

[3.1 Описание разработанных процедур и функций 20](#_Toc2974)

[4 ТЕСТИРОВАНИЕ 26](#_Toc26685)

[ЗАКЛЮЧЕНИЕ 28](#_Toc25418)

[СПИСОК ИСТОЧНИКОВ 30](#_Toc12436)

[ПРИЛОЖЕНИЕ А. Техническое задание 31](#_Toc31189)

# ВВЕДЕНИЕ

В наше время большой популярностью пользуются сервисы потоковой трансляции музыки. Основным их минусом является отсутствие возможности прослушивать музыку без соединения с интернетом. Также из минусов можно выделить сильную перегруженность приложения. Первый недостаток можно решить, используя локальный метод хранения данных. Современные устройства, зачастую, обладают большим объёмом хранилища, что позволяет использовать его, как среду для быстрого получения необходимых файлов без задержек сети. Второго недостатка можно избежать при помощи иного подхода к разработке пользовательского интерфейса без использования ненужных большинству пользователей функций и графических элементов на основе одного лишь экрана.

Целью курсового проекта является разработка мобильного приложения для проигрывания музыки, на основе хранимых в памяти устройства аудиофайлов.

Для достижения цели поставлены следующие задачи:

1. исследовать предметную область;
2. спроектировать макет приложения;
3. выбрать язык и среду программирования;
4. разработать приложение по макету;
5. протестировать полученный продукт.

# Объектом исследования является электронный способ воспроизведения аудиофайлов.

Предметом исследования является воспроизведения аудиофайлов.

Практическая значимость курсового проекта состоит в возможности решения вопросов хранения и фильтрации информации. Разработанная программа способствует сохранению важной для человека информации.

# Объектом исследования является электронный способ воспроизведения аудиофайлов.

Предметом исследования является воспроизведения аудиофайлов.

Практическая значимость курсового проекта состоит в возможности создания не перегруженного лишней информацией аудиоплеера. Разработанная программа способствует быстрому воспроизведению аудиофайлов без задержек сети.

# Исследовательский раздел

## Описание предметной области

В наше время большое количество людей по всему миру пользуются мобильными аудиоплеерами для прослушивания любимой музыки. По данным исследования FDFgroup 97% респондентов в возрасте от 16 до 25 лет слушают музыку каждый день. В среднем молодежь уделяет музыке около 4-х часов в день. Что делает эту область очень востребованной на текущий момент.

Большая часть опрошенных предпочитает слушать музыку на мобильных устройствах при помощи специальных приложений и сервисов. Основная часть использует для этого потоковые сервисы прослушивания музыки. Они имеют ряд преимуществ, например: быстрый доступ к своей музыке с условием соединения с сетью, поиск музыки в предоставленной сервисом библиотеке. В тот же момент люди забывают о том, что в некоторых сценариях пропадает возможность соединения с сетью, что приводит к потере доступа к своей музыке до восстановления соединения.

У подобных приложений и сервисов имеются аналоги, основанные на другом способе хранения информации. Такие аудио-плееры используют локальный способ хранения информации, что позволяет слушать музыку в любом месте прямо из памяти своего устройства. Большинство таких приложений имеют перегруженный интерфейс с большим количеством элементов и функций, усложняющих навигацию. На основе проведённого мной внутри группы опроса можно сделать вывод, что значительная часть пользователей приложений такого типа не использует многие из этих опций.

При помощи данного приложения пользователь сможет прослушивать свою любимую музыку в любом месте и без доступа к сети без сложностей в навигации внутри пользовательского интерфейса.

## Образ клиента

Данное приложение нацелено на аудиторию, интересующихся музыкой людей любого возраста. Основная часть аудитории – люди от 16 до 25 лет. Приложением могут пользоваться как женщины, так и мужчины.

Разработанное мобильное приложение предназначено для людей, которые хотят ежедневно слушать музыку в любом месте со своего смартфона.

## Сценарии

Евгений ездит в метро на работу каждый день и слушает музыку. Как только поезд въезжает в туннель после остановки на станции – сигнал теряется. Каждый раз, при потере соединения с сетью, проигрываемая песня начинает прерываться или произведение вовсе останавливается. Такого не случилось бы, если бы на устройстве Евгения было установлено приложение с локальным способом хранения файлов.

Елена отправляется в поход в горы, но по достижению определённой точки своего маршрута она обнаруживает, что её приложение для потокового воспроизведения музыки не воспроизводит её любимую песню по причине отсутствия соединения с сетью. Этого можно было бы избежать, предварительно установив на смартфон разработанный аудио-плеер.

Славе исполнилось восемь лет и ему хочется узнать название песни, которая сейчас играет на его телефоне, однако он не может понять, где именно можно посмотреть название песни, ведь на экране находится большое множество элементов управления и различного текста. Слава с лёгкостью разобрался бы с тем, какая пеня проигрывается на его телефоне, если бы у его приложения был простой и понятный интерфейс, не перегруженный лишними элементами, который был бы понятен даже ребёнку.

* 1. Сбор и анализ прототипов

Далее будут рассмотрены несколько приложений, предназначенных для тех же целей, что и разработанное. Для примера взяты «AIMP» и «Spotify». Рассмотрим каждое из них.

«AIMP» - бесплатный аудиопроигрыватель с закрытым исходным кодом, основанный на воспроизведении хранимых в памяти устройства аудиофайлов. Приложение имеет множество различных настроек как визуального интерфейса, так и звуковых. Список файлов выводится в лист с дополнительной информацией (Расширение файла, вес файла, режим воспроизведения, частота и др.)

## Сбор и анализ прототипов

Далее будут рассмотрены несколько приложений, предназначенных для тех же целей, что и разработанное. Для примера взяты «AIMP» и «Spotify». Рассмотрим каждое из них.

