Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Práce s kontakty na mobilní platformě Android

Plzeň 2014 David Fiedler

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 9. května 2014

David Fiedler

Abstract

This thesis is concerned with management of different contacts on the Android platform. Currently on the android platform, contact information related to one person are often scattered in many different contacts. The goal of this project is to find a way to locate these contacts and connect them together in the same place.

The application has been implemented, allowing you to select the name and photo of the linked contact. On devices with the administration rights the application can work with contacts from Facebook, it can also find more contacts to link in other tested applications. Considering all these factors, this project is making a step forward with this issue compared with other current applications that are available.

Abstrakt

Tato práce se zabývá správou kontaktů na platformě Android. Na platformě Android bývají kontaktní informace vztahující se k jedné osobě roztroušeny v mnoha kontaktech. Cílem práce je najít způsob, jak tyto kontakty vyhledat a propojit.

Zvolená aplikace byla realizována, umožňuje vybrat název a fotografii propojeného kontaktu. Na zařízeních s administrátorskými právy umí aplikace pracovat i s kontakty z Facebooku, navíc najde řádově více kontaktů k propojení než ostatní testované aplikace. Všechny tyto faktory znamenají posun vpřed ve srovnání s ostatními dostupnými aplikacemi zabývajícími se touto problematikou.

Obsah

1 Úvod **1**
- 1.1 Cíle bakalářské práce . . . 1
- 1.2 Cíle aplikace . . . 2

2 Existující aplikace pro práci s kontakty **3**
- 2.1 Testované aplikace . . . 3
  - 2.1.1 Aplikace Merge+ . . . 3
  - 2.1.2 Contact Optimizer . . . 5
  - 2.1.3 Duplicate Contacts Manager . . . 5
- 2.2 Shrnutí existujících aplikací . . . 6

3 Práce s kontakty na platformě Android **7**
- 3.1 Content Provider . . . 7
  - 3.1.1 Přístup k provideru . . . 8
  - 3.1.2 Content URI . . . 8
  - 3.1.3 Získání dat z provideru . . . 9
  - 3.1.4 Zobrazení výsledků dotazu . . . 9
  - 3.1.5 Vkládání, aktualizace a mazání dat . . . 10
  - 3.1.6 Kontrakty . . . 10
- 3.2 Contact Provider . . . 10
  - 3.2.1 Organizace kontaktů . . . 11
  - 3.2.2 Raw contacty . . . 12
  - 3.2.3 Data . . . 12
  - 3.2.4 Kontakty . . . 14
  - 3.2.5 Oprávnění potřebná pro práci s kontakty . . . 15
  - 3.2.6 Propojování kontaktů . . . 16
- 3.3 Nedostatky v oficiální dokumentaci . . . 17
  - 3.3.1 Práce s kontakty z Facebooku . . . 17
  - 3.3.2 Změna názvu kontaktu . . . 18

4 Struktura aplikace **19**
- 4.1 Prezentační vrstva . . . 19
  - 4.1.1 Fragmenty . . . 19
  - 4.1.2 Vynechání aktivit z historie . . . 21

4.1.3 Stručný popis aktivit 22
4.2 Aplikační vrstva 27
4.2.1 Systém vyhledávacích modulů 28
4.2.2 Implementované moduly 29
4.2.3 Parazitní moduly 30
4.3 Datová vrstva 30
4.3.1 Získávání kontaktů k porovnání 30
4.3.2 Propojování kontaktů 31
4.3.3 Získávání facebookových kontaktů z databáze 32
4.4 Kompatibilita 35

5 Testování aplikace **36**
5.1 Porovnání aplikací 36
5.2 Porovnání různých nastavení 36
5.2.1 Email module 38
5.2.2 Phone number module 38
5.2.3 Name module 38
5.2.4 Email/Name module 39

6 Závěr **40**

Přílohy **45**
A Instalační příručka 45
B Uživatelská příručka 45
C Sloupce tabulky kontaktů 49
C.1 Základní 50
C.2 Dostupné přes implicit-join 51
D Sloupce tabulky raw kontaktů 52
D.1 Základní 52
D.2 Dostupné přes implicit-join 53
E Sloupce tabulky dat 54
E.1 Základní 54
E.2 Dostupné přes implicit-join 54

1 Úvod

Tématem této práce je vytvořit účinný a spolehlivý nástroj pro spojování kontaktů na platformě Android, patřících k jedné fyzické osobě. Hlavní cíle práce jsou inteligentní vyhledávání kontaktů k propojení, propojování kontaktů z více zdrojů, v neposlední řadě pak rychlost a jednoduchost aplikace.

Kontakty na platformě Android v mnohém připomínají kontakty na starých telefonech bez operačního systému. Uživatelské rozhraní je sice modernější, stará struktura s dlouhým seznamem kontaktů pod sebou ale v kontaktní aplikaci zůstala. Podstatný rozdíl, kromě technické stránky, která je probrána v následující kapitole, je to, že se v zařízení s operačním systémem Android mohou nacházet nejen kontakty zadané uživatelem v zařízení (telefonu, tabletu...), ale i z jiných zdrojů. Těmito zdroji mohou být například emailové účty, či účty na sociálních sítích.

Začlenění kontaktů z více zdrojů má své výhody, uživatel má všechny informace na jednom místě, z jednoho seznamu může volat, psát mail, či najít kontaktní informace na Facebooku. S výhodami přichází ale i úskalí rozdrobení informací - různé údaje o jedné osobě jsou v seznamu kontaktů na několika místech, podle toho, z jakého účtu byly importovány. Na starších zařízeních vznikali takové kontakty jen chybou při zadání, nebylo jich tedy mnoho. Na zařízeních s operačním systémem Android, kde uživatel synchronizuje emailové účty či účty na sociálních sítích se seznamem kontaktů, jde však počet takových kontaktů do desítek, jejich hledání a ruční spojování v seznamu kontaktů o několika stech položkách je pak nelehkým úkolem.

Platforma Android má pro tuto situaci částečné řešení. Všechny vytvořené nebo importované kontakty kontroluje s již existujícími, při přesné nebo téměř přesné shodě pak nabízí propojení kontaktů. Cílem mojí práce je překonat tento systém složitějším a inteligentnějším, takovým, který nalezne naprostou většinu kontaktů patřících ke stejné osobě.

1.1 Cíle bakalářské práce

1. Prozkoumat existující aplikace na propojování kontaktů na platformě Android a zhodnotit jejich použitelnost
2. Prozkoumat technické detaily práce s kontakty na platformě Android.
3. Navrhnout a realizovat aplikaci pro propojování kontaktů na platformě Android.

1.2 Cíle aplikace

1. Propojovat kontakty z více zdrojů.
2. Vyšší počet nalezených kontaktů k propojení než konkurenční aplikace.
3. Schopnost pracovat s kontakty ze sociálních sítí.
4. Snadná rozšiřitelnost aplikace - především o další způsoby kontaktů k propojení
5. Jednoduché uživatelské rozhraní.

