VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

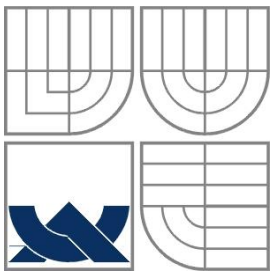
NÁSTROJ PRO ZPRACOVÁNÍ KONTAKTŮ NA PLATFORMĚ ANDROID

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

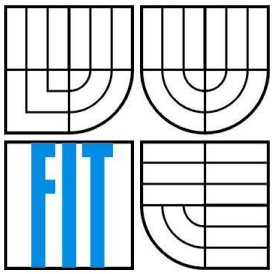
AUTOR PRÁCE
AUTHOR

Viliam Kasala

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

NÁSTROJ PRO ZPRACOVÁNÍ KONTAKTŮ NA PLATFORMĚ ANDROID

CONTACT LIST MANAGEMENT FOR ANDROID

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

Viliam Kasala

VEDOUCÍ PRÁCE
SUPERVISOR

doc. Dr. Ing. Dušan Kolář

BRNO 2012

Abstrakt

Tato bakalářská práce se zabývá návrhem a implementací nástroje, který je schopen na / z paměťové karty importovat, exportovat a synchronizovat údaje o kontaktech uložených v nativním programu systému Android. Nástroj podporuje pro export a import údaju Outlook a Thunderbid CSV formát. Výsledkem je aplikace, která je kompatibilní od Android verze 2.1 a výšší.

Abstract

This bachelor thesis describes the design and implementation of tool that is able to from / to the memory card import, export and synchronize contacts data stored in the native Android program. The tool supports for exporting and importing data Outlook and Thunderbird CSV format. The result of this thesis is application which is compatible from Android version 2.1 and higher.

Klíčová slova

Nástroj pro zpracování kontaktů, Google, Android, Java, Synchronizace, Import, Export, Kontakty, CSV

Keywords

Contact List Management, Google, Android, Java, Synchronization, Import, Export, Contacts, CSV

Citace

Kasala Viliam: Nástroj pro zpracování kontaktů na platformě Android, bakalářská práce, Brno, FIT VUT v Brně, 2012

Nástroj pro zpracování kontaktů na platformě Android

Prohlášení

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

……
Viliam Kasala
10. května 2012

Poděkování

Rád bych poděkoval doc. Dr. Ing Dušanovi Kolářovi za zajímavý námět a bezproblémovou spolupráci při tvorbě této práce.

Obsah

1 Úvod

Človek počas svojho života získava rôzne kontakty, či už telefónne čísla alebo emaily. Postupom času si ich nazhromaždí toľko, že vzniká obava straty. V tomto prípade siaha po zálohe. Inokedy sa stáva, že človek má údaje o telefónnych číslach v jednom zariadení a ďalšie údaje v inom zariadení. V takomto prípade použije synchronizačný nástroj. Avšak čo ak potrebujeme exportovať alebo synchronizovať údaje bez toho, aby o tom vedeli tretie strany?

Cieľom tejto práce je vytvoriť nástroj, ktorý dokáže importovať a synchronizovať údaje o kontaktoch z Thunderbirdového alebo MS Outlookového CSV súboru uloženého na pamäťovej karte s údajmi o kontaktoch uložených v natívnom systéme Android. Tento nástroj takisto dokáže exportovať na pamäťovú kartu údaje o kontaktoch z natívneho Android systému do Thunderbird alebo Outlook CSV formátu. Ďalej podporuje rôzne možnosti nastavení, pomocou ktorých je možné ovplyvňovať export, import alebo dokonca synchronizáciu. Táto problematika je zaujímavá hlavne preto, že podobných nástrojov neexistuje veľa. Väčšinou sa jedná iba o čiastkové aplikácie zastrešujúce iba export údajov. Čo sa týka importu a synchronizácie kontaktov, tak Google poskytuje jednoduchý spôsob ako tieto údaje synchronizovať. Avšak nevýhodou zostáva, že prístup k týmto citlivým údajom získava aj táto spoločnosť.

Kapitola *Platforma Android* obsahuje základné informácie o systéme Android, popisuje jeho architektúru, možnosti vývoja a základné prvky. Ďalej detailne popisuje štruktúru uloženia kontaktov v Androide.

Ďalšia kapitola *Štandardné formáty pre import/export údajov v MS Outlook a Thunderbird* sa zaoberá štandardnými formátmi pre export a import kontaktných údajov z programov MS Outlook a Mozilla Thunderbird.

V kapitole *Špecifikácia aplikácie* sú popísané vlastnosti a vymoženosti výslednej aplikácie a v kapitole *Návrh aplikácie* je diskutovaný návrh aplikácie a zdôvodnené postupy.

V kapitole *Implementácia* sú uvedené zaujímavosti týkajúce sa implementácie aplikácie ako diagram tried jednotlivých častí aplikácie a popis hlavných tried.

V kapitole *Testovanie* je popísaný postup pri testovaní výslednej aplikácie a verzie Android systémov, na ktorých bola aplikácia testovaná.

V poslednej kapitole sú diskutované možnosti rozšírenia a prínos práce.

2 Platforma Android

V tejto kapitole sa zameriame na popis základných vlastností a vymožeností platformy Android. Taktiež budú predstavené nevyhnutné základy pre programovanie Android aplikácii a API pre prácu s kontaktmi.

2.1 Čo je to Android

Android je open-source softwarová platforma, ktorá je optimalizovaná pre mobilné zariadenia ako sú chytré telefóny, PDA, tablety, netbooky, čítačky elektronických kníh a iné. Obsahuje operačný systém založený na Linuxe, middleware, súbor základných mobilných aplikácií a API knižnice určené na vývoj aplikácii pre túto platformu. Od iných mobilných operačných systémov sa líši hlavne v tom, že jeho kód vydala spoločnosť Google ako open-source pod Apache licenciou. Hoci sú aplikácie písané v jazyku Java, ich kód je kompilovaný a spúšťaný virtuálnym strojom Dalvik(ďalej len DVM), ktorý je upravenou verziou Java virtuálneho stroja určeného pre mobilné zariadenia. [1], [2]

2.1.1 História

Android bol pôvodne vyvíjaný firmou Android Inc., ktorá bola založená v roku 2003. Neskôr v roku 2005 bola firma odkúpená spoločnosťou Google, ktorá chcela vstúpiť na mobilný komunikačný trh. 5. novembra 2007 vzniklo konzorcium OHA vedené spoločnosťou Google, ktoré združovalo 34 významných IT spoločnosti ako HTC, Intel, Motorola, Texas Instruments, Nvidia, Motorola, LG, Qualcomm, atď. Cieľom OHA je vyvíjať a presadzovať otvorené štandardy pre mobilné zariadenia. V rovnaký deň bol predstavený aj prvý produkt tohto konzorcia - Android. [3]

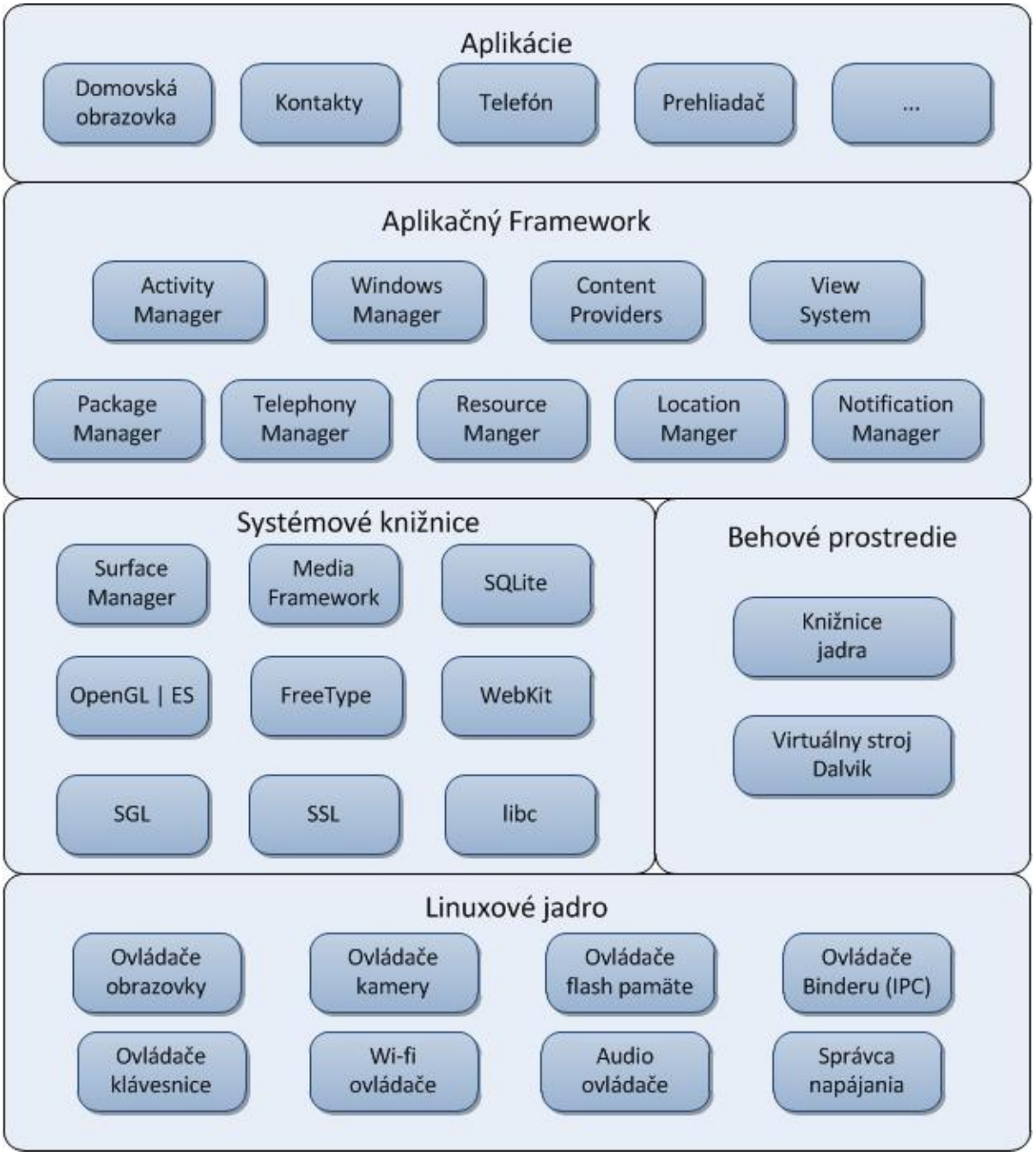
2.1.2 Verzie

Od vzniku prvého systému prešiel Android už mnohými vývojovými verziami. Každá nová verzia opravovala nedostatky predchádzajúcej verzie a rozširovala funkcie a možnosti systému. Novšie verzie sú kompatibilné so staršími z dôvodu fungovania vyvinutých aplikácií pre staršie verzie na novších verziách systému. Každá nová verzia je pomenovaná podľa názvov dezertov.

Spočiatku boli verzie určené iba pre zariadenia s menšími obrazovkami (verzie od 1.x). Neskôr príchodom tabletov vznikla potreba podporovať zariadenia aj s omnoho väčším displejom ako mali mobilné telefóny, čo malo za následok rozdelenie verzií. Generácia 2.x bola určená pre mobilné zariadenia s menšou obrazovkou a generácia 3.x pre tablety s väčšou obrazovkou. Zväčšovaním obrazoviek aj na mobilných zariadeniach došlo neskôr znova k zlúčeniu verzií od generácie 4.x. Najnovšia aktuálna verzia je *Android 4.0.4 Ice Cream Sandwich*. [3]

2.2 Architektúra platformy

Celá architektúra platformy sa skladá z niekoľkých vrstiev a prvkov, ktoré sú zobrazené na obrázku 2.1. V anglickej literatúre je často označovaná ako *Android software stack*. Zjednodušene povedané, linuxové jadro a súbor C/C++ knižníc sú vystavené programátorom prostredníctvom aplikačného frameworku, ktorý poskytuje služby pre správu a beh aplikácií. [4]



Obrázok 2.1: Diagram architektúry platformy Android [1]

2.2.1 Linuxové jadro

Najnižšia vrstva modelu, založená na jadre Linuxu verzie 2.6, poskytuje služby jadra systému – hardwarové ovládače, správu procesov a pamäti, zabezpečenie systému, sieťové služby a správu napájania. Jadro tiež pôsobí ako abstraktná vrstva medzi hardwarom a zvyškom softwarového zásobníka. [1], [4]

2.2.2 Systémové knižnice

Vrstva architektúry ležiaca nad linuxovým jadrom obsahujúca sadu C/C++ knižníc, ktoré sú používané rôznymi komponentmi systému Android. Na jednotlivé zariadenia sú predinštalované

výrobcom a kompilované pre konkrétne hardwarové zariadenie. Tieto knižnice sú dostupné vývojárom prostredníctvom Android aplikačného frameworku.

Niektoré z hlavných knižníc jadra systému : [1], [4]

- **Systémová knižnica jazyka C** – knižnica odvodená z BSD implementácie štandardnej systémovej knižnice jazyka C a zároveň upravená pre použitie vo vstavaných systémoch.
- **Multimediálne knižnice** – knižnice pre podporu prehrávania a nahrávania mnohých populárnych audio, video formátov ako aj statických obrazových súborov.
- **Surface Manager** – riadi prístup k podsystému displeja zariadenia a skladá 2D a 3D grafické vrstvy z viacerých aplikácií.
- **LibWebCore** – jadro internetového prehliadača.
- **SGL** – základný 2D grafický engine.
- **OpenGL ES** – 3D grafické knižnice využívané buď pre hardwarovú 3D akceleráciu alebo pre vysoko optimalizované 3D softwarové rastrovanie.
- **SSL** – knižnica pre šifrovanie sieťovej komunikácie.
- **FreeType** – slúži pre bitmapové a vektorové vykresľovanie písma.
- **SQLite** – výkonná a odľahčená relačná databáza, ktorá je k dispozícii pre všetky aplikácie.

2.2.3 Behové prostredie Androidu

Behové prostredie Androidu leží nad linuxovým jadrom a spolu so systémovými knižnicami tvorí základ pre aplikačný framework. Obsahuje základné knižnice programovacieho jazyka Java, ktoré poskytujú väčšinu funkcionality systémových knižníc vývojárom. Ďalej sú tu špecifické knižnice pre Android a DVM. [1]

Každá Android aplikácia je spúšťaná v jej vlastnom procese s jej vlastnou inštanciou DVM. DVM je registrovo založený virtuálny stroj, ktorý bol optimalizovaný tak, aby zariadenie mohlo spúšťať viac inštancií efektívne. Spúšťa triedy kompilované kompilátorom jazyka Java, ktoré boli transformované do *.dex* formátu. Pri nízko-úrovňovej správe pamäte a vláknach sa DVM spolieha na Linuxové jadro. [1]

2.2.4 Aplikačný framework

Nad DVM a systémovými knižnicami je aplikačný framework Androidu, ktorý obsahuje základné triedy a rozhrania pre stavbu aplikácií na vysokej úrovni, poskytuje platformovú nezávislosť a tiež poskytuje abstrakciu pre prístup k hardwarovým perifériám, spravuje užívateľské rozhranie a aplikačné zdroje. Pomocou tohto frameworku majú vývojári prístup k rovnakému API, ktoré používajú základné natívne aplikácie. Aplikačná architektúra je navrhnutá tak, aby zjednodušovala opakované použitie komponentov a aby akákoľvek aplikácia mohla zverejniť svoje funkcie. Celá architektúra dovoľuje iným aplikáciám využiť tieto funkcie. Rovnaký mechanizmus umožňuje, aby komponenty boli vymenené užívateľom. [1], [4]

Niektoré základné časti frameworku:

- **Notification Manager (správca upozornení)** – umožňuje všetkým aplikáciám zobrazovať vlastné upozornenia v stavovom riadku.
- **Content Providers (poskytovatelia obsahu)** – umožňujú aplikáciám pristupovať k dátam iných aplikácií alebo zdieľať svoje dáta (napr. kontakty).
- **Resource Manager (správca zdrojov)** – poskytujú prístup k zdrojom, ktoré nie sú súčasťou kódu aplikácie, ale sú umiestnené v špeciálnych XML súboroch ako napr. lokalizované zdroje, grafika a súbory pre tvorbu GUI.

