Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra informatiky a výpočetní techniky

# Bakalářská práce

# Záznamník přednášek pro platformu Android

Plzeň, 2015 Vitaliy Vashchenko

# Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných zdrojů.

V Plzni dne 7. května 2015

Vitaliy Vashchenko

# Poděkování

Chtěl bych poděkovat vedoucímu bakalářské práce Ing. Ladislavu Pešičkovi za odborné vedení a cenné rady při zpracování této práce.

# Abstract

The purpose of the work is to develop voice recording application with expanded bookmarks system. A user would be able to write additional information along with audio record. This application will provide recording and navigating through marked parts. In the first part I tested existing applications and described their main features. The second part is about working with audio on Android and the application implementation. The final product is a working application which provides audio recording and record labeling. The application is tested on virtual and real devices.

# Abstrakt

Cílem této práce je navržení a implementace aplikace, která umožňuje nahrávání audia a přidávání textových záložek. Aplikace umožnuje nahrávání přednášek a navigací mezi označenými částmi. V první části jsem zkoumal existující aplikace a jejich funkce. Druhá část popisuje práci s audiem na platformě Android. Konečná aplikace umožňuje zaznamenání audio a označení nahrávek. Aplikace byla otestována na virtuálním i reálném zařízení.

# Obsah

# 1 Úvod

Cílem této práce je prozkoumat vybrané aplikace pro mobilní zařízení, které poskytují podporu pro záznam hlasu včetně možnosti synchronizace s textovými poznámkami. Následujícím bodem bylo navržení a realizace aplikace pro nahrávání hlasových poznámek. Výsledná aplikace musí umožňovat hlasový záznam prezentace, označení důležitých pasáží a připsání poznámek.

V první části práce je stručný přehled systému Android, možností platformy a popis práce s médii. Analýza základních API zodpovědných za správu médií platformy Android.

Další částí práce je popis vybraných aplikací pro nahrávání hlasu, zkoumání jejich funkcionalit a vzhledu. Na základě předchozí analýzy je navržena aplikace splňující požadavky zadání a její uživatelské rozhraní.

V poslední částí je realizace navržené aplikace, popis použitých knihoven a testování funkcionality.

# 2 Platforma Android

Android – mobilní Open Source platforma podporovaná firmou Google a společností Open Handset Alliance, kterou tvoří největší výrobci mobilních zařízení, mobilní operátoři a výrobci polovodičů[1]. Od roku 2013 tvoří zařízení na platformě Android více než třetinu trhu mobilních telefonů[2]. V současné době přibližně 90 % zařízení funguje na verzi Android 4.1 a vyšší (viz Tabulka 1). Vzhledem k tomu jsem za minimální podporovanou verzi zvolil Android 4.1 API 16.

| Verze | Název | API | Rozložení |
|---|---|---|---|
| 2.2 | Froyo | 8 | 0.3% |
| 2.3.3-2.3.7 | Gingerbread | 10 | 5.7% |
| 4.0.3-4.0.4 | Ice Cream Sandwich | 15 | 5.3% |
| 4.1.x | Jelly Bean | 16 | 15.6% |
| 4.2.x | | 17 | 18.1% |
| 4.3 | | 18 | 5.5% |
| 4.4 | KitKat | 19 | 39.8% |
| 5.0 | Lolipop | 21 | 9.0% |
| 5.1 | | 22 | 0.7% |

**Tabulka 1 Rozložení verzi platformy Android[3]**

V roce 2015 přešla platforma Android na novou verzi Android Lolipop. Hlavní změnou v nové verzi je nový systémový styl Material Design[4]. Tento styl je určen k tomu, aby aplikace na platformě Android měly podobný vzhled a uživatel z toho měl lepší dojem. Přidané knihovny jsou zpětně kompatibilní a je možné nový styl použít ve starších verzích Androidu.

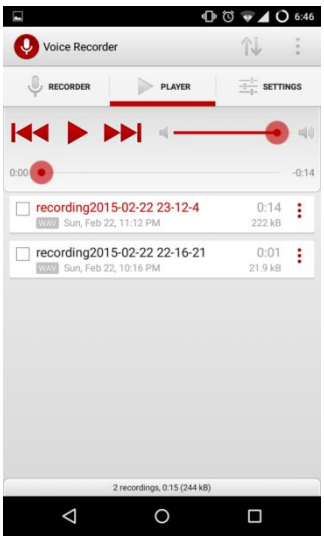
# 3 Existující aplikace a alternativní řešeni

Součástí práce je průzkum aktuální nabídky aplikací pro zaznamenávání zvuku a hodnocení jejich funkcionalit. Záznam zvuku patří mezi nejčastěji používané funkce telefonu. Ve většině případů má systém aplikaci od výrobce hardwaru, ale předem nainstalované aplikaci poskytují jen základní funkcionalitu.

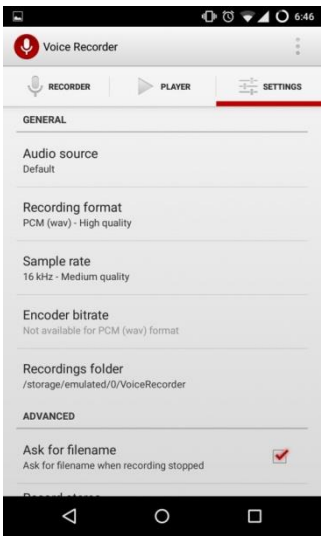
Počet aplikací určených k nahrávání hlasu na Android Play Marketu se pohybuje v řádu stovek. Prozkoumal jsem tři aplikace, z nichž každá má více než 100 tisíc instalací. Ke každé aplikaci jsou uvedeny podrobnosti v další části textu.

## 3.1 Voice Recorder (Splendid Apps)

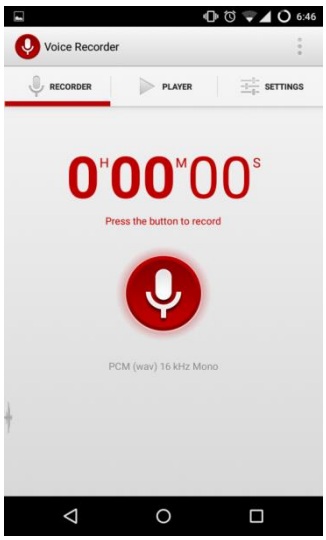
Uživatelské rozhráni aplikace je rozděleno na tři části. Jsou zobrazeny jako taby a navigace mezi nimi je pomocí tahu do stran (viz Obr. 3-1 až 3-3). První tab obsahuje rozhraní pro kontrolu nahrávání. Lze z něj spustit nové nahrávání, ukončit nahrávání a pojmenovat nahrávku. Dole je zobrazena aktuální amplituda zvuku. Nad tlačítkem s mikrofonem je časovač, který zobrazuje aktuální čas nahrávání. Další tab je určen pro přehrávání nahraných souboru. Soubory lze přejmenovávat, sdílet, smazat atd. Položky seznamu obsahují informaci o nahrávkách: formát, datum nahrávání, délka a velikost souboru. Aplikace neumožňuje přidávání poznámek nebo označení důležitých pasáží. Pod seznamem je zobrazen celkový počet nahrávek, jejích celková délka a velikost souboru. Poslední tab slouží pro zobrazení a změnu nastavení aplikace. Uživatel má možnost měnit zdroj zvuku, kvalitu a formát nahrávání, složku pro ukládání atd. K nastavením zápisu je navíc možné nastavit další vlastnosti aplikace: zastavení nahrávání v případě volání, LED oznámení a osvětlení displeje během zápisu.




**Obrázek 3-1 Hlavní fragment**



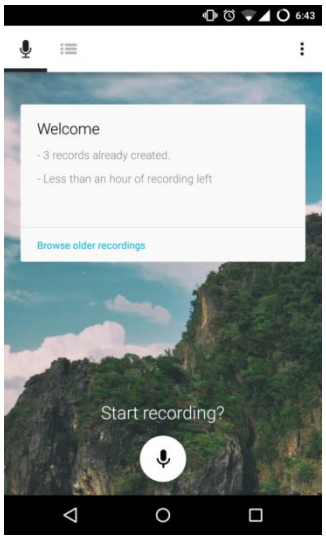

**Obrázek 3-2 Nastavení**



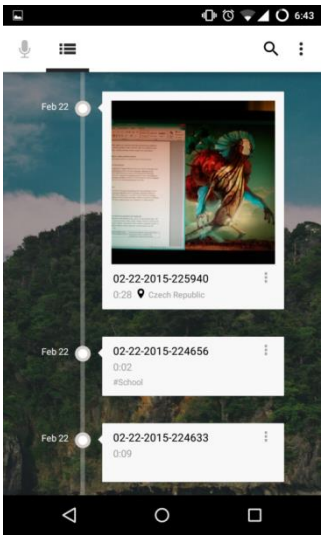

**Obrázek 3-3 Fragment nahrávače**

## 3.2 Skyro (Triveous)

Aplikace Skyro nabízí standartní funkcionalitu záznamníku zvuku, a navíc má pokročilejší zobrazení souboru. Uživatelské rozhraní je rozděleno do dvou tabů (Obr. 3-4 a 3-5). První tab slouží pro nahrávání, řízení průběhu nahrávání a zobrazení aktuální amplitudy. Druhý tab zobrazuje seznam nahraných souboru a umožňuje manipulaci s jednotlivými položkami. Po dokončení nahrávání má uživatel možnost přidat k nahrávce fotografie a ke každé nahrávce je přidělen štítek s lokací, kde byla daná nahrávka pořízena. Uživatelské rozhraní je tvořeno v minimalistickém stylu, který je dodržován v celé aplikaci. Seznam nahrávek je rozdělen podle data nahrávání. Nahrávky jsou zobrazeny jako části časové osy, každou položku lze editovat: přidat foto, změnit nebo přidat štítek atd. Důležitým rozdílem jsou široké možnosti manipulace s nahrávkami: přidávání štítků, lokací a obrázků, ale aplikace však neumožňuje označení důležitých částí nebo přidávání textových poznámek.



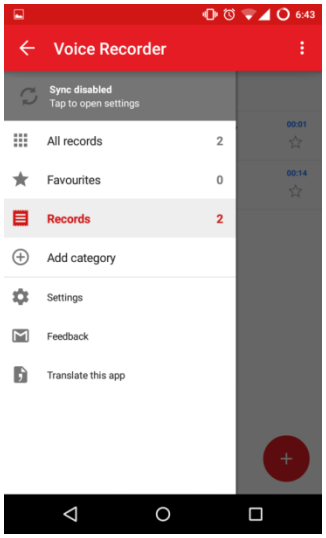

**Obrázek 3-4 Hlavní fragment**



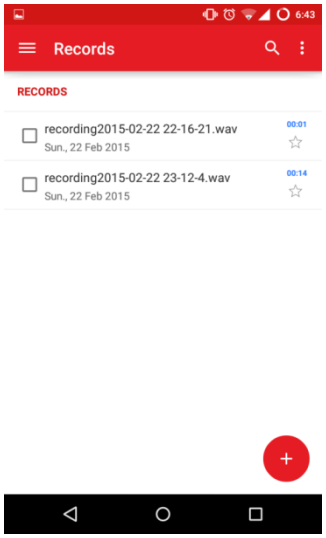

**Obrázek 3-5 Seznam nahrávek**

## 3.3 Voice Recorder (First75)

Aplikace poskytuje podobnou funkcionalitu jako předchozí (viz kap. 3.1 a 3.2), ale má jiný přístup k uživatelskému rozhraní aplikace. Navigace je založena na navigačním panelu - Navigation Drawer (Obr. 3-6). Hlavní fragment zobrazuje poslední záložky a umožňuje založení nových (Obr. 3-7). Fragment obsahující nástroje pro ovládání procesu nahrávání je zavolán stisknutím tlačítka v pravém dolním rohu. Pohybem od pravého kraje k centru displeje se vyvolá nabídka obsahující seznam kategorií nahrávek. Jednotlivé soubory lze přidávat do oblíbených, sdílet, přejmenovávat, mazat atd. V nastaveních lze měnit kvalitu a formát záznamu, přidat prefix a nastavit synchronizaci se vzdáleným úložištěm Dropbox nebo Google Drive.