2 Existující aplikace pro práci s kontakty

V této kapitole budou stručně vyjmenovány vybrané existující aplikace pro propojování kontaktů na platformě Android. Nastíněn bude stručný výčet jejich vlastností a možností, na konci pak bude zdůvodněna potřeba vytvoření další aplikace.

Při výzkumu možností a schopností cizích aplikací jsem vycházel výhradně z vlastní zkušenosti, neboť zdrojový kód těchto aplikací není zveřejněn. Proto se mohou ve výčtu schopností a možností vyskytnout chyby, hodnocení tedy nelze považovat za zcela objektivní.

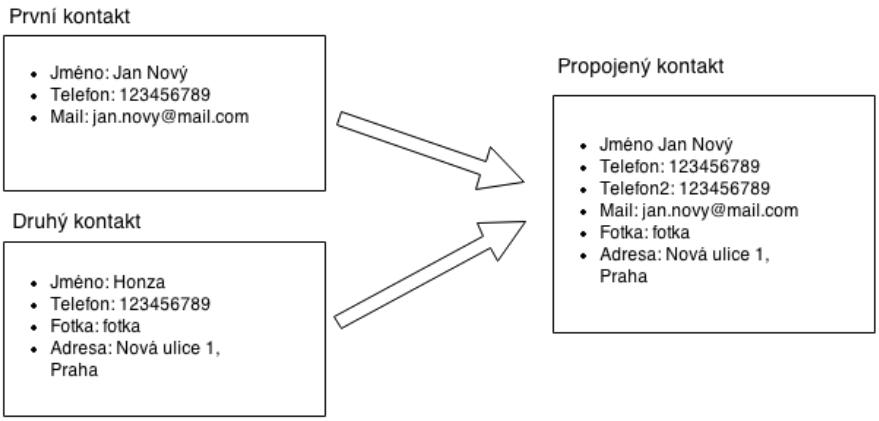
2.1 Testované aplikace

Testovány byly celkem tři aplikace. Obecně lze říci, že všechny aplikace mají většinou více funkcí než jen hledání duplicitních kontaktů. Také filozofie programů je odlišná od zadání a cílů mojí bakalářské práce, primárním cílem bývá většinou odstranění duplicit. Cílem této práce je naproti tomu využít duplicitních informací ke sloučení více kontaktů do jednoho, duplicitní data neřeší. Tento přístup sice nemusí vést k úspoře dat ani k urychlení kontaktní aplikace, ušetří však čas uživateli, který se v kontaktech lépe vyzná. Duplicitní údaj totiž nebývá jediným údajem kontaktu, spolu s ním se spojí i další, již unikátní data (viz. obrázek 2.1).

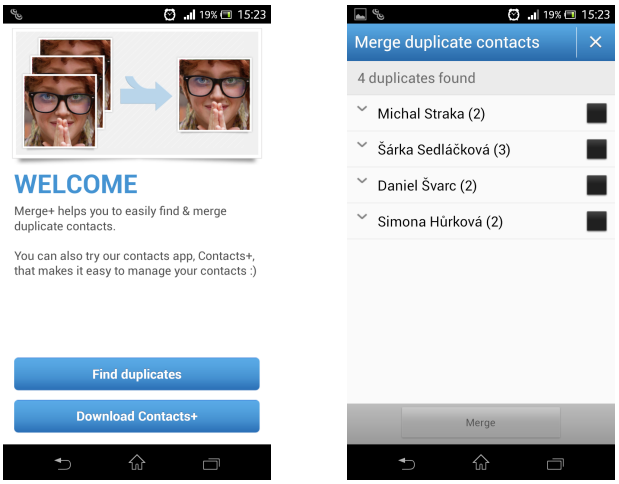
Všechny testované aplikace byly staženy z GooglePlay (hlavní centrum a obchod s aplikacemi na platformě Android) a jsou zdarma. Aplikace Duplicate contact manager má některé nadstandardní funkce placené, tyto nebyly testovány.

2.1.1 Aplikace Merge+

Aplikace Merge+ je jednoduchá až minimalistická aplikace pro propojování kontaktů. Dle mého názoru se jedná hlavně o promo na aplikaci Contacts+. Hledání je rychlé, ale nalezených výsledků není mnoho. Navíc nelze zvolit výchozí fotku ani jméno kontaktu. Uživatelské rozhraní aplikace můžeme vidět na obrázku 2.2.

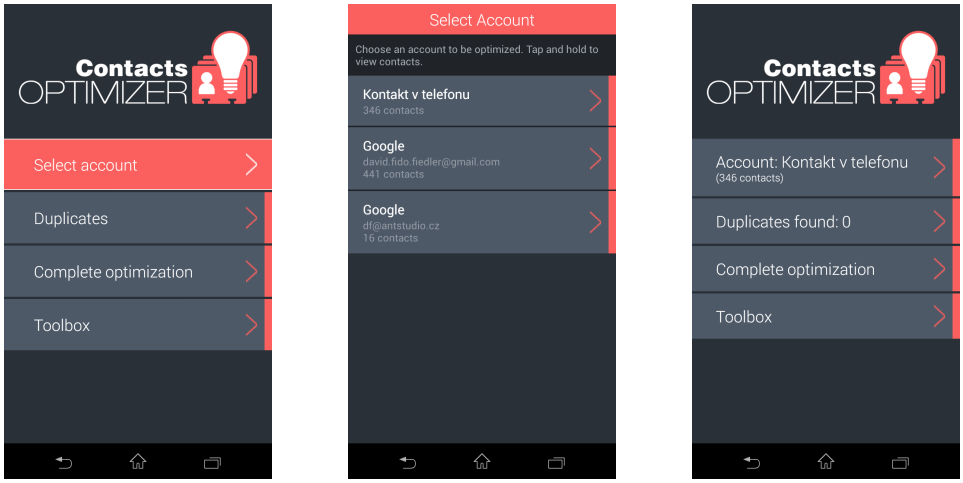


Obrázek 2.1: Výhody propojování kontaktů.



(a) Úvodní obrazovka. (b) Seznam nalezených kontaktů patřících k téže osobě.

Obrázek 2.2: Aplikace Merge+.



(a) Úvodní obrazovka. (b) Výběr účtu k optimalizaci/hledání. (c) Výsledky hledání pro kontakty v telefonu (ručně zadané uživatelem).

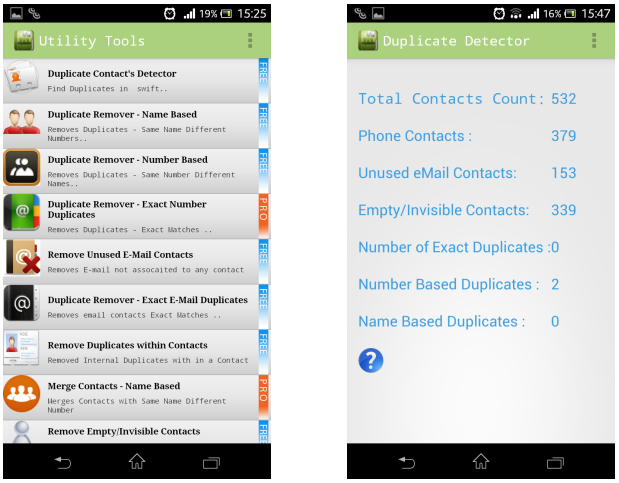
Obrázek 2.3: Aplikace Contact Optimizer.

