Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Bezpečný záznam telefonních hovorů na chytrých mobilních telefonech

Bc. Ondřej Hrdý

Diplomová práce
2012

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2011/2012**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Ondřej Hrdý**
Osobní číslo:	**I09362**
Studijní program:	**N2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Bezpečný záznam telefonních hovorů na chytrých mobilních telefonech**
Zadávající katedra:	**Katedra softwarových technologií**

Zásady pro vypracování:

V úvodní části práce je nutné provést přehled aktuálního stavu prostředí chytrých mobilních telefonů z pohledu zastoupení jednotlivých platforem na trhu, možností vývoje a distribuce software. Zvláštní pozornost bude věnována technickým omezením týkajícím se záznamu hlasové komunikace a bezpečnosti jednotlivých platforem.
Primárním cílem diplomové práce je návrh a implementace aplikace pro záznam telefonních hovorů a jejich přeposlání na externí server. Zvláštní důraz bude kladen na bezpečnost tak, aby byly pořízené záznamy jednoznačně nezmanipulovatelné.
Implementace bude provedena v jazyce Java pro zařízení postavená na Android OS.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

MARK, D.; NUTTING, J.; LAMARCHE, J. Beginning IPhone 4 Development: Exploring the IOSSDK. [s.l.] : Apress, 2011. 676 s. ISBN 978-1430230243.
ENCK, W.; ONGTANG, M.; MCDANIEL, P. Understanding android security. Security & Privacy : IEEE. 2009, 7, s. 50-57.
MEIER, R. Professional Android 2 application development. Indianapolis : Wiley Publishing,Inc., 2010. 576 s. ISBN 978-0-470-56552-0.
MURPHY, M.L. Android programming tutorials. 3rd edition. Indianapolis : CommonsWare, Llc, 2010. 434 s. ISBN 978-0981678047.

Vedoucí diplomové práce: **Ing. Radek Švestka**
RETIA, a. s.

Konzultant diplomové práce: **Ing. Lukáš Čegan, Ph.D.**
Katedra informačních technologií

Datum zadání diplomové práce: **31. října 2011**
Termín odevzdání diplomové práce: **18. května 2012**

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

prof. Ing. Antonín Kavička, Ph.D.
vedoucí katedry

V Pardubicích dne 15. listopadu 2011

PROHLÁŠENÍ

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako Školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně Univerzity Pardubice.

V Pardubicích dne 20. 8. 2012 Bc. Ondřej Hrdý

PODĚKOVÁNÍ:

Tímto bych rád poděkoval svému vedoucímu práce Ing. Švestkovi a konzultantovi Ing. Pantůčkovi. Za jejich odbornou pomoc, cenné rady a poskytnuté materiály, které mi pomohly při zpracování diplomové práce. Také bych rád poděkoval Ing. Čeganovi, PhD. a Ing. Bažantovi, PhD. za rady a připomínky týkající se obsahu této práce.

ANOTACE

Tato diplomová práce se zabývá vývojem aplikací na operačních systémech pro chytré mobilní telefony a další přenosná zařízení. Zvláštní důraz je kladen na bezpečnost. První kapitola představuje základní hardwarové vlastnosti stávajících zařízení a společné charakteristiky vývoje na příslušných operačních systémech. Dále jsou podrobněji porovnány jednotlivé platformy z pohledu vývoje, nástrojů, bezpečnosti a distribučních kanálů. Samostatná část je věnována též multiplatformnímu vývoji aplikací. V druhé části byla pro Android OS vytvořena aplikace pro bezpečný záznam telefonních hovorů. Popsány jsou vlastnosti operačního systému, které přímo ovlivnily vývoj. Dále jsou detailněji rozebrány části implementace samotné, rozdělené dle logických úseků. Poslední část se věnuje problémům, které mohou mít potenciální vliv na fungování a nasazení aplikace a uvedena jsou i možná řešení.

KLÍČOVÁ SLOVA

Android OS, Windows Phone 7, iOS, bezpečnost, záznam hovorů

TITLE

Secure voice call recording on smartphones

ANNOTATION

This master thesis deals with application development on operating systems for smartphones and other portable device. Particular emphasis is placed on security. The first chapter introduces the basic hardware features of current devices and common characteristics of development on the respective operating systems. Further individual development platforms are compared in terms of software development, tools, security and distribution channels. A separate section is devoted to cross-platform development. In the second part of this thesis application for secure voice call recording on Android OS was developed. Described are the characteristics of the operating system, that directly affected the development. Next the implementation itself is described, divided by logical sections. The last part deals with problems that may have potential impact on real world deployment of this application. Possible solutions are also listed.

KEYWORDS

Android OS, Windows Phone 7, iOS, security, voice call recording

OBSAH

SEZNAM TABULEK

SEZNAM OBRÁZKŮ

SEZNAM ZKRATEK

API	(Application Programming Interface) – aplikační programové rozhraní
CLR	(Common Language Runtime) - běžné jazykové prostředí
CPU	(Central Processng Unit) – centrální výpočetní jednotka počítače
CSR	(Certificate Signing Request) – žádost o podpisový certifikát
FOTA	(Firmware Over-the-Air) – bezdrátová distribuce firmware zařízení
FUP	(Fair User Policy) – řízení sdílených prostředků
GPRS	(General Packet Radio Service) – služba pro přenos dat v mobilních zařízeních
GPS	(Global Positioning System) – standard pro určení pozice na planetě
GPU	(Graphic Processing Unit) – grafická jednotka počítače
IDE	(Integrated Development Environment) – vývojové prostředí
IPC	(Inter-process Communication) – meziprocesorová komunikace
JSON	(JavaScript Object Notation) – JavaScriptový objektový zápis
LTE	(3GPP Long Term Evolution) – služba pro bezdrátový přenos dat
MCC	(Mobile Counry Code) – identifikátor země operátora
MNC	(Mobile Network Code) – identifikátor sítě mobilního operátor
NDA	(Non-Disclosure Agreement) – smlouva o nezveřejnění informací
NFC	(Near Field Communication) – standard pro bezdrátovou komunikaci
OHA	(Open Handset Aliance) – konsorcium společností definující standardy
SDK	(Software Development Kit) – nástroje pro vývoj software
SoC	(System On a Chip) – čip obsahující řadu odlišných komponent
SSL	(Secure Socket Layer) – Vrstva pro šifrovaný přenos dat
VOIP	(Voice Over Internet Protocol) – přenos digitalizovaného hlasu internetem
WiFi	(Wireless Fidelity) – rodina standardů pro bezdrátový přenos dat

Úvod

V posledních několika letech se setkáváme se stále rostoucím počtem nových mobilních zařízení. Pokles cen, nárůst výkonu hardware, přístupná uživatelská rozhraní a úspěšné marketingové kampaně zajistily rozšiřující se uživatelskou základnu. Začaly se objevovat i zcela nové rodiny zařízení od chytrých hodinek, přes pokročilé sportovní počítače až po tablety či televize postavené na mobilních operačních systémech. Nedílnou součástí nové generace těchto přenosných zařízení jsou právě nové operační systémy a distribuční kanály pro aplikace.

Změnil se i způsob, jakým jsou mobilní telefony používány. Uživatelé častěji přistupují k internetu a jejich osobní zařízení mohou provádět řadu úkonů dříve realizovatelných pouze na noteboocích či desktopech [1]. Toto umožnilo vznik zcela nových typů software a mobilních verzí existujících aplikací. Uživatel si se svým mobilním telefonem vytváří skutečně osobní vztah - má možnost vybrat si ze široké škály různorodého hardware, přizpůsobit si nejen vzhled uživatelského rozhraní, ale i softwarové vybavení.

Cílem této práce je definovat odlišnosti ve vývoji software pro mobilní zařízení a využít tyto znalosti k navržení a implementaci aplikace pro bezpečný záznam telefonních hovorů. Ta by měla po instalaci zcela autonomně, bez přičinění uživatele, provádět audio záznam telefonních hovorů a odesílat tyto záznamy na vzdálený server pomocí běžných datových přenosů.

Aplikace by měla obsahovat skupinu pravidel, na jejichž základě se rozhodne, zda bude konkrétní hovor zaznamenáván či ne. Kromě obecných možností, pro nahrávání všech nebo žádného hovoru, mohou být součástí těchto pravidel i časové intervaly pro nahrávání, či seznamy konkrétních telefonních čísel. Zvláštní výjimkou jsou i neznámá čísla, tedy ta neuložená v adresáři kontaktů zařízení. Obsažen by měl být i mechanizmus pro vzdálenou změnu těchto nastavení.

Zvláštní pozornost by měla být věnována zabezpečení a především znemožnění úpravy nahrávek samotných. Aplikace by měla k zajištění tohoto požadavku využívat nejen bezpečnostní mechanizmy operačního systému, ale možnosti poskytované programovacím jazykem, tedy svoji implementaci jako takovou. V případě, že nebude možné zcela zamezit úpravám záznamů, je třeba s určitou jistotou umožnit detekci neoprávněných zásahů.

Potenciální úskalí implementace a nasazení takovéhoto software je třeba přesně pojmenovat. Pokud bude možno, měla by být zmíněna možná řešení těchto problémů.

1. OPERAČNÍ SYSTÉMY PRO CHYTRÉ MOBILNÍ TELEFONY

Za operační systém pro mobilní zařízení je považován takový systém, který byl specificky navrhnut pro běh na přenosných zařízeních - chytrých mobilních telefonech, tabletech, apod. Tento systém kromě jednotného uživatelského rozhraní musí umožňovat také vývoj a instalaci aplikací třetích stran. Tímto aspektem se odlišuje od systémů pro běžné, tedy ne-chytré, mobilní telefony, které instalaci nativních aplikací třetích stran nepodporují. Pojem desktopový operační systém je v této práci použit k označení systémů známých z osobních počítačů, např. Windows 7, Mac OS X či Ubuntu 11.

1.1. Vlastnosti chytrých mobilních telefonů

Vývoj software pro chytré mobilní telefony s sebou nese řadu specifických vlastností. V této kapitole jsou zmíněny nejdůležitější faktory, které přímo či nepřímo ovlivňují prostředí vývoje pro mobilní telefony. Většina identifikovaných hardwarových vlastností se promítá do struktury mobilních OS, ať už v podobě pozitivního rozšíření možností nebo naopak restriktivních hranic často neznámých na desktopových OS.

1.1.1. Hardwarové vlastnosti

Výpočetní výkon mobilních zařízení je aktuálně jedním z hlavních bodů konkurenčního boje výrobců. V posledních dvou letech byly na trh uvedeny zařízení s dvou i čtyř-jádrovými CPU a mobilní GPU s výkonem dostatečným pro graficky náročné operace. S rostoucím výkonem si uživatelé zvykli používat větší spektrum aplikací. Ty se stávají komplexnějšími a výpočetně náročnějšími. Architektura ARM se stala etalonem v mobilních zařízeních, ačkoliv existují snahy o uvedení alternativ např. v podobě Intel Medfield a Android OS podporující x86 architekturu.

Rostoucí výkon zařízení a jejich kontinuální využívání zvyšuje nároky na spotřebu elektrické energie. Přes jistý pokrok ve výzkumu přenosných zdrojů energie, např. palivových článků [2] a nanobaterií [3], jsou tyto technologie zatím stále běžnému spotřebiteli nedostupné. To nutí výrobce SoC k zohlednění spotřeby jejich produktů a promítá se i do principů fungování OS.

Velikost paměti je dalším významným limitujícím faktorem. Operační paměť pojme jen omezený počet dat běžících aplikací a je primárně přidělována k obsloužení základních požadavků OS - vykreslování grafiky, komunikace s mobilním operátorem, hlasové hovory a SMS. Zbytek paměti je nutné přerozdělit nejen všem viditelně běžícím aplikacím, ale

i službám pracujícím na pozadí. Datové úložiště v podobě integrované paměti, slotu pro paměťové karty nebo jejich kombinace a odlišnost nakládání s touto pamětí v jednotlivých OS, jsou jednou z výzev, kterým musí vývojář čelit.

Většina moderních mobilních zařízení obsahuje hned několik různých senzorů. Mezi ty můžeme řadit např.:

- GPS – poskytující aktuální pozici zařízení na planetě,
- akcelerometr – informující o zrychlení v jednotlivých osách zařízení,
- kompas – určující světové strany dle magnetického pole planety,
- a další.

Tyto senzory jsou nyní běžnou součástí všech moderních SoC a je tedy prakticky nemožné pořídit nové zařízení bez těchto prvků.

Zcela běžnými se staly také prostředky pro bezdrátovou komunikaci na větší (WiFi) či kratší (Bluetooth) vzdálenosti. Novým trendem je také integrace NFC umožňující komunikaci pouze na bezprostřední vzdálenost. Tato vlastnost zajišťuje, že každá provedená akce byla vědomá - uživatel vědomě přiložil zařízení k jinému kompatibilnímu zařízení. Technologie NFC se tedy považuje za bezpečnou i pro realizaci plateb.

Nezanedbatelnou vlastností chytrých mobilních telefonů je také přístup k datovým sítím - internetu. Kromě již zmíněného WiFi je připojení možné realizovat pomocí sítí mobilních operátorů. Rychlosti jednotlivých technologií se liší výrazně od pomalejších, v řádech desítek kbps (GPRS), po deset a více Mbps (LTE). Ne vždy je však v zájmu uživatele tyto datové přenosy využívat (např. z finančních důvodů, FUP omezením či nedostatku baterie) a vývojář se proto nemůže spolehnout na jejich dostupnost.

Významným vývojem prošly také způsoby, jakými uživatel se zařízením komunikuje. Běžné numerické klávesnice nahradily dotykové displeje volitelně s fyzickými qwerty klávesnicemi. Od resistivního dotykového ovládání se přešlo ke kapacitním dotykovým vrstvám. Ty sice neposkytují přesnost na úrovni pixelu, ale pohodlněji se ovládají konečky prstů a umožňují vícedotykové ovládání (tzv. multitouch). Rozpoznávání mluveného slova a syntéza řeči jsou dalším prostředkem komunikace mezi zařízením a uživatelem [4].

Běžnou součástí mobilních telefonů jsou také senzory umožňující záznam videa a pořízení fotografií. Kvalita těchto snímačů stále roste a objevují se zcela nové způsoby jak je využít. Za nejvýznamnější můžeme považovat video hovory a tzv. rozšířenou realitu – augmented reality [5]. Tyto aplikace přidávají do zobrazovaného prostředí nové objekty, nejčastěji

informace založené na pozici (např. památky v bezprostředním okolí) nebo předměty herních mechanizmů.

Některé operační systémy také podporují stále širší spektrum periferií. Většina chytrých mobilních telefonů je připojitelná k externím zobrazovacím zařízením. Méně často se objevuje podpora tzv. USB-host módu, který umožňuje připojit adaptérem řadu běžných periferií – HID, flash disky, atd.

1.1.2. Softwarové vlastnosti

Hardwarové vlastnosti uvedené v předchozí části, spolu s odlišným paradigmatem užití chytrého mobilního telefonu, vyžadují od operačních systémů řešení řady problémů dříve neznámých jak v prostředí desktopů, tak přenosných zařízení. Zároveň tyto nové prvky otevírají možnosti pro nová softwarová řešení, která byla v předchozích letech nerealizovatelná nebo by měla vzhledem k značně menší uživatelské základně mnohem menší oblíbenost (např. služby postavené na aktuální lokaci uživatele - foursquare).

Omezený počet hardwarových prostředků a snížení nároků na zdroj energie se promítá do většiny mobilních operačních systémů přísnějším řízením přístupu k těmto prostředkům. Uživatel má možnost využívat v jeden moment plně jen jednu aplikaci. Multitasking nesystémových aplikací buď zcela neexistuje, nebo je omezen na seznam přesně daných úkonů (např. přehrávání hudby na pozadí - iOS, WP7). Obvyklým řešením je také vynucené ukončování dříve spuštěných aplikací ve prospěch těch systémových nebo uživatelem aktuálně používaných.

Vynucené ukončování aplikací je běžnou vlastností prakticky všech konkurenčních OS pro chytré mobilní telefony. Často totiž může nastat situace, ve které je nutné uvolnit prostředky pro obsloužení systémové operace (např. požadavek události na okamžité vyřízení - příchozí telefonní hovor) nebo získat prostředky pro jinou aplikaci třetí strany (např. spuštění výpočetně náročné aplikace - hry). Naštěstí všechny platformy volají definované metody před ukončením a před případným obnovením aplikace, takže má vývojář možnost uložit všechna nezbytná data, včetně např. aktuálního stavu. Pokud bude chtít, může je v případě obnovení aplikace znovu načíst. Z pohledu uživatele pak celý proces působí, jako by aplikace nebyla nikdy přerušena.

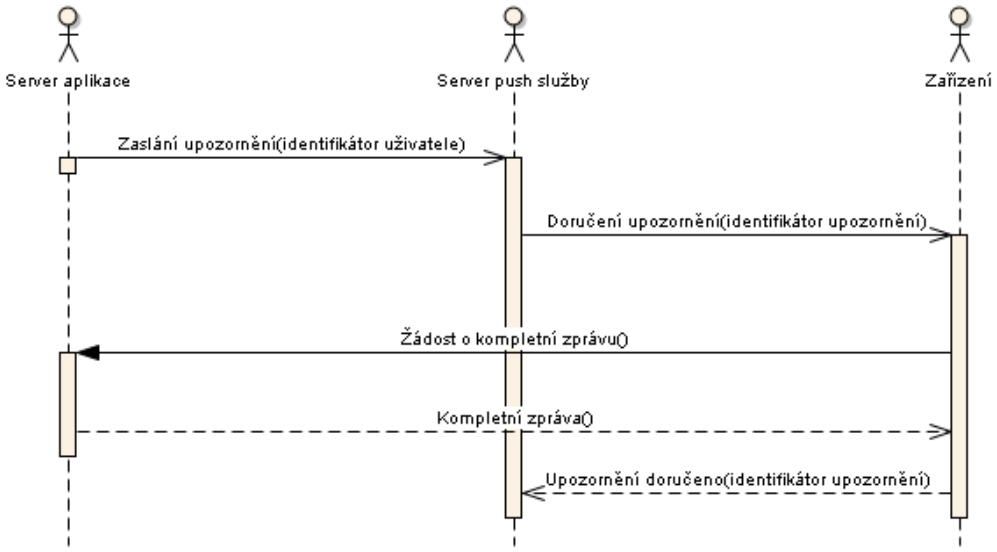
Některé operační systémy také poskytují uživateli nástroje pro správu běžících aplikací. Umožňují jejich přepínání, ukončení a případně i detailní přehled o tom, jak využívají

prostředky (např. u Android OS ve vztahu k využití baterie). Pokud tyto nástroje nejsou součástí systému, je možné tuto funkcionalitu získat instalováním aplikací třetích stran.