Obrázek 3-6 Hlavní fragment




Obrázek 3-7 Seznam nahrávek

## 3.4 Další podobné aplikace

Další aplikace poskytují podobnou funkcionalitu a jen zřídka se vyčleňují speciálními funkcemi. Například aplikace Audio Recorder od Sony Mobile Communication slouží pro práci s externím mikrofonem a podporuje ovládání pomoci chytrých hodinek.

Po průzkumu dostupných aplikací jsem měl větší představu o tom, co aplikace musí umět a jak musí vypadat uživatelské rozhraní. Označení důležitých částí a přidávání poznámek jsou žádoucí funkcionalitou, ale žádná ze zkoumaných aplikací toto neumožňuje. Nejlepší možnosti poskytuje aplikace Skyro (viz kap 3.2). Ostatní aplikace umožnují maximálně sdílení a manipulaci se souborem nahrávky.

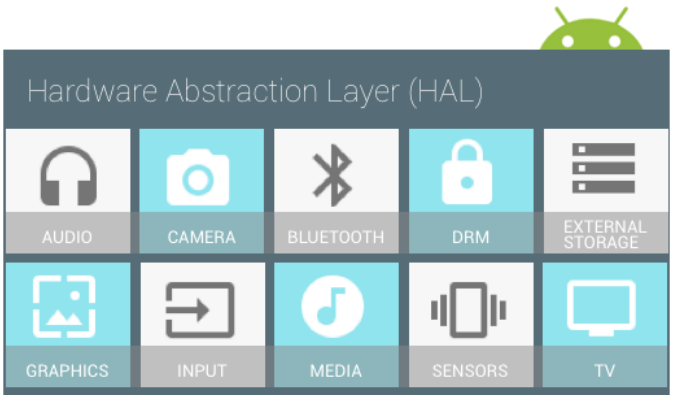
# 4 Návrh aplikace

Cílem aplikace je možnost nahrávání přednášek a jiných prezentací a označení důležitých částí záznamu při jeho pořizování. Vzhledem k tomu, že průměrná doba trvání přednášky je kolem dvou hodin, má označení důležitých informací pro konečného uživatele velký přínos. Důležité pasáže mohou být několika druhů: informace o zkoušce nebo zápočtu, důležité termíny, vysvětlení složitých části látky apod., proto musí aplikace podporovat několik typů záložek. Popis záložky zlepšuje pochopení kontextu a usnadňuje orientaci mezi nimi. Pří přehrávání nahrávek urychlují patřičné záznamy navigaci mezi důležitými částmi.

Standardní knihovny platformy Android umožňují vývoj aplikací pro základní účely, mezi které patří nahrávání audio souborů. Případné poznámky budou zapsány do vlastních souborů a při zobrazení seznamu poznámek je bude nutno načíst do aplikace. Aplikace bude umět nahrávat zvuk, přidávat různé poznámky, přehrávat nahrané soubory a pracovat s vytvořenými záložkami. Analýza se dále rozděluje do pěti částí: nahrávání audio souboru, ukládání poznámek, přehrávání souboru, zpracování vytvořených poznámek a uživatelské rozhraní. Před vlastním programováním aplikace jsem provedl rozbor toho, jak a za jakých okolností bude uživatel aplikaci používat.

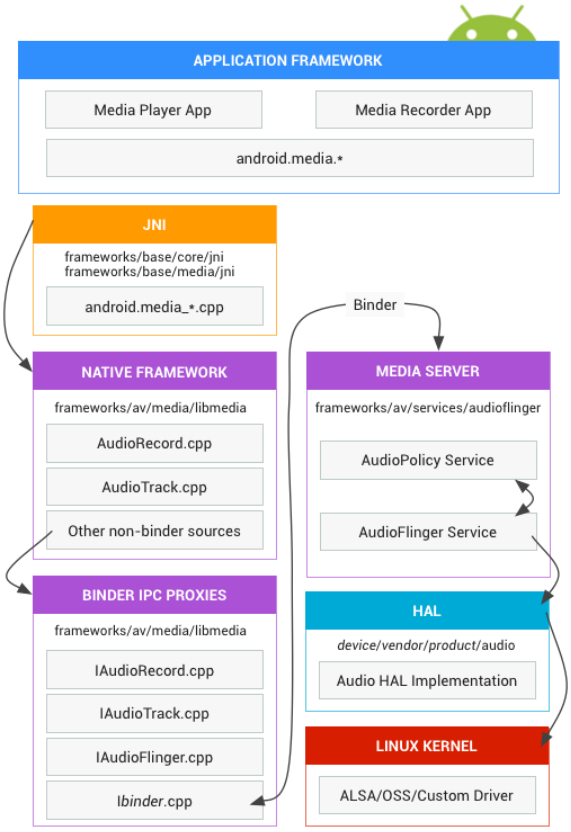
## 4.1 Práce s audiem na platformě Android

Architektura platformy je postavena na abstraktní spolupráci mezi systémem a hardwarem. Jednotlivé části patří do vrstvy, která se jmenuje **Hardware Abstraction Layer** (Obr. 4-1), tj. hardwarová abstrakční vrstva. Je to rozhraní obsahující ovladače a knihovny navržené výrobcem hardwaru. Položky dané vrstvy představují různé možnosti, které hardware může systému poskytnout. Do vrstvy patří například média, audio, práce s kamerou, práce se senzory atd.




**Obrázek 4-1 Android HAL [5]**

Manipulace s audiem na platformě Android má vlastní architekturu a je rozdělena do několika částí (Obr. 4-2). Programátor nemusí se starat o práce systému s hardwarem a proto pracuje výhradně s API android.media. Postupně se přes několik úrovní dostaneme k samotnému jádru systému. Stupně mezi API a jádrem převádí systémový kód na nativní a přes HAL v jádru je vyvolána příslušná hardwarová funkce. V práci s audiem spolupracuje jádro prostřednictvím HAL nebo alternativních architektur, například Advanced Linux Sound Architecture (ALSA), Open Sound System (OSS) nebo prostřednictvím vlastní HAL.




**Obrázek 4-2 Struktura Audio HAL [6]**

API poskytuje prostředky pro nahrávání a přehrávání audio/video souborů a skládá se z několika tříd, jež jsou používány pro specifické účely, např.: **AudioManager**, **AudioRecord**, **MediaRecorder**, **MediaPlayer** atd. API navíc zpracovává bitmapy pro detekci obličeje (třída **FaceDetector**), směrování audio a řízení upozornění (třída **AudioManager**). Proto, aby aplikace získala přístup ke zdrojům, je nutné vyžádat opatřeni v souboru AndroidManifest.xml. V ukázce kódu 1 je zobrazena část manifestu aplikace obsahující potřebná povolení.

Použití API je v Javě, ale knihovny, na nichž je systém založen, jsou implementovány v jazyce C nebo C++. Například *setDataSource()* nebo *start()* třídy MediaPlayer pod Javou jsou psány v C/C++ a přeloženy do souborů s příponou .so. O

spojení mezi vrstvami se stará JNI (Java Native Interface), který se používá pro vnoření Java kódu do systému s nativním kódem[7].

```
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"
/>
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

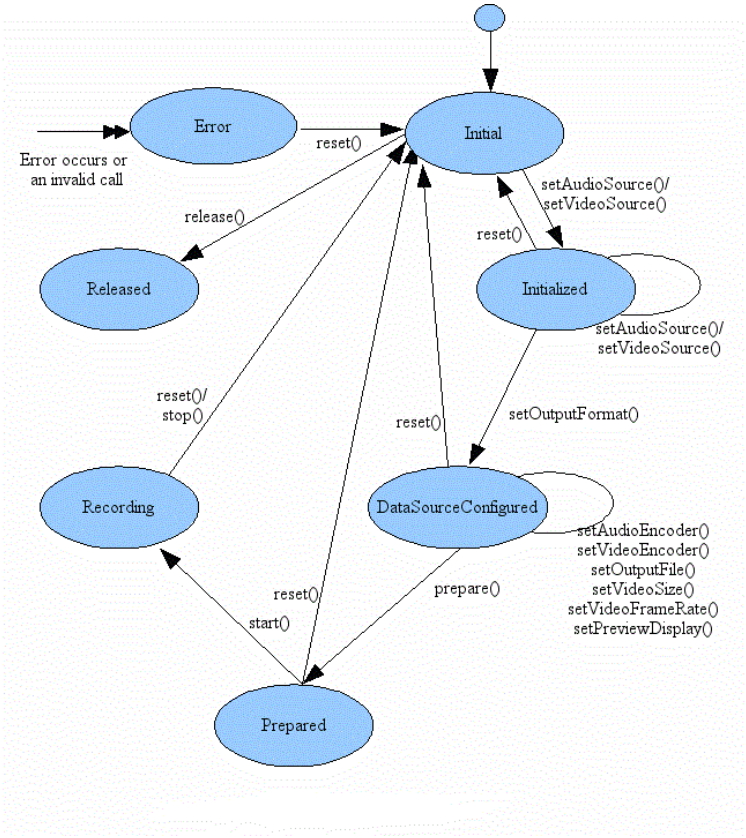
**Ukázka kódu 1 Povolení v manifest souboru**

## 4.2 Záznam zvuku

Standardní možnosti systému nabízejí několik variant pro nahrávání zvuku. Liší se v přístupu k nahrávanému souboru a stupněm volnosti při napsání kódu.

### 4.2.1 MediaRecorder

Třída MediaRecorder se používá pro nahrávání audia a videa. Před nahráváním je nutno nastavit vlastnosti nového objektu: zdroj, kvalitu a enkodér. Řízení je založeno na konečném automatu a rekordér se může nacházet v jednom určitém stavu (Obr. 4-3)[8].




**Obrázek 4-3 Stavový graf třídy MediaRecorder**

Po vytvoření nebo přenastavení je objekt ve stavu Initial. Dále se nastavuje zdroj zvuku a objekt přechází do stavu Initialized, kde programátor určuje výstupní soubor. Další fází je nastavení vlastností nahrávky (Tabulka 2) a příprava k nahrávání. Po dokončení metody *prepare()* je objekt připraven a voláním metody *start()* je zahájen proces nahrávání. V případě chyby se stav rekordéru změní na Error a vypíše se chybové hlášení. Po dokončení nahrávání voláním metody *release()* se uvolní přidělené zdroje.

MediaRecorder funguje podle principu blackbox, a proto se souborem nelze během nahrávání manipulovat. Data lze analyzovat až po dokončení zápisu do souboru. Standardní možnosti třídy neumožňují pauzy a jejich podpora musí být implementována programátorem nebo pomoci použití externích knihoven. Výhodou dané třídy je jednoduchá implementace (Ukázka kódu 2). Nahrávky lze uložit do formátů 3gp, mp4, acc a amr. Nevýhodou je nedostupnost dat při nahrávání.

