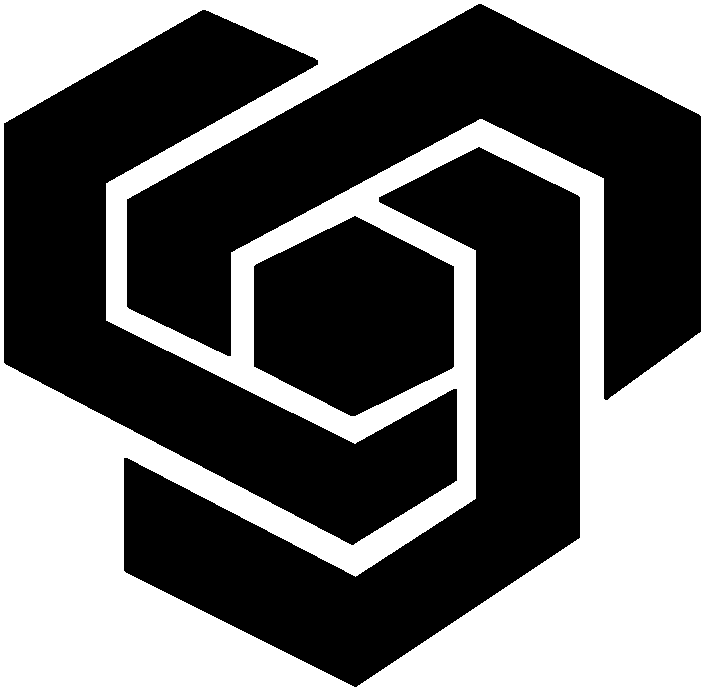
Технически университет - София



**„Програмиране за мобилни устройства“**

**Тема**

**Органайзер на музикални файлове и клипове**

Разработил: Христо Андреев Христов Преподавател:

/доц. Алексиева/

Факултет: ФКСУ

Фак.№: 501214043

Група: 51

2018

Съдържание

1. Увод
2. Анализ на съществуващи разработки
3. Проектиране
4. Реализация
5. Потребителско ръководство
6. Заключение
7. Литература
8. Приложение

Увод

Ако имате музика на телефона, то тогава е от значение вие да знаете че има бърз и лесен за начин да работите със вашите музикални файлове. Решението в случая е да се използва софтуерен инструмент за организация. В повечето случаи може да използвате вградения Media Player за слушането на музиката, но това е трудния начин понеже независимо откъде идват тези файлове, те могат да се намират в различни папки на телефона, също така те не са подредени в някакъв желан от вас ред. Повечето програми за организиране на музикални файлове нямат допълнителна функционалност като например правене на аудио запис. Музикалния органайзер ви предоставя решението на проблема вече не е нужно да търсите музикалните си файлове в различните папки на телефона си, те ще бъдат автоматично заредени от органайзера.

Анализ на съществуващи разработки

Съществуваща разработка на музикален органайзер като пример е Apple Music за Android. Той дава възможност за подреждане на музикалните файлове по Албуми, Певци и имена на песни. Предлага и възможност за слушане на радио, но няма опция за гледане на видео. На базата на музиката която даден потребител слуша дава препоръки за подобна музика която може би ще бъде харесана от потребителя.

Друг много добър пример, само че той е desktop приложение е iTunes. Той дава много голяма разновидност от функции. Дава възможност за сортирането на песните по-жанр. Също така има достъп до онлайн магазин от който могат да се закупят песни. Друга негова възможност е да предоставя на потребителя възможност за създаване на папки с различни имена и сортирането на песните вътре в тези папки. Като цяло телефонната версия няма да има толкова голяма възможност за всички тези операции, но пък ще е лесна и удобна за използване (по-съкратена).

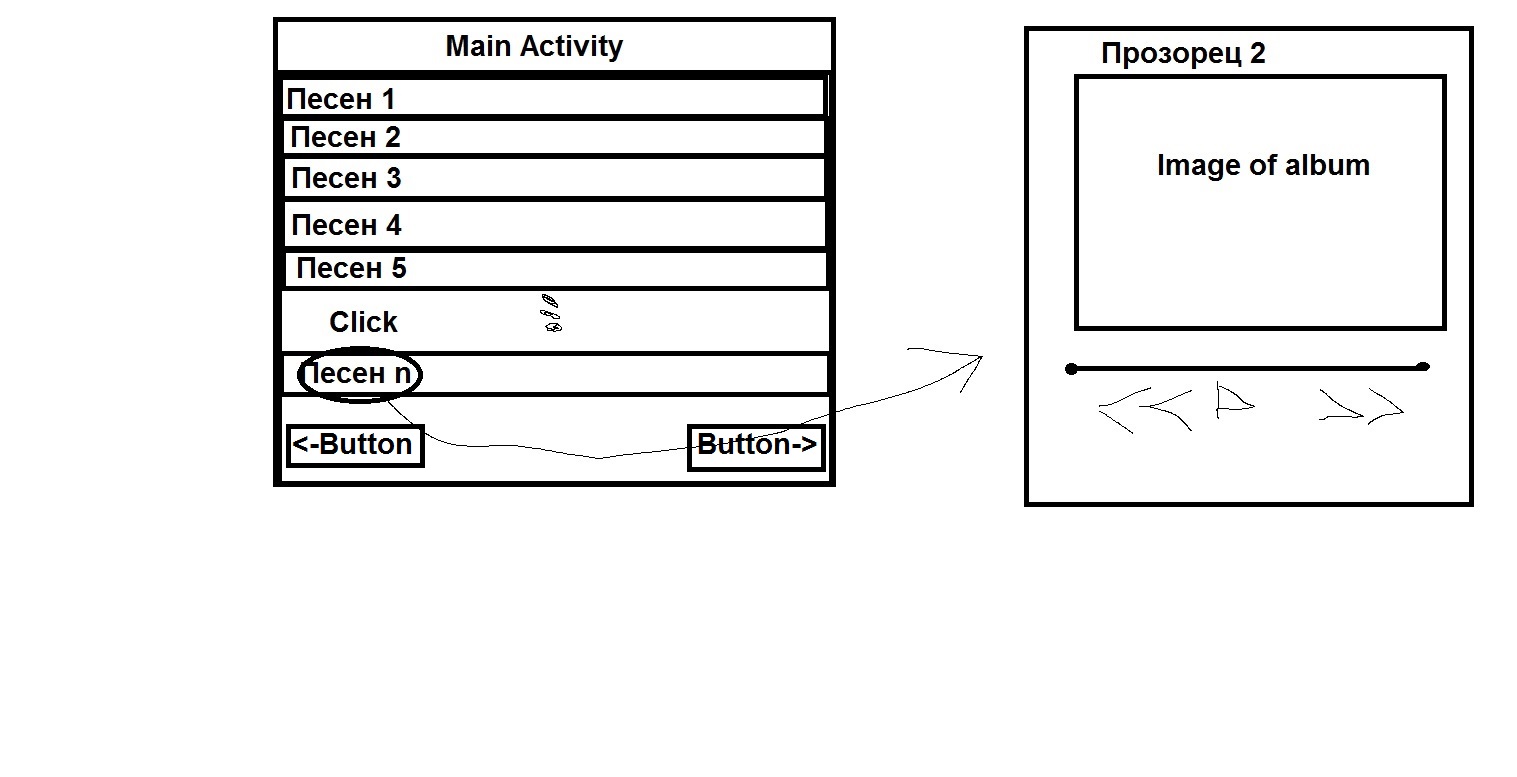
Проектиране

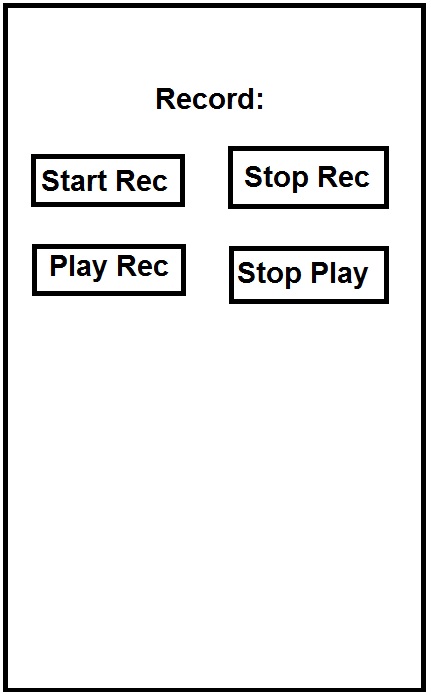
Музикалния органайзер ще има възможност за подреждане на музиката по име на песен, име на автор или по албум. Ще има Shuffle функция за разбъркване на слушаните песни. Като разбира се ще дава възможност песента да бъде превъртана напред и назад. Като допълнителна функционалност ще се използва функционалност на телефона – микрофон за да се записват аудио файлове, като ще може последния записан файл да се чуе на момента. Името на файла ще е генерирано на случаен принцип и записано автоматично в телефона.

Потребителя ще има възможност да гледа и .mp4 клипове като те ще бъдат подредени в един списък.

Програмата ще се състои от 3 активитита като навигацията от главното към второстепенните ще става чрез Gesture(„жест“) и от второстепенните към главното пак със „жест“. Проектирано е така за да наподобява едно реално приложение.