«AIMP» - бесплатный аудиопроигрыватель с закрытым исходным кодом, основанный на воспроизведении хранимых в памяти устройства аудиофайлов. Приложение имеет множество различных настроек как визуального интерфейса, так и звуковых функций. Список аудио-файлов выводится в лист с дополнительной информацией (Расширение файла, вес файла, режим воспроизведения, частота и др.).

Дизайн выполнен из сочетаний серых цветов с акцентом внимания на основе оранжевого цвета. Предлагается создание собственных альбомов на основе списка всех загруженных файлов. Сами файлы загружаются из выбранной пользователем папки. Выбор папки предлагается при первом запуске приложения.



Рисунок 1 – Список аудио-файлов в AIMP

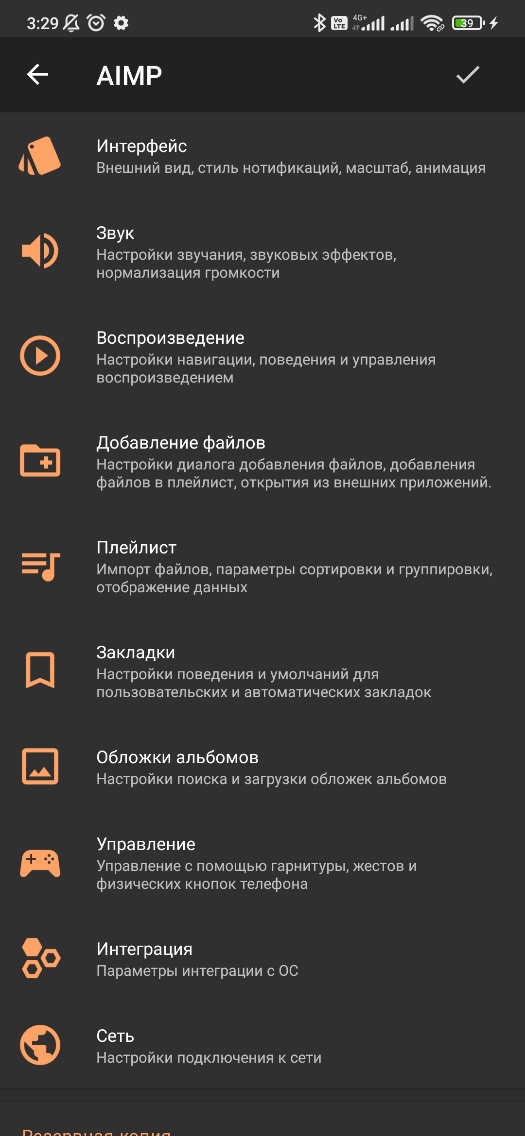


Рисунок 2 – Список настроек в AIMP

«Spotify» - это стриминговый сервис, позволяющий прослушивать музыкальные композиции, аудиокниги и подкасты, не скачивая их на устройство. Приложение имеет нижнюю навигационную панель с пунктами

«Просмотр», «Поиск», «Радио», «Ваша музыка».

Первый пункт отвечает за главную страницу приложения с отображением новинок известных и прослушиваемых исполнителей и различных новостей.

При выборе пункта «Поиск» пользователю будет доступен поиск композиций из облачной библиотеки с возможностью их прослушивания и добавления в свою коллекцию.

В третьем пункте предлагается радио, разделённое по категориям на жанры музыки. Пользователь может выбрать понравившийся жанр и приложение будет воспроизводить случайные композиции этого жанра.

В пункте «Моя музыка» отображаются композиции, добавленные пользователем. Здесь можно выполнить сортировку музыки по альбомам, прослушать выбранный трек, удалить его и добавить в специальный альбом «избранное».

Дизайн приложения выполнен в оттенках серого с акцентом внимания на основе мятно-зелёного цвета.

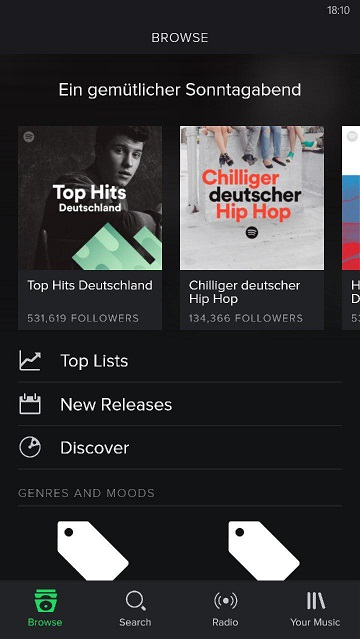


Рисунок 3 – Пункт «Просмотр» в Spotify

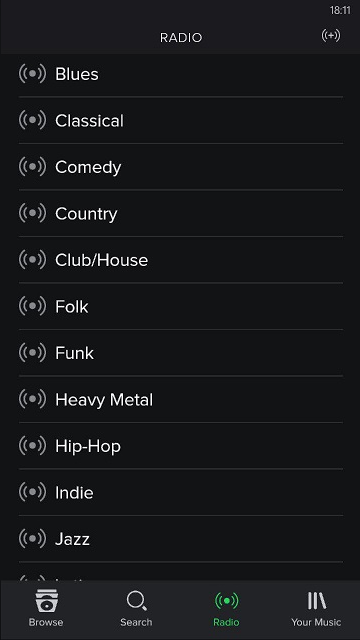


Рисунок 4 – Пункт «Радио» в Spotify

В таблице 1 представлено сравнение данных приложений по следующим критериям: возможность прослушивания музыки без соединения с интернетом, перегруженность интерфейса, возможность настройки звуковых эффектов, возможность добавления файлов с устройства, возможность создания альбомов, ограничение по количеству треков.