```
MediaRecorder recorder = new MediaRecorder();
// nastavení zdroje
 recorder.setAudioSource(MediaRecorder.AudioSource.MIC);
// nastavení kontejneru
 recorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
// nastavení enkodéru
 recorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
// nastavení místa pro výstupní soubor
 recorder.setOutputFile(PATH_NAME);
// příprava objektu
 recorder.prepare();
// zahájení procesu nahrávání
 recorder.start();
 ...
// ukončení nahrávání
 recorder.stop();
 recorder.release();
// uvolnění alokovaných zdrojů
```

**Ukázka kódu 2 Zahájení procesu nahrávání**

| Vlastnost | Popis | Možné varianty |
|---|---|---|
| setAudioChannels() | Nastavení audio kanálů | **1** – mono<br>**2** – stereo |
| setAudioEncoder() | Nastavení audio enkodéru | **DEFAULT** – implicitní enkodér<br>**AAC** – AAC jednodušší audio kodek<br>**AAC_ELD** - AAC kodek s nízkou latenci<br>**AMR_NB** – AMR úzký kodek<br>**AMR_WB** – AMR široký kodek<br>**HE_AAC** – AAC kodek vyšší kvality |
| setEncodingBitRate() | Nastavení kvality enkodéru | Hodnota z rozmezí vybraného enkodéru |
| setAudioSamplingRate() | Nastavení kvality výchozího souboru | Hodnota z rozmezí vybraného enkodéru (AAC: 8-96kHz, AMRNB: 8kHz, AMRWB: 16kHz) |
| setAudioSource() | Nastavení zdroje nahrávání | **DEFAULT** – implicitní mikrofon<br>**CAMCODER** – mikrofon se stejnou orientaci s kamerou<br>**MIC** – standartní mikrofon<br>**VOICE_CALL** – nahrávání volaní |
| setOutputFormat() | Nastavení formátu výchozího souboru | **DEFAULT** – implicitní kontejner<br>**AAC_ADTS** – AAC kontejner<br>**AMR_NB/WB** - kontejner pro AMR<br>**MPEG_4** – kontejner pro MPEG4<br>**THREE_GPP** – kontejner pro 3GPP |

**Tabulka 2 Vlastností nahrávky MediaRecorder**

### 4.2.2 AudioRecorder

Třída AudioRecord byla přidána v API ver. 3. Používá se pro zpracovávání dat v Java aplikacích. Nahrává data a ukládá je do vlastního zásobníku. Nahraná data je nutno periodicky vyjímat ze zásobníku. Vyjmutá data ukládáme do vlastního buferu, který použijeme pří zápisu do konečného souboru. Programátor musí implementovat vyjímání dat z objektu, nastavit vlastnosti nahrávky a podle toho určit velikost

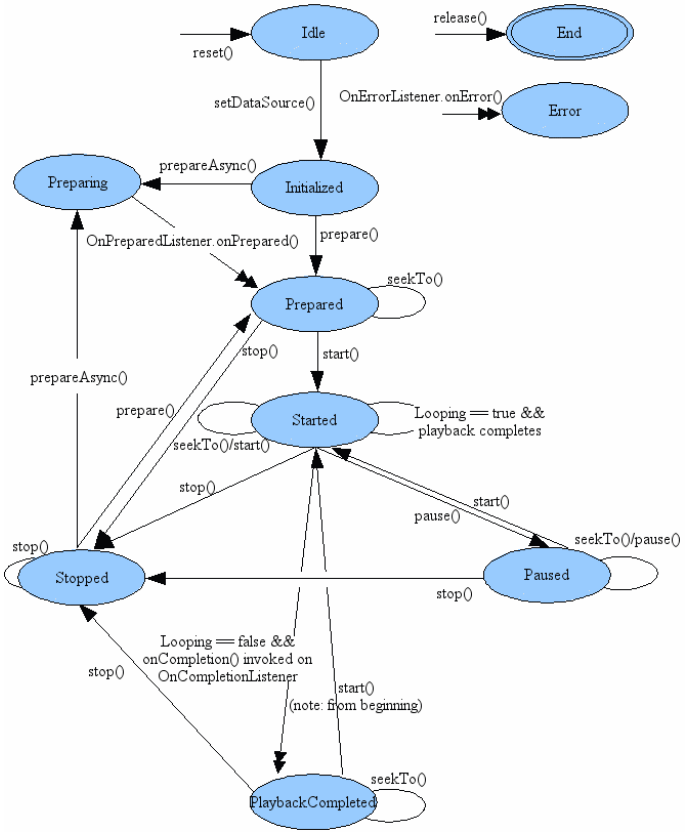
zásobníku. Konečný soubor je PCM (pulzně kódová modulace) nahrávky a pro další použitelnost ji musíme zabalit do media kontejneru. Nevýhodou je jak komplikovanější realizace procesu nahrávání, tak i balení do kontejneru. Výhodou je větší kontrola nad procesem a možnost data během nahrávání analyzovat.

## 4.3 Přehrávání audio souborů

Platforma Android podporuje velkého množství formátů audio souborů a poskytuje nástroje pro manipulaci s nimi. Jde jak o lokální zdroje média, tak i o načtené soubory ze vzdálených serveru. Základem přehrávání médií jsou dvě třídy android.media API : **MediaPlayer** a **AudioManager**. První slouží pro přehrávání audia a videa, druhá třída řídí zdroje zvuku a jeho výstup. Pro samotné přehrávání audio souborů existuje několik možností.

### 4.3.1 MediaPlayer

MediaPlayer poskytuje základní možnosti přehrávání zvuku a videa. Umí pracovat nejen s lokálními daty ale i s data z internetu. Objekt přestavuje konečný automat (Obr. 4-4) a může se nacházet v jednom z několika stavů (viz Tabulka 3) [9].




**Obrázek 4-4 Stavový graf třídy MediaPlayer**

| Stav | Popis stavu |
|---|---|
| Idle | Stav po volání metody **reset().** Objekt je v původním stavu. |
| Initialized | Objekt dostal zdroj přes **setDataSource().** Objekt nastaven. |
| Prepared | Po volání metody prepare(), objekt je připraven k přehrávání. |
| Preparing | Stav během přípravy. |
| Started | Přehrávání zdrojového souboru. |
| Stopped | Zastavení přehrávače. |
| Paused | Pauza přehrávače. |
| PlaybackCompleted | Konec souboru. |
| Error | Chybový stav. |
| End | Uvolnění přidělených zdrojů. |

**Tabulka 3 Stavy MediaPlayer**

Před přehráváním musí přehrávač načíst data do paměti aplikace, dekódovat je a následně dekódovaná data přehrát nebo zobrazit na displeji. Načtená data jsou ve speciálním zásobníku a jejich obsah je postupně zpracováván příslušným dekodérem. Struktura přehrávače je založena na hierarchii skládající se z daných tří částí[10].

Před spuštěním je nutno nastavit zdrojový soubor, uvést jeho druh a následně zavolat metodu *prepare()***.** Metodou *isPlaying()* lze zkontrolovat, zda objekt je ve stavu Started. Nastavením *setLooping()* se nastavuje přehrávání ve smyčce. Pomoci metody *seekTo()* měníme pozici přehrávání. Tato metoda je dostupná ve stavech Started**,** Prepared**,** Paused a PlaybackCompleted. V případě chyby se stav rekordéru změní na Error a vypíše se chybové hlášení. Odkaz na zdrojový soubor se předává ve tvaru řetězce nebo URI souboru v síti. Důležitou částí je pořadí při inicializací přehrávače. Volání metody *prepare()* je nezbytnou částí a musí být zavoláno před začátkem přehrávání. Nevýhodou je závislost na stavech a nestálá doba přípravy v případě přehrávání vzdáleného souboru. Dobu čekání lze zkrátit asynchronní přípravou *prepareAsync()*. Výhodou jsou jednoduchost použití (Ukázka kódu 3) a podpora velkého množství formátů a typů médií, např. mp3, mp4, aac, amr, 3gpp apod.

```
String url = "http://........"; // odkaz na vzdálený soubor
MediaPlayer mediaPlayer = new MediaPlayer();
// nastavení typu media
mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
mediaPlayer.setDataSource(url); // nastavení na zdrojový soubor
mediaPlayer.prepare(); // připravení (načítání, kontrola atd.)
mediaPlayer.start();// zahájení přehrávání
```

**Ukázka kódu 3 Inicializace instance třídy MediaPlayer**

### 4.3.2 ExoPlayer

ExoPlayer se používá jako alternativa MediaPlayeru. Jde o externí knihovnu s otevřeným zdrojovým kódem. MediaPlayer poskytuje jednoduchá řešení pro

přehrávání souboru a není zaměřen na vzdálené soubory. ExoPlayer je zaměřen na online přehrávání a proto podporuje několik funkcí navíc. Podporuje DASH(Dynamic adaptive streaming over HTTP), SmoothStreaming a trvalé kešování. Od verze API 18 (Android 4.3) podporuje přehrávání DRM(Digital Rights Management) chráněných souborů.

## 4.4 Uložení poznámek

Důležitou částí je přidávání záložek jejich a následné použití. Jednou z možností realizace daného systému je uložení záložek do souborů. Pro umožnění další práce musí mít soubory definované formátování. Jazyk XML je široce používaný standard, proto jsem se rozhodl ukládat informace o záložkách do XML soborů.

Android poskytuje několik možnosti vytváření XML souborů, jejichž přístup záleží na velikosti a složitosti výsledného souboru.

### 4.4.1 Formátování řetězce

Nejjednodušší řešení je vlastnoruční formátování řetězce. Tento způsob potřebuje funkci s ručně rozepsaným formátem souboru. Výhodou jsou jednoduchost, rychlost a nezávislost na speciálních knihovnách. Nevýhodou jsou nedostatek funkcí pro práci s XML souborem a problémy při dynamickém formátování (Ukázka kódu 4).

```
public static String writeUsingNormalOperation(Study study) {
    String format =
            "<?xml version='1.0' encoding='UTF-8'?>" +
            "<record>" +
            "   <study id='%d'>" +
            "       <topic>%s</topic>" +
            "       <content>%s</content>" +
            "       <author>%s</author>" +
            "       <date>%s</date>" +
            "   </study>" +
            "</record>";
    return String.format(format, study.mId, study.mTopic, study.mContent, s
tudy.mAuthor, study.mDate);
```

**Ukázka kódu 4 Formátování řetězce**

### 4.4.2 DOM

DOM (Document Object Model) se používá pro práci s XML, HTML a XHTML soubory, jejích vytvoření, úpravu a zpracovávání. Jde o mezinárodní specifikaci, kterou používají prohlížeče. Model DOM byl vytvořen organizací W3C a v současné době se používají verze 2 a 3. Výhodou je podpora dynamického výstupu a flexibilita XML

souborů. Nevýhoda spočívá ve velkém počtu potřebných objektů a v horším výkonu při generaci větších souborů (Ukázka kódu 5).

```
public static String writeUsingDOM(Study study) throws Exception {
    Document doc = DocumentBuilderFactory.newInstance().newDocumentBuilder(
).newDocument();  // vytvoření nového dokumentu
    Element root = doc.createElement(Study.RECORD);  // vytvoření kořenu
    doc.appendChild(root);  // přidání kořenu do dokumentu

    Element tagStudy = doc.createElement(Study.STUDY);  // vytvoření
dalšího elementu
    root.appendChild(tagStudy);  // přidání elementu do dokumentu
    tagStudy.setAttribute(Study.ID, String.valueOf(study.mId));  //
nastavení atributů elementu

     ...

    Transformer transformer = TransformerFactory.newInstance().newTransform
er();
    StringWriter writer = new StringWriter();
    StreamResult result = new StreamResult(writer);  // inicializace
výstupu
    transformer.transform(new DOMSource(doc), result);

    return writer.toString();  // vracení vytvořeného řetězce
}
```

**Ukázka kódu 5 Formátování podle DOM**

### 4.4.3 XMLSerializer

Jde o třídu, která poskytuje jednoduché nástroje pro formátování XML souborů (Ukázka kódu 6). Výhodou jsou jednoduchost implementace a menší paměťová náročnost v případě větších souborů. Žádné nevýhody použití dané knihovny jsem nenašel.

```
public static String writeUsingXMLSerializer(Study study) throws Exception
{
    XmlSerializer xmlSerializer = Xml.newSerializer();  // inicializace
instancí
    StringWriter writer = new StringWriter();

    xmlSerializer.setOutput(writer);
    xmlSerializer.startDocument("UTF-8", true);  // založení dokumentu
    xmlSerializer.startTag("", Study.RECORD);  // vytvoření nového tagu
    xmlSerializer.startTag("", Study.STUDY);  // vytvoření tagu uvnitř
předchozího
    xmlSerializer.attribute("", Study.ID, String.valueOf(study.mId));  //
nastavení atributů
    ...
    xmlSerializer.endTag("", Study.STUDY); // uzavření tagů
    xmlSerializer.endTag("", Study.RECORD);