S výše uvedeným úzce souvisí životní cyklus aplikace. Již bylo zmíněno nebezpečí, že bude aplikace nečekaně ukončena nebo bude řízení předáno jiné aplikaci. Je ale také třeba si uvědomit, že ikona v seznamu instalovaných aplikací je jen jednou možnou vstupní branou do programu samotného. Vývojář má možnost ve své aplikaci implementovat odezvu na různé systémové události (např. příchozí sms, začátek nabíjení, boot systému), využívat časově založeného plánování (např. spustit aplikaci ve stanovený čas nebo opakovaně po uběhnutí intervalu) nebo spustit službu, která poběží kontinuálně na pozadí. Struktura některých operačních systémů je natolik modulární, že umožňují vývojáři nahradit jednotlivé části OS (např. domácí obrazovka).

Všechny operační systémy také musí řešit komunikaci mezi aplikacemi a přístup ke sdíleným prostředkům. Za ty můžou být považována především data aplikací a dříve zmíněné senzory. Mobilní telefon je totiž na rozdíl od běžného počítače chápán jako exkluzivně osobní zařízení. Žádný z dále zmíněných operačních systémů nepodporuje více-uživatelské prostředí. Naopak jsou vývojářům poskytována rozhraní přístupu ke “sdíleným” informacím. Zde jde především o kontakty, seznamy hovorů či záznamy v kalendáři. Často je také možno realizovat rozšíření systémových aplikací implementací tzv. plug-inů. Tak může být například přidán nový IM protokol do aplikace zpráv. Jednotlivé operační systémy také poskytují mechanizmy pro vzájemnou komunikaci aplikací třetích stran - sdílení dat a zveřejňování API nesystémových aplikací.

Všechny aktuálně rostoucí operační systémy pro mobilní telefony také poskytují způsob jak realizovat tzv. push notifikace. Jde o upozornění, která jsou zaslána ze serveru přímo na zařízení, aniž by byla komunikace se serverem zařízením iniciována. To umožňuje vývojáři nejen předat uživateli důležitou informaci okamžitě (bez čekání na konec poll intervalu), ale zároveň odstranit zbytečné vyřizování požadavků s prázdnou odpovědí na straně serveru. Všechna řešení se shodují v principu, kterým fungují (viz Obrázek 1 – Obecný princip push upozornění). Server vývojáře kontaktuje server poskytovatele OS, ten doručí informaci na konkrétní zařízení uživatele, který následně může stáhnout požadovanou informaci ze serveru vývojáře. Okamžité doručení těchto push upozornění ale není garantováno ani jedním operačním systémem (např. uživatel nemusí být v dosahu datového připojení, na zařízení nemusí být uživatel přihlášen svým účtem). Doporučuje se proto využít je výhradně k iniciování komunikace se serverem ze strany zařízení, ne k zaslání informace samotné.



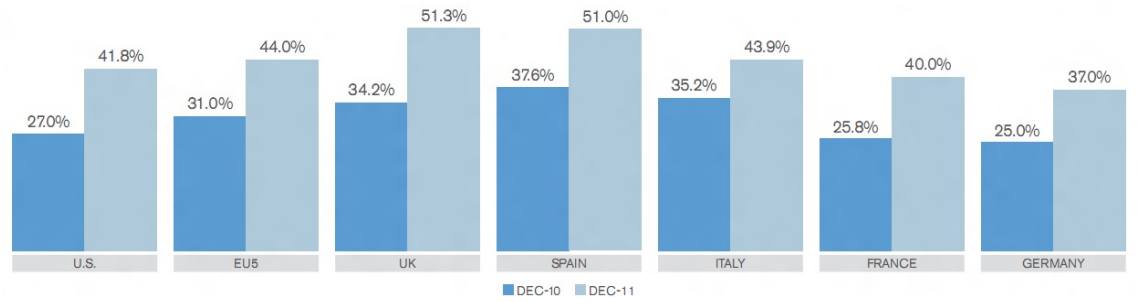
Obrázek 1 – Obecný princip push upozornění

Zdroj: vlastní zpracování

Možnosti vývoje software, bezpečnostní vlastnosti a distribuční kanály pro aplikace v jednotlivých hlavních platformách budou z důvodů významných odlišností rozebrány podrobněji v kapitole 1.4 Detailní pohled na vybrané operační systémy.

1.2. Zastoupení platforem

Trh s mobilními telefony prodělal v posledních letech několik výrazných změn. Výkonný hardware, pokles jeho cen, významný pokrok ve zpříjemnění uživatelských rozhraní a úspěšný marketing vedl k růstu prodeje chytrých mobilních telefonů na úkor telefonů bez operačního systému otevřeného k doinstalování aplikací třetích stran. Uvedené skutečnosti vedou ke vzniku nových obchodních příležitostí, obchodních modelů a distribučních kanálů nejen pro aplikace, ale i média (např. hudba, film). Data převzatá z výzkumu společnosti comScore (viz Obrázek 2 - Míra adopce chytrých mobilních telefonů) nám ukazují nárůst uživatelské základny chytrých mobilních telefonů ve vybraných zemích mezi roky 2010 a 2011.



Obrázek 2 - Míra adopce chytrých mobilních telefonů

Zdroj: [6]

Vidina nové příležitosti přilákala do prostředí ovládaného společnostmi Nokia - Symbian, Microsoft - Windows Mobile, PalmOne - PalmOS a Research In Motion - Blackberry nové konkurenty. Po prodejně úspěšném iOS společnosti Apple, představila společnost Google konkurenční platformu Android OS. Faktory zmíněné v předchozím odstavci začaly postupně lákat uživatele dříve zavedených operačních systémů. Významně klesající zájem o platformy předchozí generace donutil poskytovatele mobilních operačních systémů k inovacím, které dopadly více (Microsoft - Windows Phone 7) či méně (Palm, Inc. - webOS, Nokia - MeeGo) úspěšně. Oznámena byla také nová generace operačního systému od Research In Motion - BlackBerry 10. Detailní informace o jeho vlastnostech a dostupnosti zařízení zatím nejsou známy.

Tabulka 1 - Zastoupení operačních systémů na trhu (v %)

Operační systém	2010	2011	2012	2015
Symbian	37,6	19,2	5,2	0,1
Android	22,7	38,5	49,2	48,8
Research In Motion	16,0	13,4	12,6	11,1
Microsoft	4,2	5,6	10,8	19,5
iOS	15,7	19,4	18,9	17,2
Ostatní	3,8	3,9	3,4	3,3

Zdroj: [7]

Specifickým trhem pro chytré mobilní telefony jsou tzv. low-endová zařízení, která konkurují především nízkou cenou a cílením na rozvojové trhy. Do tohoto segmentu nasadily společnosti Nokia a Samsung systémy Symbian S40/S60, resp. BadaOS.

1.3. Aplikace pro nahrávání telefonních hovorů

Implementační část této diplomové práce návrh a realizaci vzorové aplikace pro nahrávání telefonních hovorů na chytrých mobilních telefonech. Podrobný seznam požadavků na toto softwarové řešení je uveden níže.

1.3.1. Požadavky

Jednotlivé požadavky byly rozděleny dle důležitosti na primární a sekundární. Za primární jsou považovány takové požadavky, při jejichž splnění bude možné aplikaci využít k požadovanému záměru zadavatele projektu. Sekundární požadavky usnadní případné zpracování získaných dat nebo zpříjemní uživateli použití aplikace.

Primární

- Záznam hovoru,
- upload záznamu na server,
- vizuální indikace běhu aplikace,
- uložení metadat o hovoru (např. čas pořízení, telefonní číslo),
- nezmanipulovatelnost záznamu[1],
- vzdálené nastavení,
- zohlednění principů vývoje na operačních systémech pro chytré mobilní telefony.

Sekundární

- Záznam jednotlivých směrů v oddělených kanálech,
- zpožděný upload v případě nedostupnosti připojení,
- filtr nahrávání,
- neodstranitelnost záznamu,
- uživatelské rozhraní pro nastavení.

1.3.2. Konkurenční aplikace

Pro platformu Android jsou aplikace zaznamenávající telefonní hovory dostupné přes obchod Google Play. Některé z těchto aplikací umožňují zaslání nahrávek emailem či upload

[1] *za nezmanipulovatelný záznam považujeme takový záznam, u nějž je možné s určitou jistotou stanovit, že nebyl nijak upravován, resp. naopak jsme schopni určit, že na konkrétním záznamu došlo záměrně či bez prokazatelného záměru k dále nespecifikovaným změnám.*

např. do Evernote (viz TotalRecorder od Killer Mobile). Žádná aplikace však neposkytuje požadovanou úroveň zabezpečení a vzdálenou správu nastavení.

U jednotlivých aplikací je zmíněn problém kompatibility s některými verzemi Android OS a zařízeními – viz 2.6.1 Záznamová API. Aplikace proto buď obsahují možnost volby specifických nastavení pro konkrétní zařízení a verzi OS, nebo je distribuovány spolu se seznamem otestovaných kompatibilních zařízení.

Odlišným řešením pro nahrávání telefonních hovorů je použití vzdáleného záznamového zařízení. Tato skupina aplikací, např. Call Recording od TPS Games, využívá technologie konferenčních hovorů – vzdálené záznamové zařízení je přidáno jako další účastník hovoru a celá nahrávka pak není pořízena, a ani fyzicky přítomna, na uživatelově zařízení.

1.3.3. Legislativní aspekty

Významným problémem při pořizování audio záznamu telefonního hovoru může být legislativa. Ta se nejen významně liší v jednotlivých zemích, ale často nereflektuje překotný vývoj technologií [8]. Jednotlivé právní úpravy obecně buď zakazují pořízení záznamů, nebo povolují jejich pořízení se souhlasem jedné či obou stran komunikace. Na tyto obecná pravidla se dále vztahuje řada výjimek.

Legislativa České republiky umožňuje záznam hovoru se souhlasem zaznamenané osoby. Ten nemusí být výslovný a stačí tedy, pokud je souhlas odvoditelný z okolností, za nichž byl záznam pořízen. Výjimky z těchto pravidel poskytují zákonné licence – úřední, zpravodajská a pro vědecké či umělecké účely.

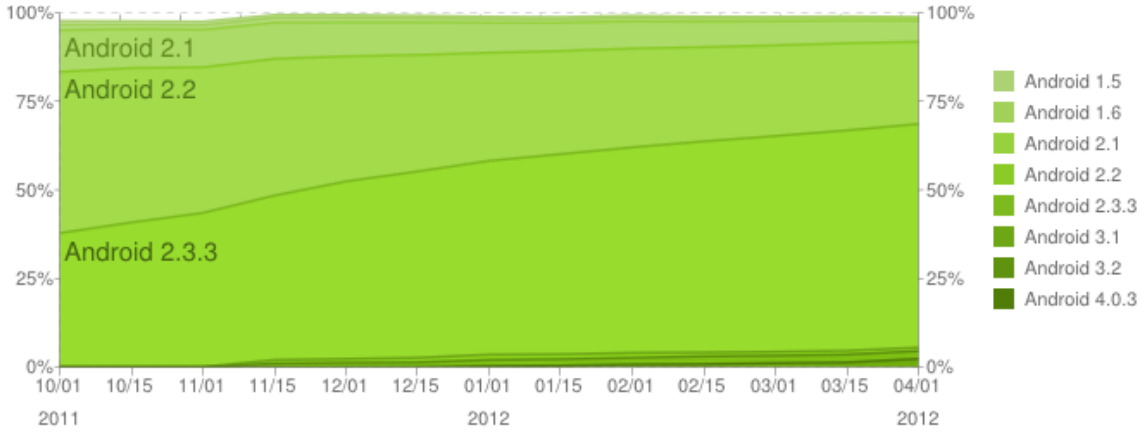
1.4. Detailní pohled na vybrané operační systémy

Tato kapitola poskytuje detailní pohled na možnost realizace popsané aplikace pro nahrávání hovorů ve stávajících operačních systémech pro mobilní telefony. Nejde o zcela vyčerpávající popis jednotlivých vlastností a důraz je kladen na prvky relevantní k realizaci požadovaného software. Jednotlivé platformy byly vybrány na základě růstu, resp. poklesu, zastoupení jednotlivých operačních systémů ve sledovaných zemích v posledních letech, viz kapitola 1.2 Zastoupení platforem.

1.4.1. Android OS

Operační systém společnosti Google postavený na OS Linux byl představen 20. září 2008. Od ostatních porovnávaných systémů se liší tím, že některé jeho verze jsou dostupné zdarma pro nasazení na zařízeních různých výrobců. Projekt je uveřejněn jako open source s relativně

krátkou délkou iterací mezi jednotlivými verzemi. Jistý odklon nastal u větve 3.* určené pro tablety. Ta nebyla zcela volně dostupná pro výrobce zařízení a v současnosti je nahrazována verzí 4.*, která představuje jednotný systém pro chytré mobilní telefony i tablety. V době psaní této práce je nejčastějším verzí OS nasazenou na zařízeních 2.3.*, viz Obrázek 3 - Zastoupení různých verzí Android OS.



Obrázek 3 - Zastoupení různých verzí Android OS

Zdroj: [9]

Programovací jazyk

K programování se využívá především jazyk Java s rozšiřujícími knihovnami, které zpřístupňují funkce OS. Další možností je provádění kódu C a C++ pomocí takzvaného NDK, native development kit. To ovšem znemožňuje přímý přístup k některým datům a funkcím. Proto je běžným přístupem volání C/C++ rutin z Java kódu.

Vývojářské nástroje

Společnost Google poskytuje celou řadu nástrojů pro vývoj aplikací. Dostupná je integrace do různých IDE, ale doporučeno je použití multiplatformního Eclipse. K němu jsou také poskytovány oficiální plug-iny. Aplikace je možné debugovat jak na zařízení, tak pomocí emulátoru. Ten je kvůli nižší rychlosti vhodný především při testování uživatelského rozhraní pro různá rozlišení a DPI.

Bezpečnost

OS Android v mnoha ohledech využívá k zajištění bezpečnosti svého původu v OS Linux. Každá aplikace běží pod jedinečným uživatelským účtem, tedy s unikátním UID procesu. Pro každou aplikaci je vytvořena příslušná složka a vývojář může nastavit přístupová práva pro

jednotlivé soubory podobně jako v Linux. Tento přístup zajišťuje naprosté oddělení aplikace a jejích dat nejen od systému, ale také od ostatních aplikací, tzv. sandboxing.

Pro vzájemnou komunikaci slouží mechanismus povolení (permissions). V souboru popisujícím aplikaci - *AndroidManifest.xml* - vývojář uvádí povolení nezbytná pro bezproblémovou funkčnost aplikace. Vyžadovat tato povolení je možné pouze při instalaci a update aplikace. Není tedy možné sadu použitých povolení měnit programově. Uživateli jsou vyžadovaná povolení zobrazena během instalace / update aplikace a v případě nestandardních požadavků (např. aplikace chce přistupovat k vytáčení hovorů, aniž by to bylo pro její funkčnost zjevně nezbytné) má možnost instalaci zrušit.

Proces aplikace je přiřazen do příslušných skupin a je mu umožněn přístup k vyžadovanému prostředku. Příkladem budiž povolení INTERNET s GID 3003 umožňující aplikaci přistupovat k internetu. K dnešnímu dni je dostupných více jak 75 různých povolení. Každá aplikace navíc může definovat vlastní sadu povolení. Ta obvykle umožňují ostatním aplikacím manipulovat s jejími daty.

K uložení dat je možné kromě prostoru v telefonu použít rozšířené paměti a paměťové karty. Ty jsou z důvodu kompatibility formátovány starším souborovým systémem VFAT, který neumožňuje nastavení přístupových práv k souborům.

Distribuce aplikací

Distribuce je prováděna pomocí APK souborů. Ty je možné vyhledávat a instalovat z oficiálního Google Play (dříve Android Market), oddělených distribučních systémů (např. Samsung Apps) i přímo z paměti telefonu (získaných překopírováním nebo ze sítě). Aplikace nejsou před uveřejněním kontrolovány správcem Google Play, pro orientaci mezi kvalitními aplikacemi tedy slouží mechanismus komunitního hodnocení a nahlašování závadné aplikace.

Pro bezproblémovou instalaci postačí, aby vývojář aplikaci podepsal. Při update aplikace je vynucováno přesně definované označování jednotlivých verzí software a musí být použit stejný klíč pro podpis aplikace. Díky tomu je aplikaci přiřazeno stejné UID a bude umožněn přístup k datům předchozí verze aplikace.

Uživatel má možnost odstranit kdykoliv celou aplikaci nebo její data. V případě, že by při tomto procesu nastal nějaký problém, má možnost spustit zařízení v bezpečném módu (safe mode). V tom je načten pouze systémový software a následně může být aplikace odstraněna. Google má také možnost nebezpečné aplikace vzdáleně ze zařízení odstranit – remote wipe.

Hardware

Android OS je licencován jako open source a výrobci jej mohou využít na svém zařízení zcela zdarma. Podporovány jsou architektury ARM, MIPS, Power Architecture a x86. Prominentní postavení získávají produkty schvalované společností Google – ty jsou označeny jako "The Google experience". Taková zařízení obsahují plnou sadu software od společnosti Google. Její součástí je mimo jiné plnohodnotná navigační aplikace či obchod Google Play.

Updaty operačního systému mohou být prováděny mechanismem firmware over-the-air (dále jen FOTA) i připojením k počítači pomocí příslušného software. Jelikož výrobci zařízení a mobilní operátoři mají možnost distribuovat systém upravený nebo rozšířený o další funkce, dostávají se aktuální verze OS na zařízení se zpožděním nebo vůbec.

Široké spektrum hardware, od hodinek po televize či ledničky, na kterém je Android OS instalován ve svých různých verzích, vedou k jevu označovanému jako fragmentace Android. Existuje několik mechanismů usnadňujících vývojáři tvorbu aplikace kompatibilní s rozdílnými zařízeními. Emulátor umožňuje simulovat tato zařízení pomocí mnoha různých nastavení, design uživatelského rozhraní může být založen na nestatických (plovoucích) elementech a balíček aplikace může obsahovat několik odlišných uživatelských rozhraní.

Root

Podobně jako ostatní platformy je možné „odemknout" i zařízení s Android OS tak, aby mohly být upravovány systémové soubory, instalován nelegální software nebo např. použit jiný mobilní operátor. Po získání přístupových práv "root" již není možné se spolehnout na jakýkoliv bezpečnostní mechanismus OS. Uživatel může např. kdykoliv odstraňovat nebo editovat soubory aplikace.