Това приложение няма ограничение за това какви хора могат да го използват. То е предназначено за обществото. Начални версии на потребителския интерфейс са дадени на следните картини:



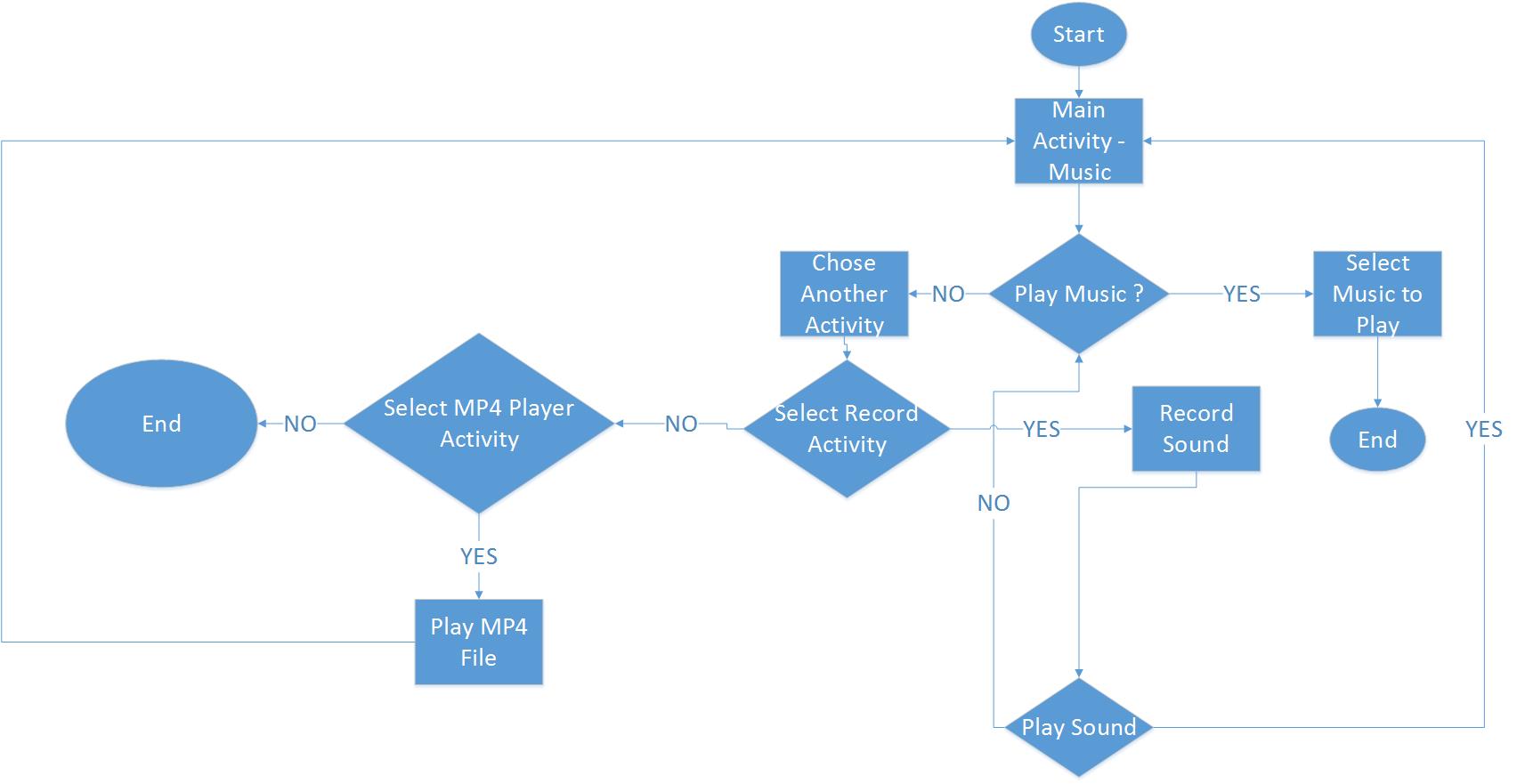


Както се вижда от първоначалния замисъл за приложението (това са картинки от преди да се напише) приложението се е променило.

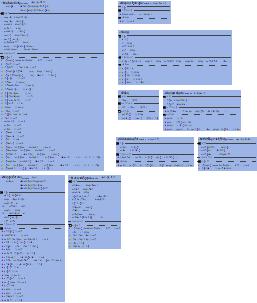
Приложението не използва База с данни затова ER диаграма няма, като данните от които моето приложение се интересува това са .mp3, mp4 и .3gpp файлове. За тези файлове приложението използва функция за търсене и когато открие такъв файл той му взима името и URI тоест там където се намира и съставя списък с имената на такива файлове. Когато списъка е готов се подава на ListView за визуализация.

Входните данни на приложението са от сорта на – Button Click, Gesture(swipe right , swipe left) и Глас – Recorder. Това приложение няма поле в което потребителя да въвежда текстова информация следователно проверка за входните данни няма, просто защото входната информация няма как да бъде нещо което е неочаквано.

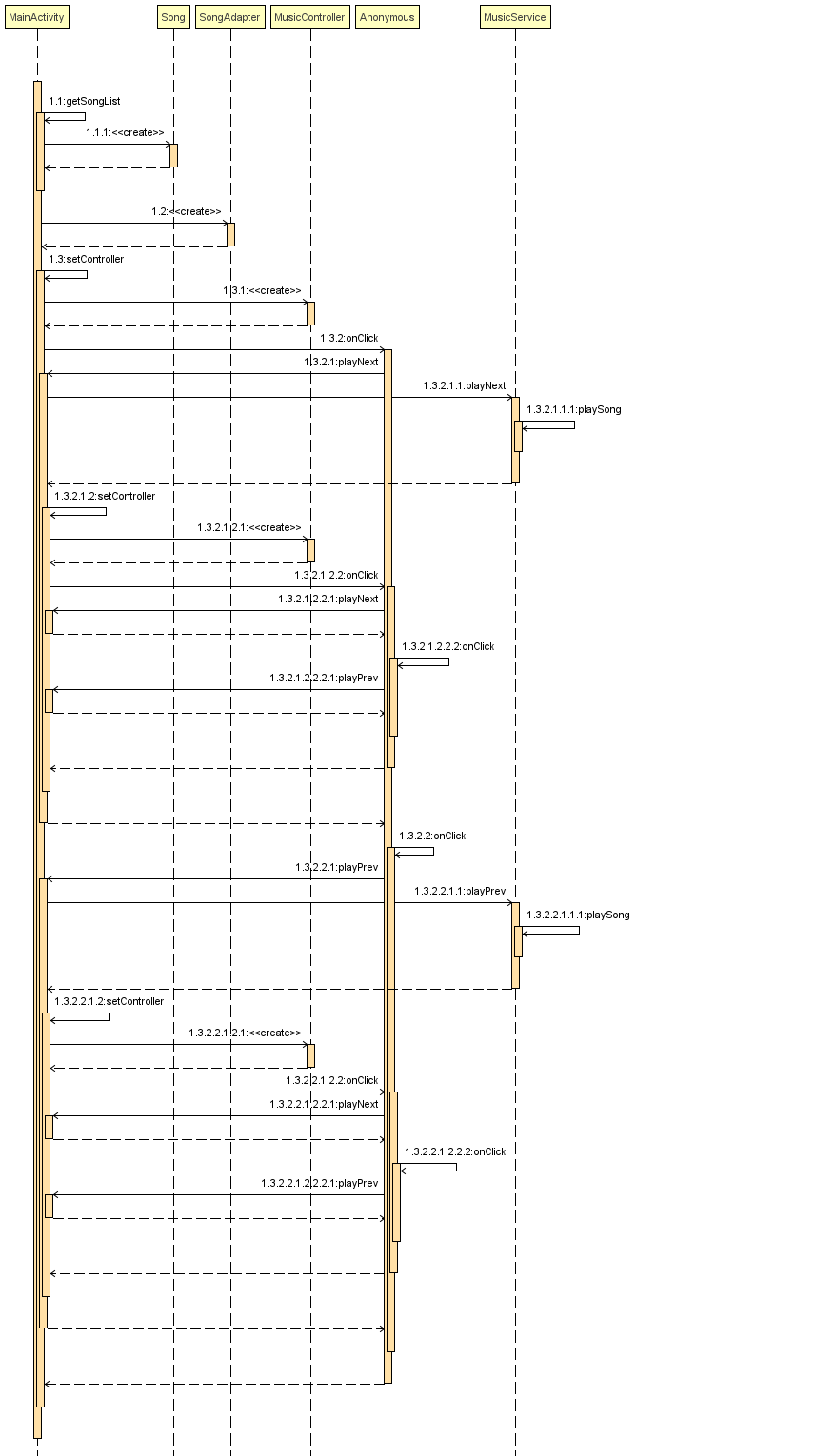
Workflow диаграма:



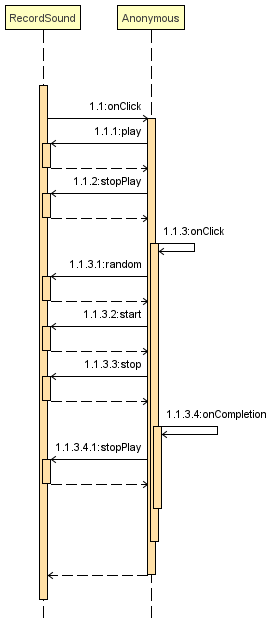
Клас диаграма:



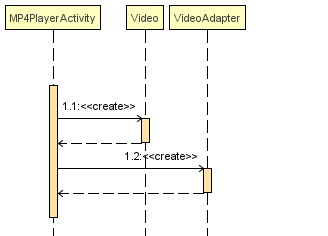
Sequence диаграма на Main Activity:



Sequence диаграма на Record Activity:



Sequence диаграма на MP4PlayerActivity:



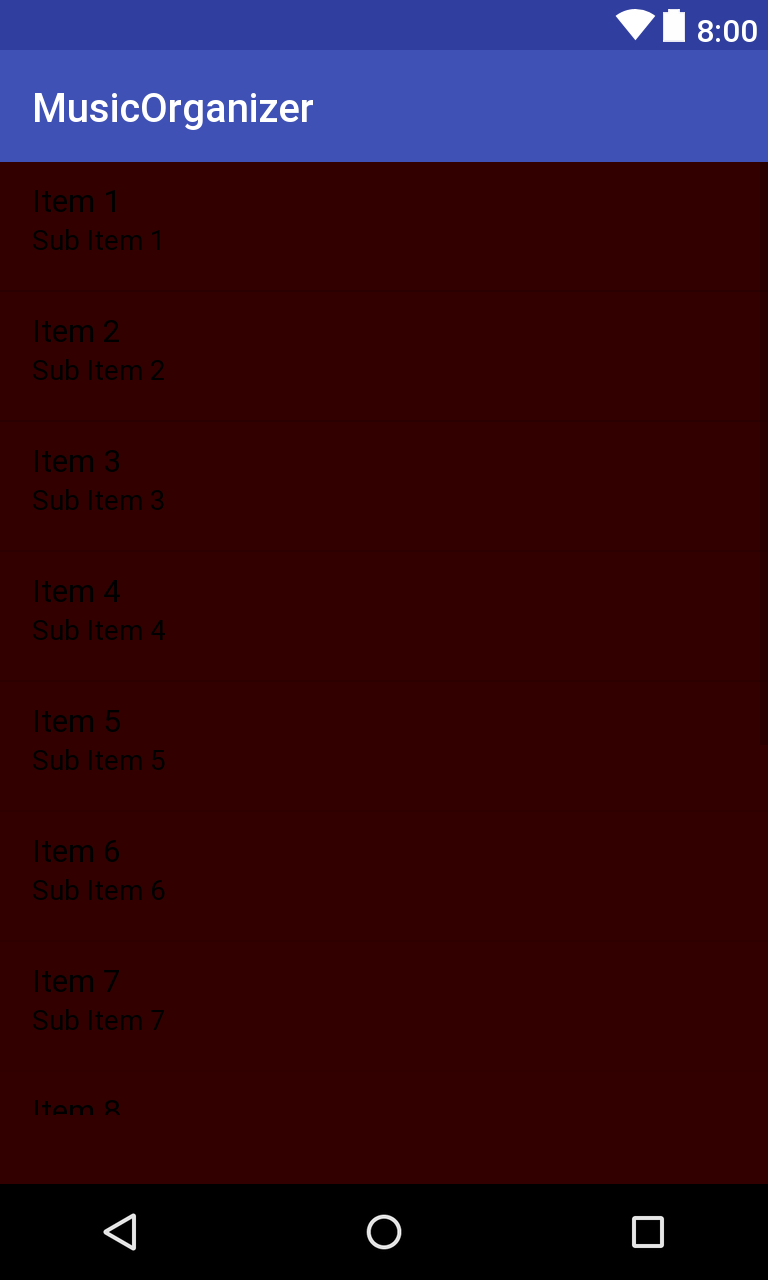
Реализация

Интерфейса на трите прозореца(activity-та) които програмата има имат следния вид в .xml файловете си:

MainActivity:

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="fill\_parent"  
 android:layout\_height="fill\_parent"  
 android:orientation="vertical"  
 android:background="#FF330000"  
 tools:context=".MainActivity" >  
  
 <!-- song list -->  
  
 <ListView  
 android:id="@+id/song\_list"  
 android:layout\_width="fill\_parent"  
 android:layout\_height="0dp"  
 android:layout\_weight="14"></ListView>  
 <LinearLayout  
 android:layout\_width="wrap\_content"  
 android:layout\_height="0dp"  
 android:layout\_weight="1"  
 android:orientation="horizontal"  
 android:gravity="center\_horizontal">  
  
 </LinearLayout>  
</LinearLayout>

Визуално има следния вид:



Като самото визуализиране ще става посредством друг .xml файл – song.xml

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="fill\_parent"  
 android:layout\_height="wrap\_content"  
 android:onClick="songPicked"  
 android:orientation="horizontal"  
 android:padding="5dp" >  
  
 <ImageView  
 android:id="@+id/song\_art"  
 android:layout\_width="100dp"  
 android:layout\_height="100dp" />  
 <LinearLayout  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical">  
 <TextView  
 android:id="@+id/song\_title"  
 android:layout\_width="fill\_parent"  
 android:layout\_height="wrap\_content"  
 android:textColor="#FFFFFF99"  
 android:textSize="20sp"  
 android:textStyle="bold" />  
  
 <TextView  
 android:id="@+id/song\_artist"  
 android:layout\_width="fill\_parent"  
 android:layout\_height="wrap\_content"  
 android:textColor="#FFFFFF99"  
 android:textSize="18sp" />  
 <TextView  
 android:id="@+id/song\_album"  
 android:layout\_width="fill\_parent"  
 android:layout\_height="wrap\_content"  
 android:textColor="#FFFFFF99"  
 android:textSize="16sp" />  
 </LinearLayout>

</LinearLayout>

Класът Song представлява прототип от него ще се създават обекти които ще представляват една песен в списъка с песни. Той има следния вид:

public class Song {  
  
 private long id;  
 private String title;  
 private String artist;  
 private String album;  
 private Bitmap albumArt;  
  
 public Song(long songID, String songTitle, String songArtist, String songAlbum,Bitmap songAlbumArt){  
 id=songID;  
 title=songTitle;  
 artist=songArtist;  
 album=songAlbum;  
 albumArt=songAlbumArt;  
 }  
  
 public long getID(){return id;}  
 public String getTitle(){return title;}  
 public String getArtist(){return artist;}  
 public String getAlbum() {return album;}  
 public Bitmap getAlbumArt(){return albumArt;}  
}

Важните функции за този код:

public void getSongList(){  
 //query external audio  
 ContentResolver musicResolver = getContentResolver();  
 Uri musicUri = android.provider.MediaStore.Audio.Media.*EXTERNAL\_CONTENT\_URI*;  
 Cursor musicCursor = musicResolver.query(musicUri, null, null, null, null);  
 //iterate over results if valid  
 if(musicCursor!=null && musicCursor.moveToFirst()){  
 //get columns  
 int titleColumn = musicCursor.getColumnIndex(android.provider.MediaStore.Audio.Media.*TITLE*);  
 int idColumn = musicCursor.getColumnIndex(android.provider.MediaStore.Audio.Media.*\_ID*);  
 int artistColumn = musicCursor.getColumnIndex(android.provider.MediaStore.Audio.Media.*ARTIST*);  
 int albumColumn = musicCursor.getColumnIndex(MediaStore.Audio.Media.*ALBUM*);  
 int albumArtColumn = musicCursor.getColumnIndex(MediaStore.Audio.Media.*ALBUM\_ID*);  
 //add songs to list  
 do {  
 long thisId = musicCursor.getLong(idColumn);  
 String thisTitle = musicCursor.getString(titleColumn);  
 String thisArtist = musicCursor.getString(artistColumn);  
 String thisAlbum = musicCursor.getString(albumColumn);  
 long thisArtID = musicCursor.getLong(albumArtColumn);  
 Bitmap bitmap=null;  
 try {  
 Cursor cursor = getContentResolver().query(MediaStore.Audio.Albums.*EXTERNAL\_CONTENT\_URI*,  
 new String[]{MediaStore.Audio.Albums.*\_ID*, MediaStore.Audio.Albums.*ALBUM\_ART*},  
 MediaStore.Audio.Albums.*\_ID* + "=?",  
 new String[]{String.*valueOf*(thisArtID)},  
 null);  
 if (cursor.moveToFirst()) {  
 String path = cursor.getString(cursor.getColumnIndex(MediaStore.Audio.Albums.*ALBUM\_ART*));  
 bitmap = BitmapFactory.*decodeFile*(path);  
 }  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
  
 songList.add(new Song(thisId, thisTitle, thisArtist,thisAlbum,bitmap));  
 }  
 while (musicCursor.moveToNext());  
 }  
}

Тази функция взема аудио информацията за песните. Тази аудио информация съдържа много неща като например – име на песен, име на артист, име на албум и т.н. общо взето цялата информация която може да има една песен се извлича чрез метода.