    xmlSerializer.endDocument();  // ukončení dokumentu

    return writer.toString();  // vrácení řetězce
}
```

**Ukázka kódu 6 Formátování řetězce použitím XMLSerializer**

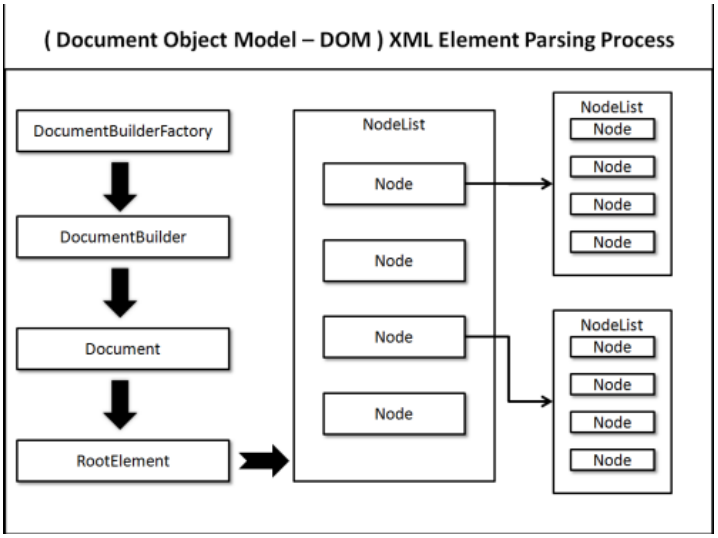
Formátování řetězce není přizpůsobeno pro generování rozšiřitelných souborů, proto je lepší použít knihovny, které jsou k tomu určeny. Po prozkoumání vlastností dvou knihoven jsem se rozhodl použít **XMLSerializer**, hlavním důvodem byla menší paměťová náročnost a jednoduchost použití.

## 4.5 Zpracování souborů se záložkami

Pro parsování XML souboru na platformě Android lze použít několik knihoven. Standardní knihovny Java a Android poskytují tři knihovny na výběr: **SAX** Parser, **DOM** Parser a **Pull** Parser.

### 4.5.1 DOM Parser

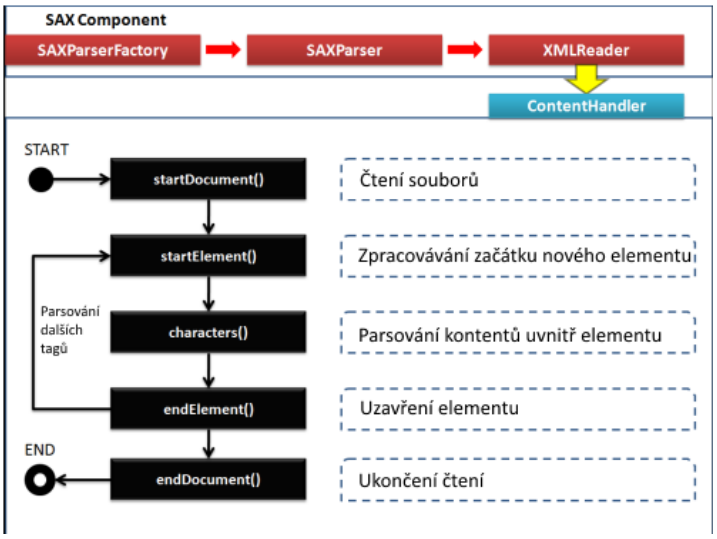
DOM Parser vytváří objekty pro každý uzel XML souboru a z vytvořených objektů se staví strom dokumentu (Obr. 4-5). Při parsování je nutno kontrolovat, zda objekt není potomkem předchozího objektu, a proto DOM parser má horší výkon při práci s většími soubory.




**Obrázek 4-5 Proces parsování DOM parseru**

## 4.5.2 SAX Parser

SAX (Simple API for XML) používá události na základě uzlů. Prochází dokumentem ve smyčce a řídí se podle událostí, které z dokumentu dostane (Obr. 4-6). Existuje pět variant: *startDocument()* — začátek dokumentu, *startElement()* — začátek elementu, *characters()* — vlastnosti elementu, *endElement()* — konec elementu a *endDocument()* — konec dokumentu. Programátor implementuje zpracování jednotlivých událostí v příslušných metodách parseru . Použitím **SAX** lze vytvořit **DOM** stromy a naopak. Standardy **DOM** a **SAX** jsou vzájemně nahraditelné[11].




**Obrázek 4-6 Postup parsování SAX parseru**

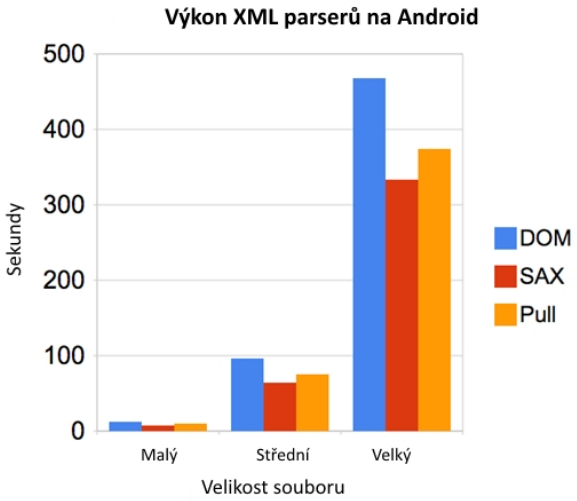
### 4.5.3 XMLPullParser

XmlPullParser je podobný parseru **SAX** (viz kap. 4.5.2). Místo události má stavy dokumentu. Existuje pět stavů: **START_DOCUMENT**, **END_DOCUMENT**, **START_TAG**, **END_TAG** a **TEXT**. Podobně jako u **SAX** parseru určuje programátor případný postup při změně stavu (Ukázka kódu 7).