1.4.2. iOS

Mobilní telefon iPhone společnosti Apple byl představen 9. ledna 2007 spolu s operačním systémem iPhone OS. Ten je aktuálně dostupný ve své 5. generaci a byl přejmenován na iOS. Jádro systému je odvozené od OS Darwin, který je kompatibilní se standardy Single UNIX specification verze 3 a POSIX UNIX aplikacemi. Na tomto operačním systému také běží další zařízení společnosti Apple, konkrétně iPad, iPod Touch a Apple TV.

Programovací jazyk

Aplikace pro iOS jsou primárně vyvíjeny v Objective-C, který je téměř výhradně využíván v produktech společnosti Apple (kromě zmiňovaného iOS také desktopový operační systém

Mac OS X). Tento jazyk je rozšířením jazyka C, resp. C++. Soubory zdrojového kódu jsou pojmenovávány *.m*, pro kombinaci C a Objective-C, nebo *.mm*, pro kombinaci C++ a Objective-C. Kód je kompilován do nativního strojového kódu pomocí GNU GCC kompileru tak, že rozšíření použitá v Objective-C jsou nejprve převedena do kódu v C/C++. Je proto možné využívat i čisté C a C++. Pro Objective-C je typické např. zasílání zpráv (podobně jako v jazyce Smalltalk), použití delegátů, nebo nutnost spravovat paměť přímo pomocí příkazů *retain* a *release* (do verze iOS 5). Zpřístupněno je také API CococaTouch, které poskytuje především prvky uživatelského rozhraní - rozpoznávání gest, animace aplikací, unifikovaný vzhled ovládacích prvků atd.

Vývojářské nástroje

Jediný plně podporovaný způsob vývoje je pomocí IDE Xcode. Tento software běží pouze pod Mac OS X a vzhledem k licenčním podmínkám tohoto systému je tedy nutné vlastnit počítač od společnosti Apple. SDK iOS je dostupné ke stažení zdarma ze stránek společnosti Apple. Pro jeho použití je ale nezbytné vlastnictví vývojářského účtu. Prodej aplikace v App Store je podmíněn účastí v iOS Developer Program. Členství je zpoplatněno ročním poplatkem, který se liší dle typu programu a země registrace.

Součástí iOS SDK je také iPhone Simulator. Ten se od emulátorů liší tím, že používá kód kompilovaný pro Mac OS X, tedy x86 platformu. Neumožňuje však použít všechny funkce a některé aplikace proto mohou být testovány pouze na fyzických zařízeních.

Bezpečnost

Kořeny, jež má Objective-C v C, jej učinily otevřeným ke známým útokům pomocí přetečení bufferu / integer či formátovaných řetězců string. V případě několikanásobného uvolnění paměti může dojít k pádům aplikace. Dekompilace aplikace je možná s pomocí správných nástrojů - otool, classdump, atd.

Balíček aplikace obsahuje kromě samotného spustitelného programu také unikátní ID, seznam požadovaných povolení, soubor metadat popisujících aplikaci, podpis aplikace a případně další nezbytná data (např. grafické prvky rozhraní, datové soubory). Aplikace je podepsána klíčem, který získá vývojář po vytvoření CSR požadavku pomocí Keychain Access Tool.

Sandobxing, tedy vzájemné oddělení aplikací a zároveň jejich oddělení od OS samotného, je v iOS řešen pomocí mechanismu Mandatory Access Controls. Žádosti o povolení přístupu k prostředku (např. použití GPS) jsou zobrazovány uživateli za běhu aplikace. V dokumentaci

je kromě výrazu sandbox často používán název modulu implementujícího tuto funkci v jádře OS, tj. Seatbelt. Každá aplikace je nainstalována ve vlastním adresáři */private/var/mobile/Applications/*, identifikovaná přiděleným GUID a bez možnosti číst či zapisovat do adresářů ostatních aplikací.

iOS poskytuje na zařízení několik způsobů jak ochránit data. Kromě šifrování, které je prokazatelně nedostatečné [10], nabízí tzv. Keychain úložiště. To využívá k ochraně dat kombinace jedinečného klíče zařízení a PIN. Aplikace nemůže k těmto párům klíč-hodnota přistupovat přímo, ale pouze programově přes definované metody. Pro zvýšení bezpečnosti jsou také vyloučeny ze záloh zařízení. Sdílené Keychain úložiště umožňuje sdílet uložená data mezi několika aplikacemi.

Distribuce aplikací

Distribuce aplikací probíhá přes Apple App Store. Aplikace před uveřejněním prochází detailní kontrolou. Nejprve jsou provedeny automatizované testy detekující chyby v implementaci – např. neuvolňování paměti. Následně je ověřeno, zda aplikace neporušuje závazná pravidla App Store - např. erotický obsah, podezření na malware, platby uvnitř aplikace. Tento proces může být časově náročný a důvody zamítnutí podléhají NDA. Vývojáři je vyplaceno 70 % z ceny každé prodané aplikace.

Alternativou je i distribuce přes uzavřený okruh zařízení v podnikovém sektoru (Enterprise App Stores). I v tomto případě podléhají aplikace kontrolnímu mechanismu. Smluvní podmínky také zajišťuji společnosti Apple možnost aplikaci na uživatelově zařízení vzdáleně deaktivovat. Poslední možností je ad-hoc instalace na až 100 zařízení, která nemusí splňovat výše uvedené podmínky.

Uživatel může kdykoliv odstranit ze zařízení aplikaci třetích stran včetně příslušných dat. Během následující synchronizace s iTunes je mu nabídnuto znovunainstalování aplikace.

Hardware

Operační systém iOS běží výhradně na zařízeních společnosti Apple. CPU těchto zařízení jsou také postavena na architektuře ARM. Podporovány jsou obvykle dvě poslední generace v jednotlivých rodinách zařízení. Tato přímá kontrola nad produktem - od výroby hardware, přes poskytování vývojářských nástrojů, až po řízení distribučních kanálů, zajišťuje vývojářům relativně jasný přehled o podporovaných zařízeních a verzích tohoto systému.

Updaty probíhají současně ve velmi krátkém intervalu systémem FOTA nebo pomocí software iTunes.

Mobilní telefony společnosti Apple využívají výhradně vnitřní paměťové úložiště. Dlouhodobé připojení externího úložiště (např. paměťových karet) není umožněno.

Jailbreak

Provedení Jailbreak poskytuje uživateli na jeho zařízení práva root. Toto je využíváno hlavně k instalaci software, který by nebylo možné získat přes App Store, a různým modifikacím operačního systému. Podobně jako u ostatních platforem, i v tomto případě dochází k zrušení bezpečnostních mechanizmů operačního systému.

Nahrávání telefonních hovorů

Záznam audio nahrávek je možné v iOS provádět pomocí metod API *CoreAudio* nebo *AVFoundation* (třída *AVAudioRecorder*) [11]. Přísně restriktivní přístup k sandboxingu aplikací však zajišťuje, že jsou aplikacím třetích stran během hovoru odebrány veškeré audio prostředky. Stávající aplikace pro záznam telefonní konverzace, prodávané v App Store, přidávají do hovoru dalšího účastníka – vzdálené zařízení pro vytváření záznamu.

1.4.3. Windows Phone 7

Tento operační systém byl uveden na trh společností Microsoft 21. října 2010. Je nástupcem předchozí platformy Windows Mobile 6, určené pro chytré mobilní telefony a PDA. Systém je nadstavbou Microsoft Windows CE 7, operačního systému pro embedded zařízení. Uživatelské rozhraní nazývané Metro je možné považovat za evoluci uživatelského rozhraní z produktů rodiny Zune.

Programovací jazyk

Aplikace pro Windows Phone 7 je možné vyvíjet ve framework .NET za použití technologií XNA a Silverlight. Microsoft XNA poskytuje řadu nástrojů usnadňujících vývoj her, jejich portování a znovupoužití kódu. Microsoft Silverlight je aplikačním frameworkem uvedeným jako konkurence pro Adobe Flash, tedy pro vývoj komplexních multimediálních internetových aplikací. Spolu s představením Windows Phone 7 byla uvedena specifická verze Silverlight for Windows Phone. Přestože je teoreticky možné použít obě technologie s jakýmkoliv jazykem .NET, tedy kompilovatelným do CLR, jsou oficiálně podporovány pouze jazyky C# a Visual Basic.

Vývojářské nástroje

Balíček Windows Phone SDK Tools je bezplatným rozšířením pro Visual Studio a Visual Studio Express verze 2010 a vyšší. Součástí jsou kromě plnohodnotného emulátoru také např. nástroje pro sledování výkonu aplikace nebo wysiwyg návrhář uživatelského rozhraní.

Bezpečnost

Vývojáři mohou k zvýšení zabezpečení aplikace využít API .NET Compact Framework. Jde například o tyto prostory jmen:

- System.Security.Cryptograhpy - šifrování a dešifrování dat,
- System.Security.Permisions - řízení přístupu k systémovým funkcím a zdrojům,
- System.Security.Principal - hlavní objekt reprezentující bezpečnostní kontext pod kterým kód běží.

Podobně jako oba výše zmíněné konkurenční systémy i Windows Phone 7 vynucuje běh aplikací třetích stran v sandbox. Ačkoliv jeho implementační detaily nejsou dostupné, opět odděluje jednotlivé aplikace od sebe a od prostředků poskytovaných zařízením. Přístup k těmto prostředkům je popsán sadou implicitně nebo explicitně definovaných žádostí o povolení. Uživatel má tedy přehled a možnost případně odepřít všechna povolení, která může aplikace během svého běhu vyžadovat. Aktuálně systém obsahuje 15 skupin možných povolení, což je výrazně méně než v Android OS. Data jsou ukládána do tzv. Izolovaného úložiště (Isolated storage). Jeho velikost se mění dle potřeby aplikace. Připojené externí úložiště (paměťová karta) je naformátováno a "uzamknuto" pomocí 128bitového klíče. Bez nového naformátování nebo znalosti hesla jej pak není možné použít v jiném zařízení, dokonce ani v jiném Windows Phone 7 telefonu.

Distribuce aplikací

Aplikace mohou být distribuovány přímo do zařízení pomocí Windows Phone Marketplace. Před uveřejněním procházejí schvalovacím procesem zaměřeným především na kontrolu porušování pravidel omezujících obsah aplikace, např. zobrazování pornografie či násilí. Za roční poplatek je vývojáři umožněno zveřejňovat své placené aplikace, resp. získá VeriSign certifikát nezbytný pro podpis a následnou instalaci aplikací. Počet bezplatně dostupných aplikací je omezen na 5 pro každý vývojářský účet. Uveřejnění další bezplatné aplikace nad tento limit je zpoplatněno jednorázovým poplatkem. Tvůrci je vyplaceno 70 % z ceny každé prodané aplikace. Vývojář má také šanci zveřejnit aplikaci ve skrytém módu. Ta je pak dostupná pouze přes přímý odkaz na instalaci aplikace.

Hardware

Operační systém Windows Phone 7 je nasazován pouze na mobilních telefonech a není výrobcům zařízení poskytován zdarma. Na zařízení s tímto OS jsou kladeny přesně stanovené minimální požadavky, viz

Tabulka 2 - Seznam minimálních hardwarových požadavků Windows Phone 7.

Takto přísně nastavený systém požadavků na hardware prakticky odstranil problémy s nekompatibilitou aplikací na různých zařízeních. V porovnání s Android OS došlo také ke snížení prodlev mezi uvedením updatů systému jednotlivými výrobci či operátory. Vývojář má pak prakticky jistotu, že uživatelovo zařízení běží na poslední verzi OS.

Menší aktualizace OS a firmware mohou být doručovány over-the-air ovšem větší updaty jsou prováděny výhradně připojením k desktopovým aplikacím Zune software nebo Windows Phone Connector.

Tabulka 2 - Seznam minimálních hardwarových požadavků Windows Phone 7

Součást	Minimální požadavek
Displej	kapacitní, 4bodový multitouch, 480x800
Procesor	ARM v7 – Snapdragon
GPU	podporující DirectX9
Paměť	256MB RAM a nejméně 4GB flash úložiště
Senzory	Akcelerometr, osvětlení, přiblížení a asistované GPS
Rádio	FM
Tlačítka	zpět, Start, vyhledávání, fotoaparát, vypnutí, hlasitost
Volitelné	kamera pro videohovory, kompas, gyroskop

Zdroj: [12]

Aktualizace aplikací probíhá ze zařízení přes Marketplace. Odinstalace je možná přímo v telefonu, bez možnosti obnovit stav nově nainstalované aplikace. Data aplikace nejsou přístupná, pokud k nim sama neposkytne uživatelské rozhraní.

Unlock

Odemčením zařízení získává uživatel možnost nejenom používat telefon v sítích jiného operátora, ale také instalovat, spouštět a debugovat nepodepsaný kód. Zpřístupněna jsou také data v Izolovaném úložišti.

Nahrávání telefonních hovorů

Pro záznam audio nahrávek může vývojář využít třídu Microphone z prostoru jmen Microsoft.Xna.Framework.Audio. Tento prostor jmen také poskytuje řadu dalších tříd, instancí a výjimek pro práci se zvukem. Metody tohoto API mohou být volány i přímo z aplikace, jejíž uživatelské rozhraní je postaveno na framework Silverlight. Podobně jako v případě iOS, jsou aplikacím třetích stran během hovoru odejmuty veškeré audio prostředky.

1.4.4. Ostatní operační systémy

Kromě výše popsaných operačních systémů jsou na trhu také další konkurenční platformy. Z dat uvedených v kapitole 1.2 Zastoupení platforem je možné vyčíst, že dochází k poklesu jejich tržního podílu.

Za nejvýznamnější je možné považovat systém kanadské společnosti Research In Motion, BlackBerry OS, aktuálně ve své sedmé verzi. Tento operační systém cílený na firemní klientelu má řadu bezpečnostních prvků [10], přesto jeho oblíbenost mezi uživateli klesá, zřejmě kvůli zastaralému uživatelskému rozhraní. V následujících měsících se připravuje představení zcela nové verze BlackBerry OS 10 postavené na systému QNX.

Znatelný podíl na trhu získal také WebOS společnosti Palm. Finanční problémy společnosti však vyústili v její odprodej společnosti HP. WebOS je nyní postupně zveřejňován jako open source projekt a nikdo neoznámil záměr vyrábět pro tento operační systém zařízení.

Zařízení s operačními systémy Bada (Samsung) a Symbian S40/S60 (Nokia) se objevují na levnějších, méně vybavených mobilních telefonech a dle vyjádření obou společností nejsou primárně cíleny na náš či západní trh.

1.5. Multiplatformní vývoj

Zastoupení jednotlivých operačních systémů na trhu je relativně vyrovnané. V případě, že není z nějakého důvodu možné nebo vhodné aplikaci realizovat jako webovou službu, je nutné rozhodnout o cílení na jednotlivé operační systémy. Pokud nebude podporována některá

z platforem, dojde ke ztrátě, resp. neoslovení části potenciálních uživatelů. Když bude naopak rozhodnuto podporovat další mobilní operační systém, bude nutné vynaložit větší náklady na oddělený vývoj a následující podporu odlišné verze aplikace.

V některých případech je možné využít jeden z produktů usnadňujících multiplatformní vývoj. Na základě přístupu k řešení tohoto problému, dělíme tyto produkty na dvě základní skupiny.

1.5.1. Webové technologie

První skupina řešení umožňuje vývojářům zobrazovat internetové stránky uvnitř nativní aplikace, využívat plně všech možností html5 (včetně lokálního persistetního úložiště). Přístup k systémovým prostředkům a funkcím je řešen přes javascriptová volání. Tento postup je samozřejmě vhodný pouze pro některé typy aplikací, neumožňuje plně využít hardware telefonu a neposkytuje ani jednotný přístup ke všem funkcím všech operačních systémů.

Mezi tato řešení je možné zařadit:

- PhoneGap,
- appcelerator,
- MoSync HTML5/JavaScript.

1.5.2. Kompilace do nativního kódu

Druhou možností je použití frameworku pro tvorbu jednotného kódu. Ten umožní následnou kompilaci do nativního kódu příslušné platformy. V takto tvořených aplikacích je třeba ošetřovat případy specifické pro jednotlivé operační systémy. Často také není možné vytvářet uživatelské rozhraní fungující napříč zařízeními. Přesto pro některé projekty mohou být tato řešení dostačující.

Do této skupiny je možné zařadit:

- Xamarin Mono,
- rhomobile,
- MoSync C/C++.

1.6. Stručný přehled operačních systémů

V následující tabulce, Tabulka 3 - Přehled operačních systémů, jsou uvedeny některé vlastnosti vybraných operačních systémů pro chytré mobilní telefony. Sledované vlastnosti byly zvoleny dle důležitosti pro tento konkrétní projekt.

Tabulka 3 - Přehled operačních systémů

Vlastnost	Android	iOS	Windows Phone 7
Tržní zastoupení v %	49,2	18,9	10,8
Tvůrce	OHA / Google	Apple	Microsoft
Aktuální verze	4.0.4	7.5	5.1
Rodina OS	Darwin	Unix	Windows CE 7
Architektury	ARM, MIPS, x86	ARM	ARM
Hardware	Různorodý, mnoho výrobců	Pouze Apple	Přesné požadavky
Licence	Free, Open Source	Proprietární, Open Souce komponenty	Proprietární
Oficiální obchod	Google Play	App Store	Marketplace
Správa balíčků aplikací	APK	iTunes	Zune software
FOTA	Ano	Ano	Menší aktualizace
Šifrování	Pomocí aplikací	Ano	Ano
Externí úložiště	Ano	Ne	Uzamknuté
SDK podpora	Unix / Linux, Windows	Mac OS X	Windows
Distribuce aplikací	APK, Google Play a další obchody	Pouze App Store	Pouze Marketplace
Podporované jazyky	Java, C, C++	C, C++, Objective-C	C#, Visual Basic
Nahrávání hovorů	Omezeně podporované	Ne	Ne
Poplatky	Zpoplatněno zveřejnění na Google Play	Ročně za zveřejnění na App Store	Ročně, podpisový certifikát

Zdroj: vlastní zpracování

2. APLIKACE PRO ZÁZNAM TELEFONNÍCH HOVORŮ V ANDROID OS

Pro implementaci modelového řešení aplikace popsané v kapitole 1.3 Aplikace pro nahrávání telefonních hovorů, byl zvolen operační systém Android. Aplikace je naprogramována zcela v jazyce Java a nevyužívá tedy nižší úrovně programování NDK. Vývoj probíhal za použití IDE Eclipse spolu s oficiálními rozšířeními od společnosti Google. Pro verzování kódu byl použit systém Subversion s příslušným serverem. K modelování UML byl použit software Enterprise Architect. Zkušební nasazení probíhalo téměř výhradně na fyzickém zařízení Samsung Galaxy Note s Android OS ve verzi 2.3.5. Dokumentace tříd a metod byla provedena pomocí JavaDoc.

