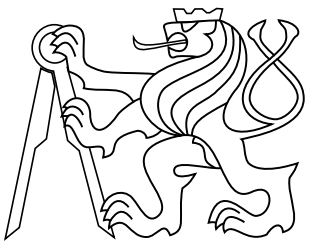
bakalářská práce

Vývoj aplikací na platformě Android

Pavel Zářecký



Květen 2016

Komárek Martin Ing.

České vysoké učení technické v Praze
Fakulta elektrotechnická, Katedra počítačů

Poděkování

Chtěl bych zde především poděkovat vedoucímu své práce, Ing. Martinu Komárkovi, za jeho rady a vedení. Určitě bych zde také rád poděkoval svým rodičům a příbuzným za podporu a za to, že jsem dnes tam, kde jsem a mohu na tomto projektu vůbec pracovat. Mé díky patří i všem ostatním, kteří mě při studiu podporovali.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 27.5.2016 .....

Abstrakt

Cílem bakalářské práce je vylepšení prototypů dvou nezávislých aplikací pro platformu Android. V první řadě je dokončena práce na vlastní aplikaci Dětská chůvička, která byla původně vyvíjena v rámci předešlé semestrální práce. Za druhé byl vylepšen prototyp aplikace na správu účtů a objednávek v restauraci, Cashbob, který jsem převzal od Tomáše Hogenauera. Nakonec byl v této aplikaci implementován tisk účtenek.

Klíčová slova

Mobilní aplikace; Android; Wi-fi; Iterativní vývoj, Bluetooth, REST, Tisk; ...

Abstrakt

The main goal of this bachelor thesis is to improve two prototypes of independent applications for the Android platform. In the first place, a work on the application Baby monitor, which was originally developed as a previous semestral project, is finished. Secondly, a functionality of a prototype of an account and order management application Cashbob, which I took over from Tomáš Hogenauer, is tested, improved upon and printing is implemented.

Keywords

Mobile application; Android; Wi-fi; Bluetooth, Iteration based development, REST, Printing; ...

Obsah

Zkratky

Význam zkratek užívaných v práci:...

API	Application program interface
REST	Representational state transfer
JVM	Java virtual machine
ART	Android runtime
XML	Extensible markup language
APK	Android application package
JAR	Java archive
IDE	Integrated development environment
SMS	Short message service
SQL	Structured query language

Seznam obrázků

Seznam tabulek

1 Úvod

Vždy jsem měl zájem o vývoj aplikací pro mobilní telefony, jelikož považuji toto odvětví jako mým zájmům nejbližší. V posledních letech zásadně rostou počty aplikací dostupných na v současnosti nejpoužívanější mobilní systém, jímž je Android[1], a ani já jsem tedy neváhal přispět - volba tématu pro mě tedy od počátku jasná.
V této práci jsem se věnoval vývoji hned dvou aplikací. V první fázi bude doveden do zdárného konce vývoj aplikace "Baby monitor", dětské chůvičky. Tu jsem již nějakou dobu postupně vyvíjel, především v rámci školního semestrálního projektu. Na závěr této fáze bude aplikace testována, nasazena a volně k dispozici v obchodě Google Play[2], což je primární zdroj softwaru pro Android.
Druhá fáze se týká pokračování vývoje restauračního systému Cashbob, respektive jeho mobilního (tabletového) modulu. CashBob je vícemodulový restaurační pokladní systém psaný v jazyce Java, vyvíjený již několik let v rámci závěrečných prací katedry počítačů[3]. Komunikační model systémů v průběhu let postupně procházel různými transformacemi, zejména přechodem z technologie Java RMI (Remote method invocation[4]) na standard REST[5]. REST využívá zmíněný Android modul systému, původně vyvinutý Tomášem Hogenauerem pro jeho závěrečnou práci[6].
Jelikož se zmíněné REST rozhraní od původního odevzdání aplikace změnilo, především z důvodu práce na nové verzi serverového modulu, budu nejdříve testovat funkčnost aplikace vůči nejnovější verzi a provedu nutné změny. Následovat bude kvalitativní kontrola kódu spolu s testy uživatelské přívětivosti a pokud to bude nutné, budou provedeny ještě další změny. Na závěr proběhne implementace tisku účtenek lokálně a vyžádání tisku na serverové tiskárně.

1.1 Struktura dokumentu

Vzhledem k tomu, že to jediné, co mají obě vyvíjené aplikace společné, je cílová platforma a jazyk, rozhodl jsem se celou práci rozdělit celkem na tři části. Zaprvé v následující kapitole nejdříve popíši zázemí a použité technologie cílové platformy. Poté kapitoly 3 až 5 jsou věnované vývoji aplikace Baby monitor, zatímco kapitoly 6 až 10 popisují postup plnění druhé části zadání, tedy testování a implementace aplikace CashBob.

Postup při implementaci aplikace Baby monitor

Dále bych rád ještě zmínil, že vývoj aplikace Baby monitor bude probíhat iterativně, tedy budou se opakovat fáze návrhu, implementace, testování a nasazení do té doby, než shledám výstup nějaké iterace uspokojivým.

2 Základní analýza platformy Android

V první kapitole bych se rád věnoval cílové platformě jako takové. Jak jsem již nastínil v úvodu, jedná se o mobilní operační systém Android. Primárním programovacím jazykem pro tento systém je Java[7], aplikace budou tedy psány v ní.

2.1 Java v Androidu

Java je interpretovaný programovací jazyk a programy v ní psané jsou spouštěné v takzvaném virtuálním stroji, neboli JVM. Díky tomu je v teorii možné přenášet aplikace mezi rozdílnými operačními systémy, jako je Windows, Linux, nebo hlavně, Android. Bohužel, virtuální stroj Androidu, Dalvik (nebo nově také ART), neobsahuje stejnou sadu API jako již zmíněné standardní JVM. Toto se týká především knihoven pro vykreslování uživatelského rozhraní Swing či AWT, které v Dalviku prostě vůbec nejsou a jsou nahrazeny proprietárním API. Tato skutečnost byla hlavni bariérou, kterou jsou při vývoji musel překonat - dříve jsem totiž aplikace vyvíjel pouze v čisté Javě.[7]

2.2 Uživatelské rozhraní

Následující informace jsem čerpal především z portálu Android Developers[8].Proprietární API Androidu sestává z několika základních částí, které při vývoji aplikací zcela jistě použiji:

Layout

Je definován většinou v XML souboru, který obsahuje rozmístění jednotlivých prvků rozhraní (Widgetů)[9]. Pokud není možné takto uživatelské rozhraní plně popsat, například při implementaci seznamu kde není předem znám počet prvků v něm, je nutné použít adaptér - prvek, který widgety do UI umístí za chodu.

XML soubory jako takové

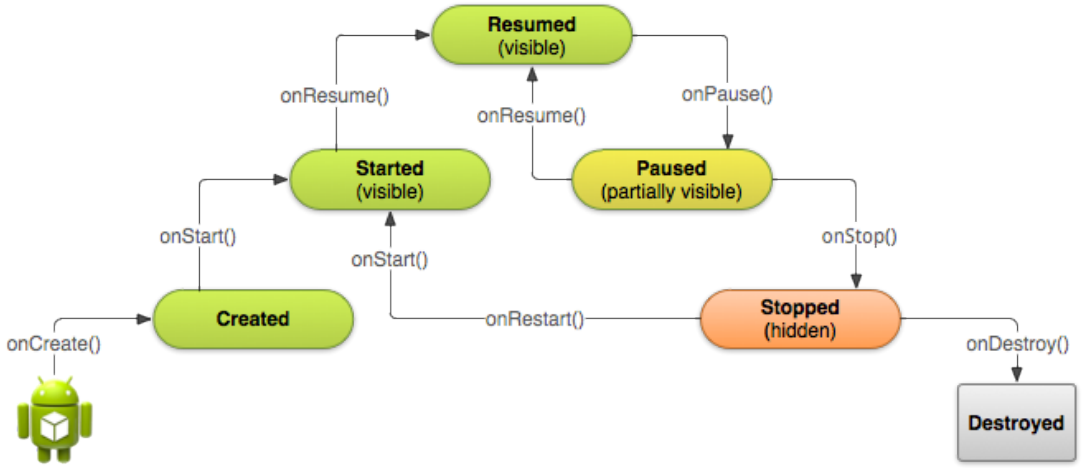
Jsou soubory obsahující text strukturovaný do speciálních tagů jako například <tag>Text</tag>. Výhodou takovýchto strukturovaných dokumentů je, že jsou čitelné jak strojově, tak člověkem a proto se hodí pro návrh rozhraní.

Aktivita

Je základní kámen Androidí aplikace a lze ji definovat jako samostatnou "obrazovku", která nahrazuje okna známá ze stolních operačních systémů. Každé aktivitě je nutné přiřadit layout, který je nejdříve nutné takzvaně nafouknout - což je proces, při kterém se jednotlivé widgety instancují do objektů, které je pak možné programaticky dohledat a editovat pomocí metody findView(), kterou každá aktivita dědí. Aktivity mohou obsahovat i fragmenty.

Fragment

Je ucelená část rozhraní dědící lyfecyklus nadřazené Aktivity, kterou popíši v některé z následujících kapitol.



Obrázek 1 Životní cyklus aktivity systému Android. Jedná se o způsob řízení, který například ve světe Desktopových aplikací vůbec neexistuje.

2.3 Formát aplikací

Aplikace pro systém Android musí být exportovány do takzvaných APK balíků, které obsahují všechny zdroje aplikace, jako jsou použité externí knihovny, bitmapy, zvukové soubory, XML layouty a samozřejmě také Java třídy přeložené do bytecode. Jedná se vlastně o speciální formu JARu.

2.4 Manifest

XML soubor obsahující nejzákladnější popis aplikace, jako například její název a jaké aktivity aplikace používá. Dále je obsažen seznam oprávnění, které aplikace potřebuje k chodu. Pro přístup k API sytému Android si aplikace musí v manifestu vyžádat povolení. Uživatel tato povolení může zkontrolovat při instalaci aplikace a proto by jich měla mít co nejméně, opravdu jen ty nejnutnější, jinak je zde risk, že uživatel aplikaci vůbec nebude používat z nedůvěry k ní. Přeci jen, aplikace obsahující tapety na plochu nepotřebuje například přístup ke kontaktům. Soubor manifestu se po překladu nachází v kořenovém adresáři APK aplikace.

2.5 Překlad a Gradle

Nejjednodušší způsob, jak přeložit aplikaci pro Android, je k tomu použít nějaké IDE, v mém případě se jednalo o Android studio, které k překladu používá Gradle[10]. Gradle je nástroj pro sestavování a kompilaci a umožňuje inkluzi proprietárních knihoven a zdrojů - tzv. Dependency. S nástrojem jsem se zde setkal úplně poprvé, práci s ním ale dle mého názoru vůbec není složitá.

3 Baby monitor - úvod a rešerše

Jak jsem shrnul v úvodu, rád bych se v této a v několika dalších kapitolách věnoval práci na vlastní aplikaci Baby monitor.

3.1 Historie

Na aplikaci jsem začal pracovat ještě v rámci semestrálního projektu také pod vedením pana Ing. Komárka. Je ale dobré zmínit, že tehdy byl cíl trochu odlišný. Aplikace měla původně jiný název a také zaměření - měla usnadnit výklad turistickému průvodci rozesíláním zvuku v reálném čase na co nejširší spektrum klientských zařízení. Při analýze samotné se ale vyskytly zásadní problémy, které zapříčinily změnu zadání. Problémy se týkaly především nemožnosti připojení dostatečného počtu zařízení, při kterém by provoz aplikace měl smysl - rešerši a analýzu původní aplikace pro turistické průvodce přikládám k nahlédnutí na přiložený disk.

3.2 Cíl

Současným cílem je vytvořit aplikaci, která by mohla zastoupit tzv. dětskou chůvičku, což je sada jednoúčelových zařízení pro monitorování spánku dětí, kde jedno vždy funguje jako dětský modul, zanechaný blízko dítěte, nahrávající zvuky v okolí pomocí mikrofonu. Ostatní zařízení v sadě se k tomuto modulu připojí a přehrávají. Můj systém nahradí tato jednoúčelová zařízení mobilními telefony, či tablety s operačním systémem Android, na kterém by běžela aplikace Baby monitor. Zařízení by místo obyčejného rádia používala ke komunikaci rozhraní, která se většinou používají na cílové platformě. Konkrétní způsoby připojení rozeberu dále.

3.3 Rešerše

Dále bych rád uvedl již existující systémy pro monitorování spánku děti. Analyzoval jsem jak zmíněná jednoúčelová zařízení, tak i systémy pro platformu Android, které se funkcionalitou blíží tomu mému. Analýzou funkcí dostupných v rámci již existujících řešení jsem posoudil, jaké funkce by aplikace měla mít, aby obstála v konkurenčním prostředí, protože závěrem bude aplikace publikována na Google Play.

3.3.1 Rešerše fyzických zařízení

Funkce/Zařízení ->	Orava BM-4	Motorola MBP8	Motorola MBP 421	Motorola MBP 27T
Přenos videa	-	-	Ano	Ano
Přenos zvuku	Ano	Ano	Ano	Ano
Dosah	100m	300m	300m	300m
Upozornění	Ano	Ano	Ano	Ano
Nastavení citlivosti	Ano	Ano	Ano	Ano
Osvětlení	Ano	Ano	Ne	Ne
Natáčení	-	-	Ano	Ano
Teploměr	Ne	Ne	Ne	Ano
Počet připojitelných z.	1:1	1:1	1:1	4:1
Noční vidění	-	-	Ano	Ano
Akumulátor	Ano	Ano	Ano	Ano
Cena	849 Kč	939 Kč	2290 Kč	4999 Kč

Tabulka 1 Srovnání jednotlivých chůviček. Vyhledáno ke dni 10. 11. 2015 skrze portál Heureka.cz

3.3.2 Zhodnocení rešerše fyzických zařízení

Dětské chůvičky tedy mají vesměs totožné funkce. Dražší modely navíc podporují i přenos videa a připojení více kamerových modulů naráz. Důležité zjištění také je, že jednoúčelové chůvičky postrádají možnost např. nahrávání zvuků/videa a také možnost odeslat SMS v případě nutnosti – Vzhledem k tomu, že jsou tato zařízení již z principu určena pro aktivní monitoring, tyto funkce vlastně ale nejsou ve většině případů třeba.

3.3.3 Rešerše aplikací na Google Play

Aplikace jsem na Google Play vybíral v závislosti na hodnocení (4.0+), počtu stažení (10000+) a dle toho, zda umí v reálném čase přenášet zvuk po síti.

BabyCam

Aplikace může fungovat ve dvou režimech: „Camera“ a „Watch“. Zařízení ve Watch módu může ovládat některé funkce Camera zařízení, jako například zvuk notifikací a svítilnu. Stream je k dispozici i prostřednictvím webového prohlížeče – Aplikace umí pracovat i jako web server.

Dormi

Aplikace Dormi opět funguje ve dvou režimech, jako i většina ostatních Android chůviček. Přenáší zvuk i video a automaticky kalibruje okolní hluk tak, aby dokázala zjistit změny okolního prostředí bez nutnosti ruční kalibrace. V rodičovském módu jsou o protějším zařízení zobrazovány užitečné informace jako například stav baterie i s předpokládanou dobou do vybití nebo seznam zmeškaných hovorů. Je také možné promlouvat k dítěti skrze rodičovské zařízení.

Baby Phonic

Jednoduchá chůvička, která svým zaměřením odpovídá spíše páru webové kamery a přehrávače – neobsahuje totiž žádné funkce speciálně pro podporu svého zařazení. Opět je možné párovat dvě zařízení naráz.

Munchkin

Aplikace na rozdíl od již zmíněných nepracuje v rámci lokální Wi-fi, ale skrze internetový server vývojáře. Díky tomu tedy má v podstatě neomezený dosah. Sám sem ji bohužel nemohl vyzkoušet, jelikož klient nepodporuje verzi Androidu mého druhého zařízení. Dle informací na Google Play ale umí mimo přenosu videa/audia také automaticky upozornit na zvýšený hluk alarmem.

3.3.4 Srovnání

Funkce/Aplikace ->	BabyCam	Dormi	Baby Phonic	Munchkin
Přenos videa	Ano	Ano	Ano	Ano
Přenos zvuku	Ano	Ano	Ano	Ano
Dosah	Lokální síť	Internet	Lokální síť	Internet
Upozornění	Ne	Ano	Ne	Ano
Nastavení citlivosti	-	Automatické	-	Ano
Osvětlení	Ano	Ne	Ne	Ano
Natáčení	Ne	Ne	Ne	Ne
Teploměr	Ne	Ne	Ne	Dle HW zařízení
Počet připojitelných z.	1:1	1:N	N:1	1:1
Promlouvání zpět	Ne	Ano	Ne	Ano
Verze aplikace	1.3	3.1	1.0.6	0.7.8
Verze OS	2.3+	2.3+	2.2+	2.3.3+
Počet stažení	100000+	500000+	10000+	50000+
Počet Hodnocení	649	6000	17	648
Hodnocení aplikace	4.1	4.4	4.0	4.0

Tabulka 2 Srovnání jednotlivých chůviček. Vyhledány ty, které mají hodnocení nad 4.0 s počtem stažení více než 10000. Vyhledáno ke dni 11. 10. 2015, od té doby se mohly na Google Play objevit další aplikace, ty již ale neměly žádný dopad na mém rozhodnutí o inkluzi funkcí.

