VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

PŘENOS A ZÁLOHOVÁNÍ TEXTOVÝCH ZPRÁV

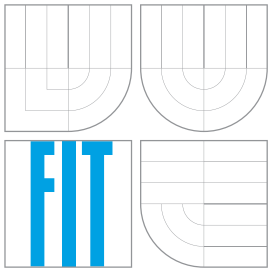
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

FRANTIŠEK ČERNÝ

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

PŘENOS A ZÁLOHOVÁNÍ TEXTOVÝCH ZPRÁV

TRANSFER AND BACKUP OF TEXT MESSAGES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

FRANTIŠEK ČERNÝ

VEDOUCÍ PRÁCE
SUPERVISOR

Doc. Dr. Ing. DUŠAN KOLÁŘ

BRNO 2015

Abstrakt

Předmětem této bakalářská práce je dvojce kooperujících aplikací pro zálohu, obnovu a prohlížení SMS zpráv. První z implementovaných aplikací určená pro operační systém Android provádí zálohu a obnovu SMS zpráv za pomocí XML souboru. Aplikace také umožňuje synchronizaci importovaného souboru s již existujícími krátkými textovými zprávami. Druhá aplikace slouží jako doplněk pro poštovního klienta Thunderbird. Tento doplněk umožňuje zobrazit importované SMS zprávy, synchronizovat uložené kontakty s příslušnými SMS vlákny, vyhledávat v nahraném souboru a provádět synchronizaci s již existujícími daty. Doplněk dovoluje také exportovat soubor s SMS zprávami, a ten poté obnovit pomocí aplikace pro operační systém Android.

Abstract

The subject of this bachelor thesis are two cooperating applications to backup, restore and to view SMS messages. The first of the implemented applications, designed for the Android operating system can backup and restore SMS messages using XML file. The application also enables synchronization of the imported file with existing short text messages. The second application is used as an extension for Thunderbird email client. This extension lets you view imported text messages, synchronize your saved short text messages with contacts in SMS threads, search in saved file and synchronize existing SMS information with newly imported file. The extension also allows exporting a file with SMS messages and uses this exported file in application for Android operating system.

Klíčová slova

Java, Android, Google, kontakty, SMS, záloha, synchronizace, obnova, Thunderbird

Keywords

Java, Android, Google, contacts, SMS, backup, synchronization, restore, Thunderbird

Citace

František Černý: Přenos a zálohování textových zpráv, bakalářská práce, Brno, FIT VUT v Brně, 2015

Přenos a zálohování textových zpráv

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana doc. Dr. Ing. Dušana Koláře. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
František Černý
18. května 2015

Poděkování

Velice rád bych chtěl poděkovat doc. Dr. Ing. Dušanovi Kolářovi za vstřícnost, jeho vynaložený čas, trpělivost a předané zkušenosti při tvorbě této bakalářské práce.

Obsah

Kapitola 1

Úvod

V dnešním světě moderních technologií a komunikací je téměř nutností vlastnit mobilní telefon. Lidé více telefonují, píší SMS zprávy či jiným způsobem komunikují prostřednictvím mobilních zařízení. Možnost tohoto typu spojení stále více využívají také poskytovatelé různých služeb. Poskytovatelé přes tyto zařízení zasílají například informační krátké textové zprávy o dokončení objednávek, registrací apod. V nemalé míře jsou zákazníkům také zasílány důležité informace, které by bylo dobré zálohovat. Zálohu SMS zpráv je vhodné provádět pravidelně, pokud o dané zprávy nechceme přijít a to třeba v případě, kdy se náš mobilní telefon stane nefunkční a nebo když jej ztratíme.

Problém uvedený v odstavci výše se pokouší řešit aplikace, které byly implementovány v rámci této bakalářské práce. První z aplikací je implementována pro operační systém Android a slouží pro zálohu a obnovu SMS zpráv v mobilním telefonu. Záloha je provedena ve formě XML souboru, který je výstupem zmíněné aplikace. Jako vhodné rozšíření je tato aplikace doplněna o možnost si exportované SMS zprávy zpět importovat do mobilního telefonu. Funkce importování je dále rozšířena o algoritmus, který se stará o správnou synchronizaci existujících SMS zpráv se zprávami nově importovanými.

V rámci této bakalářské práce je dále implementován doplněk do poštovního klienta Thunderbird. Tento doplněk je určen k prohlížení a importu výstupního XML souboru ze zmíněné aplikace. Při importování SMS zpráv do poštovního klienta Thunderbird, je i v tomto případě možnost soubor synchronizovat s již existujícími SMS zprávami. Doplněk pro poštovního klienta Thunderbird také provádí vyhledávání a synchronizaci kontaktů s SMS vlákny uloženými v poštovním klientovi Thunderbird. Jako vhodné rozšíření je doplněna možnost zpětné serializace zobrazených dat do výstupního XML souboru. Výstupní soubor je možné použít pro následný import do operačního systému Android.

V kapitole *Platforma Android* jsou popsány základní stavební bloky této platformy a dále je popsán vývoj aplikací pro tuto platformu. V závěru kapitoly je uveden způsob ukládání SMS zpráv v operačním systému Android.

V kapitole *Poštovní klient Thunderbird* jsou uvedeny základní informace o tomto programu a dále je popsán způsob uložení a práce s kontakty.

Kapitola *Specifikace a návrh aplikace* zachycuje a popisuje základní případy užití a uvádí informace o návrhu jednotlivých implementovaných aplikací.

Kapitola *Implementace* popisuje způsob implementace výsledných aplikací vytvořených v rámci této bakalářské práce.

V kapitole *Testování* je popsán způsob testování implementovaných aplikací.

Poslední kapitola popisuje možnosti dalšího rozvoje vytvořených aplikací a diskutuje o jejich přínosu.

Kapitola 2

Platforma Android

Obsahem této kapitoly je popis základní charakteristiky a základních funkčních bloků operačního systému Android. Podrobně jsou také rozebrány možnosti získávání SMS zpráv ze zařízení, na kterých je používán operační systém Android.

2.1 Charakteristika operačního systému Androidu

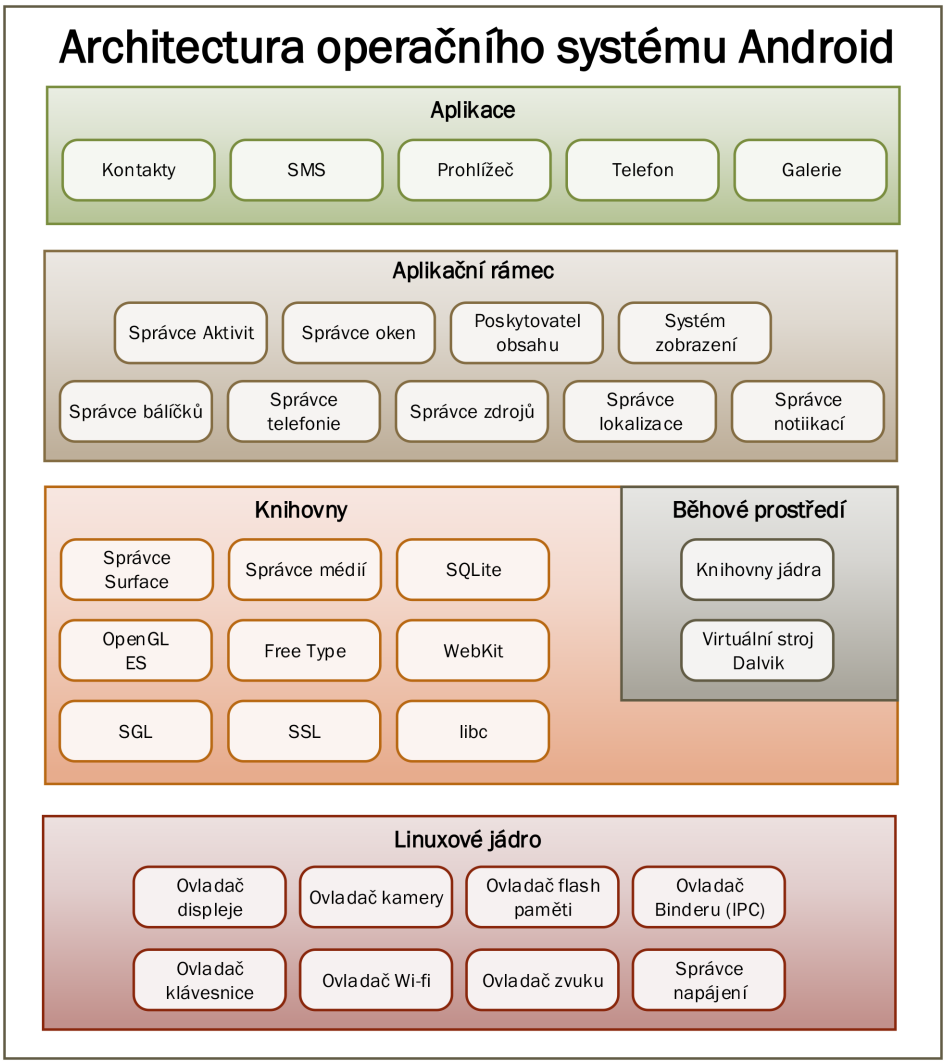
Android je softwarová platforma a také operační systém, který poskytuje bohatý aplikační rámec od samotného jádra systému, až po aplikace běžící v tomto operačním systému [18]. Operační systém Android je převážně používán na přenosných mobilních zařízeních, jako je tablet, chytrý telefon, čtečka elektronických knih a mnoho dalších podobných zařízení. Od roku 2014 je operační systém Android také provozován v chytrých televizorech a mnohých chytrých hodinkách.

Operační systém Android byl založen firmou Android Inc. v roce 2003. Projekt této firny byl ze začátku vyvíjen jen menší skupinou lidí, ale poté co se o něm v roce 2005 dozvěděla společnost Google, se jeho vývoj rapidně urychlil. Společnost Google se rozhodla, že firmu odkoupí a bude dále pokračovat ve vývoji projektu. Z firmy Android Inc. se tedy stala dceřiná společnost společnosti Google.

Po dvou letech vlastnictví firmy Android Inc. vytvořila společnost Google uskupení Open Handset Alliance [18]. Toto uskupení se sestávalo z hlavních představitelů výrobců mobilních telefonů a elektroniky. Pro vývoj operačního systému Android bylo toto uskupení velmi důležité, jelikož jeho hlavním cílem bylo vyvinout jednotný standard pro nová mobilní zařízení.

2.2 Architektura operačního systému Android

Architektura operačního systému Android se skládá z vrstev. Každá vrstva obsahuje skupinu kooperujících funkcí, které začínají na vrstvě nejspodnější (technické vybavení) a vedou až k vrstvě s funkcemi pro tvorbu aplikací. S postupným zvyšování stupně jednotlivých vrstev se zvyšuje také stupeň jejich abstrakce. Toto uskupení vrstev a jednotlivých úrovní se v anglické literatuře označuje jako *SoftwareStack* [35]. Popis vrstev a jejich prvků je zobrazen na obrázku 2.1.



Obrázek 2.1: Architektura operačního systému Android

2.2.1 Linuxové jádro

Linuxové jádro je napsáno v programovacím jazyce C a nachází se na nejspodnější vrstvě. Primárním úkolem této vrstvy je komunikace s technickým vybavením daného zařízení. Tato vrstva nám poskytuje určitou programovou abstrakci, která vývojáře oprostí od znalosti technického vybavení jednotlivých zařízení. V dnešních zařízeních je typicky možné nalézt Linuxové jádro ve verzi 3.4. Linuxové jádro jako takové se však může na různých zařízeních lišit [9].

Linuxové jádro se v operačním systému Android primárně stará o komunikaci mezi procesy (*Binder*), alokováním paměti danými procesy (*pmem*), energetické zdroje (*oom handling*) a další [7].

2.2.2 Virtuální stroj Dalvik

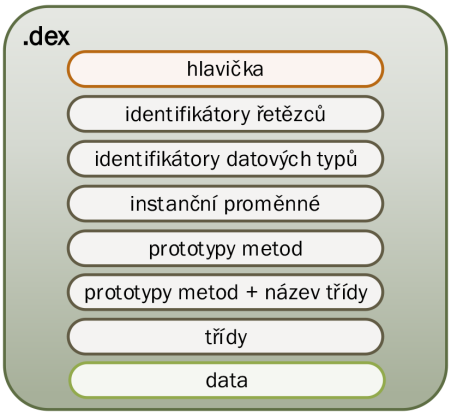
Virtuální stroj Dalvik (*Dalvik Virtual Machine* DVM) slouží jako náhrada za virtuální stroj Java (*Java Virtual Machine* JVM) a byl vytvořen ze dvou důvodů:

1. Používání JVM je placené společnosti Oracle, tedy její koncepce používání je neslučitelná s používáním operačního systému Android a základní vizí společnosti Google.

2. Odlišné požadavky na mobilní zařízení, které mají rozdílné parametry oproti stolním/přenosným počítačům (menší kapacita baterie, méně paměti atd.).

Virtuální stroj Dalvik na svůj vstup dostává binární kód, který je generován překladačem Java a za pomocí Dex kompilátoru je vytvořen speciální soubor s příponou .dex (*Dalvik Executable*) [44].

Strukturu souboru s příponou .dex je možné vidět na obrázku 2.2. Soubor s touto příponou se liší od klasických souborů s příponou .class používaných a generovaných pro JVM hlavně tím, že v jejich zapouzdřených datech jsou odstraněny vícekrát použité knihovny a jsou nahrazeny odkazem na příslušnou knihovnu [3]. Z tohoto důvodu jsou soubory s příponou .dex menší než soubory s příponou .class.



Obrázek 2.2: Struktura .dex souboru

2.2.3 Knihovny

Tato vrstva leží hned nad vrstvou linuxového jádra. Její hlavní funkcí je poskytovat funkce vrstvám na vyšších úrovních. Knihovny jsou napsány v jazyce C/C++ a liší se na základě použitého technického vybavení daného zařízení [35]. Částečný výčet knihoven operačního systému Android je uveden níže.