```
switch (eventType){  // pokračování podle události
                    case XmlPullParser.START_DOCUMENT:  // začátek
dokumentu
                        messages = new ArrayList<Message>();
                        break;
                    case XmlPullParser.START_TAG:  // začátek tagu
                        name = parser.getName();  // zpracovávání podle
jména tagu
                        if (name.equalsIgnoreCase(ITEM)){
                            currentMessage = new Message();
                        } else if (currentMessage != null){
                            if (name.equalsIgnoreCase(LINK)){
                                currentMessage.setLink(parser.nextText());
                            }
                        }
                        break;
                    case XmlPullParser.END_TAG:  // konec tagu
                        name = parser.getName();
                        if (name.equalsIgnoreCase(ITEM) &&
currentMessage != null){  // uzavření určitého tagu
                            messages.add(currentMessage);
                        } else if (name.equalsIgnoreCase(CHANNEL)){
                            done = true;
                        }
                        break;
```

**Ukázka kódu 7 Příklad parsování XMLPullParserem**

Předpokládá se, že aplikace bude pracovat z větším počtem jednoduchých XML souborů. **DOM** parser má horší výkon oproti **SAX** a **XMLPull** parserům (Obr. 4-7)[12]. Rozdílem mezi **XMLPullParser** a **SAX** parser je přístup ke zpracovávání částí souborů. **SAX** je určen pro rychlé parsování komplikovanějších XML souborů, proto mým konečným rozhodnutím bylo použít **XMLPullParser**.




**Obrázek 4-7 Výkon parserů na platformě Android**

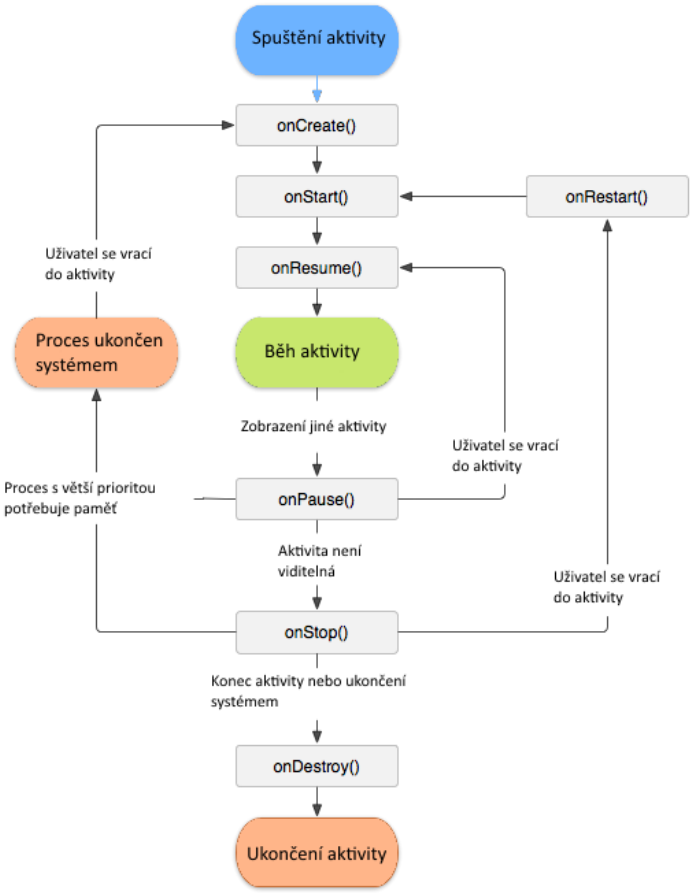
## 4.6 Uživatelské rozhraní

Neméně důležitou částí práce je uživatelské rozhraní aplikace, protože s ním bude konečný uživatel pracovat a kvalita vzhledu má velký vliv na uživatelský dojem při práci s programem.

### 4.6.1 Activity

Hlavní částí aplikace na Androidu jsou Aktivity, které spojují vzhled a funkcionalitu aplikace[13]. Každá aktivita má vlastní životní cyklus (Obr. 4-8) a může se nacházet v určitém stavu. Aktivita představuje okno, které zaplňuje plochu cele aplikace, zobrazuje se jako menší okno na jiné aktivitě nebo je částí aktivity. Každá aktivita slouží pro konkrétní účel a má vlastní vzhled. Většina aktivit a potomků třídy Activity implementuje dvě důležité metody: *onCreate()* a *onPause()*. První slouží pro inicializaci nové instance, kde jsou definovány části aktivity a její celkový vzhled. V podstatě se spojují části návrhu z XML souboru a samotná instance aktivity. V souboru AndroidManifest.xml programátor určuje, která aktivita je hlavní. Hlavní aktivita bude vytvořena a vyvolána při startu aplikace. V manifestu je nutno definovat i ostatní aktivity, ale jejich instanci musí vytvořit programátor v kódu. Metoda *onPause()* je zavolána při opuštění aktivity a obvykle se používá pro ukládání data uživatele. Aktivity procházejí celým životním cyklem a mohou být zrušeny systémem v případě konfiguračních změn, například při otočení displeje.

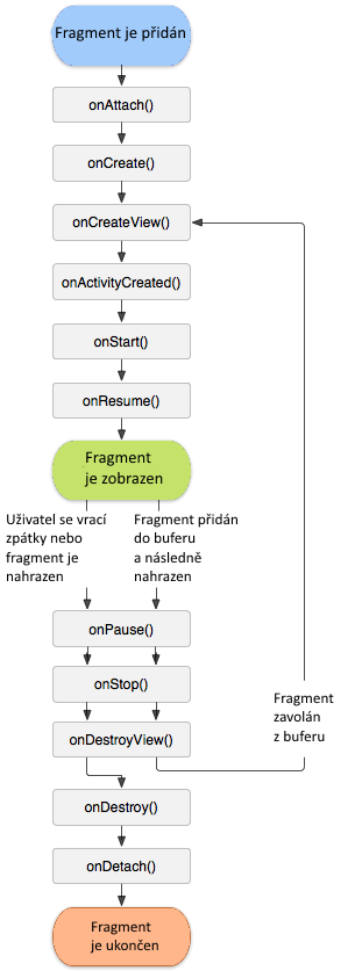
Volání aktivit a jejich změny v aplikaci jsou prováděny pomocí třídy **Intent**[14]. **Intent** slouží pro volání určitých aktivit nebo pro zobrazení seznamu aktivit podle účelu. Další důležitou funkcí je předání dat aktivitě před jejím voláním příkazem *Intent.putExtras()*.




**Obrázek 4-8 Životní cyklus aktivit**

### 4.6.2 Fragment

Ve verzi Android 3.0 byly přidány fragmenty. **Fragment** – náhrada aktivit, která je určena pro uživatelské rozhraní aplikace a je lépe škálovatelná vůči displeji zařízení. Aktivita muže obsahovat několik fragmentů najednou a programátor má možnost je dynamicky měnit[15]. Fragmenty jsou brány jako části aktivit a jejích životní cykly (Obr. 4-9) záleží na cyklech nadřízených aktivit. Při pauze aktivity jsou podřízené fragmenty zastaveny také. Obecně je fragment považován za část aplikace, kterou lze dynamický měnit, mazat a opětovně vytvářet. Životní cyklus fragmentu je podobný cyklu aktivity. Stejně jako aktivita má fragment několik základních metod: *onCreate()*, *onCreateView()* a *onPause()*. Metoda *onCreate()* je zavolána v době inicializace objektu, *onCreateView()* před zobrazením a *onPause()* při opuštění fragmentu. Fragment má vlastní vzhled a funkcionalitu. Použitím několika fragmentů lze realizovat rozhraní ve více oknech. Manipulacím s fragmenty se říká transakce a jejích provedení má pod kontrolou FragmentManager aplikace. Po zavedení fragmentů nepřestaly být aktivity nezbytnou částí programů a každá aplikace musí mít minimálně jednu aktivitu pro umístění fragmentů.




**Obrázek 4-9 Životní cyklus fragmentů**

Cílová aplikace je více účelová a proto musíme použít více než jeden fragment. Pro navigaci mezi několika fragmenty existuje několik možností. Nejpopulárnější jsou dvě: Taby a Navigační panel.

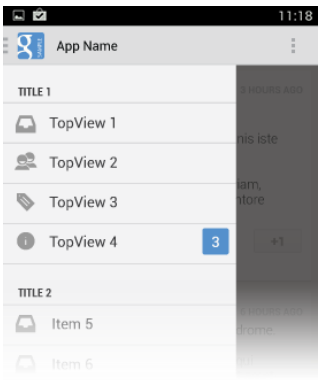
### 4.6.3 Tabs

Taby jsou jednotlivé položky v menu, které se typicky zobrazují v horní části aplikace (Obr. 4-10). Změny fragmentu jsou vyvolány pohybem vlevo nebo vpravo. Podobné pohyby se jmenují Swipy. Jsou to gesta, určená pro manipulaci s uživatelským rozhraním. Taby jsou částí Swipe vzhledů. Používají se pro horizontální navigaci mezi souvisejícími fragmenty aplikace. Jsou dvě možnosti použití tabů: posuvný bar nebo bar s nastavenou velikostí. První je určen pro aplikace s větším počtem fragmentů patřících do jednoho **Swipe View**. Taby s určenou velikosti jsou určeny pro navigaci mezi třemi nebo méně fragmenty.


**Obrázek 4-10 Taby**

### 4.6.4 Navigation Drawer

Navigation Drawer (Navigační panel), je dalším příkladem navigace mezi několika fragmenty aplikace. Pohybem od kraje displeje k centru bude vyvolán panel obsahující položky aplikace (Obr. 4-11). Program může obsahovat maximálně dva panely. Hlavní panel musí být v levém okraji. Další částí daného stylu je Action Bar nahoře. Akční bar slouží pro interakce s vybraným fragmentem. Správně navržený navigační panel je přístupný z libovolného místa v aplikaci.




**Obrázek 4-11 Navigační panel**

V poslední době je trendem používat navigační panel místo tabů. Navigační panely poskytují rychlejší navigaci a zabírají méně místa. Místo určené pro zobrazení tabů je použito akčním barem, který nabízí víc možností pro navržení uživatelského rozhraní. Proto jsem pro navigaci zvolil navigační panel. Po prozkoumání možností, které nabízí platforma Android, jsem se rozhodl použít jednu aktivitu s několika fragmenty. Konečný návrh vzhledu je ovlivněn doporučeními od Googlu a novými trendy na trhu.

# 5 Použitý software

Při navrhování aplikace jsem se řídil oficiálními doporučeními firmy Google, a to jak ohledně struktury, tak i uživatelského rozhraní projektu. Za IDE jsem si vybral Android Studio, oficiální IDE pro platformu Android.

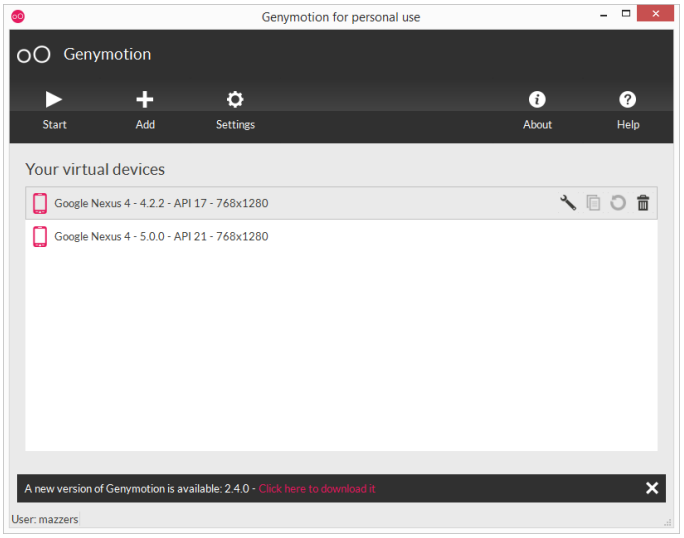
## 5.1 Android Studio

Android Studio je oficiální multiplatformní prostředí pro vývoj aplikací pro systém Android. První verze byla anoncována v květnu 2013 na konferenci Google I/O. V prosinci 2014 se dostala do stabilní verzi 1.0. Aktuální verze je 1.1.0. Aplikace Android Studio je založeno na IDE od IntelliJ IDEA JetBrains, určené pro vývoj aplikací v Javě. Od základní verze se liší větším počtem nástrojů určených k programování pro systém Android. Studio má velkou bázi hotových šablon, poskytuje nástroje pro odchycení paměťových úniků a nabízí bohatý editor pro navržení uživatelského rozhraní. Mimo již uvedené možnosti, nabízí lepší integritu se servisy od Google a spravování hesel a generování spustitelných apk souboru.

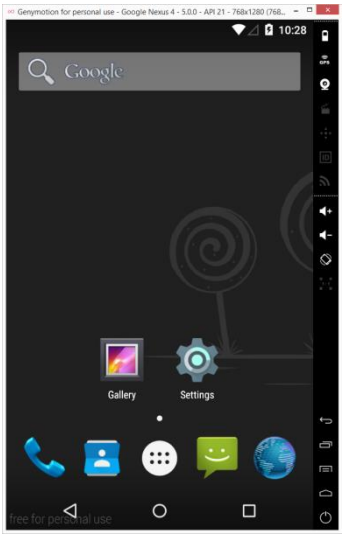
IDE potřebuje JDK (Java Development Kit), JRE (Java Runtime Environment) a Android SDK (Software Development Kit). Android SDK je součástí Android Studio a pomocí vnitřního SDK manažeru lze nainstalovat potřebnou verzi SDK. Pro testování aplikací se používá AVD manažer. Je to emulátor, který umožňuje testování na různých verzích Androidu a na různých verzích hardwaru. Emuluje mobilní zařízení s různými vlastnostmi: platforma CPU, velikost displeje, dostupná paměť atd.

## 5.2 Genymotion

Genymotion je alternativa AVD manažeru poskytující lepší výkon, jednodušší práci a více možnosti při ladění programu. Genymotion je založen na vitalizačním nástroji Oracle VM VirtualBox. Základní rozdíl a příčina lepšího výkonu je nahrazení emulováni virtualizací. Software má dvě verze - svobodnou verzi pro osobní účely a verzí s periodickou měsíční platbou pro podnikání. Placená verze poskytuje více možností při ladění. Program umožňuje vytvoření několika cílových zařízení a jejích spravování v manažeru zařízení (Obr. 5-1). Spuštěný systém se zobrazí ve vlastním okně (Obr. 5-2). Je možné spustit několik systémů najednou. Genymotion se snadno integruje do Android Studio nainstalováním příslušného rozšíření.



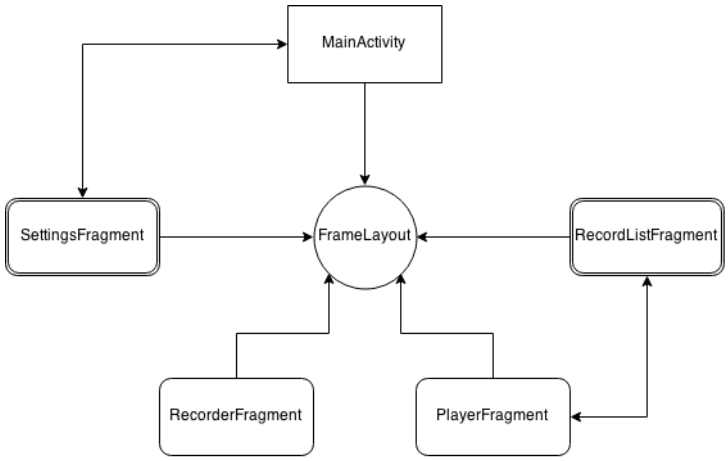

**Obrázek 5-1 Správce virtuálních zařízení**




**Obrázek 5-2 Spuštěný virtuální systém**

# 6 Implementace

Při navrhování aplikace jsem se snažil používat standardní možnosti Androidu. Aplikace jsem rozdělil na dva hlavní fragmenty: **RecorderFragment** a **PlayerFragment** a dva pomocné: **RecordListFragment** a **SettingsFragment**. Aktivita obsahuje FrameLayout kontejner, který umisťuje fragmenty aplikace (Obr. 6-1). V Inkscape jsem vytvořil ikonu aplikace a za ikony uživatelského rozhraní jsem použil sadu bezplatných ikon[16]. Během navrhování vzhledu jsem se snažil dodržovat minimalistický styl. Veškeré procesy, nesouvisející s uživatelským rozhraním, jsem navrhl tak, aby běžely ve vlastních vláknech a nepřetěžovaly vzhled aplikace.




**Obrázek 6-1 Propojenost aktivity a fragmentů**

Všechny částí uživatelského rozhraní jsou založeny na dvou částech: XML souboru se vzhledem (Ukázka kódu 8) a Java souboru, obsahujícím kód, který tento vzhled používá (Ukázka kódu 9).

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

    android:orientation="vertical"  // nastavení orientace rozložení
    android:layout_width="match_parent"
    android:layout_height="match_parent">  // nastavení velikosti rozložení

    <ListView  // vnoření seznamu do rozložení
        android:layout_width="wrap_content"  // nastavení velikosti
        android:layout_height="wrap_content"
        android:id="@+id/recordsListView"></ListView>

</RelativeLayout>
```

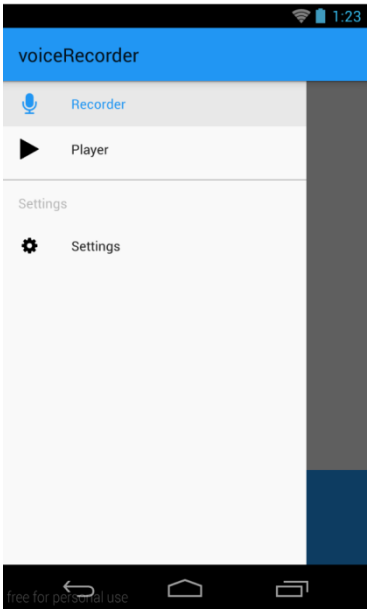
**Ukázka kódu 8 Příklad XML souboru se vzhledem**

```
public View onCreateView(LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) {
    View rootView = inflater.inflate(R.layout.record_list_layout, container
, false);  // hlavní nastavení hlavního rozložení na prvek z XML
    listView = (ListView) rootView.findViewById(R.id.recordsListView);  //
nastavení seznamu na příslušný prvek xml souboru
    recordListAdapter = new RecordListAdapter(rootView.getContext(),
mFileList);  // nastavení data pro adapter seznamu
    listView.setAdapter(recordListAdapter); // určení adapteru
    return rootView; // návrat konečného vzhledu
}
```

**Ukázka kódu 9 Použití vzhledu v kódu**

## 6.1 Hlavní aktivita

Hlavní aktivita slouží jako kontejner pro fragmenty aplikace. Součástí aktivity jsou navigační panel a FrameLayout, určený pro umístění fragmentů. Aktivity aplikace musejí být uvedené v manifestu aplikace. Při startu aktivita vytvoří fragmenty a proběhne načtení dat ze složky aplikace. V případě, že složka neexistuje, aplikace ji vytvoří. Následně vytvoří navigační panel s položkami, které volají příslušné fragmenty. Pro navigační panel jsem použil externí knihovnu **MaterialDrawer**, která poskytuje panel se vzhledem podle standardů Material Design. Material Design – systémový styl Androidu od verze 5.0 (Obr. 6-2).