Aplikace jsou k dispozici ke stažení na následujících adresách:

BabyCam `play.google.com/store/apps/details?id=com.arjonasoftware.babycam`
Dormi `http://play.google.com/store/apps/details?id=com.sleekbit.dormi`
BabyPhonic `http://play.google.com/store/apps/details?id=air.com.babyphonic.lite.v1`
Munchkin `http://play.google.com/store/apps/details?id=com.munchkinmonitor`

3.4 Závěr rešerše

Funkce jednotlivých aplikací jsou tedy velmi podobné – Přenos zvuku a videa v reálném čase v rámci lokální sítě, u některých aplikací také provázen dodatečnými vlastnostmi, jako promlouvání zpět k dítěti, ovládání osvětlení a omezení hlasitosti notifikací a vyzvánění. Záměrně jsem vynechal všechny aplikace, které nepodporovaly alespoň přenos zvuku v reálném čase. Takových je na Google Play spousta, většina z nich ke spojení s rodičem využívá SMS nebo Telefonní hovor, pokud se hluk v okolí dítěte drží po dobu několika sekund nad zvolenou hranicí. Pokud by se rozšíření mé aplikace mělo výše zmíněným vyrovnat, navrhoval bych sloučení funkcí dvou druhů chůviček: Přenos zvuku a videa v reálném čase pomocí nějakého existujícího protokolu a také notifikace pomocí SMS či zavoláním na zvolené číslo.

4 Základní návrh funkcionality aplikace Baby monitor

Na úvod jsem zpracoval sadu funkčních a nefunkčních požadavků na výslednou aplikaci. Rozhodl jsem se pro začátek nahrnout všechny funkce, které by výsledná aplikace měla obsahovat, vzhledem k výsledku rešerše a také z důvodu vlastních preferencí. Funkce výsledné aplikace se pravděpodobně budou od následujícího seznamu lišit.
Co se týče odhadu složitosti dílčích požadavků, nebyl jsem schopen stanovit vypovídající hodnotu čistě z toho důvodu, že zatím nemám dostatečné zkušenosti s prací pro daný operační systém a platformu. Snažil jsem se tedy určit alespoň slovní popis tam, kde se hodí, podle následujících kritérií:

Nízká složitost	Funkce již v API existuje s takovými vlastnostmi, že nejsou nutné téměř žádné úpravy
Střední složitost	Funkce je podpořena dostatečně rozsáhlým API, které bude možné využít při implementaci
Vysoká složitost	Funkci bude nutné vytvořit s minimální podporou cílového API

Dále každý dílčí požadavek sdílí zadavatele, prioritu a složitost nadřazeného požadavku, pokud jsem u něj neuvedl jinak.

Funční požadavky

RQ 1 - Informace o spojení

Zadavatel	Martin Komárek

Systém bude uživateli poskytovat relevantní informace o spojení a stavu k zajištění použitelnosti aplikace jako dětské chůvičky

RQ 1.1 - Informace o Slave zařízení

Zadavatel	Pavel Zářecký
Priorita	Vysoká
Složitost	Střední

Prototyp bude v režimu Master schopen poskytovat uživateli informace o Slave zařízení, aby uživatel mohl sledovat jeho stav

RQ 1.2 Informace o Master zařízení

Zadavatel	Pavel Zářecký
Priorita	Vysoká
Složitost	Střední

Prototyp bude v režimu Slave schopen poskytovat uživateli informace o Master zařízení, aby uživatel mohl sledovat jeho stav

RQ 1.2.1 - Informace o stavu baterie

Zadavatel	Pavel Zářecký
Priorita	Vysoká
Složitost	Střední

Prototyp bude v režimu Slave podporovat zobrazení stavu baterie Master zařízení, ke kterému je připojený, aby uživatel mohl včas reagovat na slábnutí baterie

RQ 2 - Přenos záznamu

Zadavatel	Martin Komárek

Systém bude podporovat přenos záznamu mezi zařízeními v reálném čase, jelikož se jedná o nutnou funkci dětské chůvičky

RQ 2.1 - Přenos zvuku kontinuálně

Zadavatel	Pavel Zářecký
Priorita	Vysoká
Složitost	Střední

Systém bude podporovat přenos zvuku v reálném čase, aby umožnil uživateli přímý odposlech

RQ 2.2 - Přenos zvuku na vyžádání

Zadavatel	Pavel Zářecký
Priorita	Nízká
Složitost	Střední

Systém bude podporovat přenos zvuku na vyžádání, aby snížil spotřebu baterie

RQ 2.3 - Přenos videa na vyžádání

Zadavatel	Pavel Zářecký
Priorita	Nízká
Složitost	Střední

Systém bude podporovat přenos videa v reálném čase, aby umožnil uživateli dokonalý přehled o tom, co se u Master zařízení odehrává

RQ 3 - Upozornění na události

Zadavatel	Martin Komárek

Systém bude automaticky upozorňovat uživatele na předem nastavené události - např. Zvýšení hluku nad nastavenou mez, či vybíjení baterie, za účelem upoutání pozornosti

RQ 3.1 - Zvukové upozornění

Zadavatel	Martin Komárek
Priorita	Vysoká
Složitost	Nízká

Aplikace bude uživatele upozorňovat hlasitým zvukovým signálem

RQ 3.2 - Upozornění pomocí SMS

Zadavatel	Pavel Zářecký
Priorita	Nízká
Složitost	Střední

Aplikace bude umožňovat i zaslání SMS na přednastavené číslo, pokud dojde k přerušení spojení, nebo žádné Slave zařízení není připojené

RQ 4 - Pořizování záznamu

Zadavatel	Pavel Zářecký
Priorita	Nízká
Složitost	Vysoká

Systém bude umožňovat uživateli nahrávání a přehrávání zvukového streamu a také analýzu pořízeného zvukového záznamu, jelikož jsem shledal takovou funkcionalitu za přínosnou

Nefunkční požadavky

RQ 5 - Operační systém Android

Zadavatel Martin Komárek

Systém bude navržen jako aplikace pro operační systém Android, protože se jedná o nedílnou součást zadání projektu

RQ 5.1 - Verze systému

Zadavatel Pavel Zářecký

Aplikace bude požadovat operační systém Android alespoň ve verzi 4.0.3, protože tato a novější verze obsáhnou alespoň 98% zařízení na trhu

RQ 6 - GUI

Zadavatel Pavel Zářecký

Aplikace bude obsahovat jednoduchý grafický interface, protože hlavním cílem je uživatelská přívětivost a navíc systém Android užití textového interface ani neumožňuje

RQ 6.1 - Dotykové rozhraní

Aplikace bude podporovat ovládání pomocí dotykové obrazovky, protože takovýto způsob ovládání je pro danou platformu (Android) výchozí

RQ 6.2 - Rychlá odezva

Grafické rozhraní bude navrženo tak, aby jeho reakce byly co nejkratší, typicky do 1 vteřiny, protože veškerá funkčnost aplikace bude v reálném čase

Volba minimálního API

Každá nová verze operačního systému Android obsahuje i novou verzi API, tedy rozhraní pro komunikaci se systémem a hardwarem. Aplikace pro Android tedy musí specifikovat, pro kterou verzi toho API je aplikace určena a na jaké minimální verzi vůbec poběží. Při výběru API pro aplikaci Baby monitor jsem se řídil statistikami na Android developers Dashboards[11].
Rozhodl jsem se aplikaci cílit na Android verze 4.0.3 (API 15), jelikož díky tomu bude aplikace funkční na téměř 98% aktivních zařízení na trhu. Tato skutečnost je důležitá, jelikož uživatelé budou s největší pravděpodobností chtít pro monitoring využít nějaký svůj starší telefon.

Využití MVC

Aplikace bude navržena v souladu s návrhovým vzorem Model, View, Controller[12]. To znamená oddělení logiky aplikace od uživatelského rozhraní. Model aplikace bude realizován ve smyslu podpůrných tříd mimo Aktivity zařízení, jejich názvy a balíku v

tuto chvíli nemohu určit, jelikož vývoj bude probíhat iterativně a funkce se mohou s každou další iterací měnit. V každé iteraci tedy budu zmiňovat nově přidané třídy a jejich popis a strukturu, bude-li se to hodit.

5 Iterace vývoje aplikace Baby monitor

Vývoj aplikace a vlastně celý postup práce na projektu budu dále vést iterativně - každá takové iterace bude konzultována se zadavatelem projektu, a jejím závěrem bude zhodnocení, které může vést i ke změnám chování a určení aplikace, a tedy bude mít přímý vliv na iteraci následující.

5.1 Řešení spojení aplikací

S ohledem na požadavky uvedené v předchozí kapitole jsem vytvořil první prototyp aplikace prozatím schopný jen přenosu zvuku pomocí Wi-fi připojení.

5.1.1 Návrh

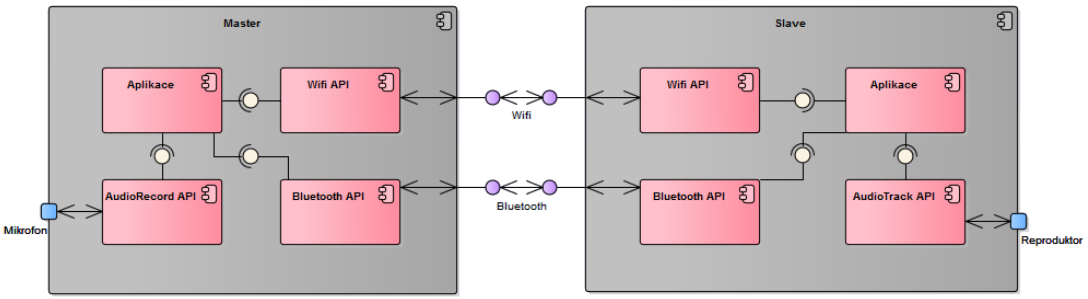
Aplikace bude fungovat ve dvou režimech – Master a Slave. Tedy dětský modul (Master) rozesílá audio stream rodičům (Slave).

Master

Pomocí třídy AudioRecord je vytvořen zásobník, který obsáhne data nahraná mikrofonem. Tato data jsou posléze v nekonečné smyčce čtena vláknem, které je rozesílá na všechna zařízení v kódovaném PCM formátu skrze datagramové pakety UDP[13] směrované na broadcast adresu sítě.

Slave

Zařízení v režimu Slave přebírá pakety. Vlákno postupně dekóduje a přehrává data pomocí třídy AudioTrack.



Obrázek 2 Struktura systému

Výběr komunikační technologie

V dnešní době je již standardem, aby zařízení s Androidem obsahovala dvě nejrozšířenější technologie použitelné pro místní komunikaci - Wi-fi a Bluetooth. Pro implementaci komunikace mezi zařízeními jsem se rozhodl využít pouze první zmíněnou technologii, a sice Wi-fi.

Proč ne Bluetooth?

Osobně neznám jediné zařízení s Androidem, které by podporovalo Bluetooth, ale ne Wi-fi, což znamená, že je zbytečné používat dva různé paralelní technologie. Navíc, dle následující tabulky, Standard bluetooth má mnohem kratší dosah oproti Wi-fi a i datová kapacita je nižší. Jediné, čím Bluetooth vyniká, je nenáročnost spotřeby energie[14] - to ale bohužel nestačí.

	Wi-Fi	Bluetooth
Typický dosah	35-95m	5-30m
Počet zařízení	až 253	až 8
Šířka pásma	800kbps	11Mbps

Tabulka 3 Srovnání technických parametrů technologií[15]

5.1.2 Implementace

UDP Broadcasting

Data jsou třídou AudioRecord nahrávána ve formátu PCM, s rychlostí vzorkování 8000Hz a velikostí vzorku 16 bitů (. Toto nastavení jsem zvolil, jelikož je naprosto dostačující k věrnému přenosu většiny člověkem slyšitelného pásma, především tedy hlasu[16]. Navíc to umožňuje přímý přenos těchto dat pomocí UDP packetů fixní velikosti (320 bytů, což je vzhledem k použitému nastavení přibližně 50ms záznamu) a naprostou nenáročnost na výpočetní výkon zařízení. Jelikož není nijak kontrolován úspěšný přenos dat z jednoho zařízení na druhé a kvůli standardu UDP ani konstantní prodleva mezi jednotlivými pakety, rozhodl jsem se zavést zásobník před třídou AudioTrack, o celkové velikosti 8000 bytů, tedy 1,25s, aby nedocházelo k vypadávání přehrávání, což má za následek čekání na znovu-naplnění vnitřního zásobníku použitého API (a tedy velmi nepříjemný výpadek).
Pakety samotné jsou Master zařízením odesílány na broadcast adresu lokální sítě. To umožní všem Slave zařízením odposlech streamu bez nutnosti zadávání adresy. Jediné co je k tomuto řešení nutné, je nastavení stejného portu na obou zařízeních. Prozatím byl použit fixní port 22222, v budoucnu budu muset vyřešit situaci, ve které by takový port byl již obsazen.

Inicializace rekordéru

Nahrávání zvuku probíhá v instanci objektu AudioRecord. Ve smyčce je volána metoda HandleBuffer, která zpracuje nahrané vzorky.

```
AudioRecord recorder = new AudioRecord(...);
recorder.startRecording();
while (!stopped) {
 int n = recorder.read(buffer, 0, buffer.length);
 Controller.get().handleBuffer(buffer);
}
```

Metoda HandleBuffer

Metoda HandleBuffer zpracovává vzorky v závislosti na vnitřním módu aplikace.

```
public void handleBuffer(byte[] data) {
        if (mode == RECORDER && streamAudio){
            BroadcasterData.sendData(data);
        }
        if (mode == PLAYER && !mute && !serverMute){
            player.playBuffer(data);
        }
        handleSoundSample(data);
    }
```

Metoda sendData

Vzorky jsou nakonec odeslány na broadcast adresu sítě.

```
public static void sendData(byte[] data) {
 if (!Sockets.isDataSocketReady()) {
  return;
 }
 DatagramPacket packet;
 packet = new DatagramPacket(data, data.length);
 packet.setPort(Controller.DataPort);
 packet.setAddress(Controller.get().getIpAddress());
 try {
  Sockets.getDataSocket().send(packet);
 } catch (NullPointerException npe) {
  npe.printStackTrace();
 } catch (IOException e) {
  e.printStackTrace();
  Log.d("Server", "Error: IO Exception");
 }
}
```

Proč ne TCP/IP?

TCP protokol by umožňoval unicastový[17] přenos dat mezi dvěma uzly, Master a Slave zařízením. Zajistila by se tím vyšší kvalita spojení pro daná zařízení, ale za cenu zvýšení zátěže sítě, jelikož každé zařízení by mohlo v jeden čas požadovat jiné pakety. Z toho důvodu považuji za dostatečné, když bude aplikace podporovat pouze přenos pomocí UDP tak, jak je popsán výše.

Vytvoření lokální sítě

Systém v současném stavu umožňuje komunikaci mezi zařízeními v rámci jakékoliv Wi-fi sítě, ke které jsou všechna zařízení připojena. Pokud by nebyl k dispozici router, uživatel může využít funkci tzv. Hotspotu, kdy na jednom zařízení spustí sdílení připojení a

druhé k němu připojí. Interface pro přechod do nastavení Hotspotu zatím implementováno nebylo ale bude implementováno v některé z příštích iterací, pravděpodobně jako odkaz do nastavení v postranním menu.

Operační systém

Prototyp jsem z důvodu usnadnění testování prozatím cílil na Android API 9, jelikož jsem v době implementace neměl k dispozici druhé zařízení s Androidem 4.0.3+. Toto bude v budoucnu změněno.

GUI

GUI prototypu je tvořeno dvěma aktivitami, každá reprezentuje jeden modul aplikace (dětský/rodičovský). Přepínání modulu je možné pomocí tzv. postranního menu, které je v Androidích aplikacích obvyklé - menu se vytahuje z levé části obrazovky a je automaticky uzavřeno po přepnutí aktivity.

5.1.3 Nasazení a testování

Prototyp byl nasazen a testován na následujících zařízeních:

	OS	Oficiální software	Verze Software
Sony Xperia Z3	Android 5.1	Ano	Sony 23.4.A.1.232
HTC Desire Z	Android 2.3	Ne	CyanogenMod cm-7.2.0-vision-stable

Tabulka 4 Technické parametry testovacích zařízení 1

Výsledek testování

Aplikace úspěšně přenáší zvuk mezi zařízeními. Spojení jsem testoval i v oblastech s větším množstvím aktivních sítí v pásmu 2,4GHz - ani to nepředstavovalo problém.

