VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Mobilní aplikace pro statistické vyhodnocení tréningových střeleckých výsledků

Mobile Application for Statistical Evaluation of Practice Shooting Results

2013 Michal Tichý

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Michal Tichý**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Mobilní aplikace pro statistické vyhodnocení tréningových střeleckých výsledků
Mobile Application for Statistical Evaluation of Practice Shooting Results

Zásady pro vypracování:

Výsledkem práce bude aplikace pro mobilní zařízení postavené na platformě Android, která musí pomocí mikrofonu zaznamenávat jednotlivé výstřely a měřit příslušné časy. Po ukončení zaznamenávání musí umožnit revizi měřených časů a dále je statisticky vyhodnotit. Výsledek bude prezentován jak v grafické tak tabulkové podobě s možností exportu a případně prezentace na Internetu - sociálních sítích.

Součástí práce musí být především:
1. Analýza možností uchovávání dat na mobilním zařízení (telefon, tablet, ...).
2. Návrh a implementace aplikace s vícejazyčnou podporou.
3. Testování.
4. Uživatelská a programátorská příručka.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Marian Mindek, Ph.D.**

Datum zadání: 16.11.2012

Datum odevzdání: 07.05.2013

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 3. května 2013 .....

Rád bych na tomto místě poděkoval panu Mgr. Ing. Michalu Krumniklovi, panu Ing. Marianu Mindekovi, Ph.D. a jeho střeleckému týmu, kteří mi s prací pomohli.

Abstrakt

Cílem této bakalářské práce je analýza možností uložení dat na mobilních zařízeních, konkrétně na platformě Android. Tyto získané informace následně využít pro návrh a implementaci aplikace pro tuto mobilní platformu. Aplikace bude pomocí mikrofonu zaznamenávat zvuk a následně z této nahrávky získá časy výstřelů. Tyto údaje bude aplikace ukládat s možností exportu ve formátu XML a sdílení pomocí sociálních sítí.

Klíčová slova: mobilní zařízení, Android, Java, zvuk, výstřel, uložení dat, SQLite, XML

Abstract

Purpose of this thesis is analyze all options of data storing on mobile devices, especially on Android platform. These obtained information then use for design and implementation an application on this mobile platform. Application will record sound, using the microphone and then get gunshots times from this record. Application will store these informations with oportunity to export in XML format and share on social sites.

Keywords: mobile devices, Android, Java, sound, gunshot, storing data, SQLite, XML

Seznam použitých zkratek a symbolů

XML	–	Extensible Markup Language
SD	–	Secure Digital
SQL	–	Structured Query Language
ARM	–	Advanced RISC Machine
API	–	Application Programming Interface
SSL	–	Secure Sockets Layer
HTTP	–	Hypertext Transfer Protocol
WAV	–	Waveform Audio File Format
RIFF	–	Resource Interchange File Format
PC	–	Personal Computer
UNIX	–	UNiplexed Information and Computing System
ISO	–	International Organization for Standardization
GPS	–	Global Positioning System
SDK	–	Software Development Kit
PCM	–	Pulse-Code Modulation

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Mobilní zařízení zažívají obrovský rozmach. Celosvětový počet uživatelů tzv. chytrých telefonů[1] překonal jednu miliardu. Rychlým tempem narůstá i počet uživatelů tabletů. Je tedy zřejmé, že v budoucnu budou tyto zařízení hrát čím dál větší roli a počet potenciálních uživatelů aplikací bude stále růst.

Tyto zařízení nám mají ulehčovat činnosti denního života, ale najdou uplatnění i ve velmi specifických situacích, například jako je využití ve střeleckém klubu. Mým úkolem je vytvořit mobilní aplikaci určenou pro nalezení výstřelů ve zvukové stopě. Jelikož je Android nejrozšířenější mobilní platforma, budu vyvíjet aplikaci právě pro ni. Aplikace bude kompatibilní s operačním systémem Android od verze 2.2 Froyo.

Nejprve provedu analýzu možností uchovávání dat na mobilních zařízeních s operačním systémem Android. Dále popisuji průběh návrhu aplikace a důvody, proč jsem se rozhodl pro různá řešení. Největší část této práce je věnována implementaci samotné aplikace, pořízení a následné zpracování zvuku a také charakteristice zvukového projevu výstřelu. Na závěr jsou uvedeny výsledky testů aplikace v reálném prostředí a jejich dopad na aplikaci.

[1] Chytrým telefonem je myšlen mobilní telefon, který využívá operační systém umožňující instalaci nebo úpravu programů.

2 Analýza možností uchování dat

V operačním systému Android lze data ukládat pěti základními způsoby. Jsou to Shared Preferences, vnitřní úložiště zařízení, externí úložiště, databáze a umístění dat v síti. Detailně budu rozebírat jednotlivé možnosti v samostatných částech.

2.1 Shared Preferences

Pro toto úložiště poskytuje operační systém framework, který usnadňuje práci s daty. Framework implementuje rozranní android.content.SharedPreferences. Slouží k uchování dat jako primitivních typů boolean, float, int, long, a string. Data jsou uložena ve formě klíč-hodnota v tzv. konfiguračním souboru, který je ve formátu XML. Data přetrvávájí i po ukončení aplikace. Po odinstalování aplikace jsou data smazána. K vytvořeným souborům může přistupovat pouze daná aplikace. Nelze přistupovat k cizím konfiguračním souborům.

Tento způsob je předurčen pro uchování konfigurace uživatele. Můžeme vytvořit více takovýchto konfiguračních souborů, které jsou společné pro celou aplikaci a rozlišují se jménem, které souboru přiřadíme při jeho vytváření. Soubory jsou umístěny ve vnitřní paměti zařízení, v adresáři s nainstalovanou aplikací. Cesta k souborům je obecně /data-/data/<package_name>/shared_prefs/<file_name>.xml.

Pokud potřebujeme konfigurační soubor přístupný pouze pro danou aktivitu, je možné vytvořit jeden bezejmenný soubor, do kterého nemají ostatní aktivity aplikace přístup.

Tento typ uchování dat jsem v aplikaci použil pro uložení uživatelského nastavení společného pro celou aplikaci.

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <int  name="gunshot_sensitivity" value="5" />
    <boolean name="delete_wav" value="false" />
    <boolean name="show_image" value="true" />
</map>
```

Výpis 1: Ukázka konfiguračního souboru aplikace

2.2 Vnitřní úložiště

Vnitřní úložiště umožňuje ukládat soubory přímo do vnitřní paměti mobilního zařízení. Soubory mohou být v textové či binární podobě, jakéhokoliv formátu. Vzhledem k tomu, že velikost vnitřní paměti mobilních zařízení bývá velmi omezená, je dobré tuto metodu využívat jen na malé soubory. Osobně toto úložiště doporučuji používat pouze pro dočasné soubory, které jsou po jejich použití smazány. Například pokud aplikace získává data po částech z internetu, je dobré tyto části ukládat zde a poté je smazat. Fyzicky se soubory ukládají uvnitř adresáře /data/data/<package_name>/files.

Je možné vytvářet další podadresáře a vytvořit si vlastní adresářovou strukturu. Není však možné zapisovat mimo složku files.

Data jsou ve výchozím stavu nepřístupná mimo aplikaci. Toto je možné změnit a umožnit buď práva ke čtení, nebo k zápisu ostatním aplikacím u jednotlivých souborů. Se soubory lze pracovat pomocí Streamů a číst je binárně. Pro optimalizaci výkonu se doporučuje používat pro čtení BufferedInputStream a pro zápis BufferedOutputStream.

Při instalaci a aktualizaci aplikací, se všechna data ukládají do tohoto úložiště. Má aplikace po instalaci zabírá 784 kB ve vnitřním úložišti.

Všechna data umístěná ve vnitřním úložišti zařízení, jsou po odinstalaci aplikace odstraněna. Toto je nutné brát v úvahu a pro uložení trvalých souborů použít úložiště jiné.

2.2.1 Dočasné soubory ve vnitřním úložišti

Pro dočasné soubory můžeme použít interní úložiště. Z Aktivity získáme absolutní cestu k dočasnému úložišti pomocí metody getCacheDir(). Umístění souborů je uvnitř adresáře s nainstalovanou aplikací. V mé aplikaci je to /data/data/tic0012.loselessoundrecord/cache/. Zde uložené soubory teoreticky nemusíme mazat, protože operační systém se o jejich správu stará sám. Ovšem začne tak činit až v případě, že dochází interní paměť.

Dokumentace [2] uvádí, že pokud klesne velikost volné vnitřní paměťi pod 1 MB, pak tyto dočasné soubory budou smazány. Nedoporučuje se však na toto „automatické čištění“ spoléhat. Tyto dočasná data by měla aplikace smazat v okamžiku, kdy je nepotřebuje. Z těchto důvodů se tato metoda nehodí k ukládání důležitých, či trvalých dat. Kdykoliv mohou být smazána. Ve své aplikaci dočasné soubory nepoužívám.

2.3 Externí úložiště

Každé zařízení kompatibilní s operačním systémem Android podporuje sdílené externí úložiště. Nejčastěji je to uživatelsky vyjímatelná SD karta. Někteří výrobci mobilních zařízení však implementují tuto paměť jako uživatelsky nevyjímatelnou. Toto úložiště je přístupné pro všechny aplikace, které mají oprávnění do něj zasahovat a uživatel je při instalaci schválí.

Data v úložišti mohou být modifikována i jinými způsoby než aplikací přímo ze zařízení. Například po připojení zařízení k počítači, případně jinému zařízení, je tato paměť dostupná.

V aplikaci není možné spoléhat na dostupnost této paměti. Úložiště mohlo být fyzicky vyjmuto, případně mohlo dojít k poruše. Vždy je potřeba ověřit, zda je externí úložiště připraveno ke čtení či zápisu. Ukázku kontroly dostupnosti úložiště z mé aplikace můžete vidět ve výpisu kódu 2.

```
String state = Environment.getExternalStorageState(); // get SD card actual state