2.1. Požadavky

Před realizací prvního modelu cílové aplikace byl vytvořen seznam funkčních a nefunkčních požadavků. Pomocí nich byl stanoven rámec řešených problémů a další vlastnosti definující či omezující realizaci projektu.

2.1.1. Funkční

- Aplikace bude zaznamenávat hovory.
- Aplikace bude vybírat hovory k zaznamenání dle příslušného filtru.
- Aplikace bude odesílat soubory server.
- Aplikace bude znemožňovat manipulaci se záznamy.
- Aplikace bude umožňovat vzdálenou správu nastavení.

2.1.2. Nefunkční

- Aplikace bude pracovat pod Android OS.
- Aplikace bude úsporně nakládat s prostředky zařízení.

2.2. Návrh architektury

Před samotnou implementací byl vytvořen jednoduchý diagram aktivit (viz Příloha A), popisující hlavní úlohy systému. Na jeho základě byly zpřesněny požadavky (viz Příloha B), vytvořena struktura balíčků aplikace a identifikovány hlavní analytické třídy objektů a jejich interakce (viz Příloha C). Diagram posloužil také k ověření splnění požadavků uvedených v kapitole 2.1 Požadavky. Protože má aplikace nejprve běžet zcela autonomně, identifikovali

jsme pouhé dva aktéry – Android OS a vzdálený server. Veškeré diagramy, včetně těch neotisknutých v této práci, jsou dostupné v příslušné složce Přílohy D.

2.3. Vlastnosti aplikací v Android OS

Jak již bylo zmíněno dříve v této práci, programování pro Android OS se v mnoha ohledech odlišuje od vývoje pro desktopové operační systémy. Tyto odlišnosti samozřejmě neovlivňují jen uživatele, ale kladou i řadu požadavků na strukturu vyvíjené aplikace. Dále jsou uvedeny některé vlastnosti Android OS, které přímo ovlivnily vývoj této aplikace.

2.3.1. Adresářová struktura

Každá aplikace vytvořená pomocí Android SDK obsahuje přesně stanovenou adresářovou strukturu. Ta kromě povinných souborů a složek může obsahovat nepovinné součásti.

Tabulka 4 - Adresářová struktura Android projektu

Adresář	Popis	Povinný
src	Zdrojové kódy aplikace.	Ano
assets	Datové soubory přibalené k aplikaci.	Ne
res	Soubory zdrojů.	Ano
bin	Přeložené binární soubory.	-
gen	Automaticky generované zdrojové kódy.	-

Zdroj: vlastní zpracování

2.3.2. Zdroje (Resources)

Adresář *res/* je řešením Android OS pro oddělené udržování zdrojových kódů a ostatních souborů balíčku aplikace. Struktura a pojmenování adresářů se řídí přísnými pravidly.

Tabulka 5 - Adresářová struktura zdrojů

Adresář	Popis
animator	xml soubory definující animace vlastností
anim	xml soubory definující animace View
color	xml soubory specifikující použité barvy
drawable	soubory kompilovatelné do vykreslovatelných objektů (bitmapy, xml)

layout	xml soubory rozložení uživatelského rozhraní
menu	xml popisující rozložení menu, kontextových nabídek, atd.
raw	soubory přístupné pomocí InputStream
values	xml soubory jednoduchých hodnot – řetězce, integery, barvy
xml	ostatní xml soubory (např. konfigurace globálního vyhledávání)

Zdroj: vlastní zpracování

Ke všem těmto souborům je možné přistupovat z jazyka Java přes třídu *Resources* a objekt *R* obsahující identifikátor zdroje. Na rozdíl od složky *assets/* neumožňuje Android OS vývojáři operovat nad adresářovou strukturou samotnou. Poskytuje ale řadu metod pro práci s uloženými daty – např. načtení řetězce metodou *Context.getString(R.string.názevŘetězce)* nebo otevření raw souboru voláním *Resources.openRawResource(R.raw.názevSouboru)*.

Kvalifikátory

Jeden balíček aplikace může obsahovat několik adresářů zdrojů odlišených definovanými kvalifikátory. Operační systém při spuštění načítá zdroje nejblíže odpovídající aktuální konfiguraci zařízení. Tento mechanizmus umožňuje distribuovat v jednom balíčku aplikace např. všechny jazykové verze a několik rozložení uživatelského rozhraní. Kvalifikátory se uvádějí oddělené pomlčkou po názvu zdroje, tedy ve formátu *názevAdresářeZdroje-kvalifikátor1-kvalifikátor2*. Kvalifikátory jsou rozděleny dle MCC/MNC, jazyku a regionu (fr-rFR), displeje (rozměr, rozlišení, dpi), orientace zařízení, módu docku, přítomnosti a typu vstupních prvků (stylus, klávesnice, pouze dotyk) či podporovaného API atd.

2.3.3. AndroidManifest.xml

Tento soubor je nezbytnou součástí každé aplikace a sloužící primárně k jejímu popisu. Kromě jiného může obsahovat seznam Aktivit (viz níže), služeb, rozhraní pro poskytování služeb či registraci odběru systémových událostí. Část je věnována též výčtu povolení nezbytných pro běh aplikace. Tento soubor je v Eclipse vytvářen automaticky při tvorbě nového Android projektu a je možné jej upravovat přímo nebo pomocí dodávaného editoru.

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

    <uses-sdk android:minSdkVersion="10" />
    <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
    <uses-permission android:name="android.permission.INTERNET"/>


    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" android:debuggable="true">
        <activity
            android:name=".DiplomAActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <receiver android:name=".PhoneStateReceiver">
            <intent-filter>
                <action android:name="android.intent.action.PHONE_STATE"/>
            </intent-filter>
        </receiver>
        <service android:name="RecordingService"></service>
    </application>

</manifest>
```

Obrázek 4 - Ukázka obsahu souboru AndroidManifest.xml

Zdroj: vlastní zpracování

2.3.4. Context

Je rozhraním ke globálním informacím o prostředí aplikace. Umožňuje přístup ke třídám aplikace a jednotlivým prostředkům. Umožňuje také volání operací na úrovni aplikace (např. spuštění aktivity). Obecněji je možné Context použít k zjištění aktuálního stavu aplikace a jejích částí. Context je možné získat voláním *getApplicationContext()*, *getContext()*, *getBaseContext()*. Uvnitř Aktivity může být také získán voláním *this*.

2.3.5. Intent

Intenty jsou mechanizmem Android OS pro předávání dat mezi procesy / aplikacemi. Jde o nejběžnější způsob realizace IPC. Tento objekt obsahuje řetězec definující záměr s jakým je Intent zasílán. Volitelnou součástí mohou být další informace sloužící ke správnému doručení (např. Uri) nebo data (např. Extras - páry klíč hodnota).

Tyto objekty mohou být použity k:

- spuštění Aktivity - pomocí *startActivity()*, např. otevření stránky v internetovém prohlížeči,
- zaslání broadcast - voláním *sendBroadcast()*, *sendStickyBroadcast()* a *sendOrderedBroadcast()*,
- komunikaci se službou - metodami *startService()*, *stopService()*, *bindService()*,
- zpřístupnění dat pomocí ContentProvider - pomocí metod *getContentResolver()* nebo *managedQuery()*, např. získání seznamu kontaktů,
- jako callback.

Pending Intent

Specifickým druhem objektu Intent je Pending Intent. Ten zajišťuje, že požadovaná akce bude provedena za použití identity odesílatele, tedy s jeho identifikátorem procesu a s jeho povoleními. Pro zajištění vyšší úrovně bezpečnosti vyžadují některé operace zaslání právě Pending Intent namísto běžnějšího Intent.

Intent filter

Aktivity, služby a receivery zveřejňují seznam obsluhovaných Intentů pomocí tzv. Intent filterů. Tento seznam je uveden v AndroidManifest.xml v elementu <intent-filter>. Po zaslání Intentu operační systém vyhledá aplikace, které mohou definovaný požadavek obsloužit. Zvolena je pak buď nejvhodnější Aktivita, nebo je uživatel vyzván k výběru požadované Aktivity modálním oknem.

2.3.6. Životní cyklus aplikace

Chápáni aplikace se v Android OS liší od zažitého konceptu aplikačního okna / oken v desktopových operačních systémech. V popředí běží zpravidla pouze jedna aplikace a její pozici může kdykoliv převzít jiná aplikace. Navíc kvůli omezenému množství prostředků může být každý proces nečekaně ukončen. Aplikace také může reagovat na různé systémové události nebo na volání od jiných aplikací.

2.3.7. Aktivity

Aktivity jsou obdobou aplikačního okna v desktopových systémech. Jsou postaveny na myšlence, že každá Aktivita, tedy "obrazovka" aplikace, plní nějakou konkrétní úlohu. V konceptu modulárního OS může být tato Aktivita nahrazena jinou Aktivitou, obsluhující

stejnou úlohu. Uživatel má pak možnost, stisknutím tlačítka zpět, zavřít aktuální Aktivitu a přejít k předchozí. Tento koncept ve spojení s modulárností systému umožňuje např. transparentní přechod z naší aplikace obsahující odkaz na emailovou adresu do Aktivity "odeslání nového emailu" aplikace pro obsluhu emailové komunikace. Jednotlivé spuštěné aplikace a jejich samostatné Aktivity jsou ukládány do zásobníku Aktivit. Pokud systém vyžaduje další prostředky, ukončuje postupně Aktivity od těch nejdéle nepoužitých, tedy z konce zásobníku.

Jelikož Aktivita může být také kdykoliv odstraněna z popředí jinou Aktivitou (např. požadavkem na vyřízení příchozího hovoru), je vývojářům poskytnuta sada metod, pomocí kterých mohou sledovat a ošetřovat aktuální stav svých Aktivit:

- *onCreate* - volána jednou při inicializaci Aktivity,
- *onStart* - volána vždy před zobrazením Aktivity,
- *onResume* - volána když je Aktivita připravena reagovat na vstupy od uživatele,
- *onPause* - volána když je Aktivita přesunuta do pozadí (od tohoto momentu může být Aktivita kdykoliv ukončena),
- *onStop* - volána v momentě, kdy je Aktivita odstraněna z manažeru úloh,
- *onRestart* - volána když se má uživateli Aktivita znovu zobrazit,
- *onDestroy* - volána vždy, když je Aktivita zrušena.

2.3.8. Služby

Je tedy zjevné, že Aktivity nejsou vhodné pro obsluhu dlouhodobě běžících úloh. Obzvlášť v případě, že nevyžadují ze strany uživatele kontinuální interakci. Pro takto zaměřené aplikace, nebo jejich části, poskytuje Android OS mechanizmus takzvaných služeb. Třída nově vytvářené služby je potomkem třídy *Service* a je možné ji vytvořit a zrušit přímo pomocí volání *startService(naseSluzba)* a *stopService(naseSluzba)* nebo navázat její metody asynchronně pomocí objektu realizujícího rozhraní *IBinder*. Služba nemá uživatelské rozhraní a může aktivovat pouze systém upozornění. V případě nedostatku prostředků může OS ukončit i běžící službu. Riziko jejího ukončení je ale nižší než u Aktivity a lze jej navíc ještě snížit vhodným nastavením parametrů při vytváření služby.

2.3.9. Události

Spuštění aplikace může být také iniciováno systémovou událostí. Příkladem těchto událostí může být příchozí hovor nebo boot systému. Vývojář zaregistruje požadavek na odběr události v AndroidManifest.xml a k jejímu ošetření přiřadí příslušnou třídu, která je

potomkem třídy *BroadcastReceiver*. Možná je i dynamická registrace pomocí metody *registerReceiver()*. Při vzniku události je volána metoda *onReceive()*. Takto vytvořený objekt existuje pouze po dobu provádění této metody a není v něm proto možno provádět asynchronní operace.

Kromě událostí vzniklých uvnitř systému může programátor také naplánovat provádění události pomocí aplikace OS nazvané AlarmManager. Událost k provedení musí být zaslána do AlarmManageru jako Pending Intent. Události je možné naplánovat pro opakované provádění dle stanoveného intervalu v milisekundách, nebo pro provedení v přesně stanovaném čase. Takto naplánované události jsou zapomenuty v případě, že dojde k vypnutí / restartu OS.

2.3.10. Povolení

Jak již bylo zmíněno v obecnějším popisu operačního systému v kapitole 1.4.1 Android OS, jsou povolení (Permissions) právy k provádění nějaké operace nebo použití konkrétního prostředku. Povolení vyžadovaná aplikací jsou uvedena v *AndroidManifest.xml*. Z programu je možné k těmto informacím přistupovat několika způsoby, např. voláním *Context.checkPermission()*. Povolení lze rozdělit do dvou skupin.

Vyžadovaná

Uživateli je při instalaci zobrazen seznam vyžadovaných povolení. Ta jsou následně všechna přidělena dokončením instalace nebo odejmuta zrušením instalace. Jednotlivá povolení jsou definována tak, aby byla pro uživatele snadno srozumitelná. Uživateli stačí vědět, že povolení READ_CONTACTS umožňuje aplikaci přistupovat ke kontaktům. Nemusí být informován o tom, že na úrovni aplikace jde o přístup ke konkrétní službě, Content Provideru či vyvolání Binderu.