5.1.4 Zhodnocení práce

Na konci první iterace stojí prototyp aplikace, který implementuje přenos zvuku po síti Wi-fi. Konkrétně byl implementován požadavek RQ 2.1 a funkční prototyp byl prezentován panu Komárkovi dne 11. 11. 2015. Test funkčnosti přenosu zvuku byl úspěšný a nic tedy nebrání na tomto konceptu postavit zbývající funkcionalitu aplikace.

5.2 Rozšíření funkcionality a přidání informací o spojení

V této iteraci proběhla implementace identifikace zařízení.

5.2.1 Návrh

Aplikace v současné podobě nepřenáší žádné informace o stavu Master zařízení (dále již budu místo Master a Slave používat termíny dětský a rodičovský pro větší přehlednost). Navrhnul jsem tedy způsob, který periodicky jednou za vteřinu odesílá na broadcast adresu informační paket obsahující stav baterie.

5.2.2 Implementace

Přechod na android 4.0.3

Projekt aplikace jsem migroval na min. API 15, jelikož již mám k dispozici druhý přístroj na testování.

Způsob distribuce informací

do aplikace jsem přidal další vlákno, které každých 1000ms získá informaci o stavu baterie a tu odešle ve formě datagramového packetu na Broadcast adresu sítě, ale jiný port, než který je používán ke streamování zvuku (port 22223).
Způsob získání informace o baterii:

```
Intent batteryChanged = registerReceiver(null,
new IntentFilter(Intent.ACTION_BATTERY_CHANGED));
int level = batteryChanged.getIntExtra(
    BatteryManager.EXTRA_LEVEL, -1
);
int scale = batteryChanged.getIntExtra(
    BatteryManager.EXTRA_SCALE, -1
);
percent = level / (float) scale;
```

5.2.3 Nasazení a testování

Aplikaci jsem nasadil a testoval na následujících zařízeních

	OS	Oficiální software	Verze Software
Sony Xperia Z3	Android 5.1	Ano	Sony 23.4.A.1.232
HTC Desire Z	Android 4.2	Ne	CyanogenMod 10.1-20140302

Tabulka 5 Technické parametry testovacích zařízení 2

Na zařízení HTC Desire Z, které v původní verzi obsahuje pouze Android 2.3.7, jsem musel pro účely testování nahrát neoficiální software s OS ve verzi 4.2. Výkon zařízení je i nadále pro testování aplikace dostačující.

Samotné testování probíhalo za pomoci debugovacích a (především) logovacích nástrojů vývojářského prostředí Android Studio 1.1[18].

5.2.4 Zhodnocení práce

Prototyp byl předveden učiteli 2.12.2015 na konzultaci. Shodli jsme se, že příští iterace bude zaměřena opět na zdokonalování funkcionality, tentokrát již ale včetně uživatelského rozhraní a stability adekvátních k případnému vystavení na Google Play store. Aplikace by také měla obsahovat ujednání pro koncového uživatele, které bude zobrazeno jednou, při prvním spuštění aplikace.

5.3 Dokončení implementace funkcí a příprava na publikaci

5.3.1 Návrh

Proběhne vylepšení funkcí aplikace, především design a přidání informací pro uživatele a také implementaci alarmu. Jedná se stále o aplikaci pro Android 4.0.3, rád bych ale i přesto tvořil aplikaci vzhledově vyhovující současnému standardu, jímž je Material design[19]. Ten je ale bohužel v plném rozsahu podporován až v Androidu 5.0+[20], čímž se dostávám k následujícímu:

API 15 a komplikace s ním spojené

Možností, jak docílit zobrazení ovládacích prvků aplikace v Material designu, je nastavení cílového a minimálního API v konfiguračním souboru pro Gradle a přidání závislostí pro materiálové postranní menu (appcompat) a seznam s kartami (cardview a recyclerview). Následuje ukázka takového souboru:

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 21
    buildToolsVersion "21.1.2"
    defaultConfig {
        applicationId "rec.babymonitor"
        minSdkVersion 14
        targetSdkVersion 21
        versionCode 1
        versionName "1.0"
    }
}
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:appcompat-v7:21.0.3'
    compile 'com.android.support:cardview-v7:21.0.+'
    compile 'com.android.support:recyclerview-v7:21.0.+'
}
```

5.3.2 Implementace

Grafické úpravy

Ucelené barevné schéma

Kombinaci tmavě a světle modré jsem vybral jako hlavní barevné schéma, opakující se v celém uživatelském rozhraní aplikace. Jednotlivé barevné kódy jsou převzaty ze stránky **https://www.google.com/design/spec/style/color.html#color-color-palette**. Barevné kódy jsem umístil do souboru /res/colors.xml, ze kterého je pak možné jednotlivé barvy referovat pomocí metody **Resources.getColor(id)**. Ukázka souboru:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <!-- Default Material style -->
    <color name="primary">#2196F3</color>
    <color name="primary_dark">#90CAF9</color>

    <!-- Custom styling -->
    <color name="colorAccent">#B2FF59</color>
    <color name="behindCards">#e4e4e4</color>
    <color name="warning">#FFDDDD</color>
</resources>
Blue    #2196F3
Blue    #90CAF9
```

Další barvy jsem zatím nepoužil, barevné schéma bude dokončeno v příští iteraci.

Ikona

Ikonu aplikace jsem vytvořil pomocí grafického editoru GIMP[21]. Jedná se o subjektivní a abstraktní vyobrazení "dudlíku", tedy věci tématicky velmi relevantní.

Obrázek 3 Ikona aplikace a také její logo

Přepsání menu aplikace

Menu teď místo obyčejného seznamu využívá výše zmíněný recyclerView, včetně ikony znázorňující odkazovanou funkci aplikace. V horní části menu je také hlavička, kterou jsem se snažil vytvořit v souladu s Material designem - Kruhová ikona aplikace a název na pozadí tvořeným sekundární barvou aplikace (#90CAF9).

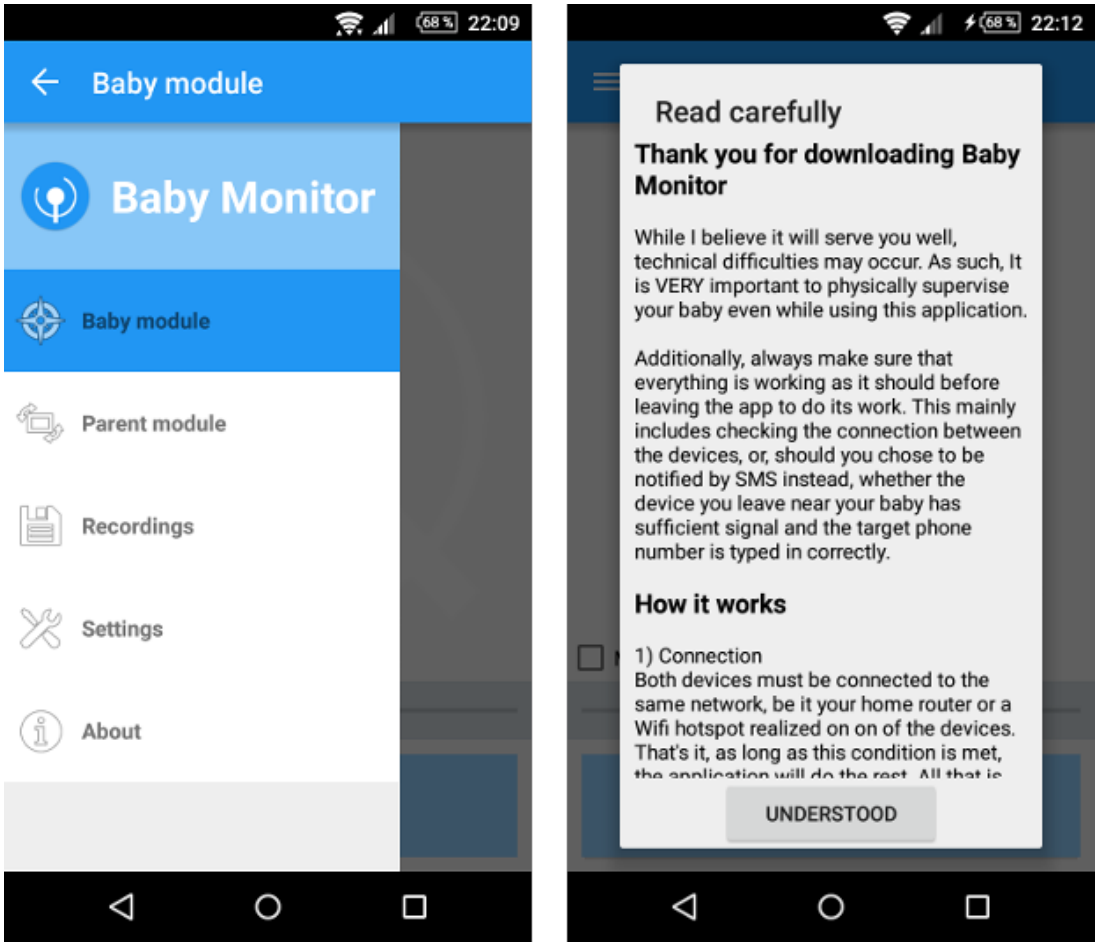
Moduly Baby a Parent

Kompletně jsem přepracoval tyto moduly, včetně úpravy ovládání, realizovaného již pouze pomocí jediného ovládacího tlačítka, za účelem co nejvyšší jednoduchosti pro uživatele. Spojení probíhá automaticky, nutné zásahy jsou jen minimální.

5.3.3 Funkční úpravy

Zajištění funkce při uspání zařízení

Systém při vypnutí zařízení přechází do tzv. režimu spánku, kdy u některých zařízení dochází k zastavení rozesílání UDP packetů[22], či dokonce k uspání chodu celé aplikace.



Obrázek 4 Prozatímní vzhled aplikace. Na obrázku vlevo je hlavní menu, na obrázku vpravo zase ujednání s koncovým uživatelem a stručný návod

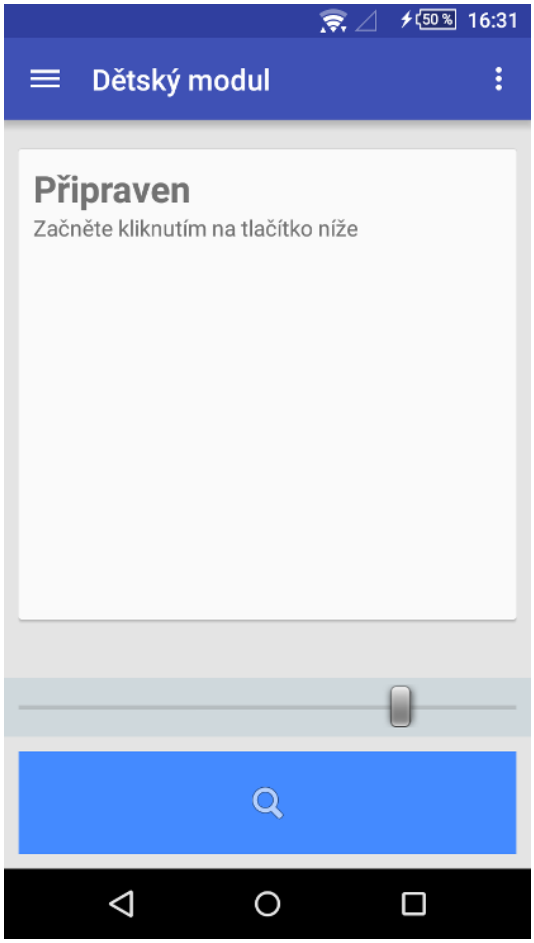
Z toho důvodu byla implementována žádost o přidělení tzv. částečného wakelocku, který zajistí, aby zařízení do tohoto režimu spánku nepřešlo.
O wakelock je žádáno pouze při zahájení aktivního přenosu a je uvolněn vždy po jeho ukončení, aby nedocházelo ke zbytečnému plýtvání energie.

Automatické obnovení spojení

Aplikace umožňuje opětovné připojení rodičovského modulu k dětskému, pokud dojde ke ztrátě spojen. Toto je realizováno pomocí neustálého monitoringu daného portu, na který rodičovský modul vysílá. V první iteraci jsem zmínil, že se jedná o port 22222 a aplikace vysílá na broadcast adresu sítě.

Nastavení limitu hluku

Bylo implementováno nastavení maximální hlasitosti nahrávaného hluku pomocí posuvníku. Tato hodnota (integer 1-100 reprezentující hlasitost v dB) je uložena jako třídní proměnná Controlleru a je dále používána pro porovnání s hlasitostí získanou z mikrofonu



Obrázek 5 Posuvník nastavení hlasitosti uprostřed obrazovky

Získání hlasitosti

Vlákno, které se stará o přebírání dat ze třídy AudioTrack, nyní zároveň volá novou metodu handleSoundSample(), která čte první dva byty dodaného vzorku následujícím způsobem:

```
public void handleSoundSample(byte[] buffer) {
 short int16 = (short) (((buffer[1] & 0xFF) << 8) |
 (buffer[0] & 0xFF));
 int absoluteValue;
 if ((absoluteValue = Math.abs(int16)) < 1) {
  absoluteValue = 1;
 }
 int soundPressure = (int) (20 * Math.log10(absoluteValue));

  ...

}
```

Dva první byty jsou nutné proto, že z mikrofonu je ukládán vždy 16 bitový vzorek. Výsledná proměnná soundPressure je referenční hlasitost v decibelech, spočtená vzorcem

přebraným z Wikipedie na adrese en.wikipedia.org/wiki/Decibel.
Metoda dále obsahuje další jednoduchou funkcionalitu, která porovnává hlasitost s tou nastavenou uživatelem pomocí posuvníku - pokud dojde k překročení, je rozezněn alarm.

Alarm

V případě zjištění nízkého stavu baterie (méně než 20%), přerušení datového toku na déle než 2 vteřiny či překročení stanoveného limitu hluku se ozve alarm. Zvukovou nahrávku jsem získal na adrese http://www.soundjay.com/bell-sound-effect.html ve formátu mp3. Ten lze v Androidu jednoduše přehrát pomocí API třídy MediaPlayer.

Pořízení nahrávky

Probíhá tak, že vlákno určené pro čtení dat ze třídy AudioRecord zároveň s odesláním paketu se vzorkem také zapíše vzorek do souboru. Pokud uživatel povolí záznam, je soubor vždy vytvořen se spuštěním přenosové relace a finalizován po ukončení této relace (nebo při řízeném pádu aplikace). Práce se soubory je v Dalviku totožná s prací ve standardní Javě, jsou k tomu tedy použity třídy FileOutputStream a File.
Při prvním pokusu o vytvoření nahrávky je vytvořena adresářová struktura v paměti zařízení, do které je ukládáno až 10 posledních záznamů délky max. 60 minut, aby nedošlo k zahlcení paměti zařízení, pokud by uživatel aplikaci zapomněl zapnutou nebo prostě zapomněl vypnout funkci nahrávání. 60 minut mi osobně přišlo jako optimální hodnota vzhledem k zaměření aplikace, své rozhodnutí tedy stavím na tom.
Záznam, jak jsem již několikrát zmínil v předešlých kapitolách, je nahráván ve formátu PCM, což vlastně není nic jiného než označení nekódovaných vzorků napětí nahraných z mikrofonu jdoucích v sekvenci za sebou[23].

Notifikace pomocí SMS

Byla implementována i funkce zasílání SMS v případě překročení nastavené hranice hluku. Pokud uživatel v nastavení aplikací zvolí tuto funkci, je na vložené telefonní číslo zaslána SMS s přednastaveným sdělením a také přesným časovým údajem pro případ, že by se SMS opozdila. Odesílání SMS programaticky je v Androidu vlastně velmi jednoduché - stačil k tomu následující kód:

```
private void sendSMS() {
 if (!sent) {
  sent = true;
  String message = "Something␣may␣be␣happening␣near␣your␣baby!" +
  "␣The␣noise␣rose␣above␣given␣threshold␣at␣" +
  (new SimpleDateFormat("HH:mm")).format((new Date()).getTime());
  SmsManager smsManager = SmsManager.getDefault();
  smsManager.sendTextMessage(number, null, message, null, null);
 }
}
```

Zobrazení informací uživateli při prvním spuštění

Při prvním spuštění aplikace jsou uživateli prezentovány informace o používání aplikace. Znovu je možné je zobrazit pomocí "About"v hlavním menu. Celé znění mimo

jiné obsahuje zřeknutí se zodpovědnosti za škody způsobené aplikací, jelikož ta bude distribuována zdarma, "ve stavu, v jakém, je". Jedná se o standardní klauzuli, kterou obsahuje většina aplikací na Google Play [24]. Dále výsledný dialog obsahuje jednoduchý návod užití a popis všech vlastností aplikace.

5.3.4 Nasazení a testování

Funkčnost aplikace byla opět testována na dvojici zařízení uvedených v sekci 5.2.3

5.3.5 Zhodnocení práce a nutné změny

Na konci této iterace tedy stojí aplikace, která má nyní již uživatelsky mnohem přívětivější rozhraní, v souladu se způsobem ovládání na cílové platformě obvyklým. Po zvážení koncového stavu s panem Ing. Komárkem jsme ale došli k závěru, že je stále ještě nutné na některých vlastnostech aplikace pracovat. V následujících odstavcích shrnu oblasti vyžadující přepracování.