**Obrázek 6-2 Navigační panel MaterialDrawer**

Panel slouží pro rychlou navigací mezi fragmenty. Každý fragment má vlastní ikonu. Za ikony v panelu jsem použil sadu ikon, kterou nabízela knihovna MaterialDrawer. Při nastavení panelu lze nastavit průhlednost status baru. Při spuštění

otevře panel první položku – rekordér. Při hardwarové změně je stav aktivity uložen do speciálního objektu typu **Bundle**. Nově vytvořená aktivita dostane tento objekt na vstup a z něj obnoví svůj stav na původní (Ukázka kódu 10).

```
@Override
protected void onSaveInstanceState(Bundle outState) {  // uložení stavu
    outState.putInt("selected", result.getCurrentSelection());  // vložení
pozice panelu do objektu
    super.onSaveInstanceState(outState);
}

@Override
protected void onRestoreInstanceState(Bundle savedInstanceState) {
    super.onRestoreInstanceState(savedInstanceState);
    int curr = savedInstanceState.getInt("selected", 1);  // přečtení
předchozí pozice z objektu
    result.setSelection(curr, false);  // obnovení pozice
}
```

**Ukázka kódu 10 Navrat aktivity do původního stavu**

Kliknutím na položku panelu bude zavolán příslušný fragment (Ukázka kódu 11). Fragment bude vyhledán pomocí tagu, který je nastaven při přidání fragmentu do aplikace. Vyhledávání fragmentů v paměti je realizováno metodou *getFragmentByTag()* instance **FragementManager**. Metoda vrací objekt třídy Fragment a je nutno ho přetypovat na odpovídající třídu. V případě, kdy FragmentManager objekt nenajde, vrátí *null* a objekt je vytvořen aplikací. Před zobrazením volaného fragmentu jsou viditelné fragmenty skryty. V případě, že volaný fragment je již viditelný, nenastane nic. Změny ve FragmentManageru jsou brány jako transakce a musejí byt ukončeny příkazem *commit()*.

```
public void toggleRecorder() {
    FragmentTransaction ft = getFragmentManager().beginTransaction();
    RecorderFragment temp = new RecorderFragment(); // vytvoření nového
fragmentu
    ft.replace(R.id.container, temp, RecorderFragment.RECORDER_TAG); //
zaměňování aktuálního fragmentu
    ft.commit(); // volání transakce
}
```

**Ukázka kódu 11 Volání fragmentu RecorderFragment**

Aktivita je spojovacím článkem pro fragmenty a umisťuje společná data, která fragmenty používají. Při startu aplikace je zavolán loader **BookmarksLoader**, který načte záložní soubory ze složky aplikace. Aktivita realizuje rozhraní pro prací s loaderem. Do rozhraní patří tři metody:

- *onCreateLoader* – vrací příslušný loader
- *onLoadFinished* – výsledek práce loaderu je zpracováván v aktivitě
- *onLoaderReset* – postup při resetování loaderu

Aktivita obsahuje rekordér a přehrávač a nově vytvořené fragmenty aktualizují svůj vzhled podle jejich stavu.

### 6.1.1 BookmarksLoader

Platforma Android má několik možností realizace vláken. Jednou z variant je třída Loader. Práce instance je rozdělena do několika částí. Hlavní proces, který zabírá nejvíce práce, běží ve vlastním vlákně a jeho výsledek je zpracováván ve vlákně uživatelského rozhraní. BookmarksLoader načítá soubory do paměti a vytváří objekty třídy **Bookmark**. Objekty jsou následně rozděleny do ArrayListů v HashMapě, jejímž klíčem je cesta k audio souboru (Ukázka kódu 12). HashMapa je uložena do hlavní aktivity pro další použití fragmentů.

```
String groupName = bookmarkArrayList.get(0).getPath(); // první soubor –
první skupina
int groupCount = 1;  // počet skupin (souborů)
listDataHeader.add(groupName);  // přidání prvního souboru

     for (int i = 0; i < bookmarkArrayList.size(); i++) {
          if (!bookmarkArrayList.get(i).getPath().equals(groupName)) {
              mItems.put(listDataHeader.get(groupCount -
 1), tempArray);  // přidání nové skupiny v případě jiného audio souboru
              groupCount++;  // zvětšení počtu skupin
              tempArray = new ArrayList<>();  // nový list pro skupinu
              groupName = bookmarkArrayList.get(i).getPath(); // skupina
podle cesty k souboru
              listDataHeader.add(groupName); // přidání skupiny do seznamu
               }
          tempArray.add(bookmarkArrayList.get(i)); // přidání do předchozí
skupiny
       }
mItems.put(listDataHeader.get(groupCount - 1), tempArray); // přidání
poslední skupiny do HashMapy
```

**Ukázka kódu 12 Třídění záložek**

### 6.1.2 Bookmark

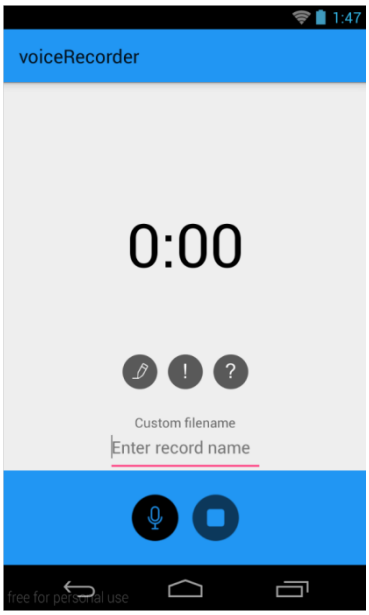
Objekty třídy představují jednotlivé záložní soubory vytvořené uživatelem. Obsahují název souboru, čas záložky, druh a případný text (Ukázka kódu 13).

```
public Bookmark(String path, String bookmarkPath, String fileName, int time
, String message, int type) {
    this.path = path;  // cesta k audio souboru
    this.bookmarkPath = bookmarkPath;  // cesta k záložce
    this.time = time;  // čas záložky
    this.fileName = fileName;  // název audio souboru
    this.message = message;  // zpráva záložky
    this.type = type;  // druh záložky
}
```

**Ukázka kódu 13 Konstruktor třídy Bookmark**

## 6.2 Fragment RecorderFragment

Fragment RecorderFragment představuje hlavní fragment aplikace. Je určen k vytvoření nahrávek a nových záložek. Rozhraní je představeno několika tlačítky a časovačem, který slouží pro zobrazení délky nahrávání (Obr. 6-3). Podle životního cyklu fragmentu je nejdříve zavolána metoda *onCreate()*, ve které je nastaven fragment s nastavením aplikace. Následně se v metodě *onViewCreated()* spojují části rozhraní s XML souboru s kódem. Stav rozhraní záleží na stavu nahrávače, o to se stará metoda *changeButtonsState()*. V závislostí na okamžitém stavu jsou tlačítka vypnuta nebo zapnuta.




**Obrázek 6-3 RecorderFragment**

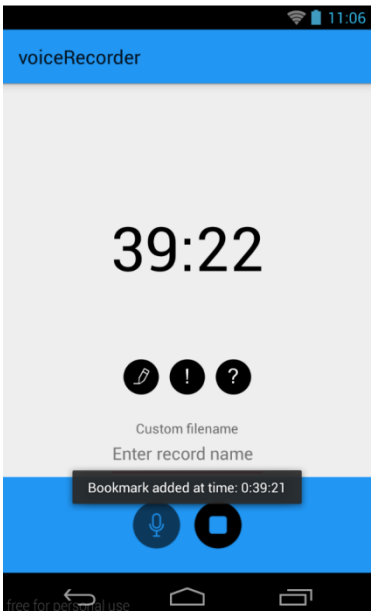
Kliknutím na tlačítko s mikrofonem bude zahájen nový proces nahrávání. Pro nahrávání jsem použil třídu **MediaRecorder**. Kliknutím na patřičné tlačítko lze nahrávání zahájit nebo ukončit. Uživatel může zadat název souboru. Jestli uživatel název nezvolí, je soubor pojmenován podle času zahájení nahrávání. Nahrávka se

vytvoří na základě nastavení aplikace, které uživatel může měnit. Před nahráváním nastaví **MediaRecorder** kvalitu a formát souboru na příslušné hodnoty (Ukázka kódu 14). Nejdříve jsou uvolněny zdroje, které alokoval předchozí nahrávač. Dále pokračuje nastavení nové instance: enkodér, kvalita nahrávky, formát nahrávky atd.

```
mediaRecorder.release();  // uvolnění alokovaných zdrojů
mediaRecorder = new MediaRecorder();  // nová instance
mediaRecorder.setAudioSource(MediaRecorder.AudioSource.DEFAULT); //
nastavení zdroje
mediaRecorder.setOutputFormat(formatChoice);  // formát podle nastavení
uživatele
mediaRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);  //
enkodér
mediaRecorder.setAudioEncodingBitRate(16);  // kvalita enkodéru
mediaRecorder.setAudioSamplingRate(qualityChoice);  // kvalita nahrávky
mediaRecorder.setOutputFile(filePathAudio); // konečný soubor
```

**Ukázka kódu 14 Zahájení nahrávání**

Během nahrávání má uživatel možnost přidávat záložky několika druhů. Každý druh má vlastní ikonu. Po kliknutí je zavoláno vlákno, které vytvoří nový záložní soubor. V případě textové záložky se otevře dialog, do kterého má uživatel možnost napsat vlastní poznámky. Uživatel je informován o nově vytvořené záložce prostřednictvím plovoucí zprávy – **Toast** (Obr. 6-4). Kvůli tomu, že rekordér není uzpůsoben delším nahráváním, je délka nahrávání omezená na 2 hodiny.




**Obrázek 6-4 Oznámení o nové záložce**

### 6.2.1 WriteToXML

Je to vlákno, které formátuje vstupní hodnoty a zapisuje je do XML souborů. Konstruktor dostává název souboru, čas záložky, druh atd. a předává je **XmlSerializeru** (Ukázka kódu 15). Po formátování je vytvořen nový soubor obsahující veškeré informace potřebné pro další zpracovávání.

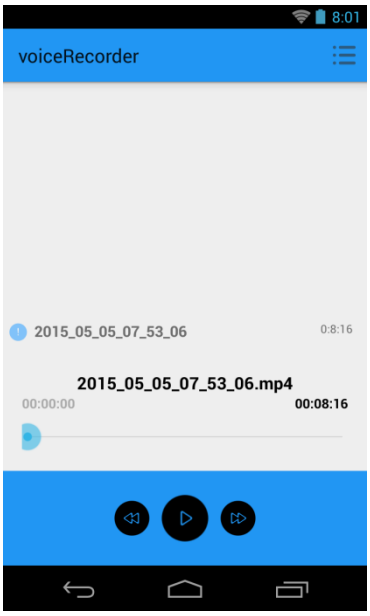
```
public WriteToXML(File fileBook, long duration, int type, String message,
String fileAudioName, String filePathAudio) {
       this.fileBook = fileBook;  // nastavení hodnot podle příchozích
       this.duration = duration;
       this.message = message;
       this.type = type;
       this.fileAudioName = fileAudioName;
       this.filePathAudio = filePathAudio;
     }
private String WriteDataToXML() throws IOException {
       XmlSerializer xmlSerializer = Xml.newSerializer();  // formátování
souboru a vyplnění atributů
       StringWriter writer = new StringWriter();
       xmlSerializer.setOutput(writer);
       xmlSerializer.startDocument("UTF-8", true);
       xmlSerializer.startTag("", "root");
       xmlSerializer.startTag("", "path");
       xmlSerializer.attribute("", "value", filePathAudio);
       xmlSerializer.endTag("", "path");
       ...
       xmlSerializer.endTag("", "root");
       xmlSerializer.endDocument();
       return writer.toString();  // naformátovaný řetězec
   }
```

**Ukázka kódu 15 Formátování údajů**

## 6.3 Fragment PlayerFragment

Fragment je určen k přehrávání nahraných souborů. Vzhled je rozdělen do dvou částí. Uprostřed jsou zobrazeny záložky patřící aktuálně přehrávanému souboru. Dole jsou tlačítka pro ovládání přehrávání. Během nahrávání jsou položky s odpovídajícím časem označeny jinou barvou (Obr. 6-5). Při otevření nového souboru vyhledá aplikace patřičné záložky a zobrazí je v seznamu nad tlačítky. **MediaPlayer** není optimalizován pro velké soubory a načtení může trvat několik sekund v závislosti na velikosti souboru. Proto, aby načtení neovlivňovalo běh fragmentu, je příprava volána asynchronně. Po změně aktuálního souboru je zavolána metoda *prepareAsync()*, která načte soubor v jiném vlákně a po dokončení předá soubor aplikaci. Dokončení načtení sleduje *onPreparedListener()*. Během načítání jsou tlačítka fragmentu neaktivní a je

zobrazen **ProgressDialog**. V momentě, kdy aplikace načte soubor, jsou aktivována tlačítka ovládaní a je aktualizován vzhled fragmentu.