if (!Environment.MEDIA_MOUNTED.equals(state)) { // SD card is not ready
    throw new SDCardException("SD card is not mounted");
}
```

Výpis 2: Kontrola dostupnosti externího úložiště

U této metody uložení dat může aplikace měnit libovolně adresářovou strukturu. Všechny zde uložené soubory jsou veřejně přístupné. Úložiště je určeno pro ukládání velkých souborů, určených k delšímu uchování. Obecně se doporučuje, aby se data ukládala v tomto úložišti do složky /Android/data/<package_name>/files/. Je to z důvodu, že po odinstalaci aplikace budou tato data smazána. Pokud chceme data zachovat i po odinstalování aplikace je možné je umístit kdekoliv jinde. Pro jednodušší získání cesty do adresáře určeného pro aplikaci slouží metoda getExternalStorageDirectory().

V případě že chceme uložit data veřejného typu (např. fotky, video záznamy, atd.), je možné tyto soubory umístit do systémových adresářů k tomu určených. Soubory v těchto složkách indexuje tzv. Media Scanner a tím se automaticky řadí např. do galerie obrázků, či seznamu hudebních skladeb v přehrávači médií a podobně. Konkrétně jsou to adresáře

- Music: pro hudební soubory
- Podcasts: pro zvukové soubory tzv. Podcasty.
- Ringtones: pro zvukové soubory zobrazeny při nastavení vyzvánění mobilního telefonu.
- Alarms: pro zvukové soubory zařazeny mezi upozornění, jako je např. budík apod.
- Notifications: pro zvukové soubory zařazeny mezi notifikační zvuky, jako je příchozí sms zpráva, email, apod.
- Pictures: pro obrázky zařazeny do galerie, automaticky jsou zde umístěny např. screenshoty obrazovky
- Movies: pro videozáznamy
- Download: pro obsah stažený z internetu

Cestu k těmto adresářům lze získat pomocí metody Environment.getExternalStoragePublicDirectory(String type), kde jako parametr uvedeme typ požadované složky. V případě, že adresář pro požadovaný typ souboru ještě neexistuje, je automaticky vytvořen a nemusíme jeho existenci kontrolovat. Pokud chceme umístit soubory do těchto veřejných složek s médii, ale nechceme aby je Media Scanner zaindexoval a zveřejnil, je možné vytvořit adresář s vlastním obsahem a do něj umístit prázdný soubor s názvem `.nomedia`. Přítomnost tohoto souboru v jakémkoliv adresáři způsobí, že Media Scanner bude ignorovat její obsah.

Od verze Androidu 4.0 není podporováno připojení zařízení k počítači v režimu Mass Storage. Absence tohoto režimu znamená, že se zařízení po připojení k počítači tváří jako multimediální zařízení. Z toho mimo jiné vyplývá, že soubory které nebyly zaindexovány Media Scannerem, se ve výchozím stavu nezobrazí.

Na základě těchto informací je evidentní, že pro mou aplikaci bude nejlepší ukládat pořízené zvukové nahrávky do veřejné složky Music v externím úložišti. Ve výchozím

stavu se tak skutečně děje a všechny zvukové nahrávky z aplikace se ukládají do hudební sbírky a zobrazují se v přehrávači médií. Každá nahrávka je po uložení okamžitě zaindexována ručním vyvoláním požadavku aplikace o indexaci nového souboru. Tuto možnost lze však v nastavení aplikace vypnout a nahrávky se budou ukládat do adresáře aplikace, který je po odinstalování smazán.

Pro zjištění přibližných nároků mé aplikace na externí paměť, jsem vytvořil nahrávku běžného ruchu o délce deseti vteřin. Tato nahrávka byla velká 843.34 kB. Jelikož předpokládám, že jednotlivé záznamy budou dlouhé okolo deseti vteřin, je tato velikost při požadované kvalitě záznamu přijatelná.

2.3.1 Dočasné soubory v externím úložišti

Dočasné soubory v externím úložišti jsou umístěny v adresáři Android/data/<package_name>/cache/. Pro soubory umístěné v tomto adresáři platí stejné podmínky, jako pro dočasné soubory ve vnitřním úložišti, které jsem popsal v kapitole 2.2.1.

2.4 Databáze

Operační systém Android má plnou podporu pro SQLite databáze ve verzi 3. Všechny databáze jsou přístupné pouze pro aplikaci, která je vytvořila. Databáze samotné jsou uloženy ve vnitřním úložišti ve složce s aplikací. Fyzicky lze databáze přesunout na externí úložiště. Lze tak pracovat s teoreticky neomezenou velikostí databáze.

Umístění databáze na externím úložišti však sebou nese rizika. Nachází se ve veřejném prostoru a nemůžeme zajistit, že databáze nebude změněna, či smazána cizí aplikací, případně přímo uživatelem.

Nad databází lze provádět veškeré dotazy, které podporuje samotné SQLite. Pro složitější dotazy je možné použít třídu SQLiteQueryBuilder. Databáze je možné ladit a testovat pomocí Android Debug Bridge Remote Shell.