- *Bionic* – knihovna založena na *libc* a optimalizována pro použití ve vestavěných zařízeních [4]

- Správce Surface (*Surface Manager*) – stará se o uspořádání grafických prvků a také o transparentnost oken v aplikacích [35]

- Správce médií (*Media Framework*) – poskytuje rozdílné druhy knihoven a funkcí pro přehrávání či nahrávání různého typu dat [34]

- *WebKit* – hlavní knihovna pro zobrazování webových stránek [45]
- *SQLite* – knihovna pracující s SQL databázemi (kontakty, SMS, uživatelské databáze a jiné) [8]
- *OpenGL* – zobrazuje dvourozměrné a trojrozměrné grafické prvky na displej [13]
- *Skia Graphics Engine* – základní grafická knihovna operačního systému Android, která pracuje spolu s nativními knihovnami *Window* a *Surface Manager* [19]
- *Free Type* – knihovna starající se o vykreslování fontů [37]
- *SSL* – slouží k zabezpečení síťové komunikace

2.2.4 Aplikační rámec

Aplikační rámec je nejvíce používaná vrstva samotnými vývojáři. Tato vrstva obsahuje všechny důležité prvky pro tvorbu aplikací od správy balíčků aplikací nainstalovaných na mobilním zařízení, až po správu zabývající se oznamovacími zprávami. Výčet některých prvků, které jsou obsaženy v aplikačním rámci bude popsán v následujícím seznamu odrážek [11].

- Správa aktivit (*Activity manager*) – Správa aktivit slouží a stará se o klíčový prvek v každé Android aplikaci, Aktivitu.
- Správa telekomunikace (*Telephony manager*) – Poskytuje přístup zejména k prvkům sloužícím k telefonování a posílání SMS zpráv.
- Správa oznamovacích zpráv (*Notification manager*) – Stará se o oznamovací oblast (*Notification area*) v operačním systému Android (příchozí hovor, zpráva...).
- Poskytovatel obsahu (*Content provider*) – Poskytovatel obsahu umožňuje přistupovat k obsahu různých aplikací (SMS zprávy, kontakty, záložky...).
- Správa zdrojů (*Resource manager*) – Spravuje všechny potřebné zdroje dat pro aplikaci (kromě zdrojových kódů).

2.2.5 Aplikace

Aplikace vyvíjené pro operační systém Android jsou napsány převážně v programovacím jazyce Java. Aby byla ulehčena dostupnost a propagace vyvinutých aplikacích je zřízen oficiální elektronický obchod (*Play Store*). Na tomto místě jsou nabízeny aplikace vyvíjené různými programátory a vývojáři aplikací pro operační systém Android. Vyvíjená aplikace je distribuována jako jeden samostatný soubor s příponou *.apk* a obsahuje všechny potřebné zdrojové, grafické a multimediální soubory nutné pro správný běh aplikace [11].

Soubor formát apk

Soubory formát *apk* vychází s balíčkových souborových formátů typu *jar* a *zip*. Tento formát je analogií k již známým souborovým formátům od Windows *msi* nebo Debianu *deb* [41].

Soubor s příponou *.apk* vznikne za pomocí nástrojů poskytovaných společností Google. Výsledná implementovaná aplikace je přeložena za pomocí nástroje *aapt*. Nástroj *aapt* je

dostupný přímo v Android SDK [29]. Při vytváření souboru s příponou *.apk* však většina vývojářů tento nástroj nepoužívá přímo, ale nechá vývojové prostředí aby za ně výsledný soubor s příponou *.apk* vytvořilo.

Zabezpečení aplikace

Každá aplikace má povolen přístup pouze ke svým zdrojům a je oddělena od ostatních aplikací. Každá aplikace v operačním systému Android představuje jednoho uživatele s unikátním identifikačním číslem přiřazeným operačním systémem Android (samotná aplikace nazná své identifikační číslo) [11].

Při spouštění aplikace na mobilním zařízení je vytvořen proces, který je přiřazen právě k jednomu virtuálnímu stroji. Aplikace tak běží odděleně v izolovaném prostředí. Běh aplikace může být zastaven samotnou aplikací, nebo ji může okamžitě ukončit operační systém Android (například z důvodu nedostatku operační paměti).

Pokud vývojář chce, aby aplikace sdílely svá data, je toho možné dosáhnout přiřazením stejného uživatelského ID dvěma různým aplikacím. Tyto aplikace pak mají možnost navzájem sdílet své soubory [11].

Pokud aplikace potřebuje přístup k dalším zdrojům, které jsou přístupné jiným aplikací (GPS, Wifi, Bluetooth...) musí být jejich použití povoleno při instalaci aplikace na dané zařízení.

2.3 Vývoj aplikace a základní bloky

Základními bloky operačního systému Android jsou Aktivita, Služba, Poskytovatel obsahu a Přijímač všesměrových událostí. Detailní popis zmíněných bloků bude uveden v dalších kapitolách.

2.3.1 Vývojové nástroje

Aplikace vyvíjené pro operační systém Android jsou primárně napsány za pomocí Android SDK v programovacím jazyce Java. Tento způsob implementace aplikací však není jediný a operační systém Android nabízí možnost psát aplikace také ze pomocí knihoven v programovacím jazyce C/C++. Takto napsané aplikace používají stejné principy jako aplikace psané v programovacím jazyce Java [41].

Android SDK a NDK

Vývoj aplikace pomocí Android SDK je možný několika způsoby [15]:

- Programování v textovém editoru a manuální překlad souborů z příkazové řádky.
- Instalace oficiálního zásuvného modulu Android ADT pro integrované vývojové prostředí Eclipse.
- Používání oficiálního kompletního vývojové prostředí Android Studio.

Vývojový nástroj Android NDK slouží pro vývoj aplikací za pomocí knihoven operačního systému Android. Výhoda tohoto způsobu tvorby aplikací je možnost vyvíjet aplikace v menší úrovní abstrakce, tudíž může mít vývojář větší přehled o tom co se v aplikaci děje a uzpůsobit více její konfiguraci pro specifický účel. Při tomto způsobu vývoje aplikací

má vývojář také možnost využít již vestavěné existující knihovny. Nevýhodou tohoto vývoje je větší komplikovanost a nepřehlednost zdrojových kódů [10]. Z tohoto důvodu by se tento způsob vývoje aplikací měl využívat jen v případech, kdy je velmi nutná optimalizace aplikace (například z důvodu velkého nároku na technické vybavení daného zařízení).

2.3.2 Aktivita

Aktivita (*Activity*) je jedna z nejdůležitějších komponent operačního systému Android. Obsahuje viditelné uživatelské rozhraní a dovoluje uživateli aplikaci ovládat. Každá aplikace může obsahovat více Aktivit a každá Aktivita by měla být navržena tak, aby byla použita pouze pro jeden druh operace. Příklad takového rozdělení Aktivit je například zobrazení seznamu SMS zpráv a zobrazení příslušné SMS zprávy.

Každé Aktivitě je přiřazen prostor, do kterého může být vykreslena. Typicky je aktivita zobrazena přes celou obrazovku, ale v některých případech může být zobrazena pouze přes její část.

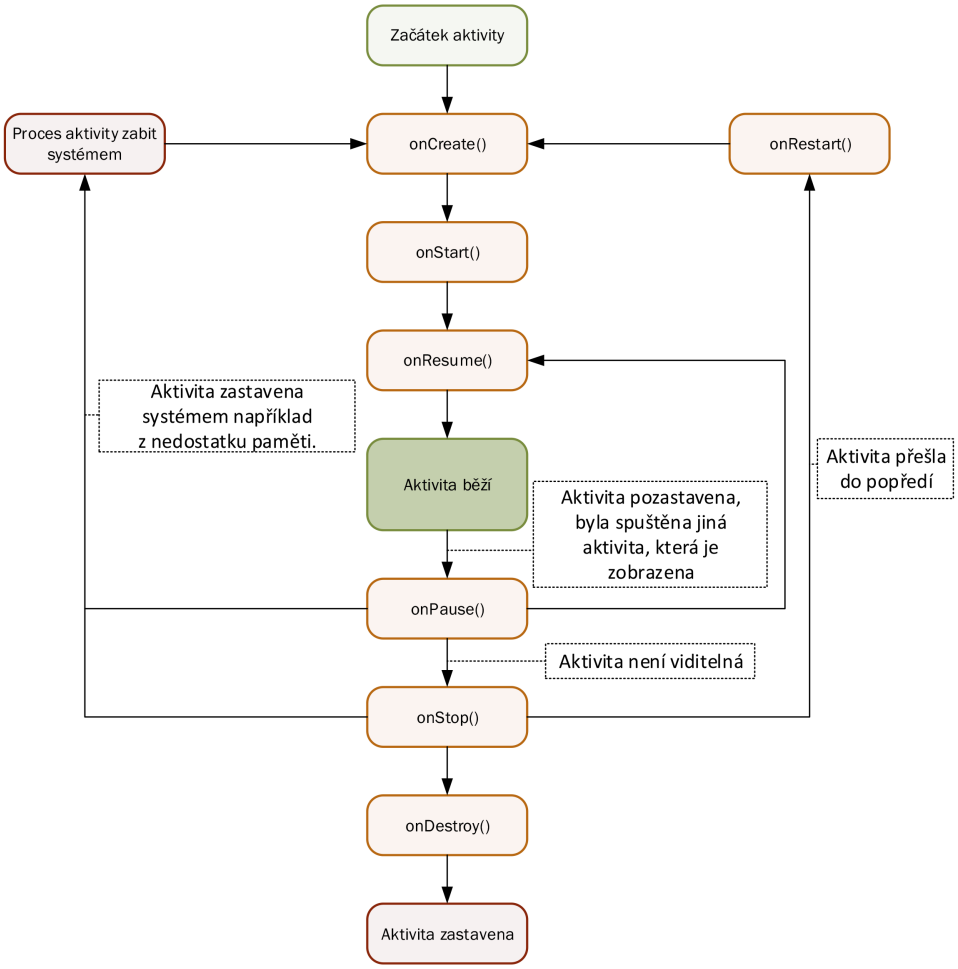
Při spouštění aplikace je zobrazena aktivita, která je označena jako hlavní a slouží jako vstupní bod do aplikace. Tato Aktivita je řízena vlastním životním cyklem [16], který je zobrazen na obrázku 2.3. Jak je možné vidět, aplikace prochází jednotlivými stavy a tyto stavy jsou reprezentovány metodami ve třídě ***android.app.Activity***. Popis jednotlivých stavů (metod) Aktivity je uveden v následujícím odstavci.

- ***onCreate*** – Nejdůležitější z metod, vytváří Aktivitu a inicializuje potřebné zdroje.
- ***onStart*** – Spouští Aktivitu, v tomto stavu je uživateli zobrazena.
- ***onResume*** – Tato metoda je volána, když se uživatel vrátí do dané aktivity, po ukončení interakce s jinou aktivitou.
- ***onPause*** – Aktivita je pozastavena, typicky z důvodu spouštění jiné aktivity, může být zrušena systémem [14].
- ***onStop*** – Aktivita je zastavena a není viditelná uživateli, může být zrušena systémem [17].
- ***onRestart*** – Aktivita je znovu spuštěna.
- ***onDestroy*** – Aktivita je zrušena a jsou uvolněny všechny zdroje, které používala.

Úloha

Úloha (*Task*) obsahuje posloupnost Aktivit, které mezi sebou mají nějaký vztah. Ve většině případů to budou Aktivity, které poskytuje jedna aplikace, ale nemusí tomu tak být vždy. Úloha nám tak zajišťuje určité logické spojení Aktivit, i když nejsou v jedné aplikaci [41]. Uživatel používající naší aplikaci tedy nepozná, zda je daná funkčnost (Aktivita) implementována v naší aplikaci, nebo je spuštěna Aktivita aplikace jiné. Všechny úlohy jsou uloženy do datové struktury zásobník.

Zásobník úloh (*Task Back Stack*) je datová struktura typu zásobník (LIFO). Jejím úkolem je uchovávat úlohy, které nejsou uživateli viditelné, ale jsou stále spuštěné [41]. Zásobník úloh může provádět dvě základní operace `PUSH` a `POP`. Jako parametry těchto operací jsou potom jednotlivé Aktivity.



Obrázek 2.3: Životní cyklus Aktivity

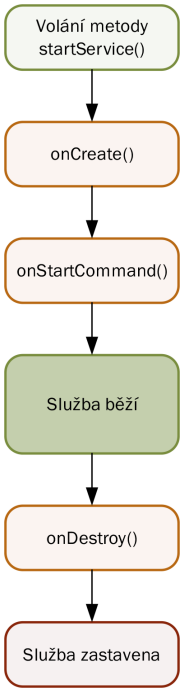
2.3.3 Služba

Služba (*Service*) je další ze základních často používaných prvků operačního systému Android. Oproti Aktivitě Služba neobsahuje žádné uživatelské rozhraní. Služba je primárně určena pro dlouho běžící operace, typicky na pozadí, jako je stahování dat či přehrávání hudby. Vyvolání určitého typu Služby se běžně provádí přes Aktivitu, která danou Službu spouští. Jakmile je Služba spuštěna, běží v závislosti na Aktivitě, která ji spustila. Stejně jako Aktivita i Služba má svůj životní cyklus [46]. Grafické znázornění životního cyklu Služby je na obrázcích 2.4 a 2.5.

Služby jsou dostupné v balíčku ***android.app***. Android nabízí dva různé druhy služeb.

První ze služeb se označuje jako nevázaná služba (*Started Service*) je jednoduchá na používání a pracuje pouze s jednou operací. Její nevýhodou je absence možnosti vrátit výsledek operace volajícímu.

Druhá služba se nazývá vázaná služba (*Bind Service*) vytváří komunikační rozhraní mezi volajícím (typicky Aktivitou) a Službou. Přes vytvořené rozhraní je pak možné komunikovat a zjišťovat tak stav Služby. Na službu může být napojeno i více klientů a získávat o ní aktuální informace [46].