- **Activity Manager (správca aktivít)** – riadi životný cyklus aplikácií a spravuje spoločný zásobník slúžiaci k navigácii medzi aplikáciami a ich aktivitami.

2.2.5 Aplikácie

Poslednou, najvyššou vrstvou sú samotné aplikácie napísané v programovacom jazyku Java s využitím aplikačného frameworku. Sú v ňom napísané aj všetky natívne aplikácie dodávané štandardne so systémom ako Emailový klient, Kontakty, Kalendár a pod. Aplikácie sú programy, ktoré môžu bežať buď na popredí alebo na pozadí systému a určitým spôsobom interagovať s užívateľom. Každá vyvinutá aplikácia je skompilovaná pomocou Android SDK (viď. 2.3.1) do Android balíčku (*Android package*), čo je v podstate ZIP archív s príponou *.apk*. Slúži k distribúcií aplikácií a k ich inštalácií na jednotlivé zariadenia. Celý kód, dátové a zdrojové súbory sú považované za jednu aplikáciu. [5]

Po inštalácií na zariadenie, každá aplikácia funguje vo svojej vlastnej oblasti zabezpečenia:

- V predvolenom nastavení beží každá aplikácia vo vlastnom linuxovom procese. Android vytvára proces, až keď akýkoľvek komponent aplikácie má byť spustený a ukončí proces v prípade, ak už nie je potrebný alebo ak systém musí obnoviť pamäť pre iné aplikácie.
- Každý proces má pridelený vlastný virtuálny stroj DVM, a preto je kód aplikácie vykonávaný v izolácií od iných aplikácií.
- Každej aplikácii systém pridelí jedinečné linuxové užívateľské ID (číslo sa používa iba v systéme a nie je známe aplikácii). Systém stanovuje povolenia pre všetky súbory aplikácie tak, že len samotná aplikácia a užívateľ s týmto ID má prístup k samotným dátam.

2.3 Možnosti vývoja

Existuje viac možností ako vyvíjať aplikácie pre Android platformu. Základom každej je mať nainštalovaný Android SDK, bez ktorého nie je možný vývoj. Ďalšie možnosti sú popísané v nasledujúcich podkapitolách. [6]

2.3.1 Android SDK

Android SDK je súbor vývojových nástrojov pre tvorbu aplikácií pre Android platformu. Obsahuje knižnice, emulátory, dokumentácie pre všetky aktuálne dostupné verzie aplikačného frameworku a nástroje pre kompiláciu zdrojových kódov.

Emulátor je desktopová aplikácia, ktorá umožňuje simuláciu rôznych konfigurácií Android zariadení, pričom je možné testovať aplikáciu bez nutnosti vlastniť reálne zariadenie. Okrem toho, emulátor umožňuje konfiguráciu rozlíšenia obrazovky, veľkosti operačnej pamäte i hardwarových zariadení ako kameru, mikrofón, GPS, atď. Tiež je možné simulovať akcie reálneho zariadenia, ako sú príchod MMS, SMS, volania. [6], [7]

2.3.2 Android NDK

Android NDK je sada nástrojov, ktorá dovoľuje vložiť komponenty využívajúce natívny kód jazykov ako C, C++ v Android aplikáciách, čo poskytuje výhody niektorým skupinám aplikácií v podobe opätovného využitia už existujúceho kódu, či zvýšenia rýchlosti aplikácie. Typický príklad, kde je možné využiť NDK, sú hry, grafika a operácie, ktoré intenzívne vyťažujú CPU. Pri použití NDK by

mal vývojár zvážiť jeho výhody a nevýhody, pretože použitie natívneho kódu nemá za následok automatické zvýšenie výkonu, no vždy sa zvyšuje zložitosť aplikácie. [8]

2.4 Základ pre tvorbu aplikácií

Súčasťou aplikácie sú základné stavebné komponenty aplikácie. Aplikácia sa musí skladať aspoň z jedného komponentu. Každý komponent je odlišným bodom, ktorým môže aplikácia vstúpiť do systému. Nie všetky komponenty sú skutočné vstupné body pre užívateľov. Niektoré sú dokonca na sebe vzájomne závislé. Avšak každý jeden existuje ako vlastná entita a má špecifickú rolu – predstavuje jedinečný stavebný prvok, ktorý pomáha aplikácii definovať celkové správanie. [5]

Existujú 4 základné typy aplikačných komponentov, pričom každý slúži k inému účelu a má vlastný životný cyklus, ktorý definuje ako komponent vzniká a zaniká. Jednotlivé komponenty sú definované v manifeste (viď. 2.4.6). Nasledujúce podkapitoly popisujú jednotlivé komponenty. [5]

2.4.1 Aktivity

Aktivita (*Activity*) reprezentuje obrazovku s GUI, s ktorou môže užívateľ komunikovať, aby niečo vykonal, napr. vytočil telefónne číslo. Každej aktivite je pridelené okno, do ktorého si vykreslí aktivita svoje GUI. Okno väčšinou vyplní celú obrazovku, ale môže byť aj menšie ako celá obrazovka a vtedy je zobrazený nad iným oknom. Aplikácia sa väčšinou skladá z niekoľkých aktivít, ktoré sú navzájom previazané. Typicky býva jedna aktivita v aplikácii označená ako vstupný bod a táto aktivita (jej GUI) je potom zobrazená ako prvá pri spustení aplikácie. [5], [9]

Každá aktivita v systéme sa ukladá na zásobník aktivít (*Activity Stack*) a beží v samostatnom procese DVM. Aktuálna aktivita, ktorá je na popredí, je na vrchole tohto zásobníka. Ak sa spustí nová aktivita, predchádzajúca aktivita je pozastavená a systém ju uchová v zásobníku. Zásobník je typu LIFO, čo v podstate znamená, že ak užívateľ stlačí hardwarové tlačidlo *Späť*, systém vyberie súčasnú aktivitu zo zásobníka (zrušená) a predchádzajúca aktivita pokračuje v činnosti. Systém si úplne sám riadi beh procesov, preto nie je v Androide garantovaný beh žiadneho procesu. Inými slovami, najskôr sú zrušené procesy, ktoré majú aktivity na pozadí (nie sú na vrchole zásobníku). [9]

Aktivity sú dedené z triedy ***android.app.Activity***. Majú vlastný životný cyklus (viď. obrázok 2.2) a prechádzajú niekoľkými stavmi. Tieto stavy sú reprezentované jednotlivými metódami, ktoré sú podľa určitej udalosti volané: [9]

- **onCreate(Bundle)** – je volaná raz, keď je aktivita po prvýkrát vytvorená. Je to vhodné miesto pre statické nastavenia ako vytvoriť GUI, naviazať dáta do zoznamov, zaregistrovať obsluhy udalostí. Metóde je predaný ***Bundle*** objekt obsahujúci stav predchádzajúcej aktivity, ak jej stav bol uchovaný.
- **onRestart()** – je volaná potom, ako bola činnosť aktivity pozastavená a aktivita sa má znovu zobraziť užívateľovi.
- **onStart()** – je volaná vždy pred zobrazením aktivity užívateľovi.
- **onResume()** – je volaná vtedy, keď je aktivita pripravená na interakciu s užívateľom. Na tomto mieste je aktivita na vrchole zásobníka.
- **onPause()** – je zavolaná pred tým, ako je aktivita presunutá do pozadia a systém sa chystá pokračovať v inej aktivite. Je to vhodné miesto na uloženie rôznych perzistentných dát, zastavenie rôznych aplikácii a iných vecí, ktoré môžu vyťažovať CPU. Táto metóda by mala byť veľmi rýchla, pretože v nasledujúcej aktivite sa nebude pokračovať, pokiaľ neskončí táto metóda. Pre uloženie špeciálnych dát sa využíva objekt ***Bundle*** a metóda ***onSaveInstanceState(Bundle)***, ktorá prijíma ako parameter tento objekt.

- **onStop()** – je volaná vtedy, keď aktivita už nie je viditeľná pre užívateľa. K tomu môže dôjsť, pretože je práve ničená alebo sa obnovila iná aktivita a prekrýva ju.
- **onDestroy()** – volá sa v prípade, ak je aktivita zrušená. Toto je posledná možnosť vykonať určitú činnosť pred tým, ako bude aktivita zničená.

Aktivity nie sú určené k dlhodobým operáciám a výpočtom, pretože blokujú GUI. Ak chceme vytvoriť aplikáciu, ktorá bude na pozadí vykonávať určitú činnosť, je potrebné zvážiť použitie vlákien. [9]

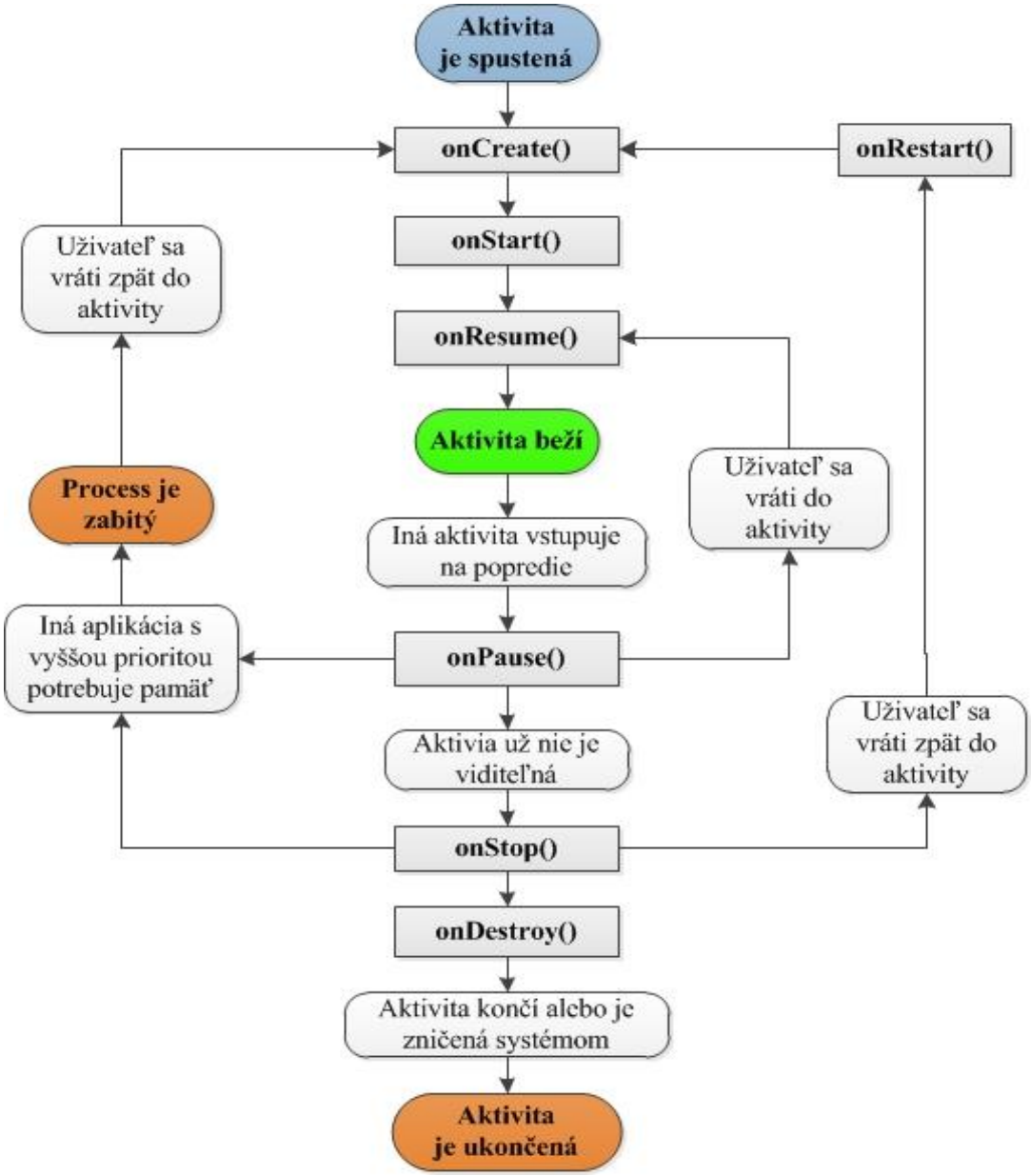
2.4.2 Služby

Služba (*Service*) je komponent určený k vykonávaniu dlhotrvajúcej operácie počas neurčitej doby bez priamej interakcie s užívateľom. Neobsahuje vlastné GUI, pomocou ktorého by bola ovládaná. Avšak môže byť ovládaná pomocou interakcie s aktivitou, ku ktorej je služba pripojená alebo je tiež schopná aktivovať systém notifikácií (viď. 2.4.3). Iný aplikačný komponent môže spustiť službu, ktorá bude bežať na pozadí aj v prípade, ak sa užívateľ prepne do inej aplikácie. Väčšinou sa využíva na prehrávanie hudby, sťahovanie dát z internetu alebo medziprocesovú komunikáciu. [10]

Služby sú dedené z triedy ***android.app.Service*.** K prebiehajúcim službám sa možno pripojiť pomocou operácie ***bind()*** a následne komunikovať pomocou rozhrania, ktoré poskytuje nadradená trieda. Aby nedochádzalo k blokovaniu iných komponentov alebo GUI aplikácie, tak sú služby vytvárane väčšinou vo vlastných vláknach. [10]

2.4.3 Prijímače broadcastu

Prijímače broadcastu (*Broadcast Recievers*) sú komponenty, ktoré reagujú na určité oznámenia. Oznámenia môžu prichádzať zo systému alebo aj z rôznych aplikácií, ako napr. batéria vybitá, obrazovka sa zapla, nový email, obrázok bol vyfotený. Na tieto oznámenia reagujú ďalšie aplikácie rôznym spôsobom. Aj napriek tomu, že prijímače nemajú vlastné GUI, môžu vložiť upozornenie do stavového riadku či vyvolať aktivitu nejakej aplikácie. Všetky prijímače rozširujú abstraktnú triedu ***android.content.BroadcastReciever*** a musia implementovať jednu abstraktnú metódu ***onRecieve()***. [5], [11]



Obrázok 2.2: Životný cyklus aktivity [9]

2.4.4 Poskytovatelia obsahu

Poskytovatelia obsahu (*Content Providers*) sú komponenty poskytujúce štandardné rozhranie, pomocou ktorého je možné pristúpiť k dátam iných aplikácií alebo dokonca zdieľať vlastné dáta iným aplikáciám. Príkladom je čítanie informácií o kontaktoch zo základnej aplikácie *Kontakty*. Poskytovatelia obsahu riadia prístup k usporiadanej množine dát aplikácie, zapuzdrujú tieto dáta a poskytujú mechanizmus pre ich bezpečnosť. Umožňujú oddeliť aplikačnú vrstvu od dátovej. [4], [5]

Poskytovatelia obsahu môžu byť dotazovaní na určité dáta, môžu pridávať, modifikovať a mazať dáta, len v prípade ak má aplikácia príslušné oprávnenia. Takýmto spôsobom je možné upravovať aj natívne Android databázy. Tieto dáta sú prezentované vonkajším aplikáciám ako jedna alebo viac tabuliek, ktoré sú veľmi podobné relačným databázam. Prístup k ním je možný pomocou URI modelu so schémou **content://**, ktorá býva väčšinou v tvare: [12]

content://<meno_poskytovateľa_obsahu>/<cesta_ku_zdroju>

Prístup k týmto dátam je možný pomocou objektu ***android.content.ContentResolver***, ktorý získame metódou ***getContentResolver()***. Nad ním je už možné volať metódy ako ***query()***, ***insert()***, ***update()***, ***delete()***. Dáta čítame cez rozhranie typu ***android.content.Cursor***, ktorý nám vracia metóda ***query()*** a ktorý má definované metódy pre zistenie počtu riadkov a stĺpcov – ***getCount()***, ***getColumnCount()***, ďalej tiež metódy na zistenie názvu stĺpca – ***getColumnNames()*** a taktiež samotné metódy na získanie dát z určitého stĺpca podľa typu dát – ***getString()***, ***getInt()***, ***getLong()***. V tabuľke číslo 2.1 je možné vidieť porovnanie parametrov metódy ***query()*** s SQL dotazom. [12]

Query() parameter	**SELECT kľúčové slovo/parameter**	**Poznámka**
Uri	**FROM názov_tabuľky**	*Uri* mapuje do tabuľky poskytovateľa podľa názov_tabuľky.
projection	**col1, col2, col3, ...**	*projection* je pole stĺpcov, ktoré by mali byť zahrnuté v každom riadku, ktorý má byť vytiahnutý z tabuľky.
selection	**WHERE col = hodnota**	*selection* upresňuje kritéria pre výber riadkov.
selectionArgs	(Žiadny presný ekvivalent. Selection argumenty nahradia zástupné symboly **?** v selection klauzule.)	*selectionArgs* je pole reťazcov, ktoré majú byť vložené namiesto **?** do selection klauzule.
sortOrder	**ORDER BY col1, col2, ...**	*sortOrder* určuje poradie, v ktorom sa objavia stĺpce vráteného *Cursor-u*.