```
adb -s emulator-5554 shell
sqlite3 /data/data/tic0012.loselessoundrecord/databases/gunshots_db
```

Výpis 3: Připojení k databázi z konzole

Databáze se hodí pro ukládání dat ve formě primitivních typů. Ukládání binárních dat způsobí obrovský nárůst velikost databáze. Pro rámcovou analýzu zaměřenou na velikost databáze jsem provedl obecný zátěžový test databáze. Test jsem prováděl v emulovaném prostředí operačního systému Android ve verzi 4.1.2 - API verze 16 s emulovaným procesorem ARM Cortex-A8, s operační pamětí 1024MB.

Databázi jsem naplnil takovým množstvím dat, které s největší pravděpodobností při používání aplikace uživatel nikdy nedosáhne. Přesto velikost databáze není tak velká, aby bylo nutné ji umístit do externí paměti zařízení. Výsledky testu jsou umístěny v tabulce 1.

Entity	Počet
kategorie	10
záznamu v každé kategorii	100
výstřelů v každém záznamu	60
celkem záznamů	60000
Velikost databáze	692 kB

Tabulka 1: Test velikosti databáze

2.5 Data na síti

Z hlediska úspory datového prostoru je nejvhodnější řešení uchovávání dat mimo samotné zařízení. Nevýhoda tohoto řešení je nutnost připojení k internetu pro získání dat. Další omezení je rychlost připojení, které komplikuje získávání objemnějších dat.

Jako největší nevýhodu lze považovat bezpečnost přenosu. Data se přenáší bezdrátovými technologiemi. Tím vzniká možnost odposlechu dat, jejich změny, případně podvržení. Přenos je možné realizovat šifrovaně pomocí SSL, to ale přináší mnohem větší výpočetní nároky na obě strany přenosu. V dnešní době se varianta šifrovaného přenosu však stává standardem.

Při přenosu je potřeba počítat s výpadky, či úplným ukončením spojení. Výhoda je, že data nejsou závislá na zařízení a v případě poruchy zařízení, ztrátě atd. data zůstávají uchována na serveru a neubírají paměťovou kapacitu zařízení.

Se sítí je možné pracovat pomocí balíčku java.net.$*$, který poskytuje základní operace se streamy, sockety, práci s internetovými adresami a zpracování HTTP požadavků. Balíček android.net.$*$ je nadstavba nad základním balíčkem pro zjednodušení práce se sítí.

3 Návrh aplikace

3.1 Formát zvukového záznamu

Důležitá část aplikace je pořízení co nejkvalitnějšího zvukového záznamu. Operační systém Android nabízí připravené API pro záznam komprimovaného zvuku. Kvalita takto pořízených nahrávek však pro účely aplikace není dostatečná. Dochází ke značnému zkreslení a nelze dostatečně přesně určit čas výstřelu. Bylo tedy nutné implementovat vlastní řešení pro nahrávání bezeztrátového zvuku.

Pomocí systémové třídy android.media.AudioRecord je možné číst data po blocích přímo z mikorofonu zařízení. S použitím této systémové třídy jsem implementoval vlastní řešení, které zajišťuje pořízení nahrávky a její následné uložení do patřičného formátu.

Předpokládá se, že pořízené nahrávky budou analyzovány i později mimo zařízení, na kterém byly pořízeny. Proto bylo nutné zvolit kompatibilní formát pro uložení nahrávek. Jako vhodný jsem shledal formát WAV, který je odnoží formátu RIFF. Formát vyvinuly firmy Microsoft a IBM pro ukládání zvuku na PC. WAV jsem si zvolil pro jeho jednoduchost implementace. Nejsložitější část implementace je správně vytvořit hlavičku souboru, která je umístěna na začátku souboru a nese všechny údaje o zvukové stopě. Za tuto hlavičku se umístí surová data a celý soubor se uloží s příponou `wav`, případně méně častou `wave`. Detailní popis tohoto formátu je na adrese:
https://ccrma.stanford.edu/courses/422/projects/WaveFormat/.

3.2 Entity v aplikaci

Aplikace umožňuje uchování nalezených výstřelů. Aby správa výstřelů byla co nejjednodušší, jsou výstřely seskupeny vždy k patřičné nahrávce, v níž byly nalezeny. Pro uživatelskou přehlednost jsou tyto nahrávky s výstřely rozděleny do kategorií s libovolným jménem. V aplikaci budu mít tedy tři entity: **výstřel**, **nahrávku** a **kategorii**. Vztahy mezi těmito entitami jsou vyjádřeny kardinalitou vztahu 1:N. Tedy jedna kategorie může obsahovat 0 až N nahrávek a jedna nahrávka může obsahovat 0 až N výstřelů. Tyto vazby budu v aplikaci udržovat pomocí objektů, kdy např. instance třídy reprezentující kategorii obsahuje pole instancí třídy reprezentující nahrávky a ty zase obsahují pole instancí tříd, které reprezentují jednotlivé výstřely.

Pro uložení dat aplikace je nejvhodnější použít relační databázi. Databáze poskytuje rychlou práci s daty. Je možné data velmi rychle přidat, vyhledat, třídit, editovat, či odstranit. Tedy vše co v aplikaci budu potřebovat. Zvolil jsem databázi SQLite, kterou operační systém Android nabízí. Tato databáze má velmi zjednodušené datové typy, které jsou však pro účely aplikace dostačující. Jejich přehled, je v tabulce 2.

SQLite databáze bohužel neposkytuje pro datum přímo datový typ. V tabulce record tedy datum ukládám v UNIX formátu, což je počet vteřin od 1. ledna 1970 00:00:00. U každé entity jsem identifikační sloupec pojmenoval „_id“. Jedná se o konvenci v pojmenování. Přítomnost identifikačního sloupce s tímto jménem je vyžadována například pro správnou funkčnost třídy CursorAdapter a je dobrým zvykem tuto konvenci dodržovat.

Kompletní návrh databáze pro aplikaci je na obrázku 1.

Typ	Popis
NULL	Prázdné hodnoty
INTEGER	celé kladné, nebo záporné čísla v rozsahu 1, 2, 3, 4, 6, nebo 8 bytů v závislosti na velikosti vložených hodnot
REAL	čísla s plovoucí desetinnou čárkou
TEXT	řetězce
BLOB	binární data

Tabulka 2: SQLite 3 datové typy

Jméno	Dat. Typ	Klíč	NULL	Popis
_id	INTEGER	PK	NE	Jednoznačný identifikátor výstřelu
time	REAL	NE	NE	Čas výstřelu v sekundách
record_id	INTEGER	CK	NE	ID nahrávky, cizí klíč z tabulky record

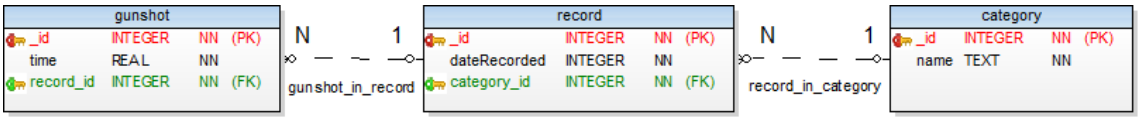
Tabulka 3: Datový slovník pro výstřel

Jméno	Dat. Typ	Klíč	NULL	Popis
_id	INTEGER	PK	NE	Jednoznačný identifikátor nahrávky
dateRecorded	INTEGER	NE	NE	Datum a čas pořízení nahrávky v UNIX formátu
category_id	INTEGER	CK	NE	ID kategorie, cizí klíč z tabulky category

Tabulka 4: Datový slovník pro nahrávku

Jméno	Dat. Typ	Klíč	NULL	Popis
_id	INTEGER	PK	NE	Jednoznačný identifikátor kategorie
name	TEXT	NE	NE	Jméno kategorie

Tabulka 5: Datový slovník pro kategorii



Obrázek 1: ER diagram

3.3 Architektura aplikace

Pro rozvržení architektury aplikace jsem použil návrhový vzor **Model–View–Controller**. Tento návrhový vzor v aplikaci rozlišuje tři základní vrstvy, které jsou od sebe logicky odděleny. Základním důvodem proč tento návrhový vzor použít, je oddělení aplikační logiky od prezentační vrstvy. Jedná se pravděpodobně o nejčastěji používaný návrhový vzor. Prakticky jakákoliv aplikace, která zpracovává požadavky od uživatele, může tento vzor použít. V aplikaci to přinese značné zpřehlednění a především odstraní skryté závisloti, což je velmi důležité, jak uvádí i Martin Fowler ve své publikaci o návrhových vzorech [4], ze které jsem čerpal (překlad z anglického originálu): „*Oddělení prezentace a modelu je jedna z nejdůležitějších zásad návrhu softwaru.* “.

Rozdělení jednotlivých vrstev v aplikaci zajistím pomocí balíčků, což je reprezentace jmenných prostorů na platformě Java.

3.3.1 Model

Zahrnuje data a aplikační logiku. Tato vrstva je srdcem aplikace. Obsahuje např. práci s databází či jiným datovým úložištěm. Provádí všechny výpočty a operace aplikace. Neformátovaná data z této vrstvy jsou použity pro prezentaci uživateli ve vrstvě *View*.

V aplikaci bude tato vrstva obstarávat veškerou práci s databází a entitami.

3.3.2 View

Tato vrstva tvoří tzv. pohledy aplikace. Jedná se o prezentační vrstvu, ve které se zobrazují patřičně naformátovaná data získaná z *Model* vrstvy. Představuje interaktivní rozhraní, se kterým pracuje uživatel a posílá případné požadavky do vrstvy *Controller*. Při změně Modelu se změní i tato vrstva, aby byla zobrazovaná data vždy konzistentní.

V aplikaci v této vrstvě použiji různé grafické rozvržení pro jednotlivé části aplikace. Budu zde prezentovat uložená data.

3.3.3 Controller

Tato vrstva je jakousi spojnicí mezi View a Modelem. Zde se zpracovávají požadavky uživatele, které přichází z View. V závislosti na požadavek se připraví data z Model vrstvy, provedou patřičné operace a případně se změní View uživatele. Komponenty této vrstvy jsou specifické pro konkrétní aplikaci, tudíž nejsou znovupoužitelné, na rozdíl od ostatních vrstev. *Controller* reaguje na změny v Modelu a View.

V aplikaci použiji jako Controller systémovou třídu Activity, která na operačním systému Android reprezentuje komponentu vrstvy Controller.

3.4 Vícejazyčná podpora

Operační systém Android má velmi dobrou podporu pro vícejazyčné aplikace. Používá k tomu identifikaci zdrojů na základě hardwarových parametrů a uživatelské konfigurace zařízení. Pokud chci například v aplikaci rozlišit zdroje pro různé jazykové mu-

tace, tak v případě řetězců vytvořím adresář `values`, který bude obsahovat XML soubor `strings.xml`. V tomto souboru budou uloženy výchozí textové řetězce.

Každý řetězec musí obsahovat unikátní atribut `name`. Tento atribut slouží pro jednoznačnou identifikaci řetězce v rámci celé aplikace. Řetězce je vhodné ukládat heslovitě, případně věty jako nedělitelné celky.

Přidání podpory další jazykové mutace se provede vytvořením adresáře `values-cs`. Adresář s přidaným ISO kódem jazyka poté slouží pro identifikaci zdrojů. Takto je vytvořena podpora češtiny. Je však nutné do adresáře umístit XML soubor, který musí obsahovat všechny řetězce z výchozího souboru.

Identifikace a použití zdrojů se provádí automaticky při spuštění aplikace. Pokud bude v operačním systému nastaven jako jazyk čeština, použijí se řetězce ze souboru umístěné v adresáři `values-cs`. V případě jakéhokoliv jiného jazyka, který nemá definován vlastní řetězce s překladem, se použijí řetězce z výchozí složky `values`. Takto lze přidávat liboblný počet jazykových mutací aplikace.

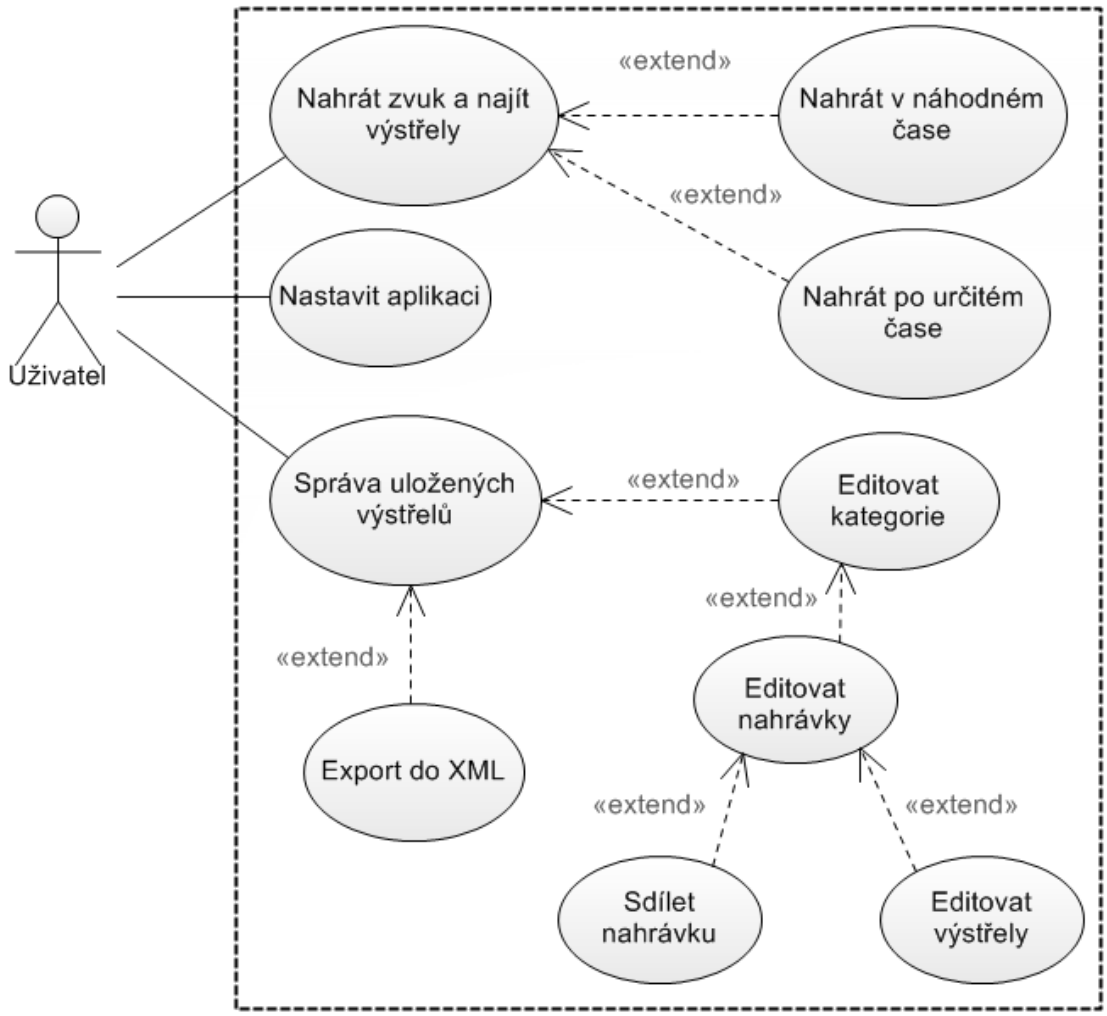
Možnosti rozlišení zrdojů nejsou omezeny pouze podle aktuálního jazyku v operačním systému. Lze například rozlišit zdroje pro zařízení s různým rozlišením displeje, nebo například pro zařízení s hardwarovou klávesnicí a bez ní. Identifikátory lze kombinovat a měnit jejich pořadí, čímž se mění jejich priorita. Operační systém začíná zdroje rozlišovat vždy zleva postupně přes jednotlivé identifikátory. Jiné rozlišení zdrojů v aplikaci není potřeba, takže je nevyužiji. Vytvořím anglickou mutaci jako výchozí a českou verzi v případě, že bude jazyk operačního systému nastaven na český.

3.5 Scénář případů užití

Aplikace musí splňovat následující požadavky:

1. zaznamenat čas výstřelů pomocí mikrofonu
2. umožnit revizi nalezených výstřelů
3. graficky znázornit výstřely
4. správa uložených výsledků
5. možnost exportu a sdílení výsledků
6. možnost změnit citlivost na výstřel

Zaznamenávání výstřelů je rozšířeno ještě o možnost spuštění v náhodném čase a o spuštění po nastaveném čase. Provázanost těchto aktivit nejlépe reprezentuje diagram užití na obrázku 2.



Obrázek 2: Use Case diagram

4 Implementace

4.1 Nutné předpoklady

Platforma Android obsahuje bezpečnostní opatření v podobě schvalování přístupu aplikace k hardwarovým či systémovým prostředkům. Tato metoda spočívá v tom, že všechny tyto operace musí být uvedeny v souboru AndroidManifest.xml, který obsahuje základní informace o aplikaci. Pokud zde není uvedena nějaká operace a pokusíme se například využít mikrofon, aplikace skončí výjimkou. Všechny tyto požadavky jsou uživateli zobrazeny při instalaci aplikace a musí je schválit. V opačném případě nebude instalace pokračovat. V tomto souboru jsou uvedeny i všechny hardwarové požadavky zařízení jako přítomnost GPS, bluetooth, akcelerometr a další. Je zde uvedena i minimální verze operačního systému. Bez splnění těchto požadavků, nepůjde aplikaci na zařízení nainstalovat. Verze Androidu se udává podle verze SDK. V mém případě je to verze SDK 8, což odpovídá verzi Androidu 2.2 Froyo.

V aplikaci budu využívat mikrofon, musím tedy uvést toto oprávnění. Požadavek na fyzickou přítomnost mikrofonu v zařízení nemusím již udávat. Ten se implicitně použije při vložení povolení o přístupu k mikrofonu. Další povolení, které pro aplikaci potřebuji, je možnost čtení a zápis na externí úložiště, protože zde bude aplikace ukládat pořízené zvukové záznamy a exportovaná data ve formátu XML. Poslední povolení, které aplikace potřebuje, je možnost zabránění přechodu zařízení do režimu spánku. Toto v aplikaci využívám při zobrazení časomíry, aby nedošlo k pohasnutí displeje, či uspání zařízení.

Soubor AndroidManifest.xml obsahuje dále jméno balíčku s aplikací, verzi kódu, nastavení grafických prvků jako logo, zobrazené jméno aplikace, použitý tématický styl a především se zde nachází kompletní seznam Aktivit. U každé aktivity lze nastavit její titulek, výchozí orientaci displeje a další. Je možné nastavit nadřazenou aktivitu, která se spustí při stisku tlačítka zpět. jednu aktivitu je nutné označit jako spouštěcí. Taková aktivita se spustí při zapnutí aplikace.

```
<?xml version='1.0' encoding='utf-8' ?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="tic0012.loselessoundrecord"
    android:versionCode="1"
    android:versionName="1.0" >

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

Výpis 4: Nutné povolení aplikace

4.2 Struktura aplikace

V této kapitole popisuji jednotlivé vrstvy aplikace. Kompletní třídní diagram s umístěním jednotlivých tříd do patřičných balíčků naleznete na obrázku 8.

4.2.1 Aktivity

Na operačním systému Android se vše odvíjí od Aktivit. Jsou to komponenty aplikace poskytující obrazovku, na které může uživatel provádět různé operace. Každá aktivita představuje okno v aplikaci, jejíž bližší specifikaci poskytuje View. Spadají do návrhové vrstvy Controller. Všechny Aktivity v aplikaci jsou potomky abstraktní třídy BaseActivity. Jména Aktivit jsou vytvořeny podle koncepce <activityName>Activity. Popis jednotlivých Aktivit se nachází níže.