2.1.2 Contact Optimizer

Aplikace Contact Optimizer nabízí mnohem více funkcí než jen hledání duplicit a propojování kontaktů. Pokud se ale podíváme jen na tuto funkčnost, výsledky nejsou příliš dobré. Aplikace hledá duplicity jen v rámci jednoho účtu (jeden emailový účet, kontakty v telefonu..), navíc nabízí jen smazat jeden z kontaktů. Propojování kontaktů probíhá odděleně v jiné části programu nazvané optimalizace. Zde ovšem výsledek vůbec nelze zhodnotit, k propojení jsou nabízené i kontakty již dávno propojené, celá logika tohoto procesu je tak pochybná. Navíc při přepnutí na jinou aplikaci a zpět je nutno opakovat celý proces od začátku, nalezené výsledky se nikam neuloží. Aplikaci můžeme vidět na obrázku 2.3.

2.1.3 Duplicate Contacts Manager

Další aplikace s mnoha funkcemi zaujme hned zpočátku patkovým písmem, pro digitální techniku tak netypickým. Na rozdíl od předchozí aplikace se většina funkcí týká hledání duplicitních kontaktů. V hledání duplicit ale program zaostává, při pokusu s modulem Duplicite Contacts Detector nalezl jen několik málo shod dle čísla. Vzhled aplikace vidíme na obrázku 2.4.



(a) Úvodní obrazovka. (b) Nalezené výsledky při použití funkce duplicate contact detector.

Obrázek 2.4: Aplikace Contact Optimizer.

2.2 Shrnutí existujících aplikací

Z průzkumu aplikací jednoznačně vyplynula potřeba vytvořit novou aplikaci. Počet nalezených shod byl ve všech třech programech velmi malý. Není možné to ověřit, ale výsledky hledání nasvědčovali tomu, že se testuje pouze přesná shoda nalezených údajů.

Dále v aplikacích nelze zvolit jméno nebo fotku propojeného kontaktu. Práce s kontakty pocházejícími ze sítě Facebook pak není možná opět v ani jednom z programů. Velkým rozdílem, jak již bylo řečeno v úvodu kapitoly, je rozdílná filozofie, kdy testované programy až na výjimky duplicity mažou, o propojení kontaktů se nesnaží. Všechny tyto skutečnosti pak, dle mého názoru, vedou k potřebě vytvořit novou aplikaci, zaměřenou pouze na propojování kontaktů, která bude mít tyto schopnosti.

3 Práce s kontakty na platformě Android

Na platformě Android jsou v *databázi kontaktů* (tedy v databázi kde jsou uloženy data kontaktů) uloženy všechny informace nejen o všech *místních kontaktech* (kontakty v telefonu, nesouvisející s jinou službou), ale i údaje z profilů na sociálních sítích (Facebook) nebo účtech různých služeb (Google). Tyto kontakty jsou využívány kontaktní a *vytáčecí aplikací* (dialer application) telefonu, ale také dalšími aplikacemi jako jsou emailový klient, programy sociálních sítí, kalendář, atd. Pomocí kontaktů je v telefonu uložen dokonce i profil uživatele telefonu.

Kontakty jsou na platformě Android uložené v centrálním úložišti dat[9], tedy databázi ve které jsou uložena systémová data telefonu. Aplikace k tomuto úložišti nemohou přistupovat přímo, díky čemuž je úložiště zabezpečené proti mazání či zneužití dat[14]. K přístupu k těmto datům se používají objekty typu `ContentResolver` na straně klientské aplikace a objekty typu `ContentProvider` na straně úložiště dat.[15, 14]

Objektem typu *content resolver* (dále jen content resolver) nazýváme instanci třídy odvozené od abstraktní třídy `ContentResolver`. Obdobně objektem typu *content provider* (dále jen content provider) nazýváme instanci třídy odvozené od abstraktní třídy `ContentProvider`.

Content resolver je automaticky (prostřednictvím metody `getContentResolver` v instanci *aktivity*) obsažen v každé aplikaci[1]. Pro práci s daty je však třeba důkladně prozkoumat příslušný content provider.

3.1 Content Provider

Content provider slouží k přístupu ke strukturovaným datům v centrálním úložišti dat. Zapouzdřuje přístup k datům, k nimž nepřistupujeme přímo, ale jen skrz content provider. Nejčastěji se využívá při meziprocesové komunikaci, kdy na straně úložiště dat stojí objekt content provider, na druhé straně pak objekt content resolver klientské aplikace. Takový způsob komunikace zajišťuje konzistentní a bezpečný přístup k datům.

Content provider obdrží od content resolveru klientské aplikace požadavky, provede požadovanou akci s daty, a navrátí klientovi výsledky.

Content provider zpřístupňuje data aplikací v podobě tabulek, tak jak jsme

na to zvyklí například z relačních databází. Jeden řádek tak reprezentuje instanci objektu, který databáze uchovává, sloupec pak datové pole obsahující jeden typ dat.

3.1.1 Přístup k provideru

Jak bylo napsáno výše, aplikace přistupuje k provideru skrze content resolver. Tento objekt má metody, které volají ekvivalentní (i stejně pojmenované) metody provideru[14]. Content resolver nabízí klasické "CRUD"metody (create, retrieve, update, and delete)[14].

Content resolver v klientské aplikaci a content provider v aplikaci vlastnící přístup k datům automaticky zvládají meziprocesovou komunikaci. Content provider navíc zajišťuje vrstvu abstrakce pro klientskou aplikaci, kvůli níž se data jeví jako množina tabulek v relační databázi[14].

Typický přístup vypadá tak jak můžeme vidět v kódu 3.1:

```
resolver.query(uri, projection, selection, selectionArgs,
   sortOrder)
```

Kód 3.1: Ukázka metody `query` ze třídy `contentResolver`[14].

kde:

- **uri** určuje jméno (cestu) tabulky o kterou máme zájem - podobně jako SQL FROM
- **projection** určuje sloupce které chceme vrátit - podobně jako SQL col, col,col
- **selection** a sortOrder určují o které řádky tabulky máme zájem - podobně jako SQL WHERE
- **sortOrder** určuje seřazení vrácených řádků tabulky - SQL ORDER BY

3.1.2 Content URI

Content URI je cesta která identifikuje data v provideru. Content URI se skládá ze symbolického jména provideru (authority) a z názvu tabulky (path).

ContentResolver nejprve porovná jméno providera v URI se systémovou tabulkou providerů, které jsou k dispozici. Následně pošle zbylé argumenty včetně druhé části URI vybranému provideru. ContentProvider pak využije druhou část URI - jméno tabulky k vybrání správné tabulky.