Obrázek 2.4: Životní cyklus Nevázané služby (Start Service)

Obrázek 2.5: Životní cyklus Vázané služby (Bind Service)

2.3.4 Příjemce všesměrových událostí

Přijímač všesměrových událostí (*Broadcast Recievер*) je další ze základní komponent operačního systému Android. Primárním úkolem této komponenty je odpovídat a přijímat události vysílané operačním systémem Android nebo jinou aplikací. Pod pojmem všesměrová událost si můžeme představit například příchozí hovor, informace o stavu baterie v zařízení, příchozí zprávu SMS a další.

Událost, která je přijímána aplikací za pomocí přijímače všesměrových událostí se nazývá Záměr (anglicky označován jako *Intent*). Záměry primárně slouží k přenášení událostí mezi Aktivitami. Typicky slouží k oznamování, že má být daná Aktivita spuštěna. Pokud operační systém Android detekuje odesílání nějakého Záměru, tak určí podle nastavení spuštěných Aktivit, která z nich má na daný Záměr zareagovat [47]. Pokud není specifikované, která Aktivita má na Záměr reagovat, vyhledá operační systém Android vhodné Aktivity, které jsou schopny reagovat na daný Záměr.

V operačním systému Android jsou dostupné dva odlišné druhy Záměrů. Je možné se setkat s Záměry aplikačními nebo systémovými. Systémové Záměry jsou předem definovány a uloženy v operačním systému Android. Uživatelské Záměry jsou definovány programátorem.

Jednomu Záměru může příslušet až několik příjemců všesměrových událostí. Při zjištění, že je daná událost (Záměr) vysílána, příjme toto vysílání pouze aplikace, která má příjem daného vysílání povolen. Příjem daného vysílání je možné pomocí metody ***onRecieve***

ve třídě ***android.content.BroadcastReciever*** zachytit a danou událost (Záměr) dále zpracovávat[20].

2.3.5 Poskytovatel obsahu

Poskytovatel obsahu (*Content Provider*) se řadí mezi jeden ze čtyř nejdůležitějších prvků operačního systému Androidu. Hlavním úkolem, jak již název napovídá, je poskytnout přístup aplikacím k úložišti různých druhů informací dostupných v operačním systému Android [49]. Mezi takové typy dat patří například kontakty, záložky webového prohlížeče, SMS/MMS zprávy a další [21].

Klíčovou vlastností Poskytovatele obsahu je možnost přístupu více aplikací. Oproti přímému přístupu k SQLite databázi poskytují Poskytovatelé obsahu také větší míru zabezpečení [48]. Typickým příkladem zabezpečení je přístup do databáze a změna dat souběžně běžícími procesy. Jelikož poskytovatel obsahu nemusí pracovat pouze nad databází SQLite, je možné jeho chování změnit a měnit tak data například v souborech nebo dokonce i na vzdáleném serveru [1].

2.3.6 Soubor Manifest

Soubor Manifest je ve formátu XML a musí jej obsahovat každá aplikace pro operační systém Android. Soubor obsahuje základní informace o aplikaci jako je jméno, seznam spustitelných Aktivit a dalších implementovaných prvků. Součástí tohoto souboru je také seznam povolení, který udává se kterými funkcemi a vlastnostmi mobilního telefonu a operačního systému Android, může aplikace pracovat.

Důležitou součástí souboru Manifest je také seznam akcí, který náleží příslušným implementovaným Aktivitám [25]. Podle těchto nastavených operační systém Android vybírá Aktivity, které budou zobrazeny při vykonávání daných akcí v operačním systému Android (otevření souboru, odeslání SMS zprávy, přijetí SMS zprávy a další).

2.4 SMS zprávy v operačním systému Android

Jelikož se v operačním systému Android od verze 4.4 KitKat zpřísnila bezpečnost práce s SMS zprávami, bude v této kapitole popsáno také správné nastavení aplikace pro operační systém Android od této verze [38].

2.4.1 Zápis SMS zpráv

Systém Android uchovává SMS zprávy v relační databázi SQLite. Tato databáze obsahuje celkem 18 databázových tabulek. Struktura dvou nejdůležitějších tabulek pro ukládání a správu SMS zpráv je uvedena v tabulkách A.2 a A.1 uvedených v příloze této bakalářské práce.

Při zápisu SMS zpráv v operačním systému Android od verze API 19 (Android 4.4 KitKat) je nutné, aby aplikace splňovala určité požadavky. Nejenže aplikace musí být nově nastavena jako výchozí aplikace pro práci s SMS zprávami, aplikace musí také implementovat funkce pro práci s SMS zprávami (odesílání, načítání, naslouchání příchodů nové SMS zprávy) i v případě, že dané funkce vůbec nebude používat.

Aplikace, kterým je umožněn zápis do SMS databáze musí obsahovat třídy, které musí dědit z jedné z následujících tříd v operačním Systému Android [38]. Pokud chybí některá

z níže uvedených tříd, operační systém Android neuzná aplikaci za vhodnou pro práci s SMS zprávami a tato aplikace nemůže být nastavena jako výchozí.

- ***android.app.Activity*** – zobrazení
- ***android.app.Service*** – dlouho běžící operace na pozadí
- ***android.content.BroadcastReceiver*** – příjem SMS zpráv
- ***android.content.BroadcastReceiver*** – příjem MMS zpráv

Kromě výše uvedených tříd, které musí aplikace využívat, je také nutné příslušné třídy spárovat s danými povoleními v manifest souboru. Více informací je možné nalézt v [38].

SMS

Databázová tabulka s SMS zprávami v sobě uchovává ty nejdůležitější informace o každé SMS zprávě. Popis nejdůležitějších atributů, které byly získány za pomocí aplikace SQLite Debugger [6]:

- *_id* – Jednoznačně identifikuje danou SMS zprávu.
- *thread_id* – Atribut identifikující vlákno, ke kterému patří daná SMS zpráva.
- *address* – Identifikuje osobu, která poslala příslušnou SMS zprávu pomocí telefonního čísla.
- *person* – V případě, že je v mobilním telefonu uložen kontakt pro telefonní číslo, ze kterého byla přijata SMS zpráva, obsahuje tento atribut identifikátor pro daný kontakt, odkazující se do seznamu kontaktů. Pokud je tato hodnota `NULL` a pokud se jedná o přijatou SMS zprávu znamená to, že dané telefonní číslo nemá přiřazen kontakt v seznamu kontaktů.
- *type* – Udává o jaký typ SMS zprávy se jedná (přijatá, odeslaná, koncept a další).
- *body* – Text SMS zprávy.
- *read* – Atribut udávající zda byla daná SMS zpráva přečtena.
- *date* – Datum přijetí příslušné SMS zprávy.

SMS vlákna

Vlákna v souvislosti s SMS zprávami zaručují rozlišení konverzací pomocí SMS zpráv mezi jednotlivými kontakty. Popis nejdůležitějších atributů:

- *_id* – Jednoznačně identifikuje dané vlákno.
- *message_count* – Počet SMS zpráv v daném vlákně.
- *snippet* – Prvních 45 znaků z textu poslední SMS zprávy v daném vlákně.
- *recipient_ids* – Identifikační čísla pro telefonní čísla, která se podílely na konverzaci v daném vlákně. (Jestliže obsahuje tento atribut více kontaktů jsou identifikátory odděleny mezerou.)

Kapitola 3

Poštovní klient Thunderbird

V následující kapitole bude v krátkosti popsán program Thunderbird. Důraz bude v této kapitole kladen zejména na ukládání a správu kontaktů. Ve druhé části bude uveden způsob tvorby doplňků pro tohoto poštovního klienta.

3.1 Základní popis

Thunderbird je program, který primárně slouží ke správě elektronické pošty. Jádro tohoto klienta elektronické pošty je založeno na jádře populárního webového prohlížeče Firefox. Poštovní klient Thunderbird je implementován v programovacím jazyce C/C++ [43]. Mezi klíčové vlastnosti a odlišnosti klienta elektronické pošty Thunderbird patří licence a rozšiřitelnost. Program Thunderbird je publikován pod licencí GPL. Rozšiřitelností je myšlena možnost do programu doinstalovat další doplňky jako jsou kalendář, správce kontaktů či detektor nevyžádané pošty.

Poštovní klient Thunderbird byl vytvořena společností Mozilla 28. července 2003. Prvotní implementace této aplikace se jmenovala Minotaur, avšak vývoj této aplikace byl v raném stádiu zastaven. Vývoj aplikace byl znovu obnoven po úspěchu aplikace Firefox od společnosti Mozilla [2] nově pod jménem Thunderbird.

3.2 Adresář kontaktů

Adresář kontaktů je jeden z nejdůležitějších prvků každého klienta elektronické pošty. Poštovní klient Thunderbird obsahuje také možnost ukládat a spravovat uložené kontakty. Kontakty jsou ukládány do textového souboru. Formát souboru, ve kterém jsou informace uloženy, se nazývá Mork [51]. Tento typ souboru má standardně jednu z přípon *.mab*, *.msf* nebo *.dat*. Při vytváření nového adresáře kontaktů ve správci kontaktů poštovního klienta Thunderbird je tedy vytvořen soubor typu Mork. Struktura souboru s uloženými kontakty je popsána v následující kapitole.

3.2.1 Souborový formát Mork

Jak již bylo uvedeno v kapitole 3.2 Mork je textový souborový formát. Byl navržen tak, aby výsledný formát měl co možná nejmenší velikost a podobal se jednoduchému textovému souboru. Souborový formát Mork se skládá z entit a každá entita má přiřazenou svou značku v souboru.

Základním modelem uspořádání obsahu v souboru typu Mork je tabulka [39]. Ta se jako klasická tabulka skládá z řádků a každý řádek obsahuje buňky. Každá buňka je členem právě jednoho sloupce. Buňka v sobě uchovává hodnotu a tato hodnota může být přímá nebo reprezentována odkazem. Odkazy můžou být i ve více úrovních.

3.3 Přístupnost vestavěných funkcí

Aby doplňky implementované pro poštovního klienta Thunderbird mohly využívat téměř všechny funkce, které jsou uvnitř něj implementovaný využívá univerzálního standardu zkráceně označovaný jako CORBA (*Common Object Request Broker Architecture*). Tato architektura dovoluje vývojářům přistupovat k objektům, které jsou napsány v různých programovacích jazycích [40]. Objekty které jsou poté takto zpřístupněny jsou popsány syntaxí označovanou jako IDL Interface Definition Language. V případě tvorby doplňku v programovacím jazyce Javascript, můžeme k těmto objektům přistupovat například takto:

```
Components.classes["@mozilla.org/abmanager;1"]
```

3.4 Vývoj doplňků pro poštovního klienta Thunderbird

Vývoj doplňku pro poštovního klienta Thunderbird je možné tvořit za pomocí programovacího jazyka Javascript a značkovacího jazyka založeného na struktuře souborů XML. Při vývoji aplikace je programová logika tvořena za pomocí programovacího jazyka Javascript a zobrazení grafické uživatelského rozhraní je zajištěno soubory XUL a definovány různými grafickými elementy (tlačítko, menu a jiné) [40]. Popis funkcí sloužících pro práci s adresářem kontaktů, který je nezbytný pro vývoj doplňku do poštovního klienta Thunderbird v rámci této bakalářské práce, bude popsán v kapitole 3.5.

3.5 Práce s kontakty v poštovním klientovi Thunderbird

Pro uložení a správu kontaktů slouží souborový formát Mork (kapitola 3.2.1). Jak bylo zmíněno dříve, struktura tohoto souboru je velice nestandardní. Nástroje pro tvorbu a správu kontaktů nám však nabízejí funkce, které dokáží s tímto souborem a tedy i se samotnými kontakty velice jednoduše pracovat.

Nejdůležitější z funkcí pro získání informací o kontaktech nám zajišťuje rozhraní ***nsIAbManger*** [42]. Toto rozhraní obsahuje všechny složky s veškerými kontakty v adresáři kontaktů poštovního klienta Thunderibrd. Pro přístup ke všem složkám z adresáře kontaktů je nutné získat objekt ***nsISimpleEnumerator***. Z položek tohoto objektu je poté možné získat všechny kontakty. Kontakty jsou uloženy jako objekt, který implementuje rozhraní ***nsIAbCollection***. Objekt obsahuje všechny přístupné vlastnosti pro daný kontakt (jméno, příjmení, adresa, telefonní číslo...). Položky (výčet všech dostupných položek v [50]) kontaktu jsou identifikovatelné jejich názvem a jednotlivé informace zadaného kontaktu získáme pomocí metody ***getCardFromProperty***.

Kapitola 4

Specifikace a návrh aplikace

Kapitola popisuje návrh a specifikaci dvou kooperujících aplikací. První z aplikací, která je vyvíjena v rámci této bakalářské práce, bude sloužit pro exportování a importování SMS zpráv v mobilních zařízeních s operačním systémem Android. SMS zprávy, které budou exportovány z operačního systému Android, bude možné nahrát do další aplikace, která bude implementována jako doplněk do poštovního klienta Thunderbird. Aplikace bude řešit různé druhy konfliktů při synchronizaci SMS zpráv a každou SMS zprávu bude možné přiřadit ke kontaktu který bude uložen v aplikaci Thunderbird.