4.2.1.1 BaseActivity Tuto třídu používám jako předka všech Aktivit ve své aplikaci. Jelikož se nejedná o plnohodnotnou systémovou aktivitu a nechci, aby bylo možné vytvořit její instanci, je tato třída abstraktní.

Ve všech aktivitách je několik společných aspektů, které není vhodné řešit zvlášť na několika místech. Je zde například držena instance objektu `SharedPreferences`, který slouží pro uchování uživatelské konfigurace. Dále zde řeším obsluhu vybrání položek z kontextové nabídky, které jsou společné pro celou aplikaci. Např. výběr nastavení aplikace, či zobrazení uložených výstřelů.

Je zde řešena kompletní obsluha konfiguračního dialogu. Toto řešení konfigurace je pro uživatele mnohem přehlednější a rychlejší. Po výběru nastavení z kontextové nabídky kdekoliv v aplikaci, se otevře dialogové okno. Toto okno je instance třídy AlertDialog. Obsah je tvořen vytvořeným View, které se skládá z posuvníku pro nastavení citlivosti a několika „zaškrtávacími“ políčky. Po stisku tlačítka pro potvrzení, se všechny změny v konfiguraci uloží do objektu `SharedPreferences`. Jeho detailní popis jsem uvedl v kapitole 2.1. Největší výhoda tohoto řešení je, že uživatel neztrácí přehled a změnu nastavení provádí přímo na místě, kde se právě nachází.

V aplikaci se poměrně často mění aktuální Aktivity. Ve většině případů se jedná o jednoduché vytvoření instance požadované aktivity a přesunutí ji do popředí. Případně může být potřeba aktuální aktivitu ukončit, aby byla odebrána ze zásobníku aktivit a nebylo možné se k ní vrátit pomocí tlačítka zpět. Pro tyto účely jsem si vytvořil metodu setActivity, která tuto činnost velmi zjednodušuje a je přístupná všem potomkům této třídy. Jejími parametry jsou třída, kterou chceme spustit a příznak, zda chceme nynější aktivitu ukončit. V případě, že se rozhodneme aktivitu ukončit, nebude možné se k ní vrátit pomocí tlačítka zpět. Ukázka této metody je uvedena ve výpisu kódu 5.

```
protected void setActivity(Class<?> cls, boolean terminate) {
    // start new activity
    Intent intent = new Intent(this.getApplicationContext(), cls);
    intent.setFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT);
    this.startActivity(intent);

    // end this activity
    if (terminate){
        this.finish();
    }
}
```

Výpis 5: Metoda pro změnu aktuální Aktivity

4.2.1.2 FinalRecordActivity Jedná se o spouštěcí Aktivitu, která obsluhuje pořízení zvukového záznamu, včetně rozšířených možností spuštění nahrávání. Nastavení a obsluhu rozšířených možností spuštění nahrávání jsem vyřešil pomocí dialogových oken. Detailní postup získání zvukového záznamu popisuji v kapitole 4.3.

Tato aktivita je zároveň jakýmsi vstupním bodem aplikace. Odtud se spouští aktivita pro zpracování pořízené zvukové nahrávky a lze odtud přejít do správy uložených výstřelů. Jako hlavní ovládací prvek této aktivity je tlačítko pro okamžité spuštění záznamu zvuku. Pro přehlednost jsem ostatní ovládací prvky umístil do kontextové nabídky. Toto uspořádání jsem zvolil proto, že ve většině případů se bude využívat okamžité nahrávání a umístění více ovládacích prvků přímo na obrazovku by působilo rušivě a mohlo vést k nechtěným příkazům.

4.2.1.3 ReadActivity Zde se zpracovává pořízená zvuková nahrávka a hledají se výstřely. Podrobně hledání výstřelů popisuji v kapitole 4.4. Výsledek je prezentován v grafické podobě, s vyznačenými místy, kde byl nalezen výstřel. Výstřely jsou také zobrazeny v seznamu s přesnými časy, s možností vybrat platné nálezy a poté je uložit. Případně přidat nenalezený výstřel ručně. Detailně se prezentaci výsledků věnuji v kapitole 4.5. Aktivitě je při spuštění jako parametr předána cesta k nahrávce. Pokud tento parametr není uveden, aktivita se ukončí.

Tato aktivita je vždy po ukončení odebrána ze zásobníku aktivit, aby se předešlo navrácení pomocí tlačítka zpět.

4.2.1.4 StoredActivity V této Aktivitě se nachází správa kategorií a nahrávek v nich obsažených. Je možné kategorie vytvářet, přejmenovat, či mazat. Jednotlivé nahrávky lze sdílet na sociálních sítích, či smazat. Sdílení je řešeno pomocí obecného Intentu typu SEND, který umožní uživateli akci dokončit pomocí aplikace v telefonu dle jeho výběru. Požadavku je předán textový parametr, který obsahuje časy jednotlivých výstřelů a datum jejich pořízení. Po „klepnutí“ na nahrávku se spustí nová aktivita a zobrazí se její detail. Všechny tyto úkony lze provádět pomocí dlouhého stisku jednotlivých položek.

Zde se provádí i export všech dat do XML souboru. Po dokončení exportu je XML soubor uložen na SD kartu zařízení a uživateli ja nabídnuta možnost soubor dále zpracovat. Je možné jej poslat na jiné zařízení pomocí technologie Bluetooth, zálohovat např. pomocí služby Dropbox, či poslat jako přílohu e-mailovou zprávou. Strukturu exportovaných dat můžete vidět v ukázce kódu 6.

```
<?xml version='1.0' encoding='UTF-8' standalone='yes' ?>

<categories>
    <category id="1">
        <name>Vychozi</name>
        <records>
            <record id="1" date="2013-04-05 01:13:44">
                <gunshots>
                    <gunshot id="1" time="0.0013605442" />
                    <gunshot id="2" time="0.002743764" />
                </gunshots>
            </record>
        </records>
    </category>
</categories>
```

Výpis 6: Struktura exportovaných dat v XML souboru

4.2.1.5 GunshotsActivity

Detail uloženého zvukového záznamu a výpis všech výstřelů, které se v něm nachází, je realizován zde. Další výstřely lze ručně přidat. Je možné odstranit vybrané výstřely.

4.2.2 View

View slouží pro grafickou prezentaci dat a ovládacích prvků. Je reprezentováno XML souborem, který popisuje vlastnosti a vzhled samotného View, ale také všech komponent, které se v něm nachází. V aplikaci mám ke každé Aktivitě přiřazeno unikátní View. Pro dialogová okna jsem vytvořil také zvláštní View, ve kterých definuji jejich vzhled a ovládací prvky. Příklad vytvořeného dialogového okna můžete vidět na obrázku 5.

4.2.3 Model

V této vrstvě je samotné jádro aplikace. Je zde veškerá práce s databází a také zpracování a analýza zvukového záznamu.

Pro práci s databází je určena třída `DatabaseHandler`. Třída v sobě nese informace o kompletní struktuře databázových tabulek. Obsluhuje změnu verze databáze a v případě, že databáze ještě v zařízení neexistuje, tak ji třída vytvoří. V aplikaci je to nejnižší vrstva, která pracuje s databází. Jelikož v aplikaci nepotřebuji více aktivních spojení s databází, třída implementuje návrhový vzor Singleton. Tím je zajištěna maximálně jedna instance této třídy v rámci celé aplikace. V operačním systému Android se může stát, že je

vytvořeno více instancí aktivit z jedné aplikace. Je tedy nutné zajistit, aby bylo vytváření jedné instance ošetřeno i v rámci více vláken. Implementace je v ukázce kódu 7.

Jsou zde umístěny modely, které pracují se všemi entitami v aplikaci. Konkrétně jsou to třídy GunshotModel, RecordModel a CategoryModel. Každá z těchto tříd implementuje rozhraní IModel, které zajišťuje základní práci s entitami, jako je uložení, smazání a získání entity podle jednoznačného ID.

Vytvořil jsem několik vlastních výjimek, které mi umožňují lépe identifikovat a zpracovat chyby, které mohou nastat při běhu aplikace. Například výjimka CannotDeleteException je vyhozena v případě, že se uživatel pokusí smazat výchozí kategorii. Další výjimka je SDCardException, kterou aplikace vyhodí v případě, že nastane jakýkoliv problém s externím úložištěm a není možné s ním pracovat.

```
private static DatabaseHandler self;

private DatabaseHandler(Context context) {
    super(context, DB_NAME, null, DATABASE_VERSION);
}

public static synchronized DatabaseHandler getInstance(Context context) {
    if (self == null) {
        self = new DatabaseHandler(context);
    }

    return self;
}
```

Výpis 7: Implementace návrhového vzoru Singleton

4.3 Pořízení záznamu

Pro pořízení co nejkvalitnějšího zvukového záznamu jsem implementoval vlastní třídu s názvem UncompressedAudioRecorder. Pomocí systémové třídy AudioRecord se při nahrávání kontinuálně čtou data přímo z mikrofonu zařízení. Nahrávání je blokující operace. Aby nedošlo k zablokování hlavního vlákna a aplikace se nestala neovladatelnou, je nutné samotné nahrávání realizovat v separátním vlákně.

Vzorkovací frekvenci pro záznam jsem použil 44 100 Hz. Tuto frekvenci jsem zvolil z důvodu, že se jedná o stadnard Red Book pro Audio CD. Tato konkrétní frekvence je také jediná, která má zaručenou podporu na všech zařízeních, jak uvádí oficiální dokumentace pro Android. Jednotlivé vzorky jsou získány pulzně kódovou modulací(PCM), která slouží pro převod analogového signálu na digitální.

Každý vzorek je reprezentován 16 bity dat a obsahuje i zápornou složku amplitudy. Je možné zvolit i 8 bitové rozlišení, ale toto nenese informaci o záporné složce. Zápornou složku záznamu pořizuji záměrně, protože se může použít při pozdější analýze mimo mobilní zařízení.

Záznam je pořizován jako jednokanálový, tedy v režimu MONO. Zde není nutné použít více kanálů, protože kvalita nahrávky by tím nevzrostla a pouze by narostla da-

tová velikost výsledného záznamu. Také by se poměrně zkomplikovala analýza zvukové stopy a hledání jednotlivých výstřelů.

Při nahrávání se data ukládají do dočasného souboru pomocí třídy FileOutputStream. Po zastavení nahrávání se vytvoří nový soubor, do kterého se na začátek po jednotlivých bytech vloží hlavička se všemi informacemi o zvukové nahrávce. Za hlavičku se zkopírují data z dočasného souboru, který je po tomto kroku smazán. Struktura hlavičky a informací v ní je detailně popsána na adrese
https://ccrma.stanford.edu/courses/422/projects/WaveFormat/.

Aby nedošlo k nechtěnému přepsání již předešlé nahrávky, je název souboru tvořen UNIX časem, což je aktuální počet vteřin od 1. 1. 1970. Pokud je zvoleno ukládání nahrávek do veřejné složky s hudbou, zašle se systémový požadavek o přidání nahrávky do indexu hudebních souborů.

Jako výsledný zvukový formát jsem kvůli jednoduchosti a široké podpoře zvolil formát WAV. Tím je zajištěna kompatibilita s nejrůznějšími zvukovými editory a analyzátory. Případně je možné zvukovou nahrávku přehrát jako běžnou skladbu.

Třída UncompressedAudioRecorder zajišťující nahrávání bezeztrátového zvuku, nemá žádné závislosti na vnějších komponentách. Je možné ji bez jakýchkoliv úprav použít v jiném projektu pro platformu Android. Jediné nutné opatření je nastavení povolení pro přístupu k mikrofonu zařízení.