3.1.3 Získání dat z provideru

K získání dat z provideru potřebujeme splnit dva předpoklady:

1. zajistit si právo pro přístup
2. sestavit správný dotaz

Získání přístupových práv

Pro práci s daty pomocí provideru je třeba, aby klientská aplikace měla příslušná oprávnění. Tato oprávnění nemohou být získána za běhu aplikace, místo toho je třeba si je vyžádat v souboru `AndroidManifest.xml` patřícímu ke klientské aplikaci. K tomu se používá element `<uses-permission>`. Přesné jméno povolení závisí na tom, jaký provider používáme a jaká práva požadujeme. Tato práva jsou pak požadována při instalaci aplikace, jejím schválením jsou pak potvrzena.

Vytvoření dotazu

Dotaz vytvoříme příkazem z úvodu kapitoly (Kód 3.1). Výraz, který určuje jaké řádky vrátit, je rozdělen do dvou argumentů - `selection` a `selectionArgs`. Argument `selection` je kombinací logických výrazů. názvů sloupců a hodnot. Hodnoty lze nahradit znakem `?` v tomto případě jsou pak doplněny z pole hodnot `SelectionArgs`.

Toto rozdělení, ačkoliv není vyžadováno, zvyšuje bezpečnost, neboť zamezuje použití injektáže příkazu (vložení příkazů do proměnných, které mají doplnit data).

3.1.4 Zobrazení výsledků dotazu

Metoda `ContentResolver.query()` vrací vždy `Cursor` (objekt implementující rozhraní Cursor, dále jen kurzor) obsahující údaje z tabulky určené `uri`, přesněji ty sloupce, které jsou specifikované argumentem `projection` v řádcích

které jsou určené argumenty `selection` a `selectionArgs`. Kurzor umožňuje čtení dat, které obsahuje. Pomocí kurzoru pak můžeme iterovat skrze navrácená data po řádcích. Některé kurzory dokonce reflektují změny ve zdrojových datech, či na ně alespoň upozorňují.

Pro zobrazení dat z kurzoru můžeme:

1. zobrazit data pomocí adaptéru (třída implementující rozhraní `Adapter`), do kterého předáme kurzor jako parametr
2. iterovat kurzorem přes metodu `moveToNext()`, data získáme pomocí `getString(index)` a obdobných metod (`getInt(index)`...)

3.1.5 Vkládání, aktualizace a mazání dat

Používají se opět metody content resolveru, konkrétně:

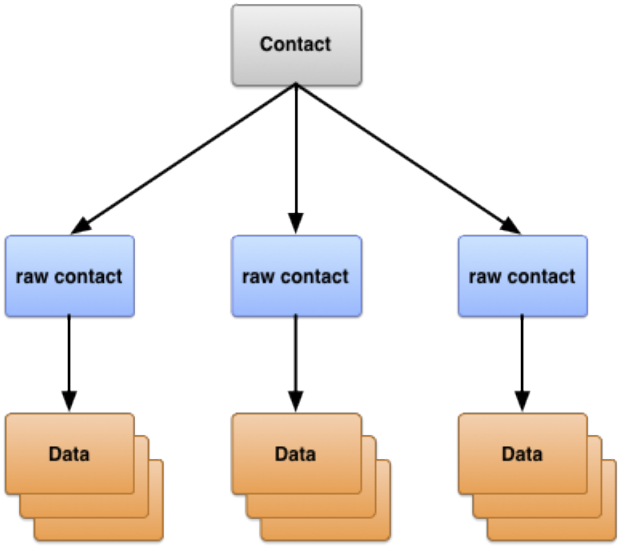
- ContentResolver.insert()
- ContentResolver.update()
- ContentResolver.delete()

3.1.6 Kontrakty

Kontraktní třída je třída, která definuje konstanty, které pomáhají klientským aplikacím pracovat s URI, jmény sloupců, a dalšími funkcemi content provideru. Kontraktní třídy neexistují ke content provideru automaticky, vývojář příslušného provideru je musí implementovat. Mnoho providerů z platformy Android má svoje kontaktní třídy v balíčku android.provider.

3.2 Contact Provider

Content provider pro práci s kontakty je třída `ContactProvider` (dále jen contact provider). Platforma Android umožňuje ukládat do zařízení mnoho různých kontaktních informací, navíc z více zdrojů. Tomu odpovídá poměrně složitý návrh organizace těchto dat, a velmi široká paleta tříd a rozhraní pro přístup k datům.



Obrázek 3.1: Struktura kontaktů na platformě Android[9]

3.2.1 Organizace kontaktů

Jak můžeme vidět na obrázku 3.1, existují tři typy dat o kontaktech[9]:

1. **Kontakty:** řádky reprezentují různé osoby na základě spojení řádků z tabulky `RawContacts`. Toto jsou kontakty, které se zobrazují uživateli v seznamu kontaktů v telefonu.
2. **Raw kontakty:** řádky obsahující souhrn osobních dat kontaktu vzešlých z jednoho účtu (místní kontakt, google, facebook...).
3. **Data:** řádky reprezentují jeden kontaktní údaj (email, telefoní číslo..).

Ke každému z těchto typů tabulek přistupujeme pomocí jejich kontraktních tříd. Tyto třídy definují konstanty pro jména tabulek, sloupců a další. Obalovací třída pro všechny kontraktní třídy pro práci s kontakty je třída `ContactContract`.

Kromě `ContactsContract.Contacts`, `ContactsContract.Data` a `ContactsContract.RawContacts` existují ještě další kontraktní třídy. Jsou to například pomocné tabulky, které contact provider používá k řízení svých činností nebo k podpoře specifických funkcí telefonních aplikací.

3.2.2 Raw contacty

Raw kontakt obsahuje všechna data vztahující se k jedné osobě na jednom účtu (účet je např user@gmail.com, nebo místní kontakt). Toto tvrzení nemusí být přesné pokud má uživatel na účtu duplicitní kontakty, pak je na jednom účtu více raw kontaktů pro jednu osobu. Protože contact provider umožňuje používat více účtů jako zdrojů kontaktních dat, můžeme mít více raw kontaktů vztahujících se k jedné fyzické osobě. Také je možné kombinovat data z více účtů stejného typu (user1@gmail.com., user2@gmail.com), údaj z každého takového účtu má také svůj raw kontakt, i když jde o stejnou osobu.

Většina osobních dat však není uložena v tabulce raw kontaktů. Nachází se totiž v tabulce dat. Každý řádek z této tabulky pak obsahuje sloupec `Data.RAW_CONTACT_ID`, ve kterém je uloženo `RawContacts._ID`, které ukazuje na příslušný raw kontakt - tedy na řádek v tabulce raw kontaktů.

Důležité sloupce tabulky rawContacts:

- `ACCOUNT_NAME` - Jméno účtu ze kterého pocházejí data pro raw kontakt. Například emailová adresa pro účet na googlu.
- `ACCOUNT_TYPE` - Typ účtu, ze kterého pochází data pro raw kontakt. Například com.google pro google účet.
- `DELETED` - Značka určující, zdali je kontakt platný. Slouží k synchronizaci s online službami (aby byl kontakt smazán i z cloudu).