Таблица 1 – Сравнение прототипов

|  |  |  |
| --- | --- | --- |
| Название приложения | «AIMP» | «Spotify» |
| Возможность прослушивания музыки без соединения с интернетом | Имеется | Отсутствует |
| Перегруженность интерфейса | Перегружен | Не перегружен |
| Возможность настройки звуковых эффектов | Имеется | Отсутствует |
| Возможность добавления файлов с устройства | Имеется | Отсутствует |
| Возможность создания альбомов | Имеется | Имеется |
| Ограничение по количеству треков | Без ограничений | Ограничение без платной подписки |

Рассмотрев несколько приложений, выполняющих похожие задачи, было решено написать приложение, которое имело бы простой интерфейс без лишних функций и с акцентом внимания на важных для пользователя элементов, которое могло бы самостоятельно составлять список музыки из имеющихся на устройстве аудиофайлов.

# ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ

# UI/UX дизайн проекта

* + 1. Логотип и цветовая схема

Для проектирования приложения был выбран сервис для разработки различных дизайнов и интерфейсов «Figma».

Для мобильного приложения был разработан логотип, представленный на рисунке 5.



Рисунок 5 — Логотип

Определена цветовая схема приложения для основной темы, которая представлена на рисунке 6.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| # FF8A00 | # 454544 | # 5C5C5C | # DCDCDC | #FFFFFF | # B75656 |

Рисунок 6 — Цветовая схема приложения

* + 1. Загрузочный экран

Загрузочный экран содержит в себе логотип приложения (Рисунок 7). Экран необходим для скрытия различных процессов работы.



Рисунок 7 — Загрузочный экран

* + 1. Основной экран

Основной экран содержит список музыки, добавленной при запуске приложения и проверке накопителя (Рисунок 8). Для запуска воспроизведения достаточно нажать на желаемый, после чего изображение его альбома, название трека и название группы отобразятся в панели управления воспроизведением, находящейся в самом низу экрана. При длительном нажатии на трек запустится режим удаления файлов, строка выделится цветом и соответствующий трек пометится на удаление. При этом появится специальная панель с информацией об удалении. Таким образом можно выбрать несколько треков и удалить их с накопителя устройства прямо из приложения (Рисунок 9). При нажатии на пиктограмму меню на верхней панели появится выпадающий список состоящий из пунктов «Android-эквалайзер» и «Об авторе» (Рисунок 10). При выборе первого пункта произойдёт переход на экран «Эквалайзер», при выборе первого отобразится всплывающее оповещение с информацией о разработчике (В моём случае это «Малов Н.В. ПР-315»).

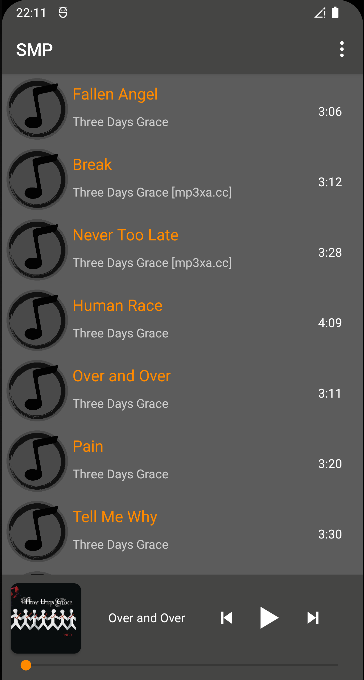


Рисунок 8 — Главный экран

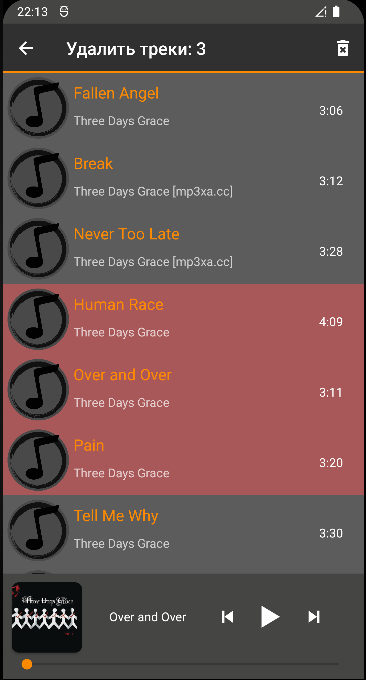


Рисунок 9 — Пометка на удаление

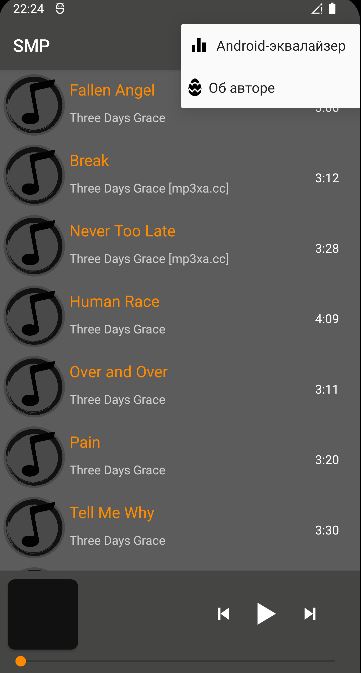


Рисунок 10 — Выпадающий список

* + 1. Эквалайзер

Эквалайзер (Рисунок 11) содержит в себе настройки повышения/понижения различных частот для изменения звучания композиции. Сам эквалайзер представляет из себя встроенную в операционную систему устройства утилиту, которая имеет различный внешний вид на разных устройствах и может отличаться по функционалу. Для примера приведён эквалайзер устройства Pixel 2 XL на API 31.