Tabuľka 2.1 Metóda ***query()*** porovnávaná s SQL dotazom [12]

V prípade, ak chceme poskytnúť vlastné dáta iným aplikáciám, je nevyhnutné vytvoriť vlastného poskytovateľa obsahu, ktorý rozširuje abstraktnú triedu ***android.content.ContentProvider()*** a implementuje všetky jeho abstraktné metódy. [13]

2.4.5 Zámer

Jednotlivé komponenty sú spúšťané pomocou zámerov (*Intents*). Jedná sa o asynchronnú správu, ktorá predstavuje pasívnu dátovú štruktúru obsahujúcu abstraktný popis operácie, ktorá sa má vykonať. Zámer sa používa na spúšťanie nasledovných komponentov – aktivity, služby a prijímače broadcastu. Každý intent môže niesť rôzne atribúty. Uvedieme aspoň pár z nich: [5]

- **akcia (action)** – určuje, akú akciu má zámer vyvolať. Jednotlivé akcie sú definované konštantami v triede *Intent*.
- **typ (MIME type)** – určuje typ dát, napr. text/plain, image/jpeg, text/csv, atď.
- **URI schéma** – špecifikuje, kde sa dáta nachádzajú.

Zámery sa delia do 2 kategórii:[5]

- **Explicitné zámery** – využívajú sa na vyvolanie komponentov v rámci aplikácie, kde presne špecifikujeme, ktorý komponent má byť spustený.
- **Implicitné zámery** – nešpecifikujeme, ako sa má činnosť vykonať, ale necháme systém rozhodnúť, ktorý komponent inej aplikácie zámer obslúži.

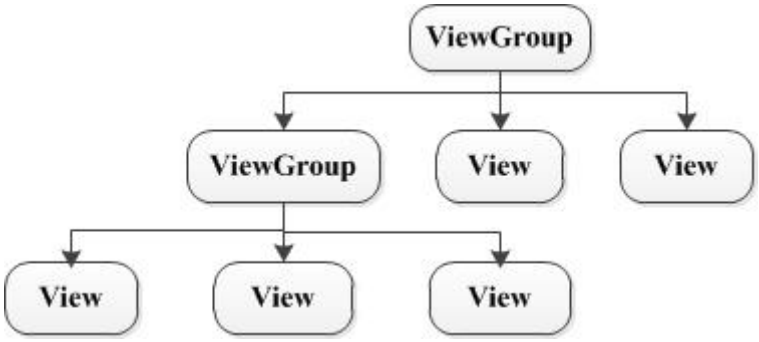
2.4.6 Manifest súbor

Každá aplikácia musí obsahovať súbor s formátom XML a s názvom *AndroidManifest.xml* (ďalej len manifest). Je to základný súbor, v ktorom sú uvedené nevyhnutné informácie pre beh aplikácie. V tejto časti musia byť deklarované všetky komponenty, ktoré aplikácia využíva. Niektoré dôležité informácie obsiahnuté v manifeste: [5]

- Špecifikuje všetky užívateľské oprávnenia, ktoré potrebuje aplikácia k svojmu behu, ako napr. prístup k internetu, prístup k čítaniu alebo zapisovaniu užívateľských kontaktov, atď.
- Určuje minimálnu verziu Android API, ktorú aplikácia vyžaduje pre svoj beh.
- Definuje hardwarové alebo softwarové funkcie nevyhnutné pre beh aplikácie.
- Popisuje jednotlivé komponenty aplikácie.

2.4.7 Tvorba GUI

GUI samotných Android aplikácií je zložené z dvoch základných prvkov ***View*** (reprezentuje prvky ako tlačidlá, zaškrtávacie tlačidlá) a ***ViewGroup*** (reprezentuje správcov rozmiestnenia). Všetky prvky sú potomkami triedy ***View***. Pri vytváraní GUI sa vytvára hierarchický strom objektov (viď. Obrázok 2.3) a jednotlivé objekty je možné do seba ďalej zanorovať. [14]



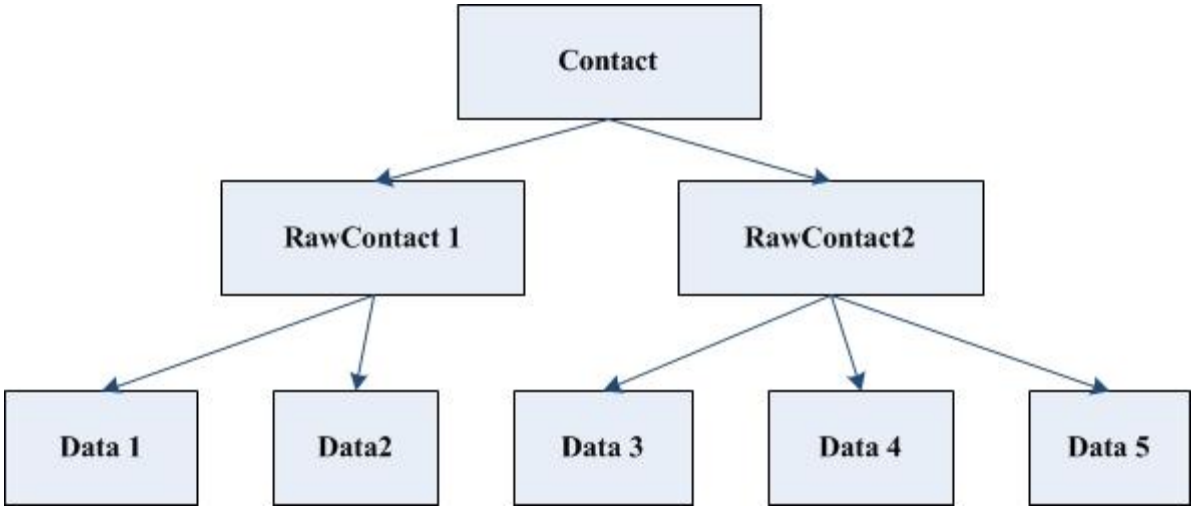
Obrázok 2.3 Hierachický strom objektov GUI [14]

2.5 Kontakty

Od verzie 2.0 Android poskytuje vylepšené API pre správu kontaktov z viacerých účtov a z iných dátových zdrojov. Kontakty sú prístupné aplikáciám cez poskytovateľa obsahu kontaktov (***Contacts Contract Content Provider***) a cez vhodné užívateľské oprávnenia ***READ_CONTACTS*** a ***WRITE_CONTACTS***. V prípade, ak by sa dáta z viacerých zdrojov prekrývali, poskytovateľ obsahu kontaktov agreguje podobné kontakty a prezentuje ich užívateľovi ako jeden celok. [4], [15]

2.5.1 Datová štruktúra kontaktov

Nové API poskytuje 3-vrstvový dátový model (viď. Obrázok 2.4) na ukladanie informácií o kontakte. Skladá sa primárne z 3 tabuliek: ***Contacts***, ***RawContacts*** a ***Data***.



Obrázok 2.4 3-vrstvový datový model kontaktov [15]

2.5.2 Tabuľka Data

Tabuľka ***Data*** je generická tabuľka, ktorá ukladá všetky údaje spojené s jedným ***RawContact***-om. Každý riadok definuje súbor osobných údajov určitého druhu (napr. telefónne čísla, e-mailové adresy, atď.). Každý riadok je označený MIME typom, ktorý identifikuje, aký typ údajov môže riadok obsahovať naprieč celým stĺpcom. Stĺpce sú tiež generické a typ informácií, ktoré obsahujú, určuje druh údajov uložených v každom riadku. Napríklad, ak je riadok druhu ***Phone.CONTENT_ITEM-_TYPE***, tak prvý stĺpec ukladá telefónne číslo, ale ak je druhu ***Email.CONTENT_ITEM_TYPE***, tak stĺpec ukladá emailovú adresu. [4], [15]

Všetky možné MIME typy riadkov sú definované ako konštanty v triede ***android.provider.ContactsContract.CommonDataKinds*** a sú nimi údaje ako email, poznámky, štruktúrovaná adresa, štruktúrované meno, prezývka, organizácia, telefónne číslo, webová stránka, atď. Tiež je možné definovať vlastné druhy a pridávať rôzne nové údaje o kontakte.

Ak chceme pridávať nové údaje do tabuľky ***Data***, tak musíme špecifikovať ***RawContact***, ku ktorému patria pridávané údaje. [4], [15]

2.5.3 Tabuľka RawContacts

Od Android verzie 2.0 môžu užívatelia využívať viac účtov na uloženie kontaktov v telefóne, ako napr. Gmail, Facebook, Exchange, Skype alebo si dokonca vytvoriť vlastné účty. Každý riadok v tabuľke ***RawContacts*** definuje účet, ku ktorému je množina osobných údajov z tabuľky ***Data*** pripojená. Niektoré účty obsahujú ochranu proti pridávaniu, modifikovaniu alebo mazaniu údajov.[4], [15]

2.5.4 Tabuľka Contacts

Jeden riadok v tabuľke ***Contacts*** reprezentuje agregáciu jedného alebo viacerých informácií z tabuľky ***RawContacts*** a popisuje rovnakú osobu. Riadky a jednotlivé položky tejto tabuľky sú chránené proti modifikácií, pretože o tieto údaje a aj samotnú agregáciu sa stará systém Android sám. [4], [15]

2.5.5 Agregácia

Ak je pridávaný nový alebo modifikovaný, už existujúci ***RawContact***, tak systém vyhľadáva zhodujúce sa ***RawContact***-y, s ktorými sa agreguje. Ak nenájde žiadnu zhodu, tak vytvorí agregovaný kontakt (v tabuľke ***Contacts***), ktorý obsahuje len originálny ***RawContact***. V prípade, ak nájde práve jeden podobný, vytvorí nový kontakt (v tabuľke ***Contacts***) obsahujúci tieto dva ***RawContact***-y. A ak nájde viac zhodných ***RawContact***-ov, tak rozhodne najbližšia zhoda. [14]

Systém považuje dva ***RawContact***-y za zhodné v prípade, ak:

1. sa zhodujú v mene.
2. majú vymenené poradie mien (napr. „Bob Parr“ a „Parr Bob“).
3. časť mena je skratkou druhého mena (napr. „Bob Parr“ a „Robert Parr“).
4. jeden obsahuje len krstné meno alebo priezvisko a to sa zhoduje s menom druhého kontaktu. Toto pravidlo je menej spoľahlivé, preto je aplikované len vtedy, ak kontakty majú zhodné aj iné údaje ako telefónne číslo, emailovú adresu prípadne prezývku (napr. Helen[prezývka] a Helen Parr[prezývka]).
5. aspoň jednému chýba celé meno a obaja majú rovnaké telefónne číslo, emailovú adresu alebo prezývku (napr. Bob Parr[incredible@android.com] a incredible@android.com).

Keď dochádza k porovnávaniu mien, tak systém ignoruje veľké a malé písmena i diakritické znaky. Keď dochádza k porovnávaniu dvoch telefónnych čísel, tak systém ignoruje špeciálne znaky ako *, #, (,) a biele znaky. [14]

2.5.6 Pohľad RawContactEntity

Pohľad ***RawContactEntity*** je vonkajším spojením (outer join) tabuliek ***RawContacts*** a ***Data***. Je striktne obmedzený iba na čítanie. Dotaz na získanie všetkých údajov z tabuľky ***Data*** o ***RawContact***-e by mal byť volaný nad týmto pohľadom. V prípade, ak by sme použili dva separované dotazy, najprv na ***RawContacts*** a následne na ich dáta, mohlo by sa stať, že medzi týmito dotazmi dôjde k zmene údajov. Výhodou použitia i dotazu nad týmto pohľadom je načítanie všetkých údajov v jednej transakcii. [15]

3 Štandardné formáty pre import/export údajov v MS Outlook a Thunderbird

V tejto kapitole si priblížime jednotlivé podporované štandardné formáty pre export/import údajov v prostredí MS Outlook a Thunderbird a detailnejšie popíšeme dôležitejšie formáty.

V dnešnej dobe by sme márne hľadali lepšie nástroje pre správu pošty, ako sú MS Outlook alebo Mozilla Thunderbird. Sú to špičkové nástroje, ktoré poskytujú vymoženosti nielen správcu pracovných a osobných emailov, ale aj adresár kontaktov, kalendár a mnoho ďalších prídavkov.

Tieto programy umožňujú svojim užívateľom pracovať zároveň s viacerými vzdialenými emailovými účtami, mať ich sústredené v jednej aplikácii a mnohokrát im poskytujú prepracovanejšie a zároveň jednoduchšie grafické rozhranie. Okrem toho ponúkajú široké možnosti rôznych nastavení. Jednou z možností je aj export/import osobných údajov (informácií o osobách, firmách) či dokonca vlastných emailov v preddefinovaných štandardných formátoch. Stručný prehľad podporovaných formátov pre import/export kontaktov z adresára je uvedený v tabuľke 3.1.

Formát	Export/Import	Outlook /Thunderbird	Popis
vCard (.vcf)	Nie/Áno	Áno/Áno	Súborový formát pre výmenu osobných dát, hlavne elektronických obchodných vizitiek.
CSV, (.csv)	Áno/Áno	**Áno/Áno**	Jednoduchý súborový formát určený pre výmenu tabuľkových hodnôt používajúci ako oddeľovač čiarku CSV.
TAB (.tab)	Áno/Áno	**Áno/Áno**	Jednoduchý súborový formát určený pre výmenu tabuľkových hodnôt používajúci ako oddeľovač znak tabulátoru.
Outlook Data File (.pst)	Áno/Áno	Áno/Nie	Otvorený formát súborov používaný na ukladanie kópii správ, udalosti kalendára, kontaktov a iných položiek v rámci programov spoločnosti Microsoft.
LDIF (.ldif)	Áno/Áno	Nie/Áno	Štandardizovaný formát pre reprezentáciu a aktualizáciu dát na LDAP servery.