Výčet všech sloupců tabulky `ContactsContract.RawContacts` dostupných přes contact provider je uveden v příloze D.

3.2.3 Data

V tabulce `ContactsContract.Data` jsou uložená v řádcích jednotlivá data, která jsou s raw kontaktem spojena pomocí sloupce `_ID`. Takové uspořádání umožňuje, aby měl jeden raw kontakt více dat stejného typu, tedy např. více telefonních čísel.

V tabulce jsou uloženy všechny typy dat, které by mohly ke kontaktu patřit, jako zobrazované jméno, telefonní číslo, email, adresa nebo fotografie. Aby neměla tabulka `ContactsContract.Data` příliš mnoho sloupců pro potřeby různých druhů dat, má jen nějaké sloupce pro určený typ záznamu, a zbylé s generickým jménem, tedy pro libovolná data.

Sloupce s určeným obsahem:

- `RAW_CONTACT_ID`: Identifikátor, který spojuje data s raw kontaktem.
- `MIMETYPE`: Typ dat v řádku. Používá se MIME type, typy jsou definované v kontraktní třídě `ContactsContract.CommonDataKinds`.
- `IS_PRIMARY`: Značí, jestli jsou data v rámci raw kontaktu primární. Má to význam v případě, že jeden raw kontakt ukazuje na více instancí stejného typu dat (více telefonních čísel, mailů...).

Dále je v tabulce 15 sloupců pro generická data pojmenovaných `DATA1` až `DATA15` a další 4 generické sloupce `SYNC1` až `SYNC4`, které by měly být použity pouze adaptéry pro synchronizaci. K těmto sloupcům lze vždy přistupovat pomocí generického jména, bez ohledu na to, který datový typ obsahují. Stejně tak lze ale k polím přistupovat pomocí konstant z kontraktních tříd konkrétních typů dat.

Sloupec `DATA1` je indexován. Contacts Provider používá tento sloupec pro data, která bývají požadována nejčastěji (pokud se tedy jedná o email, obsahuje emailovou adresu).

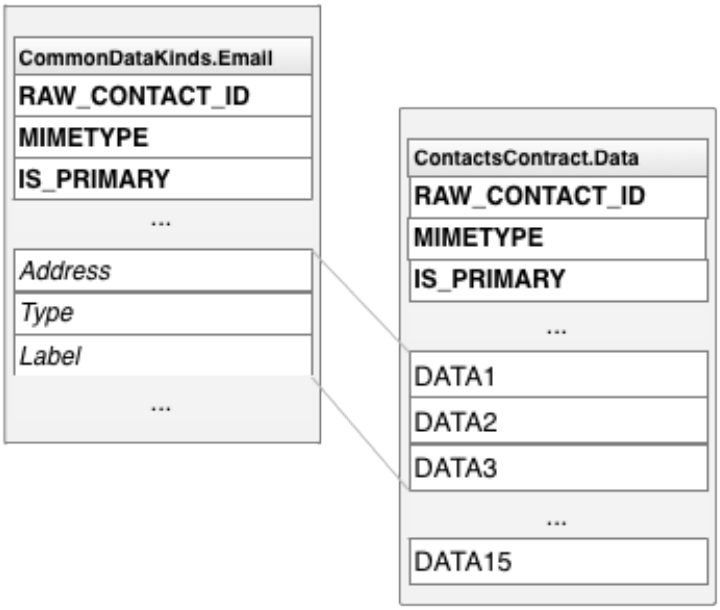
Podle konvence je sloupec `DATA15` rezervován pro Binary Large Object (BLOB), což jsou například náhledy fotek. Výčet všech sloupců tabulky `ContactsContract.Data` dostupných přes contact provider je uveden v příloze E.

Jak bylo již zmíněno výše, jména generovaných sloupců dle typu dat lze najít v příslušných kontraktních třídách. Tyto kontraktní třídy jsou definovány ve třídě `ContactsContract.CommonDataKinds`. Konstanty v kontaktních třídách pomáhají při identifikaci sloupců v kódu. Jak to funguje, přiblíží obrázek 3.2.

Napravo je vidět jeden řádek tabulky `Data`, tak jak je uložen v databázi, nalevo je pak jeho interpretace pomocí konstant ze třídy `ContactsContract.CommonDataKinds.Email`

Častými datovými typy jsou:

- `ContactsContract.CommonDataKinds.StructuredName`
- `ContactsContract.CommonDataKinds.Phone`
- `ContactsContract.CommonDataKinds.Email`
- `ContactsContract.CommonDataKinds.Photo`
- `ContactsContract.CommonDataKinds.GroupMembership`



Obrázek 3.2: Struktura jednoho řádku v tabulce data[9]

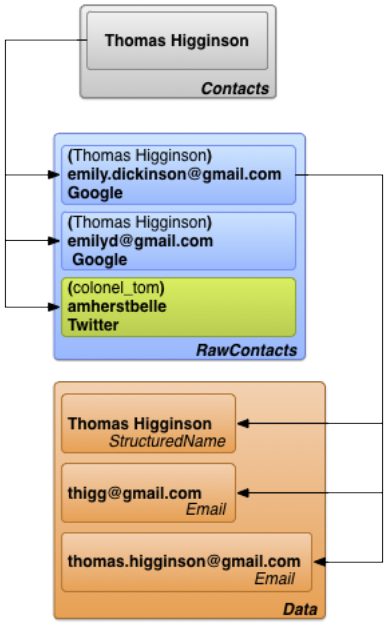
3.2.4 Kontakty

Kontakt kombinuje všechny raw contacty všech typů a jmen účtů. Tím zobrazí všechna uložená data o jedné fyzické osobě. Contacts provider se stará o vytvoření nových řádek tabulky kontaktů, a spojování raw kontaktů pod jedním řádkem v tabulce `contact`. Aplikace ani adaptéry synchronizace nemohou přidávat kontakty do tabulky contact, navíc některé sloupce tabulky jsou jen ke čtení[9] .

Contacts Provider vytváří nový kontakt v případě, že byl vytvořen nový raw kontakt, který nepatří k žádnému z existujících kontaktů. Provider vytvoří nový kontakt také tehdy, pokud se raw kontakt změní takovým způsobem, že už nemůže být přiřazen k současnému kontaktu. Pokud aplikace nebo adaptér pro synchronizaci vytvoří nový raw kontakt, který lze přiřadit k existujícímu kontaktu, nový kontakt se nevytváří.

Contacts provider připojuje kontakt k raw kontaktu pomocí sloupce `_ID` v tabulce `ContactsContract.Contacts`, sloupec `CONTACT_ID` v tabulce raw kontaktů `ContactsContract.RawContacts` obsahuje hodnoty _ID, které ukazují na příslušný řádek tabulky kontaktů.