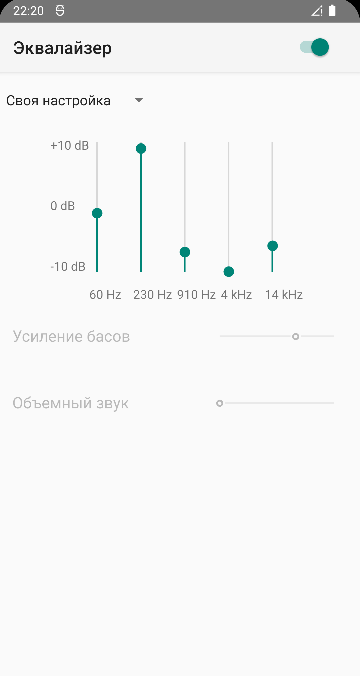


Рисунок 11 — Эквалайзер

* + 1. Отображение уведомления для управления воспроизведением

При запуске какого-либо трека из приложения в центре уведомлений будет отображено специальное уведомление (Рисунок 12) с возможностью управления воспроизведением. В уведомлении будет содержаться такая же информация, как и в панели управления воспроизведением в нижней части главного экрана.

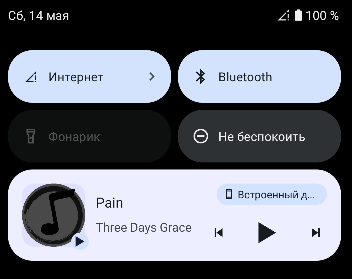


Рисунок 12 — Уведомление для управления воспроизведением

# Выбор технологии, языка и среды программирования

Мобильное приложение разрабатывается под Android, так как по статистике Google в мире насчитывается более 3,5 миллиардов пользователей, пользующихся устройствами под управлением этой ОС. По этой причине в качестве среды разработки выбрана Android Studio, как рекомендуемая IDE от самих производителей Android. Android Studio обладает конструктором пользовательских интерфейсов, способствующим упрощению визуального проектирования разрабатываемого приложения. Также в этой среде разработки имеется опционально подключаемый эмулятор устройства с предустановленной ОС и системой контроля версий.

В качестве языка программирования выбран Kotlin, так как по сравнению с Java, синтаксис Kotlin более легкий, тогда как функционал языков практически не отличается. Преимущества Kotlin заключаются в повышенной производительности, достигаемой за счёт интуитивно понятного синтаксиса.

Код, написанный на Java, выполняющий определённый функционал, занимает значительно меньше места на Kotlin, что способствует ускорению разработки и уменьшению занимаемого пространства на диске.

# РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ

# Описание разработанных процедур и функций

* + 1. Модель данных

Модель, заполняемая данными трека (название песни, номер трека, длительность, путь, имя альбома, название исполнителя и др.). Используется как для вывода информации о песне на пользовательский интерфейс, так и для работы внутренней логики (Листинг 1). Представляет из себя класс, который может вызываться из любого документа приложения.

data class Song(  
 val title: String,  
 val trackNumber: Int,  
 val year: Int,  
 val duration: Int,  
 val path: String?,  
 val albumName: String,  
 val artistId: Int,  
 val artistName: String)

Листинг 1 — Класс Song для работы с данными о треке

* + 1. Вывод треков

Для реализации вывода треков используется адаптер (Листинг 2) для сбора данных о треке через модель (Листинг 1) и чертёж (Листинг 3), заполняющий все генерирующиеся блоки в списке на главном экране. В функции адаптера addSongs заполняется список песен, которые имеются на накопителе устройства пользователя с помощью метода addAll, затем в классе чертежа главного экрана MainActivityBluePrint происходит вывод списка песен на главный экран с помощью адаптера, который вызывается при запуске приложения.

class RecyclerAdapter :  
 androidx.recyclerview.widget.RecyclerView.Adapter<RecyclerAdapter.ViewHolder>() {  
 private var onLongClick: OnLongClick? = null  
 private var onSongClicked: SongClicked? = null  
 private var songsSelected: SongsSelected? = null  
 private var songsList = *mutableListOf*<Song>()  
 private var selectedSongs = *mutableListOf*<Song>()  
 private var selectionModeActive = false  
  
 override fun onCreateViewHolder(viewGroup: ViewGroup, i: Int): ViewHolder {  
 return ViewHolder(LayoutInflater.from(viewGroup.*context*).inflate(R.layout.*track\_item*, viewGroup, false))  
 }  
 override fun onBindViewHolder(viewHolder: ViewHolder, position: Int) {  
 val song = songsList[position]  
 viewHolder.bind(song, position)  
 viewHolder.mainItem.*isSelected* = selectedSongs.contains(song)  
 viewHolder.mainItem.setOnLongClickListener **{** onLongClick?.onSongLongClicked(position)  
 if (!selectionModeActive) {  
 selectionModeActive = true  
 }  
 false  
 **}** viewHolder.mainItem.setOnClickListener **{** if (!selectionModeActive) {  
 onSongClicked?.onSongClicked(song)  
 } else {  
 if (selectedSongs.contains(song)) {  
 selectedSongs.remove(song)  
 songsSelected?.onSelectSongs(getSelectedSongs())  
  
 } else {  
 selectedSongs.add(song)  
  
 songsSelected?.onSelectSongs(getSelectedSongs())  
  