Tabuľka 3.1 Prehľad formátov pre import/export kontaktov v MS Outlook a Thunderbird[17, 18, 19]

3.1 LDIF

Jedná sa o jednoduchú textovú reprezentáciu záznamov v adresári. Každý záznam je reprezentovaný ako skupina atribútov a je oddelený od ďalších záznamov prázdnymi riadkami. Jednotlivé atribúty sú

potom v zázname obsiahnuté ako *názov_atribútu: hodnota*. Hodnoty atribútov, ktoré nepatria do prenosnej podmnožiny ASCII znakov, sú označené pomocou *::* po názve atribútu. Následne sú prekódované do ASCII pomocou *base64* kódovania. [19]

3.2 CSV

Súbor vo formáte CSV pozostáva z riadkov, v ktorom sú jednotlivé položky oddelené znakom čiarka. Hodnoty položiek môžu byť uzavreté do úvodzoviek, čo umožňuje, aby text položky mohol obsahovať čiarku a biele znaky. Ak text obsahuje úvodzovky, tak úvodzovky sú potom zdvojené. Prvý riadok väčšinou obsahuje názvy jednotlivých položiek. [17]

Oba nástroje pri importe a exporte kontaktov z vlastného adresára využívajú na kódovanie a dekódovanie znakov defaultné kódovanie používané operačným systémom MS Windows.

4 Špecifikácia aplikácie

V tejto kapitole si predstavíme aplikáciu, ktorá je predmetom tejto bakalárskej práce. Ďalej si popíšeme jednotlivé časti, z ktorých sa aplikácia skladá. Následne budú predstavené funkcie a vlastnosti týchto častí a diagramy prípadov užitia.

Hlavným účelom aplikácie je export, import a synchronizácia kontaktov na pamäťovú kartu/z pamäťovej karty telefónu z/do systému Androidu. Preto sa aplikácia skladá z dvoch základných častí:

- **Exportér kontaktov** – zabezpečuje export kontaktov z telefónu.
- **Importér kontaktov** – zabezpečuje import kontaktov do telefónu a ich synchronizáciu.

4.1 Exportér kontaktov

Exportér aplikácie umožňuje:

- exportovať údaje o kontaktoch zo systému Android do súboru a ukladať súbor na pamäťovú kartu.
- exportovať iba kontakty z vybraných účtov Android telefónu.
- exportovať iba určité údaje z kontaktu ako telefónne číslo, emailové adresy, adresu, atď.
- exportovať súbor v Thunderbird i Outlook CSV formáte.
- ukladať súbor do rôznych kódovaní.
- ukladať údaje do súboru bez diakritických znamienok.

4.1.1 Diagram prípadov použitia exportéra

Diagram prípadov užitia exportéra je zobrazený na obrázku 4.1. Hlavným aktérom je užívateľ, ktorý môže prostredníctvom rôznych nastavení ovplyvňovať spôsob exportu kontaktov (vybrať exportované účty, exportované údaje, atď.) a následne spustiť export samotných kontaktov.

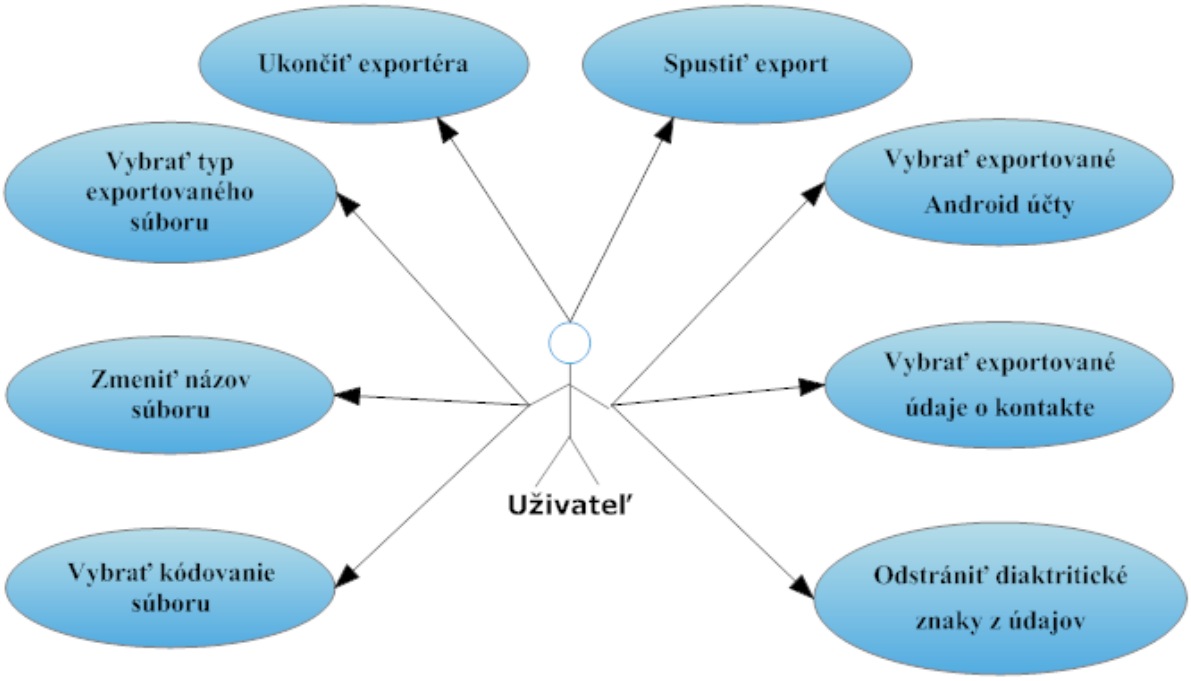
4.2 Importér kontaktov

Importér aplikácie umožňuje:

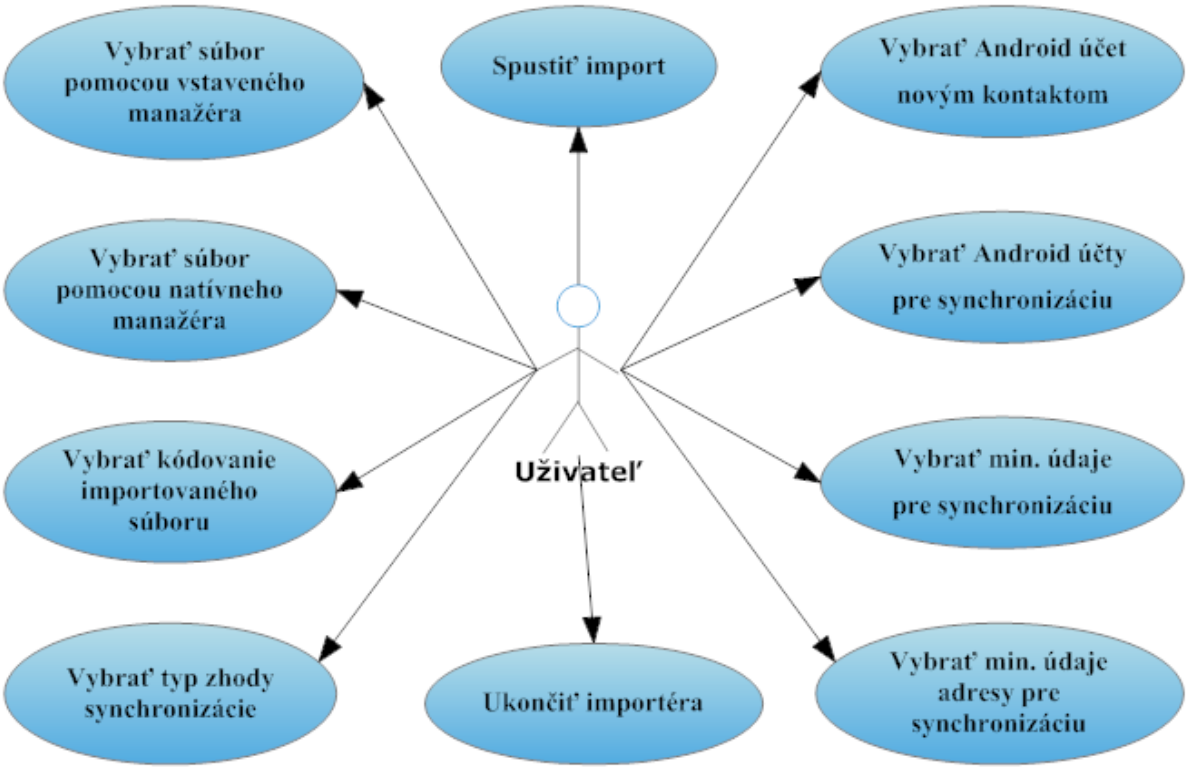
- importovať údaje o kontaktoch zo súboru do systému Android, pričom v prípade zhody kontaktov dochádza k synchronizácii.
- vybrať súbor pomocou natívneho správcu súborov alebo vstavaného manažéra.
- vybrať kódovanie importovaného súboru.
- vybrať Android účet, kam budú nové/novovytvorené kontakty patriť.
- synchronizovať nové kontakty len s určitými Android kontaktmi patriacimi pod zvolené účty.
- synchronizovať na základe zvolených zhodných údajov dvoch kontaktov, ako napr. meno, email, telefón.
- synchronizovať na základe zvoleného typu zhody dvoch kontaktov, a to na presnú zhodu, zhodu s ignorovaním veľkých/malých písmen, presnú zhodu s ignorovaním diakritických znamienok, zhodu s ignorovaním veľkých/malých písmen i diakritických znamienok.
- ovplyvniť synchronizáciu adresy.

4.2.1 Diagram prípadov použitia importéra

Diagram prípadov užitia importéra je zobrazený na obrázku 4.2. Hlavným aktérom je užívateľ, ktorý môže rôznymi nastaveniami ovplyvňovať spôsob importu a synchronizácie kontaktov a spustiť import kontaktov.



Obrázok 4.1 Diagram prípadov použitia exportéra



Obrázok 4.2 Diagram prípadov použitia importéra

5 Návrh aplikácie

Táto kapitola približuje jednotlivé častí aplikácie, charakterizuje ich vlastnosti a vysvetľuje, prečo sme sa rozhodli práve pre využitie vybraných vlastností a funkcií jednotlivých častí.

Zo všetkých možných formátov, ktoré podporujú Thunderbird a MS Outlook (viď. 2), bol zvolený CSV formát. Základnými prednosťami tohto formátu je práve podpora obomi nástrojmi, jednoduchá práca a parsovanie. Kvôli podpore CSV formátu je potrebný parser, ktorý budú vyžívať obe časti aplikácie. Všetky požiadavky na parser sú uvedené v popise CSV formátu (viď. 3.2).

Oba nástroje kódujú a dekódujú súbor pre export alebo import v defaultnom nastavení operačného systému (viď. 3.2), a preto musia obe časti aplikácie podporovať vhodné kódovanie a dekódovanie.

Keďže každý kontakt musí v Androide patriť pod určitý účet (viď. 2.5.4) a nie všetky účty dovoľujú plnú manipuláciu s ich kontaktmi, tak si musí uživateľ sám ustrážiť, či pracuje s účtom, ktorý dovoluje manipuláciu s jeho kontaktmi.

5.1 Exportér

5.1.1 Mapovanie

Podrobné mapovanie jednotlivých údajov Android kontaktu do Thunderbird CSV súboru je zobrazené v *Prílohe A* a *Príloha B* zachytáva podrobné mapovanie jednotlivých údajov Android kontaktu do MS Outlook CSV súboru. Z príloh je možné vidieť, že CSV súbory majú obmedzený počet položiek a oba nástroje podporujú ukladanie údajov v rôznom počte pre jednotlivé kontakty. Z tohto dôvodu nie je možné uložiť všetky údaje z Android kontaktu do súboru, čo je spôsobené rôznym fungovaním samotných nástrojov, ktoré podporujú len svoje špecifické údaje, ktoré ukladajú ku kontaktu. Oba formáty neobsahujú položky na ukladanie údajov o IM.

5.1.2 Duplicitné kontakty

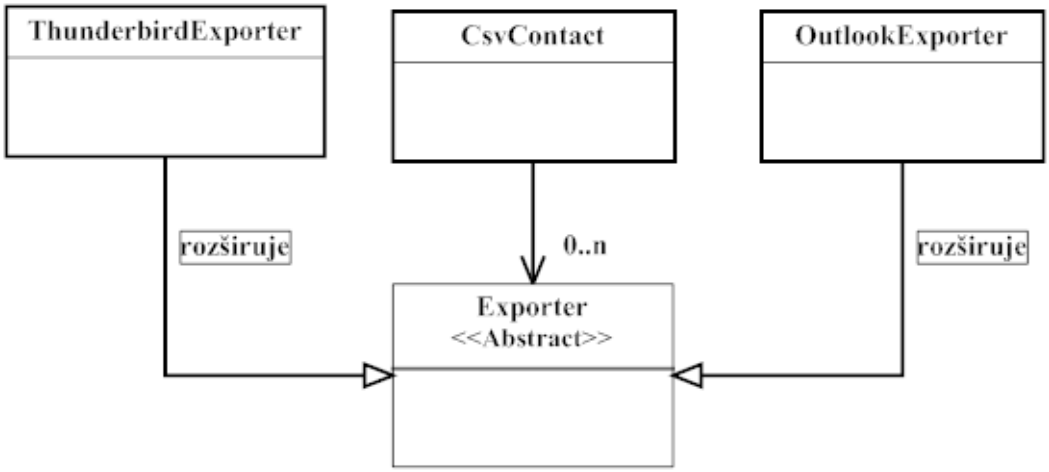
Nakoľko CSV formáty obsahujú obmedzený počet rovnakých typov údajov pre jeden kontakt a Android kontakt môže obsahovať relatívne neobmedzený počet rovnakých údajov, je nevyhnutné sa vysporiadať so situáciou, ak prenášaný kontakt obsahuje viac údajov ako je CSV kontakt schopný uložiť. Aplikácia opisovanú situáciu rieši dvoma spôsobmi.

Po prvé, aplikácia vytvorí duplicitný kontakt s rovnakým menom a do položky *Notes* uloží špeciálny príznak, ktorý identifikuje duplicitný kontakt. Súčasne uloží do položiek, ktoré by sa inak nevyužili k exportu údajov, emailové adresy kvôli podpore spätného importu do Androidu (viď. 5.2). U Thunderbird CSV formátu sú to *Custom 1* a *Custom 2*, kde sa uložia maximálne dve emailové adresy. U Outlook CSV formátu sú to *User 1*, *User 2* a *User 3*, kde sa uložia maximálne tri emailové adresy.

Po druhé, aplikácia naplní kontakt maximálne toľkými údajmi, koľko sa do CSV kontaktu zmestí. Z užívateľského hľadiska je dôležité, aby sa užívateľ sám rozhodol, či chce exportovaný súbor využiť pre zálohu kontaktov alebo pre import s jedným nástrojom.