Pro okamžité spuštění nahrávání slouží přepínací tláčítko, které se může nacházet ve dvou stavech: aktivní a neaktivní. Reaguji na událost stisku tlačítka. V případě, že je stav aktivní, spustím nahrávání. V opačném případě nahrávání zastavím a spustím zpracování nahrávky. Další možnost jak začít nahrávat zvuk, je automatické spuštění po určitém čase. Případně náhodně v určitých mezích.

Pro načasované spuštění používám třídu CountDownTimer jako předka pro své třídy, které implementují metody spouštěné po nastaveném čase.

4.4 Hledání výstřelů

Z práce pana Roberta Mahera [3] a z pořízených nahrávek zvuku jsem zjistil, že výstřel je poměrně krátký časový úsek, kdy amplituda [2] nabude téměř maximálních hodnot a poté pomalu klesá. Výstřel lze poměrně přesně určit, protože rozdíl hodnot amplitudy mezi běžným ruchem či mluvenou řečí a hodnotou výstřelu je velmi značný. Délka úseku, který se dá jednoznačně označit za výstřel je dlouhý přibližně 0.8 vteřiny. Poté se hodnoty amplitudy blíží k běžnému ruchu, či mluvené řeči. Pro nalezení výstřelu budu tedy hledat určitý časový úsek, ve kterém součet amplitud přesáhne určitou mez. Průběh amplitudy výstřelu můžete vidět na obrázku 3, kde v první části je zaznaménana mluvená řeč a běžný ruch. Přibližně v čase 0.11s dochází k výstřelu. Hledaná plocha je znázorněna červenou barvou.

Zpracování zvukové nahrávky je výpočetně poměrně náročná operace a není možné ji provádět v hlavním vlákně aplikace. Pro tento účel jsem použil systémovou třídu AsyncTask. Tato třída umožňuje provádět náročné operace na pozadí, zobrazovat průběh operace v

[2] Amplituda je maximální výchylka akustického tlaku. Udává přímo úměrně sílu zvuku.

hlavním vlákně a nakonec předá výsledek operace. Tuto třídu jsem použil jako předka pro třídu vlastní, která implementuje potřebné metody. Nejprve se pokusím soubor s nahrávkou otevřít a připravit jej ke čtení. Pokud byl mezitím soubor smazán, či poškozen a nepodaří se jej otevřít, ukončím celé zpracování a informuji uživatele o absenci nahrávky. Dále nastavím maximální hodnotu v ukazateli průběhu čtení tak, aby odpovídal počtu bytů v souboru. To mi zajistí pohodlnou změnu průběhu při samotném čtení.

Při návrhu algoritmu, pro nalezení výstřelů, jsem se inspiroval diplomovou prací pana Bc. Davida Vybírala [5], který používá analýzu amplitudy pro nalezení kritických míst a ty dále zkoumá pomocí modifikované Fourierovy transformace. V práci také uvádí, že analýza pomocí Fourierovy transformace je výpočetně velmi náročná. Proto jsem se tuto pokročilou metodu rozhodl neimplementovat. Zpracování nahrávky by bylo neúměrně dlouhé a použitelnost aplikace v praxi, by byla takřka nemožná.

Analýza zvukového souboru probíhá jeho lineárním čtením. Nejprve je nutné přeskočit hlavičku s informacemi o zvukovém formátu. Jelikož vím, že se jedná o hlavičku pro formát WAV, je nutné přeskočit 44 bytů. Což je fixně daná velikost hlavičky. Dále pokračuji čtením souboru vždy po dvou bytech, protože každý vzorek amplitudy je reprezentován šestnácti bity. Tyto dva byty získám odděleně jako dva prvky pole. Je potřeba každé tyto oddělené byty spojit pomocí binárního posuvu a získat tak jedno číslo, které reprezentuje velikost amplitudy v daném vzorku. Ukázka čtení souboru a získání amplitudy vzorku je ve výpisu kódu 8. Velikost amplitudy je v rozsahu od -32 768 po 32 767. Takto bych získal 44 100 vzorků amplitud na jednu vteřinu záznamu. To je pro nalezení výstřelu zbytečně mnoho a zpracování takovéhoto množství dat by trvalo velmi dlouho dobu. Zavedl jsem jistý druh komprese a zpracuji pouze každý čtvrtý vzorek. Tím se počet vzorků zredukuje na 11 025 za vteřinu. Což je stále dostačující pro úspěšné nalezení výstřelů, ale zpracování nahrávky se tím podstatně zkrátí.

Pokud bych výstřel hledal po úsecích, které jsou rovné jeho času, mohlo by se stát, že jej nezaznamenám. Zvolil jsem tedy úseky poloviční, tedy 0.4 vteřiny. Díky tomu výstřel zaznamenám i v případě, že se úsek nebude nacházet přesně na začátku výstřelu. V případě, že bych převedl délku zkoumaného úseku na počet vzorků, tak při 11 025 vzorků za vteřinu a délce 0.4 vteřiny, je to úsek o počtu 441 vzorků.

Při čtení záznamu ukládám kladné vzorky do pole a při jeho naplnění provedu jejich součet. Tím získám plochu kladné amplitudy v tomto časovém úseku. Záporné hodnoty amplitudy pro mne nemají žádný význam. Velikost spočítané plochy porovnávám s referenční hodnotou, kterou jsem získal při výpočtu plochy záznamu, kde se nacházel pouze výstřel. Těchto hodnot pro porovnávání plochy jsem vytvořil jedenáct a jsou odstupňovány. Jedenáct proto, že jsem v aplikaci umožnil konfiguraci citlivosti a je možné nastavit právě jedenáct stupňů. Pokud je velikost plochy větší, nebo rovna referenční, znamená to, že jsme nalezli výstřel. Je uložen čas a pozice prvního vzorku ve zkoumaném úseku.

V každém cyklu čtení je nutné kontrolovat, zda uživatel nezrušil operaci zpracování. Pokud ano, je nutné celý proces přerušit. Je také potřeba informovat uživatele o aktuálním stavu čtení souboru. Po skončení čtení, se získaná data předají aktivitě a uživatel má možnost zrevidovat výsledky, případně shlédnout graf průběhu.

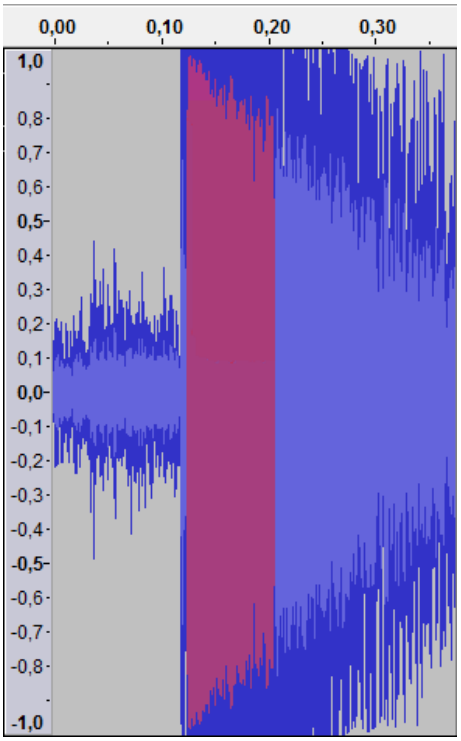
```
while (buffInStream.read(myBuff, 0, 2) > 1) {  // read WAF file by 2 bytes
    if (isCancelled()) {  // check if user canceled work
        break;
    }

    if (readedForCompression < toSkip && this.compression) { // skip byte for compression
        readedForCompression++;
        continue;
    }

    readedForCompression = 0;

    // get number representation of amplitude
    ampl = ((short) ((myBuff[0] & 0xff) | (myBuff[1] << 8)));
}
```

Výpis 8: Čtení zvukové nahrávky a získání amplitudy



Obrázek 3: Zvukový projet výstřelu, hledaná část je vyznačena červeně

4.5 Prezentace výsledku

Při čtení zvukové nahrávky se uchovávají hodnoty amplitud pro grafické znázornění zvuku. Je potřeba rozdělit kladné a záporné amplitudy. Každý vzorek je tedy reprezentován dvouprvkovým polem. Tato jednotlivé pole ukládám do seznamu, který použiji pro vykreslení průběhu. Uchovávat všechny amplitudy by bylo zbytečně paměťově náročné, protože průběh vykresluji ručně přímo do Bitmapy. Takto vytvořený obrázek není fyzicky uložen, ale je uchováván v operační paměti zařízení. Velikost Bitmapy je na operačním systému omezena a proto vykresluji pouze prvních patnáct vteřin záznamu. Ukládám tedy pouze amplitudy, které jsou potřebné pro vykreslení grafu.

Při tvorbě grafu dochází k další kompresi. Kdybych vykresloval všechny amplitudy, měl by graf zobrazující jednu vteřinu záznamu 11 025 pixelů na šířku. Zvolil jsem kompresní poměr 1:60, při kterém jsou vypovídající vlastnosti a velikost obrázku vyhovující. Kompresi provádím tak, že vždy sečtu šedesát vzorků, spočítám jejich průměrnou hodnotu a tento vzorek vykreslím. Převodem musí projít i čísla vzorků nalezených výstřelů, aby jejich vykreslení odpovídalo skutečnosti. Nalezené výstřely vykresluji do grafu pomocí červené ikonky kříže.

Samotné vykreslování grafu provádím tak, že si připravím plátno o patřičných rozměrech. Šířku obrázku získám z počtu vzorků, kdy jeden vzorek je roven jednomu pixelu. Výšku plátna určuji podle výšky obrazovky na aktuálním zařízení, přičemž musím odečíst výšku systémové lišty. Nastavím potřebnou barvu pozadí a v cyklu procházím jednotlivé vzorky a vždy vykreslím od vertikálního středu kladnou a zápornou hodnotu amplitudy. V každém cyklu vykreslování kontroluji, zda se vzorek nerovná vzorku, kdy byl nalezen výstřel. Pokud ano, vykreslím na toto místo ikonu výstřelu. Dále kontroluji, zda se cyklus nenachází přesně v půlce vteřiny. Pokud ano, vykreslím vertikální čáru a umístím vedle ní časovou značku. Nakonec vykreslím horizontální čáru, symbolizující nulovou hodnotu amplitudy.

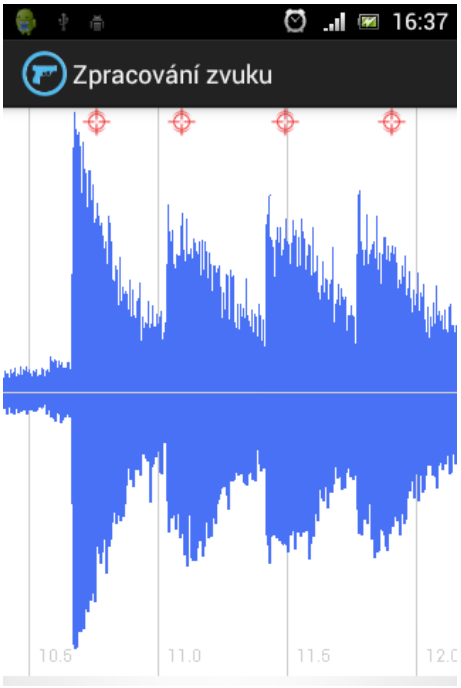
Pro umístění grafu do obrazovky jsem použil ImageView. Toto View slouží jako obálka pro obrázky. Lze do něj vložit prakticky jakýkoliv bitmapový obrázek a toto View jej vykreslí. S největší pravděpodobností bude výsledný graf na šířku větší, než je zobrazovací plocha zařízení. Umístil jsem tedy celé ImageView, které obrázek vykresluje, ještě do nadřazeného HorizontalScrollView, které umožňuje posouvat obsahem v horizontálním směru. Ukázku vytvořeného grafu včetně vyznačených výstřelů můžete vidět na obrázku 4.