 }  
 notifyItemChanged(position)  
 }  
 **}** }  
  
 override fun getItemCount(): Int {  
 return songsList.size  
 }  
  
 fun addSongs(songs: MutableList<Song>) {  
 songsList.clear()  
 songsList.addAll(songs)  
 notifyDataSetChanged()  
 }  
  
 fun removeSelection() {  
 selectionModeActive = false  
 selectedSongs.clear()  
 notifyDataSetChanged()  
 }  
  
 fun getSelectedSongs(): MutableList<Song> {  
 return selectedSongs  
 }  
  
 fun updateRemoved(song: Song){  
 songsList.remove(song)  
 notifyDataSetChanged()  
 }  
 fun setOnSongClicked(songClick: SongClicked) {  
 this.onSongClicked = songClick  
 }  
  
 fun setOnLongClick(longClick: OnLongClick) {  
 this.onLongClick = longClick  
 }  
  
 fun setSongsSelected(selection: SongsSelected) {  
 this.songsSelected = selection  
 }  
 interface SongsSelected {  
 fun onSelectSongs(selectedSongs: MutableList<Song>)  
 }  
 interface SongClicked {  
 fun onSongClicked(song: Song)  
 }  
  
 interface OnLongClick {  
 fun onSongLongClicked(position: Int)  
 }  
  
 inner class ViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {  
 private val duration: TextView = itemView.findViewById(R.id.*duration*)  
 private val title: TextView = itemView.findViewById(R.id.*textViewSongTitle*)  
 private val artist: TextView = itemView.findViewById(R.id.*textViewArtistName*)  
 val mainItem: ConstraintLayout = itemView.findViewById(R.id.*mainConstraint*)  
 private var position: Int? = null  
  
 fun bind(song: Song, pos: Int) {  
 var dur = (song.duration/60).toString()  
 var start\_dur = dur[0]  
 var end\_dur = dur.*substring*(1, dur.length)  
 end\_dur = end\_dur.*substring*(0,2)  
 end\_dur = (end\_dur.*toInt*() \* 0.6).toInt().toString()  
 if(end\_dur.*toInt*() < 10)  
 {  
 end\_dur = "0" + end\_dur  
 }  
 else{}  
 dur = start\_dur + ":" + end\_dur  
 duration.*text* = dur  
 title.*text* = song.title  
 artist.*text* = song.artistName  
 position = pos  
 }

Листинг 2 — Класс RecyclerAdapter

abstract class MainActivityBluePrint : AppCompatActivity(), ActionMode.Callback, RecyclerAdapter.OnLongClick,  
 RecyclerAdapter.SongsSelected, RecyclerAdapter.SongClicked {  
 private var actionMode: ActionMode? = null  
 private var songAdapter: RecyclerAdapter? = null  
 private var deviceMusic = *mutableListOf*<Song>()  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_main*)  
 songAdapter = RecyclerAdapter()  
 setViews()  
 }  
  
 override fun onSongLongClicked(position: Int) {  
 if (actionMode == null) {  
 actionMode = startActionMode(this)  
 }  
 }  
  
 override fun onSelectSongs(selectedSongs: MutableList<Song>) {  
 if (selectedSongs.isEmpty()) {  
 actionMode?.finish()  
 songAdapter?.removeSelection()  
 } else {  
 val title = "Удалить треки: ${selectedSongs.size}"  
 actionMode?.*title* = title  
 }  
 }  
  
 override fun onCreateActionMode(mode: ActionMode?, menu: Menu?): Boolean {  
 val inflater = mode?.*menuInflater* inflater?.inflate(R.menu.*action\_mode\_menu*, menu!!)  
 toolbar.*visibility*= View.*GONE* return true  
 }  
  
 override fun onPrepareActionMode(mode: ActionMode?, menu: Menu?): Boolean {  
 return false  
 }  
  
 override fun onActionItemClicked(mode: ActionMode?, item: MenuItem?): Boolean {  
 when (item?.*itemId*) {  
 R.id.*action\_delete* -> {  
 val songs = songAdapter?.getSelectedSongs()  
 songs?.*forEach* **{** val file = File(**it**.path)  
 Utils.delete(this@MainActivityBluePrint, file)  
 songAdapter?.updateRemoved(**it**)  
 **}** Toast.makeText(this, "Удалено треков: ${songs?.size}", Toast.*LENGTH\_SHORT*).show()  
 mode?.finish()  
 songAdapter?.removeSelection()  
 return true  
 }  
  
 }  
 return false  
 }  
  
 override fun onDestroyActionMode(mode: ActionMode?) {  
 songAdapter?.removeSelection()  
 toolbar.*visibility*= View.*VISIBLE* actionMode = null  
 }  
  
 private fun setViews() {  
  
 songAdapter?.setOnLongClick(this)  
 songAdapter?.setSongsSelected(this)  
 songAdapter?.setOnSongClicked(this)  
 recyclerView?.*apply* **{** *adapter* = songAdapter  
 *layoutManager* = LinearLayoutManager(this@MainActivityBluePrint)  
 hasFixedSize()  
 **}** }  
  
 fun getMusic(){  
 deviceMusic.addAll(SongProvider.getAllDeviceSongs(this))  
 songAdapter?.addSongs(deviceMusic)  
 }  
}

Листинг 3 — Класс чертежа MainActivityBluePrint

* + 1. Вызов эквалайзера

Пользователь может вызвать эквалайзер при помощи выбора соответствующего пункта выпадающего меню. После нажатия происходит вызов утилиты эквалайзера через объект EqualizerUtils (Листинг 4) и его функцию openEqualizer, что позволяет влиять на воспроизведения звука внутри аудиофокуса, который на момент запуска принадлежит данному приложению.