5.1.3 Návrh základných tried exportéra

Exportér podporuje dva druhy CSV formátov, pričom každý z nich dokáže uložiť rôzny počet údajov a rovnaké údaje sú uložené na iných miestach – iné položky a indexovanie. Exportér využíva rovnaké funkcie pre vyhľadávanie v tabuľkách, ale iné, špecifickejšie funkcie pre ukladanie dát z Androidu do CSV formátu. Návrh základných tried je zobrazený na obrázku:



Obrázok 5.1 Obecný diagram tried exportéra

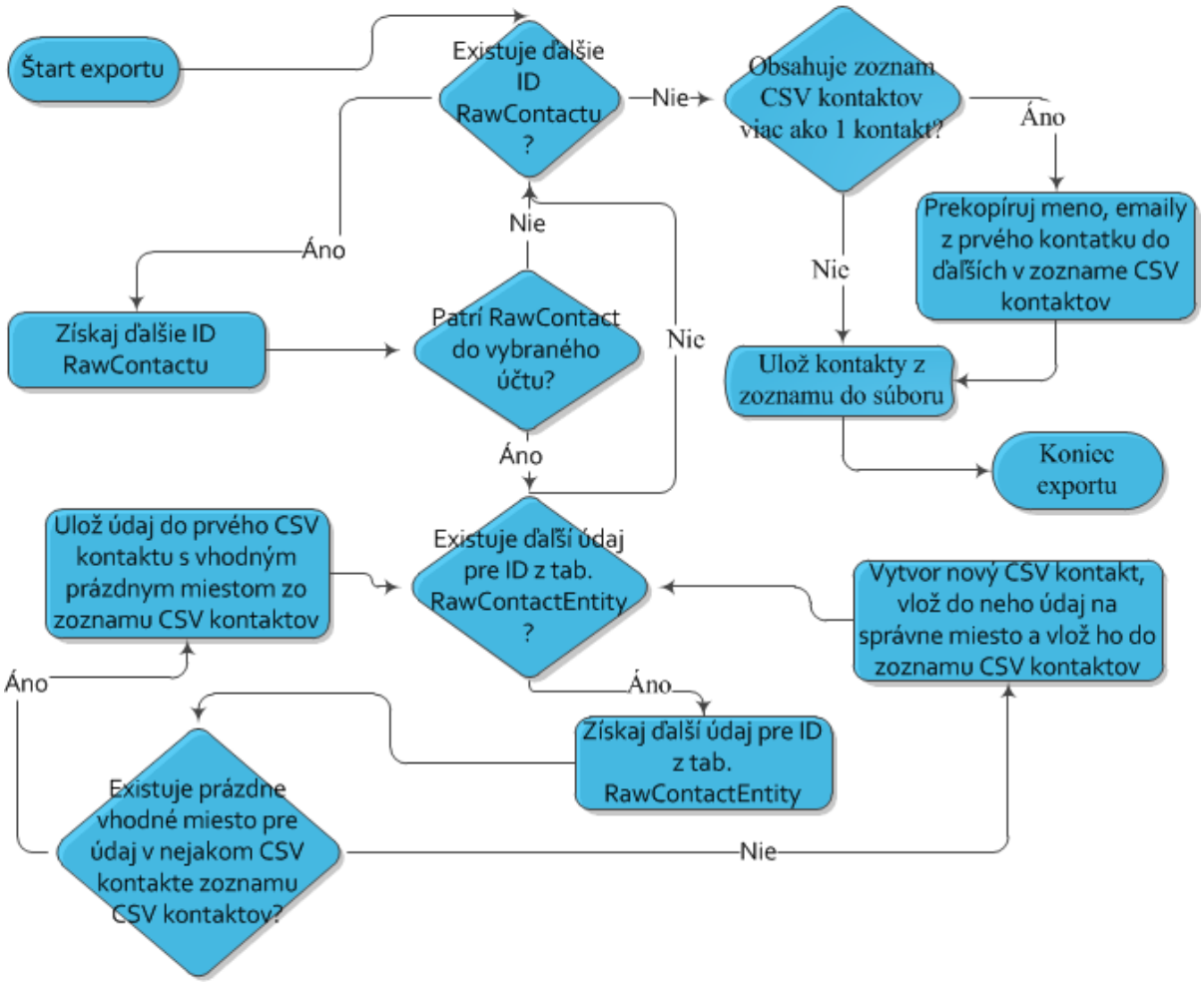
Trieda `CsvContact` reprezentuje 1 riadok CSV kontaktu. Dokáže uložiť aj Thunderbird aj Outlook CSV formát.

Trieda `Exporter` je navrhnutá ako abstraktná trieda. Je jadrom celého exportéra. Obsahuje abstraktné metódy, ktoré reprezentujú čiastkové funkcie, ktoré sú závislé na type CSV formátu. Takými funkciami sú napr. ukladanie údajov na správne miesta. Ďalej obsahuje implementované metódy, ktoré riadia celý export (viď. 5.1.4) a využívajú abstraktné metódy.

Triedy `OutlookExporter` a `ThunderbirdExporter` rozširujú triedu `Exporter`. Musia implementovať abstraktné metódy, ktoré sú špecifické pre jednotlivý formát. Pracujú s objektom triedy `CsvContact` a ukladajú údaje na správne pozície.

5.1.4 Princíp exportéra

V záujme získania všetkých údajov o kontakte uložených v Androide je potrebné siahnuť postupne do niekoľkých tabuliek. Získané údaje si následne uložíme do entity, ktorá reprezentuje jeden CSV kontakt. Najskôr je potrebné získať všetky ***RAW_CONTACT_ID*** kontaktov z tabuľky ***RawContacts***, ktoré patria do vhodného účtu. Pomocou získaného ***RAW_CONTACT_ID*** je potom možné získať z pohľadu ***RawContactEntity*** všetky údaje, ktoré sa uložia do CSV kontaktu na vhodné miesta. Duplicitné kontakty sa vytvoria podľa 5.1.2. Pri tvorbe návrhu exportéra sa navrhol zjednodušený vývojový diagram (viď. Obrázok 5.1), ktorý popisuje správanie exportéra v rôznych situáciách.



Obrázok 5.2 Vývojový diagram exportéra

5.2 Importér

5.2.1 Synchronizácia

Importér pri porovnávaní mien dvoch kontaktov používa väčšinu pravidiel agregácie kontaktov v Androide (viď. 2.5.5) okrem pravidla č. 3 a 5. Na komparáciu viac položkových údajov kontaktov je navrhnutá nová množina porovnávacích príznakov (viď. Tabuľka 5.1), ktorá je ďalej využitá na identifikáciu zhodnosti mien a poštových adries, ale aj na synchronizáciu údajov adries.

Princíp použitia príznakov vysvetlíme na príklade 2 poštových adries. Každý údaj adresy sa skladá zo 6 položiek ako ulica, pobox, mesto, PSČ, región, krajina. Vyberieme dva údaje adries z telefónu a z karty. Pre každú dvojicu položiek určíme príznak. Tiež je potrebné určiť, ktoré položky musia byť zhodné, aby mohli byť údaje prehlásené za zhodné, pretože v každom štáte sa používajú iné údaje pre ukladanie adresy. Práve z tohto dôvodu sme umožnili užívateľovi ako hlavnému aktérovi, aby si sám zvolil minimálne položky, v ktorých sa adresy musia zhodovať. Ak je príznak nejakej povinnej dvojice rovný **NEQ** alebo ani jeden z príznakov z povinných dvojíc nie je rovný **EQ**, tak adresa je prehlásená za nezhodnú. Ak je aspoň jeden príznak z povinných dvojíc rovný **EQ** a ostatné príznaky povinných dvojíc sú odlišné od **NEQ**, tak je dvojica adries prehlásená za zhodnú a

dochádza k synchronizácii položiek údajov. Tabuľka 5.1 popisuje postup pri synchronizácií, ktorý sa líši v závislosti od hodnoty príznaku dvojice položiek.

Táto nová množina príznakov je využitá aj na testovanie zhodnosti dvoch mien, pričom minimálne položky, v ktorých sa musia kontakty zhodovať, sú krstné meno a priezvisko. Vďaka tomuto prístupu je možné zistiť zhodu mien, ak jeden obsahuje iba krstné meno a to sa súčasne zhoduje s krstným menom druhého.

Príznak	Položka v telefóne	Operátor	Položka na karte	Popis, čo sa deje pri synchronizácii
NEQ	NEPRÁZDNA	!=	NEPRÁZDNA	Položky sú rôzne. Položka z karty sa kopíruje do položky v telefóne.
EQ	NEPRÁZDNA	==	NEPRÁZDNA	Položky sú zhodné. Zachováva sa položka v telefóne.
LOAD	PRÁZDNA		NEPRÁZDNA	Dochádza k nakopírovaniu položky z karty do telefónu.
OMIT	NEPRÁZDNA		PRÁZDNA	Zachováva sa položka v telefóne.
EMPTY	PRÁZDNA		PRÁZDNA	Obi dva sú prázdne. Zachováva sa položka v telefóne.

Tabuľka 5.1 Nová množina príznakov pre porovnanie zhody pri viac položkových údajoch

5.2.2 Princíp Importéra

Importér sa skladá z nasledovných troch procesov:

- Proces hľadania zhodných kontaktov,
- Proces vytvárania nového kontaktu,
- Proces synchronizácie 2 kontaktov.

Proces hľadania zhodných kontaktov sa začína získaním všetkých ***CONTACT_ID*** z tabuľky ***Contacts***, ktorých mená sa zhodujú a ich ukladaním do množiny ***CONTACT_ID***. Meno z telefónu je získané zo stĺpca *DISPLAY_NAME* z tabuľky ***Contacts***. Väčšinou je vyskladané z čiastkových položiek mena z tabuľky ***Data***. Jeho tvar je: <*krstné_meno stredné_meno priezvisko, sufix*>. CSV Thunderbird kontakt obsahuje iba dve položky, a to krstné meno a priezvisko. A preto sa môže stať, že kvôli schéme *DISPLAY_NAME* nebude dvojica kontaktov prehlásená ako zhodná.

Proces pokračuje v hľadaní a porovnávaní mena CSV kontaktu s telefónnym kontaktom v tabuľke ***Data***, kde porovnáva meno iba s druhom údajov označených ako *StructuredName*. ***CONTACT_ID*** kontaktov je ukladané do tej istej množiny ***CONTACT_ID***. V tomto kroku sa tiež uložia kontakty, ktoré majú prehodené mená alebo ak jeden kontakt obsahuje iba jedno meno a druhý obsahuje obidve.

Ďalej proces zisťuje, či získané ***CONTACT_ID*** kontaktov patrí k vybraným účtom. Postupuje tak, že vyhľadá v tabuľke ***RawContacts*** záznamy so zhodným ***CONTACT_ID***. Ak nájde zhodu, uloží si ***RAW_CONTACT_ID*** do množiny ***RAW_CONTACT_ID***. Ak užívateľ zadal, že k synchronizácii kontaktov má dochádzať okrem zhodného mena aj na základe aspoň jedného zhodného emailu a aspoň jedného zhodného telefónneho čísla, tak proces ďalej kontroluje, či nájdené ***RAW_CONTACT_ID*** majú zhodný nejaký email a súčasne nejaké telefónne číslo. V prípade, ak nájde zhodu, ponechá ich v množine ***RAW_CONTACT_ID***. V opačnom pripáde sú presunuté do ďalšej množiny, ktorá obsahuje kontakty, ktoré sú určené pre manuálnu synchronizáciu užívateľom.

Po ukončení procesu hľadania zhodných kontaktov sa rozhoduje na základe množín, či bude nasledovať proces vytvárania nového kontaktu alebo proces synchronizácie 2 kontaktov.

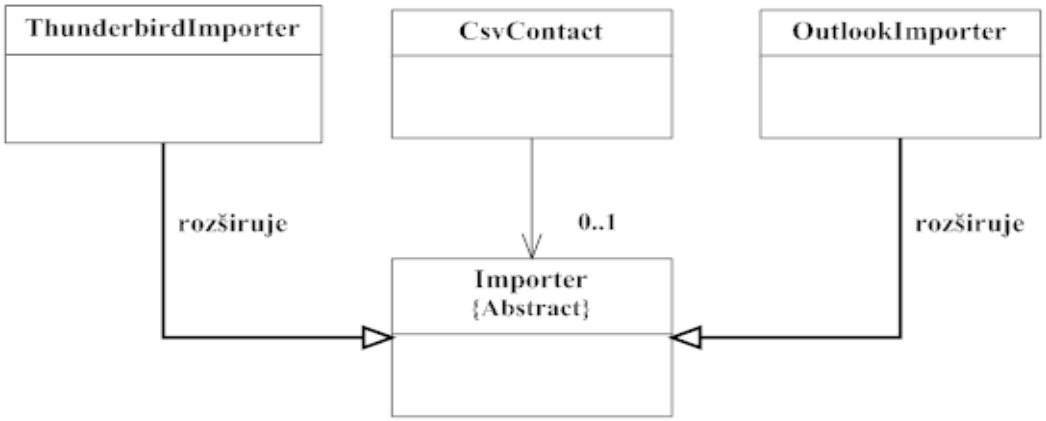
V procese **synchronizácie 2 kontaktov** dochádza k pripojeniu nových údajov CSV kontaktu k telefónnemu kontaktu.

V procese **vytvárania nového kontaktu** dochádza k vytvoreniu nového kontaktu z CSV súboru v telefóne pod vybraným účtom.

Tieto procesy sa opakujú pre každý CSV kontakt, až dovtedy, kým nenastane situácia, že už nemáme aký CSV kontakt importovať. Nakoniec sa zobrazia užívateľovi kontakty určené k manuálnej synchronizácii.

5.2.3 Návrh základných tried importéra

Pri čítaní dát z CSV kontaktu je nutné siahnuť k údajom na správnu pozíciu. Preto bol pri návrhu obecného diagramu tried importéra (viď. Obrázok 5.3) použitý rovnaký postup ako pri návrhu exportéra (viď. 5.1.2).



Obrázok 5.3 Obecný diagram tried importéra

Trieda `Importer` je navrhnutá ako abstraktná trieda, je jadrom celého importéra. Obsahuje abstraktné metódy, ktoré reprezentujú funkcie importéra a ktoré sú závislé na type CSV formátu. Takými funkciami sú napr. čítanie údajov zo správnych miest, vytváranie nového kontaktu alebo synchronizácia dvoch kontaktov. Táto trieda ďalej obsahuje implementované metódy, ktoré riadia celý import (viď. 5.1.4) a využívajú abstraktné metódy tejto triedy.

Triedy `OutlookImporter` a `ThunderbirdImporter` rozširujú triedu `Importer`. Musia implementovať abstraktné metódy, ktoré sú špecifické pre jednotlivý formát. Pracujú s objektom triedy `CsvContact` a čítajú údaje zo správnych pozícii.

6 Implementácia aplikácie

V tejto kapitole charakterizujeme spôsob implementácie dôležitých častí aplikácie, významných pomocných tried a GUI s dôrazom na odlišnosti implementácie od návrhu.

Celá práca je implementovaná v jazyku Java a frameworku Android SDK. Tento framework obsahuje všetky základné stavebné prvky na stavbu aplikácii pre Android. Pre tvorbu GUI je využitá definícia v XML externých súboroch, ktoré sú priradené k jednotlivým aktivitám.

6.1 Aplikácia

V súbore `AndroidManifest.xml` sú definované oprávnenia, ktoré aplikácia potrebuje pre manipuláciu s rôznymi časťami Android systému. Sú nimi ***READ_CONTACTS***, ***WRITE_CONTACTS***, ***WRITE_EXTERNAL_STORAGE***, ***GET_ACCOUNTS***. Tento súbor ďalej obsahuje deklaráciu využitých komponentov v tejto aplikácií a definíciu minimálnej verzie Androidu, ktorou je verzia 2.1.

V priečinku projektu `res/layout` sú definované XML layouty pre jednotlivé komponenty aplikácie a v priečinku `res/raw` sú uložené hlavičky CSV súborov, ktoré sú potrebné pre vytváranie nových CSV súborov.

Obe časti aplikácie využívajú pre načítanie a ukladanie údajov z/do CSV súboru ***opencsv*** knižnicu, ktorá zapuzdruje všetky vlastnosti a vymoženosti CSV formátu do metód a poskytuje jednoduché ovládanie. Táto knižnica je prevzatá z [20].

Trieda `StringUtils` slúži na nahradzovanie diakritických znakov z reťazca znakmi bez diakritiky. Je prevzatá z [21] a rozšírená o podporu slovenských a českých písmen s diakritickými znakmi.