Grafické rozhraní

Aplikace zatím ještě stále není graficky přívětivá. Použité ikony nesymbolizují funkce k nim přiřazené a to se týká především hlavního ovládacího tlačítka, které za všech okolností používá ikonu lupy. Koncový uživatel neví, že aplikace v obou módech (Master i Slave) vlastně vyhledává svůj protějšek - ikona je tedy zavádějící.

Navigace aplikací

Jednotlivé moduly (Dětský, Rodičovský, Nahrávky, Nastavení) jsou v současném stavu jednotlivé aktivity které musí být znovu inicializovány pokaždé, když uživatel přejde z jedné do druhé. Toto je provázeno zbytečným čekáním při načítání a také probliknutím bílé obrazovky. Zároveň je také kvůli tomu nefunkční tlačítko zpět, jelikož aplikace má ne-jednoznačnou hierarchii. Z toho důvodu bych rád aplikace přepsal tak, aby využívala fragmenty - jak toho docílím popíši podrobněji v budoucí kapitole tomu věnované.

Odesílání SMS

Jediný způsob, jak nyní zvolit číslo pro odeslání varovné SMS, je ručně ho vypsat do kolonky v Nastavení aplikace. Toto je evidentně uživatelsky nepřívětivé a v další iteraci určitě implementuji výběr z kontaktů v telefonu. Dale je také zbytečné, aby se funkčnost vůbec zobrazovala na telefonech, které nemají telefonní modul, nebo jen přímo nepodporují odesílání SMS, jako například některé tablety s 3G/4G modulem, ale bez GSM.

Souběžný chod několika Master a Slave zařízení v jedné síti

Původně jsem bohužel nevzal v potaz situaci, při které by mohly být zařízení připojena k současné infrastruktuře, v podobě externího routeru. Za těchto podmínek by nebylo možné provozování další nezávislé instance Master a Slave zařízení, jelikož současné pakety neobsahují žádný identifikátor - pouze se odesílají na předem připravené porty. Provozování dvou instancí by nejen nebylo možné, pokus o realizaci dalších připojení by také narušil chod instance původní. V následující iteraci budu muset přepsat způsob, jak spolu jednotlivé aplikace komunikují - novou specifikaci také nově podrobně popíšu, jelikož s největší pravděpodobností bude finální.

Zabezpečení spojení pomocí hesla

Jak jsem zmínil výše, provozování aplikací na veřejné síti bude možné. To s sebou nese riziko odposlouchávání, v případě zapomenutí Master zařízení v nahrávacím módu. V příští iteraci tedy implementuji možnost připojení Slave zařízení pouze za předpokladu dodání nějakého autorizačního tokenu (s největší pravděpodobností to bude krátký PIN).

Generování IP adresy

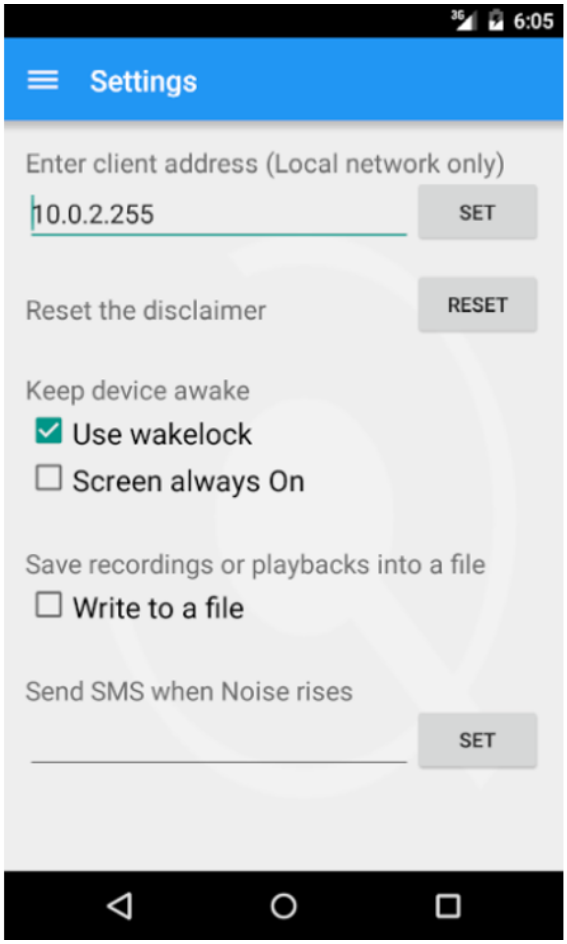
Najití správné Broadcast adresy WLAN adaptéru zařízení se provádí pouze při startu aplikace. Pokud uživatel například odpojí a znovu připojí WLAN, nebo se zkrátka připojí až po startu aplikace, aplikace nadále nefunguje. Je nutná změna tohoto chování.

Jazyk

Všechny řetězce používané v uživatelském rozhraní natvrdo přímo v kódu, jelikož jsem nepředpokládal použití jiného jazyka než angličtiny. Popis převodu na dynamický způsob určování řetězců bude také součástí příští iterace.

Nastavení aplikace

Interface pro nastavení aplikace je nedostačující - grafická podoba je "chaotická", nevyhovující zavedeným způsobům platformy [25]). Chybí například jasné oddělení kategorií a popis voleb.



Obrázek 6 Problémy vyčtené výše jsou na obrázku patrné na první pohled

5.4 Finální prototyp dětské chůvičky

V této kapitole popíši poslední změny aplikace před zakončením vývoje. Bylo vytvořeno kompletně nové uživatelské rozhraní a vylepšen způsob komunikace mezi zařízeními. Zároveň jsem navrhl a implementoval nové funkce.

5.4.1 Návrh funkcí

K dosavadnímu návrhu Dětské chůvičky přidávám následující změny. Číslování funkčních a nefunkčních požadavků je kompatibilní s tím v kapitole 4, dále pravidla pro určení složitosti jsou opět schodná.

Aktualizace funkčních požadavků

RQ 1.3 - Informace o spojení

Zadavatel Martin Komárek

Systém bude uživateli poskytovat relevantní informace o spojení a stavu k zajištění použitelnosti aplikace jako dětské chůvičky.

RQ 7 - Možnost souběžnosti více instancí monitorování

Zadavatel Martin Komárek

Aplikace by měla umožňovat souběžnou funkci více Master či Slave zařízení z důvodů popsaných v konkrétních požadavcích RQ 7.1 a RQ 7.2.

RQ 7.1 - Více Master zařízení vysílajících naráz

Zadavatel Martin Komárek
Priorita Vysoká
Složitost Vysoká

Vnitřní implementace aplikace bude umožňovat dvěma a více Master modulům vysílat naráz v rámci jedné sítě a uživateli umožňovat výběr požadovaného modulu, protože aplikace umožňuje provoz i pomocí externího Wi-fi routeru ke kterému jsou zařízení připojena a proto musí být situace více souběžných instancí brána v potaz.

RQ 7.2 - Více Slave zařízení připojených k jednomu Master zařízení

Zadavatel Pavel Zářecký
Priorita Nízká
Složitost Vysoká

Aplikace bude umožňovat připojení více Slave zařízení k jednomu Master zařízení, jelikož to použité technologie umožňují a jedná se, dle mého názoru, o užitečnou funkci.

RQ 8 - Možnost zabezpečení vysílání pomocí hesla (PINu)

Zadavatel Martin Komárek

Systém bude umožňovat zabezpečení vysílání pomocí čtyřmístného pinu za účelem redukce možnosti nechtěného odposlechu v rámci Wi-Fi sítě.

Aktualizace nefunkčních požadavků

RQ 6.3 - Material Design

Zadavatel Pavel Zářecký
Priorita Střední
Složitost Střední

Design aplikace bude nyní vyhovovat směrnicím Material Designu do nejvyšší míry kterou jsem schopen implementovat, jelikož Material Design je od verze Androidu 5.0 obecně preferovaným designem [19]

RQ 6.4 - Animace

Přechody mezi obrazovkami a také některé akce uživatele budou doprovázeny animacemi, jelikož animace jsou nedílnou součástí Material Designu

Zadavatel	Pavel Zářecký
Priorita	Střední
Složitost	Nízká

5.4.2 Implementace

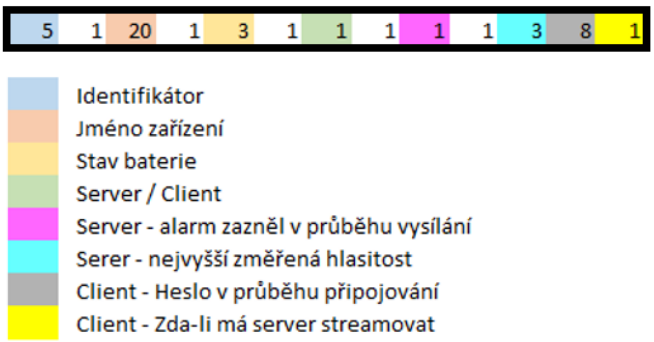
V poslední iteraci jsou implementovány všechny funkční a nefunkční požadavky aplikace, s výjimkou RQ 2.3, který z časových důvodů implementovat nebudu.

Formát odesílaného rámce a identifikace zařízení

Master i Slave zařízení používají k řídící komunikaci stejný formát rámce UDP paketu, jelikož se komunikované informace liší jen minimálně. Rámec obsahuje informace jako je jednoznačné ID zařízení a jméno, které je získáno následujícím způsobem:

```
id.append(android.os.Build.MANUFACTURER);
id.append(" ");
id.append(android.os.Build.PRODUCT);
```

Nevýhoda tohoto řešení je ta, že přímo závisí na tom, zda výrobce zařízení použil nějaký smysluplný název zařízení. Telefony značky Sony se hlásí uživatelsky přívětivě (např. Sony Z3 Compact), telefony značky Samsung ale bohužel používají vnitřní kód výrobce (jako např. samsung "trtlex"místo "Note 4"). Alternativou by bylo užití identifikátoru Bluetooth, přístup k němu by ale bohužel vyžadoval povolení, a protože by aplikace jinak Bluetooth vůbec nevyužívala, mohlo by to v uživateli vzbuzovat nedůvěru.
Zbytek informací obsažených v rámci již obsahuje informace, které aplikace používá především ke komunikaci svého stavu s ostatními aplikacemi v okolí, například Master aplikace rozesílá své pakety na Broadcast adresu sítě v konstantních intervalech, což umožňuje její nalezení Slave aplikací. Ta, v případě podnětu uživatele zpět odpovídá na adresu Masteru, nejdříve s obsaženým heslem, pokud ho aplikace požaduje - po úspěšném připojení se již heslo nevysílá, pokud je spojení aktivní. Na následujícím obrázku již kompletní schéma rámce:

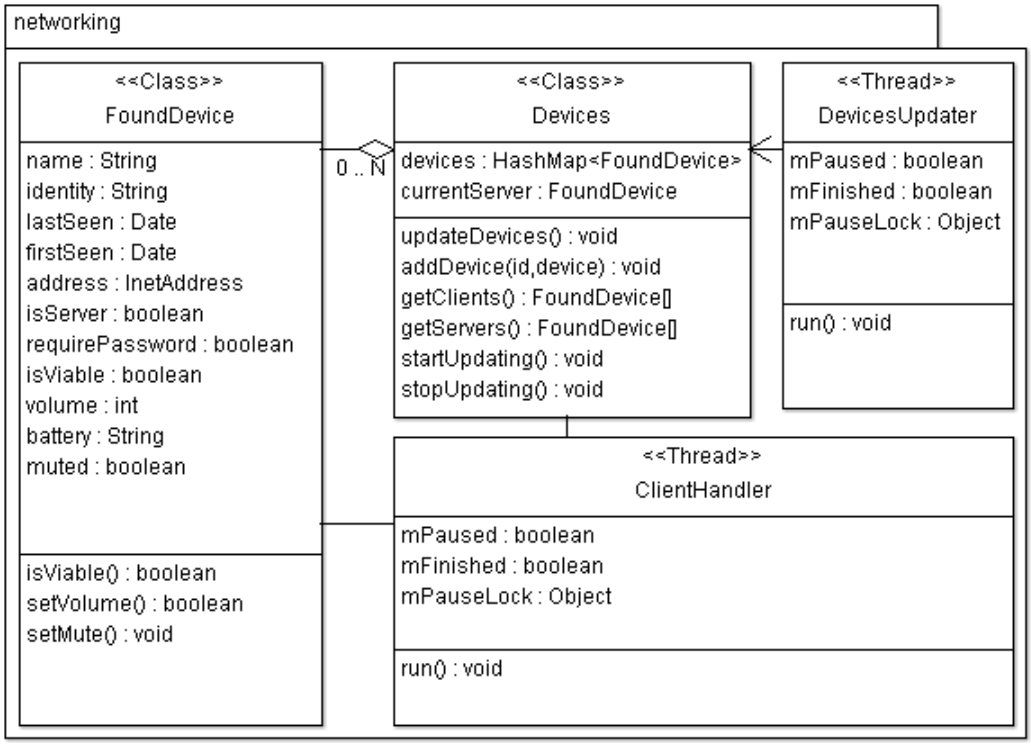


Obrázek 7 Schéma formátu rámce

Zpráva připojených zařízení Master i Slave

O vyhledávání se stará vlákno ClientHandler, které parsuje pakety příchozí na řídící socket. Parsované zařízení je vnitřně reprezentováno třídou FoundDevice, která je ulo-

žena v "kontejnerové"třídě Devices, již zmíněné v útržku kódu v subsekci o komunikaci zařízení výše. Pokud je již zařízení v kontejneru obsaženo (řídí se dle kombinace ID názvu), pouze se obnoví relevantní informace, jako čas poslední komunikace, stav baterie, a podobně.
O udržování aktuálnosti kontejneru se zařízeními se stará další pomocné vlákno - DeviceUpdater. Periodicky prochází uložená zařízení a hledá ta, která za posledních několik vteřin vůbec nekomunikovala - taková jsou posléze odstraněna z kontejneru úplně.



Obrázek 8 Třídní diagram funkcionality správy nalezených zařízení

Změna funkce metody sendData() třídy Broadcaster

Metoda pro odesílání dat nyní nemusí tato data směrovat na Broadcast adresu sítě, protože dětský modul má po implementaci správy připojených zařízení k dispozici jejich IP adresy. Díky tomu již nadále aplikace nebude zbytečně zatěžovat síť rozesíláním datových paketů i na zařízení, která je zahazují.

```
public static void sendData(byte[] data) {
    if (!Sockets.isDataSocketReady()) {
        return;
    }
    DatagramPacket packet;
    packet = new DatagramPacket(data, data.length);
    packet.setPort(Controller.DataPort);
    ClientsContainer devices = Devices.getClients();
    if (devices.count < 1) return;
    for (int i = 0; i < devices.count; i++) {
        packet.setAddress(devices.devices[i].address);
        try {
            Sockets.getDataSocket().send(packet);
        } catch (NullPointerException npe) {
            npe.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
            Log.d("Server", "Error: IO Exception");
        }
    }
}
```

Zabezpečení heslem

Výše jsem zmínil zabezpečení spojení. To je volitelně zajištěno čtyřmístným pinem, který je Slave zařízením předáván Master zařízení vždy, když zrovna neprobíhá aktivní přenos zvukových dat.

5.4.3 Perzistence v rámci aplikace

Aplikace samozřejmě ukládá svůj stav tak, aby byl k dispozici při dalším spuštění. Jedná se především o pořízené nahrávky a uživatelské nastavení. Způsob ukládání těchto dvou věcí se liší.