object EqualizerUtils {  
 fun hasEqualizer(context: Context): Boolean {  
 val effects = Intent(AudioEffect.*ACTION\_DISPLAY\_AUDIO\_EFFECT\_CONTROL\_PANEL*)  
 val pm = context.*packageManager* val ri = pm.resolveActivity(effects, 0)  
 return ri != null  
 }  
  
 internal fun openAudioEffectSession(context: Context, sessionId: Int) {  
 val intent = Intent(AudioEffect.*ACTION\_OPEN\_AUDIO\_EFFECT\_CONTROL\_SESSION*)  
 intent.putExtra(AudioEffect.*EXTRA\_AUDIO\_SESSION*, sessionId)  
 intent.putExtra(AudioEffect.*EXTRA\_PACKAGE\_NAME*, context.*packageName*)  
 intent.putExtra(AudioEffect.*EXTRA\_CONTENT\_TYPE*, AudioEffect.*CONTENT\_TYPE\_MUSIC*)  
 context.sendBroadcast(intent)  
 }  
  
 internal fun closeAudioEffectSession(context: Context, sessionId: Int) {  
 val audioEffectsIntent = Intent(AudioEffect.*ACTION\_CLOSE\_AUDIO\_EFFECT\_CONTROL\_SESSION*)  
 audioEffectsIntent.putExtra(AudioEffect.*EXTRA\_AUDIO\_SESSION*, sessionId)  
 audioEffectsIntent.putExtra(AudioEffect.*EXTRA\_PACKAGE\_NAME*, context.*packageName*)  
 context.sendBroadcast(audioEffectsIntent)  
 }  
  
 internal fun openEqualizer(activity: Activity, mediaPlayer: MediaPlayer?) {  
 val sessionId = mediaPlayer?.*audioSessionId* if (sessionId == AudioEffect.*ERROR\_BAD\_VALUE*) {  
 notifyNoSessionId(activity)  
 } else {  
 try {  
 val effects = Intent(AudioEffect.*ACTION\_DISPLAY\_AUDIO\_EFFECT\_CONTROL\_PANEL*)  
 effects.putExtra(AudioEffect.*EXTRA\_AUDIO\_SESSION*, sessionId)  
 effects.putExtra(AudioEffect.*EXTRA\_CONTENT\_TYPE*, AudioEffect.*CONTENT\_TYPE\_MUSIC*)  
 activity.startActivityForResult(effects, 0)  
 } catch (notFound: ActivityNotFoundException) {  
 notFound.printStackTrace()  
 }  
  
 }  
 }

Листинг 4 — Объект EqualizerUtils

Таким образом, при использовании адаптера, который собирает данные о музыке на накопителе устройства пользователя, и чертежа, который выводит данные через шаблоны в список, происходит отображение этого самого списка на главном экране. Также, с помощью класса EqualizerUtils имеется доступ к системному эквалайзеру, который позволяет изменять звучание песен так, как пожелает пользователь.

# ТЕСТИРОВАНИЕ

Тестирование дизайна проводилось на минимальном (Android API 26) и на позднем (Android API 31) с различной диагональю экранов для проверки разметки страниц и вёрстки приложения.

Каждый экран прошёл проверку на наличие грамматических ошибок. Также каждый экран прошёл проверку на разных API на корректное отображение элементов соответствующих экранов. Примеры проверок отображения элементов на экране представлены на   
рисунках 13, 14 и 15 соответственно.

В процессе тестирование не было выявлено ошибок в дизайне приложения и его логике.