6.1.1 Balík *bp.iemanager.csvcontact*

V balíku *bp.iemanager.csvcontact* sú obsiahnuté triedy pre prácu s Thunderbird a Outlook CSV formátom.

Trieda `CsvContact` reprezentuje jeden riadok CSV súboru – teda 1 CSV kontakt. Obsahuje atribúty ako pole reťazcov `csvLine`, ktorých počet sa mení v závislosti od typu CSV kontaktu a príznak `csvType`, ktorý určuje, o aký typ CSV kontaktu sa jedná. Obsahuje metódy, ktoré zistia, či sú na určitých pozíciach uložené požadované údaje, napr. `checkEmailThunderbird()`, `checkPostalWorkOutlook()`, atď. Táto trieda tiež obsahuje metódy pre uloženie hodnôt na určenú pozíciu `setString(poistion, string)`, načítanie hodnoty z pozície `getString(position)`. Podrobnejší popis je v priloženej Java dokumentácii.

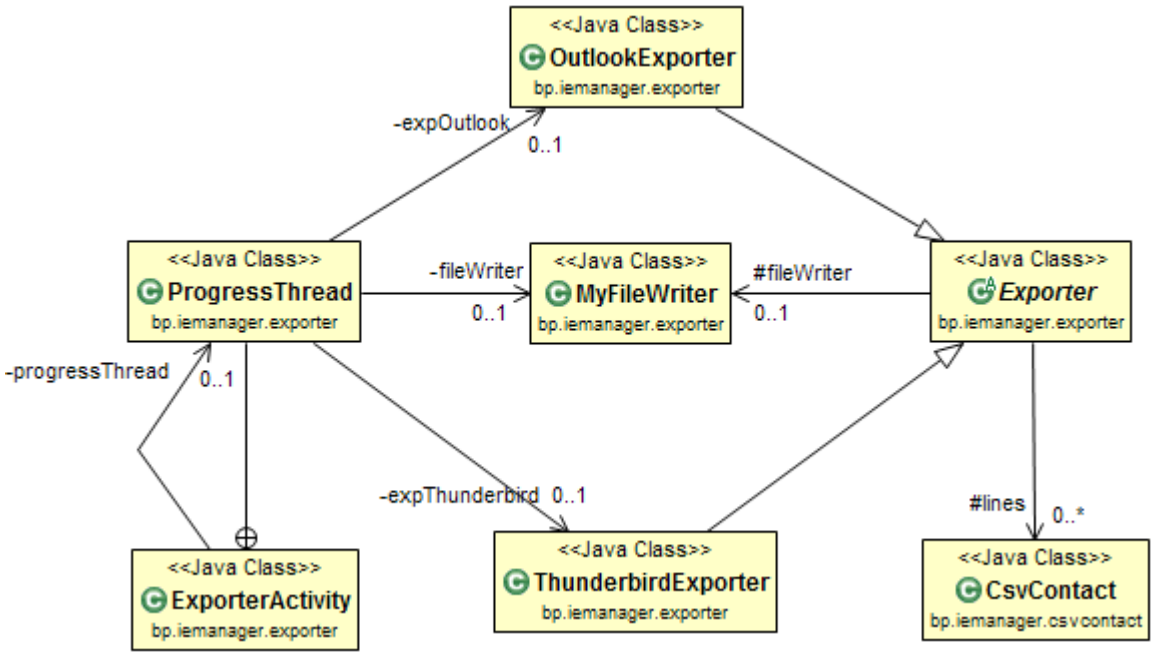
Triedy `OutlookConstants` a `ThunderbirdConstants` obsahujú definované statické konštanty, ktoré určujú pozície jednotlivých položiek údajov v CSV kontakte.

6.1.2 Trieda *IEManagerActivty*

Táto trieda reprezentuje úvodné okno, ktoré sa objaví pri spustení aplikácie. Dedí od triedy ***android.app.Activity*** a prekrýva niektoré jej metódy. GUI tejto aplikácie je definované v externom súbore `main.xml`. Obsahuje iba tri tlačidlá triedy ***Button***, pomocou ktorých je možné spustiť exportéra, importéra a ukončiť celú aplikáciu. Taktiež kontroluje, či je SD karta v zariadení dostupná.

6.2 Exportér

Na obrázku 6.1 je zobrazený diagram tried exportéra. Všetky triedy okrem `CsvContact` patria do balíka `bp.iemanager.exporter`, ktorý sa výhradne stará o export kontaktov a ich ukladanie do súboru na pamäťovú kartu. Najdôležitejšie triedy sú popísané v podkapitolách.



Obrázok 6.1 Diagram tried implementácie exportéra

6.2.1 Trieda *ExporterActivity*

Trieda `ExporterActivity` dedí od triedy ***android.app.Activity***. Je hlavným a zároveň jediným oknom exportéra. GUI je definované v externom súbore `export_layout.xml`. Obsahuje dve tlačidlá triedy ***Button*** pre spustenie exportu a prípadne navrátenie k hlavnému oknu aplikácie prvky triedy ***TextView***, ktoré umožňujú výber nastavení a ich vizualizáciu a prvok triedy ***EditText***, v ktorom je možné editovať názov exportovaného súboru. Všetky nastavenia je možné vyberať pomocou dialógov, ktoré sú implementované v preťaženej metóde `onDialogCreate()`. Vybrané nastavenia sú uložené do nastavovacích atribútov *`pickedAccounts`*, *`pickedExportType`*, *`fitFields`*, *`fileName`*, *`pickedCharset`*, *`removeAccentsFlag`*. V metóde `onCreate()` sú uložené referencie na jednotlivé prvky GUI, zaregistrované ich obsluhy v prípade stlačenia, inicializované a zobrazené hodnoty nastavovacích atribútov v príslušných prvkoch GUI.

Dialóg `DIALOG_PICK_ACCOUNTS` je určený k výberu účtov, z ktorých majú byť exportované kontakty, obsahuje iba vybrané účty (viď. 5). Získame ich pomocou metódy ***AccountManager.get(this).getAccounts()***. Táto metóda vracia všetky dostupné účty z Androidu.

Kódovanie výsledného súboru je zabezpečené pomocou dialógu `DIALOG_PICK_CHARSET_TYPE`. Všetky dostupné podporované kódovania sú získané metódou ***Charset.availableCharsets()*** z SDK javy. Tieto údaje sú zobrazené užívateľovi k výberu. Defaultne je nastavené kódovanie UTF-8.

V prípade stlačenia tlačidla (metóda `onStartExportButtonClicked()`) určeného k spusteniu exportu dochádza najskôr k overeniu, či boli zvolené všetky potrebné nastavenia. Ak áno,

tak sa vytvorí inštancia vnorenej triedy `ProgressThread`, ktorá spúšťa export v inom vlákne a zobrazí sa ***ProgressBar***, ktorý informuje o stave exportu. V opačnom prípade sa zobrazia nastavenia, ktoré neboli vybraté.

Vnorená trieda `ProgressThread` rozširuje triedu ***Thread***. `ProgressThread` sa stará o export kontaktov a priebežnú aktualizáciu ***ProgressBar***-u. Export musí prebiehať v inom vlákne, aby nespomaľoval hlavné okno exportéra. V konštruktore je vytvorená podľa typu exportovaného súboru (atribút `picketType`) buď inštancia triedy `ThunderbirdExporter` alebo `Outlook-Exporter` (viď. 6.2.3). Obom triedam sú predané nastavovacie atribúty triedy `Exporter-Activity`. V metóde `run()` dochádza k spusteniu exportu.

6.2.2 Trieda *Exporter*

Táto trieda je navrhnutá ako abstraktná trieda (viď. 5.1.3), ktorá implementuje iba metódy, ktoré sú spoločné pre obe triedy `OutlookExporter` a `ThunderbirdExporter`. K získaniu údajov je použitý rovnaký postup v oboch nástrojoch, avšak na určitých miestach treba uložiť údaje na iné pozície alebo inak pracovať s atribútom triedy. Tieto popísané funkcie sú definované pomocou abstraktných metód.

V konštruktore triedy sú predané parametre, ktoré si užívateľ sám navolil, aby ovplyvňovali samotný proces exportu. Trieda obsahuje atribút `listOfCsvContacts`, ktorý predstavuje zoznam objektov typu `CsvContact` (viď. 6.1.2). Metódou `startExport()` je spustený proces exportu (viď. 5.1.4). Tá následne volá metódu `queryAllRawContacts()`, ktorá najprv získa metódou `getCursorOfAllRawContacts()` objekt typu ***Cursor***, v ktorom sú všetky ***RAW_CONTACT_ID*** kontaktov z tabuľky ***RawContacts***, ktoré patria pod užívateľom nastavené účty. Metóda ďalej prechádza tento získaný ***Cursor***, z neho získava ***RAW_CONTACT_ID***, ktoré predáva metóde *`queryAllDetailedInformation(long RawContactId)`* a metódou `sendMessageToHandler()` posiela informácie ***ProgressBar***-u o stave exportu.

Metóda `queryAllDetailedInformation()` najprv získa ***Cursor*** obsahujúci všetky údaje o kontakte z tabuľky ***RawContactEntity***. Metóda potom prechádza postupne údaje z ***Cursor***-u a podľa typu údaju ich ukladá na správne miesto do objektu `CsvContact`. Ukladanie na správne miesto sa deje pomocou abstraktných metód `processEmail(Cursor)`, `process-Phone(Cursor)`, `processNote(Cursor)`, `processWebPages(Cursor)`, `process-NickName(Cursor)`, `processStructuredPostal(Cursor)`, `processStructured-Name(Cursor)`. Po získaní všetkých údajov metóda zapíše kontakt do súboru metódou `writeContactToFile()`. Metóda `writeContactToFile()` ešte pred samotným zápisom upravuje `CsvContact` podľa nastavených parametrov exportu. Napríklad môže odstraňovať diakritické znaky (metóda `replaceAllAccentChars()`), upraviť(metóda `editArrayList-Items()`) a ukladať kontakty (metóda `fileWriter.writeStrings()`).

6.2.3 Triedy *OutlookExporter* a *ThunderbirdExporter*

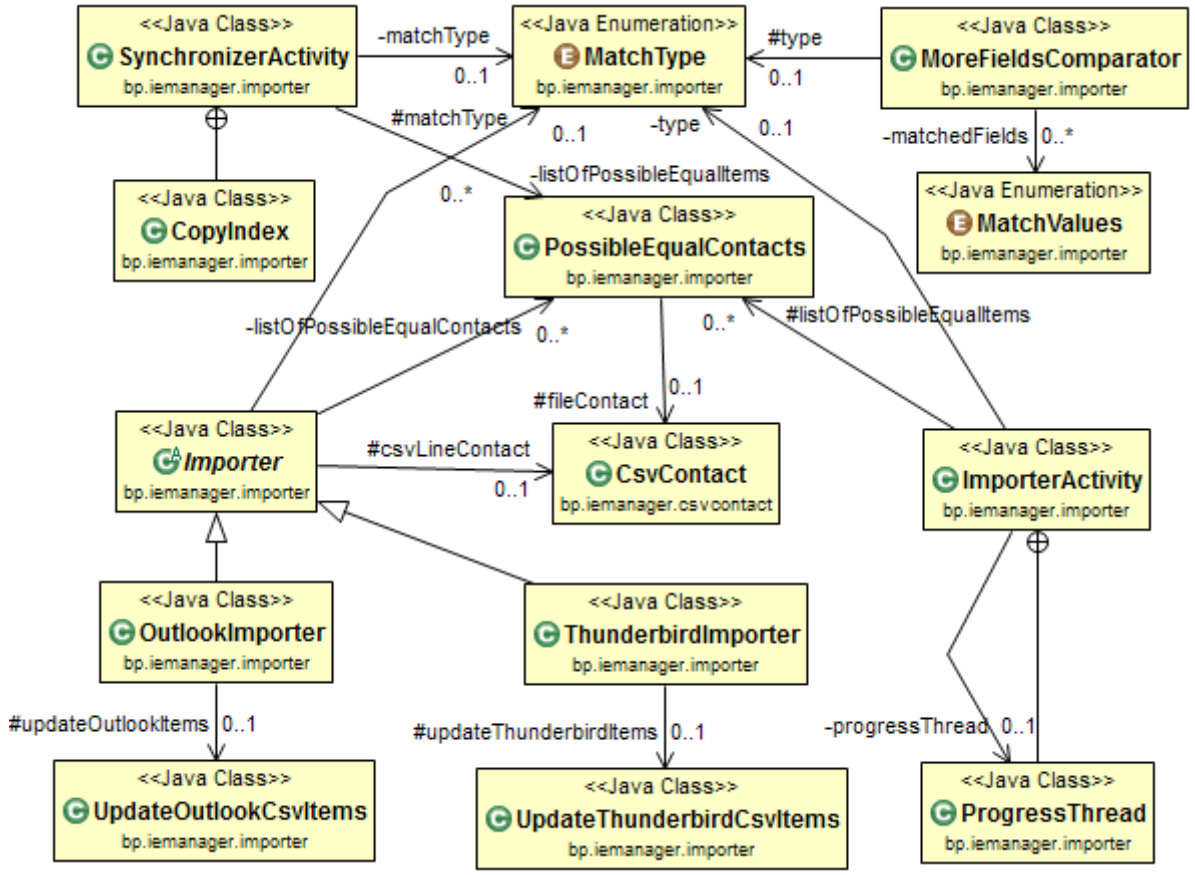
Obe triedy rozširujú triedu `Exporter` a implementujú všetky abstraktné metódy bázovej triedy uvedené v 6.2.2. Všetky tieto metódy ukladajú predané údaje do prvého `CsvContact`-u s voľným miestom vhodným pre údaj z rodičovského atribútu `listOfCsvContacts`. Indexy pre jednotlivé údaje získavajú z balíka `bp.iemanager.csvcontact` a konkrétne z tried `Thunderbird-Constants` a `OutlookConstants`. V prípade, ak chcú triedy ukladať do `CsvContact`-u, ktorý už má obsadený daný údaj, tak vytvárajú nový `CsvContact` s daným údajom a vkladajú ho do atri-

bútu `listOfCsvContacts` pomocou metódy `saveStringToArrayList(String, position)`.

Metóda `editArrayListItems()` prekopíruje údaje o mene z prvého `CsvContact`-u uloženého v `listOfCsvContacts` do všetkých ostatných `CsvContact`-ov z tohto zoznamu a označí ich špeciálnym príznakom (položka *Notes)*. Taktiež z prvého kontaktu skopíruje emaily do ostatných na pozície podľa typu exportovaného súboru (viď. 5.1.2).

6.3 Importér

Na Obrázku 6.2 je zobrazený diagram tried importéra. Všetky triedy okrem `CsvContact` patria do balíka `bp.iemanager.importer`, ktorý sa výhradne stará o import a synchronizáciu kontaktov. Najdôležitejšie triedy sú popísané v podkapitolách.



Obrázok 6.2 Diagram tried implementácie importéra

6.3.1 Pomocné triedy

Trieda `PossibleEqualContacts` ukladá dvojicu kontaktov, ktoré sú označené k manuálnej synchronizácii. Obsahuje atribúty `fileContact` typu `CsvContact`, ktorý má v sebe obsiahnuté všetky údaje o kontakte zo súboru a atribút `rawContactID` typu ***long***, ktorý reprezentuje možno zhodný Android kontakt.