Ukládání a invokace nastavení

K perzistenci uživatelských nastavení a proměnných aplikaci využívám Androidí **Preference Manager**. Ukládání a invokace probíhá následovně:

```
SharedPreferences sharedPref;
sharedPref = PreferenceManager.
 getDefaultSharedPreferences(this);
SharedPreferences.Editor editor = sharedPref.edit();
editor.putBoolean("showDisclaimer", false);
editor.apply();

...

sharedPref.getBoolean("showDisclaimer", true)
```

Nové rozhraní nastavení

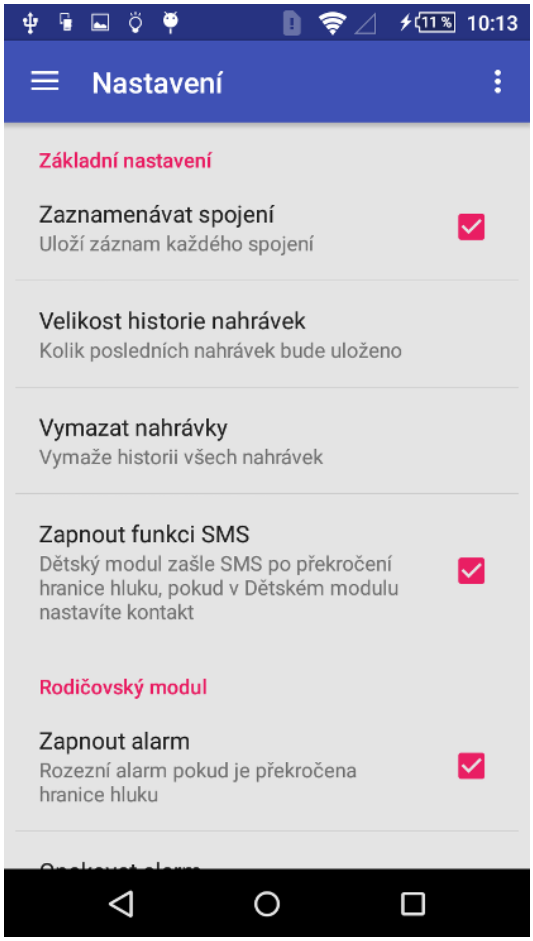
Rozhraní nastavení jsem kompletně změnil. Nyní používám pro vykreslení UI speciální XML s **PreferenceScreen**, které vypadá následovně:

```
<PreferenceScreen
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/preference_screen">
    <CheckBoxPreference
            android:key="write"
            android:title="@string/settings_saverec_title"
            android:summary="@string/settings_saverec_sum"
            android:defaultValue="false"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />

    ...

</PreferenceScreen>
```

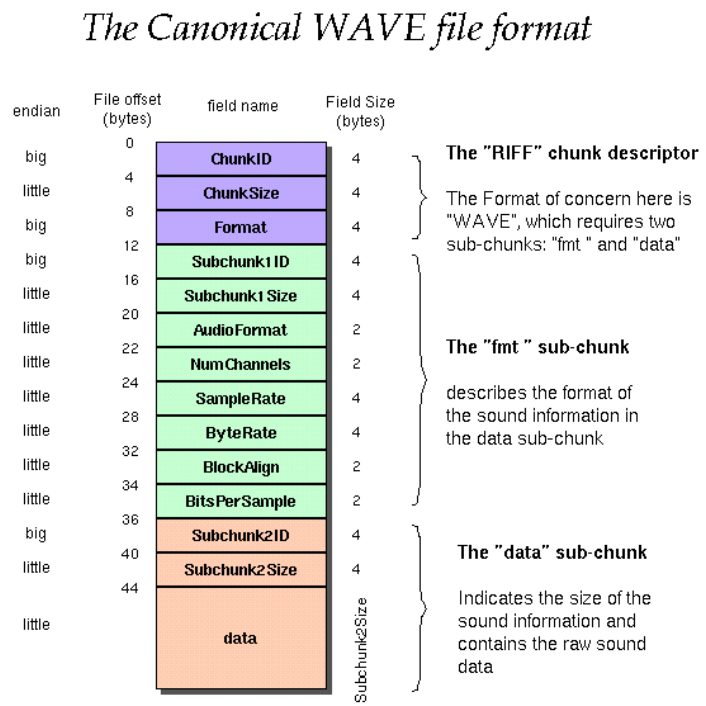
Jednotlivá nastavení jsou v souboru definována jako widgety <...Preference/> a atribut android:key určuje klíč, pod jakým jsou v PreferenceManager dohledatelné. Samotná grafická podoba je nyní již čistě v režii systému Android:



Obrázek 9 Nové nastavení

Formáty WAV a PCM a ukládání

Nahrávky jsou ukládány do vygenerované složky v kořenovém adresáři zařízení. Každý záznam je ve formátu PCM 16bit, 8kHz a uživatel má možnost takovouto nahrávku vyexportovat do formátu WAV, který informace o vzorkování také obsahuje - díky tomu je možné nahrávky dále přehrávat a používat jinde, než jen v aplikaci dětské chůvičky.



Obrázek 10 Formát hlavičky WAV souboru. PCM nahrávka je v sekci data. Obrázek převzat z http://soundfile.sapp.org/doc/WaveFormat/

5.4.4 Design UI

V poslední iteraci aplikace prošla největšími grafickými změnami, jelikož jsem konečně měl jistotu, že skladba funkcí se dále nebude měnit.

Přechod na Fragmenty

V původních iteracích aplikace používala pro každou obrazovku vlastní aktivitu, jak jsem již psal dříve. **Aktivita** je ucelená, samostatně stojící část aplikace, která může (ale nemusí) mít přiřazené uživatelské rozhraní, které se vždy musí při volání metody **onCreate** "nafukovat"- **inflate**, včetně úplně všech součástí, jako je horní panel, menu aplikace **Drawer** a všech grafických prvků. To se negativně podepsalo na mnoha místech, především na uživatelské přívětivosti a času nutného pro přechod mezi aktivitami. Přepsal jsem tedy základ aplikace tak, aby používala pouze jednu jedinou aktivitu a jednotlivé obrazovky byly realizovány pomocí fragmentů uživatelského rozhraní. Díky tomu se výrazně zrychlila navigace aplikací a byly odstraněny všechny problémy s tlačítkem zpět, což popíši dále.

Životní cyklus fragmentu a aktivity

Každý fragment dědí životní cyklus aktivity, ve které běží. Díky tomu nebylo nutné příliš měnit jednotlivou funkčnost kódu v aktivitách, pouze se převedla do metod jednotlivých fragmentů, které jsou pouze volány (nyní již jedinou) nadřazenou aktivitou **Main** Cyklus života aktivity v mé aplikaci vypadá následovně (viz obrázek v kapitole 2):

onCreate()

Vytvoření aktivity Main, postranního menu **Drawer**, inicializace Controlleru a také výchozího fragmentu - Dětského modulu

onStop()

Pokud aplikace zrovna neběží (není aktivní žádné spojení a uživatel ani aktivně nevyhledává zařízení), pozastaví se všechna pomocná vlákna, aby aplikace zbytečně nemrhala časem procesoru, když je na pozadí

onStart()

Opětovné probuzení všech vláken

onDestroy()

Pouze defaultní chování - jelikož v aplikaci nepoužívám NDK a tedy C++, není nutné uvolňovat prostředky - o to se postará Garbage collector v Javě

Animování přechodů

Využívám dva způsoby animace elementů:

objectAnimator

Zde stačí definovat animaci pomocí XML - dráha animace, velikost, která vlastnost se bude animovat - to vše je nutné definovat následujícím způsobem:

```
<set
    xmlns:android="http://schemas.android.com/apk/res/android">
    <objectAnimator
        android:valueFrom="-100dp" android:valueTo="0dp"
        android:valueType="floatType"
        android:propertyName="translationX"
        android:duration="@android:integer/
␣␣␣␣␣␣␣␣config_mediumAnimTime" />

    <objectAnimator
        android:valueFrom="0.0" android:valueTo="1.0"
        android:valueType="floatType"
        android:propertyName="alpha"
        android:duration="@android:integer/
␣␣␣␣␣␣␣␣config_mediumAnimTime" />
</set>
```

Animátor pak stačí volat pomocí setCustomAnimations(), například při přechodu fragmentů

propertyAnimator

Velmi jednoduchý způsob animace, stačí volat na "property"(objekt widgetu) takto:

```
fab.animate().scaleX(1);
fab.animate().scaleY(1);
```

Součástí je také AnimationListener, kterým kontroluji, zda-li již animace proběhla celá - stane-li se tak, tlačítko opět reaktivuji pomocnou proměnnou:

```
fab.setListener(new Animator.AnimatorListener() {
    @Override
    public void onAnimationEnd(Animator animator) {
        animationDone = true;
    }
});
```

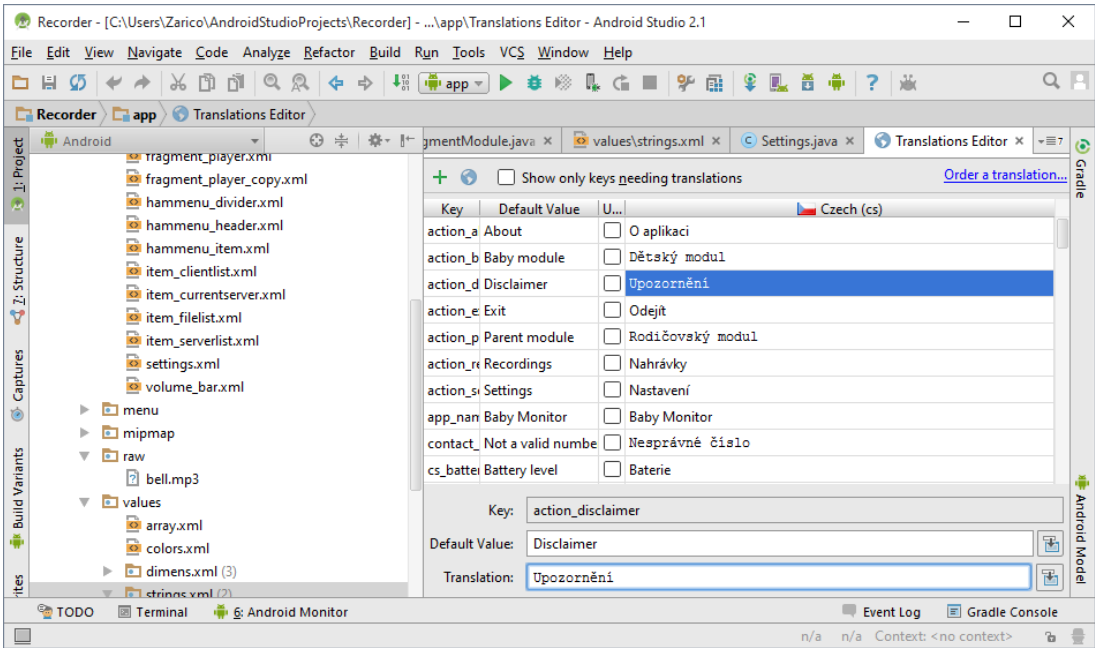
Chování tlačítka zpět

Při stisku tlačítka zpět vždy dojde ke změně fragmentu na ten, ve kterém byl uživatel předtím - aplikace si jej pamatuje pomocí proměnné **prevPosition**.

```
public void onBackPressed() {
    if (Controller.prevPosition == 0) {
        resolveFragmentChange(Controller.prevPosition);
        ft.replace(
            getActivity.getResources().getIdentifier(
            "content_frame", "id",
            getActivity.getPackageName()),
            new FragmentModule()
        );
        interfaceRecorderDefault();
    }
}
```

Jazyky

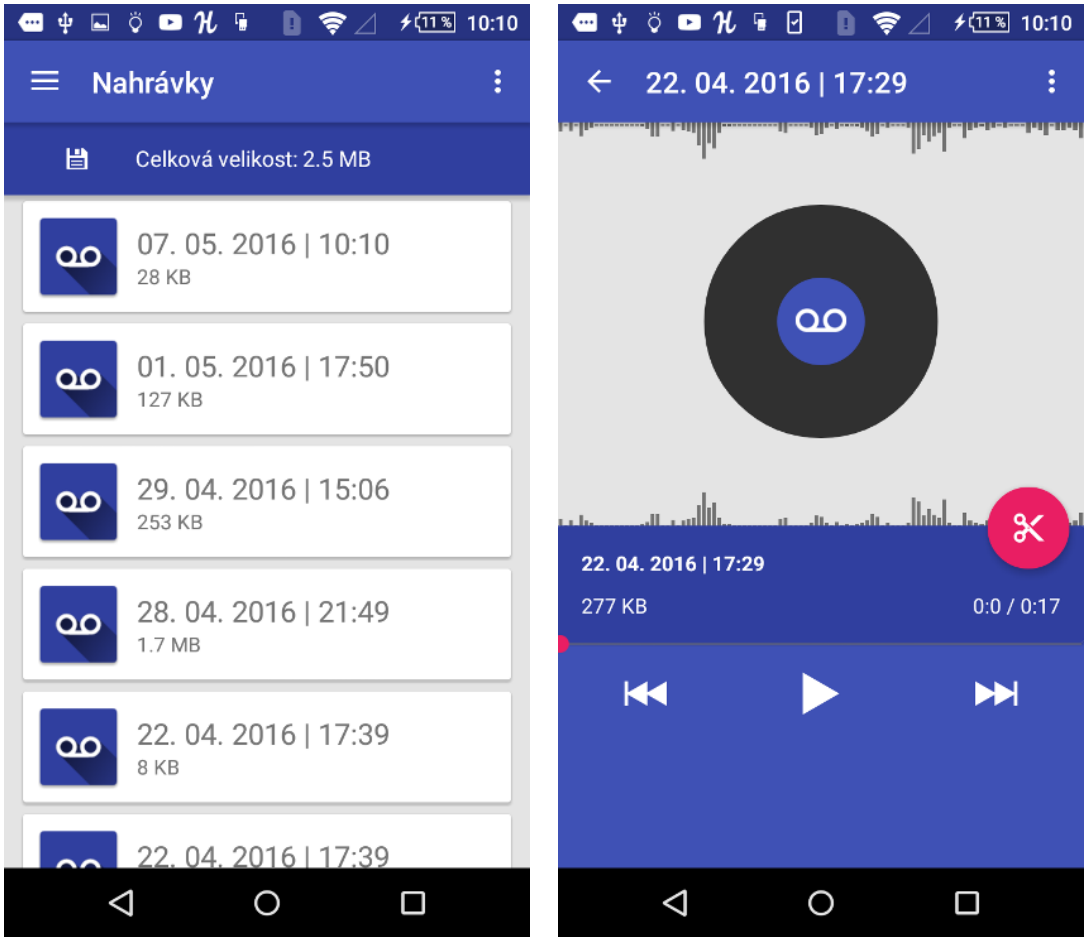
Aplikace nyní podporuje automatické přepínání jazyků dle prostředí systému - o to se stará operační systém sám, díky zabudované podpoře načítání řetězců z **Resource** XML souborů jednotlivých jazyků. Editace jsem prováděl následovně, pomocí interface v Android studiu:



Obrázek 11 Editace řetězců v prostředí Android Studio

Přehrávání nahrávek a význam přehrávače

Pořízené nahrávky je samozřejmě možné přehrávat a spravovat uvnitř aplikace, pomocí integrovaného přehrávače - přehrávač zároveň dokáže analyzovat nahrávku a zobrazit uživateli hlasitostní diagram, díky kterému je možné jednoduše přejít na tu část nahrávky, která je podstatná. Aplikaci lze díky tomu použít také jako jednoduchý spánkový monitor.



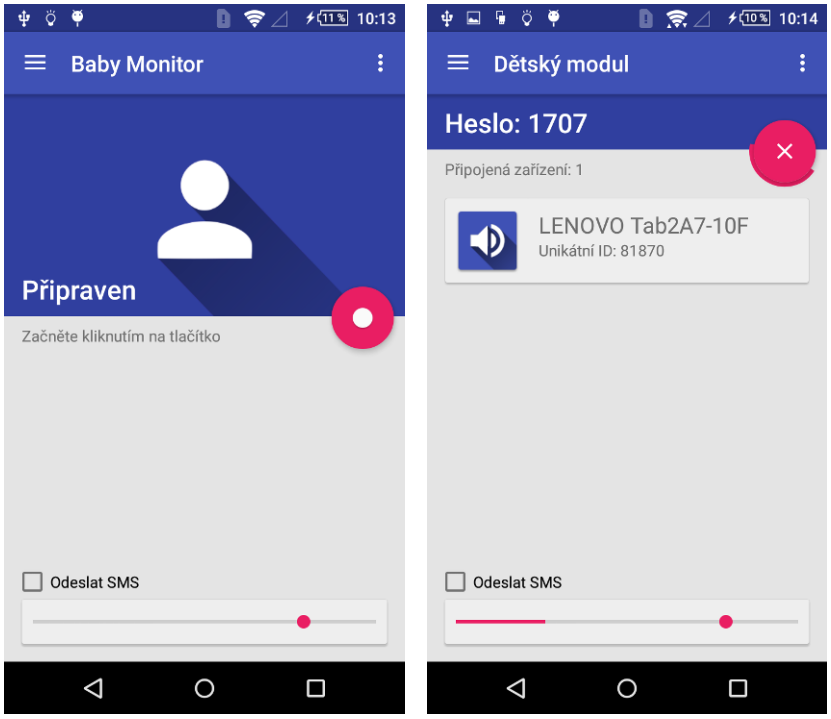
Obrázek 12 Správa nahrávek - přehrávač obsahuje diagram, ze kterého si uživatel může hned vybrat požadovaný segment

Generování diagramu

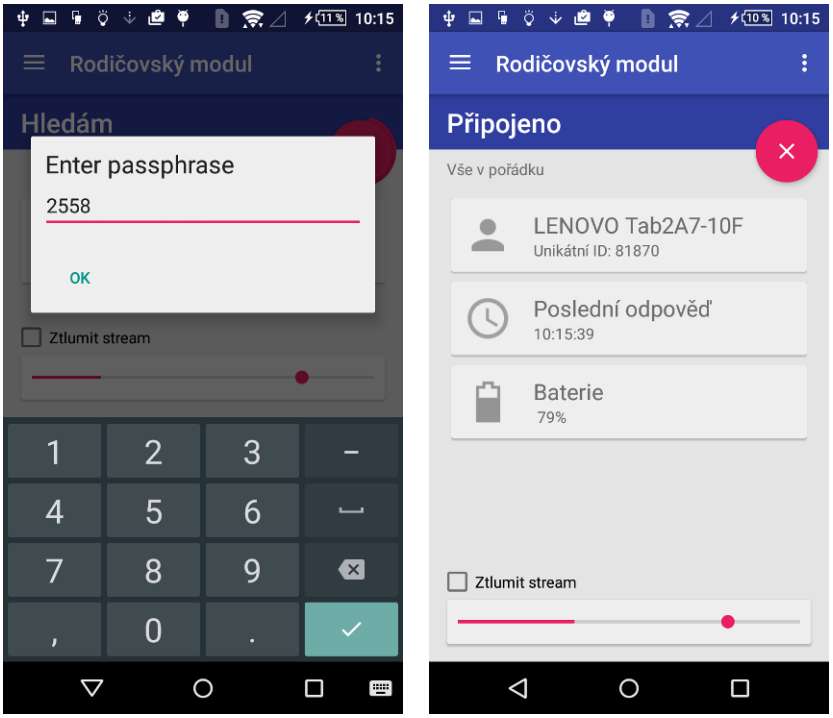
Soubor s nahrávkou je načten do paměti a je rozdělen na 100 stejně velkých části. V každé části jsou postupně procházeny vzorky a z každého 320-tého byte (320 bytů je velikost nahrávaného bufferu) je spočtena hlasitost způsobem, jak jsem jej popsal v předešlé iteraci. Výsledné hlasitosti jsou zprůměrovány a ze sta takovýchto průměru je sestaven diagram, který lze vidět na obrázku výše.