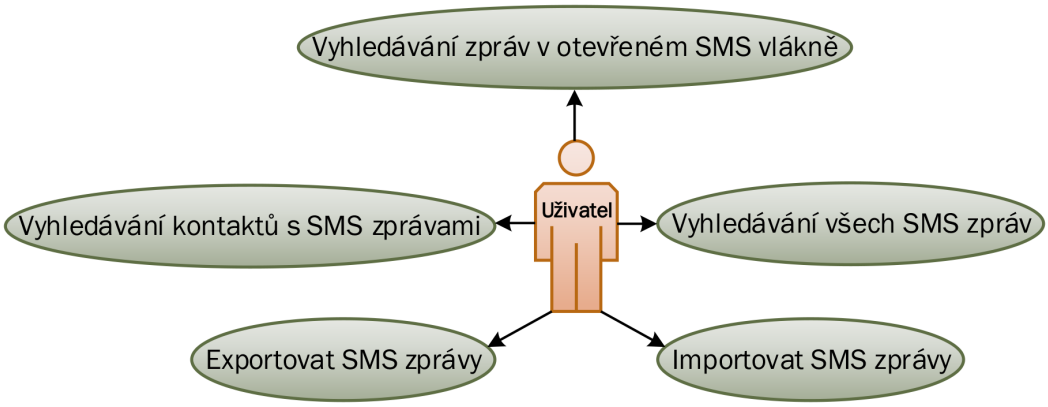
Android aplikace bude také synchronizovat seznam kontaktů pomocí Aktivit z aplikace, kterou vytvořil Ing. Viliam Kasala. Více informací o této aplikaci je možné získat z [36]. Dále bude popsán import a export SMS zpráv z/do mobilního telefonu s operačním systémem Android. Důraz bude v této kapitole kladen zejména na popis vyhledávání shody mezi SMS vláknem a příslušným kontaktem v poštovním klientovi Thunderbird.

4.1 Export a import SMS zpráv z poštovního klienta Thunderbird

Export souboru z poštovního klientovi Thunderbird vytvoří výstupní XML souboru s SMS zpráv naimportovaných a synchronizovaných v doplňku Thunderbird. Výstupní soubor je pak možné importovat jak do doplňku poštovního klienta Thunderbird tak do aplikace běžící na operačním systému Android. Při importování SMS zpráv bude doplněk v programu Thunderbird zpracovávat a procházet soubor typu XML, který bude zadán na jeho vstup. Soubor bude synchronizován s již existujícími SMS vlákny v poštovním klientu Thunderbird. Pokud bude při importování SMS zpráv příslušné SMS vlákno nalezeno, budou importovány pouze ty SMS zprávy, které jsou novější jak poslední SMS zpráva z daného vlákna. Na obrázku 4.1 je zobrazen diagram případu užití navrhovaného doplňku pro poštovního klienta Thunderbird.

4.2 Export a import SMS zpráv z operačního systému Android

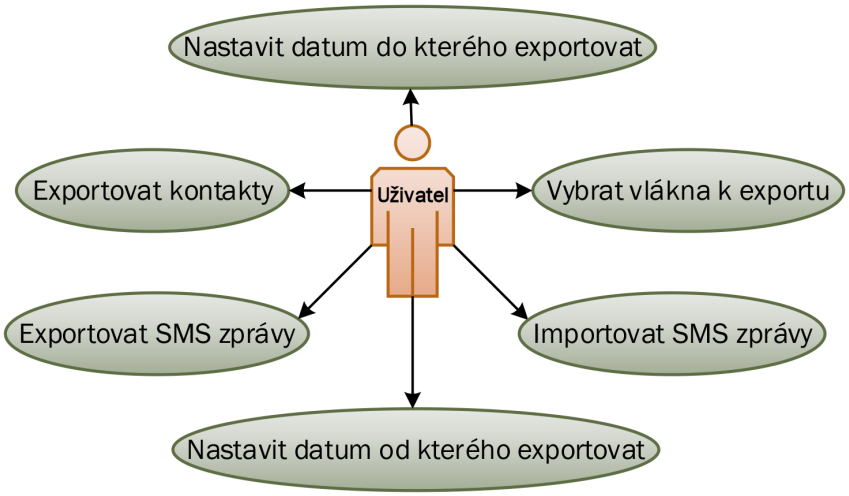
Při exportování SMS zpráv z telefonu s operačním systémem Android budou k dispozici funkce, které je možné vidět na obrázku 4.2. Při exportování SMS zpráv si uživatel bude moci vybrat, která SMS vlákna chce do výstupního exportovaného souboru zahrnout. Další



Obrázek 4.1: Diagram případu užití pro doplněk programu Thunderbird

kritérium, které bude možné při exportování SMS zpráv nastavit je časové rozmezí, ze kterého mají být dané SMS zprávy exportovány.

Při importování souboru bude aplikace Android procházet a analyzovat vstupní XML soubor. Důležitou funkčností při importování SMS zpráv do telefonu s operačním systémem Android bude synchronizace a přidávání SMS zpráv k již existujícím SMS zprávám. Při importování bude aplikace muset zajistit, aby importované SMS zprávy a SMS vlákna netvořily duplicity. Na obrázku 4.2 je zobrazen diagram případu užití navrhované aplikace pro operační systém Android.

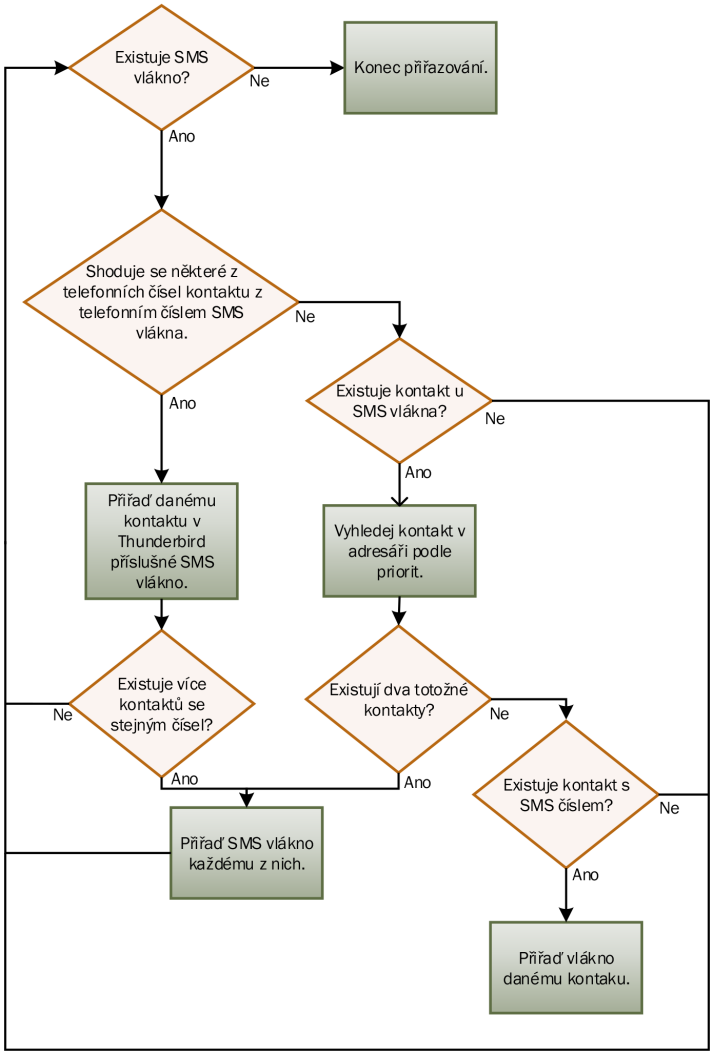


Obrázek 4.2: Diagram případu užití pro aplikaci operačního systému Android

4.3 Přiřazování SMS vláken daným kontaktům

Důležitou funkčností vyvíjeného doplňku pro poštovního klienta Thunderbird je přiřazování SMS vláken příslušným kontaktům. Doplněk bude spojovat kontakty a příslušná SMS

vlákna na základě shody stejných telefonních čísel u SMS vlákna a kontaktu v poštovním klientovi Thunderbird. Celý proces přiřazování kontaktů se bude řídit vývojovým diagramem na obrázku 4.3.



Obrázek 4.3: Vývojový diagram spojování kontaktů s SMS vlákny

4.4 Výstupní formát souboru

SMS zprávy zálohované pomocí vyvíjené aplikace pro operační systém Android budou exportovány jako jeden výstupní XML soubor. Pro více informací o výstupním souboru s kontakty viz. [36].

Vzhledem k rozsáhlému formátu XSD, který se běžně používá pro popis XML dokumentů, je v následujícím odstavci zobrazen pouze popis pomocí DTD. V popisu DTD jsou uvedeny všechny elementy a jejich atributy pro popis SMS zpráv a SMS vláken ve výstupním souboru.[1]

[1] XSD formát výstupního souboru je součástí přiloženého CD

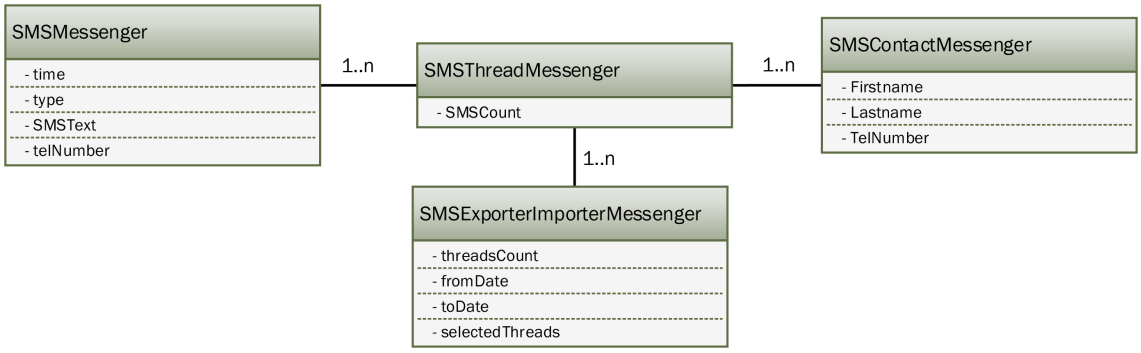
```
<!ELEMENT snippet (#PCDATA)>
<!ELEMENT sms_count (#PCDATA)>
<!ELEMENT contacts (#PCDATA)>
<!ATTLIST contacts
      thread_contact_id CDATA #REQUIRED
      tel_number CDATA #REQUIRED>
<!ELEMENT date (#PCDATA)>
<!ELEMENT text (#PCDATA)>
<!ELEMENT type (#PCDATA)>
<!ELEMENT tel_number (#PCDATA)>
<!ELEMENT contact EMPTY>
<!ATTLIST contact sms_contact_id CDATA #REQUIRED>
<!ELEMENT sms (date,text,type,tel_number,contact)>
<!ELEMENT thread (snippet,sms_count,(contacts)+,(sms)+)>
<!ATTLIST thread id CDATA #REQUIRED>
<!ELEMENT root (thread)+>
```

4.5 Návrh exportéra a importéra SMS zpráv

Aplikace se budou skládat z několika navzájem provázaných tříd. Základní třídu, která v sobě bude mít uchovány informace o příslušném vlákně se nazývá `SMSThreadMessenger`. Tato třída v sobě bude uchovávat jak informace o přijatých SMS zprávách tak informace o číslech, která v daném vlákně komunikují.

Objekt `SMSMessenger` bude uchovávat základní informace o dané SMS zprávě. Seznam nejdůležitějších informací, které bude objekt `SMSMessenger` obsahovat je možné vidět na obrázku 4.4.

Jestli bude dané číslo `SMSNumber` obsahovat také informace o příslušném kontaktu, bude tento kontakt uložen do výsledného objektu `SMSContactMessenger`. Ten bude obsahovat informace o příslušném kontaktu, který se podílel na konverzaci v daném SMS vlákně.



Obrázek 4.4: Diagram tříd pro exportování/importování SMS zpráv

Kapitola 5

Implementace

V této kapitole je popsána implementace aplikace pro operační systém Android a implementace doplňku do poštovního klienta Thunderbird. Aplikace pro operační systém Android je implementována v programovacím jazyce Java za pomocí nástrojů Android SDK ve vývojovém prostředí Android Studio. Aplikace je navržena a implementována pro Android 3.0 Honeycomb (API 11) a vyšší.

Doplněk pro poštovního klienta Thunderbird je implementován v programovacím jazyce Javascript pro poštovního klienta ve verzi 30.0.0 a vyšší. Doplněk je implementován ve vývojovém prostředí *NetBeans* za pomocí nástroje *foxbeans*.

Grafické uživatelské rozhraní je v případě aplikace pro operační systém Android implementováno za pomocí XML souborů. Uživatelské rozhraní pro doplněk poštovního klienta Thunderbird je implementována za pomocí XUL souborů, které mají vnitřní strukturu také definovanou pomocí XML.

5.1 Exportování SMS zpráv v Android aplikaci

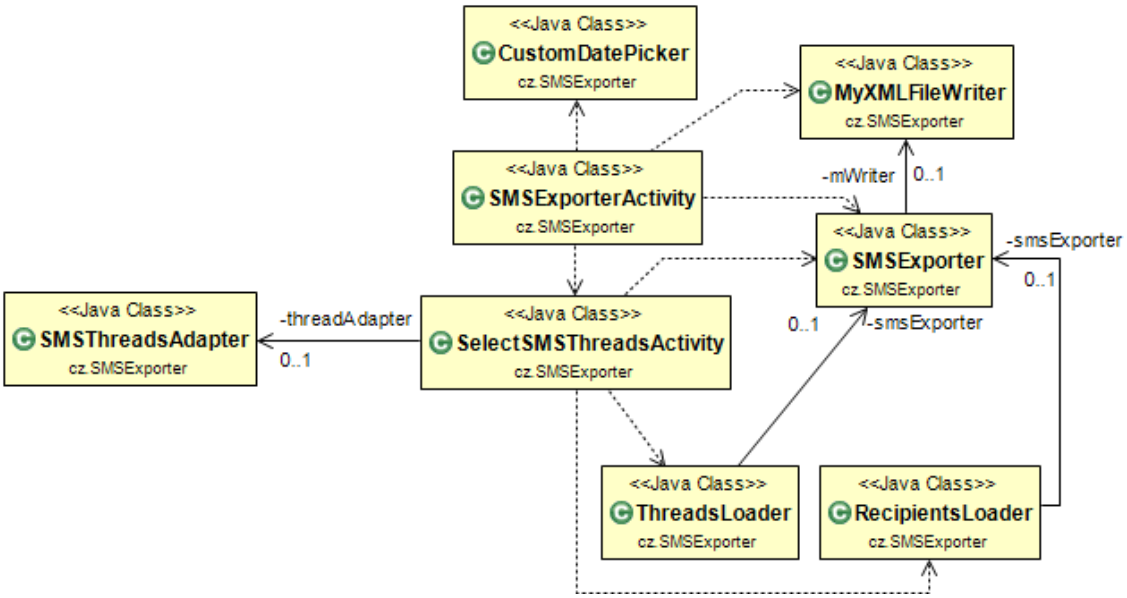
Exportování SMS zpráv a SMS vláken je klíčovou vlastností výsledné aplikace pro operační systém Android. Při exportování SMS zpráv je vytvořen výstupní XML soubor, který dodržuje strukturu podle definice DTD, která je uvedena v kapitole 4.4. Exportování a vytvoření výstupního XML souboru mají na starost třídy `MyXMLFileWriter`, `SMSExporterActitvity` a `SMSExporter`. Diagram tříd pro export SMS zpráv je na obrázku 5.1.