Класът SongAdapter служи за свързване на списъка с песни с главното View на MainActivity. Използва се LayoutInflater за да се „надуе“ Layout на MainActivity-то със всички песни.

Класът MusicService ще позволи потребителя да има достъп по Play функции за песните които кликва. Използва се вградени функции за Play, Rewind бутоните и като цяло On Screen менюто което се появява, дава цялата функционалност по слушането на песента, тя се делегира на този клас:

public class MusicService extends Service implements  
 MediaPlayer.OnPreparedListener, MediaPlayer.OnErrorListener,  
 MediaPlayer.OnCompletionListener

На този прозорец ние искаме да можем да му зададем лист с песните които той разполага за да знае с каква информация работи:

//pass song list  
public void setList(ArrayList<Song> theSongs){  
 songs=theSongs;  
}

Трябва ни е „Binder“ клас който да укаже че Услугата за слушането на музика е свързана с листа с песни:

//binder  
public class MusicBinder extends Binder {  
 MusicService getService()  
 {  
 return MusicService.this;  
 }  
}

Сега като вече имаме услугата за слушане на музика ние искаме да създадем инстанция от нея и да я свържем с главното. Резултата който се получава е че ние контролираме слушането на музика с класа MusicService, но го управляваме през MainActivity-то:

//connect to the service  
private ServiceConnection musicConnection = new ServiceConnection(){  
  
 @Override  
 public void onServiceConnected(ComponentName name, IBinder service) {  
 MusicBinder binder = (MusicBinder)service;  
 //get service  
 musicSrv = binder.getService();  
 //pass list  
 musicSrv.setList(songList);  
 musicBound = true;  
 }  
  
 @Override  
 public void onServiceDisconnected(ComponentName name) {  
 musicBound = false;  
 }  
};

Когато се стартира MainActivity-то трябва да се стартира и MusicService услугата.

Като сега ни трябва начин да визуализираме контролите – Функционалността за контрола на слушане на песента я представихме чрез MusicService, да но самото визуализиране на тези контроли става чрез:

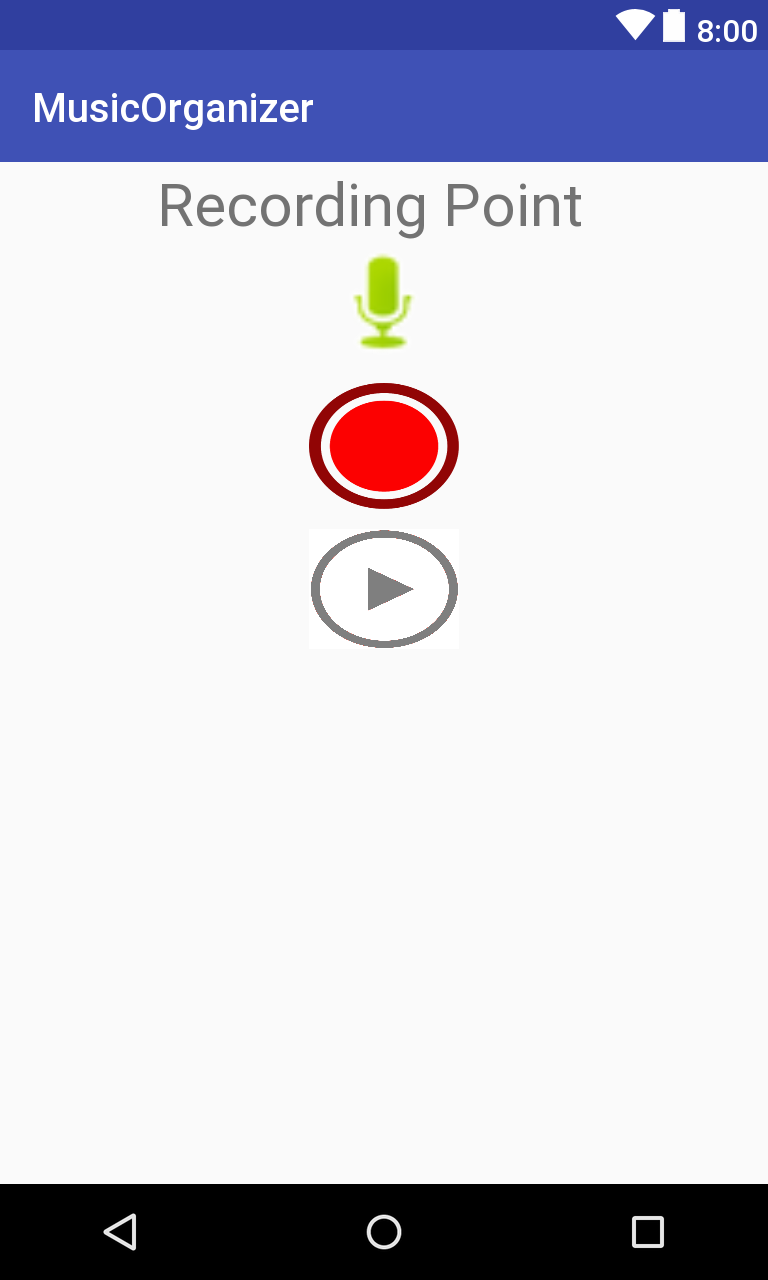
public class MusicController extends MediaController {  
  
 public MusicController(Context c){  
 super(c);  
 }  
  
 public void hide(){}  
  
}

Това е Widget клас който ни предоставя default-ни контроли за тази работа.

Останалата част на кода смятам че е self-explanatory затова няма да й обръщам голяма внимание, но само ще добавя снимки за изгледите на 2-та останали прозореца – Record и MP4Player.

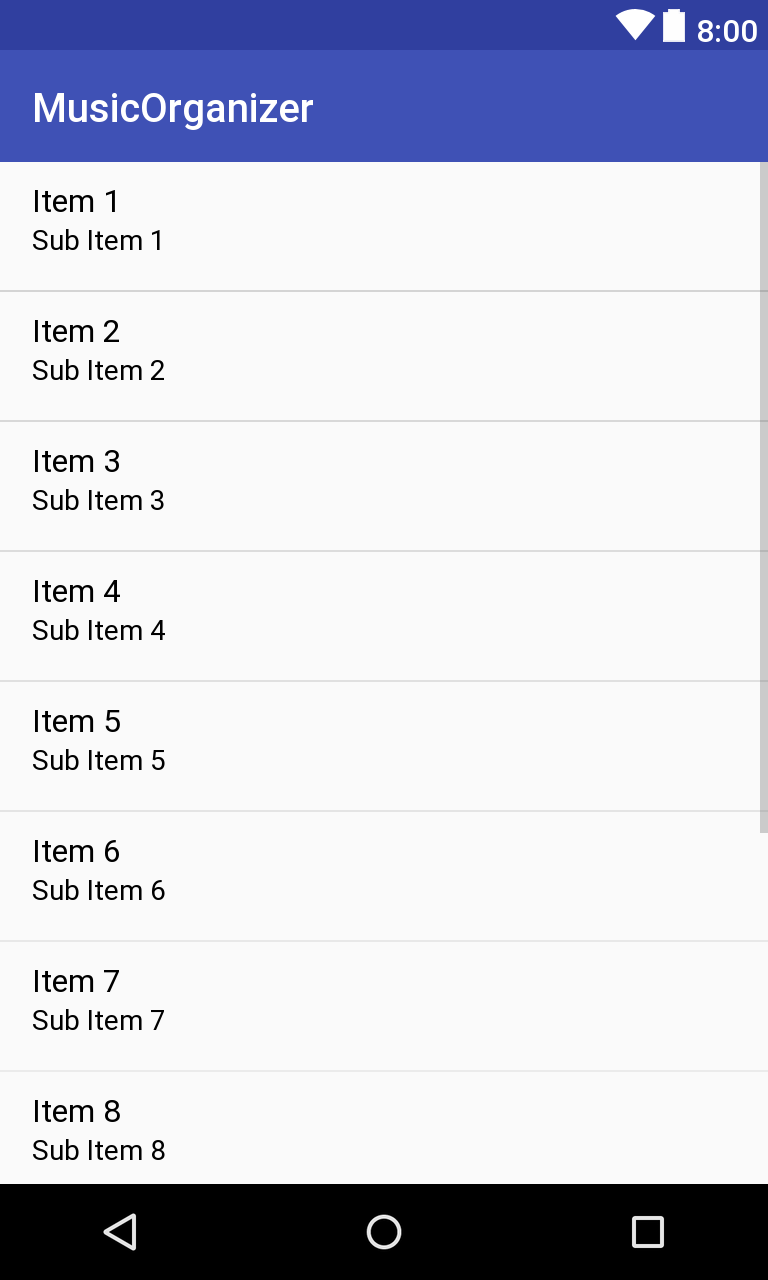
Record:

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="fill\_parent"  
 android:layout\_height="fill\_parent"  
 android:orientation="vertical"  
 android:gravity="center\_horizontal"  
 tools:context=".MainActivity" >  
  
 <TextView  
 android:id="@+id/text1"  
 android:layout\_width="227dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentTop="true"  
 android:layout\_centerHorizontal="true"  
 android:text="Recording Point"  
 android:textSize="30dp" />  
  