Ostatní grafické prvky

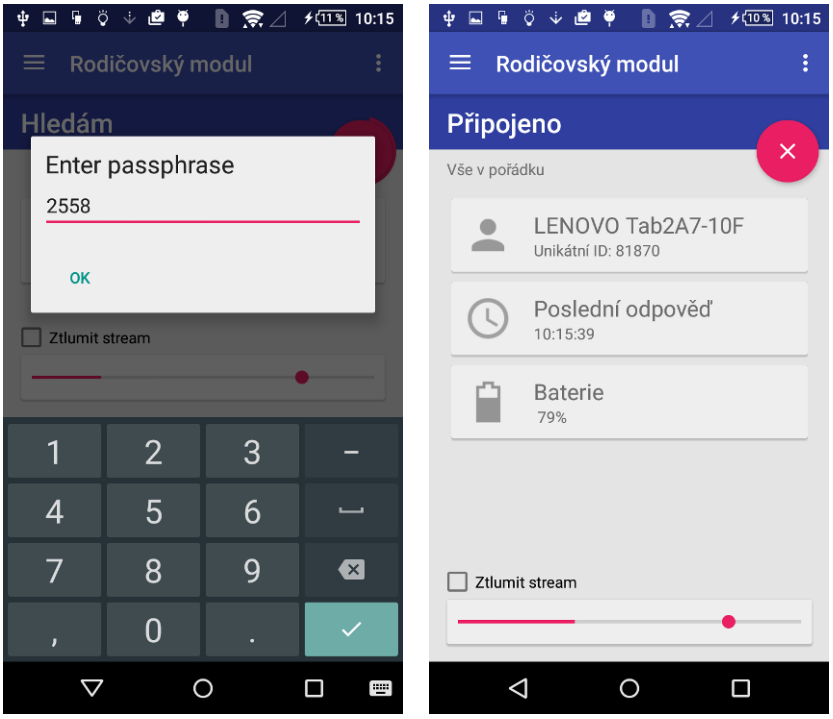
Na závěr bych rád představil několik obrázků aplikace a popsal funkci obrazovek na nich, jelikož zobrazují fragmenty, u nichž jsem pouze měnil grafickou reprezentaci - funkcionalita je již popsána na předchozích řádcích.



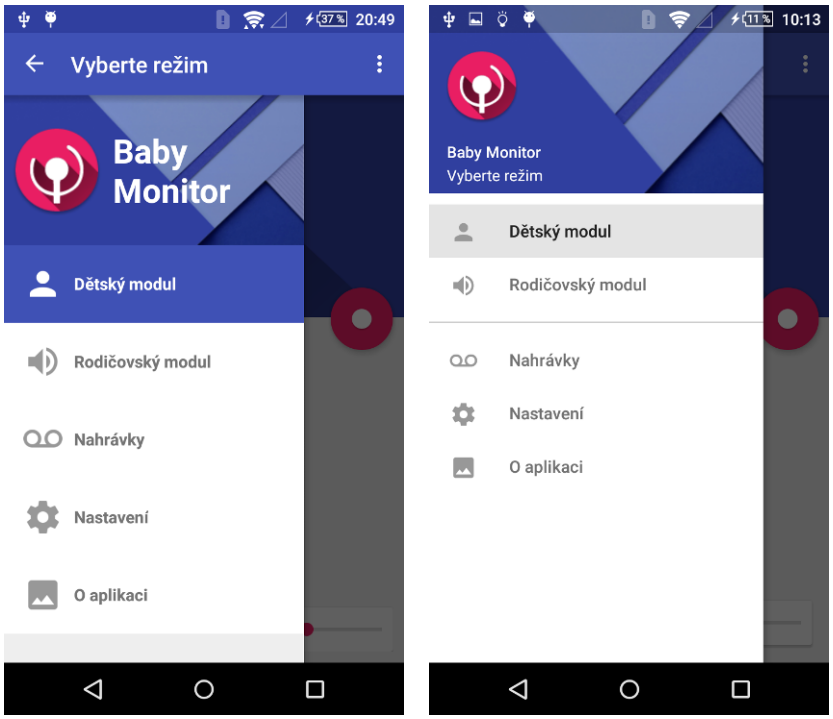
Obrázek 13 Prostředí dětského modulu



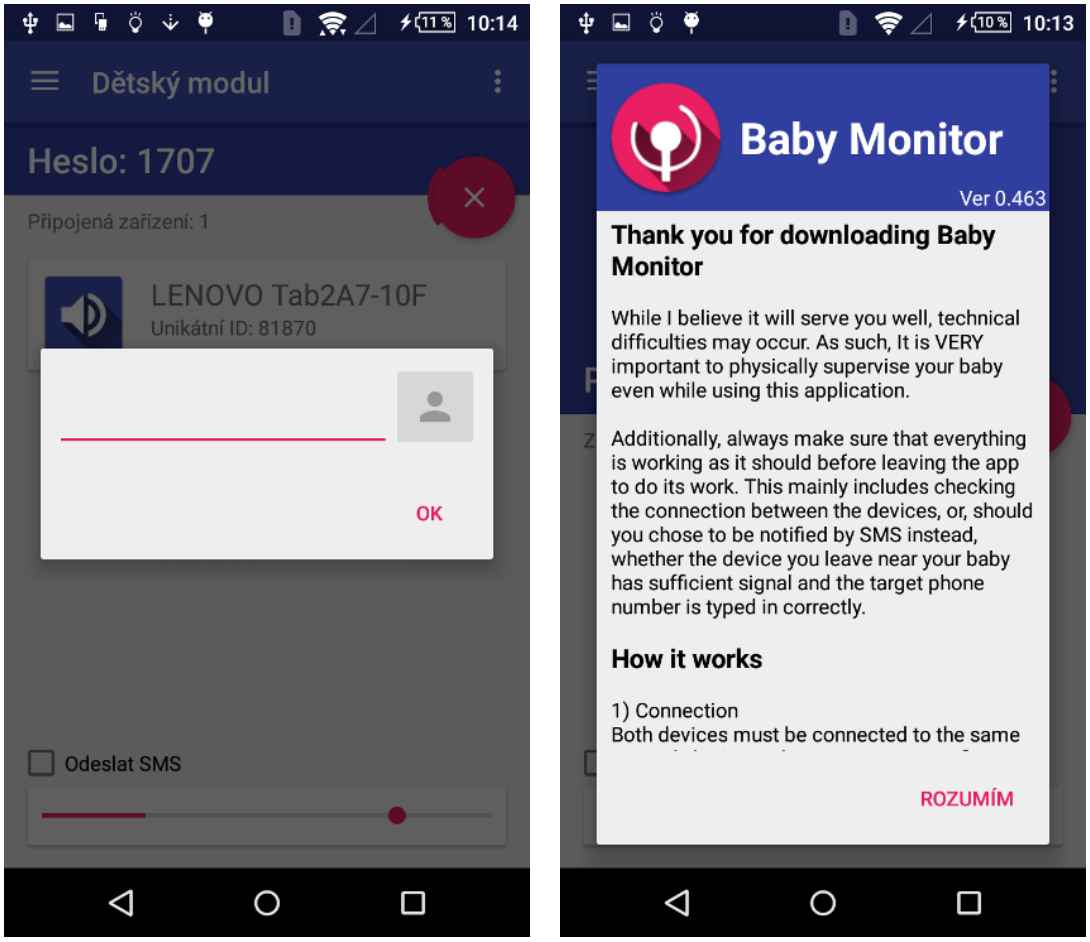
Obrázek 14 Prostředí rodičovského modulu



Obrázek 15 Prostředí rodičovského modulu



Obrázek 16 Změna nabídky Drawer do podoby, která vyhovuje nejnovějším standardům Material Design



Obrázek 17 Výběr kontaktu a také přepracovaný Disclaimer

5.4.5 Finální testování

Statické testování

Po implementaci všech funkcí jsem aplikaci podrobil tzv. statickým testům, které analyzují bytecode a hledají chyby v kódu. K testování jsem užil nástroje dostupné pro IDE Android studio - FindBugs a PMD.

PMD

Nástroj PMD provádí jednoduchou analýzu kódu - vyhledává např. nepoužité proměnné, prázdné catch bloky, zbytečně vytvářené objekty, nesprávné užití if klauzulí, apod.[26]

FindBugs

Provádi hloubkovou analýzu, především hledá chyby způsobené špatnými praktikami programátora, chyby ovlivňující výkon aplikace a také zranitelnosti v kódu.[27]

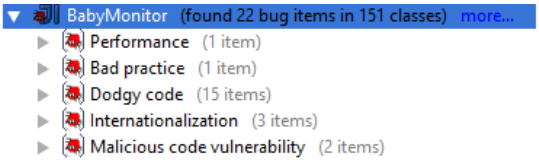
Výsledek testování

PMD

Nástroj PMD nalezl několik "chyb"týkajících se především kvality kódu. U spousty z nich jsem shledal případnou opravu nepotřebnou, nicméně pomocí nástroje jsem opravil nedostatky jako užití operátoru + místo StringBuilderu, prázdné catch klauzule, zanechané nepoužívané proměnné, if bez závorek {} a podobně.

Findbugs

V nástroji jsem nastavil pouze hledání chyb, u kterých má nastroj jistotu, že se o chyby opravdu jedná. Při nastavení medium totiž nástroj reportoval i části kódu, nad kterými jsem buď neměl kontrolu, jako např. knihovny nalinkované Gradlem, nebo se dle mého názoru vůbec o chyby nejednalo. Analýza přesto nalezla několik chyb kódu, které jsem opravil.



Obrázek 18 Výsledek analýzy před opravami problémů

Dodgy code

Manipulace statické proměnné instancí

V několika případech nástroj zjistil, že instance objektů zapisují do statických třídních proměnných. To by mohl být problém, pokud by v aplikaci existovalo více instancí jednoho objektu a ty vy si takto navzájem přepisovaly proměnné. V mé aplikaci ale všechny třídy, až na jedinou výjimku v podobě FoundDevice, existují vždy pouze v jediném objektu. Vzájemné přepisování tedy nehrozí a rozhodl jsem se tento nález prozatím neopravovat.

Nepoužívané proměnné

Nástroj mi dále byl užitečný v eliminaci nepoužívaných, leč deklarovaných proměnných, které jsem při ruční kontrole přehlédl.

Neporovnávání hashe

Ve třídě FoundDevice jsem implementoval vlastní metodu equals(), u které nástroj zjistil, že neporovnává hash. V tomto případě ale porovnávání hashe není na místě, jelikož nejde o porovnávání totožnosti objektů, ale jen o kontrolu několika proměnných, jako například jména zařízení a id. Kontrolu hashe jsem tedy neimplementoval.

Performance

Zde nástroj zjistil, že by bylo rychlejší při porovnávání čísel použít metodu Long.compare(l1,l2) místo užití Long.valueOf(l1).compareTo(l2). Bohužel tohoto jsem nemohl využít, protože užití první metody by vyžadovalo novější Android API. Dále bylo opět nalezeno několik proměnných do kterých je size zapisováno, ale nikdy z nich nečteno, což má negativní vliv na výkon.

Testování v reálném provozu

Funkčnost aplikace byla opět testována na stejných zařízeních jako v kapitole 5.2.3 a pomocí metodiky popsané tamtéž.

6 Aplikace CashBob - mobilní klient

Druhým bodem zadání mé práce je převzetí aplikace CashBob pro Android a provedení testování vůči nové revizi rozhraní REST[5] realizovaném na serverové části aplikace a především implementace tisku lokálně i na serverové tiskárně.
CashBob je více modulový restaurační pokladní systém psaný v jazyce Java, vyvíjený již několik let v rámci závěrečných prací katedry počítačů. Komunikační model systémů v průběhu let postupně procházel různými transformacemi, zejména přechodem z technologie Java RMI (Remote method invocation) na standard REST. Toho využívá Android modul systému, původně vyvinutý Tomášem Hogenauerem pro jeho závěrečnou práci - tento Androidí modul budu ve své práci dále rozšiřovat.

6.1 Standard REST

V původním odstavci jsem zmínil užití standardu komunikace REST (Representational state transfer). Jedná se o architekturu rozhraní, která dovoluje číst, modifikovat a mazat data na serveru za pomoci jednoduchých HTTP volání. Data jsou reprezentována tzv. zdroji, kde každý zdroj má svou HTTP adresu (URI), pomocí které je ke zdrojům přistupovánorest.

6.1.1 Komunikace

Ke zdrojů, je přistupováno pomocí čtyř základních metod, které koncepčně odpovídají čtyřem základním operacím CRUD nad úložištěm SQL.

REST	CRUD	Popis
POST	Create	Vytvoření nového zdroje
PUT	Update	Úprava existujícího zdroje
GET	Retrieve	Získání zdroje
DELETE	Delete	Smazání zdroje

Tabulka 6 Popis REST metod

6.1.2 Předávaná data

Zdroje samotné jsou skrze HTTP protokol předávány ve tvaru tzv. JSON objektů, které mohou vypadat například takto:

```
{"exampleArray":[
    {"name":"name1", "surrname":"surname1"},
    {"name":"name2", "surrname":"surname2"},
    {"name":"name3", "surrname":"surname3"}
]}
```

Vnitřní implementace objektů v aplikaci se tedy před přenosem musí nejdříve serializovat na tvar uvedený výše, a po přenosu opět parsovat do podoby objektu cílového prostředí (např. tedy Java entita).

6.2 Tisk

Aplikace bude umožňovat tisk účtenek. Účtenky samotné jsou generovány na serveru - aplikace si tisk vyžádá pomocí REST URI. Tvar a následné zpracování odpovědi závisí na tom, zda-li bude vyžádán tisk na serverové tiskárně, nebo lokálně.

6.2.1 Vyžádání tisku

Vyžádání probíhá na následující URI, která byla dohodnuta s Tomášem Červenkou, který v době psaní této práce vyvíjel nový webový server. Jedná se o užití metody POST:

```
http://%server%/rest/account/%account%/print
```

%server%

Adresa serveru CashBobu, musí obsahovat i port 9998

%account%

Jméno účtu, ke kterému je účtenka přiřazena. Jak toto přesně funguje popíší dále, v kapitole věnované analýze kódu.

6.2.2 Tisk na serveru

CashBob server sám spravuje tiskárny, které jsou připojeny k počítači, na kterém běží. Z toho důvodu stačí v požadavku pouze specifikovat, zda-li server vrátí hotovou účtenku, nebo nikoliv.

6.2.3 Tisk lokálně

V případě specifikování lokálního tisku server také zašle hotovou účtenku ve formátu PDF. Ta bude vytištěna tiskárnou STAR SM-T300, připojenou přes Bluetooth. Podrobnosti o implementaci budou dále v kapitole věnované samotné implementaci.

7 Analýza aplikace CashBob

V této kapitole provedu analýzu aplikace dle zadání. Bude se jednat především o kontrolu kvality kódu, kontrolu chování aplikace v rámci konvencí operačního systému Android a nakonec test funkčnosti implementované REST komunikace vůči nové verzi REST rozhraní webové aplikace, v době psaní práce vyvíjené Tomášem Červenkou.

7.1 Ruční analýza aplikace

Aplikace je psána v jazyce Java, bez užití NDK a zdrojový kód byl k dispozici ve formátu projektu pro Android Studio. Jednotlivé třídy jsou umístěny do jediného balíku com.example.tomas.cashbob. Vzhledem k tomu, že se jedná o aplikaci pro virtuální stroj Androidu (Dalvik/ART) a ne pro standardní verzi Javy, budu analýzu již přímo vztahovat ke konvencím cílového operačního systému.