Trieda `MoreFieldsComparator` zabezpečuje porovnanie viacerých položiek údaju dvoch kontaktov. Jednotlivé položky sú predané pomocou polí reťazcov v konštruktore triedy. Taktiež sú

mu predané indexy položiek, ktoré sa musia zhodovať. Metóda `compareStrings()` porovná jednotlivé dvojice položiek údajov, ktorých výsledky uloží do atribútu `matchedFields`. Metódy `areOrganisationsEqual()`, `areAddressesEqual(fields)` a `areNamesEqual()` analyzujú výsledky jednotlivých porovnaní a zisťujú či sú údaje zhodné. Pomocou metódy `createNewSyncData()` sa synchronizujú jednotlivé údaje, ktoré sú vrátené v poli reťazcov. Metóda `checkEqualMandatoryFields(fields)` zistí, či aspoň jedna z povinných položiek je zhodná a ostatné nie sú rozdielne.

Triedy `UpdateOutlookCsvItems` a `UpdateThunderbirdCsvItems` obsahujú množinu príznakov určených k zisteniu, ktoré údaje CSV kontaktu sú určené k pripojeniu ku Android kontaktu. Príznaky sú reprezentované pomocou atribútov typu ***boolean***. V konštruktore sú nastavené všetky príznaky údajov, ktoré sa v súbore nachádzajú na hodnotu ***true***, čo predstavuje sa má údaj pripájať ku telefónnemu kontaktu.

Enumerácia `MatchType` reprezentuje typy porovnávaní, ktoré sú použité pri porovnávaní údajov dvoch kontaktov (viď. 4.2). Enumerácia `MatchValues` reprezentuje novú množinu typov porovnaní (viď. Tabuľka 5.1).

6.3.2 Trieda *ImporterActivity*

Trieda `ImporterActivity` dedí od triedy ***android.app.Activity***. Je hlavným oknom importéra, kde sa dajú nastavovať rôzne možnosti importu a synchronizácie. GUI je definované v externom súbore `import_layout.xml`. Obsahuje tlačidlá triedy ***Button*** pre spustenie importu, navrátenie k hlavnému oknu aplikácie, výber súboru pomocou inej aplikácie a spustenie dialógu pre výber súboru s defaultného priečinku aplikácie. Ďalej obsahuje prvky triedy ***TextView***, ktoré umožňujú výber nastavení a ich vizualizáciu. Všetky nastavenia je možné vyberať pomocou dialógov, ktoré sú implementované v preťaženej metóde `onDialogCreate()`. Vybrané nastavenia sú uložené do nastavovacích atribútov triedy `pickedAccountForNewContact`, `matchType`, `pickedAccountsForSync`, `pickedFileWithFullPath`, `pickedCharset`, `minAddressFieldsToMatch`, `minFieldsForMatchTySync`. Posledný menovaný atribút udáva, na základe ktorých údajov bude dochádzať k synchronizácii (email, telefón) a predposledný menovaný na základe ktorých položiek je dvojica adries prehlásená za zhodnú. V metóde `onCreate()` sú uložené referencie na jednotlivé prvky GUI, zaregistrované ich obsluhy v prípade stlačenia, inicializované a zobrazené hodnoty nastavovacích atribútov v príslušných prvkoch GUI.

Atribút `listOfPossibleEqualContacts` ukladá všetky dvojice kontaktov určené k manuálnej synchronizácii.

Dialógové okná `DIALOG_PICK_CHARSET_TYPE`, `DIALOG_PICK_ACCOUNT` a `DIALOG_PICK_WITH_ACCOUNTS_TO_SYNCHRONIZE` sú implementované rovnako ako v triede `ExporterActivity` (viď. 6.2.1).

V prípade stlačenia tlačidla (metóda `onStartExportButtonClicked()`) určeného k spusteniu importu dochádza najskôr k overeniu, či boli zvolené všetky potrebné nastavenia. Ak áno, tak sa vytvorí inštancia vnorenej triedy `ProgressThread`, ktorá spúšťa import v inom vlákne a zobrazí sa `ProgressBar`, ktorý informuje o stave exportu. V opačnom prípade zobrazí, ktoré nastavenia neboli vybrané.

Vnorenú trieda `ProgressThread` rozširuje triedu ***Thread***. `ProgressThread` sa stará o import kontaktov a priebežnú aktualizáciu ***ProgressBar***-u. V metóde `run()` dochádza k importu kontaktov súboru. Každý kontakt je predaný novej inštancii, buď triedy `ThunderbirdExporter` alebo `OutlookExporter` (viď. 6.3.4) v závislosti na type CSV súboru, spolu s nastavovacími

atribútmi a atribútom `listOfPossibleEqualContacts`. Po dokončení importu sa kontroluje, či sú nejaké kontakty označené k manuálnej synchronizácii. Ak áno, tak je spustená aktivita `SynchronizerActivity`.

6.3.3 Trieda *SynchronizerActivity*

Trieda `SynchronizerActivity` dedí od triedy ***android.app.Activity***. Je hlavným oknom manuálnej synchronizácie. GUI je definované v externom súbore `synchronizer_layout.xml`. Obrazovka je rozdelená na 2 rovnako veľké časti, pričom obe obsahujú prvky triedy ***TextView*** slúžiace k zobrazeniu jednotlivých údajov o kontakte. Ľavá časť zobrazuje údaje o Android kontakte a pravá o CSV kontakte. GUI ďalej obsahuje tri tlačidlá triedy ***Button*** na ovládanie synchronizácii. Prvé synchronizuje aktuálne zobrazenú dvojicu kontaktov, druhé zobrazuje nasledujúci pár kontaktov a posledné ukončuje aktivitu `SynchronizerActivity` a vracia sa k aktivite exportéra.

Dvojice kontaktov, ktoré majú byť zobrazené užívateľovi sú uložené v atribúte `listOfPossibleEqualContacts`. Pomocou metód ako `populateWithThunderbird-Name(fileContact, phoneContact)`, `populateWithOutlookName(fileContact, phoneContact)` a podobných, sú jednotlivé údaje porovnané a zobrazené vo vhodnej dvojici prvkov ***TextView***. Ak sú údaje o kontaktoch zhodné, tak sú zobrazené na červeno pomocou HTML tagu span.

Údaje o Android kontakte sú získane pomocou metódy `Exporter.queryAll-DetailedInformation(rawContactId)` a následne z objektu `Exporter.getArray-List()`, ktorá vracia zoznam objektov `CsvContact`.

6.3.4 Trieda *Importer*

Táto trieda je navrhnutá ako abstraktná trieda (viď. 5.2.2), ktorá implementuje metódy `findContacts()`, `findInRawContacts()`. Obsahujú v sebe základný algoritmus pre import a synchronizáciu kontaktu. Na sporných miestach sa volajú abstraktné metódy, ktoré vyžadujú použitie konkrétnych údajov z CSV kontaktu.

Parametre konštruktoru obsahujú užívateľom zvolené nastavenia, ktoré ovplyvňujú beh importu a synchronizácie a údaje o importovanom kontakte. Tieto parametre sú uložené do atribútov tejto triedy. Trieda ďalej pozostáva z atribútov reprezentujúcich množiny využívajúce k synchronizácii:

- `setOfSureEqualRawContactId` – množina nájdených ***RAW_CONTACT_ID*** kontaktov určených k automatickej synchronizácii. Kontakty označené ako zhodné.
- `setOfSureEqualRawContactIdPhone` – množina nájdených ***RAW_CONTACT_ID*** kontaktov so zhodným telefónnym číslom.
- `setOfSureEqualRawCotanctIdEmail` – množina nájdených ***RAW_CONTACT_ID*** kontaktov so zhodným emailom.
- `setOfPossibleEqualRawContactId` – množina nájdených ***RAW_CONTACT_ID*** kontaktov určených k manuálnej synchronizácii. Kontakty označené ako možno zhodné.
- `setOfDisplayNameEqualContactsID` – množina nájdených ***CONTACT_ID*** kontaktov so zhodným menom.

Metódou `startImport()`, ktorá volá v sebe metódu `findContact()`, je spustený proces importu (viď. 5.2.2). Pre každý kontakt zo súboru je metóda `startImport()` spustená práve

jedenkrát. V metóde `findContact()` začína proces importu hľadaním zhodných kontaktov v telefóne.

Hľadanie zhodného kontaktu začína nájdením všetkých ***CONTACT_ID*** kontaktov z tabuľky ***Contacts*** so zhodným menom. Deje sa to tak, že sa najskôr získa objekt typu ***Cursor*** obsahujúci všetky ***CONTACT_ID*** kontaktov so zhodným menom pomocou metódy `queryForContactDisplayName(selection, selectionArgs)`, ktorému sú predané vhodné parametre pre dotaz. Všetky ***CONTACT_ID*** sú uložené do množiny `setOfDisplayNameEqualContactsID`. Ďalej sa pomocou abstraktnej metódy `checkStructureNameDeeply()` hľadajú kontakty s vymenenými menami alebo len čiastočnou zhodou mien. Posledná menovaná metóda tiež ukladá nájdené ***CONTACT_ID*** kontaktov do tejto množiny.

Hľadanie zhodných kontaktov pokračuje volaním metódy `findInRawContacts()`, ktorá najprv pre každé ***CONTACT_ID*** z množiny `setOfDisplayNameEqualContactsID` získa všetky ***RAW_CONTACT_ID*** kontakty uložené v tabuľke ***RawContacts*** patriace pod nastavené účty (metóda `queryForRawContatIdsOfContacts(contact_id)`). Podľa nastavení porovnania sa ďalej tieto ***RAW_CONTACT_ID*** ukladajú do vhodných množín.

Ak užívateľ nastavil možnosť, že kontakty sa majú synchronizovať iba na základe mena, tak sú získané ***RAW_CONTACT_ID*** presunuté do množiny `setOfSureEqualRawContactId`.

Ak užívateľ nastavil možnosť, že sa synchronizácia má vykonať na základe mena, emailu a telefónu, tak sú ***RAW_CONTACT_ID*** kontaktov s rovnakým emailom uložené do množiny `setOfSureEqualRawContactIdEmail` a s rovnakým telefónom uložené do množiny `setOfSureEqualRawContactIdPhone`. Samotné ukladanie sa vykonáva pomocou abstraktných metód `sortRawContactEmail(email, rawContactID)` a `sortRawContactPhone(phone, rawContactID)`. Keďže sú údaje o emaili alebo telefóne získavané z objektu typu ***Cursor*** pomocou cyklu, tak sú v prípade nezhody emailu alebo telefónu ***RAW_CONTACT_ID*** ukladané do množiny `setOfPossibleEqualRawContactId`. Čiže sa môže stať, že metóda uloží ***RAW_CONTACT_ID*** aj do množiny s možno zhodnými kontaktmi a aj do množiny so zhodnými kontaktmi. Preto treba množinu `setOfPossibleEqualRawContactId` neskôr upraviť.

Ak je nastavené, že sa kontakty majú synchronizovať aj na základe iného údaju ako mena, tak sa pomocou metódy `rearrangePartSetsToOne()` preorganizujú ***RAW_CONTACT_ID*** z množín podľa nastavení hľadania zhodných kontaktov do množiny `setOfSureEqualRawContactId`. A týmto končí hľadanie zhodných kontaktov.

Proces importu ďalej pokračuje zisťovaním počtu nájdených zhodných kontaktov a možno zhodných kontaktov podľa počtu prvok množín `setOfPossibleEqualRawContactId` a `setOfSureEqualRawContactId`. Ak nie je nájdený ani jeden taký kontakt, tak dochádza k vytvoreniu nového telefónneho kontaktu z karty pomocou abstraktnej metódy `createNewContact()`. Ak sú nájdené zhodné kontakty, tak pre každý sa volá abstraktná metóda `updateRawContact(rawContactID)`. A ak sú nájdené možno zhodné, tak najprv dochádza k rozdielu množiny `setOfPossibleEqualRawContactId` a `setOfSureEqualRawContactId`. Potom sú výsledné ***RAW_CONTACT_ID*** kontaktov z rozdielu množín uložené do `listOfPossibleEqualContactsId`, v ktorej sú obsiahnuté všetky dvojice kontaktov určené k manuálnej synchronizácii.

Ak je nájdený zhodný kontakt a splňa nastavenia užívateľa, tak dochádza k synchronizácii pomocou abstraktnej metódy `updateRawContact()`. Ak je označený nejaký kontakt ako možno zhodný, tak dochádza k uloženiu jeho ***RAW_CONTACT_ID*** do atribútu `listOfPossibleEqualContacts`. A ak nie je nájdený ani žiadny zhodný kontakt a ani možno zhodný kontakt, tak

dochádza k vytvoreniu nového kontaktu pod nastaveným účtom pomocou metódy `createNewContact()`.

Taktiež obsahuje statickú metódu `equal(string1, string, type)`, ktorá porovnáva dva reťazce na základe nastaveného typu porovnania.

6.3.5 Triedy *ThunderbirdImporter* a *OutlookImporter*

Obe triedy rozširujú triedu `Importer` a implementujú všetky abstraktné metódy bázovej triedy. Starajú sa o synchronizáciu kontaktu s iným kontaktom pomocou metódy `updateRawContact(rawContactID)` a o vytvorenie nového kontaktu pod nastaveným Android účtom pomocou metódy `createNewContact(csvContacts, accName, accType)`.

Trieda `ThunderbirdImporter` obsahuje atribút `updateThunderbirdItems` a trieda `OutlookImporter` obsahuje atribút `updateOutlookItems`. Slúžia k uloženiu príznakov o údajoch CSV kontaktu, ktoré majú byť vytvorené u nájdeného telefónneho kontaktu.

Metóda `updateRawContact(rawContactID)` porovnáva údaje CSV kontaktu s údajmi telefónneho kontaktu špecifikovaného parametrom metódy `rawContactID`. Ak sú porovnávané typy údajov, ako napr. poštová adresa, meno alebo organizácia a dvojica údajov sa zhoduje, tak sa do telefónneho údaju nahrajú položky z CSV údaju, ktoré telefónny údaj neobsahuje. Predchadzajúca opisovaná situácia sa vykoná pomocou metód `synchronizePostal()`, `synchronizeStructuredName()`, `synchronizeOrganization()`. Taktiež sa označí vhodný príznak nájdeného údaja v atribúte *`updateThunderbirdItems`* alebo v atribúte `updateOutlookItems` ako nájdený (hodnota ***false***). Ak sú porovnávané ostatné typy údajov a dvojica údajov sa zhoduje, tak sa iba označí vhodný príznak nájdeného údaja v atribúte `updateThunderbirdItems` alebo v atribúte `updateOutlookItems` ako nájdený (hodnota ***false***). Vykonáva sa to pomocou metód `synchronizePostal()`, `synchronizeEmails()`, `synchronizeWebSite()`, `synchronizeNote()`.

Nakoniec metódy `updateRawContact(rawContactID)` sa zavolá metóda `addNewItemsFromCsvToRawContact(ops, rawContactID)`, ktorá zabezpečuje uloženie všetkých nových údajov ku kontaktu špecifikovaného parametrom `rawContactID`. Metóda vie, ktoré údaje z CSV kontaktu má vybrať na základe príznakov (hodnota ***true***) údajov v atribúte *`updateThunderbirdItems`* alebo v atribúte `updateOutlookItems`.

7 Testovanie

V tejto kapitole sú popísané fázy a typy verzií a zariadení, na ktorých bola aplikácia testovaná.

7.1.1 Testovanie počas vývoja aplikácie na emulátore

Prvotné testovanie prebiehalo počas vývoja aplikácie na emulátore systému Android s verziou 2.1, ktorý je súčasťou SDK (viď. 2.3.1) a na verzii 2.3. V emulátore bol vytvorený špeciálny účet, ktorému bola prispôsobená aplikácia, pretože emulátor nepodporuje telefónny účet a Gmail účet podporuje len v zmenenej podobe. Testovanie prebiehalo v poradí, v akom boli jednotlivé časti aplikácie vyvíjané.