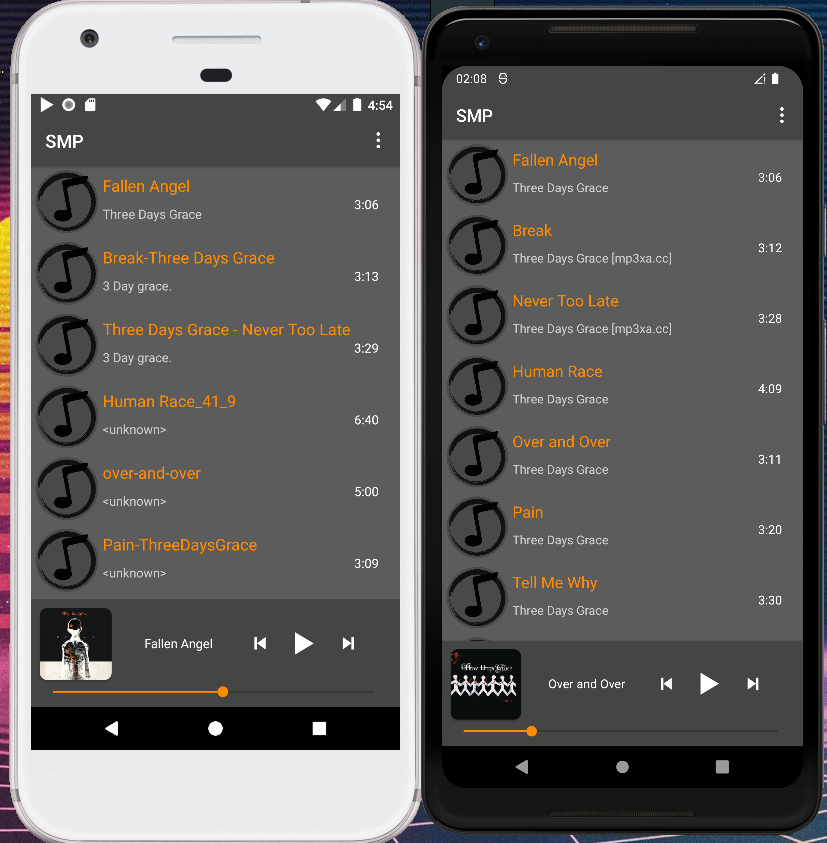


Рисунок 13 — Главный экран на 26 и 31 API соответственно

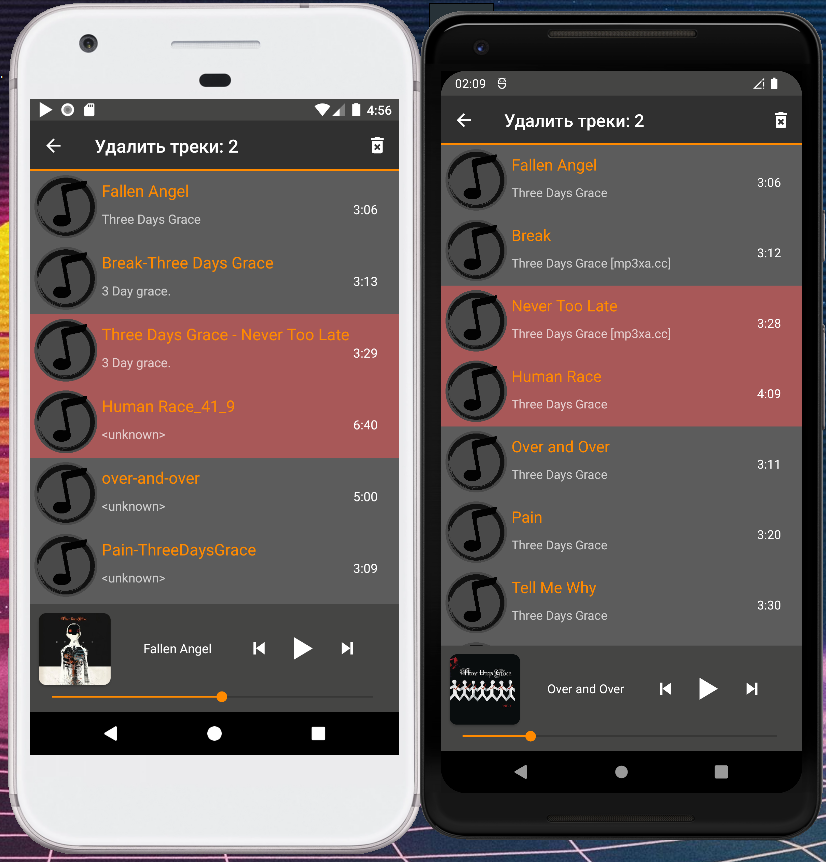


Рисунок 14 — Главный экран с пометкой песен на удаление на 26 и 31 API соответственно

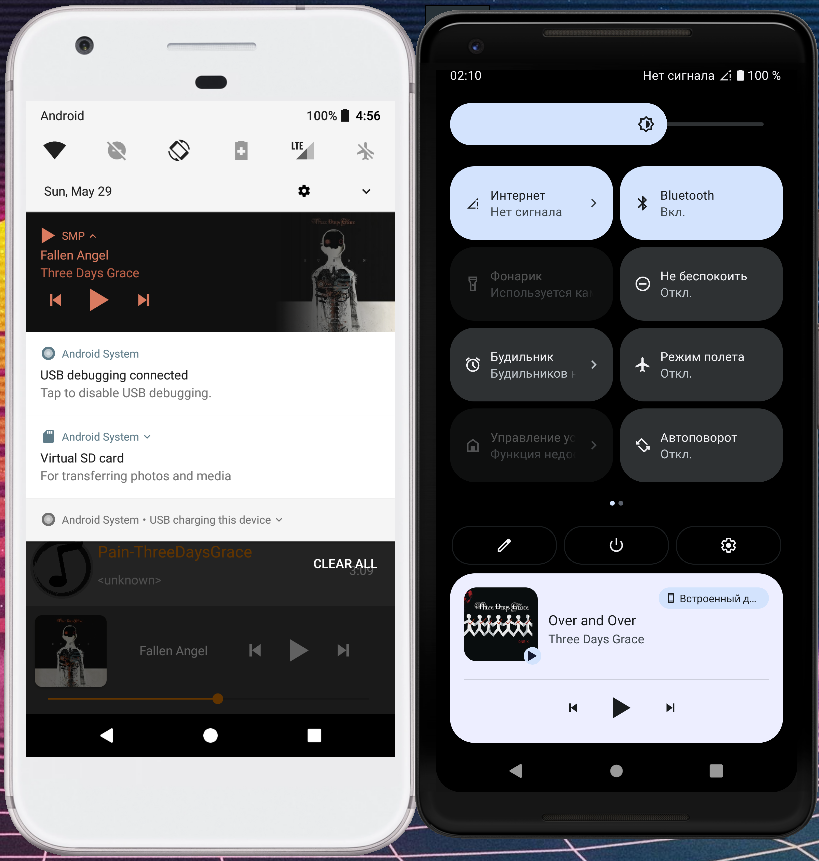


Рисунок 15 — Отображение оповещения в центре оповещений

# ЗАКЛЮЧЕНИЕ

В ходе исследования предметной области было выяснено, что множество приложений для прослушивания музыки на мобильном устройстве имеют большое количество минусов, таких как перегруженный интерфейс с малоиспользуемым, или вообще не используемым функционалом, который лишь занимает место и мешает навигации внутри приложения; невозможность прослушивания музыки без соединения с интернетом, или же некорректное её воспроизведение при нестабильном соединении с сетью; сложности с добавлением музыки в список, вызванные дополнительными действиями, требуемыми со стороны пользователя, особенно проявляющиеся на поздних версиях операционной системы Android, где безопасности пользователя уделено большое внимание, что приводит к отображению нескольких форм подтверждения действия, подтверждения доступа к внутреннему хранилищу устройства и подтверждению доступа к выбранной пользователем папке.

На основе приведённой выше собранной информации было принято решение разработать приложение, не имеющее перечисленных недостатков. Для этого был использован локальный метод хранения информации, который позволяет воспроизводить хранимые аудио-файлы из памяти устройства. Был разработан интерфейс с минимальным необходимым и часто используемым функционалом, не перегруженный лишними элементами и с универсальной цветовой темой, подходящей как для тёмного времени суток, так и при условиях повышенной освещённости за счёт контрастного и яркого акцента внимания в сочетании с тёмной цветовой палитрой, состоящей из оттенков серого. Для упрощения доступа к файлам был разработан соответствующий функционал, позволяющий приложению добавлять на главный экран музыку, хранимую на устройстве пользователя, без выбора папок и с единственным подтверждением при первом запуске приложения.