7.1.1 Struktura

Základní struktura aplikace je tvořena dvěma aktivitami - LoginActivity a MainActivity. Co jsou to aktivity a fragmenty jsem již zmínil ve své práci dříve zmínil při popisu vývoje aplikace BabyMonitor a také v kapitole 2.

LoginActivity

Aktivita obsahuje textboxy pro vložení uživatelského jména a hesla. Jejich obsah se nekontroluje, není to třeba - to dělá až server. Dále se v aktivitě provádí ukládání nastavení aplikace - adresa serveru a měna. Tyto informace jsou ukládané do SharedPreferences systému Android, a tedy k dispozici i po ukončení z znovuzavedení aplikace.

MainActivity

Hlavní aktivita aplikace. Obsahuje TabHost, což je prvek uživatelského rozhraní, který dovoluje přepínání fragmentů. Jednotlivé fragmenty, které reprezentují funkce aplikace, obsahují vždy inicializaci svých uživatelských prvků tak, jak je zvykem a zejména také implementace jednotlivých metod REST rozhraní, realizovaných pomocí AsyncTask[28] - což je způsob, jak provádět operace v systému android na pozadí tak, aby neovlivňovaly plynulý chod uživatelského rozhraní.

7.1.2 Závěr ruční analýzy

Kód aplikace se jeví jako vyhovující kvalitativním standardům Androidu. Jednotlivé třídy, metody a balíky dodržují konvenci názvů, komponenty uživatelského rozhraní jsou inicializovány v metodě onCreate(), stav aplikace je ukládán do SharedPreferences a znovu načítán při změně stavu (překlopení displeje, znovu-načtení aplikace).
Čitelnost kódu je na dostatečné úrovni - neměl jsem žádný problém s orientací v něm. Jediné, co by určitě mohlo být zlepšeno, je formátování kódu, které je na některých místech chaotické.
Následuje statická analýza, která může poodhalit nedostatky, kterých jsem si nevšiml.

7.1.3 Statická analýza aplikace a její výsledky

K testování jsem použil stejný nástroj jako dříve u aplikace Baby monitor, a sice FindBugs[27], dostupný v Android studiu. Nalezené problémy:

Internationalization

Závislost na výchozím kódování platformy

Nástroj zjistil, že se v kódu používají konstrukce, jako je tato:

```
String credBase64 =
Base64.encodeToString(
 credentials.getBytes(),
 Base64.DEFAULT
).replace("\n", "");
```

To má za následek závislost takového řetězce na platformě Java a jejím výchozím kódování. Jelikož je ale více než nepravděpodobné, že by REST server někdy v budoucnu používal jiný programovací jazyk, neshledávám toto chování jako problematické, nicméně jsem u metod odesílajících řetězce i tak specifikoval formátování UTF-8.

Dodgy code

Manipulace statické proměnné instancí

V několika případech nástroj zjistil, že instance objektů zapisují do statických třídních proměnných. V kapitole 5 jsem zmínil, že toto by byl problém jedině v případě, že by aplikace používala více instancí takovéto "problémové"třídy. Ani v aplikaci Cashbob tomu ale tak není a proto nález opět neshledávám problematickým.

Nepoužívané proměnné

V kódu bylo objeveno několik nepoužívaných proměnných do kterých pouze ukládáno ale nečteno z nich. Tyto jsem tedy odstranil. Pár z nich ale mělo svůj význam - například vytvoření JSON objektu z odpovědi serveru za účelem zjištění řádné odpovědi.

7.1.4 Testování REST

V původním zadání bylo uvedeno, že mým úkolem je otestovat funkčnost dosavadních REST požadavků vůči novému rozhraní webového serveru. Nové REST rozhraní, dostupné na adrese 147.32.80.149:8080, zatím nepodporuje autorizaci - všechny URI a metody (POST,PUT,...) jsou dostupné bez nutnosti přihlášení. Kvůli tomu jsem přihlašování do aplikace nemohl otestovat a přihlašovací aktivitu jsem dočasně upravil, aby nekontrolovala odpověď serveru a automaticky udělila přístup do funkční části aplikace. Dále jsem také nemohl po implementaci testovat tisk, o tom ale až dále.

Jak jsem testoval

Nejdříve jsem k testování použil starý desktopový server dodaný Tomášem Červenkou. abych zkontroloval funkčnost vůči němu - přikládám jej jako přílohu mé práce. Server stačí rozbalit a pouštět pomocí následujícího příkazu - je také nutná Java 1.7, na jiné verzi server padá.

```
java -jar casbob.jar
```

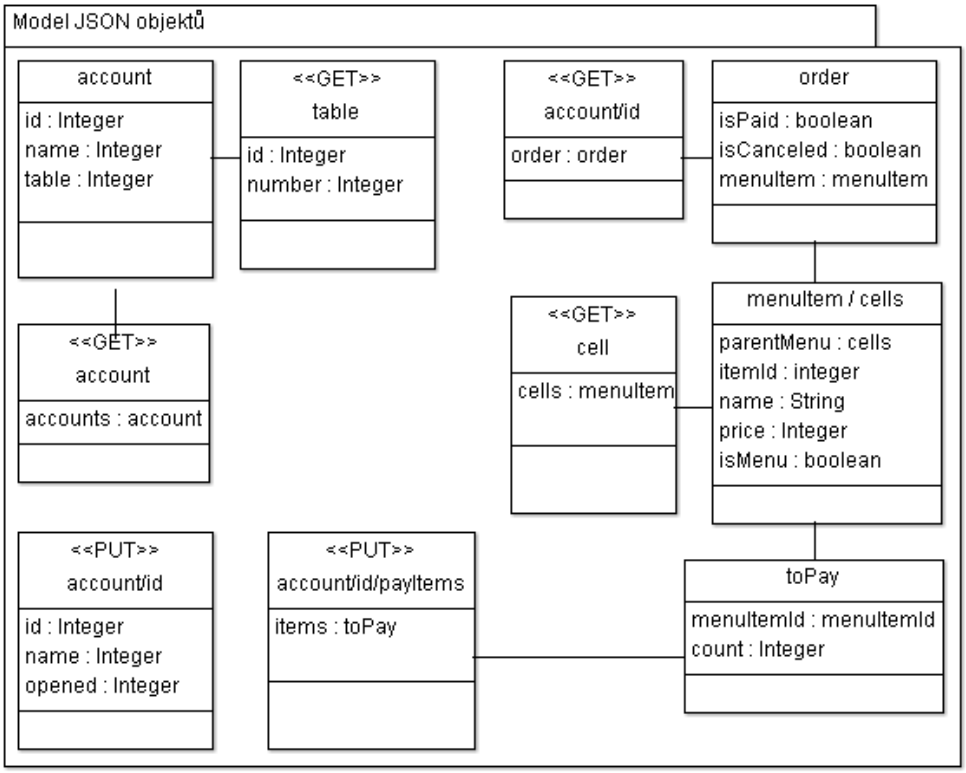
Všechny metody fungovaly tak, jak je v práci Tomáše Hogenauera uvedeno[6] - postoupil jsem k testování vůči novému API a na následujících zaznamenal nalezené nedostatky a popis jejich řešení.

Test API

Dle tvrzení Tomáše Červenky nové API kopíruje strukturu JSON objektů API starého, což jsem osobně z větší části potvrdil testováním pomocí internetového prohlížeče a ručním voláním API skrze adresní řádek. Objevil jsem ale několik nesrovnalostí v názvech proměnných těchto objektů a také ve výchozích hodnotách, kde se funkcionalita mírně lišila. Podrobněji v dalších odstavcích.

Model JSON objektů

Zpracoval jsem velmi zjednodušený diagram, který zachycuje vazby mezi JSON objekty, které aplikace využívá. Jedná se o Class diagram, jelikož REST rozhraní vlastně "simuluje"práci s databází. U objektů, které se volají přímo pomocí jejich URI jsem znázornil, v jaké metodě se používají (POST, PUT, ...).



Obrázek 19 Class diagram JSON objektů

7.1.5 Implementace REST v jednotlivých fragmentech

tabAccounts

Fragment reprezentující funkci výběru účtu. Implementuje tyto rest služby:

URI	JSON objekt
account GET	accs: [opened, id, name]

Tabulka 7 Popis použitých rest metod v tabAccounts

Výsledek testování

Funkcionalita nemohla být otestována, jelikož nové REST API zatím nepodporuje autorizaci a identifikaci v plném rozsahu.

tabNewAccount

Fragment reprezentující funkci vytváření účtu. Implementuje tyto rest služby:

URI	JSON objekt
table GET	tables: [id, tablenumber]
account POST	name,table: id

Tabulka 8 Popis použitých rest metod v tabNewAccount

Výsledek testování a provedené změny

Metoda pro získání stolů funguje bez problémů, Ovšem při volání metody POST account server vrací kód odpovědi 200, který značí, že nedošlo k žádným změnám databáze. To jsem ovšem potvrdil jako nesmysl, protože účet byl na serveru vytvořen - byl dohledatelný s použitím webového prohlížeče. Chybu jsem vystopoval v nesprávném volání metody - místo POST by se správně měla volat metoda PUT. Zkusil jsem tedy upravit AsyncTask tak, aby místo httppost využil httpput. Po volání takovéto upravené metody ale server začal vracet chybu a účet nebyl vytvořen vůbec. Toto chování dává smysl, jelikož návrh API opravdu počítal v tomto případě s voláním POST a ne PUT a proto jsem jako jediné řešení v této situaci viděl přepsání kontroly odpovědi serveru tak, aby kód 200 také uznala za správný.

tabOrder

URI	JSON objekt
account/id GET	order: [ispaid, iscanceled, menuItem: [menuitemid, name, price]]
cell GET	cells: [parentMenu: menuId, name, itemId, name, price, isMenu]
account/id/order POST	items: [manuItemId, count]

Tabulka 9 Popis použitých rest metod v tabOrder

Výsledek testování a provedené změny

Zde byl úplně stejný problém u vytváření objednávky - při získávání odpovědi ze serveru jako kód bylo vráceno 200 místo 201, proto jsem situaci řešil identicky. Dále se zde objevilo několik problémů při parsování dat odpovědi na volání GET cell. V původní verzi pokud menu cell nemělo nadřazenou položku, nebyl parametr parentmenu vůbec do JSON objektu vkládán. V nové verzi ale parametr v objektu existuje vždy a může mít hodnotu null - toto zabránilo správné funkci generování hierarchie menu a aplikace jednoduše spadla. Musel jsem proto ošetření null do metody přidat. Po opravě se objevil problém číslo dvě - původní atribut "menuId"je v nové verzi API přejmenován na "menuid", bez velkého písmena. Řešení bylo triviální - pouze jsem atribut v kódu aplikace přejmenoval.

tabPay

URI	JSON objekt
account/id GET	order: [ispaid, iscanceled, menuItem: [menuitemid, name, price]]
account/id PUT	[id, name, opened]
account/id/payItems PUT	items: [manuItemId, count]

Tabulka 10 Popis použitých rest metod v tabPay

Výsledek testování a provedené změny

Komunikace s novým API bez chyby.

tabTransfer

URI	JSON objekt
account/id/moveItems PUT	targetAccId, items: [manuItemId, count]

Tabulka 11 Popis použitých rest metod v tabTransfer

Výsledek testování a provedené změny

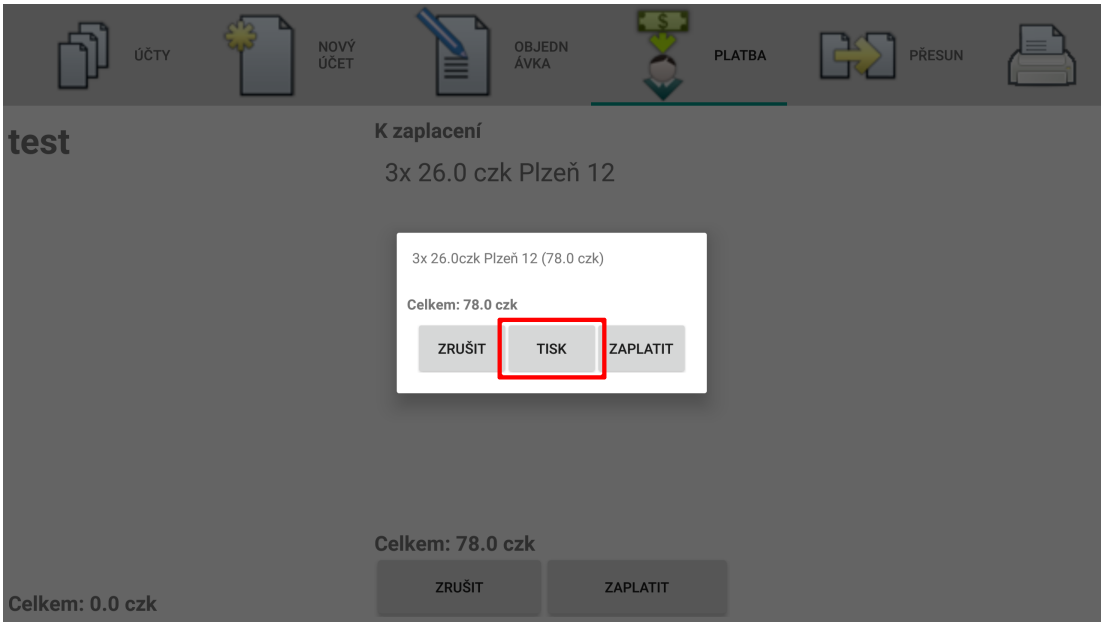
Komunikace s novým API bez chyby.

8 Návrh nových funkcí aplikace CashBob

8.1 Tisk

8.1.1 Úprava uživatelského rozhraní

Tisk by měl být k dispozici ve formě pouhého stisknutí tlačítka ve fragmentu tabPay, pravděpodobně přímo ve formuláři pro finalizaci platby, takto:



Obrázek 20 Přidání tlačítka pro tisk účtenky

8.1.2 Způsob zahájení tisku

Tisk by měl být pevně spjat s uzavřením objednávky a proto bude zahájen automaticky po stisknutí tlačítka "zaplatit". Pokud by z nějakého důvodu nebyl tisk možný (tiskárna nedostupná, není papír,...), zobrazilo by se varování uživateli, ve kterém by mohla být volba pro pro pokračování platby bez tisku.

8.1.3 REST požadavek

URI požadavku jsem specifikoval v předchozí kapitole, samotný JSON objekt by měl obsahovat příznak rozlišující tisk lokálně a na serveru a také seznam všech položek objednávky spolu s počtem kusů. S Tomášem Červenkou jsme se dohodli na následujícím tvaru objektu:

```
{
    "printOnServer": false,
    "orders": [{
        "name": "pizza",
        "price": "59",
        "quantity": "5"
    }, {
        "name": "pizza",
        "price": "59",
        "quantity": "5"
    },....]
}
```

8.1.4 Komunikace s tiskárnou

Tiskárna Star SM-T300 používá ke komunikaci se zařízením Bluetooth. Kvůli omezením daného standardu je nejdříve nutné spárovat zařízení, veškerá další komunikace probíhá prostřednictvím Star SDK pro Android, které při implementaci použiji. Další podrobnosti uvedu v následující kapitole, poté co dané SDK prostuduji.

8.2 Nastavení tisku

Pro nastavení tiskárny a otestování spojení navrhuji použít vlastní záložku a začlenit ji do již existujícího tabHostu v hlavní aktivitě aplikace. Zde by bylo vybrat síťovou, či lokální tiskárnu a zkontrolovat stav (papír, fyzická připravenost,...).

9 Implementace v aplikaci CashBob

9.1 Star SDK

Star Micronics nabízí SDK pro své tiskárny ve formě kompleetní aplikace, ve které jsou implementovány všechny funkce pro daný typ a model. Dostupné zde:

http://www.starmicronics.com/support/sdkdocumentation.aspx

Z tohoto celku jsem musel vybrat ty části, které jsou vhodné pro implementaci tisku účtenky ve formě PDF (nebo spíše bitmapy, o čemž dále v odstavci o zpracování PDF).