```
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.RECORD_AUDIO"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.PROCESS_OUTGOING_CALLS"/>
<uses-permission android:name="android.permission.READ_CONTACTS"/>
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
<uses-permission android:name="android.permission.VIBRATE"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.WRITE_CONTACTS"/>
```

Obrázek 5 - Ukázka seznamu povolení aplikace

Zdroj: vlastní zpracování

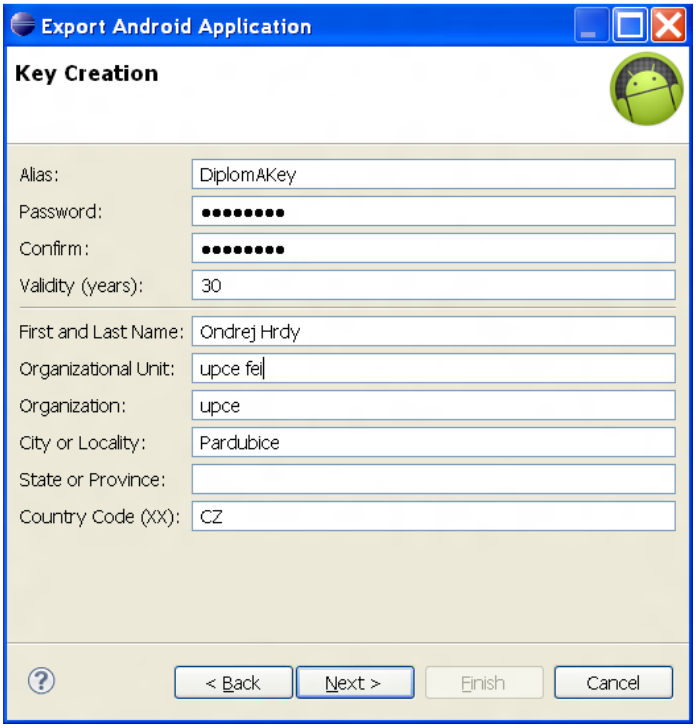
Povolení vyžadovaná aplikaci nemohou být měněna. Z důvodů bezpečnosti se doporučuje při volbě povolení dodržovat princip nejnižších privilegií. Každé povolení je uvedeno v elementu <*uses-permission*> souboru *AndroidManifest*, viz Obrázek 5 - Ukázka seznamu povolení aplikace.

Nabízená

Každá aplikace také může zveřejnit seznam svých specifických povolení. Ta jsou nejčastěji využívána jako rozhraní pro komunikaci s ostatními aplikacemi. Uživatel tak během instalace bude informován o požadavku na přístup k datům naší aplikace. Tato povolení jsou adresována pomocí názvu sestavení a názvu elementu <*permission*>.

2.3.11. Podpis aplikace

Operační systém Android k ověření identity vývojáře vyžaduje, aby každá aplikace byla podepsána. Tento proces lze realizovat pomocí self-signed certifikátu, který může být zdarma vygenerován pomocí nástrojů SDK, viz Obrázek 6 - Generování podpisového klíče. Aplikace podepsané stejným klíčem mohou od OS požadovat běh pod shodným UID. Toto je nezbytné např. ke zpřístupnění původních dat při update aplikace.



Obrázek 6 - Generování podpisového klíče

Zdroj: vlastní zpracování

Tento mechanizmus zajišťuje ověření autorství aplikace, resp. několika aplikací při použití stejného klíče. Neprobíhá však certifikace vývojáře a neinformuje nás tedy o jeho skutečné identitě.

2.3.12. Grafické uživatelské rozhraní

Uživatelské rozhraní aplikace v Android OS může být vytvořeno dvěma způsoby. Jeho vzhled je buď popsán v xml souboru a využívá standardizovaných prvků nebo může být vytvořen pomocí API OpenGL ES. Jelikož je v případě použití OpenGL implementace vzhledu zcela v rukách vývojáře, budeme se nadále zabývat pouze první možností.

Rozložení (Layout)

Jednotlivá uživatelem definovaná rozložení jsou uložena jako xml soubory ve složce *res/layout*. V těchto souborech jsou elementy představující standardní ovládací (např. *Button*, *CheckBox*) a zobrazovací (*ListView*, *TextView*, atd.) prvky rozhraní. Pro popis pozice prvku slouží další skupina elementů reprezentovaná např. *LinearLayout* či *RelativeLayout*. Na všechny tyto prvky je možné aplikovat řadu atributů, které mohou dále specifikovat jejich vzhled, chování a pozici.

Programátor může nastavit Aktivitě, aby použila konkrétní rozložení voláním *Activity.setContentView(idZdroje)*. Má možnost také dynamicky vytvářet a rušit objekty v aktivním layout. Toto je možné provádět přímo nebo i použitím takzvaných adaptérů, které samy převedou instanci konkrétní třídy na její příslušnou vizuální reprezentaci – např. SQLite relaci na skrolovatelný seznam dvouřádkových záznamů.

Některá přednastavená rozložení jsou již součástí SDK a je možné je tak snadno použít v aplikaci. Jde o nejčastější objekty reprezentující např. jedno či dvouřádkový záznam v seznamu či označené a neoznačené položky pro výběrová modální okna.

Prvky

Přednastavené prvky uživatelského rozhraní jsou součástí balíčku *android.widget*. Vývojář má možnost vytvářet vlastní rozšířením třídy *View* a jejích potomků. Tyto prvky se skládají z Java implementace chování, xml souboru definujícího element pro použití v rozloženích a volitelného xml souboru popisujícího rozložení prvku samotného.

K identifikaci prvků se používají jejich jedinečné identifikátory v xml reprezentované jako atributy *android:id*. Ty jsou následně dostupné přes *R.id.názevIdentifikátoru*. Např. tlačítko je možné z jazyka Java navázat voláním:

Button mBtn=(Button)Activity.findViewById(R.id.idTlacitka);

Lokalizace

Vývojář má možnost uložit veškeré texty aplikace do příslušných souborů adresáře *res/values/*. Při založení nového projektu je vytvořen standardně soubor *strings.xml*. Ten obsahuje elementy s atributem identifikátoru *name* a samotnou hodnotu. K těm je pak možné přistupovat programově voláním Context.getString(idZdroje) nebo přímo z jednotlivých rozložení. Texty aplikace se tak mění na základě toho, který soubor hodnot je v aplikaci aktuálně načten. Načtení konkrétního souboru je možné provést přímo nebo ponechat rozhodnutí na operačním systému.

2.4. Implementace

Aplikace je napsána pomocí programovacího jazyka Java. Využívány jsou standardní knihovny tohoto jazyka a také balíček zpřístupňující funkce operačního systému. Dále jsou v této kapitole uvedeny vybrané části řešení. Ty byly zvoleny dle důležitosti v rámci projektu a rozděleny dle logické příslušnosti. Měly by také poskytnout představu o struktuře a fungování představené aplikace.

2.4.1. Balíčky

Aplikace je rozdělena do několika balíčků. Jednotlivé třídy jsou přiděleny do těchto balíčků dle své logické a funkční příslušnosti. Android OS vyžaduje, aby název balíčků byl unikátní v rámci celého systému – doporučováno je doménové pojmenování reflektující název společnosti / osoby vývojáře. Názvy všech balíčků proto začínají řetězcem *com.ondrejhrdy*, který není v následující tabulce uváděn.

Tabulka 6 - Přehled balíčků řešení

Název balíčku	Popis
.audiorecording	obsluha zázamu zvuku a hovoru
.buffer	obsluha zpožděného provádění operací
.callfilter	nastavení filteru, filtrování hovoru
.diploma	hlavní balíček aplikace, Aktivity a receivery
.encryption	šifrování a třída testující šifrovací metody
.fileobserving	monitorování přístupu a změn souborů

.general	nastavení aplikace, databáze, třídy a metody nepřímo související s touto konkrétní aplikací
.internet	komunikace se serverem, upload a download
.other	obecné třídy projektu
.recordedcall	třída zaznamenaného hovoru, práce s balíčkem záznamu

Zdroj: vlastní zpracování

2.4.2. Ukládání dat

K ukládání většiny dat aplikace byly zvoleny soubory. Práce s nimi je relativně snadná a operace nad nimi poskytují prakticky všechny programovací jazyky. To zajišťuje snadnou přenositelnost mezi platformami a případné hromadné zpracování.

Proprietární formát souborů, spolu se šifrováním jejich obsahu, zajišťuje základní úroveň zabezpečení dat. Snadné vytváření kontrolních součtů také umožňuje sledování případných změn souboru. Android OS poskytuje také mechanizmus přístupových práv souborů a nástroje pro sledování změn souborů. Tyto vlastnosti jsou součástí operačního systému Linux a mohou být využity jako další úroveň zabezpečení.

2.4.3. Adresářová struktura

Pro zvýšení přehlednosti jsou jednotlivé datové soubory uloženy v oddělených adresářích. Seznam složek a jejich obsah je uveden v Tabulka 7 - Adresářová struktura aplikace. Pro demonstrativní účely jsou jednotlivé složky pojmenovány dle příslušné funkce. Pro zvýšení bezpečnosti doporučujeme nahradit tyto přehledné názvy adresářů jejich méně přehlednými alternativami, např. náhodnými řetězci.

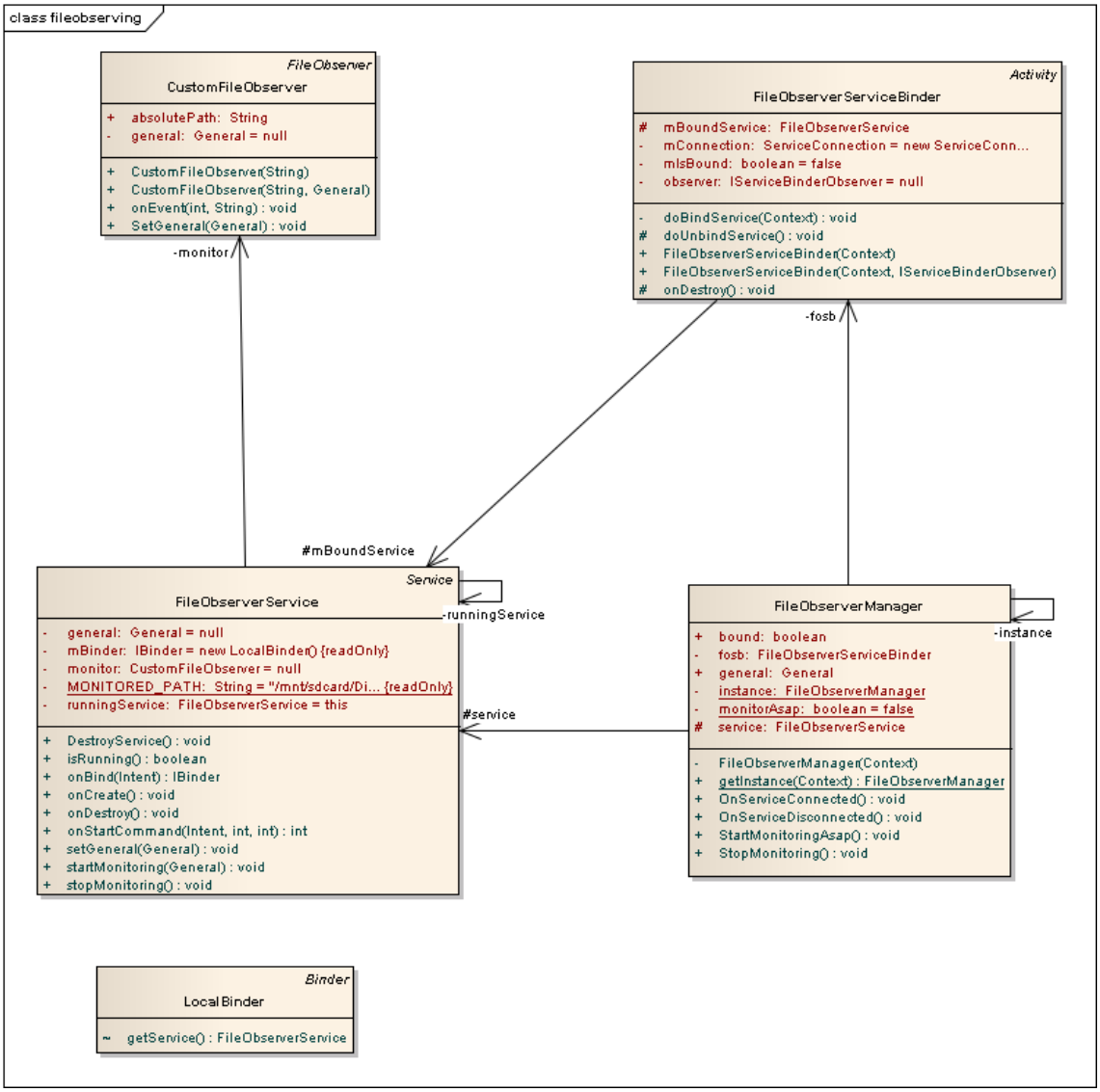
Tabulka 7 - Adresářová struktura aplikace

Adresář	Popis
.	Úložiště pro klíč, aktuální soubor nastavení a filtru.
./Audio/	Nezpracované audiozáznamy.
./Buffer/Upload/Packages/	Serializované objekty UploadBufferItem obsahující balíčky záznamů.
./PackagedCalls/	Soubory balíčků obsahujících záznamy hovorů a metadata.

Zdroj: vlastní zpracování

2.4.4. Sledování souborů

Operační systém Android umožňuje vývojáři monitorovat změny souborového systému pomocí třídy *FileObserver*. Ta je na úrovni Linux realizována nástrojem inotify.



Obrázek 7 - Diagram tříd sledování souborů

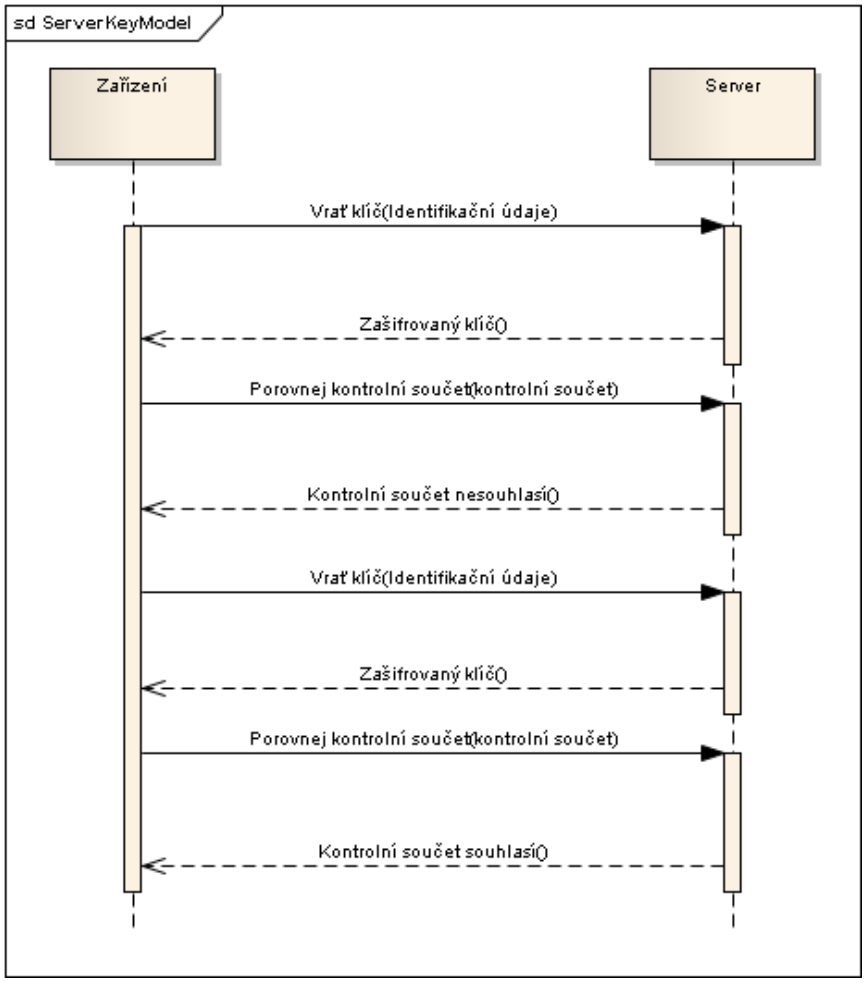
Zdroj: vlastní zpracování

Ten je v tomto projektu použit ke sledování změn nad datovými soubory aplikace. Jednotlivé operace (např. otevření, zapsání, odstranění) jsou logovány do databáze a můžou následně umožnit detekci případného pokusu o manipulaci s nahrávkou.

Jelikož objekt třídy *FileObserver* může být odstraněn garbage collectorem je třeba udržovat v aplikaci referenci na tento objekt po celou dobu běhu operačního systému. Po boot operačního systému proto spouštíme službu *FileObserverService* užívající námi definovaný objekt třídy *CustomFileObserver*, potomka třídy *FileObserver*.

2.4.5. Privátní klíč a veřejný identifikátor

Aplikace pravidelně komunikuje se serverem a přenáší na něj data. Pro zvýšení bezpečnosti byl proto použit mechanizmus založený na kombinaci veřejného identifikátoru a privátního klíče. Obě strany, tedy server i zařízení, znají obě tyto hodnoty, ale veřejně je přenášen pouze veřejný identifikátor.



Obrázek 8 - Získání privátního klíče zařízením

Zdroj: vlastní zpracování

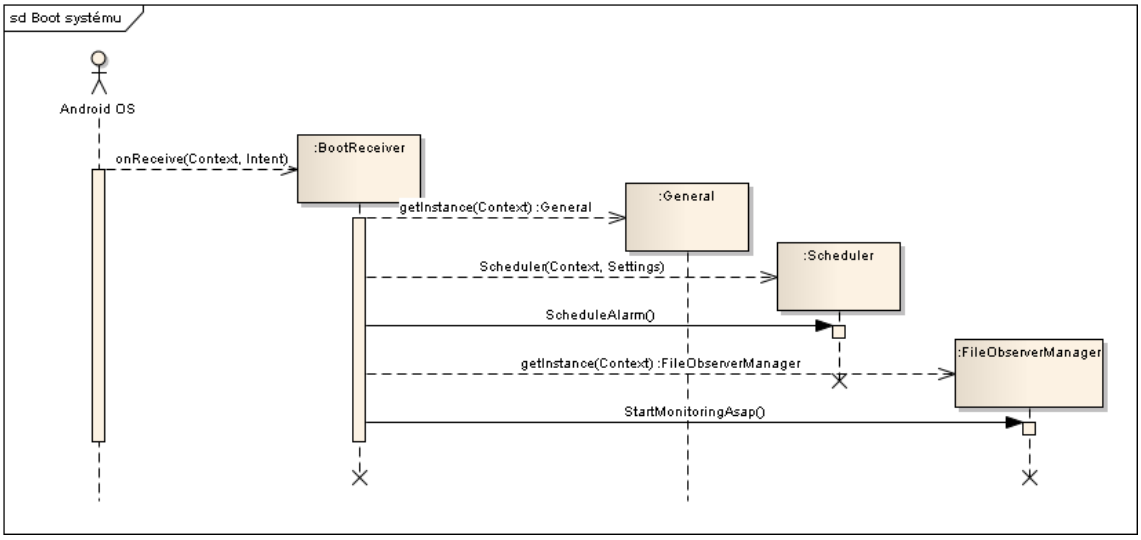