5.1.1 Třída *SMSExporter*

Třída `SMSExporter` obsahuje hlavní funkce pro exportování a sběr informací o SMS vláknech, SMS zprávách a kontaktech. Jelikož třída vykonává operace, které zapisují data do interní paměti telefonu a také provádí operace, které jsou výpočetně náročné, jsou tyto operace prováděny v odděleném vlákně. Aby mohly být operace prováděny v odděleném vlákně dědí třída `SMSExporter` ze třídy ***android.content.AsyncTask*** [26]. Třída potom přepisuje metodu ***loadInBackground***, která zajišťuje, že příkazy které budou vykonávány v této metodě, poběží v novém vlákně.

Při exportování SMS zpráv do výstupního souboru je jako první provedena metoda, která načte všechna SMS vlákna s příslušnými SMS zprávami do instance třídy `SMSExporterImporterMessenger`. Tyto informace jsou získávány z interní SQLite databáze

operačního systému Android (soubor *mmssms.db*). Přístup k databázi a databázové tabulce s SMS vlákny je zajištěn pomocí ***android.content.ContentResolver*** [30]. Tato třída zajišťuje přístup k datům za pomocí identifikátoru zdroje informací (*Unified Resource Identifier - URI*). V případě přístupu k SMS vláknům je to identifikátor *content://sms/conversations/*. Při získávání informací pomocí ***android.content.ContentResolver*** je také možné specifikovat, které položky a informace v databázové tabulce mají být při vyhledávání brány v úvahu. Výsledná data získaná za pomocí ***android.content.ContentResolver*** jsou uložena v instanci třídy ***android.database.Cursor***. Přístup k takto získaným datům se provádí posouváním ukazatele v instanci třídy ***android.database.Cursor*** na jednotlivé získané položky. Posouvání ukazatele je zajištěno pomocí iterace a volání metody ***moveToNext*** třídy ***android.database.Cursor*** [12]. Při každé iteraci je také vytvořena nová instance třídy `SMSThreadMessenger`, která uchovává informace o počtu SMS zpráv v příslušném SMS vlákně (`msg_count`), identifikátor SMS vlákna (`thread_id`) a také část poslední SMS zprávy v daném vlákně (`snippet`).



Obrázek 5.1: Diagram tříd pro exportování SMS zpráv

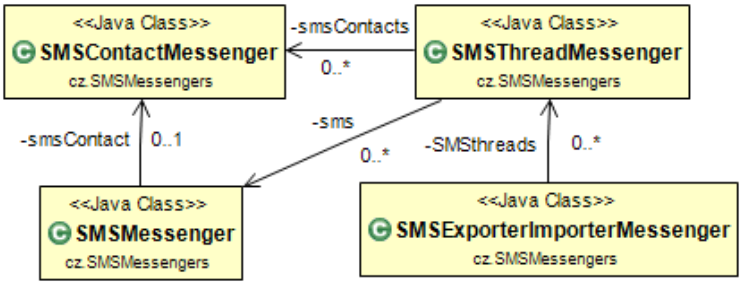
Data, která jsou uložena v instanci třídy `SMSThreadMessenger`, již obsahují všechna SMS vlákna, která jsou určena k exportu. Každé SMS vlákno však ještě neobsahuje informaci o tom, která čísla se účastní příslušné SMS konverzace. Načítání SMS čísel pro SMS vlákno je zajištěno pomocí metody `loadSMSNumbers`. Tato metoda pracuje z daty, která jsou přístupná přes URI `content://sms/`. Takto identifikovaná data obsahují informace o všech SMS zprávách v operačním systému Android. K těmto datům je možné přistoupit i v tomto případě pomocí ***android.content.ContentResolver*** a specifikovat jeho parametry tak, aby ve výsledné instanci třídy ***android.database.Cursor*** byla pouze informace s telefonním číslem k příslušnému SMS vláknu. Data poté uložíme do instance třídy `SMSThreadMessenger`. Jelikož má každá SMS zpráva uloženo číslo příjemce, musíme kontrolovat, zda již nemáme toto číslo uloženo ve výsledné instanci třídy `SMSThreadMessenger`, aby se netvořily duplicity stejných SMS čísel v jednom SMS vlákně.

Výstupní exportovaná data jsou v dalším kroku aktualizována o kontaktní informace,

které přísluší danému telefonnímu číslu. Načítání informací o kontaktech má na starost metoda `loadSMSRecipients`. Metoda pracuje obdobným způsobem, který byl uveden výše. Data jsou získávána pomocí URI `ContactsContract.PhoneLookup.CONTENT_FILTER_URI`, která má jako parametr zadáno telefoní číslo. Pokud byl podle zadaných informací nalezen nějaký kontakt je do exportovaného SMS vlákna uložen jeho identifikátor `_ID` a také jeho jméno a příjmení `DISPLAY_NAME`.

Když jsou načteny všechny informace týkající se SMS vláken je volána metoda pro načtení SMS zpráv a přiřazení těchto zpráv k daným SMS vláknům. K načtení dat je opět použita třída ***android.content.ContentResolver*** a stejný způsob získávání informací uvedený výše. Informace jsou získávány ze zdroje dat pomocí následující URI `content://sms/conversations/`. V URI je dále specifikováno identifikační číslo vlákna, pro které mají být SMS zprávy získány. Ze získaných informací o SMS zprávě si uložíme její text (`body`), datum přijetí SMS zprávy (`date`), informaci o tom zda je SMS zpráva příchozí či odchozí (`type`) a telefonní číslo, od kterého je SMS zpráva přijata (`address`).

Během získávání těchto SMS zpráv jsou také porovnávány příchozí časy jednotlivých SMS zpráv s časy nastavenými před zahájením exportu. SMS zpráva je přidána do výstupního souboru pouze v případě, pokud datum přijetí dané SMS zprávy náleží časovému intervalu, který byl nastaven před začátkem exportu.



Obrázek 5.2: Skladba objektů v aplikaci pro operační systém Android

V tomto okamžiku jsou již k dispozici všechna potřebná data a zbývá je jen zapsat do výstupního XML souboru. O zápis dat do XML souboru se starají metody třídy `MyXMLFileWriter` a jejich implementace je popsána v následující kapitole.

Při exportování dat do výstupního souboru je také uživateli zobrazen průběh exportování pomocí grafického prvku ***android.app.ProgressDialog***. Tento grafický prvek zobrazuje průběh exportování, který je aktualizován při každém zápisu SMS vlákna do výstupního souboru pomocí metody ***publishProgress***. Při každé aktualizaci tohoto grafického prvku je také změněno jméno kontaktu jehož vlákno je právě exportováno.

5.1.2 Třída *MyXMLFileWriter*

Třída `MyXMLFileWriter` má za úkol zapsat data získaná pomocí metod popsaných výše do výstupního souboru. O zápis dat z objektu `SMSExporterImporterMessenger` do výstupního souboru se stará třída ***org.xmlpull.v1.XmlSerializer*** [23]. Tato třída pomocí metod ***startTag***, ***endTag***, ***attribute*** a ***text*** zapíše a serializuje objekt `SMSExporterImporterMessenger` do výstupního souboru za pomocí kódování UTF-8. Výstupní soubor je vytvořen pomocí předpisu DTD, který je uveden v kapitole 4.4.

5.1.3 Serializace a desirializace Emoji

Emoji jsou piktogramy, které jsou typicky zobrazeny jako kreslené obrázky uvnitř textu [5]. Při serializaci těchto piktogramů pomocí třídy ***org.xmlpull.v1.XmlSerializer*** však nastává problém, piktogramy nejsou při serializaci rozpoznány jako validní symboly a XML serializér bere tyto znaky jako nevalidní. Tento problém nastává v situaci kdy je serializován piktogram, který je složen z více bajtů. Vzhledem k tomu, že kódování UTF-8 podporuje až 4 bytové znaky [52] nastává tento problém i v tomto případě.

Při serializaci dat je tedy v případě, že se v serializovaném textu vyskytne nějaký nevalidní znak, nutné tento znak rozpoznat a zakódovat pomocí metody *encodeNonValidCharacters* definovanou ve statické třídě *MyCustomFunctions*. Metoda kóduje nevalidní znaky a převádí je na reprezentaci dvou bajtové číselné hodnoty. Pro následné dekódování a rozpoznání této hodnoty je předcházena sekvencí znaků `\u`. Tato sekvence je rozpoznávána při převodu jednotlivých bajtů zpět do piktogramů Emoji.

Pro zpětný převod ze serializovaných dat na piktogramy Emoji slouží v aplikaci pro operační systém Android metoda `convertToEmoji` statiké třídy `MyCustomFunction`. U doplňku pro poštovního klienta Thunderbird je text dekódován funkcí `convertEmojiSequence`.

Metoda pro dekódování znaků v aplikaci operační systému Android převede nevalidní detekovaný znak na hodnotu anglicky označovanou jako *Surrogates pairs*. *Surrogates pairs* se skládá ze sekvence dvou dvoubajtových hodnot. Tyto hodnoty jsou vkládány do pole s celočíselnými hodnotami. Takto vytvořené pole celočíselných hodnot je poté za pomocí konstruktoru pro tvorbu řetězců převedeno na řetězec s piktogramem Emoji. Tato posloupnost výše popsaných úkonů probíhá tak dlouho, dokud je v textu nalezena aspoň jedna dvojce se zakódovanými hodnotami.

Funkce `convertEmojiSequence` v doplňku pro poštovního klienta pracuje obdobným způsobem jako metoda `convertToEmoji`. Piktogram Emoji je i v tomto případě vytvořen ze sekvence dvou dvoubajtových hodnot. V tomto případě je však použita vestavěná metoda `fromCharCode` objektu `String`. Tato metoda převede hodnotu v bajtech na odpovídající piktogram Emoji.

Pro správné zobrazení piktogramů Emoji na operačním systému Linux a starších operačních systémech od firmy Microsoft, je nutné nainstalovat font *Symbola*.

5.1.4 Třída *SMSExporterActitvity*

Třída `SMSExporterActitvity` dědí ze třídy ***android.app.Activity***. Kromě toho, že tato třída obsahuje grafické prvky typu ***Button*** pro zahájení exportu SMS zpráv a tlačítko pro návrat zpět z aktivity, tak obsahuje také tlačítka, která po stisknutí zobrazí dialogové okno s časovým údajem. Tento časový údaj slouží k vymezení časového intervalu exportovaný SMS zpráv ve výstupním souboru.

5.2 Zobrazování a načítání SMS vláken v Android aplikaci

Pro zobrazování a načítání SMS vláken slouží třídy `ThreadsLoader` a `RecipientsLoader`. Obě třídy dědí ze třídy ***android.content.AsyncTaskLoader***. Tato třída se stará o načítání dat v odděleném vlákně [32]. Třída tak zajišťuje, že zobrazení bude rychlejší a hlavně nebude zpomalovat vlákno, které má na starost zobrazování grafického uživatelského rozhraní.

Třída ***android.content.AsyncTaskLoader*** [27] dědí také ze třídy ***android.content.-Loader***. Tato třída slouží k načítání dat z určitého datového zdroje a v tomto konkrétním případě je zdrojem dat databáze SMS vláken a databáze kontaktů.

5.2.1 Třída *ThreadsLoader*

Třída `ThreadsLoader` dědí ze třídy ***android.content.AsyncTaskLoader***. Aby načítání dat probíhalo v odlišném vlákně je nutné algoritmus starající se o načítání SMS vláken z databáze přesunout do metody ***loadInBackground***. V této metodě je volána metoda `loadSMSRecipients`. Tato metoda je jedna z klíčových metod hlavní třídy pro exportování SMS zpráv `SMSExporter`. Metoda `loadSMSRecipients` přebírá jako jediný parametr objekt `SMSExporterImporterMessenger` a jako návratovou hodnotu vrací aktualizovaný objekt `SMSExporterImporterMessenger` s přidanými SMS vlákny a čísly, které k daným vláknům patří. Jelikož jsou v tomto okamžiku načtena pouze telefonní čísla, je nutné aby byly informace o SMS vláknech zobrazené uživateli aktualizovány o příslušná jména kontaktů, která přísluší danému SMS číslu. Třída `ThreadsLoader` a zobrazení jednotlivých telefonních čísel slouží k rychlejšímu zobrazení SMS vláken. Přímé zobrazení seznamu SMS vláken s načtenými jmény jednotlivých kontaktů je totiž příliš zdlouhavé, a proto je prováděno postupně a odděleně.