Tabulka kontaktů má také sloupec `LOOKUP_KEY`, který je permanentním odkazem na jeden záznam v tabulce kontaktů. To je nutné z toho důvodu, že contact provider může změnit ID kontaktu na základě slučování raw kontaktů nebo dalších



Obrázek 3.3: Vzájemný vztah kontaktů raw kontaktů a dat[9]

změn automaticky. I když se to stane, content `URI CONTENT_LOOKUP_URI` v kombinaci s `LOOKUP_KEY` kontaktu bude stále odkazovat na tentýž kontakt. Tento sloupec má formátování nezávislé na sloupci `_ID`. Výčet dalších sloupců tabulky `ContactsContract.Contacts` dostupných přes contact provider je uveden v příloze C.

Na obrázku 3.3 vidíme opět vztah všech tří tabulek. V telefonu (patřícímu Emily Dickinsonové) se nachází kontakt Thomas Higginson. Tento kontakt v sobě slučuje tři raw kontakty, které se všechny vztahují k osobě Thomase Higginsona. Dva pochází z Emilyných účtů na gmailu, jeden z twitteru. Každý z těchto raw kontaktů pak ukazuje na několik řádků v tabulce data (poslední blok dole). Jeden ze tří raw kontaktů tak obsahuje hned tři emaily (na tutéž osobu).

3.2.5 Oprávnění potřebná pro práci s kontakty

Aplikace, které chtějí používat contacts provider, musí požádat při instalaci o následující práva:

- pro čtení kontaktů: `android.permission.READ_CONTACTS`
- pro zápis do tabulky kontaktů: `android.permission.WRITE_CONTACTS`

Tato práva je třeba uvést do souboru `AndroidManifest.xml` (do kořenového elementu)

3.2.6 Propojování kontaktů

Propojením kontaktů na platformě Android rozumíme proces, kdy spojíme dva či více raw kontaktů pod jeden kontakt. Systém Android takto spojuje nebo nabízí ke spojení kontakty, které považuje za shodné - patřící ke stejné osobě. Kontakty slučuje, pokud platí jedna z následujících podmínek[22]:

1. Souhlasí jméno i příjmení.
2. Souhlasí jméno i příjmení, liší se pořadí jména a příjmení.
3. Příjmení souhlasí, křestní jméno je zkrácené (je však důvod se domnívat že tato funkcionalita je omezená jen pro angličtinu, nebo několik hlavních světových jazyků, pro češtinu nefunguje).
4. Souhlasí příjmení nebo jméno, a zároveň má jeden z kontaktů jen jméno nebo příjmení a zároveň souhlasí telefonní číslo nebo mail.
5. Jeden z kontaktů nemá jméno ani příjmení a souhlasí mail nebo telefonní číslo.

Příslušná pole která je třeba změnit, abychom kontakty propojili z vlastní aplikace, nejsou bohužel přes content provider dostupná. Jediným řešením, jak kontakty manuálně spojit je tabulka `AggregationExceptions`.

Aggregation Exceptions

Aggregation exceptions je tabulka obsahující identifikátory dvojic raw kontaktů, jejichž propojování nemá probíhat automaticky. Ke každé dvojici je ve třetím sloupci tabulky přiřazena jedna ze tří hodnot[11]:

- `TYPE_AUTOMATIC`: Propojování probíhá automaticky, řídí ho contact provider.
- `TYPE_KEEP_SEPARATE`: Kontakty se nepropojí, přesto že by contact provider nalezl shodu.

- `TYPE_KEEP_TOGETHER`: Kontakty se ihned spojí a zůstanou propojené, dokud je uživatel nerozpojí (a tento záznam se tak smaže).

Kontraktní třídou pro tuto tabulku je třída `AggregationException`. Sloučení kontaktů v Aplikaci se pak provede například tak jak to vidíme v kódu 3.2:

```
ContentValues cv = new ContentValues();
cv.put(AggregationExceptions.TYPE,
   AggregationExceptions.TYPE_KEEP_TOGETHER);
cv.put(AggregationExceptions.RAW_CONTACT_ID1,
   rawKontaktIdKontaku1);
cv.put(AggregationExceptions.RAW_CONTACT_ID2,
   rawKontakIdKontaku2);
getContentResolver().update(AggregationExceptions.CONTENT_URI,
   cv, null, null);
```

Kód 3.2: Příklad kódu pro sloučení dvou kontaktů [26].

3.3 Nedostatky v oficiální dokumentaci

Přestože je dokumentace platformy Android velice obsáhlá a přehledná, některé podstatné aspekty práce s kontakty tam popsány nejsou. Právě těmto nezdokumentovaným tématům bude věnována následující část bakalářské práce.

3.3.1 Práce s kontakty z Facebooku

Pokud při dotazování na kontakty přes contact provider neuvedeme parametr `selection` (respektive zavoláme metodu `ContentResolver.query` s parametrem `selection = null`), měl by nám contact provider vrátit kurzor se všemi kontakty. Ve skutečnosti však vrátí všechny kontakty s výjimkou kontaktů pocházejících z Facebooku[20, 17].

Vlastním průzkumem v databázi kontaktů jsem dospěl k závěru, že je za to zodpovědný sloupec `is_restricted`, který je u kontaktů z Facebooku nastaven na hodnotu jedna. Tento sloupec je skrytý, contact provider jeho obsah neposkytuje, uživatel ani programátor se tak o existenci takového sloupce vůbec nedozví, dokud se nepodívá přímo do databáze kontaktů (která je na běžném zařízení nepřístupná).

Důvod tohoto omezení ani jeho existence nejsou oficiálně známi, z různých zdrojů[3, 16] lze však dovodit, že se možná jedná o neveřejnou dohodu mezi společností Google a Facebook.

S kontakty z Facebooku tak nelze na běžném zařízení s operačním systémem Android vůbec pracovat. Jedinou možností je kontakty nejprve odemknout v databázi. Takové řešení je však dostupné pouze na zařízení, kde má uživatel administrátorská práva (standardně prodávaná zařízení nemají administrátorská práva, pro jejich získání je třeba nakonfigurovat zařízení pomocí PC).

Zajímavostí dále může být to, že defaultní kontaktní aplikace telefonu přístup ke kontaktům Facebooku mají (pokud uživatel nastavil synchronizaci se službou Facebook). Ze zdrojového kódu je však patrné, že dotaz prostřednictvím content resolveru je zcela standardní. Rozpoznávací mechanismus tak bude přímo v provideru, což je logické i z hlediska bezpečnosti, neboť takový systém nelze obejít.

3.3.2 Změna názvu kontaktu

Ze specifikace kontraktní třídy pro tabulku kontaktů je zřejmé, že většina polí v této tabulce je pouze ke čtení, měnit je nelze[12]. To je překážkou v případě že chceme u propojených kontaktů určit zobrazovanou fotku a název (tedy chtěli bychom změnit pole `photo_id` a `neme_raw_contact_id`).