Privátní klíč je použit při šifrování a vytváření otisků pomocí hashovacích funkcí v různých částech programu. Znalost privátního klíče na straně serveru tak umožňuje ověřovat autenticitu příchozích dat.

Privátní klíč je na straně zařízení uložen v souboru uvnitř nepřístupného datového úložiště procesu a aplikace k němu přistupuje pomocí statických metod třídy *PrivateKey*. Tohoto souboru se týkají stejná bezpečností rizika jako u všech ostatních dat aplikace - soubor či aplikace mohou být odstraněny uživatelem. V případě přístupu s právy uživatele root může být dokonce čten či modifikován. Po nové instalaci nebo v případě, že se na zařízení nenachází soubor klíče (byl odstraněn), jsou serveru zaslána data potřebná k vygenerování a zašifrování nového klíče. Ten je pak bezprostředně zaslán zpět a aplikací použit.

Jelikož je tento klíč používán k šifrování není možné jej samotný šifrovat. Vznikl by poté opakující se problém uložení hesla k rozšifrování klíče. Pokud zamítneme možnost použití vnějšího úložiště pro tento klíč (např. ručního zadávání hesla uživatelem) zbývá pouze zvýšení bezpečnosti utajením implementačních detailů aplikace, tzv. security through obscurity. Aby klíč nebyl na zařízení uložen ve své plain-text podobě, je šifrován za použití hesla složeného z IMEI zařízení a UID procesu.

2.4.6. Boot systému

Jednou ze systémových událostí, na které aplikace reaguje je boot systému. Ten je ošetřen námi definovaným objektem třídy *BootReceiver*, který je potomkem třídy *BroadcastReceiver*.



Obrázek 9 - Sekvenční diagram reakce na boot systému

Zdroj: vlastní zpracování

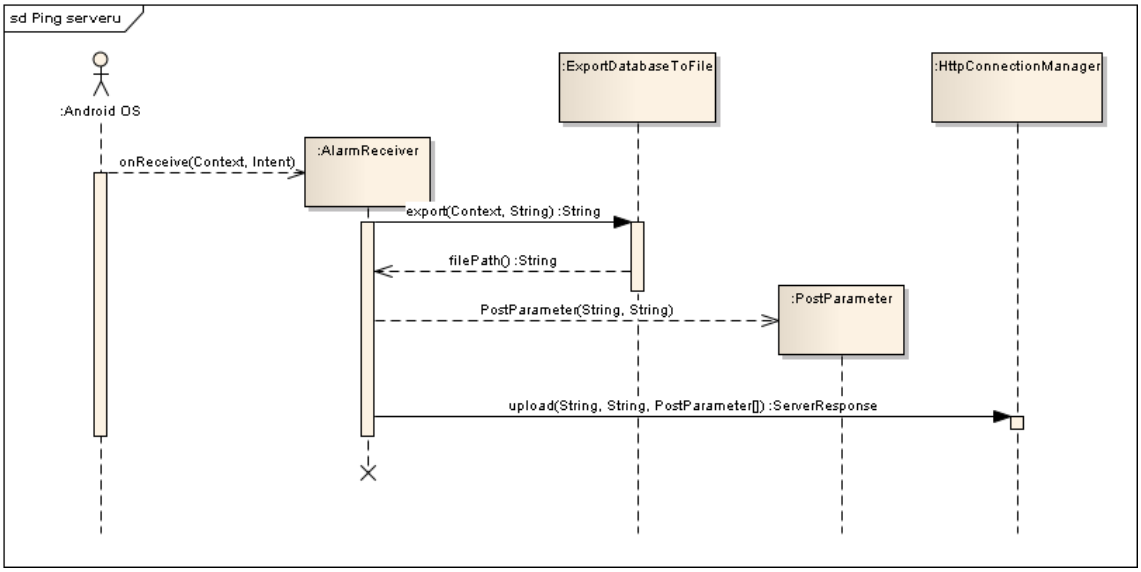
V metodě *onReceive* jsou nejprve načtena nastavení aplikace a boot systému je zalogován do databáze. Následně jsou v systému zaregistrovány naplánované události a je vytvořena služba pro sledování změn souborů.

2.4.7. Plánované události

Operační systém Android poskytuje službu *AlarmManager* k plánování událostí na základě času. Námi definovaná třída *Scheduler* využívá tohoto mechanizmu k naplánování pravidelného pokusu o kontaktování serveru voláním třídy *AlarmReceiver* pomocí objektu Pending Intent.

2.4.8. Komunikace se serverem

V aplikaci byl implementován poll systém komunikace se serverem. Ta je inciována v pravidelných intervalech voláním třídy *AlarmReceiver*, v které je získán singleton objekt *ServerCommunicator* a volána jeho metoda *PingServer()*.



Obrázek 10 - Sekvenční diagram ping serveru

Zdroj: vlastní zpracování

Ta nejprve zjišťuje, zda je povoleno použití internetu pomocí stávajícího připojení. Pokud ano, je na server odeslána záloha aktuálního logu databáze. Následně je zjištěno, zda buffer reprezentovaný třídou *UploadBuffer*, obsahuje nahrávky připravené k zaslání na server. Pokud ano, jsou postupně uploadovány na server.

Po každém přenosu si komunikující strany vymění kontrolní součty přenesených souborů. Tak zjistí, zda proběhl přenos bez chyb. V případě neúspěchu je přenos opakován. Server po

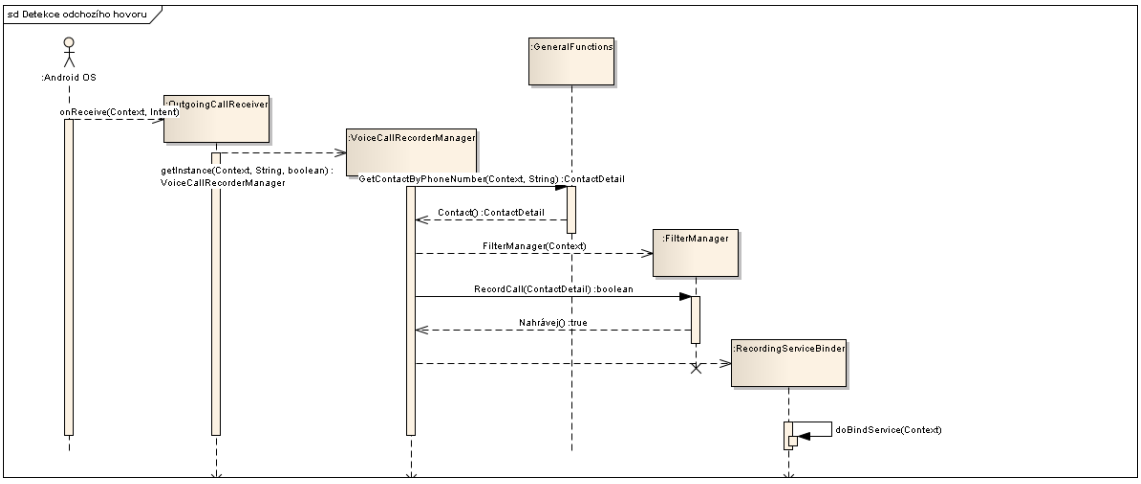
každé operaci zasílá odpověď ve formátu JSON. Ta potvrzuje úspěšnost poslední akce (ukončuje komunikaci), upozorňuje na chybu nebo žádá aplikaci o provedení akce.

Touto akcí může být např. stažení nového šifrovaného souboru nastavení aplikace. Zařízení odpovídá buď informací o úspěchu stažení a ukončením komunikace, nebo zasláním požadavku na novou akci – stažení stejného nebo dalšího souboru (např. nového nastavení filtru hovorů). Cyklus se opakuje, dokud server neoznámí ukončení komunikace. Tento jednoduchý mechanizmus zajišťuje vzdálenou správu aplikace. Jeho nevýhodou je, že komunikaci vždy iniciuje zařízení.

Bezpečnost přenosu samotného je zvýšena šifrováním přenosu. Na straně serveru proto očekáváme existenci důvěryhodného SSL certifikátu pro protokol https.

2.4.9. Detekce hovoru

Mechanizmy detekce příchozího a odchozího hovoru se v Android OS mírně liší. Pro odchozí hovor je nutné registrovat se k odběru události NEW_OUTGOING_CALL a následně sledovat změny stavu modulu telefonního hovoru pomocí odběru události PHONE_STATE. V případě příchozího hovoru stačí sledovat událost změny stavu PHONE_STATE.



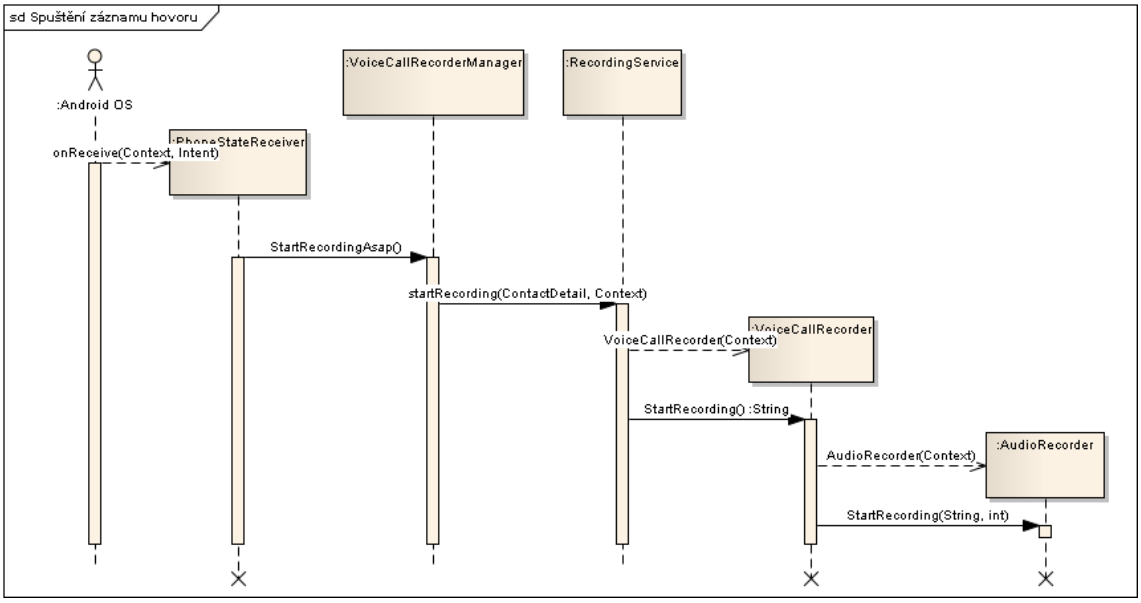
Obrázek 11 - Sekvenční diagram detekce odchozího hovoru

Zdroj: vlastní zpracování

Nový odchozí hovor je registrován třídou *OutgoingCallRecevier*, potomkem třídy *BroadcastReceiver*. Po zavolání metody *onReceive()* získáme telefonní číslo volajícího z parametru typu Intent. Pomocí tohoto čísla naplníme objekt *ContactDetail* informacemi o volajícím. Tento objekt je zaslán do filtru hovorů *FilterManager*. Ten na základě

stanovených pravidel vrátí informaci o tom, zda má být hovor zaznamenáván. Následně je při změně stavu telefonního modulu zahájeno nahrávání hovoru pomocí služby *VoiceCallRecorderService* navázané objektem třídy *VoiceCallRecorderManager*.

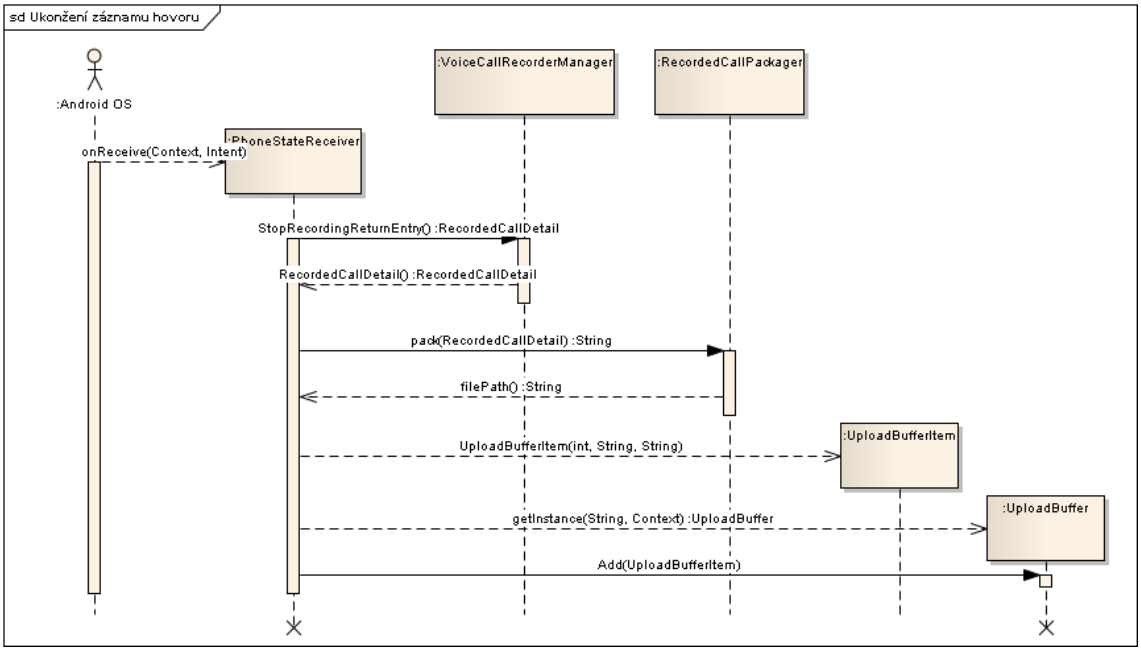
Příchozí hovor je detekován třídou *PhoneStateReceiver*. Opět je vytvořen objekt *ContactDetail* a použit filtr k ověření, zda má být hovor nahráván. Řízení je pak předáno službě *VoiceCallRecorderService*. Běh této služby je následně zastaven v momentě, kdy třída *PhoneStateReceiver* obdrží informace o změně stavu telefonního modulu z probíhajícího hovoru na nečinný.



Obrázek 12 - Sekvenční diagram spuštění záznamu hovoru

Zdroj: vlastní zpracování

Následně je vytvořen objekt třídy *RecordedCallDetail*, který obsahuje kromě odkazu na soubor nahrávky i další metadata. Konkrétně jde o čas začátku a konce nahrávky, velikost souboru nahrávky a objekt třídy *ContactDetail* obsahující informace o druhém účastníku hovoru. Tento objekt je následně zapsán a zašifrován spolu s audio nahrávkou do jednoho souboru voláním statické metody *pack()* třídy *RecordedCallPackager*. Nákladnost této operace roste spolu s délkou záznamu. Je proto prováděna v odděleném vláknu tak, aby zbytek aplikace nepřestal reagovat na příchozí události. Tento soubor je poté připraven k odeslání na server vytvořením objektu *UploadBufferItem* a jeho přidáním do třídy objektu *UploadBuffer*.



Obrázek 13 - Sekvenční diagram ukončení záznamu hovoru

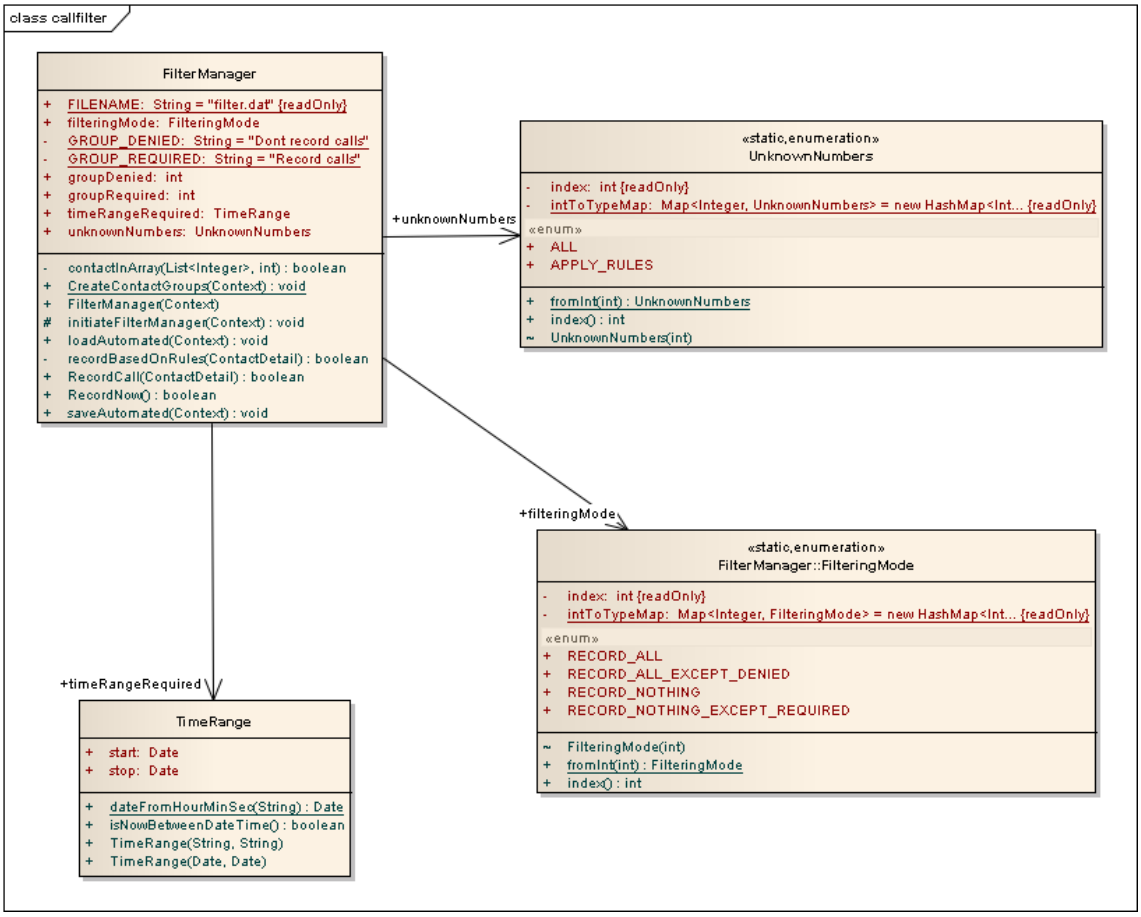
Zdroj: vlastní zpracování

2.4.10. Filtr hovorů

Pro zjištění zda má být příchozí či odchozí hovor nahráván je použita třída *FilterManager*. Uživateli je zpřístupněno několik typů parametrů, které umožňují detailní nastavení výběru hovorů k záznamu. Základní volbou je mód filtrování reprezentovaný výčtovým typem *FilteringMode*. Uživatel má možnost zvolit nahrávání všech hovorů, žádného hovoru a alternativy těchto dvou možností rozšířené resp. zpřísněné o zbylé volby filtrování. Výčtový typ *UnknownNumbers* umožňuje uživateli zvolit, zda se mají nahrávat všechny hovory s neznámými telefonními čísly či zda se má rozhodovat dle ostatních parametrů filtrování.

Při instalaci aplikace jsou vytvořeny v telefonu také dvě skupiny uživatelů. První pro kontakty, které mají být vždy nahrávány, a druhá pro kontakty, které nemají být nikdy nahrávány. Přidáním a odebráním záznamů z těchto skupin je možné detailně regulovat nastavení filtru.

Poslední možností filtru je definice časového rozsahu třídy *TimeRange*. Ten rozdělí den na čas, kdy se mají, resp. nemají hovory zaznamenávat.



Obrázek 14 - Diagram tříd filtr hovorů

Zdroj: vlastní zpracování

2.4.11. Audio nahrávky

K pořizování audio nahrávek poskytuje operační systém android dvojici API. *AudioRecord* umožňuje přímé čtení zvukových dat z audio bufferu představovaného polem bytů. Vývojář je nucen postarat se o průběžný zápis do cílového souboru. Data jsou v nekomprimovaném PCM s velikostí vzorku 8 či 16 bitů. Dokumentace uvádí, že všechna zařízení by měla podporovat vzorkovací frekvenci 44100 Hz.

Druhým API je *MediaRecorder*. Tato třída provádí záznam přímo do souboru. Není proto možné pracovat s daty samotnými během probíhajícího nahrávání. Struktura výstupního souboru je určena parametry *AudioEncoder* a *OutputFormat*.

Záznamová metoda byla zvolena pomocí opakovaného testu vytvoření 60ti sekundové nahrávky, viz Tabulka 8 - Porovnání záznamových metod. Jednotlivé parametry byly nastaveny dle své vhodnosti pro záznam hlasové komunikace. Využití CPU bylo monitorováno nástrojem *top*. Na základě těchto zjištění bylo rozhodnuto o použití

MediaRecorder API a kodeku AMR Wideband, který byl vyvinut společností Nokia právě pro kvalitní digitální reprezentaci hlasové komunikace.

Tabulka 8 - Porovnání záznamových metod

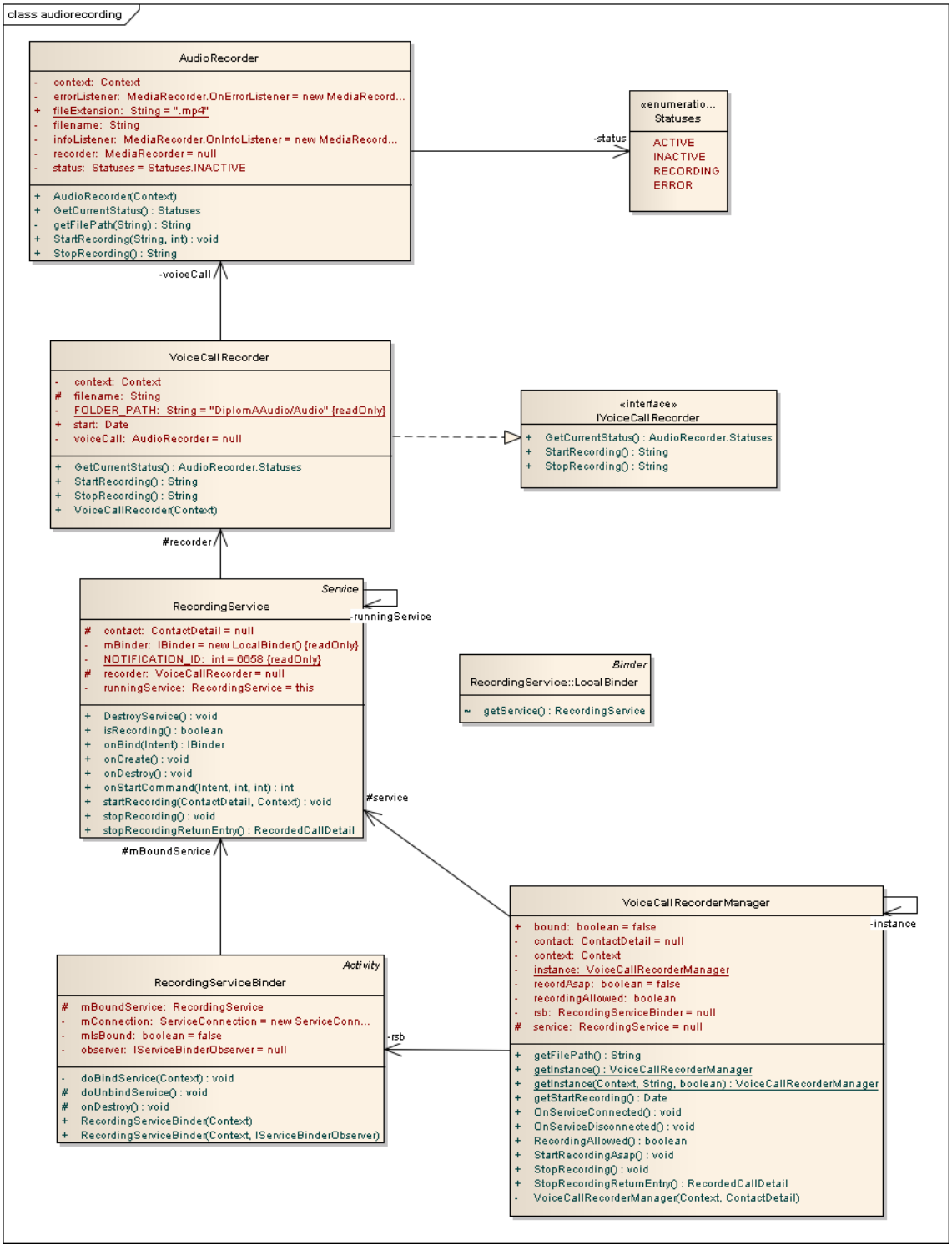
API	Nastavení	Velikost v kB	Využití CPU celkem	Využití CPU aplikací
bez nahrávání	-	-	13 %	0 %
AudioRecord	16bit, 44,1 kHz	10076	18 %	3 %
MediaRecorder	AMR WB	95	20 %	1 %

Zdroj: vlastní zpracování

Pro obsluhu samotného nahrávání byla vytvořena třída *AudioRecorder*. Ta kromě objektu *MediaRecorder* poskytuje aktuální stav nahrávání, absolutní cestu k vytvářenému souboru a formát souboru nahrávky.