5.2.2 Třída *RecipientsLoader*

Načítání informací o kontaktech má za úkol zajistit třída `RecipientsLoader`. Třída opět dědí ze třídy ***AsyncTaskLoader***. Tato třída a její hlavní metoda ***loadInBackground***, ve které jsou načítány informace o kontaktech v SMS vlákně je volána hned po dokončení načítání všech SMS vláken s telefonními čísly. Instance třídy `SMSExporterImporterMessenger`, která již obsahuje SMS vlákna s příslušnými telefonními čísly, je nyní aktualizována pomocí metody `loadSMSRecipients` třídy `SMSExporterImporterMessenger`. Tato metoda opět vrací jako návratovou hodnotu instanci třídy `SMSExporterImporterMessenger` s již aktualizovanými informacemi o kontaktech u všech načtených SMS vláken. Po dokončení načítání všech dat je tedy uživateli zobrazen aktualizovaný seznam vláken, kde místo telefonních čísel jsou již zobrazena jména kontaktů, která přísluší daným telefonním číslům.

5.2.3 Třída *SelectSMSThreadsActivity*

O zobrazení načtených dat uživateli se stará třída `SelectSMSThreadsActivity`. Tato třída dědí ze třídy ***android.app.Activity*** a navíc se také stará o příjem *callback* metod ze tříd `RecipientsLoader` a `ThreadsLoader`. Aby mohly být tyto *callback* metody přijímány, je nutné aby třída implementovala rozhraní ***android.app.LoaderManager.LoaderCallbacks*** [22].

Při spouštění této aktivity jsou v metodě `onCreate` nastaveny všechny potřebné ovládací prvky pro komunikaci s grafickým uživatelským rozhraním. Pro rozložení všech prvků ve spouštěné aktivitě a zobrazení všech načtených SMS vláken je použit XML soubor `select_sms_threads_activity` ze složky `/res/layouts`. Tento soubor s grafickým rozložením všech prvků obsahuje kromě ovládacích prvků také základní grafický prvek typu ***ListView***. Tento grafický prvek v sobě uchovává data a informace o každém načteném SMS vlákně. Každé SMS vlákno je zobrazeno na samostatném řádku jako jedna položka grafického prvku ***ListView***.

Při implementování rozhraní ***android.app.LoaderManager.LoaderCallbacks*** je nejdůležitější *callback* metoda `onLoadFinished`. Metoda je volána v okamžiku, kdy jsou dostupná nová data ze tříd `RecipientsLoader` nebo `ThreadsLoader`. Data, která byla získána jednou z těchto tříd jsou poté použita v metodě `setData` třídy `SMSThreadsAdapter`, která je popsána v následující kapitole.

5.2.4 Třída *SMSThreadsAdapter*

Třída `SMSThreadsAdapter` dědí vlastnosti ze třídy ***android.widget.BaseAdapter***. Třída ***android.widget.BaseAdapter*** se stará zejména o úpravu a změnu grafického rozložení prvků v každé položce grafického prvku ***ListView*** [28]. Pro grafické rozložení prvků je podobně jako v případě `SelectSMSThreadsActivity` použit soubor `sms_thread_row_view` ze složky `/res/layouts`. Soubor obsahuje zejména grafické prvky pro uchovávání textu. Pro tento účel slouží v operačním systému Android prvek ***TextView*** [33]. Prvky ***TextView*** jsou v tomto případě použity pro zobrazení informací o kontaktech, které se účastní konverzace v daném SMS vlákně. Dále je zde grafický prvek s částí textu z poslední SMS zprávy v daném SMS vlákně a také grafický prvek, který obsahuje informaci o tom kolik SMS zpráv je v dané konverzaci. Mimo prvky pro uchování textu je zde také zaškrtávací políčko ***CheckBox***. ***CheckBox*** slouží pro zobrazení informace o tom, zda se má příslušné SMS vlákno zahrnout do výstupní exportovaného souboru.

Pro správné zobrazení všech prvků v ***ListView*** je nutné, aby byly přepsány metody `getCount`, `getItem`, `getItemId` a `getView` třídy ***android.widget.Adapter*** [24]. Z těchto uvedených metod jsou první tři zodpovědné za správné získávání právě označeného řádku v prvku ***ListView***. Metoda `getView` je nejdůležitější a stará se o naplnění každého řádku elementu ***ListView*** informacemi, které byly získány při načítání z databáze SMS zpráv a kontaktů.

5.3 Importování SMS zpráv v aplikaci pro operační Android

Při importování SMS zpráv a SMS vláken do operačního systému Android je kromě správné analýzy a uložení vstupního importovaného souboru, také důležité provést správně synchronizaci s již existujícími SMS zprávami a SMS vlákny. V následujících kapitolách bude tedy kladen důraz na popis algoritmu synchronizace. Bude také popsána funkčnost a implementace analyzátoru vstupního souboru.

Jelikož je při importování nutné zapisovat SMS zprávy do operačního systému Android, je kromě nastavení potřebných oprávnění v manifest souboru, také nutné nastavit aplikaci od verze operačního systému Android API 19 (Android 4.4 Kitkat) jako výchozí (více informací v kapitole 2.4.1). [1]

5.3.1 Třída *SMSImporter*

Třída `SMSImporter` provádí důležité algoritmy pro synchronizaci importovaného souboru s SMS zprávami a SMS vlákny v operačním systému Android. Podobně jako třída `SMSExporter` dědí třída `SMSImporter` ze třídy ***AsyncTask***. Tato dědičnost zaručuje, že prováděné operace, které slouží k synchronizaci, budou probíhat v odděleném výpočetním vlákně.

[1] Třídy potřebné pro zápis do databáze jsou uvedeny v balíčku `importerDefaultSMSApp`

Při importování SMS zpráv z externího souboru je na začátku soubor analyzován v metodě `parseXML` a informace z tohoto souboru jsou následně uloženy do instance třídy `SMSExporterImporterMessenger`. Více informací o analýze souboru a ukládání potřebných informací je uveden v kapitole 5.3.2.

Instance třídy `SMSExporterImporterMessenger`, která je již naplněna potřebnými daty je nyní v metodě `importSMSMessages` znovu analyzována a SMS zprávy a SMS vlákna jsou pomocí algoritmu zobrazeného na obrázku 5.3 importovány do operačního systému Android. V algoritmu pro importování souboru je kromě metod starajících se o synchronizaci `threadExists`, `hasSameNumbers`, `isNewerSmsMessage` také volána metoda pro vkládání SMS zpráv. SMS zprávy jsou vkládány do interní databáze pomocí ***android.content.ContentResolver*** [30].

5.3.2 Analýza vstupního importovaného souboru

Analýzu vstupního importovaného souboru a následné uložení všech potřebných informací má na starost třída `ElementXMLHandler`. Tato třída dědí vlastnosti ze třídy ***org.xml.sax.helpers.DefaultHandler***, která má na starost příjem *callback* metod ***startElement*** a ***endElement*** [31]. První ze zmíněných metod je volána v případě, kdy je při analýze vstupního souboru nalezen začínající element. Hodnota tohoto elementu je poté porovnávána se všemi možnými hodnotami počátečního elementu. Při shodě elementu je provedena příslušná operace a data jsou uložena do instance třídy `SMSExporterImporterMessenger`.

Metoda ***endElement*** je volána a prováděna v době, kdy je nalezen ukončovací element. Proces vyhledání a práce s daty je obdobná jako při volaní metody ***startElement***.

5.3.3 Zobrazení exportovaných souborů

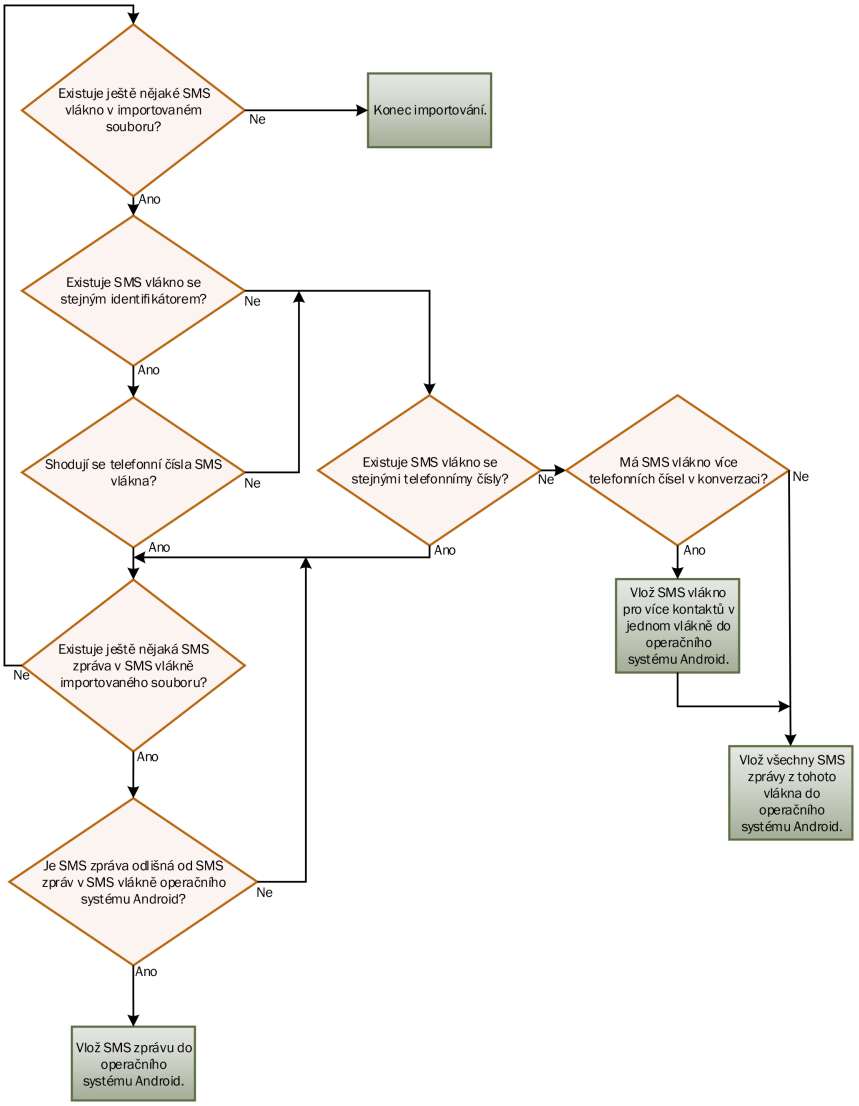
Pro zobrazení exportovaných souborů pomocí aplikace pro operační systém Android je vytvořena Aktivita implementována pomocí třídy `ImportSMSFilePickerActivity`. Pomocí této Aktivity je možné prohlížet exportované soubory. Aktivita je nastavena tak, aby se zobrazila v návrhu aplikací, které umožňují zobrazení a následný výběr souborů. Implementace této Aktivity ve výsledné aplikaci tedy uživatele oprostí od nutnosti mít nainstalovaný jakýkoliv prohlížeč souborů.

Aktivita pro zobrazení položek seznamu exportovaných souborů má definované grafické rozhraní pro jednotlivé položky v souboru `file_row_view` (složka `res/layout`). Každá položka obsahuje informaci o názvu exportovaného souboru a čase jeho vytvoření. Načítání souborů je prováděno v odděleném výpočetním vlákně za pomocí třídy `FilesLoader`. Tato třída analyzuje složku s exportovanými soubory a poté vytvoří seznam všech exportovaných souborů. Vzhledem k tomu, že se do této složky ukládají i exportované soubory s kontakty, jsou zobrazeny jen ty soubory, které mají příponu *.xml*.

5.4 Importování SMS zpráv v programu Thunderbird

Při importování vstupního souboru s SMS zprávami a SMS vlákny do doplňku poštovního klienta Thunderbird je v případě, že se jedná o první export vytvořen soubor `sms.xml` v klientské složce programu Thunderbird. XML soubor sms.xml je hlavním souborem, který obsahuje všechny importované SMS zprávy a SMS vlákna.

Import souboru, který je proveden v době, kdy již existuje soubor *sms.xml* je proveden pomocí funkce `importSMSMessages`.

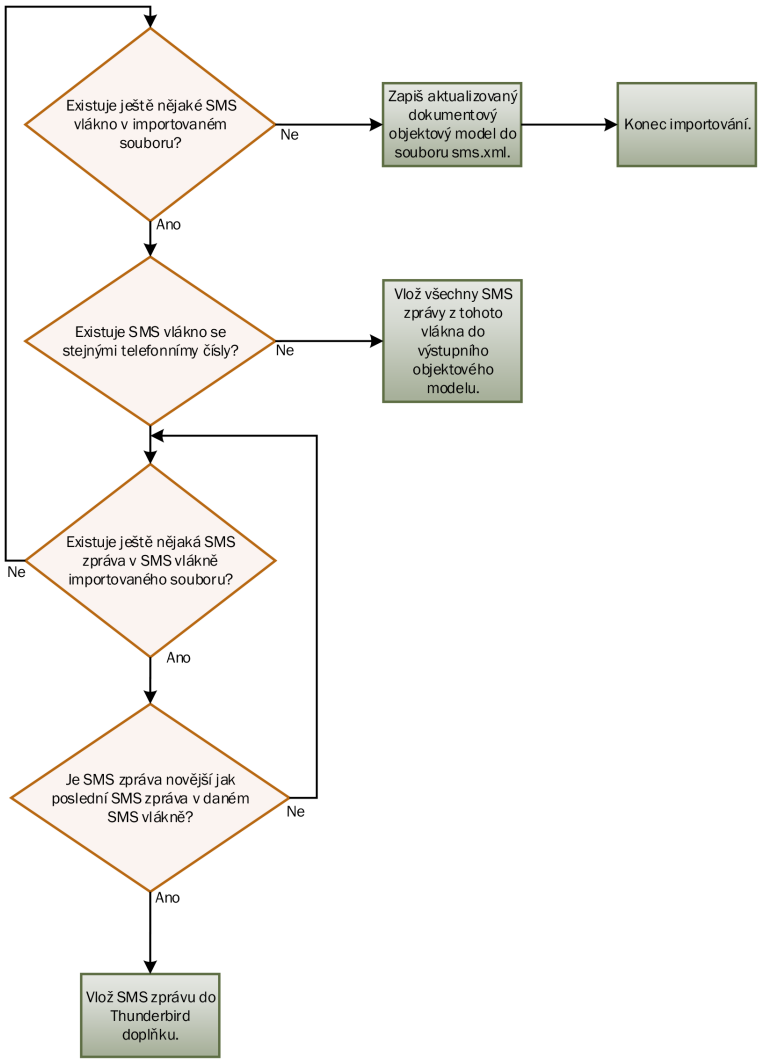


Obrázek 5.3: Algoritmus pro synchronizaci SMS zpráv v operačním systému Android

5.4.1 Soubor importSMS.js

Soubor `importSMS.js` obsahuje funkce pro importování a synchronizaci SMS zpráv a SMS vláken v doplňku poštovního klienta Thunderbird. Při procesu importování je vstupní soubor analyzován pomocí objektu ***DOMParser*** a jeho metody ***parseFromString***. Metodě ***parseFromString*** jsou předány data ze vstupního souboru a jako výstup této metody je vytvořen objektový model analyzovaného souboru.