**Obrázek 6-5 PlayerFragment**

Vyhledávání záložek je provedeno prostřednictvím vyhledávání cesty k souboru v HashMapě (Ukázka kódu 16). Patřičné záložky jsou přidány do seznamu **ListView** nad tlačítky. Po kliknutí na položku seznamu je přehrávání přemístěno na čas záložky. Dlouhým kliknutím na záložku je vyvoláno menu, jehož pomocí lze záložku smazat. Po změně dat je o tom adaptér seznamu informován a aktualizuje vzhled.

```
if (mItems.containsKey(path)) {  // hledání záložek podle cesty k souboru
    bookmarkArrayList.clear();  // mazání starého seznamu
    bookmarkArrayList.addAll(mItems.get(name)); // přidání vyhledaných
záložek
    listViewAdapter.notifyDataSetChanged();  // oznámení adapteru o změně
dat
```

**Ukázka kódu 16 Vyhledávání záložek**

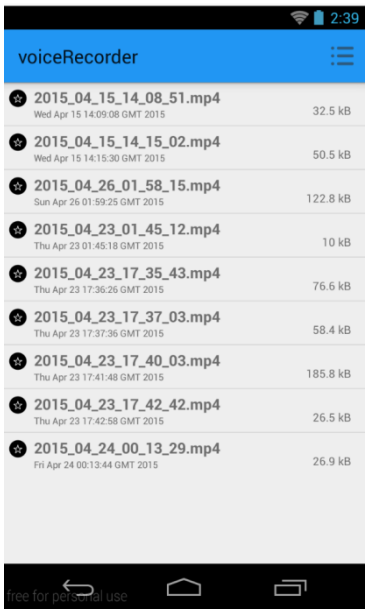
Paralelně s přehráváním běží dvě vlákna, která aktualizují vzhled fragmentu. Jedno vlákno aktualizuje čas přehrávání (Ukázka kódu 17), druhé sleduje, zda nenastal čas záložky ze seznamu. Čas v **MediaPlayeru** je měřen v Long a pro formátování long do řetězců jsem vytvořil pomocnou třídu **Utils**, která obsahuje metody pro formátování data, času a velikosti souborů do řetězců.

```
long totalDuration = mediaPlayer.getDuration();  // celková délka souboru
long currentDuration = mediaPlayer.getCurrentPosition();  // aktuální
pozice

songTotalDurationLabel.setText("" + Utils.milliSecondsToTimer(totalDuration
));
songCurrentDurationLabel.setText("" + Utils.milliSecondsToTimer(currentDura
tion));  // nastavení textu na aktuální hodnoty

int progress = (Utils.getProgressPercentage(currentDuration, totalDuration)
);
```

**Ukázka kódu 17 Aktualizace vzhledu PlayerFragment**

## 6.4 Fragment RecordListFragment

Fragment slouží pro zobrazení nahraných souborů. Položky obsahují tyto údaje: datum vytvoření, velikost, atd. (Obr. 6-6). Základním elementem fragmentu je seznam **ListView**, který obsahuje pole souborů ve složce aplikace. Data se do seznamu umisťují pomocí speciálního adaptéru **recordListAdapter**. Daný adapter zpracovává jednotlivé soubory a nastavuje vzhled položek podle jejich detailu. Detaily se vkládají do vnitřní třídy-kontejneru **ViewHolder**. Po kliknutí na položku je vyvolán fragment s přehrávačem. Po dlouhém kliknutí se otevře menu, ze kterého lze smazat soubor a jeho záložky. Při otevření fragmentu je zavolán loader **FilesLoader**, který načte soubory s příponou mp4 ze složky aplikace.




**Obrázek 6-6 RecordListFragment**

### 6.4.1 FilesLoader

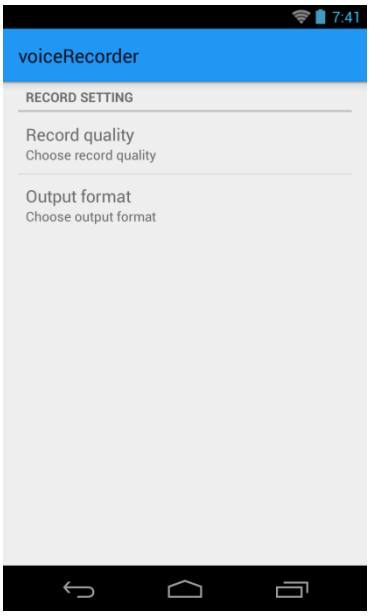
Je to loader, který načítá mp4 a 3gpp soubory ze složky aplikace (Ukázka kódu 18). Loader ignoruje prázdné soubory. Po dokončení procesu předá pole souboru. Pracuje ve vlastním vlákně a proto nezpomaluje vlákno uživatelského rozhraní.

```
List<File> parseDir(File newDir){  // načítání souborů ze složky
    List<File> fileList = new ArrayList<>();
    if (newDir.listFiles(new FileExtensionFilter()).length > 0){
        for (File file : dir.listFiles(new FileExtensionFilter())){
            if (file.length()>0){
                fileList.add(file);  // přidání neprázdných souborů
            }
        }
    }
    return fileList;  // vrácení seznamu nahrávek
}
```

**Ukázka kódu 18 Práce FilesLoaderu**

## 6.5 SettingsFragment

Fragment obsahující nastavení aplikace. Umožňuje nastavit kvalitu a formát nahrávky (Obr. 6-7). Nastavení aplikace jsou ukládána do **SharedPreferences**, které rekordér načítá před zahájením nahrávání (Ukázka kódu 19).



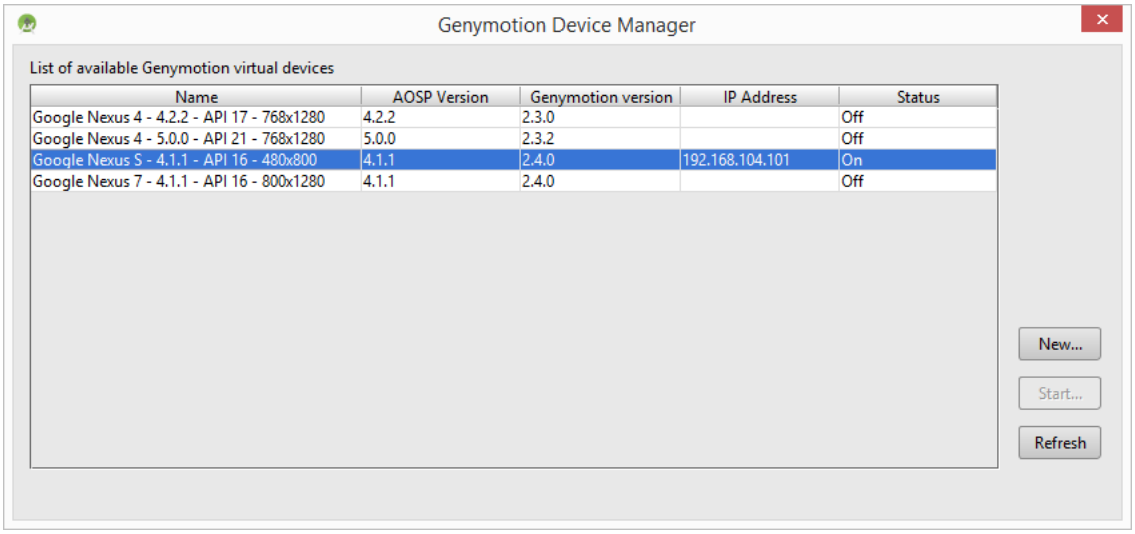

**Obrázek 6-7 SettingsFragment**

```
String quality = sharedPreferences.getString("quality", "2");
String format = sharedPreferences.getString("format", "2");
int qualityChoice = quality.equalsIgnoreCase("2") ? 44100 : 22050;  //
kvalita a formát podle nastavení aplikace
int formatChoice;
if (format.equalsIgnoreCase("mp4")) {
       formatChoice = MediaRecorder.OutputFormat.MPEG_4;
       filePathAudio = RecordsFolder + fileAudioName + ".mp4";
 } else {
       formatChoice = MediaRecorder.OutputFormat.THREE_GPP;
       filePathAudio = RecordsFolder + fileAudioName + ".3gpp";
        }
```

**Ukázka kódu 19 Vytvoření rekordéru podle nastavení**

# 7 Testování aplikace

Pro testování aplikace jsem použil manažer virtuálního zařízení Genymotion. Funkčnost aplikace byla otestována na verzích systému android 4.1.1, 4.2.2 a 5.0 v emulátoru a na verzích 5.0 a 5.1 s použitím reálného zařízení (Obr. 7-1). Zařízení Nexus S má 512 RAM MiB a rozlišení displeje 480x800. Tablet Nexus 7 má displej 800x1280 a 1024 MiB paměti. Za reálné zařízení byl použit telefon Nexus 4 s posledním dostupným systémem (Android 5.1 API 22). Displej Nexus 4 je 768x1280, velikost paměti zařízení je 2048 MiB.




**Obrázek 7-1 Seznam zařízení**

## 7.1 Funkčnost aplikace

Aplikace umožňuje nahrávání audio souborů a přidávání poznámek paralelně s nahráváním. Maximální délka nahrávání je 2 hodiny. Nahrané soubory a vytvořené záložky zpracovává a poskytuje nástroje pro jejich přehrávání. Velikost nahraného souboru záleží na vybrané kvalitě a délce nahrávání. Otevření souborů s délkou větší než hodina trvá přibližně 2 sekundy. Funkčnost aplikace jsem ověřil nahráváním audio souborů s délkou od 1 minuty do 2 hodin. Během nahrávání jsem náhodně přidával poznámky a záložky. Při přehrávání odpovídaly vytvořené záložky časovým hodnotám a jejich detaily se shodovaly s původními.

## 7.2 Možnost rozšíření aplikace

Nejobtížnější částí aplikace je přehrávání souborů s délkou více než dvě hodiny. MediaPlayer není uzpůsoben k přehrávání souborů s větší délkou. Aplikaci bych urychlil použitím externí knihovny nebo navržením vlastního přehrávače. Samotné procesy přehrávání a nahrávání lze předělat na objekty třídy **Service** a tehdy by nebyly tak závislé na aktivitě aplikace. Použitím externích knihoven lze přidat nahrávání mp3 souborů.

Uživatelské rozhraní a práce s fragmenty je další možností optimalizace. Aktualizace uživatelského rozhraní a přidání dalších detailů o nahrávce, například obrázků, lokací nebo tagů. Vhodné by bylo také přidání panelu do StatusBaru a několik widgetů.

# 8 Závěr

V rámci práce jsem prozkoumal záznamníky pro platformu Android. Detailněji jsem poznal, jak systém pracuje s médií a jaké možnosti nabízí standardní knihovny Androidu. V době psaní práce přešla platforma Android na novou verzi a při návrhu aplikace jsem se řídil novými doporučeními.

Výsledkem mé práce je záznamník, který poskytuje nástroje pro uložení poznámek a označení různých části nahrávky. Aplikace je zaměřena na studenty, kteří si potřebují zaznamenávat přednášky a lekce. Vytvořené záložky jsou určeny pro lepší pochopení nahraného materiálu a rychlou navigaci ve větších nahrávkách. Při implementaci jsem použil jednu pomocnou knihovnu, která rozšiřuje základní možnosti navigačního panelu a celkového vzhledu aplikace.

Zaměření aplikace umožňuje další vylepšování a přidání nových funkci, které by byly užitečné pro konečného uživatele. Nedostatek aplikací podporujících označení důležitých pasáží znamená, že daná aplikace bude žádána uživateli, kterým tato možnost chybí.

# Použité termíny a zkratky

- **API** (Application Programming Interface) – rozhraní pro programování aplikací.
- **AVD manager** (Android Virtual Device Manager) – manažer virtuálních zařízení Android.
- **DRM** (Digital Rights Managements) – souhrn metod, které ovládají nebo omezují ovládaní kontentu v souladu s autorskými právy.
- **Dropbox** – online servis pro ukládání dat.
- **Google Drive** – online servis pro ukládání dat od společnosti Google.
- **HTML** (HyperText Markup Language) – značkovací jazyk, standard sítě internet.
- **Layout** – rozložení instance na obrazovce.
- **URI** (Uniform Resource identifier) – řetěz který se odkazuje na zdroj v síti.
- **W3C** (World Wide Web Consortium) – mezinárodní konsorcium, jehož členové vyvíjejí standardy pro WWW.
- **XML** (Extensible Markup Language) – značkovací jazyk používaný pro uložení dat.
- **XHTML** (Extensible hypertext Markup Language) – obdoba HTML, spojena s XML.