9.1.1 Použité třídy

Třída	Důvod výběru
MiniPrinterFunctions	Přístup k funkcím přenosné tiskárny - často se odkazuje na funkce ze třídy PrinterFunctions
PrinterFunctions	Základní přístup k STAR Api
RasterDocument	Implementace raster dokumentu. Možnost užití pro manuální sestavení účtenky, nicméně v současné době využit není.
StarBitmap	Implementace bitmap dokumentu. PDF do něj musí být převedeno před tiskem

Tabulka 12 Vybrané funkce z balíku se Star SDK

Nakonec bylo třeba do aplikace přiložit i přeloženou knihovnu StarIOPort3.1.jar, na kterou se dodané třídy odkazují. Tu jsem vložil do složky /CashBob/app/libs a dále upravil Gradle skript přídáním následujícího kódu:

```
dependencies {
    ...
    compile files('libs/StarIOPort3.1.jar')
    ...
}
\subsectio
```

9.2 Implementace REST požadavku

Všechny REST požadavky aplikace doposud zpracovávala v samostatných instancích AsyncTask třídy, a proto mi přišlo samozřejmé v tomto pokračovat. Zjednodušený kód a popis částí:

```
class RestApiPostPrint extends AsyncTask<Void, Void> {
        **variables
        boolean notProblem = true;
        @Override
        protected void onPreExecute() {
        **zobrazeni dialogu "please wait"
        }
        @Override
        protected Void doInBackground(Void... param) {
            JSONObject dato = new JSONObject();
            try {
            **vlozeni polozek uctenky do dato objektu
            } catch (JSONException e) {
                e.printStackTrace();
            }
            **dokonceni tvorby REST pozadavku a odeslani
            return null;
        }
        @Override
        protected void onPostExecute(Void aVoid) {
        **parsovani odpovedi, Pokud vse probehne v poradku,
        **vytiskne se uctenka a pokracuje placeni.
        **Pokud ne, uzivatel ma moznost pokracovat ci akci
        **ukoncit
        }
    }
```

9.3 Implementace tisku bitmapy

9.4 Zpracování PDF

Bohužel, dodané SDK vůbec nepodporuje tisk PDF, je k dispozici pouze možnost tisku bitmapy. A vzhledem k tomu, že ani Android neobsahuje žádnou funkcionalitu pro práci s PDF až do verze API 21(citace), bylo nutné použít externí knihovnu pro převod. Zvolil jsem pro tuto funkci Apache PDFBox, konkrétně jeho Androidí port dostupný zde:

https://github.com/TomRoush/PdfBox-Android.

Jelikož se jedná o knihovnu s licencí Apache-Commons, nic by nebránilo v budoucí komerční distribuci aplikace (citace). Knihovna je dostupná pouze ve formě zdrojového kódu projektu pro Android Studio. Musel jsem ji tedy přeložit sám - výsledkem je knihovna ve formátu aar (Android archive). Archív posléze stačilo uložit do stejné složky jako knihovnu pro tisk - tedy /CashBob/app/libs a zapsat ji do Gradle skriptu.

9.5 Implementace tisku bitmapy

Bitmapou se rozumí třída android.graphics.Bitmap. SDK nabízí metodu třídu pro předzpracování takové bitmapy do dokumentu pro tisk - StarBitmap.
Dále je nutné otevřít port pro tisk a dodat správné nastavení - V případě použité tiskárny se defaultně jedná o jméno portu "BT:Star Micronics"a nastavení "portable;escpos". Samotné otevření portu probíhá následovně:

```
sendCommand(context, portName, portSettings, commands)) {
 noProblem = false;
}
```

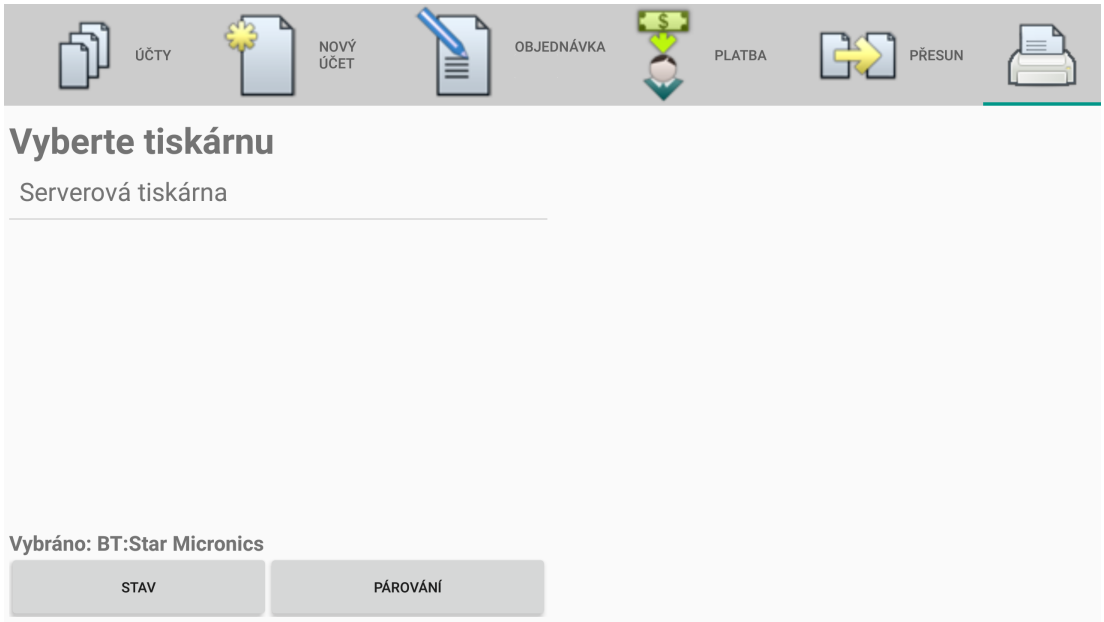
Atribut commands obsahuje data ze třídy StarBitmap. Pokud v této fázi tisk selže, je uživateli nabídnuto zda pokračovat v platbě či nikoliv.

9.6 Problémy při tisku a řešení

Bohužel, při testování tisku byl zjištěn velký problém - Knihovna PDFBox pro Android je velmi pomalá a příprava tisku trvala dokonce několik vteřin. To je pro tak jednoduchou operaci nepřípustné a proto jsem se dodatečně s Tomášem Červenkou dohodl, že převod do bitmapy bude probíhat na serveru a v přijatém JSON objektu bude již rovnou obrázek ve formátu PNG. K tomu jsem mu připravil klasickou ne-Androidí verzi PDFBox knihovny a její implementaci, kterou do serveru přidal.
Při vlastním testování rychlosti převodu PDF do PNG na počítači k žádné prodlevě nedocházelo - převod trval jen okamžik.

9.7 Nastavení tisku

Do aplikace jsem také přidal další fragment, obsahující několik jednoduchých widgetů pro výběr tiskárny - ty dovolují výběr mezi spárovanými tiskárnami nebo tiskárnou serverovou. Metoda pro získání použitelných tiskáren byla již přímo obsažena v API. Do aplikace jsem přidal další záložku - nastavení tiskárny (server/bluetooth) Vypadá to takto:



Obrázek 21 Nastavení tisku

10 Testy aplikace CashBob

Po implementaci funkcí byly provedeny testy uživatelské přívětivosti. Účelem těchto testů je pozorovat počínání uživatelů při provádění zadaných pokynů a shromáždění poznatků o míře úspěšnosti za účelem odstranění nedostatků a nelogičností uživatelského rozhraní.

10.1 Metodika testování

V testování mi pomohli 3 uživatelé, kteří nikdy před tím neměli příležitost aplikaci vyzkoušet a tedy šli testovat bez předchozí přípravy.

Uživatel	O uživateli
Uživatel 1	Muž 46 let, základní zkušenosti práce s PC
Uživatel 2	Žena 45 let, minimální zkušenosti práce s PC
Uživatel 3	Žena 20 let, mírně pokročilé zkušenosti práce s PC

Uživatelům bylo předloženo zadání úkolů a po splnění jsem se ještě ptal na několik detailů.

10.2 Zadání úkolů

1 Nastavte aplikaci podle zadání
2 Přihlaste se pod následujícím jménem a heslem
3 Vytvořte nový účet pro stůl 4 a pojmenujte jej dle libosti
4 Vytvořte novou objednávku dle dodaného soupisu
5 Přesuňte některé položky do nového účtu
5 Proveďte platbu jedné vybrané položky z prvního účtu a vytiskněte účtenku

10.3 Výsledky testování

Nejdříve bych shrnul poznatky, které jsou pro všechny uživatele společné. Přihlašování proběhlo bez problému, byla zde ale tendence přihlásit se ještě před zadáním uživatelského jména a hesla - aplikace ale v chybové hlášce specifikovala problém načež došlo k nápravě a doplnění nastavení. Vytvoření účtu proběhlo bez problémů u všech tří uživatelů - přejmenování ale provázely problémy - ani jeden uživatel nedokázal napoprvé najít způsob jak přejmenovat účet - Textové pole se totiž "tváří"jako obyčejný text bez zjevných indikací.
Přidání položek na účet u uživatelů 2 a 3 proběhlo v pořádku, uživatel 1 potřeboval moment v zorientování se v systému výběru položky.
Pouze uživatel 3 dokázal na první pokus vytvořit nový účet přímo v dané záložce a

přesunout položky.
Platba obecně proběhla v pořádku u všech tří uživatelů - systém přesunu položek je stejný jako při objednávání.

10.4 Poznatky

10.4.1 Uživatel 1

Uživatel 1 si stěžoval především na způsob výběru položek - zejména výběr více než jedné, rád by místo klikání spíše tlačítko, které by zadalo specifikovaný počet položek. Dále navrhl přejmenování tlačítka o "Úroveň výše"na "Zpět"

10.4.2 Uživatel 2

Uživatel nepodal žádný speciální názor či připomínku, kterou sem již nezmínil.

10.4.3 Uživatel 3

Taktéž pro uživatele číslo 3.

10.5 Náprava zjištěných nedostatků

10.5.1 Pojmenování účtu

Přidal jsem nadpis "Přejmenujte účet"nad textbox.

10.5.2 Výběr položek

Přejmenoval jsem tlačítko pro postup v menu o úroveň výše, nyní se jmenuje jednoduše "Zpět". Také jsem u jednotlivých položek nastavil formátování ceny na "bold", což zjednodušilo rozlišení mezi kategorií a zbožím.

10.5.3 Přesun zboží

Přesun zboží jsem nechal tak, jak je, jelikož si myslím, že změny nepotřebuje - pouze jeden uživatel byl rozložením mírně zmaten, ostatní s ním neměli problém.

10.6 Testování funkčnosti aplikace CashBob

Vyjma testů přívětivosti a opětovného testování pomocí statických testů jsem také testoval v reálném provozu s použitím desktopového serveru pomocí uživatele "cisnik"s heslem "heslo"a také na serveru Tomáše Červenky, kde není nutná identifikace. Jelikož aplikace samotná neukládá žádný stav (vyjma vybraného účtu) lokálně, zúžilo se testování na kontrolu správného zpracování dat poskytnutých serverem, což proběhlo úspěšně.

11 Závěr a zhodnocení vývoje obou aplikací

Závěrem bych rád zhodnotil splnění cílů této bakalářské práce. Hlavním bodem společným pro obě aplikací byl iterativní vývoj, který se mi podařilo s úspěchem provést při implementaci všech funkcí aplikace Baby monitor. Každá iterace vždy přinesla něco nového, nějaký nový nápad, či způsob řešení problémů. Několikrát jsem dokonce pozměnil návrh funkcionality - což je dle mého názoru hlavní účel iterativního programování. Z toho důvodu je škoda, že se mi z nedostatku času nepodařilo to samé provést u implementací funkcí aplikace CashBob.
S ohledem na velikost výsledného dokumentu musím konstatovat, že jsem toho na práci měl poměrně dost - práce samotná mě ale bavila, na projektu Baby monitor budu v budoucnu určitě dále pracovat a zdokonalovat jej - byla by škoda jej zanechat v současném stavu, přestože se mi v průběhu iterativního vývoje aplikace vlastně podařilo vytvořit celek, který funkčně splňuje mé prvotní představy. Implementovány byly všechny funkční i nefunkční požadavky původního návrhu, až na jednu hlavní výjimku, a sice přenos videa na vyžádání. Mrzí mě, že jsem zatím nebyl schopen touto funkcí aplikaci obohatit a v budoucnu bych toto tedy ještě rád napravil.
S jistotou mohu říci, že jsem si toho z projektu mnoho odnesl - vzhledem k rozdělení zadání na dvě části a mému rozhodnutí vypracovat tyto části samostatně jsem mohl část zkušeností nabraných při vývoji aplikace Baby monitor převést i na vývoj aplikace CashBob, byť tedy jen tisku.
Co se ostatních bodů zadání týče, myslím, že se mi podařilo splnit je všechny tak, jak jsem očekával.

11.1 Publikace Baby monitoru na Google Play

Závěrem vývoje aplikace by dle zadání měla být i publikace na Google Play. Bohužel jsem z důvodu dodatečného vylepšování aplikace nemohl panu Ing. Komárkovi poskytnout podklady a APK aplikace dostatečně brzo na to, abych zde mohl zahrnout finální odkaz na stažení. Jelikož se ale url v obchodě většinou určuje pomocí balíku aplikace zmíněného v manifestu, je velmi pravděpodobné, že odkaz má následující znění:

https://play.google.com/store/apps/details?id=zarecky.babymonitor

11.2 Poznámka k implementaci tisku v aplikaci CashBob

Implementace tisku nemohla být řádně dokončena, protože neexistoval způsob, jak komunikaci s API otestovat - metoda na testovacím serveru Tomáše Červenky nebyla zatím vůbec implementována. Z toho důvodu jsem musel testovat lokálně - s použitím obrázku účtenky v domovském adresáři systému. Zmíněný obrázek "receipt.png"jsem pro úplnost také přiložil na CD.

Literatura

[1] *Smartphone OS Market Share, 2015 Q2.* URL: http://www.idc.com/prodserv/smartphone-os-market-share.jsp (cit. 19. 04. 2016).

[2] *Google Play store.* URL: play.google.com/store (cit. 28. 08. 2015).

[3] *Bakalářská práce Lukáše Vyhlídky, 2013.* URL: https://dip.felk.cvut.cz/browse/pdfcache/vyhliluk_2013dipl.pdf (cit. 14. 04. 2016).

[4] *Java RMI.* URL: https://docs.oracle.com/javase/tutorial/rmi/ (cit. 22. 05. 2016).

[5] *REST.* URL: https://en.wikipedia.org/wiki/Representational_state_transfer (cit. 22. 05. 2016).

[6] *Bakalářská práce Tomáše Hogenauera, 2016.* URL: https://dspace.cvut.cz/bitstream/handle/10467/62692/F3-BP-2016-Hogenauer-Tomas-Android%20klient%20restauracniho%20systemu.pdf?sequence=1 (cit. 14. 04. 2016).

[7] *Differences of Java in Android.* URL: https://en.wikipedia.org/wiki/Comparison_of_Java_and_Android_API (cit. 14. 04. 2016).

[8] *Android developer guide.* URL: https://developer.android.com/guide/index.html (cit. 14. 04. 2016).

[9] *Android layout.* URL: https://developer.android.com/guide/topics/ui/declaring-layout.html (cit. 14. 04. 2016).

[10] *Gradle.* URL: http://gradle.org/ (cit. 14. 04. 2016).

[11] *Android developer dashboards.* URL: https://developer.android.com/about/dashboards/index.html (cit. 14. 04. 2016).

[12] *Model-view-controller, Wikipedia).* URL: https://en.wikipedia.org/wiki/Model%C3%A2%C2%80%C2%93view%C3%A2%C2%80%C2%93controller (cit. 14. 04. 2016).

[13] *UDP, Wikipedia.* URL: https://en.wikipedia.org/wiki/User_Datagram_Protocol (cit. 16. 05. 2016).

[14] *Bluetooth vs WIFI power consumption on Android - a practical test using Youtube.* URL: http://www.clearevo.com/ecodroidlink/bluetooth_vs_wifi_on_android_battery_consumption/ (cit. 14. 04. 2016).

[15] *Diffen.* URL: http://www.diffen.com/difference/Bluetooth_vs_Wifi (cit. 28. 08. 2015).

[16] *Tutorial: Selecting an Optimal Recording Format.* URL: http://www.totalrecorder.com/recording_format.htm (cit. 14. 04. 2016).

[17] *Unicast, Wikipedia.* URL: https://en.wikipedia.org/wiki/Unicast (cit. 14. 04. 2016).

[18] *Android Studio.* URL: https://developer.android.com/studio/intro/index.html (cit. 14. 04. 2016).

[19] *Material Design introduction*. URL: https://www.google.com/design/spec/material-design/introduction.html# (cit. 14. 04. 2016).

[20] *Material Design compatibility*. URL: https://developer.android.com/training/material/compatibility.html (cit. 14. 04. 2016).

[21] *GNU Imaging tool, GIMP*. URL: https://www.gimp.org/ (cit. 14. 04. 2016).

[22] *Google Groups, Reception of UDP packets in sleep mode*. URL: http://groups.google.com/forum/?fromgroups=#!topic/android-platform/OpbSdp9FTmA (cit. 19. 05. 2015).

[23] *Pulse-code modulation, Wikipedia*. URL: https://en.wikipedia.org/wiki/Pulse-code_modulation (cit. 14. 04. 2016).

[24] *As is, Wikipedia*. URL: https://en.wikipedia.org/wiki/As_is (cit. 14. 04. 2016).

[25] *As is, Wikipedia*. URL: https://developer.android.com/guide/topics/ui/settings.html (cit. 14. 04. 2016).

[26] *PMD*. URL: http://pmd.github.io/ (cit. 14. 04. 2016).

[27] *FindBugs*. URL: http://findbugs.sourceforge.net/ (cit. 14. 04. 2016).

[28] *AsyncTask*. URL: https://developer.android.com/reference/android/os/AsyncTask.html (cit. 14. 04. 2016).

Příloha A

Obsah CD