Samotná služba nahrávající hovory, *RecordingService*, však pracuje pouze s objektem realizujícím rozhraní *IVoiceCallRecorder*. To umožňovalo během vývoje snadnou záměnu různých tříd poskytujících nahrávací funkcionalitu.

Použitá třída *VoiceCallRecorder* proto realizuje rozhraní *IVoiceCallRecorder*. Obsahuje objekt třídy *AudioRecorder*, informaci o času začátku nahrávání a poskytuje metody nezbytné pro pořízení záznamu samotného.



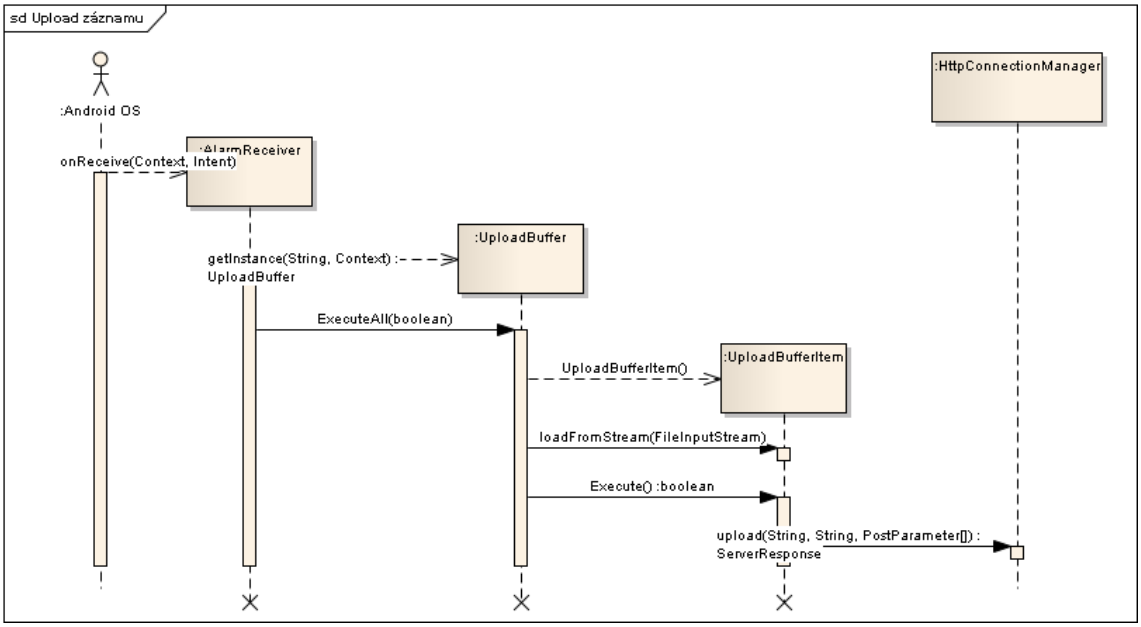
Obrázek 15 - Diagram tříd záznam hovorů

Zdroj: vlastní zpracování

2.4.12. Upload nahrávek

Při dokončení nahrávání hovoru je voláním statické metody *RecordedCallPackager.pack()* vytvořen soubor balíčku obsahující nahrávku a všechny příslušná data. Součástí tohoto balíčku jsou i kontrolní součty, které je možné po rozbalení použít k detekci případných změn souboru.

Následně je vytvořen objekt třídy *UploadBufferItem*, který je serializován do příslušné složky a obsahuje informaci o požadavku na upload balíčku nahrávky. Aplikace vždy po dokončení nahrávky nebo v pravidelných intervalech vytvoří objekt třídy *UploadBuffer* a volá jeho metodu *ExecuteAll()*. V případě, že je povoleno připojení k internetu, třída postupně deserializuje jednotlivé záznamy *UploadBufferItem* a volá jejich metodu *Execute()*, která provede odeslání balíčku audio-záznamu samotného.



Obrázek 16 - Sekvenční diagram upload záznamu

Zdroj: vlastní zpracování

Třídy *UploadBuffer* a *UploadBufferItem* jsou potomky tříd *Buffer*, resp. *BufferItem*. Ty poskytují základní logiku pro řešení zpožděného provádění operací s využitím objektů, obsahujících popis konkrétní operace, serializovaných do souborů.

2.4.13. Šifrování

K šifrování souborů obsahujících nahrávku a metadata byl použit symetrický 128 bit algoritmus AES z balíčku *javax.crypto*. Použití symetrického algoritmu dává uživateli možnost přehrát nahrávku na zařízení, aniž by musela být ukládána její šifrovaná

i nešifrovaná verze. Bez znalosti klíče jsou takto zašifrovaná data v rozumném čase nezískatelná. Z důvodu omezeného množství prostředků bylo provedeno měření ukazující výpočetní náročnost operace šifrování při použití tohoto algoritmu.

Pro zjištění nákladnosti šifrování souboru byla na zařízení provedena série testů. Vytvořen byl desetiminutový záznam hovoru obsahující všechna příslušná metadata. Ten byl následně na zařízení stokrát zašifrován a rozšifrován. Jako alternativa k porovnání bylo navrhnuto zašifrování pouze prvního kilobytu dat. Ten obsahuje metadata hovoru a hlavičku se začátkem záznamu samotného. Tento přístup by případnému útočníkovi značně ztížil identifikaci dat. Textové soubory obsahující hodnoty naměřené při jednotlivých pokusech jsou součástí přílohy této práce.

Tabulka 9 - Test nákladnosti šifrování (v ms)

Operace	Celý soubor	Začátek souboru
Zašifrování průměr	443	21
Rozšifrování průměr	517	32

Zdroj: vlastní zpracování

Rozdíl obou přístupů je významný. V případě šifrování celého souboru navíc poroste nákladnost této operace spolu s délkou záznamu. Po přihlédnutí k celkovému množství prostředků zařízení, bylo rozhodnuto použít šifrování celých souborů. Velikost šifrovaného souboru roste proti nešifrovanému o cca 10 %.

2.4.14. Záznam událostí

Důležité události vzniklé za běhu aplikace jsou ukládány do SQLite databáze, relace LOG. Za důležitou událost je považován vznik chyb, přístup k souborům (viz 2.4.4 Sledování souborů), vznik systémové události, vytvoření balíčku záznamu či úspěšný download souboru ze serveru. Všechny druhy událostí jsou vyjmenovány ve výčtovém typu *LogEvents*. V každém záznamu je kromě typu události vytvořen také otisk složený z vybraných hodnot řádku. Toto umožňuje zpětně sledovat běh aplikace a detekovat případný pokus o útok. Pokud útočník nezná způsob, jakým je vytvářen otisk záznamu, může být rozpoznána i případná manipulace s databázovými daty. Atribut DATA1 slouží k ukládání informací upřesňujících záznam – např. při vzniku výjimky je to její text a při vytvoření souboru jeho kontrolní součet.

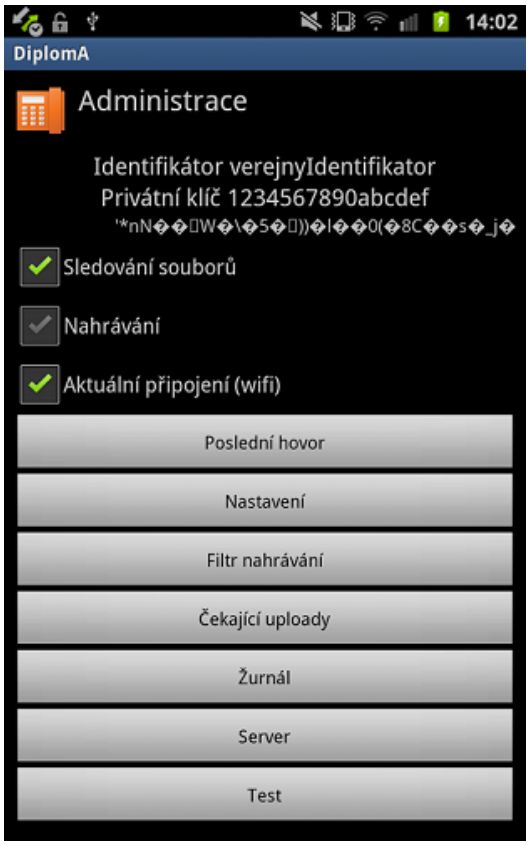
Tabulka 10 - Struktura relace LOG

Atribut	Datový typ	Poznámka
_id	integer	PK, autoincrement
CREATED	integer	čas vytvoření záznamu
EVENT_ID	integer	LogEvents index
EVENT_NAME	text	název události
DATA1	text	doplňková data
HASH	text	otisk řádku

Zdroj: vlastní zpracování

2.4.15. Uživatelské rozhraní administrátora

Aplikace obsahuje několik Aktivit. Ty poskytují rychlý přehled informující o aktuálním stavu běhu aplikace a jejích částí. Umožňují také rychlý přístup k hodnotám jednotlivých nastavení. Tyto informace jsou vhodné spíše pro prezentaci funkčnosti aplikace a její ladění.



Obrázek 17 - Ukázka uživatelského rozhraní aplikace

Zdroj: vlastní zpracování

2.4.16. Zkušební nasazení

Aplikace byla zkušebně nasazena v emulátoru a následně na fyzickém zařízení Samsung Galaxy Note s operačním systémem Android 2.3.5. Aplikace byla v reálném prostředí po dobu delší než jeden měsíc.

Nainstalována byla i na zařízení HTC Wildfire S na Android OS verze 2.3.5. V momentě začátku nahrávání byla vyvolána výjimka a záznam nebyl proveden. Tímto byla potvrzena nekonzistentnost implementace API pro tvorbu audio záznamů hovorů.

2.5. Server

Cílem této práce nebylo vypracování serverové strany řešení. Pro účely testování a případné demonstrace funkčnosti bylo vytvořeno několik jednoduchých php skriptů, které jsou schopny obsloužit veškerou komunikaci s aplikací nainstalovanou v zařízení. V případě nasazení serveru podporujícího Java je možné použít podstatnou část zdrojových kódů aplikace. Především tedy algoritmy deserializace balíčků záznamů, vytvoření kontrolních součtů souborů či dešifrování záznamů.

2.6. Nevýhody stávajícího řešení

Během vývoje této vzorové aplikace bylo odhaleno několik problémů. Tyto problémy by měly být brány v potaz při rozhodování o reálném nasazení této aplikace. Případné návrhy na řešení či snížení dopadu problému jsou uvedeny v jednotlivých kapitolách.

2.6.1. Záznamová API

Za největší problém je jistě možné považovat nekonzistentnost implementace *MediaRecorder* a *AudioRecord* API. Během podrobného studia problému se ukázalo, že některé verze Android OS nepodporují toto rozhraní, resp. obsahují chybu znemožňující použití tohoto prostředku k pořízení právě záznamu telefonního hovoru.

Navíc bylo zjištěno, že stávající hardwarová architektura odděluje subsystém obsluhující telefonní hovory od subsystému, na kterém pracuje operační systém. Správná funkčnost nahrávání hovorů proto záleží na rozsahu implementace výrobcem ve firmware zařízení. Dle vyjádření zaměstnanců společnosti Google, ve veřejně dostupném bug reportu, nemá řešení tohoto problému vysokou prioritu. Není proto možné s jistotou říci, že nová zařízení a verze firmware budou tato API plně podporovat.

Návrh řešení

V případě, že by tato aplikace měla práva uživatele root, bylo by možné kopírovat na úrovni operačního systém Linux obsah sdílené paměti a obdržet tak přímo data předávaná mezi telefonním subsystémem, OS, mikrofonem a reproduktorem. Takto vytvořená aplikace by však byla funkční pouze na telefonech běžících s právy uživatele root.

2.6.2. Úložiště

Dalším potenciálním zdrojem problémů je úložiště. Vnitřní, bezpečné úložiště procesu v operačním systému Android má omezenou kapacitu, o kterou se musí dělit s ostatními aplikacemi.

Externí úložiště je uživateli přímo přístupné, poskytuje tedy nižší úroveň zabezpečení. Toto externí úložiště je také často realizováno jako slot pro paměťové karty. Ty mohou být kdykoliv odpojeny a naše data tak budou ztracena.

Velikost pořízených nahrávek uložených v jeden moment na zařízení se také může významně lišit na základě několika podmínek - nastavení filtru nahrávání, počtu a délky záznamů a době, po kterou nebyly záznamy odeslány na server. Navíc operační systém neposkytuje žádný mechanizmus pro rezervaci volné kapacity paměťového úložiště.

Návrh řešení

Snížit rizika související s velikostí paměťového úložiště by bylo možné informováním uživatele v případě poklesu volné paměti pod stanovenou hranici. Bylo by také možné implementovat vlastní řešení zajišťující rezervaci paměťového prostoru, např. vytvořením souboru s náhodným obsahem o stanovené velikosti, který by byl zvětšován a zmenšován tak, jak by se měnila velikost a počet datových souborů naší aplikace.

2.6.3. Uživatelské pravomoci

Operační systém Android umožňuje uživateli kdykoliv odstranit nainstalované aplikace. V případě, že se aplikace snaží zabránit svému odinstalování, může uživatel spustit telefon v bezpečném módu. V něm jsou spuštěny jenom nejdůležitější systémové funkce a uživatel pak může aplikaci odstranit.

Uživatel má také možnost pomocí správce aplikací odstranit všechna data, která aplikace od svého nainstalování vytvořila ve vnitřním úložišti procesu. Je tak obnoven stav bezprostředně po nainstalování aplikace.