Для всего вышеперечисленного была выбрана среда разработки Android Studio с языком программирования Kotlin. Для разметки использовался встроенный язык XML.

# СПИСОК ИСТОЧНИКОВ

1. Android Developers [Электронный ресурс]: RecyclerView.Adapter. Режим доступа к руководству: [https://developer.android.com/reference/androidx/ recyclerview/widget/RecyclerView.Adapter](https://developer.android.com/reference/androidx/recyclerview/%20widget/RecyclerView.Adapter)
2. Kotlin Programming Language [Электронный ресурс]: Replace. Режим доступа к руководству:   https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.text/replace.html
3. Kotlin Programming Language [Электронный ресурс]: MutableList. Режим доступа к руководству:  https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.collections/-mutable-list/
4. Android Developers [Электронный ресурс]: SeekBar. Режим доступа к руководству:  https://developer.android.com/reference/android/widget/SeekBar
5. Android Developers [Электронный ресурс]: AudioEffect. Режим доступа к руководству: https://developer.android.com/reference/android/media/audiofx/AudioEffect
6. Android Developers [Электронный ресурс]: Toast. Режим доступа к руководству: https://developer.android.com/reference/kotlin/android/widget/Toast
7. Android Developers [Электронный ресурс]: MediaPlayer. Режим доступа к руководству: https://developer.android.com/reference/kotlin/android/media/MediaPlayer

# ПРИЛОЖЕНИЕ А

# (Обязательное)

Техническое задание

# ВВЕДЕНИЕ

Настоящее техническое задание распространяется на разработку мобильного приложения «Разработка мобильного аудио-плеера» для воспроизведения аудио-контента и его управлением.

Наименование программы: «Simple Music Player», «SMP». Далее по тексту Приложение.

Краткая характеристика области применения: мобильное приложение предоставляющее возможность воспроизводить аудио-файлы.

# 1 ОСНОВАНИЯ ДЛЯ РАЗРАБОТКИ

Основание для проведения разработки является Приказ №уч-041/4 от 17 марта 2022 года.

Наименование темы разработки − «Разработка мобильного аудио-плеера».

# 2 НАЗНАЧЕНИЯ ДЛЯ РАЗРАБОТКИ

Функциональным назначением приложения является обеспечение простого и удобного пользовательского интерфейса для возможности быстрой навигации в списке аудио-файлов и их воспроизведении.

# 3 Требование к приложению

# 3.1 Требования к функциональным характеристикам

Система должна обеспечивать возможность выполнения нижеперечисленных функций, описанных в таблице А.1.

Таблица А.1 – Выполняемые функции приложения

|  |  |
| --- | --- |
| Номер | Функция |
|  | Добавление аудио-файлов в список для визуального представления и их удаление |
|  | Воспроизведение аудио-файла и управление его воспроизведением (перемещение по временной линии трека при помощи ползунка и переключение на предыдущий, или следующий трек) |
|  | Отображение уведомления в центре уведомлений с возможностью просмотра информации о текущем проигрываемом треке и следующим набором функционала (переключение на предыдущий трек, постановка на паузу, возобновление воспроизведения на паузу, переключение на следующий трек) |
|  | Возможность перехода к системному эквалайзеру на актуальных версиях операционной системы. |

# 3.2 Требования к надёжности

Требования к надёжности не предоставляются

# 3.3 Условия эксплуатации

Пользователь должен иметь практические навыки использования мобильного устройства под управлением операционной системы Android.

# 3.4 Требования к составу и параметрам технических средств

Для работы приложения необходимо мобильное устройство с установленной операционной системой Android не ниже версии 9.0.

# 3.5 Требования к информационной и программной совместимости

Разработка приложения ведется на языке программирования Kotlin.

Для работы приложения необходимо мобильное устройство с установленной операционной системой Android с версией SDK не ниже 28.

# 3.6 Требования к защите информации

Требования не представляются.

# 3.7 Требования к маркировке и упаковке

Требования не представляются.

# 3.8 Специальные требования

Специальные требования не предоставляются.

# 4 ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

Состав программной документации должен включать в себя:

* техническое задание;
* пояснительная записка.

# 5 ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ

Экономические преимущества разработки и ориентировочная экономическая эффективность не рассчитывается.

# СТАДИИ И ЭТАПЫ РАЗРАБОТКИ

Таблица A.2 − Стадии разработки

|  |  |  |
| --- | --- | --- |
| Этапы разработки КП | Срок выполнения | Отчётность |
| Определение цели и задач, объекта и предмета исследования | 26.03.2022 | Пояснительная записка |
| Описание предметной области | 02.04.2022 | Пояснительная записка |
| Выбор технологии, языка и среды программирования | 09.04.2022 | Пояснительная записка |
| Оформление технического задания | 16.04.2022 | Пояснительная записка |
| Проектирование UI/UX дизайна | 23.04.2022 | Пояснительная записка |
| Разработка мобильного приложения | 30.04.2022 | Пояснительная записка |
| Разработка базы данных | 07.05.2022 | Пояснительная записка |
| Отладка и тестирование приложения | 14.05.2022 | Пояснительная записка |
| Оформление документации | 21.05.2022 |  |
| Защита | 30.05.2022 |  |

7 Порядок контроля и приЁмки

Виды испытаний – защита проекта.

Общее требования к приёмке:

* техническое задание;
* пояснительная записка;
* программный продукт.