 <ImageView  
 android:id="@+id/micImage"  
 android:layout\_width="match\_parent"  
 android:layout\_height="60dp"  
 android:layout\_alignParentStart="true"  
 android:layout\_below="@+id/text1"  
 android:src="@android:drawable/presence\_audio\_online" />  
  
 <RelativeLayout  
 android:id="@+id/linear1"  
 android:layout\_width="match\_parent"  
 android:layout\_height="63dp"  
 android:layout\_below="@+id/micImage"  
 android:gravity="center\_horizontal"  
 android:layout\_marginTop="10dp"  
 android:orientation="horizontal">  
  
 <ImageButton  
 android:id="@+id/start"  
 android:layout\_width="75dp"  
 android:layout\_height="match\_parent"  
 android:background="@drawable/record"  
 android:text="start" />  
  
 </RelativeLayout>  
  
 <RelativeLayout  
 android:id="@+id/linear2"  
 android:layout\_width="match\_parent"  
 android:layout\_height="60dp"  
 android:layout\_below="@+id/linear1"  
 android:layout\_marginTop="10dp"  
 android:gravity="center\_horizontal"  
 android:orientation="horizontal">  
  
 <ImageButton  
 android:id="@+id/play"  
 android:layout\_width="75dp"  
 android:layout\_height="match\_parent"  
 android:background="@drawable/disabled\_play"  
 android:text="play" />  
  
 </RelativeLayout>  
  
</RelativeLayout>

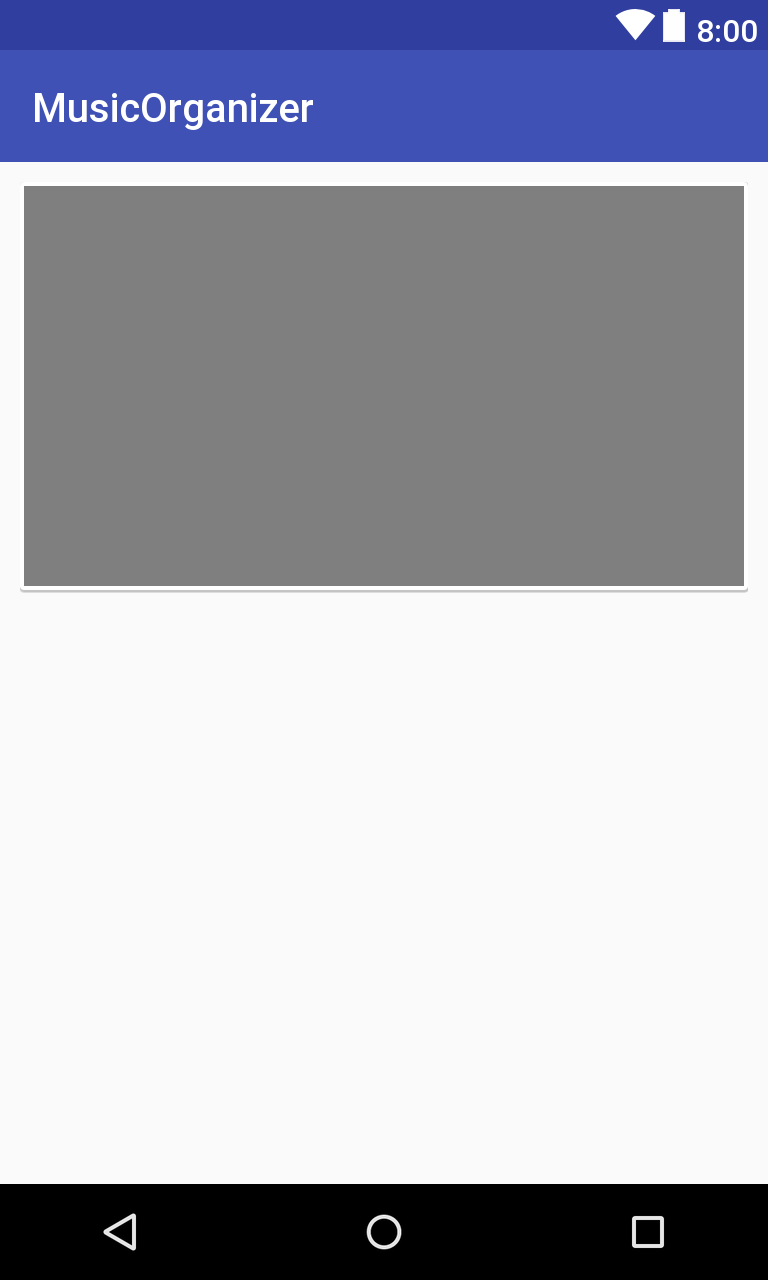


MP4Player:

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 tools:context=".MP4PlayerActivity">  
  
 <ListView  
 android:id="@+id/videoListView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"/>  
  
</LinearLayout>



<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:padding="10dp"  
 android:orientation="vertical">  
  
  
 <android.support.v7.widget.CardView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content">  
  
 <VideoView  
 android:layout\_gravity="center"  
 android:layout\_margin="2dp"  
 android:id="@+id/videoView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="200dp" />  
  
 </android.support.v7.widget.CardView>  
  
  
  
</LinearLayout>

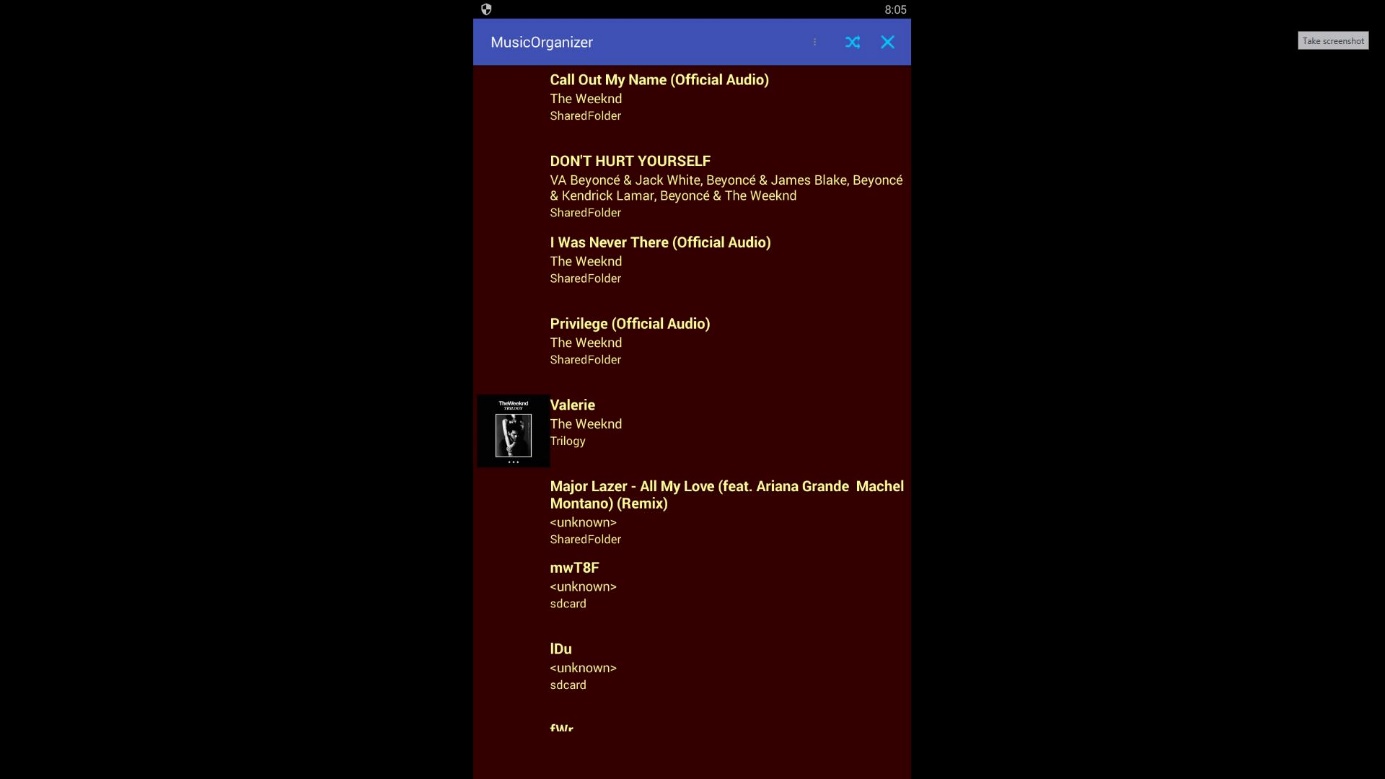


Като последен пример (примера е само един но не е единствен в кода) – Exception Handling:

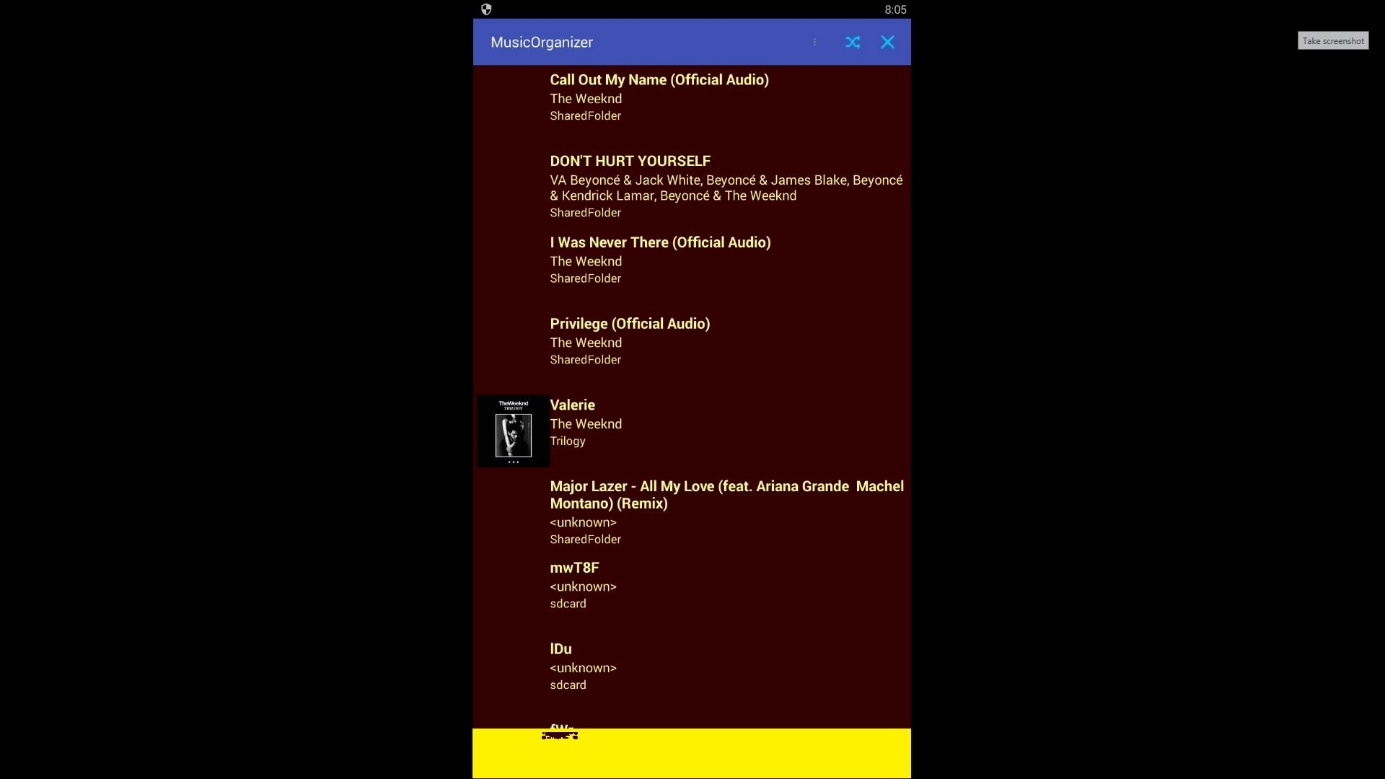
public void start(View view){  
 try {  
 myRecorder.prepare();  
 myRecorder.start();  
  
 } catch (IllegalStateException e) {  
 // start:it is called before prepare()  
 // prepare: it is called after start() or before setOutputFormat()  
 e.printStackTrace();  
 } catch (IOException e) {  
 // prepare() fails  
 e.printStackTrace();  
 }catch (Exception e) {  
 e.printStackTrace();  
 }  
 Toast.*makeText*(getApplicationContext(), "Start recording...", Toast.*LENGTH\_SHORT*).show();  
}  
public void stop(View view){  
 try {  
 myRecorder.stop();  
 myRecorder.reset();  
 myRecorder.release();  
  
 myRecorder = new MediaRecorder();  
 myRecorder.setAudioSource(MediaRecorder.AudioSource.*MIC*);  
 myRecorder.setOutputFormat(MediaRecorder.OutputFormat.*THREE\_GPP*);  
 myRecorder.setAudioEncoder(MediaRecorder.OutputFormat.*AMR\_NB*);  
  
 Toast.*makeText*(getApplicationContext(), "Stop recording...", Toast.*LENGTH\_SHORT*).show();  
 } catch (IllegalStateException e) {  
 // it is called before start()  
 e.printStackTrace();  
 } catch (RuntimeException e) {  
 // no valid audio/video data has been received  
 e.printStackTrace();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
}  
public void play() {  
 try{  
 myPlayer.setDataSource(outputFile);  
 myPlayer.prepare();  
 myPlayer.start();  
  
 Toast.*makeText*(getApplicationContext(), "Start play the recording...", Toast.*LENGTH\_SHORT*).show();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
}  
public void stopPlay() {  
 try {  
 myPlayer.stop();  
 myPlayer.reset();  
  
 Toast.*makeText*(getApplicationContext(), "Stop playing the recording...", Toast.*LENGTH\_SHORT*).show();  
  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
}

Това са няколко функции от RecordSound activity-то в които се проверяват MediaRecorder и MediaPlayer обектите за валидни операции.

Потребителско ръководство

След като бъде стартирана програмата пред потребителя ще се отвори следния прозорец

Това е главния прозорец на програмата от него чрез „жест“ може да се премине към двата останали прозореца като мястото на жеста трябва да е в долния край на приложението:



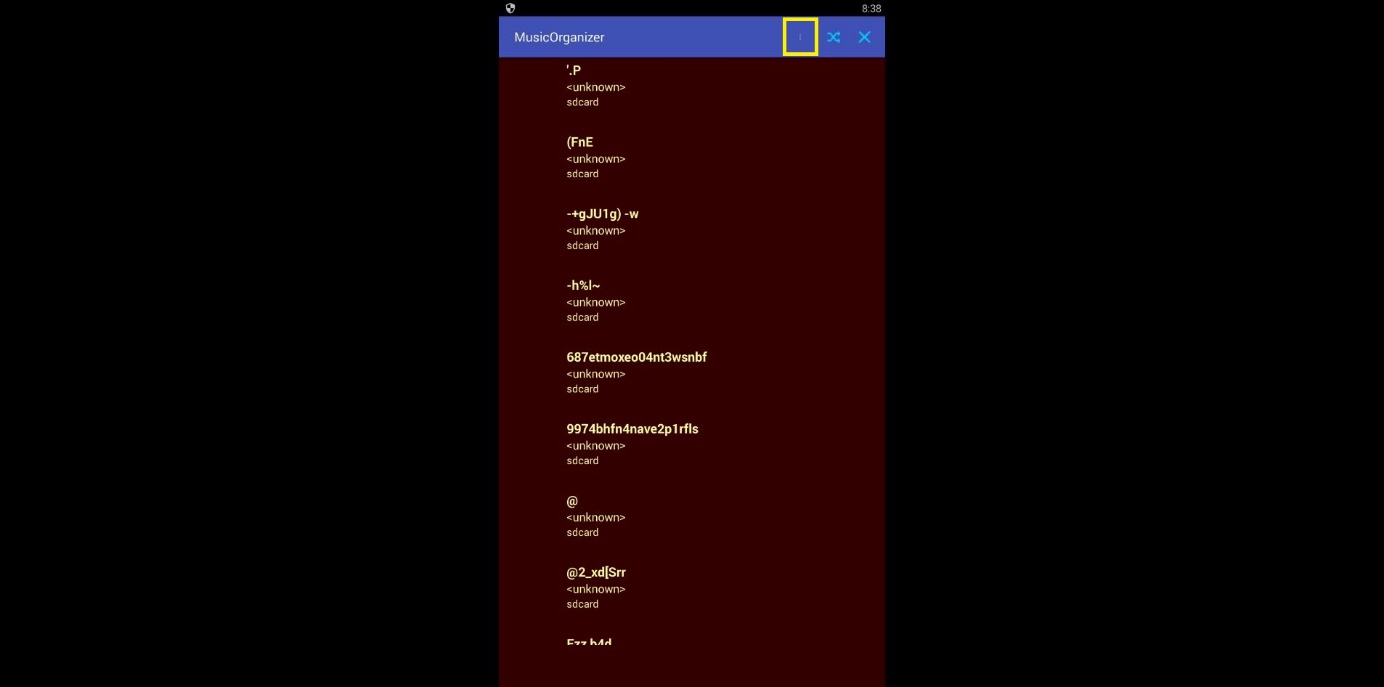
Оцветената в жълто част е тази в която трябва да се извърши жеста (Swipe left и Swipe right).