Objektový model vstupního souboru je následně analyzován a porovnáván s již existujícím dokumentovým objektovým modelem ze souboru `sms.xml`. Porovnávání a import probíhá podle obrázku 5.4. O synchronizaci se starají funkce `findThreadByTelNumber`, `comapreSMSDateTime`, `addNewerSMS`, `mergeSMSMessages` a `writeDomToFile`.



Obrázek 5.4: Algoritmus pro synchronizaci SMS zpráv v poštovním klientovi Thunderbird

5.5 Zobrazování SMS zpráv a SMS vláken v programu Thunderbird

O zobrazení všech SMS zpráv a SMS vláken z uloženého souboru se starají funkce v souboru `ThreadsView.js`. Při zobrazení SMS zpráv a SMS vláken je při analýze dokumentového objektového modelu, který je vytvořen ze souboru *sms.xml* vytvářen seznam všech SMS vláken v grafickém prvku ***vbox***. Každé SMS vlákno je reprezentováno jako jeden grafický prvek ***hbox***, který obsahuje prvek ***description*** a ***image***. V grafickém prvku ***description*** je pak zobrazen text obsahující telefonní čísla nebo jména z daného SMS vlákna. Při kliknutí na jeden z grafických prvků ***hbox*** je volána funkce loadSMSonclick, které je předáno identifikační číslo SMS vlákna, na které uživatel provedl operaci kliknutí. Podle identifikačního čísla SMS vlákna jsou vyhledány všechny SMS zprávy a následně jsou zobrazeny v grafickém prvku ***vbox*** s identifikátorem `sms`. Zprávy SMS jsou pro lepší orientaci také barevně rozlišeny podle jejich typu (příchozí, odchozí).

5.6 Přiřazování SMS vláken existujícím kontaktům

O přiřazování SMS vláken existujícím kontaktům v poštovním klientovi Thunderibrd se starají funkce v souboru `ContactsView.js`. SMS vlákna jsou kontaktům přiřazována na základě telefonních čísel, které jsou u nich uloženy. Podle telefonních čísel jsou poté při kliknutí na jednotlivé kontakty vyhledány a následně zobrazeny všechny relevantní SMS vlákna uložená v poštovním klientovi Thunderbird.

Při zobrazení a přiřazování SMS zpráv kontaktům jsou pomocí funkce ***getAllAddressBooks()*** získány všechny adresáře s kontakty. Nalezené adresáře s kontakty jsou poté vloženy a zpracovány tak, aby vznikl jeden seznam všech kontaktů ze všech adresářů s kontakty.

Při vyhledávání SMS vláken pro daný kontakt jsou v každé zobrazené položce kontaktu (grafický prvek ***hbox***) a jeho parametru `id` čárkou odděleny telefonní čísla z položek `WorkPhone`, `HomePhone` a `CellularNumber` daného kontaktu. Telefonní čísla jsou poté předány do funkce `loadSMSThreadsOnclick`, která je volána při kliknutí na daný kontakt. V pravé části obrazovky se poté zobrazí importovaná SMS vlákna podle SMS čísel ve vybraném kontaktu.

5.7 Vyhledávání SMS zpráv v programu Thunderbird

Při vyhledávání SMS zpráv je možné filtrovat SMS zprávy podle textu v SMS zprávě. Je možné použít hledání ve všech SMS zprávách a nebo pouze v SMS zprávách, které jsou právě zobrazeny. Při vyhledávání je také možnost aplikovat filtr na zobrazené kontakty v SMS vláknech.

Pro vyhledávání SMS zpráv jsou vytvořeny dvě funkce `filterRecords` a `searchSMS` v souboru `search.js`. Funkce `filterRecords` filtruje právě zobrazené grafické prvky a to buďto v rámci jména, příjmení a telefonních čísel kontaktů SMS vláken a nebo v rámci právě zobrazených SMS v příslušném zobrazeném vlákně. Druhá z funkcí pro vyhledávání `searchSMS` slouží ke hledání SMS zpráv v rámci všech SMS zpráv. Při hledání pomocí této funkce je prohledáván celý dokumentový objektový model z uloženého souboru v poštovním klientovi Thunderbird.

Při změně obsahu grafického prvku ***textbox*** je volána funkce `searchSMS`, která zobrazí a vyhledá příslušné SMS zprávy odpovídající zadanému textu. Pro shodu textu je využito metody ***indexOf***, která je definována nad textovými řetězci. Tento způsob hledání je použit i v případě vyhledávání pomocí metody `filterRecords`.

5.8 Exportování SMS zpráv v programu Thunderbird

Při exportování SMS zpráv je provedena serializace dokumentového objektového modelu importovaných SMS zpráv a SMS vláken. Export je proveden ve funkci `exportSMSFile` v souboru `exportSMS.js`. Funkce `exportSMSFile` provede serializaci dokumentového objektového modelu pomocí metody ***serializeToString*** objektu ***XMLSerializer***. Serializovaná data jsou poté uložena do příslušného souboru, jehož jméno a místo k uložení určí uživatel. Kódování pro importovaný soubor je neměnné a je nastavené na kódování textu pomocí *UTF-8*.

Kapitola 6

Testování

Aplikace pro operační systém Android a doplněk pro poštovního klienta Thunderbird byly testovány jak během samotného vývoje, tak i po dokončení jejich implementace.

6.1 Testování aplikace pro operační systém Android

Při testování aplikace pro operační systém Android byly použity softwarové nástroje z vývojového prostředí Android Studio a také hardwarová zařízení s operačním systémem Android. Testování probíhalo převážně na operačních systémech Android od verze 4.0 a výše (zařízení Samsung Galaxy S2). Vzhledem k tomu, že od verze Android 4.4 je v operačním systému zavedena jiná konvence pro ukládání SMS zpráv, byla aplikace také testována na operačním systému Android ve verzi 4.4 (zařízení LG F60).

Jelikož aplikace zapisuje při importování SMS zpráv data do databáze operačního systému, byla na hardwarových telefonech testována také správná struktura nahraných a uložených dat. Pro přístup k takto uloženým datům a jejich prohlížení bylo nutné získat oprávnění uživatele *root*. Samotné prohlížení a kontrola jednotlivých záznamů a dat byla prováděna pomocí aplikace *SQLite Debugger*.

Testování pomocí emulátoru a nástrojů v Android SDK probíhalo pomocí vizuální kontroly importovaných a exportovaných dat pomocí výchozí aplikace pro prohlížení SMS zpráv *Messaging*. Jelikož je v softwarových nástrojích velice obtížné získat oprávnění uživatele *root* a tudíž prohlížet SMS zprávy pomocí aplikace *SQLite Debugger* přímo v databázi operačního systému, byl pro testování nainstalován doplněk pro Android Studio *Genymotion*, který nabízí ihned po instalaci softwarový emulátor s oprávněními uživatele *root*.

Testování během implementace aplikace probíhalo vždy po dokončení jednotlivých funkcí aplikace (filtrování vláken, výběr datového intervalu atd.).

6.1.1 Aplikace pro operační systém Android – Export

Testování funkce exportu bylo prováděno následujícím způsobem:

1. Spuštění aplikace na zařízení s operačním systémem Android.
2. Výběr SMS vláken a následné nastavení intervalu jednotlivých kalendářních dat (od/do), ve kterém mají být dané SMS zprávy exportovány.
3. Proveden export.

4. Exportovaný soubor byl přenesen do počítače pomocí sběrnice USB a následně otevřen pomocí webové prohlížeče Mozilla Firefox, který umožňoval snadnější vizuální kontrolu.

5. Porovnávání přenesených dat zobrazených ve webovém prohlížeči a dat zobrazených v aplikaci Messaging v operačním systému Android.

Testování funkce pro exportování SMS zpráv probíhalo podle následujících testovacích případů: [1]

- Export všech SMS vláken.
- Export vybraných SMS vláken.
- Export všech vláken s možností nastaveného času.
- Export vybraných SMS vláken s možností nastaveného času.
- Export SMS zpráv obsahujících piktogramy *Emoji*.

6.1.2 Aplikace pro operační systém Android – Import

Testování funkce import bylo prováděno následujícím způsobem:

1. Spuštění aplikace na zařízení s operačním systémem Android.
2. Výběr souboru, který byl určen pro následný import do telefonu.
3. Proveden import vybraného souboru.
4. Vizuální kontrola přenesených dat pomocí aplikace Messaging. Zvláštní důraz byl kladen na kontrolu duplicitních SMS zpráv či SMS vláken.

Testování funkce pro importování SMS zpráv probíhalo podle následujících testovacích případů: [1]

- Import neexistujícího SMS vlákna.
- Import existujícího SMS vlákna se stejnými SMS zprávami.
- Import existujícího SMS vlákna s různými SMS zprávami (datum, typ, text).
- Tři výše zmíněné body s piktogramy *Emoji* v textu zprávy.
- Import SMS zpráv s nevalidním datem (nastaven den importu).
- Import SMS zpráv s nevalidní sekvencí bytů pro zápis piktogramů *Emoji*.
- Import SMS zpráv s nevalidním typem SMS zprávy (SMS zpráva nastavena jako příchozí).

[1] Přiložené CD k této bakalářské práci obsahuje testovací protokol se vstupními soubory a jednotlivými testovacími případy.

6.2 Testování doplňku poštovního klienta Thunderbird

Testování implementovaného doplňku probíhalo na poštovním klientovi Thunderbird ve verzi 31.6.0. Poštovní klient byl instalován na operačním systému Windows 8.1. Při testování byl převážně testován algoritmus, který provádí import souboru s SMS zprávami a také byla testována následná synchronizace s již existujícími SMS zprávami. Testováno bylo také vyhledávání v SMS zprávách a kontaktech v poštovním klientovi Thunderbird.

Testování funkce importování v poštovním klientovi probíhalo následovně. Z telefonu s operačním systémem Android byl za pomocí aplikace vytvořen výstupní XML soubor. XML soubor byl dále překopírován pomocí sběrnice USB do počítače. V doplňku do poštovního klienta Thunderbird byla vybrána možnost pro importování XML souboru a poté byl vybrán překopírovaný soubor. Data se dále nahrála do doplňku poštovního klienta Thunderbird. Po takto nahraných datech, byla provedena kontrola a shodnost zobrazených dat v mobilním telefonu a v poštovním klientovi Thunderbird.

6.2.1 Doplněk poštovního klienta Thunderbird Export/Import

Testování výsledné aplikace probíhalo podle následujících testovacích případů: [1]

- Import XML souboru při prvním spuštění doplňku (žádná data v doplňku).
- Import XML souboru s novějšími SMS zprávami (porovnávání kalendářních dat u SMS zpráv).
- Import XML souboru se stejnými SMS zprávami (stejná kalendářní data jak v importovaném souboru tak v existujících datech).
- Import XML souboru s žádnými novými zprávami.
- Import XML souboru, který obsahuje nové neexistující SMS vlákno.

Vzhledem k tomu, že exportování bylo prováděno za pomocí vestavěné funkce, nebylo nutné tuto funkčnost důkladně testovat.

[1] Přiložené CD k této bakalářské práci obsahuje testovací protokol se vstupními soubory a jednotlivými testovacími případy.

Kapitola 7

Závěr

V rámci této bakalářské práce byly implementovány aplikace, které mají za úkol zejména zálohovat a obnovovat SMS zprávy. Velkou výhodou je také možnost prohlížet SMS zprávy na osobním počítači a provádět synchronizaci již importovaných souborů s nově importovanými SMS zprávami. Při prohlížení SMS zpráv mají uživatelé možnost v zobrazených SMS zprávách vyhledávat. Aplikace také umožňuje přiřadit příslušná vlákna kontaktům, které jsou uloženy v poštovním klientovi Thunderbird. V synchronizovaných SMS vláknech a kontaktech je tak možné vyhledávat i v případě, kdy daný kontakt nemáme v mobilním zařízení.

Možnost dalšího vývoje je velice rozsáhlý a to hlavně díky výstupnímu exportovanému souboru s SMS zprávami. Jedna z možností je implementace programu, který umožní editaci výstupního souboru. Vzhledem k tomu, že aplikace pro operační systém Android umožňuje zálohovat pouze SMS zprávy, bylo by vhodné vytvořit exportování a importování MMS zpráv. Aplikaci v operačním systému Android by bylo také možné rozšířit o automatickou aktualizaci exportovaného souboru při nově příchozí SMS zprávě. Synchronizace, která je v tomto okamžiku implementována pomocí nahrávání souborů by mohla být řešena automaticky pomocí dat, která budou ukládána na server a z tohoto serveru pak budou data stahována a synchronizována s jiným telefonem či aplikací pro stolní počítače (doplněk poštovního klienta Thunderbird).

Literatura

[1] Allen, G.: *Android 4 Průvodce programováním mobilních aplikací*. Computer Press, a.s Brno, 2013 [cit. 2014-12-17], ISBN 9788025137826.

[2] Baker, M.: Building and Running Overview [online]. https://blog.lizardwrangler.com/2012/07/06/thunderbird-stability-and-community-innovation/, 2015 [cit. 2015-04-27].