# Literatura

[1] OHA Members [online]. 2015 [cit. 2015-04-04]. Dostupné z:

http://www.openhandsetalliance.com/oha_members.html

[2] Smartphone OS market share [online]. 2014 [cit. 2015-04-04]. Dostupné z:

http://www.idc.com/prodserv/smartphone-os-market-share.jsp

[3] Android Versions [online]. 2015 [cit. 2015-03-01]. Dostupné z:

https://developer.android.com/about/dashboards/index.html

[4] Material Design [online]. 2015 [cit. 2015-04-04]. Dostupné z:

http://www.google.com/design/spec/material-design/introduction.html

[5] Android Interfaces [online]. 2015 [cit. 2015-04-04]. Dostupné z:

https://source.android.com/devices/

[6] Android Audio HAL [online]. 2015 [cit. 2015-04-05]. Dostupné z:

https://source.android.com/devices/audio/index.html

[7] Research on Architecture of Multimedia and Its Design Based on Android. 2010 [cit. 2015-03-10].

E-ISBN: 978-1-4244-5143-2. ISBN:978-1-4244-5142-5. Dostupné z:

http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5566650

[8] MediaRecorder [online]. 2015 [cit. 2015-03-10]. Dostupné z:

http://developer.android.com/reference/android/media/MediaRecorder.html

[9] MediaPlayer [online] 2015 [cit. 2015-03-10]. Dostupné z:

http://developer.android.com/reference/android/media/MediaPlayer.html

[10] Design and Implementation of Media Player Based on Android. 2010 [cit. 2015-03-10].

E-ISBN: 978-1-4244-3709-2. ISBN: 978-1-4244-3708-5. Dostupné z:

http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5600199

[11] Understanding SAX [online] 2003 [cit. 2015-03-14]. Dostupné z:

http://www.ibm.com/developerworks/xml/tutorials/x-usax/x-usax.html

[12] Android XML parser performance [online] 2010 [cit. 2015-03-14]. Dostupné z:

http://www.developer.com/ws/android/development-tools/Android-XML-Parser-Performance-3824221-2.htm

[13] Activities [online]. 2015 [cit. 2015-04-17]. Dostupné z:

http://developer.android.com/guide/components/activities.html

[14] Ryan Cohen, Tao Wang. GUI Design for Android Apps, Part 3: Designing CompleApplications.

Apress 2014 [cit. 2015-04-18]. ISBN: 978-1-4842-0383-5. Dostupné z:
http://link.springer.com/chapter/10.1007/978-1-4842-0382-8_3

[15] Fragments [online]. 2015 [cit. 2015-04-17]. Dostupné z:

http://developer.android.com/guide/components/fragments.html

[16] Free vector icons [online]. 2015 [cit. 2014-11-23]. Dostupné z:

http://www.flaticon.com/

# Seznam obrázků

Obr. 4-11 – Navigační panel. 2014 [cit. 2015-01-15]. Dostupné z:

http://developer.android.com/design/patterns/navigation-drawer.html

Obr. 5-1, 5-2 – Aplikace Genymotion. Dostupné z:

https://www.genymotion.com

Obr. 6-1 – Propojenost aktivity a fragmentů. Vlastní zdroj.

Obr. 6-2 – Navigační panel MaterialDrawer. Dostupné z:

https://github.com/mikepenz/MaterialDrawer

Obr. 6-3 – RecorderFragment. Vlastní zdroj.

Obr. 6-4 – Oznámení o nové záložce. Vlastní zdroj.

Obr. 6-5 – PlayerFragment. Vlastní zdroj.

Obr. 6-6 – RecordListFragment. Vlastní zdroj.

Obr. 6-7 – SettingsFragment. Vlastní zdroj.

Obr. 7-1 – Seznam zařízení. Vlastní zdroj.

# Přílohy

## Seznam příloh:

- CD
- Instalační příručka
- Uživatelská příručka

## CD

K práci je přiloženo CD s následující strukturou:

- /dokument/ - tento document
- /javadoc/ - vygenerovaná dokumentace projektu
- /aplikace/ - Android Studio projekt obsahující zdrojové kódy aplikace

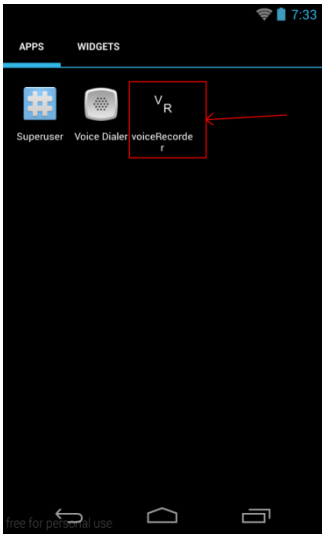
## Instalační příručka

Proto aby uživatel mohl program nainstalovat je nutné povolit instalování z neznámých zdrojů. Povolení lze nastavit v Nastavení – Zabezpečení – povolit položku „Neznámé zdroje“. Jde o bezpečnostní omezení pro aplikace mimo Google Play market. Instalace je zahájena kliknutím na stažený apk soubor aplikace. Po dokončení instalace je program přidán na panel aplikací.

# Uživatelská příručka

## 1.1 Spuštění aplikace

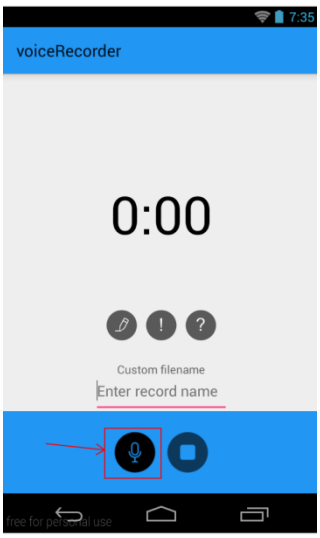
Aplikace bude spuštěna po kliknutí příslušné ikony v aplikačním panelu Android (Obr.A1).




**Obrázek A1 Aplikační panel**

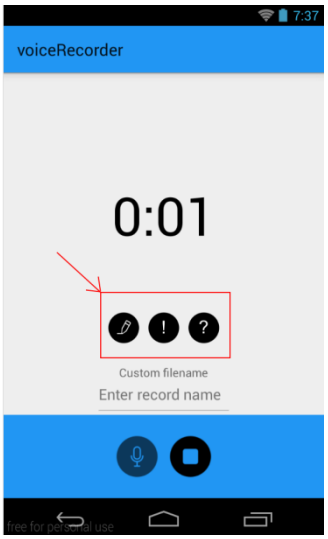
## 2.2 Ovládání aplikace

Při spuštění bude zobrazen fragment určeny k nahrávání audio. Proces nahrávání je zahájen kliknutím ikony s mikrofonem (Obr. A2).

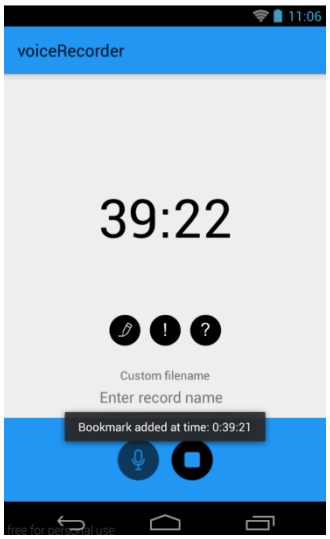



**Obrázek A2 Zahájení nahrávání**

Záložky jsou přidávaný kliknutím jedné z příslušných ikon (Obr. A3). Každá ikona slouží pro vytváření speciálního typu záložky. Uživatel je informován o vytvoření nové záložky prostřednictvím zprávy na obrazovce (Obr. A4). Nahrávání je ukončeno kliknutím ikony stop nebo po dosažení maximální doby nahrávání – 2 hodiny.

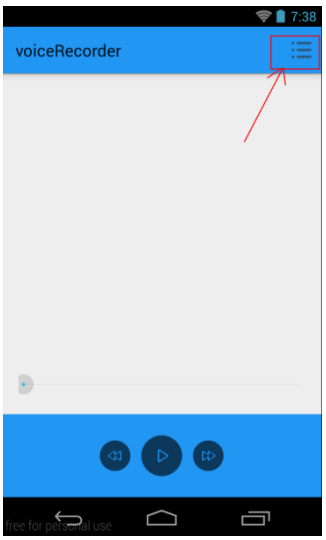



**Obrázek A3 Tlačítka pro záložky**

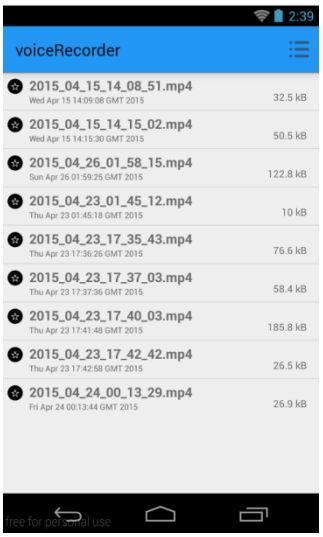



**Obrázek A4 Oznámení o nové záložce**

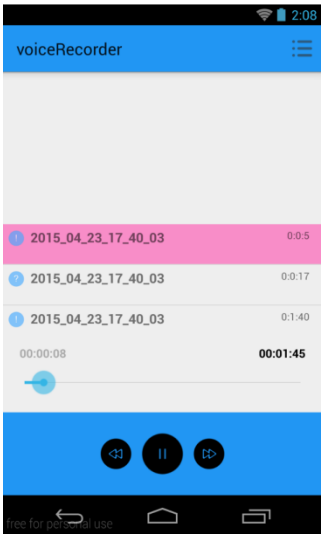
Pro přehrávání je určen fragment PlayerFragment, který lze spustit z navigačního panelu. Před přehráváním je nutno nahrávku vybrat ze seznamu, který je zavolán kliknutím ikony v pravém horním rohu (Obr. A5). Kliknutím na položku seznamu (Obr.A6) je zpátky zobrazen fragment přehrávače. Přehrávání je možné až po načtení souboru aplikací. Případné záložky budou zobrazený nad tlačítky (Obr. A7).

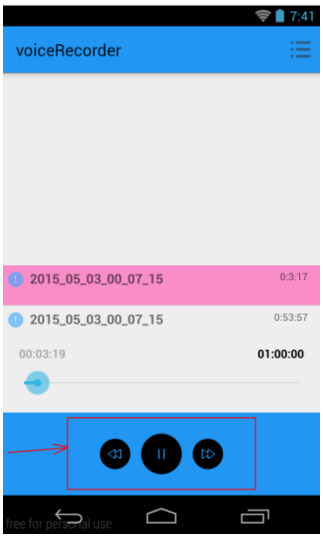



**Obrázek A5 Volání seznamu**




**Obrázek A6 Seznam nahrávek**



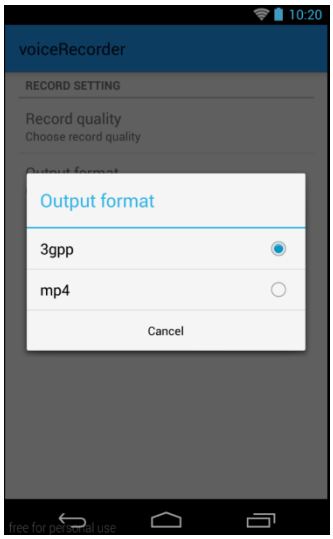




**Obrázek A7 Příslušné záložky**

**Obrázek A8 Ovládání přehrávače**

Přehrávání je spuštěno kliknutím ikony s trojúhelníkem. Kliknutím na položce seznamu je přehrávání přemístěno na čas, kdy byla záložka vytvořena. Záložka s odpovídajícím časem bude za přehrávání zvýrazněna. Přehrávání lze ovládat pomoci tlačítky pod seznamem záložek (Obr. A8).

Nastavení aplikace lze změnit v příslušném fragmentu (Obr. A9 ). Je možný změnit formát a kvalitu nahrávky.




**Obrázek A9 Nastaveni formátu**