Kompletní výpis všech nalezených výstřelů je k dispozici po skrytí grafu. Jednotlivé výstřely jsou zobrazeny v přehledném seznamu, kdy každý výstřel je reprezentován časem jeho nálezu. Uživatel má možnost odebrat neplatné výstřely, případně ručně přidat výstřel další. Pro vykreslení seznamu jsem použil systémový prvek ListView.

Zobrazení seznamu s výstřely, až po skrytí grafu jsem zvolil proto, že seznam reprezentovaný ListView, umožňuje vertikální posun obsahu, pokud se již nevejde do zobrazovací plochy. Tudíž není možné, aby byl zobrazen graf a pod ním se nacházel seznam s výstřely, protože by byly v sobě vnořeny dva posuvníky.

ListView lze nadefinovat parametr možnosti výběru na hodnotu *„multipleChoice“*. Toto nastavení umožní uživateli vybrat jednotlivé položky. Jako datový vstup pro tento

seznam je pole řetězců, které naplním časy nalezených výstřelů. Po naplnění seznamu daty, všechny tyto prvky označím, aby uživatel nemusel označovat spoustu prvků, ale mohl pohodlně vybrat pouze neplatné výstřely. Po revizi výstřelů se označené výstřely uloží do vybrané kategorie. Po uložení může uživatel ihned začít s dalším nahráváním.



Obrázek 4: Graf průběhu zvuku s nalezenými výstřely

4.6 Správa uložených výstřelů

4.6.1 Kategorie a nahrávky

Jak jsem uvedl v kapitole 3.2, výstřely jsou seskupeny podle nahrávek a ty jsou uloženy v kategoriích. Po vybrání možnosti „Uložené výstřely“ z kontextové nabídky v aplikaci, je spuštěna aktivita StoredActivity. Tato aktivita se stará o editaci kategorií a záznamů, které uživatel vytvořil.

Záznamy jsou zobrazeny v dvoúrovňovém seznamu, kdy v první úrovni jsou kategorie a ve druhé úrovni jsou umístěny záznamy. Použil jsem proto třídu ExpandableListView. Tato třída je potomkem třídy ListView. V tomto případě již nelze implementovat jednoduchý datový vstup jako v případě jednoúrovňového seznamu. Pro tento účel jsem vytvořil třídu CategoryListAdapter, která je potomkem BaseExpandableListAdapter. V této třídě implementuji metody pro práci s prvky seznamu, jako získání prvků podle jejich indexu, získání celé skupiny záznamů a podobně. Dále ve třídě dynamicky vytvářím vzhled jednotlivých položek. Samozřejmě jsem mohl vytvořit standardní XML soubor

pro definici vzhledu prvků v seznamu, ale pro osvojení co nejvíce možností, jsem zvolil tuto metodu.

View tvořící seznam umožňuje zpracovat události dotyku na jednotlivých prvcích. Krátký dotyk na kategorii nepotřebuji zpracovávat. Rozbalování prvků v druhé úrovni je implementováno v samotném view. Dotyk provedený na prvcích z druhé úrovně seznamu, tedy na zvukovém záznamu, již zpracovávám. Po stisku záznamu provedu spuštění nové aktivity, které jako parametr předám ID zvukové nahrávky. Nynější aktivitu neukončím, takže zůstane v zásobníku a při stisku tlačítka zpět, se uživatel vrátí na seznam kategorií.

Export všech uložených dat do XML souboru provádím pomocí třídy XmlSerializer a FileOutputStream. Výsledný XML soubor se ukládá na externí úložiště do složky GunshotRecorder. Strukturu souboru můžete vidět ve výpisu kódu 6. Po uložení souboru vytvořím systémový požadavek o zaslání dat. Jako parametr přiložím vytvořený soubor a uvedu jeho typ. Díky tomuto požadavku bude mít uživatel možnost soubor ihned například poslat jako přílohu e-mailu, umístit do DropBoxu, poslat na jiné zařízení pomocí Bluetooth a podobně.

Na každém prvku seznamu zpracovávám události dlouhého dotyku. Dlouhý dotyk je poměrně rozšířené gesto, které slouží pro editaci, či smazání prvku. Při dlouhém dotyku na kategorii zobrazím nabídku, která umožní kategorii editovat a změnit její název, nebo celou kategorii i s jejím obsahem smazat.

Pro změnu názvu kategorie používám opět dialogové okno, ve kterém je umístěn prvek pro editaci textu. Při potvrzení editace kontroluji, zda není název prázdný. Při výběru volby pro smazání kategorie, kontroluji zda není vybrána výchozí kategorie. Jelikož musí být všechny nahrávky umístěny v nějaké kategorii, je nutné zajistit, aby existovala alespoň jedna. V případě, že se uživatel pokusí smazat výchozí kategorii, je vyhozena výjimka CannotDeleteException. Pokud je tato výjimka zachycena, nedojde ke smazání a uživatel je informován, že tuto kategorii nelze smazat. Při dlouhém stisku na zvukovém záznamu, je k dispozici možnost jej sdílet či smazat. Po odstranění výstřelu, nebo celé kategorie je vždy aktualizováno celé view se seznamem.

Sdílení je vytvořeno pomocí systémového požadavku pro poslání textu a jako parametr vložím naformátovaný text s časy jednotlivých výstřelů v záznamu. Toto řešení dává uživateli na výběr, jakým způsobem chce informace sdílet. Dostupné možnosti závisí na aplikacích, které má na svém zařízení nainstalovány. Na výběr jsou například sociální sítě Facebook, Google+, Twitter, ale i jiné služby jako e-mail, či textová zpráva SMS.

4.6.2 Jednotlivé výstřely

Po výběr nahrávky ze seznamu je spuštěna aktivita GunshotsActivity, která se stará o správu jednotlivých nahrávek. Detail nahrávky je tvořen datem, kdy byla pořízena a pod ním je seznam nalezených výstřelů. Pro vykreslení seznamu jsem použil ListView, kterému jako datový vstup slouží pole s časy jednotlivých výstřelů. Jednotlivé výstřely je možné označit a následné tyto vybrané výstřely odstranit. Označení se provede krátkým

stiskem. Odstranění vybraných výstřelů se provede výběrem této možnosti z kontextové nabídky. Po odstranění provedu aktualizaci datového vstupu pro seznam.

Pomocí kontextové nabídky lze také označit všechny výstřely, přidat ručně nový výstřel a sdílet celou nahrávku. Přidání výstřelu jsem opět vyřešil dialogovým oknem, kde je nutné vyplnit ručně čas výstřelu.

4.7 Konfigurace aplikace

Nastavení aplikace je rozděleno do dvou částí. Jednu část může konfigurovat uživatel a druhou část pouze programátor. Všechny konfigurovatelné položky jsou umístěny v XML souboru defaultSettings.xml a uloženy jako primitivní datové typy. Tento soubor je zkompilován do balíčku s aplikací.

V případě uživatelsky přístupných hodnot se jedná o výchozí konfiguraci. Změny v nastavení, které uživatel provede se udržují pomocí třídy SharedPreferences, její detailní popis se nachází v kapitole 2.1. Díky tomu, že všechny aktivity v aplikaci jsou potomky třídy BaseActivity, je konfigurace jednotná pro celou aplikaci, protože vytvoření a obsluha konfiguračního dialogu je umístěna pouze v této bázové třídě. Toto řešení pomáhá předcházet k nekonzistenci, či případným chybám v aplikaci. Navíc je zajištěno, že v celé aplikaci bude vždy dostupná právě aktuální konfigurace.

V aplikaci se nachází několik možností nastavení, které nejsou uživatelsky editovatelné. Jedná se o hodnoty, ke kterým se přistupuje na více místech aplikace. Jejich případná změna by znamenala úpravu kódu a jejich roztroušenost zvyšuje výskyt chyby. Proto je vhodné je umístit centrálně. Jelikož změna těchto hodnot silně ovlivňuje chování aplikace a jejich změna může způsobit nestabilitu, či nefunkčnost aplikace, nejsou proto uživatelsky editovatelné. Jedná se například o nastavení kompresního poměru při čtení zvukového souboru, případné úplné vypnutí této komprese, nebo maximální časový úsek zobrazovaný v grafu.

4.8 Grafická prezentace

Jako grafické elementy jsem se snažil co nejvíce použít univerzální systémové prvky. Tyto prvky korespondují se vzhledem operačního systému a designově zapadají do celku. Jednoduchý a hlavně použitelný vzhled je důležitá součást aplikace. Samozřejmě některé prvky je potřeba vytvořit vlastní. Elementy jako logo aplikace, případně hlavní charakteristický prvek, kterým se aplikace odlišuje od ostatních je potřeba vytvořit unikátní.

V mé aplikaci jsem jako logo použil obrázek zbraně uvnitř kruhu. Ikona je překryta modrou barvou, která se v mírně odlišných odstínech vyskytuje na více místech aplikace, například při vykreslování grafu. Ikona barevně a provedením zapadá do celkové stylizace aplikace. Tvorbu ikony aplikace není dobré podceňovat, protože je to prvek, který uživatelé uvidí nejčastěji. Ikona by měla být zapamatovatelná. Měla by jít lehce rozeznat mezi ostatními ikonami a co je nejdůležitější, měla by na první pohled vypovídat o povaze aplikace.

Dalším charakteristickým prvkem aplikace je nahrávací tlačítko umístěné na úvodní obrazovce. Tlačítko je tvořeno jednoduchou reprezentací mikrofonu. Celou obrazovku

jsem chtěl vytvořit tak, aby byla jednoduchá a hlavně lehce ovladatelná. Tlačítko je jediný ovládací prvek na úvodní obrazovce. Učinil jsem tak záměrně, aby nedocházelo k překlepům a uživatel se nemusel na činnost příliš soustředit. Navíc je takto aplikace ihned připravena k použití, stačí jen spustit a zmáčknout tlačítko. Na první pohled je jasné, že stiskem začne nahrávání zvuku i člověku, který aplikaci spustil poprvé a nemá o ní žádné informace.

Další obrázky aplikace můžete nalézt v příloze.

4.9 API dokumentace

Pro programátora je pro orientaci v aplikaci nejdůležitější API dokumentace. Na generování dokumentace jsou nástroje, které tuto činnost zautomatizují. Pro dokumentaci k této aplikaci jsem použil velmi populární nástroj JavaDoc. Pro tvorbu dokumentace se používá speciální druh komentářů ve zdrojovém kódu.

Kompletní dokumentaci naleznete na přiloženém CD a také na webu http://homel.vsb.cz/~tic0012/bc-prace/dokumentace.

5 Uživatelská příručka

Zde se nachází zjednodušená příručka pro nového uživatele. Detailnější popis a návod k aplikaci naleznete na adrese http://homel.vsb.cz/~tic0012/bc-prace/.

Uživatelská příručka ve formě webové stránky a instalační balíček aplikace se nachází na CD v příloze.

5.1 Instalace

Aplikace bohužel není umístěna v oficiálním obchodu s aplikacemi pro operační systém Android, tedy Google Play. Pro nainstalování aplikace je nutné stáhnout instalační balíček z webu určeného pro aplikaci http://homel.vsb.cz/~tic0012/bc-prace/. Tento balíček je potřeba nahrát na SD kartu Vašeho zařízení a poté spustit, tím se aplikace nainstaluje. Pro instalace aplikací z SD karty, je nutné mít v zařízení povolenu instalaci aplikací, které nepocházejí ze služby Market.