Z dokumentace kontraktní třídy pro tabulku `data` ale vyplývá, že by mělo být možné primární kontakt pro fotku stejně jako pro název *zastínit* pomocí sloupce `is_super_primary`[13]. Tento postup funguje pro kontaktní fotku, pro název je ale hodnota sloupce ignorována. Nelze tak ani tímto způsobem určit, ze kterého z kontaktů se vybere zobrazovací jméno pro propojený kontakt. Tento problém už je navrhnut k řešení nejméně od roku 2010[7], stále ale není vyřešen.

4 Struktura aplikace

Vyhotovená aplikace důsledně používá třívrstvou architekturu. Byla vybrána z důvodu snadné orientace ve zdrojovém kódu a snazší nahraditelnosti jednotlivých částí. Dále toto rozdělení značně usnadňuje platforma Android, která má vytvořené dobré zázemí pro prezentační vrstvu aplikací. Zjednodušené rozdělení do tří vrstev můžeme vidět na obrázku 4.1.

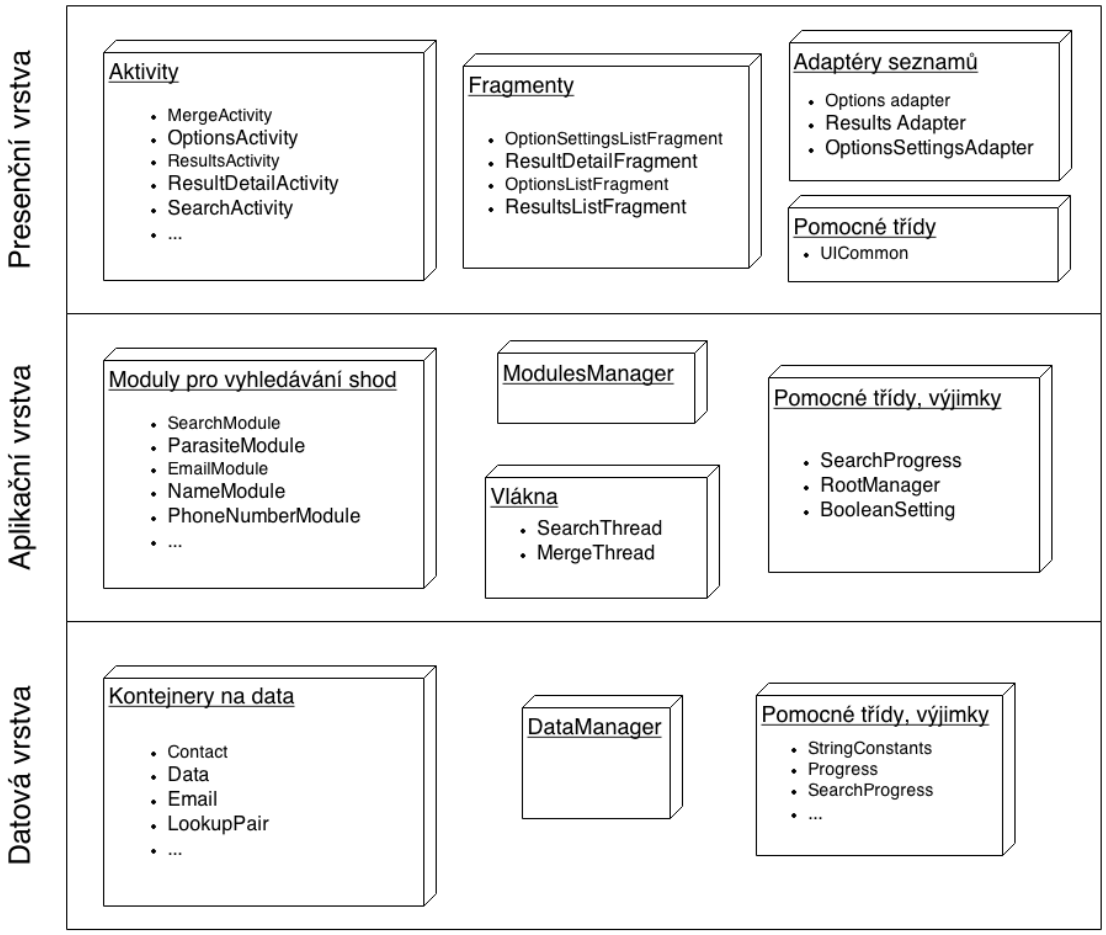
Aplikace byla nazvána prostým anglickým překladem Contact Merge. Hlavní ikonu aplikace vytvořila Veronika Česalová (Chelsea College of Art - Londýn). Defeultní ikona kontaktu pochází ze stránek www.iconfinder.com[10], kde je k dispozici volnému použití.

4.1 Prezentační vrstva

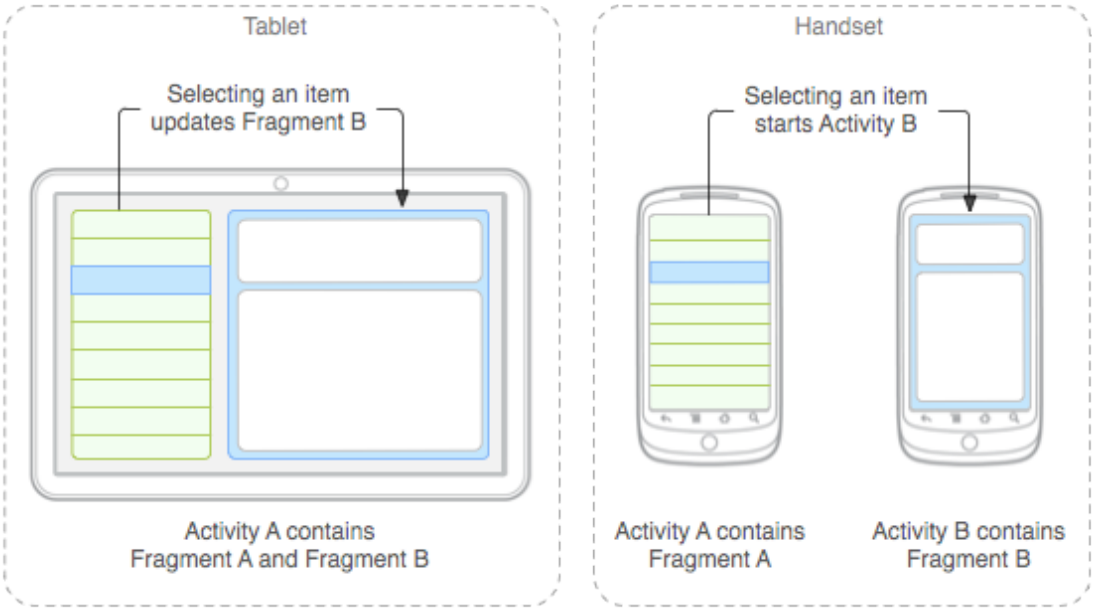
Prezentační vrstva je, jak je tomu na platformě Android obvyklé, tvořena třídami odvozenými od třídy `android.app.Activity` (dále jen aktivita). Každá aktivita zastupuje jeden pohled na aplikaci. Mezi těmito pohledy se pohybujeme pomocí tlačítek uživatelského rozhraní (dále UI). Aplikace Contact Merge obsahuje šest aktivit. Každá bude krátce probrána níže. Celé schéma presenční vrstvy z pohledu uživatele se nachází na obrázku B.1 v přílohách.