Návrh řešení

Výjimkou jsou systémové aplikace v adresáři */system/app*. Tyto aplikace jsou však součástí firmware a není je možné do telefonu dostat běžným postupem.

2.6.4. Root

Řada uživatelů také využívá tzv. "root" telefonu. Ten umožňuje pracovat s právy uživatele root v operačním systému Linux, jehož je Android OS nadstavbou. Mimo jiné tak uživatel získá přístup k našim datovým souborům, souborům aplikace a souboru SQLite databáze. Tento stav představuje bezpečnostní riziko a zvyšuje šanci na případnou manipulaci s nahrávkou.

2.6.5. Kompatibilita s ostatními platformami

Tato aplikace je napsána v jazyce Java. Bohužel tento jazyk není přímo podporován ani jednou z dalších rostoucích mobilních platforem. Tento fakt znemožňuje snadné nasazení této aplikace na jiném operačním systému pro mobilní telefony.

Návrh řešení

Možným řešením by byl převod co největší části aplikace do programovacího jazyka podporovaného širší škálou zařízení. To by snížilo náklady na vývoj odlišných aplikací pro jednotlivé operační systémy pro mobilní telefony. Jediným možným kandidátem se zdá být jazyk C, resp. C++. Ten je podporován přímo na Android OS, může být použit v iOS a společnost Microsoft oznámila jeho přímou podporu v jednom z nadcházejících updatů Windwos Phone 7.

2.6.6. Řízení prostředků

Všechny zkoumané operační systémy mají možnost v případě nedostatku prostředků vynuceně ukončovat aplikace třetích stran. Přestože námi navržené řešení aplikace se snaží tomuto jevu zabránit použitím služby s vysokou prioritou běhu, může v mezní situaci dojít k ukončení běhu aplikace přímo během nahrávání hovoru.

Návrh řešení

Jediné aplikace, které nemohou být vynuceně ukončeny operačním systémem Android, jsou aplikace systémové. Zabránit ukončování je tedy možné pouze v případě vytvoření vlastní větve distribuce tohoto operačního systému.

2.6.7. Poll komunikace se serverem

Stávající řešení se snaží kontaktovat server v přesně stanovených intervalech. Nereaguje tedy dynamicky na aktuální situaci. Někdy může být server kontaktován, aniž by vznikly v telefonu nové události. Naopak v jiném případě mohou nahromaděné záznamy zbytečně čekat na odeslání do konce poll intervalu. Podobný problém nastává v případě nutnosti bezodkladně iniciovat akci ze strany serveru. Zařízení také může být v momentě, kdy se aplikace pokouší server kontaktovat, mimo dosah vyžadovaného typu připojení.

Návrh řešení

Aplikaci by bylo vhodné rozšířit o několik dalších prvků. Jedním by měl být mechanizmus push upozornění Android OS, který však nezaručuje doručení zpráv. Druhým je registrace k odběru události změny připojení k síti. To umožní okamžitě detekovat síť, v níž je povolena komunikace se serverem. Proškolený uživatel by měl mít možnost iniciovat komunikaci sám přímo z Aktivity aplikace.

ZÁVĚR

Cílem diplomové práce bylo vytvoření aplikace pro bezpečný záznam telefonních hovorů. Během implementace samotné bylo identifikováno několik potenciálních problémů, jejichž zdrojem je nejen mechanizmus fungování operačního systému Android, ale i rozdílný rozsah implementace některých API u jednotlivých výrobců zařízení.

Aplikace, tak jak byla popsána v kapitole 2.1 Požadavky, je v mnoha ohledech pod Android OS realizovatelná. Vzhledem k odlišné implementaci *MediaRecorder* i *AudioRecord* API v jednotlivých verzích operačního systému i na jednotlivých zařízeních, však nemůže být zaručena plná funkčnost této aplikace na všech chytrých telefonech s tímto operačním systémem.

Uživateli, resp. případnému útočníkovi, může být zabráněno, nebo významně ztíženo, editovat nahrávky použitím bezpečného úložiště aplikace, vytvářením kontrolních součtů a šifrováním dat. Za běžných podmínek je nemožné ochránit data aplikace či aplikaci samotnou. Uživatel má možnost ji kdykoliv odinstalovat, nebo snadno odstranit všechna data i ze zabezpečeného úložiště. Tyto zásahy je však možné detekovat a mít tedy alespoň informaci o pokusu o útok.

Použití takto vytvořené aplikace je proto vhodné pouze v prostředí, kde bude pořizování záznamů hovorů v zájmu uživatele. Poněkud jednodušší by také měla být situace v případě nasazení uvnitř kontrolované skupiny uživatelů, tedy např. ve společnosti, kde by zaměstnanci používali jeden konkrétní otestovaný model zařízení a operačního systému. I v tomto případě je však nutné brát zřetel na právní úpravu týkající se záznamu hovorů a stále rostoucí oblibu VOIP volání, jehož použití není možné v systému běžným způsobem detekovat a tedy ani automaticky vytvářet záznamy.

Další případný vývoj aplikace by měl směřovat k implementaci řešení problémů uvedených v kapitole 2.6 Nevýhody stávajícího řešení. Bylo by také vhodné definovat přesně cílovou skupinu zařízení, pro které bude vývoj prováděn.

Operační systémy Windows Phone 7 a iOS aktuálně neumožňují záznam telefonních hovorů přímo na zařízení. Protože je tato aplikace napsána v jazyce Java, není ani možné ji, či některou její část, na těchto dvou platformách nasadit. V případě potřeby multiplatformního použití a snížení nákladů na vývoj je proto doporučen převod co největší části aplikace na jazyk podporovaný cílovými systémy - v tomto konkrétním případě zřejmě C/C++.

POUŽITÁ LITERATURA

[1] FALAKI, H., R. MAHAJAN, S. KANDULA, D. LYMBEROPOULOS, R. GOVINDAN a D. ESTRIN. Diversity in smartphone usage. Proceedings of the 8th international conference on Mobile systems, applications, and services. 2010. Dostupné z: http://enl.usc.edu/papers/cache/Falaki10.pdf

[2] ANDERSON, P. Mobile and PDA Technologies: Looking around the corner. JISC Technology & Standards Watch. 2005. Dostupné z: http://www.jisc.ac.uk/media/documents/techwatch/jisctsw_05_04pdf.pdf

[3] GOWDA, S.R., A. LEELA MOHANA REDDY, X. ZHAN a P.M. AJAYAN. Building energy storage device on a single nanowire. Nano letters. Washington: American Chemical Society, 2011. ISSN 1530-6984.

[4] SCHALKWYK, J., D. BEEFERMAN, F. BEAUFAYS, B. BYRNE, C. CHELBA, M. COHEN, M. KAMVAR a B. STROPE. Google Search by Voice: A case study. Springer, 2010. Dostupné z: http://research.google.com/pubs/archive/36340.pdf

[5] ROSE, S., D. POTTER a M. NEWCOMBE. Augmented Reality: A Review of available Augmented Reality packages and evaluation of their potential use in an educational context. 2010. Dostupné z: http://blogs.exeter.ac.uk/augmentedreality/files/2010/11/Augmented-Reality-final.pdf

[6] COMSCORE. 202 Mobile future in focus: Key insights from 2011 and chat Theky mean for the coming year. 2012, 49 s.

[7] GARTNER GROUP. Forecast Analysis: Mobile Devices, Worldwide, 2008--2015, 1Q11 Update (2011).

[8] *Právní úprava odposlechů a kryptoanalýzy*. Brno, 2008. Diplomová práce. Právnická fakulta Masarykovy univerzity v Brně.

[9] GOOGLE INC. *Android Developers* [online]. 2012 [cit. 2012-04-27]. Dostupné z: http://developer.android.com/index.html

[10] DWIVEDI, Himanshu, Chris CLARK a David V THIEL. Mobile application security. New York: McGraw-Hill, c2010, 408 s. ISBN 00-716-3356-1.

[11] APPLE INC. *IOS Developer Library* [online]. 2010 [cit. 2012-04-20]. Dostupné z: http://developer.apple.com/library/ios/navigation/

[12] TRIF, Silvia a Adrian VISOIU. A Windows Phone 7 Oriented Secure Architecture for Business Intelligence Mobile Applications. Informatica Economica. 2011, č. 15, s. 11.

[13] SARAH ALLEN, Vidal Graupera. Pro smartphone cross-platform development: iPhone, BlackBerry, Windows Mobile, and Android development and distribution. New Edition. New York, N.Y.:Apress, 2010. ISBN 978-143-0228-684.

[14] YEE AU, Kathy Wain, YiFan ZHOU, Zhen HUANG, Phillip GILL a David LIE. Short Paper: A Look at Smart Phone Permission Models. In: Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices. Chicago, Illinois, USA: ACM, 2011, s. 5.

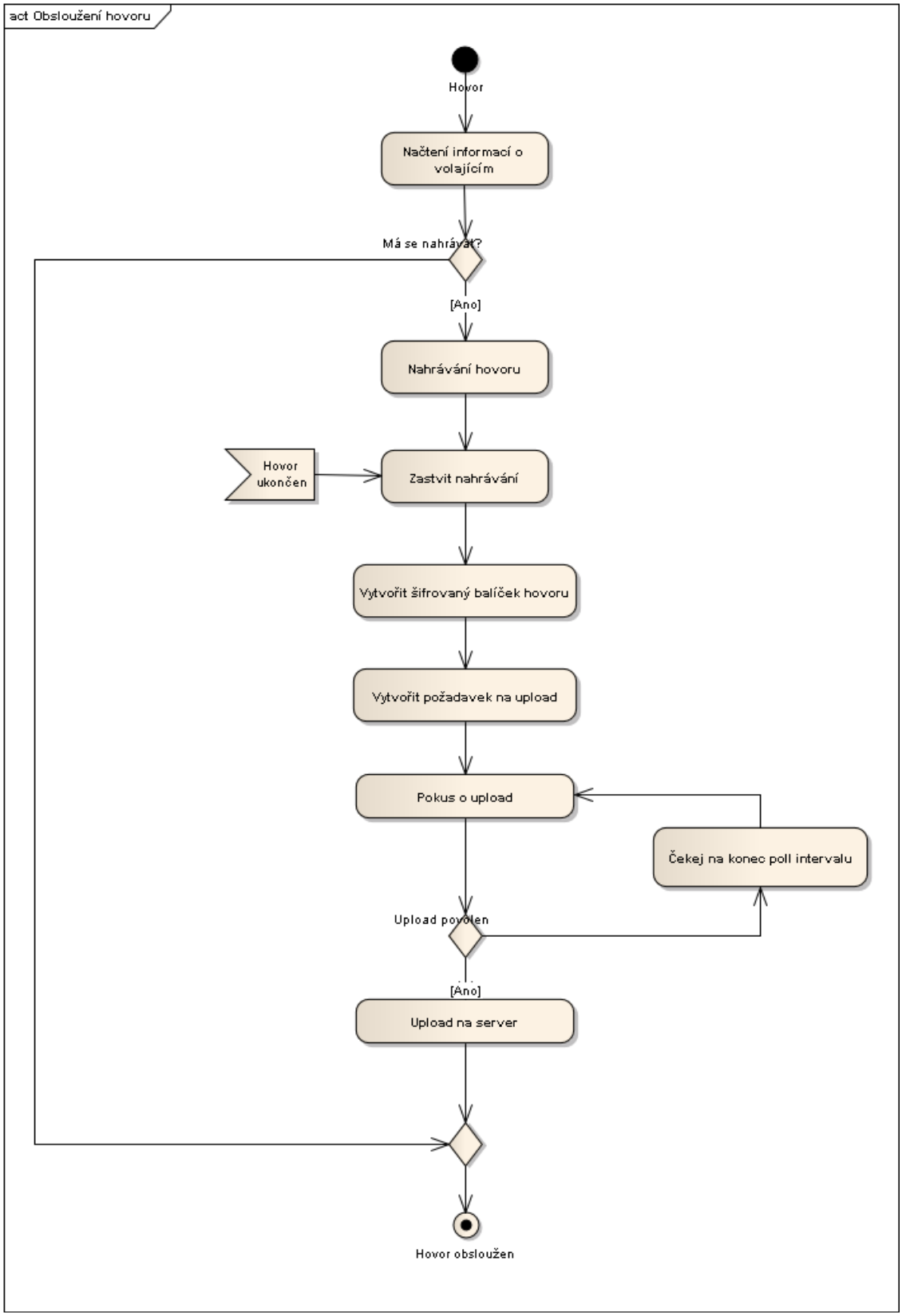
[15] SÁZEL, Vojtěch. Vývoj aplikací na platformě Android. Praha, 2010. Bakalářská práce. Unicorn College.

[16] MICROSOFT. *Phone – platform development* [online]. 2012 [cit. 2012-04-22]. Dostupné z: http://msdn.microsoft.com/en-us/ff380145.aspx

[17] DOROKHOVA, R. a N. AMELICHEV. Comparison of Modern Mobile Platforms from the Developer Standpoint. Screen. 2009, č. 3, s. 5. Dostupné z: http://osll.spb.ru/attachments/download/429/evaluation_2.doc

SEZNAM PŘÍLOH

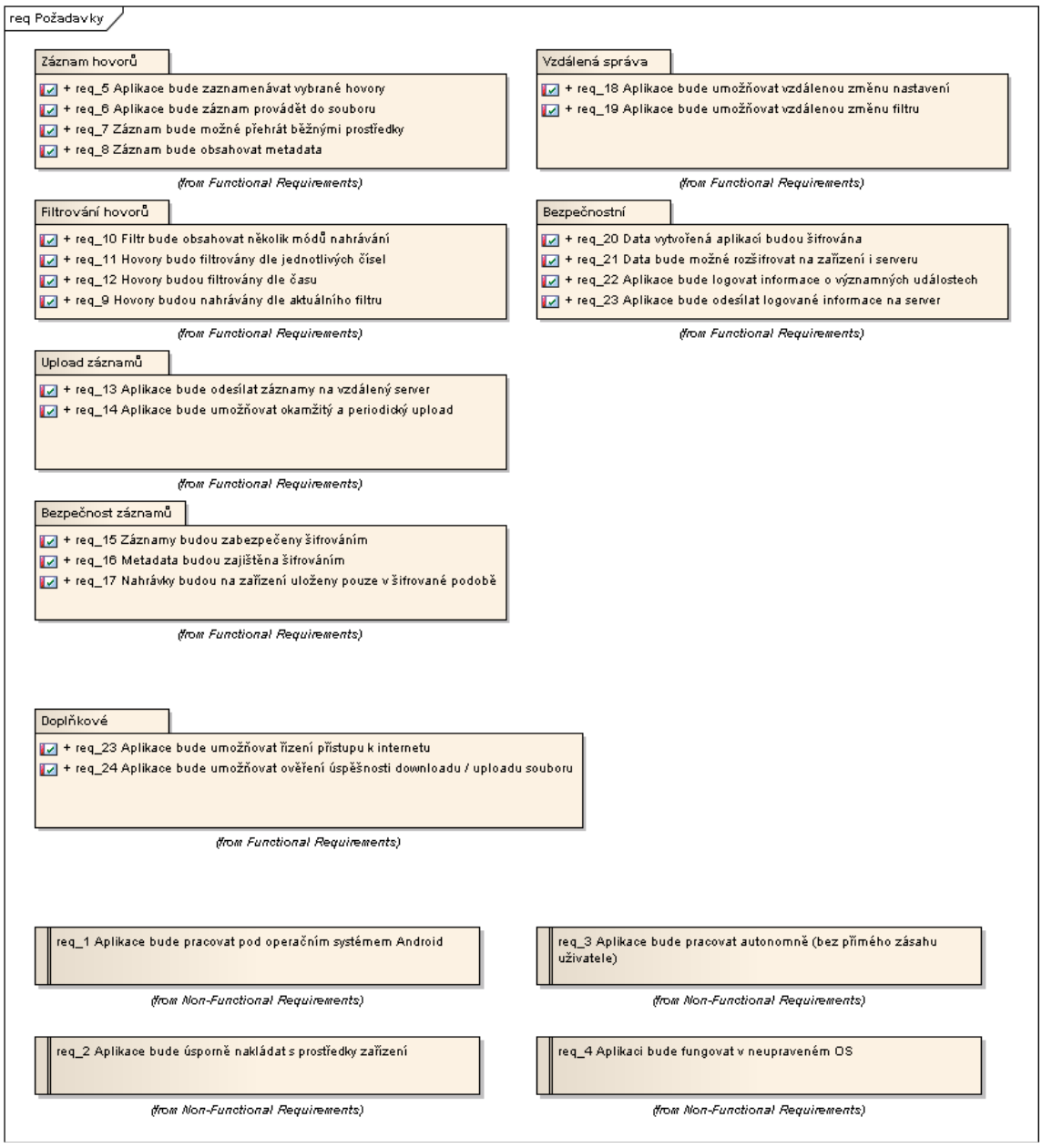
Příloha A – Diagram aktivity



Obrázek A. 1 - Diagram aktivit

Zdroj: vlastní zpracování

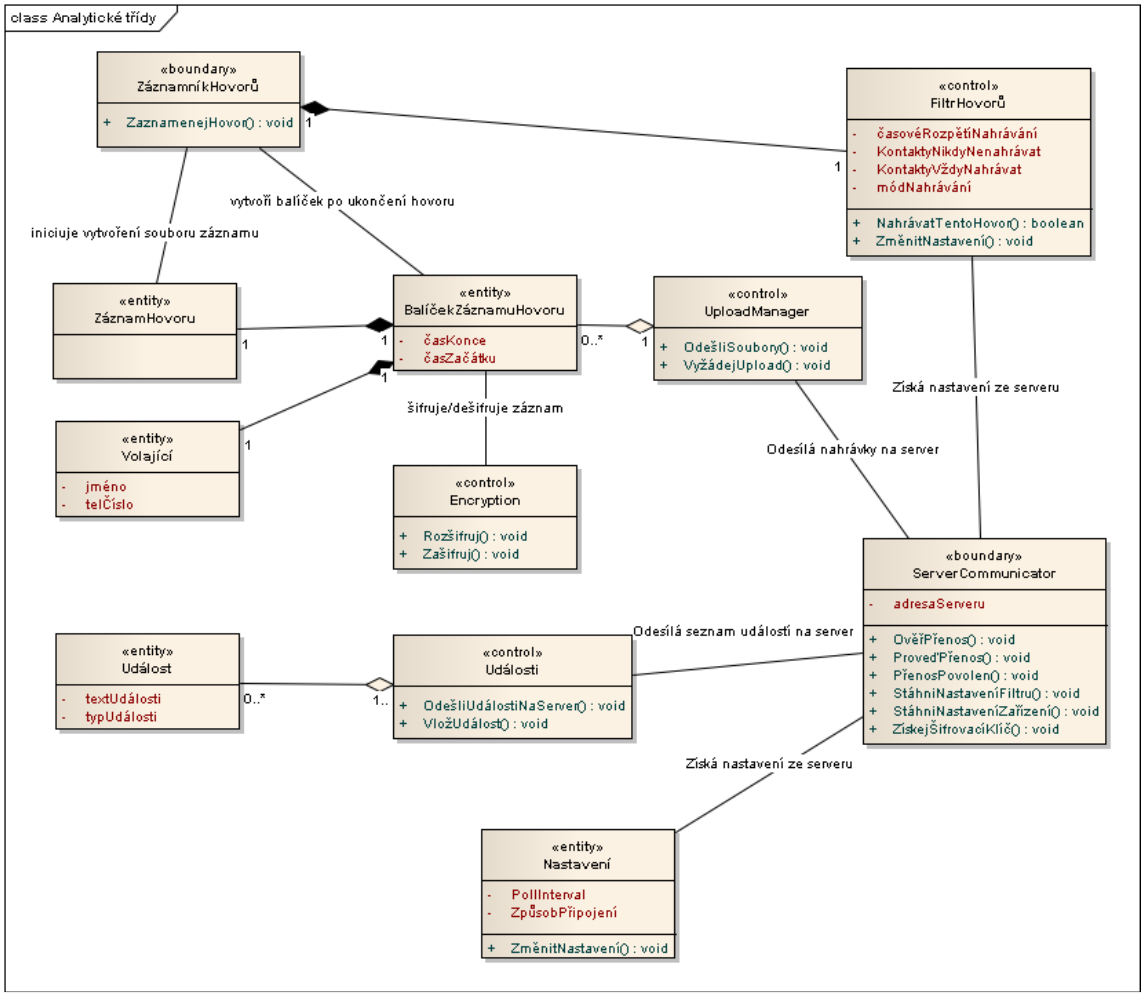
Příloha B – Diagram požadavků



Obrázek A. 2 - Diagram požadavků

Zdroj: vlastní zpracování

Příloha C – Diagram analytických tříd



Obrázek A. 3 - Diagram analytických tříd

Zdroj: vlastní zpracování

Příloha D – CD

Součástí práce je CD s následujícím obsahem:

- *apk/* - balíček aplikace,
- *diplomova_prace/* - text diplomové práce,
- *doc/* - dokumentace zdrojových kódů,
- *source/* - zdrojové kódy,
- *uml/* - diagramy.