5.2 Ovládání

Základní možnost nahrávání je stisknutí tlačítka uprostřed obrazovky. Tlačítko je reprezentováno obrázkem mikrofonu. Po zahájení nahrávání se jeho barva změní na oranžovou. Pro zastavení nahrávání je potřeba stisknout tlačítko znovu. Tím se zahájí zpracování zvukového záznamu. Ve výchozím stavu se všechny pořízené nahrávky ukládají do hudební sbírky na SD kartě. Konkrétně do složky GunshotRecorder.

Rozšířené možnosti jsou spuštění nahrávání po uplynutí zvoleného času a náhodné spuštění nahrávání v nastaveném časovém rozmezí. Obě tyto volby jsou řešeny pomocí dialogů s volbou času. V případě náhodného spuštění se jedná o dvě dialogová okna. Nejprve pro dolní a poté pro horní časové rozmezí. Po nastavení parametrů pro spuštění, se zobrazí dialogové okno s informacemi. V případě náhodného spuštění je zobrazena informace střelci, že má vyčkat zvukového signálu. V případě nastavení po určité době, je zobrazen odpočet času. Po započetí nahrávání v obou případech zazní zvuková signalizace.

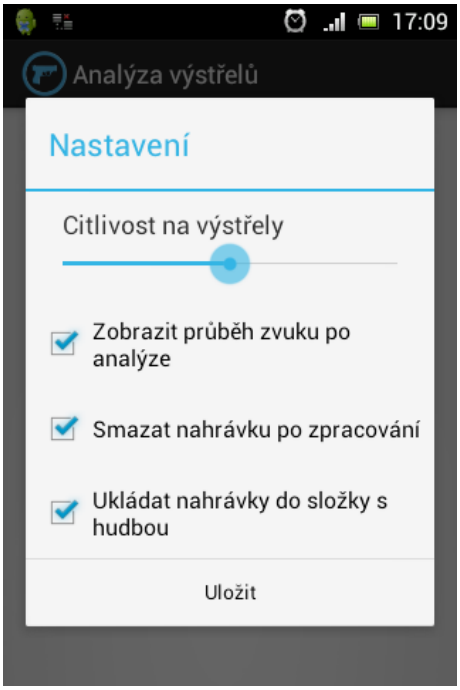
Po ukončení nahrávání a zpracování záznamu, je možné si prohlédnout graf, který zobrazuje průběh zvukového signálu. Pro pokračování dál, je nutné tento graf skrýt. Toto lze provést dlouhým stiskem na grafu, nebo stiskem kontextové nabídky a následným výběrem položky „*Schovat obrázek*“. Po schování obrázku, se zobrazí seznam všech nalezených výstřelů. Je možné přidat další výstřel ručně. Případně odebrat označení u neplatných výstřelů. Po stisknutí tlačítka uložit, se uloží pouze vybrané výstřely.

Do správy uložených výstřelů je možné přistoupit výběrem této možnosti z kontextové nabídky. Je možné editovat kategorie jejich dlouhým stiskem. Zálohu veškerých dat lze provést pomocí XML exportu. Tato možnost je v kontextové nabídce. XML soubor se ukládá na SD kartě do složky GunshotRecorder. Každý soubor je pojmenován podle data, kdy byl export pořízen.

Do detailu nahrávek a k samotným výstřelům lze přistoupit krátkým stiskem nahrávky v seznamu. Jejím dlouhým stiskem ji lze sdílet na sociálních sítích, případně pomocí dalších služeb. V detailu nahrávky lze ručně přidávat, či odebírat výstřely.

5.3 Nastavení

Nastavení aplikace je přístupné pomocí kontextové nabídky ze všech částí aplikace. Po stisku nabídky se otevře dialogové okno, kde je možné změnit citlivost detekce výstřelu, vypnout či zapnout zobrazení grafu po analýze zvuku. Dále zda se mají nahrávky ukládat do hudební sbírky a zda se má pořízená nahrávka po zpracování smazat. Během změn v nastavení aplikace se nacházíte na stejném místě v aplikaci a neztratíte tak přehled. Vzhled dialogového okna pro nastavení můžete vidět na obrázku 5.



Obrázek 5: Dialogové okno pro nastavení aplikace

5.4 Odinstalace

Odinstalování aplikace probíhá standardním způsobem. Tedy v nastavení telefonu v kategorii aplikace vyberete „Analýza výstřelů“ v českém jazyze a „Gunshot sound analyzer“ v případě, že je prostředí telefonu v angličtině. Po odinstalování aplikace, zůstanou zachovány všechny nahrávky umístěné v hudební sbírce. Zachovány zůstanou také všechny exporty dat, které jsou umístěny na externím úložišti, ve složce GunshotRecorder. Všechny ostatní data budou smazána.

6 Testování

K testování aplikace byly použity reálné nahrávky výstřelů, které obsahují ruch v pozadí, mluvenou řeč střelců, nabíjení zbraně, podavač terčů a další ruchy okolí. Při testování vyšlo najevo několik „nedostatků“ aplikace, které byly do finální verze odstraněny. Jedním z nich byla rychlost zpracování záznamu. Vyšlo najevo, že zpracování na zařízeních se slabší hardwarovou výbavou trvalo příliš dlouho. Implementoval jsem proto kompresi., kterou detailněji popisuji v kapitole 4.4.

Testování probíhalo na mém mobilním telefonu Sony Ericsson Xperia Mini PRO(SK17i). Telefon je vybaven jednojádrovým procesorem Qualcomm MSM8255 s typem jádra Scorpion o taktu 1 000 MHz. Velikost operační paměti 512MB. Verze operačního systému Android 4.0.4 Ice Cream Sandwich. Druhé testovací zařízení byl mobilní telefon Samsung Galaxy S II. Telefon je vybaven dvoujádrovým procesorem Samsung Exynos 4210 s jádry typu Cortex-A9 o taktu 1 200 MHz. Velikost operační paměti 1 024 MB. Verze operačního systému Android 4.1.2 Jelly Bean.

Pokud není uvedeno jinak, tak testy probíhaly ve výchozí konfiguraci aplikace, tedy s citlivostí nastavenu na střední hodnotu. Z důvodu úspory místa, je uvedeno vždy prvních šest výstřelů z nahrávky. Kompletní soupis jednotlivých testů se nachází v kapitole 6.2. Všechny testovací nahrávky jsou umístěny na CD jako příloha. Číslo odpovídající nahrávky je vždy uvedeno u patřičné tabulky s výsledky.

6.1 Poznámky k testům

První dva záznamy obsahují výstřely „běžných“ střelců. Od třetího záznamu nahrávky obsahují účastníky šampionátu Israeli Open 2013 v divizi Production včetně vítěze tohoto šampionátu. Zde se již projevuje jistá chybovost aplikace.

Jako čas skutečného výstřelu je brán první okamžik, kdy amplituda vystoupila na maximální hodnotu, tedy počáteční hrana výstřelu. U záznamu číslo 2 aplikace vyhodnotila druhý výstřel dvakrát. Poprvé v čase 7.629s a podruhé 7.874s. Výsledky tohoto záznamu jsou v tabulce 7 a ukázka na obrázku 6. Tento jev se objevuje velmi zřídka. Jako řešení jsem zvolil minimální časový odstup mezi jednotlivými výstřely. Toto se však ukázalo jako ne příliš spolehlivé. Lepším řešením, by bylo odfiltrování ruchů z nahrávky například pomocí Fourierovy transformace. Falešený výstřel nebyl nalezen žádný.

6.2 Výsledky testů

Aplikace v záznamu číslo 3 nezaznamnela velký počet výstřelů. Je to způsobeno velmi velkou frekvencí střelby. V záznamu je šest výstřelů za vteřinu. To způsobilo, že amplitudy jednotlivých výstřelů jsou velmi blízko sebe a téměř splývají. Tento problém by vyřešil důkladnější algoritmus, který použil Fourierovu transformaci pro spektrální analýzu zvuku. Tato metoda je však výpočetně velmi náročná. Jelikož se takto rychlá střelba vyskytla pouze v jednom testovacím záznamu, nezapočítal jsem jej do celkové odchylky. Odchylky jednotlivých záznamů jsou umístěny v tabulce 14. Průměrná odchylka nalezení výstřelu je 35.68ms.

	výstřel 1	výstřel 2	výstřel 3	výstřel 4	výstřel 5	výstřel 6
Skutečný čas [s]	6.082	9.544	10.746	11.117	11.504	11.863
Nalezený čas [s]	6.143	9.619	10.827	11.152	11.551	11.960

Tabulka 6: Testovací nahrávka č. 1, tréningová střelba, citlivost 5/10

	výstřel 1	výstřel 2	výstřel 3	výstřel 4	výstřel 5	výstřel 6
Skutečný čas [s]	2.865	7.653	13.231	16.793	23.060	27.594
Nalezený čas [s]	2.882	7.629	13.289	16.823	23.093	27.624

Tabulka 7: Testovací nahrávka č. 2, tréningová střelba, citlivost 5/10

	výstřel 1	výstřel 2	výstřel 3	výstřel 4	výstřel 5	výstřel 6
Skutečný čas [s]	4.358	4.619	4.754	5.000	5.147	5.245
Nalezený čas [s]	4.378s	4.619s	-	5.015s	-	-

Tabulka 8: Testovací nahrávka č. 3, šampionát Israeli Open 2013, citlivost 8/10

	výstřel 1	výstřel 2	výstřel 3	výstřel 4	výstřel 5	výstřel 6
Skutečný čas [s]	22.750	22.883	23.070	23.188	23.450	23.687
Nalezený čas [s]	22.777	22.939	23.103	23.273	23.433	23.681

Tabulka 9: Testovací nahrávka č. 4, šampionát Israeli Open 2013, citlivost 8/10

	výstřel 1	výstřel 2	výstřel 3	výstřel 4	výstřel 5	výstřel 6
Skutečný čas [s]	3.141	3.427	3.668	3.879	4.102	4.655
Nalezený čas [s]	3.194	3.435	3.683	3.927	4.172	4.729

Tabulka 10: Testovací nahrávka č. 5, šampionát Israeli Open 2013, citlivost 6/10

	výstřel 1	výstřel 2	výstřel 3	výstřel 4	výstřel 5	výstřel 6
Skutečný čas [s]	1.435	1.698	2.284	2.643	5.944	6.578
Nalezený čas [s]	1.474	1.719	2.284	2.695	5.960	6.604

Tabulka 11: Testovací nahrávka č. 6, šampionát Israeli Open 2013, citlivost 7/10

	výstřel 1	výstřel 2	výstřel 3	výstřel 4	výstřel 5	výstřel 6
Skutečný čas [s]	0.022	2.660	2.964	3.333	3.668	6.517
Nalezený čas [s]	0.062	2.654	2.977	3.380	3.704	6.559

Tabulka 12: Testovací nahrávka č. 7, šampionát Israeli Open 2013, citlivost 9/10

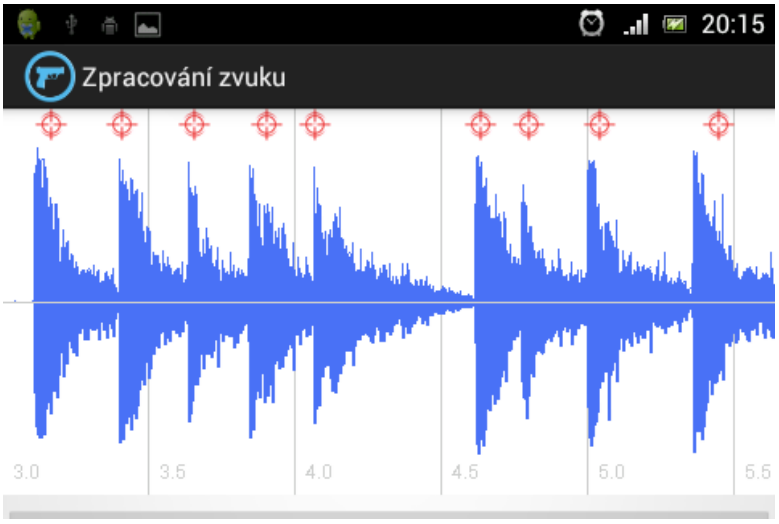
	výstřel 1	výstřel 2	výstřel 3	výstřel 4	výstřel 5	výstřel 6
Skutečný čas [s]	1.953	2.157	2.774	3.023	3.387	3.612
Nalezený čas [s]	2.000	2.163	2.814	3.054	3.448	3.570

Tabulka 13: Testovací nahrávka č. 8, šampionát Israeli Open 2013, citlivost 8/10

Záznam č.	1	2	3	4	5	6	7	8
Odchylka [ms]	49.5	24	-	37.3	44.7	25.7	30.7	37.84

Tabulka 14: Průměrné odchylky jednotlivých testů

Obrázek 6: Dvakrát označený výstřel v testu



Obrázek 7: Průběh testovací nahrávky č. 5

7 Závěr

V této práci jsem provedl analýzu všech možností, jak ukládat data na platformě Android. Zhodnotil jsem jejich přednosti i nevýhody a uvedl příklady využití jednotlivých možností. Tyto informace mohou být značným informačním přínosem například pro začínající vývojáře. Navíc osvětlí značné omezení datové kapacity mobilních zařízení.