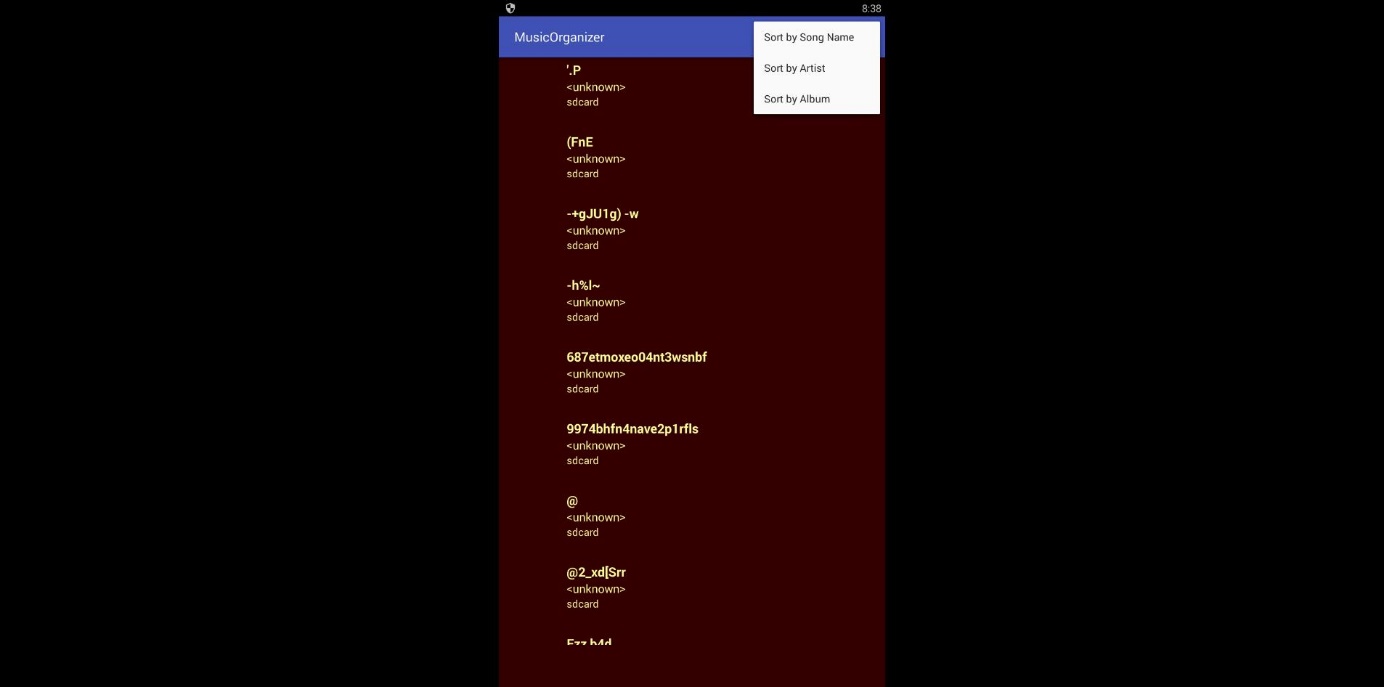
Ако жеста е от дясно на ляво се отваря прозорецът с гледане на видео и обратно ако жеста е от ляво на дясно се отваря прозорецът с рекордера на звук.

Когато сме в един от двата допълнителни прозореца – Рекордера или Прозореца за гледане на видео, може да се върнем към главния на програмата като изпълним жест Swipe от ляво на дясно къде и да е по екрана.

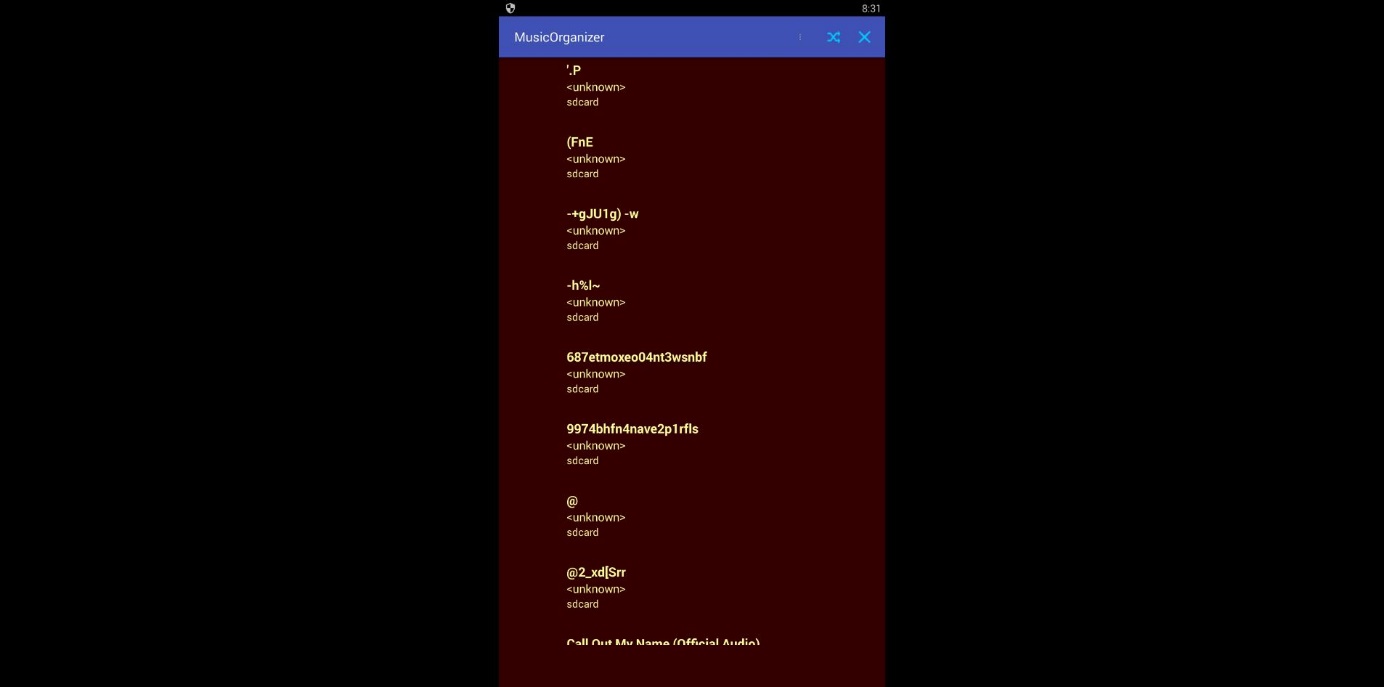
Резултата от работата на програмата няма как да го демонстрирам тук понеже той е звук или видео – слушане на песен или слушане на запис или гледане на клип. Единствената функционалност която мога да покажа е сортиране на песни по име на песен, албум и артист.

Сортирането се извършва като от менюто горе в дясно на екрана се натиснат трите точки и пред потребителя се отварят опциите:

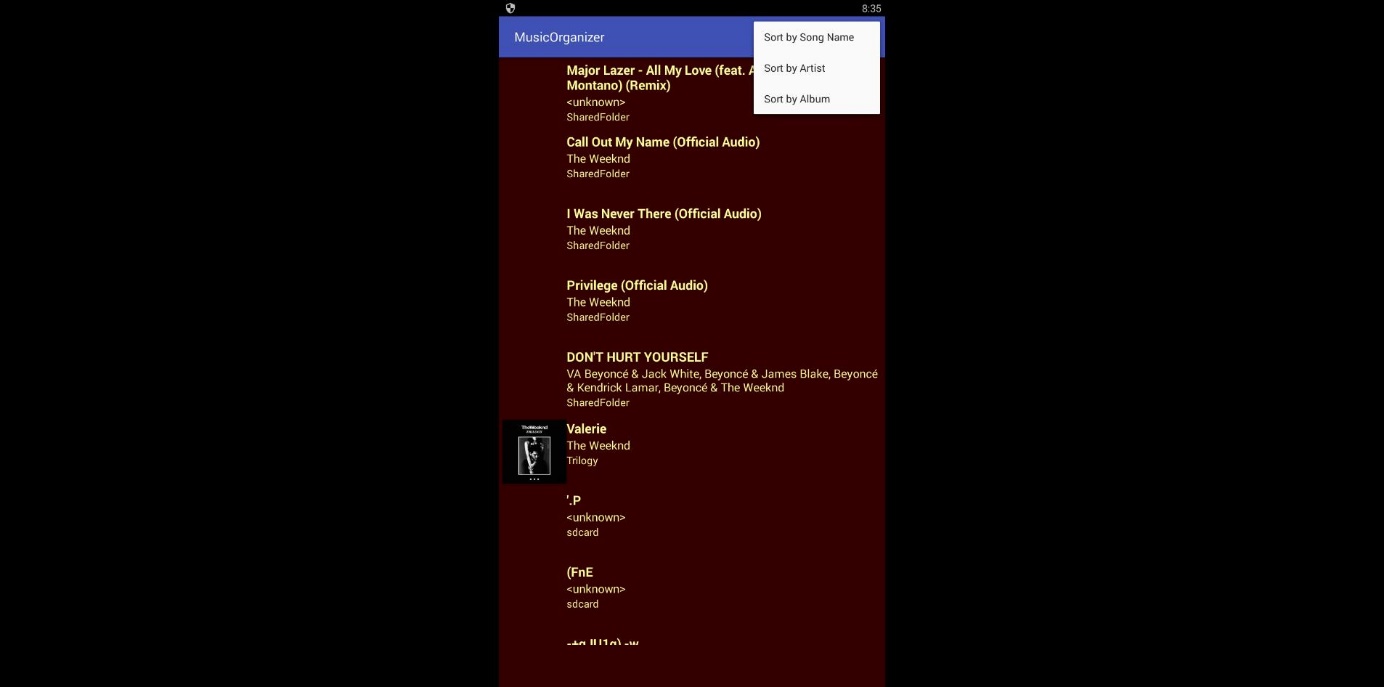




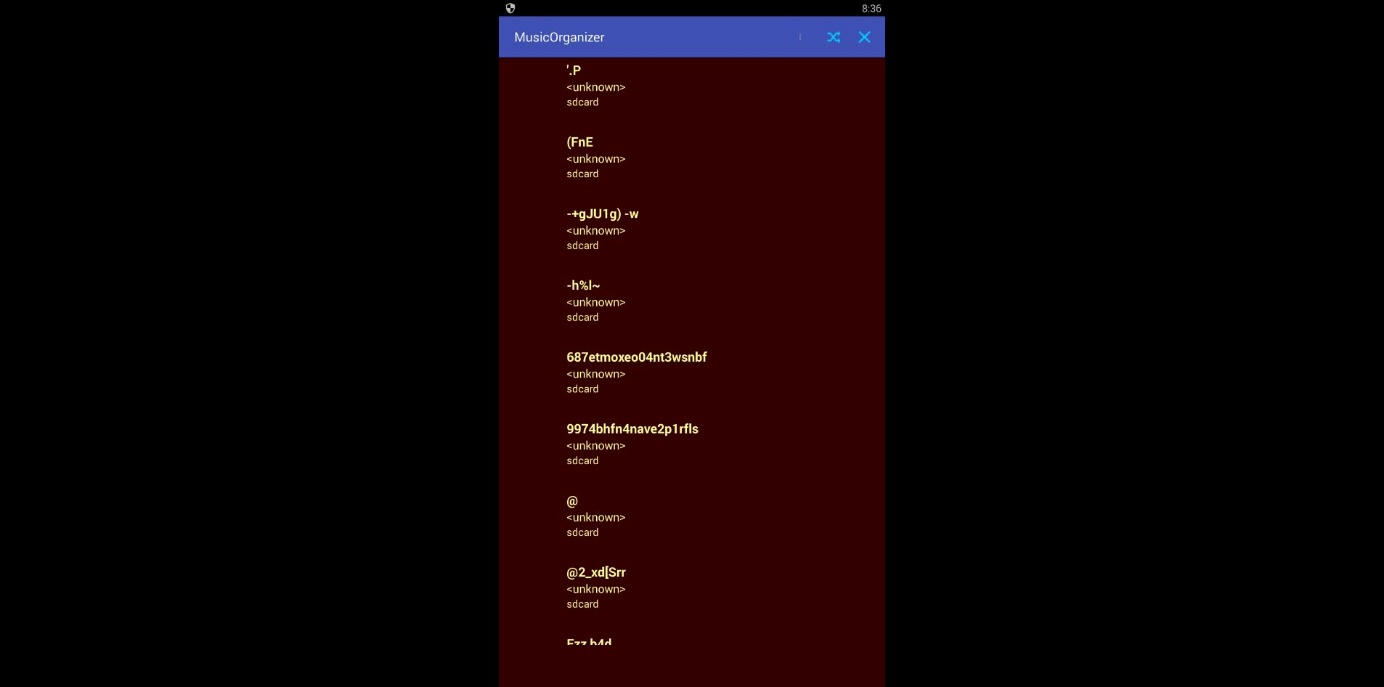
Сега са сортирани по име на песен:



Сега са сортирани по Албум:



Сега са сортирани по име на артист:



Заключение

Продукта който съм създал смятам, че е готов за употреба тъй като той предлага базова функционалност на един музикален органайзер. Отличава се с това че освен гледането на видео и слушането на музика дава възможност на потребителя да прави аудио запис като повечето музикални органайзери не предлагат такава функционалност. До колко е конкурентноспособен аз лично не мога да кажа понеже според мен продукта би трябвало да имплементира още функционалност като например : слушане на радио, връзка със съществуващ online магазин за продажба на песни, за да може потребителя да се сдобива с нова музика лесно и удобно. Със сигурност има много по-добри приложения от това което аз съм създал на пазара и ако трябва да гледаме реално на нещата това приложение трябва да имплементира още доста неща, за да може да се нарече конкурентоспособно.

Литература

1. <https://stackoverflow.com/>
2. <https://code.tutsplus.com/tutorials/create-a-music-player-on-android-project-setup--mobile-22764>
3. <https://examples.javacodegeeks.com/android/core/media/mediarecorder/android-audio-capture-example/>
4. <https://androidexample.com/Swipe_screen_left__right__top_bottom/index.php?view=article_discription&aid=95>
5. <https://www.tutorialspoint.com/android/android_studio.htm>

Приложение

MainActivity:

@Override  
protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
 //retrieve list view  
 songView = (ListView)findViewById(R.id.*song\_list*);  
 //instantiate list  
 songList = new ArrayList<Song>();  
 //get songs from device  
 getSongList();  
 //create and set adapter  
 SongAdapter songAdt = new SongAdapter(this, songList);  
 songView.setAdapter(songAdt);  
 //setup controller  
 setController();  
  
 detector = new GestureDetectorCompat(this, this);  
}

RecordSound Activity:

@Override  
protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_record\_sound*);  
 text = (TextView) findViewById(R.id.*text1*);  
  
 myRecorder = new MediaRecorder();  
 myRecorder.setAudioSource(MediaRecorder.AudioSource.*MIC*);  
 myRecorder.setOutputFormat(MediaRecorder.OutputFormat.*THREE\_GPP*);  
 myRecorder.setAudioEncoder(MediaRecorder.OutputFormat.*AMR\_NB*);  
  
 myPlayer = new MediaPlayer();  
  
 play\_stop\_start\_Btn = (ImageButton)findViewById(R.id.*play*);  
 play\_stop\_start\_Btn.setOnClickListener(new OnClickListener() {  
  
 @Override  
 public void onClick(View v) {  
  
 if (makeClickable) {  
 if (!isPlaying) {  
 record\_start\_stop\_Btn.setClickable(false);  
 record\_start\_stop\_Btn.setBackgroundResource(R.drawable.*record\_disabled*);  
 play();  
 play\_stop\_start\_Btn.setBackgroundResource(R.drawable.*stop*);  
 } else {  
 stopPlay();  
 record\_start\_stop\_Btn.setClickable(true);  
 record\_start\_stop\_Btn.setBackgroundResource(R.drawable.*record*);  
 play\_stop\_start\_Btn.setBackgroundResource(R.drawable.*playrec*);  
 }  
 isPlaying = !isPlaying;  
 }  
 }  
 });  
  
 record\_start\_stop\_Btn = (ImageButton)findViewById(R.id.*start*);  
 record\_start\_stop\_Btn.setOnClickListener(new OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if(!myPlayer.isPlaying()) {  
 if (!isRecording) {  
 outputFile = Environment.*getExternalStorageDirectory*().getAbsolutePath() + "/" + random() + ".3gpp";  
 myRecorder.setOutputFile(outputFile);  
 start(v);  
 record\_start\_stop\_Btn.setBackgroundResource(R.drawable.*stop*);  
 play\_stop\_start\_Btn.setClickable(false);  
 makeClickable=false;  
 play\_stop\_start\_Btn.setBackgroundResource(R.drawable.*disabled\_play*);  
 } else {  
 stop(v);  
 record\_start\_stop\_Btn.setBackgroundResource(R.drawable.*record*);  
 play\_stop\_start\_Btn.setClickable(true);  
 makeClickable=true;  
 play\_stop\_start\_Btn.setBackgroundResource(R.drawable.*playrec*);  
 }  
 isRecording = !isRecording;  
 }  
 }  
 });  
  
 try{  
 myPlayer.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {  
 @Override  
 public void onCompletion(MediaPlayer mp) {  
 stopPlay();  
 play\_stop\_start\_Btn.setBackgroundResource(R.drawable.*playrec*);  
 isPlaying = !isPlaying;  
 record\_start\_stop\_Btn.setClickable(true);  
 record\_start\_stop\_Btn.setBackgroundResource(R.drawable.*record*);  
 }  
 });  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 Slidr.*attach*(this);  
}

MP4PlayerActivity:

@Override  
protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_mp4\_player*);  
  
 //assign video  
 mVideosListView = (ListView) findViewById(R.id.*videoListView*);  
  
 try {  
 ArrayList<String> list=getAllMedia();  
 MediaMetadataRetriever retriever = new MediaMetadataRetriever();  
  
 for(String i : list)  
 {  
 retriever.setDataSource(this, Uri.*parse*(i));  
  
 String hasVideo = retriever.extractMetadata(MediaMetadataRetriever.*METADATA\_KEY\_HAS\_VIDEO*);  
 boolean isVideo = "yes".equals(hasVideo);  
 if(isVideo) {  
 mVideosList.add(new Video(i));  
 }  
 }  
  
  
 }catch (Exception e)  
 {  
 e.printStackTrace();  
 }  
  
 */\*\*\*populate video list to adapter\*\*/* mVideoAdapter = new VideoAdapter(this, mVideosList);  
 mVideosListView.setAdapter(mVideoAdapter);  
  
 Slidr.*attach*(this);  
}

Останалата важна част от кода е обяснена в предишната точка – Реализация. Примерите които са дадени там съдържат най-важните неща.