[3] Bornstein, D.: Dalvik VM Internals [video]. https://www.youtube.com/watch?v=ptjedOZEXPM, 2008 [cit. 2014-12-17].

[4] Brady, P.: Google I/O 2008 - Anatomy and Physiology of an Android [video]. https://www.youtube.com/watch?v=G-36noTCaiA, 2008 [cit. 2015-4-3].

[5] Davis, M.: Draft Unicode Technical Report 51 [online]. http://unicode.org/reports/tr51/, 2015 [cit. 2015-5-7].

[6] Ehrenmüller, O.: Google Play - SQLite Debugger[online]. https://play.google.com/store/apps/details?id=oliver.ehrenmueller.dbadmin&hl=cs, 2014 [cit. 2015-04-27].

[7] eLinux: Android Kernel Features [online]. http://elinux.org/Android_Kernel_Features, 2013 [cit. 2014-12-17].

[8] Fossil: SQLite Android Bindings [online]. https://www.sqlite.org/android/doc/trunk/www/index.wiki, 2014 [cit. 2014-12-17].

[9] Gargenta, M.: Learn about Android Internals and NDK [video]. https://www.youtube.com/watch?v=byFTAhXVF7k, 2010 [cit. 2014-12-17].

[10] Google: Android Developers - Android NDK [online]. https://developer.android.com/tools/sdk/ndk/index.html, 2014 [cit. 2014-12-17].

[11] Google: Android Developers - Application Fundamentals [online]. https://developer.android.com/guide/components/fundamentals.html, 2014 [cit. 2014-12-17].

[12] Google: Android Developers - Cursor [online]. http://developer.android.com/reference/android/database/Cursor.html, 2014 [cit. 2014-12-17].

[13] Google: Android Developers - OpenGL ES [online]. http://developer.android.com/guide/topics/graphics/opengl.html, 2014 [cit. 2014-12-17].

[14] Google: Android Developers - Pausing and Resuming an Activity [online]. http://developer.android.com/training/basics/activity-lifecycle/pausing.html, 2014 [cit. 2014-12-17].

[15] Google: Android Developers - SDK [online]. https://developer.android.com/sdk/index.html, 2014 [cit. 2014-12-17].

[16] Google: Android Developers - Starting an Activity [online]. http://developer.android.com/training/basics/activity-lifecycle/starting.html, 2014 [cit. 2014-12-17].

[17] Google: Android Developers - Stopping and Restarting an Activity [online]. http://developer.android.com/training/basics/activity-lifecycle/stopping.html, 2014 [cit. 2014-12-17].

[18] Google: Android (operating system) [online]. http://en.wikipedia.org/wiki/Android_(operating_system), 2014 [cit. 2014-12-17].

[19] Google: skia - 2D Graphics Library[online]. https://code.google.com/p/skia/, 2014 [cit. 2014-12-17].

[20] Google: Android Developers - BroadcastReceiver [online]. http://developer.android.com/reference/android/content/BroadcastReceiver.html, 2014 [cit. 2014-12-18].

[21] Google: Android Developers - ContentProvider [online]. http://developer.android.com/reference/android/content/ContentProvider.html, 2014 [cit. 2014-12-18].

[22] Google: Android Developers - LoaderCallbacks[online]. http://developer.android.com/reference/android/app/LoaderManager.LoaderCallbacks.html, 2014 [cit. 2015-04-27].

[23] Google: Android Developers - XmlSerializer[online]. http://developer.android.com/reference/org/xmlpull/v1/XmlSerializer.html, 2014 [cit. 2015-04-27].

[24] Google: Android Developers - Adapter [online]. http://developer.android.com/reference/android/widget/Adapter.html, 2015 [cit. 2015-04-27].

[25] Google: Android Developers - App Manifest [online]. http://developer.android.com/guide/topics/manifest/manifest-intro.html, 2015 [cit. 2015-04-27].

[26] Google: Android Developers - AsyncTask [online]. http://developer.android.com/reference/android/os/AsyncTask.html, 2015 [cit. 2015-04-27].

[27] Google: Android Developers - AsyncTaskLoader [online]. http://developer.android.com/reference/android/content/AsyncTaskLoader.html, 2015 [cit. 2015-04-27].

[28] Google: Android Developers - BaseAdapter [online]. http://developer.android.com/reference/android/widget/BaseAdapter.html, 2015 [cit. 2015-04-27].

[29] Google: Android Developers - Building and Running Overview [online]. https://developer.android.com/tools/building/index.html, 2015 [cit. 2015-04-27].

[30] Google: Android Developers - ContentResolver [online]. http://developer.android.com/reference/android/content/ContentResolver.html, 2015 [cit. 2015-04-27].

[31] Google: Android Developers - DefaultHandler [online]. http://developer.android.com/reference/org/xml/sax/helpers/DefaultHandler.html, 2015 [cit. 2015-04-27].

[32] Google: Android Developers - Loaders [online]. http://developer.android.com/guide/components/loaders.html, 2015 [cit. 2015-04-27].

[33] Google: Android Developers - TextView [online]. http://developer.android.com/reference/android/widget/TextView.html, 2015 [cit. 2015-04-27].

[34] Google: Android - Media [online]. https://source.android.com/devices/media.html, 2015 [cit. 2015-4-3].

[35] John, S. P.: Android Architecture [online]. http://www.eazytutz.com/android/android-architecture/, 2015 [cit. 2015-4-3].

[36] Kasala, V.: Nástroje pro zpracování kontaktů na platformě Android. Brno, 2012. Bakalářská práce. Vysoké Učení Technické v Brně, Fakulta Informačních technologií. [cit. 2014-02-01].

[37] Lemberg, W.: Free Type[online]. http://www.freetype.org/, 2014 [cit. 2014-12-17].

[38] Main, S.: Getting Your SMS Apps Ready for KitKat [online]. http://android-developers.blogspot.cz/2013/10/getting-your-sms-apps-ready-for-kitkat.html, 2013 [cit. 2014-12-17].

[39] McCusker, D.: Mork Structure [online]. https://developer.mozilla.org/en-US/docs/Mork_Structure, 2012 [cit. 2014-12-20].

[40] McFarlane, N.: *Rapid Application Development with Mozilla*. Prentice Hall PTR 2003, 2003 [cit. 2015-04-27], ISBN 9780131423435.

[41] Mednieks, Z.: *Programming Android*. O'Reilly Cambridge, 2011 [cit. 2014-12-17], ISBN 978-1-449-38969-7.

[42] Mills, C.: Address Book examples [online]. https://developer.mozilla.org/en-US/docs/Mozilla/Thunderbird/Address_Book_Examples, 2013 [cit. 2015-04-27].

[43] MozillaZine: Getting started with Thunderbird [online]. http://kb.mozillazine.org/Getting_started_with_Thunderbird, 2014 [cit. 2015-05-13].

[44] Nolan, G.: *Decompiling Android [online]*. Apress, 2012 [cit. 2014-12-17], ISBN 978-1-4302-4248-2.

[45] Project, T. W. O. S.: The WebKit Open Source Project [online]. https://www.webkit.org/, 2014 [cit. 2014-12-17].

[46] Schmidt, D.: Android Services and Local IPC [video]. https://www.youtube.com/watch?v=gxj4sQX9m5g, 2013 [cit. 2014-12-17].

[47] Schmidt, D.: Overview of Android Components: Broadcast Receivers [video]. https://www.youtube.com/watch?v=GZmM_x-PJBY, 2014 [cit. 2014-12-18].

[48] Schmidt, D.: Overview of Android Components: Content Providers [video]. https://www.youtube.com/watch?v=b3m9UIeSpBY, 2014 [cit. 2014-12-18].

[49] Schmidt, D.: Lecture 24: Android Persistent Storage and Content Providers (parts 1 and 2) [video]. https://www.youtube.com/watch?v=IWP2-qkhtiM, 2014 [cit. 2014-12-19].

[50] Shepherd, E.: nsIAbCard/Thunderbird3 [online]. https://developer.mozilla.org/en-US/docs/nsIAbCard_Thunderbird3, 2008 [cit. 2015-04-27].

[51] mozilla wiki: Address Book [online]. https://wiki.mozilla.org/Address_Book, 2014 [cit. 2015-04-27].

[52] Yergeau, F.: UTF-8, a transformation format of ISO 10646 [online]. https://tools.ietf.org/html/rfc3629, 2003 [cit. 2015-4-3].

Seznam použitých zkratek a symbolů

ADT *Android Development Tools* – Doplněk pro vývojové prostředí Eclipse. Přidává možnost práce s Android projekty.

API *Application Programming Interface* – Rozhraní pro programování aplikací.

CORBA *Common Object Request Broker Architecture* – Standard definující komunikaci mezi systémy, které byly vyvinuty na různých platformách.

CSV *Comma-separated values* – Jednoduchý souborový formát určený pro výměnu tabulkových dat, data oddělena čárkami.

DOM *Document Object Model* – Objektově orientovaná reprezentace XML nebo HTML dokumentu.

DTD *Document Type Definition* – Jazyk pro popis struktury XML dokumentu.

DVM *Dalvik Virtual Machine* – Virtuální stroj běžící na operačním systému Android optimalizovaný pro mobilní zařízení.

GPL *General Public License* – Licence pro svobodný software.

IDL *Interface description language* – Jazyk slouží pro popis rozhraní softwarových komponent.

JVM *Java Virtual Machine* – Virtuálního stroje ke spuštění programů a skriptů vytvořených v programovacím jazyce Java.

LIFO *Last in, first out* – Abstraktní datová struktura typu zásobník pro ukládání dat.

MMS *Multimedia Messaging Service* – Služba multimediálních zpráv.

NDK *Native Development Kit* – Sada nástrojů pro vývoj aplikací pro operační systém Android, za pomocí nativních knihoven C/C++.

SDK *Software Development Kit* – Sada nástrojů pro vývoj aplikací pro operační systém Android.

SMS *Short message service* – Služba krátkých textových zpráv.

SQL *Structured Query Language* – Dotazovací jazyk, který je používán pro práci s daty v relačních databázích.

SSL *Secure Sockets Layer* – Protokol, který poskytuje zabezpečení komunikace šifrováním a autentizací.

URI *Uniform Resource Identifier* – Textový řetězec s definovanou strukturou, který slouží k přesné specifikaci zdroje informací.

USB *Universal Serial Bus* – Univerzální sériová sběrnice, která umožňuje připojit různé periferie k počítači.

UTF-8 *UCS Transformation Format* – Způsob kódování řetězců *UNICODE*, v tomto případě na 1 bajt.

XML *Extensible Markup Language* – Obecný značkovací jazyk, který byl vyvinut a standardizován konsorciem W3C.

XUL *XML User Interface Language* – Jazyk sloužící pro tvorbu uživatelských rozhraních, založen na XML.

Seznam příloh

- Atributy databázových tabulek pro práci s SMS zprávami
- Obsah CD
- Manuál

Příloha A

Atributy databázových tabulek pro práci s SMS zprávami

Atribut	Datový typ
_id	INTEGER
data	INTEGER
error	INTEGER
has_attachment	INTEGER
message_count	INTEGER
read	INTEGER
recipient_ids	TEXT
snippet	TEXT
snippet_cs	INTEGER

Tabulka A.1: Kompletní seznam všech parametrů databázové tabulky pro uložení SMS vláken

Atribut	Datový typ
_id	INTEGER
address	TEXT
body	TEXT
date	INTEGER
date_sent	INTEGER
error_code	INTEGER
locked	INTEGER
person	INTEGER
read	INTEGER
replay_path_present	INTEGER
seen	INTEGER
service_center	TEXT
subject	TEXT
status	INTEGER
thread_id	INTEGER

Tabulka A.2: Kompletní seznam všech parametrů databázové tabulky pro uložení SMS zprávy

Příloha B

Obsah CD

Přiložené CD obsahuje následující materiály

- **Písemná zpráva** – *zprava.pdf*
- **Programová dokumentace** – *android_doc/*
- **Spustitelné aplikace** – *apps/*
- **Testy a testovací protokol** – *tests/*
- **Uživatelská příručka** – *manual.pdf*
- **Zdrojové soubory** – *source_codes/*
- **Zdrojový tvar písemné zprávy** – *tex/*

Příloha C

Manuál

C.1 Instalace aplikace pro operační systém Android

Pro instalaci aplikace pro operační systém Android je nutné vlastnit zařízení, na kterém běží verze Android 3.0 a vyšší. Vzhledem k tomu, že aplikace není dostupná z oficiálního internetového obchodu s aplikacemi *Play Store*, je nutné aby měl uživatel povoleny instalace aplikací z Neznámých zdrojů (Nastavení – Zabezpečení – Neznámé zdroje). Instalace aplikace probíhá podle následujících kroků:

1. Překopírovat aplikaci SMS Export - Import ze složky *apps* (součást CD přiložené k této bakalářské práci) do zařízení s operačním systémem Android.

2. V zařízení s operačním systémem Android nainstalovat aplikaci SMS Export - Import (aplikace je umístěna ve složce, kterou jsme zvolili při kopírování z CD).

C.2 Instalace doplňku pro poštovního klienta Thunderbird

Pro instalaci doplňku do poštovního klienta Thuderbird je nutné mít nainstalovanou verzi klienta 30.0.0 a vyšší. Při instalaci postupujeme dle následujících kroků:

1. Překopírovat doplněk SMS Extension ze složky *apps* (součást CD přiložené k této bakalářské práci) do počítače, ve kterém je nainstalován poštovní klient Thunderbird.

2. V poštovním klientovi Thunderbird vybrat možnost Nástroje – Správce doplňků a dále kliknout na tlačítko Nástroje doplňků a vybrat možnost Instalovat doplněk ze souboru. Doplněk je umístěn ve složce, kterou jsme zvolili při kopírování doplňku z CD.

3. Spustitelná verze doplňku se po restartování poštovního klienta Thunderbird objeví po kliknutí na možnosti Nástroje v horním části poštovního klienta pod názvem *SMS Extension*.