Vytvořil jsem třídu pro zaznamenání bezeztrátového zvuku z mikorofonu zařízení a jeho následné uložení do multiplatformního formátu. Tato třída v sobě nenese žádné závislosti a byla vyvíjena s důrazem na co největší kombatibilitu mobilních zařízení s platformou Android. Je samostatně použitelná a může se bez úprav použít v dalších aplikacích.

Implementoval jsem vlastní algoritmus pro nalezení výstřelů ve zvukové stopě.

Všechny tyto dílčí kroky jsem využil při tvorbě aplikace pro operační systém Android. Při návrhu a implementaci byl kladen důraz na čistotu architektury a použití návrhových vzorů. Především vzoru MVC a Dependency Injection. Aplikace byla s pomocí střeleckého klubu testována v reálném prostředí a dosahovala poměrně dobrých výsledků. Test ukázal, že u zkušených střelců, je nutné upravit citlivost. Je tedy dobré, vždy provést alespoň jeden testovací záznam pro konfiguraci a na základě něj upravit citlivost apliakce. Na základě přání samotných střelců, které vznikly při testování, se do aplikace implementovaly dodatečné funkční prvky, aby co nejvíce vyhovovala jejich požadavkům. Na přání jsem přidal možnost automatického zastavení nahrávání zvuku po nastaveném čase, či přidání výstřelu pomocí dlouhého stisku na obrázku průběhu zvuku.

Do budoucna hodlám aplikaci podporovat, především co se týká přání ze strany uživatelů. Věřím, že jim aplikace bude dobře sloužit.

8 Reference

[1] KRUMNIKL, Michal. VŠB – TECHNICKÁ UNIVERZITA OSTRAVA. *TAMZ II* [online]. Ostrava, 2011, 2012 [cit. duben 2013]. Dostupné z: http://tamz2.mrl.cz/

[2] Options. GOOGLE. *Android Developers* [online]. [cit. duben 2013]. Dostupné z: http://developer.android.com/guide/topics/data/data-storage.html

[3] MAHER, Robert C.. Electrical and Computer Engineering Montana State University. *Acoustical Characterization of Gunshots* [online]. [cit. duben 2013]. Dostupné z: http://www.coe.montana.edu/ee/rmaher/publications/maher_ieeesafe_0407_prezo.pdf

[4] FOWLER, Martin. *Patterns of enterprise application architecture.* Boston: Addison-Wesley, c2003, xxiv, 533 p. ISBN 03-211-2742-0.

[5] VYBÍRAL, David. *Mobilní aplikace pro analýzu zvukových vstupů mobilního zařízení za účelem detekce epileptických záchvatů.* Ostrava, 2012. DIPLOMOVÁ PRÁCE. VŠB – Technická univerzita Ostrava. Vedoucí práce doc. Ing. Ondřej Krejcar, Ph.D.

[6] Activities. GOOGLE. *Android Developers* [online]. [cit. duben 2013]. Dostupné z: http://developer.android.com/guide/components/activities.html

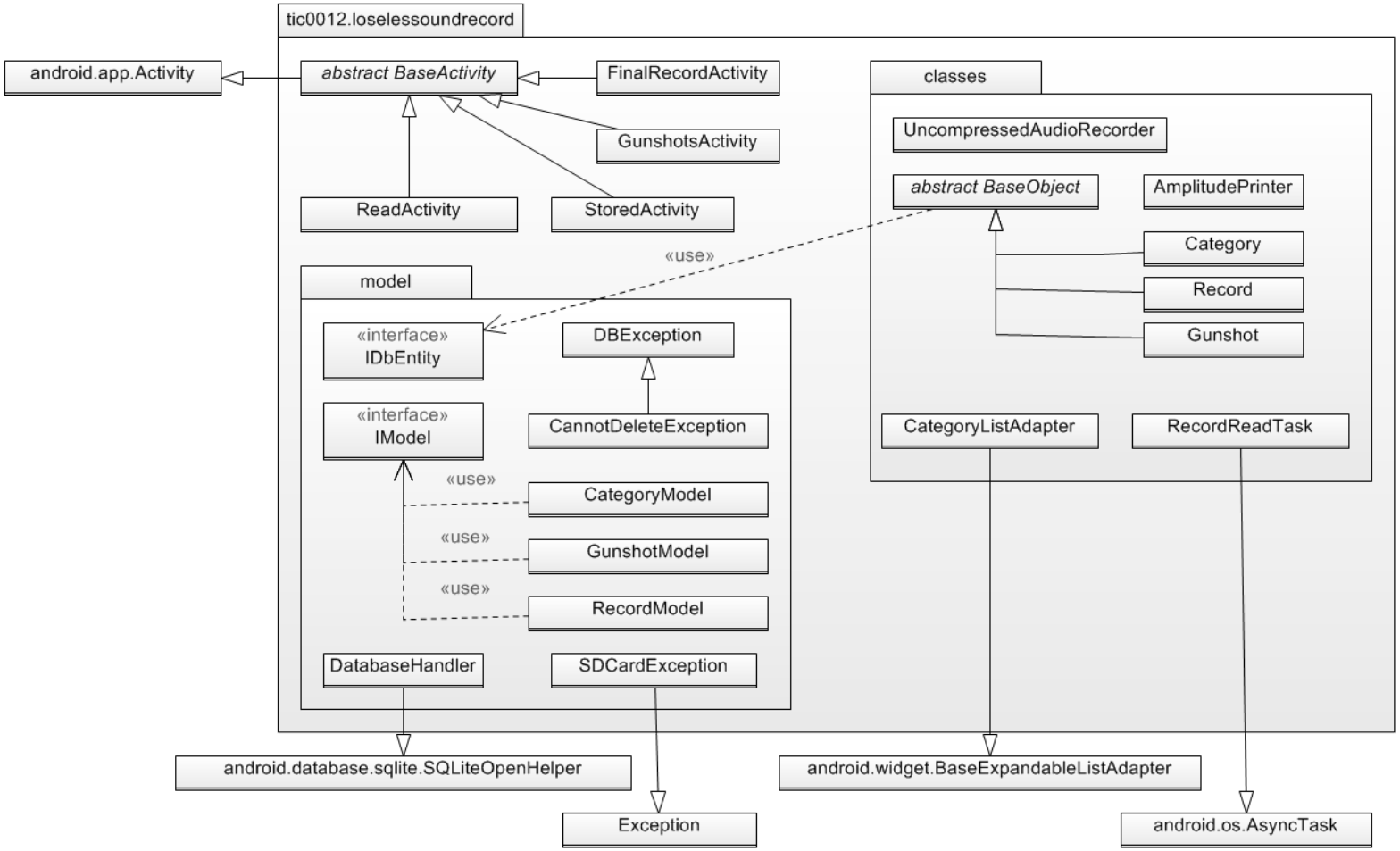
[7] Vyvíjíme pro Android: Fragmenty a SQLite databáze. *Zdroják* [online]. 2012 [cit. duben 2013]. Dostupné z: http://www.zdrojak.cz/clanky/vyvijime-pro-android-fragmenty-a-sqlite-databaze/

[8] Datatypes In SQLite Version 3. *SQLite* [online]. [cit. duben 2013]. Dostupné z: http://www.sqlite.org/datatype3.html

[9] WAVE PCM soundfile format. [online]. [cit. duben 2013]. Dostupné z: https://ccrma.stanford.edu/courses/422/projects/WaveFormat/

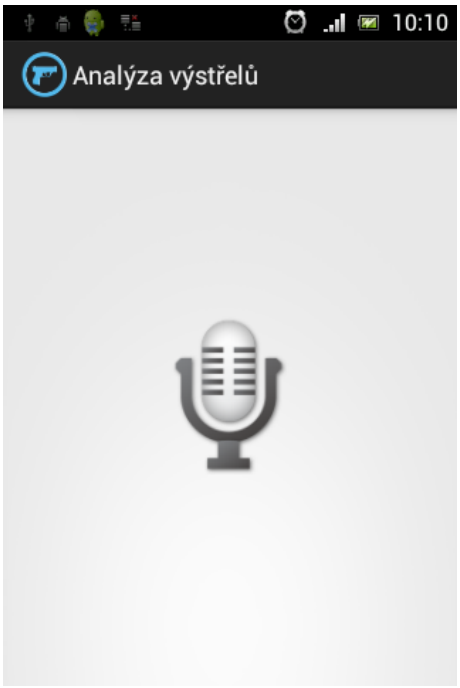
[10] AudioRecord. GOOGLE. *Android Developers* [online]. [cit. duben 2013]. Dostupné z: http://developer.android.com/reference/android/media/AudioRecord.html

A Diagramy

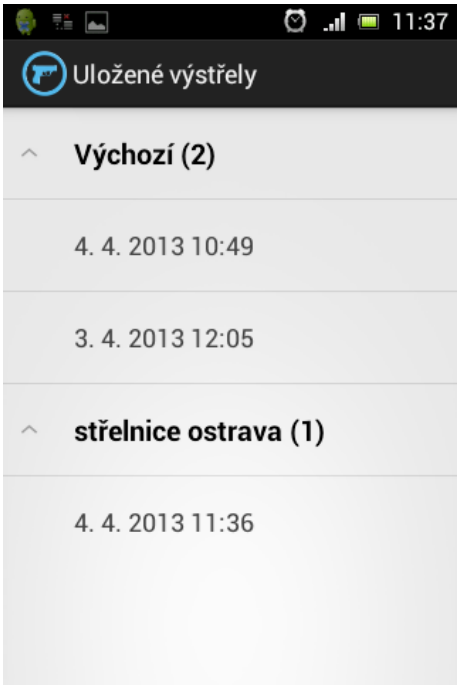


Obrázek 8: Třídní diagram

B Obrázky z aplikace



Obrázek 9: Úvodní obrazovka



Obrázek 10: Správa kategorií a nahrávek

C Obsah CD

- **bc_prace.pdf** - bakalářská práce v elektronické podobě
- **LoselesSoundRecord.apk** - instalační balíček aplikace
- **LoselesSoundRecord/** - kompletní projekt se zdrojovými kódy aplikace
- **dokumentace/** - API dokumentace
- **web/** - informační web s návodem
- **testovaci_nahravky/** - nahrávky použité pro testování aplikace