Najskôr bol testovaný export kontaktov, za účelom zistiť, či sa jednotlivé položky údajov z telefónu správne mapujú do položiek CSV súboru podľa návrhu mapovania (viď. 5.1.1) a či sa export správa podľa zvolených nastavení. Testovanie rôzne zvolených nastavení zahŕňalo vytváranie duplicitných kontaktov, export z viacerých účtov, podpora rôznych typov súboru a výsledné kódovanie súboru. Kontrola testov prebiehala na základe vizuálnej interakcie. V prípade, ak testy nedopadli podľa očakávaní, tak sa hľadala chyba, ktorá bola opravená. Ak testy prebehli úspešne, tak sa začal testovať import aplikáciou vyexportovaného súboru v nástrojoch MS Outlook a Thunderbird.

Ďalej bol testovaný import kontaktov a jeho nastavenia. Najprv sa testovalo korektné mapovanie jednotlivých položiek údajov z telefónu do položiek CSV súboru podľa návrhu mapovania (viď. 5.1.1). Potom sa testovalo vytváranie nových kontaktov a po ňom nasledovalo testovanie synchronizácie na základe mena a aspoň jednej emailovej adresy. Ak výsledky testov prebehli podľa očakávaní, tak bola testovaná synchronizácia aj na základe telefónneho čísla a neskôr na základe telefónneho čísla i emailovej adresy. Predposledným testom tejto časti aplikácie, bol test manuálnej synchronizácie. Bol vytvorený vlastný súbor s údajmi a ten bol importovaný do telefónneho zariadenia. Testovalo sa, či boli označené všetky kontakty v telefóne na manuálnu synchronizáciu a či sa zobrazené zhodné údaje kontaktov v manuálnej synchronizácii zafarbujú na červeno. Nakoniec sa testovali rôzne nastavenia importéra a synchronizácie. Testy nastavení zahŕňali synchronizáciu kontaktov s viacerými účtami telefónu a rôzne nastavenia typov porovnaní údajov. Ak všetky spomenuté testy prebehli úspešne, tak boli vyexportované súbory z nástrojov MS Outlook a Thunderbird, ktoré boli importované do telefónu. Kontrola všetkých testov prebiehala na základe vizuálnej interakcie.

7.1.2 Záverečné testovanie na reálnom zariadení

Záverečné testovanie prebiehalo na reálnom zariadení typu SE Xperia s verziou Androidu 2.3.3. Bol skúšaný import mojich kontaktov z nástroja Thunderbird a MS Outlook s kontaktmi môjho telefónu. Vizuálne sa kontrolovalo, či dochádza k nájdeniu zhodných kontaktov a správnemu pridruženiu údajov a či sa vytvárajú nové kontakty.

Taktiež bol testovaný export podobnými testmi, ako sú uvedené v 7.1.1.

8 Záver

Cieľom bakalárskej práce bolo vytvoriť nástroj pre mobilné zariadenia s operačným systémom Android, ktorý je schopný z/do pamäťovej karty importovať, exportovať údaje o kontaktoch uložených v natívnom programe systému Android a tieto kontakty zároveň synchronizovať. Hlavnou výhodou tohto nástroja je široká škála nastavovacích možností, ktoré si môže každý užívateľ zvoliť sám. Okrem toho nástroj je schopný pri importe kritických kontaktov ponúknuť užívateľovi možnosť synchronizácie s vyznačením zhodných kontaktov pomocou farby. Taktiež poskytuje funkciu celkového zálohovania všetkých kontaktov z telefónu , v porovnaní s inými aplikáciami poskytuje aj funkcie exportu a importu.

V budúcnosti môže byť nástroj rozšírený o podporu ďalších účtov systému Android, porovnanie zhodnosti kontaktov na základe viacerých údajov, či môže byť rozšírená a upravená trieda `SynchronizerActivity` k zobrazeniu manuálnej synchronizácie všetkých kontaktov alebo nástroj možno ešte prípadne rozšíriť o ďalšie formáty súborov.

Literatúra

[1] GOOGLE. *Android Developers*: *What is Android?* [online]. 2012 [cit. 2012-05-02]. Dostupné na URL: <http://developer.android.com/guide/basics/what-is-android.html>

[2] *Android-sk*: *Základné informácie o Androide* [online]. 2009 [cit. 2012-05-02]. Dostupné na URL: <http://android-sk.sk/index.php/co-je-android/44-zakladne-informacie-o-androide>

[3] WIKIPEDIA. *Wikipedia*: *Android (operating system)* [online]. 2010 [cit. 2012-05-02]. Dostupné na URL: <http://en.wikipedia.org/wiki/Android_(operating_system)#History>

[4] MEIER, Reto. *Professional Android 2 application development*. Indianapolis: Wiley, c2010, 543 s. Wrox programmer to programmer. ISBN 978-0-470-56552-0.

[5] GOOGLE. *Android Developers*: *Application Fundamentals* [online]. 2012 [cit. 2012-05-02]. Dostupné na URL: <http://developer.android.com/guide/topics/fundamentals.html>

[6] GOOGLE. *Android Developres*: *Installing the SDK* [online]. 2012 [cit. 2012-05-02]. Dostupné na URL: <http://developer.android.com/sdk/installing.html>

[7] *ABC Linuxu*: *Vyvíjíme pro Android – úvod* [online]. 2011 [cit. 2012-05-02]. Dostupné na URL: <http://www.abclinuxu.cz/clanky/vyvijime-pro-android-uvod>

[8] GOOGLE. *Android Developers*: *What is the NDK?* [online]. 2012 [cit. 2012-05-02]. Dostupné na URL: < http://developer.android.com/sdk/ndk/overview.html>

[9] GOOGLE. *Android Developers*: *Activities* [online]. 2012 [cit. 2012-05-02]. Dostupné na URL: <http://developer.android.com/guide/topics/fundamentals/activities.html>

[10] GOOGLE. *Android Developers*: *Services* [online]. 2012 [cit. 2012-05-02]. Dostupné na URL: <http://developer.android.com/guide/topics/fundamentals/services.html>

[11] GOOGLE. *Android Developers*: *BroadcastReceiver* [online]. 2012 [cit. 2012-05-02]. Dostupné na URL: <http://developer.android.com/reference/android/content/BroadcastReceiver.html>

[12] GOOGLE. *Android Developers*: *Content Provider Basics* [online]. 2012 [cit. 2012-05-02]. Dostupné na URL: <http://developer.android.com/guide/topics/providers/content-provider-basics.html>

[13] GOOGLE. *Android Developers*: *ContentProvider* [online]. 2012 [cit. 2012-05-02]. Dostupné na URL: <http://developer.android.com/reference/android/content/ContentProvider.html>

[14] GOOGLE. *Android Developers*: *User Interface* [online]. 2012 [cit. 2012-05-02]. Dostupné na URL: <http://developer.android.com/guide/topics/ui/index.html>

[15] GOOGLE. *Android Developers*: *Using the Contacts API* [online]. 2012 [cit. 2012-05-02]. Dostupné na URL: <http://developer.android.com/resources/articles/contacts.html>

[16] GOOGLE. *Android Developers*: *RawContactsEntity* [online]. 2012 [cit. 2012-05-02]. Dostupné na URL: <http://developer.android.com/reference/android/provider/ContactsContract.RawContactsEntity.html>

[17] IETF. *Common Format and MIME Type for Comma-Separated Values (CSV) Files* [online]. 2005 [cit. 2012-05-09]. Dostupné na URL: <http://www.ietf.org/rfc/rfc4180.txt>

[18] IETF. *VCard MIME Directory Profile* [online]. 1998 [cit. 2012-05-09]. Dostupné na URL: <http://www.ietf.org/rfc/rfc2426.txt>

[19] IETF. *The LDAP Data Interchange Format (LDIF) - Technical Specification* [online]. 2000 [cit. 2012-05-09]. Dostupné na URL: <http://www.ietf.org/rfc/rfc2849.txt>

[20] *Opencsv* [online]. 2011 [cit. 2012-05-03]. Dostupné na URL: <http://opencsv.sourceforge.net>

[21] *Real's How To* [online]. 2011 [cit. 2012-05-03]. Dostupné na URL: <http://www.rgagnon.com/javadetails/java-0456.html>

Zoznam použitých skratiek a symbolov

API	Application programming interface – rozhranie pre programovanie aplikácií.
CPU	Central Processing Unit – hlavný procesor počítača.
CSV	Comma seperated values – súborovy formát určený pre výmenu tabuľkových hodnôt používajúci ako oddeľovač čiarku CSV.
GUI	Graphical User Interface – grafické uživateľské rozhranie
HTML	HyperText Markup Language – značkovací jazyk na vytváranie webových stránok a iných informácií zobraziteľných vo webovom prehliadači.
LDIF	LDAP Data Interchange Format – formát pre reprezentáciu a aktualizáciu dát na LDAP servery.
LIFO	Last in, first out – postup ukladania do pamäte, pri ktorom sa prvky, ktoré sa uložili ako posledné, vyberú ako prvé.
NDK	Native Development Kit – je sada nástrojov používaných len v spojení s Android SDK, ktoré umožňujú vývojárom vytvárať kritické časti ich aplikácii s využitím natívneho C/C++ kódu.
OHA	Open handset alliance – konzorcium 84 firiem vyvíjajúcich otvorené štandardy pre mobilné zariadenia.
PSČ	Poštové smerové číslo.
SDK	Software Development Kit – softwarový balík pre vyvíjanie softwaru pre Android zariadenia.
TAB	Tab seperated values – súborovy formát určený pre výmenu tabuľkových hodnôt používajúci ako oddeľovač znak tabulátoru.
URI	Uniform Resource Identifier – kompaktný reťazec znakov používaný na indetifikáciu alebo pomenovanie zdroja.
XML	eXtensible Markup Language – rozšíriteľný ynačkovací jazyk, ktorý bol vivinutý a štandardizovaný konzorciom W3C ako pokračovanie jazyka SGML a zovšeobecnenie jazyka HTML.

Zoznam obrázkov

Zoznam tabuliek

Zoznam príloh

Príloha A: Mapovanie údajov Thunderbird CSV formátu

Mapovanie údajov Android kontaktu do Thunderbird CSV formátu

First Name	Last Name	Display Name	Nickname	Primary Email	Secondary Email
SN.GIVEN_NAME	*SN.FAIMILY_NAME*	*SN.DISPLAY_NAME*	*NM.NAME*	*E.ADDRESS*	*E.ADDRESS*
Screen Name	Work Phone	Home Phone	Fax Number	Pager Number	Mobile Number
x	*P.TYPE_WORK*	*P.TYPE_HOME*	*P.FAX_WORK || P.FAX_HOME*	*P.TYPE_PAGER*	*P.TYPE_MOBILE*
Home Address	Home Address 2	Home City	Home State	Home ZipCode	Home Country
SP.STREET	*SP.TYPE_HOME*	*SP.CITY*	*SP.REGION*	*SP.POSTCODE*	*SP.COUNTRY*
Work Address	Work Address 2	Work City	Work State	Work ZipCode	Work Country
SP.STREET	*SP.TYPE_WORK*	*SP.CITY*	*SP.REGION*	*SP.POSTCODE*	*SP.COUNTRY*
Job Title	Department	Organization	Web Page 1	Web Page 2	Birth Year
x	*O.TITLE*	*O.COMPANY*	*W.URL*	*W.URL*	*x*
Birth Month	Birth Day	Custom 1	Custom 2	Custom 3	Custom 4
x	*x*	*x*	*x*	*x*	*x*
Notes					
N.NOTE					

Legenda

Názov triedy konštant z balíku CommonDataKinds	Popis	Zkratka
StructuredName	druh údajov reprezentujúci meno	SN
StructuredPostal	druh údajov reprezentujúci poštovú adresu	SP
Email	druh údajov reprezentujúci emailovú adresu	E
Phone	druh údajov reprezentujúci telefónne čísla	P
Organization	druh údajov reprezentujúci organizáciu	O
Im	druh údajov reprezentujúci IM adresu	IM
Nickname	druh údajov reprezentujúci prezývku	NM
Website	druh údajov reprezentujúci webovú stranku	W
Note	poznámky o kontakte	NM

Mapovanie údajov Android kontaktu do Outlook CSV formátu

Title	First Name	Middle Name	Last Name	Suffix	Company	Department
SN.PREFIX	*SN.GIVEN_NAME*	*SN.MIDDLE_NAME*	*SN.FAIMILY_NAME*	*SN.SUFFIX*	*O.COMPANY*	*x*
Job Title	Business Street	Business Street 2	Business Street 3	Business City	Business State	Business PostCode
O.TITLE	*SP.STREET*	*SP.TYPE_WORK*	*x*	*SP.CITY*	*SP.REGION*	*SP.POSTCODE*
Business Country/Region	Home Street	Home Street 2	Home Street 3	Home City	Home State	Home PostCode
SP.COUNTRY	*SP.STREET*	*SP.TYPE_HOME*	*x*	*SP.CITY*	*SP.REGION*	*SP.POSTCODE*
Home Country/Region	Other Street	Other Street 2	Other Street 3	Other City	Other State	Other PostCode
SP.COUNTRY	*SP.STREET*	*SP.TYPE_OTHER*	*x*	*SP.CITY*	*SP.REGION*	*SP.POSTCODE*
Other Country/Region	Assistant's Phone	Business Fax	Business Phone	Business Phone 2	Callback	Car Phone
SP.COUNTRY	*x*	*P.FAX_WORK*	*P.TYPE_WORK*	*P.TYPE_WORK*	*x*	*x*
Company Main Phone	Home Fax	Home Phone	Home Phone 2	ISDN	Mobile Phone	Other Fax
x	*P.FAX_HOME*	*P.TYPE_HOME*	*P.TYPE_HOME*	*x*	*P.TYPE_MOBILE*	*P.FAX_OTHER*
Other Phone	Pager	Primary Phone	Radio Phone	TTY/TDD Phone	Telex	Account
P.TYPE_OTHER	*P.TYPE_PAGER*	*x*	*x*	*x*	*x*	*x*
Anniversary	Assistant's Name	Billing Information	Birthday	Business Addr PO Box	Categories	Children
x	*x*	*x*	*x*	*SP.POBOX*	*x*	*x*
Directory Server	E-mail Address	E-mail Type	E-mail DisplayName	E-mail 2 Address	E-mail 2 Type	E-mail 2 DisplayName
x	*E.ADDRESS*	*x*	*x*	*E.ADDRESS*	*x*	*x*
E-mail 3 Address	E-mail 3 Type	E-mail 3 Display Name	Gender	Government ID Num	Hobby	Home Addr POBox
E.ADDRESS	*x*	*x*	*x*	*x*	*x*	*SP.POBOX*
Initials	Internet Free Busy	Keywords	Language	Location	Manager's Name	Mileage
x	*x*	*x*	*x*	*x*	*x*	*x*
Notes	Office Location	Organizational ID Num	Other Addr POBox	Priority	Private	Profession
N.NOTE	*x*	*x*	*SP.POBOX*	*x*	*x*	*x*
Referred By	Sensitivity	Spouse	User 1	User 2	User 3	User 4
x	*x*	*x*	*x*	*x*	*x*	*x*
Web Page						
W.URL						

Legenda
Rovnaká ako u predchádzajúceho

Príloha C: Manuál

Najprv je potrebné nahrať *.apk* súbor aplikácie z priloženého CD do Android zariadenia na pamäťovú kartu a následne spustiť inštaláciu tohto súboru. Systém Android aplikáciu sám nainštaluje a vytvorí odkaz na aplikáciu.

Príloha D: Obsah CD

Priložené CD obsahuje nasledujúce materiály:

- **Technická písomná správa** – súbor *sprava.pdf*
- **Technická písomná správa** – súbor *sprava.docx*
- **Zdrojové kódy nástroja** – adresár *./source_codes*
- **Výsledná preložená aplikácia** – adresár *./app*
- **Vygenerovaná javadoc dokumentácia** – adresár *./javadoc*
- **Uživateľská príručka** – súbor *user_manual.pdf*