4.1.1 Fragmenty

V uživatelském rozhraní je na mnoha místech použito takzvaných fragmentů. Tyto třídy dostupné od API 11[18] značně zjednodušují možnost v budoucnu optimalizovat aplikaci pro zařízení s většími displayi, například tablety[5]. Dva fragmenty, které jsou na telefonu ve dvou aktivitách můžeme dát na tabletu do jedné aktivity vedle sebe, jak je vidět na obrázku 4.2. Takové použití je možné například u tříd `ResultListFragment` a `ResultDetailFragment`. Třída `ResultListFragment` dědí od speciální třídy `ListFragment`, která je protějškem třídy `ListActivity` pro fragmenty[5]. Zatím rozhraní aplikace pro více velikostí obrazovky optimalizováno není, cesta pro to, aby to tak v budoucnu bylo, je ale díky fragmentům připravená.



Obrázek 4.1: Struktura aplikace Contact Merge.



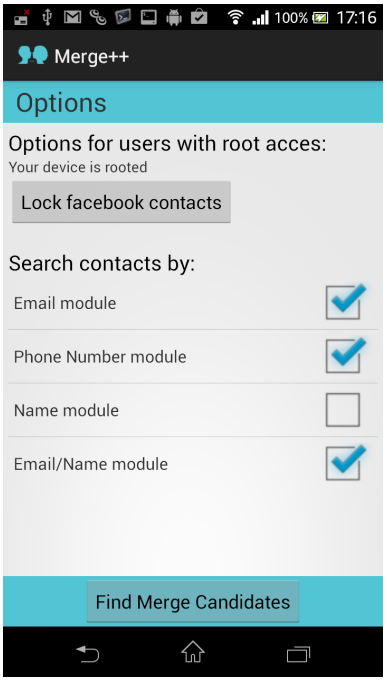
Obrázek 4.2: Výhoda použití fragmentů[19]

4.1.2 Vynechání aktivit z historie

Běžné aplikace na platformě android mají hierarchickou strukturu. Tlačítko zpět pak v takové aplikaci vede k nadřazené aktivitě, případně zavírá celou aplikaci. V aplikaci Contact Merge je však přítomna i lineární struktura, od úvodní aktivity přes aktivitu hledání až k aktivitě nalezených výsledků (viz. obr B.1 v přílohách). Při navigaci pomocí tlačítka zpět se v takovém případě někdy nechceme dostat na předchozí aktivitu. Například dostat se z aktivity nalezených výsledků k aktivitě hledání nedává žádný smysl. Tento problém se nejlépe řeší deklarativní cestou v souboru `AndroidManifest.xml`, jak je vidět v kódu 4.1 (atribut `noHistory`).

```
<activity
   android:name="fido.contacts.merge.ui.SearchActivity"
   android:theme="@android:style/Theme.Holo.Dialog"
   android:noHistory="true" >
</activity>
```

Kód 4.1: Nastavení aktivity tak, aby byla ignorována při navigaci pomocí tlačítka zpět.



Obrázek 4.3: Úvodní aktivita aplikace Contact Merge.

4.1.3 Stručný popis aktivit

Úvodní aktivita

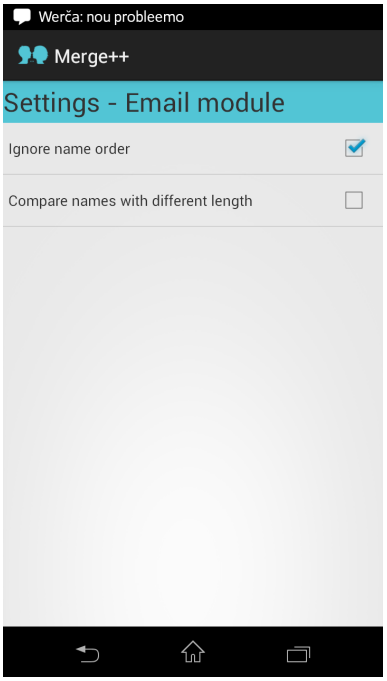
Úvodní aktivitu můžeme vidět na obrázku 4.3.

Zde má uživatel možnost vybrat, jaké způsoby hledání chce použít. Způsoby jsou rozděleny do modulů, z nichž každý bude porovnávat jiný typ dat, nebo stejný typ, ale zásadně odlišným způsobem. Více zaškrtnutých modulů znamená více času pro běh programu, ale také může znamenat lepší výsledky. Každý modul se dá konfigurovat v nastavení modulu, do kterého se dostaneme po klepnutí na název modulu.

Moduly se načítají do seznamu z aplikační vrstvy, můžeme tedy přidat další aniž bychom zasahovali do UI.

Tlačítkem v dolní části obrazovky započne hledání dle námi navolené konfigurace. Tlačítkem unlock facebook contacts pak můžeme na telefontech s administrátorskými právy odemknout kontakty ze sítě Facebook (nutnost odemčení byla popsána v teoretické části v kapitole 3.3.1, realizace dále v kapitole 4.3.3).

V této aktivitě se při spuštění (v metodě `onCreate`) inicializuje `ModuleManager` a `RootManager` (jejich funkce popsána dále). Při startu ak-



Obrázek 4.4: Aktivita nastavení.

tivity je třeba zjistit, zdali už oba managery neexistují, při opakované inicializaci by došlo k výjimce. Takové situace může nastat například pokud uživatel otočí telefon a operační systém se pokusí zobrazit aktivitu „na šířku“ [2].

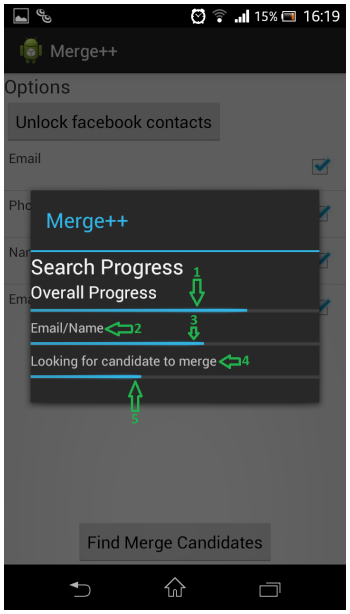
Aktivita nastavení modulu

Tato aktivita slouží pro konfiguraci modulu. Modul by měl být defaultně nakonfigurován tak, aby našel co nejvíce výsledků, ale některé způsoby hledání, které jsou příliš náročné na čas nebo přinášejí falešné kandidáty na spojení (program navrhne spojit kontakty, které nepatří k téže osobě), mohou být implicitně vypnuty.

Na obrázku 4.4 vidíme pouze jednu možnost s checkboxem, ale aplikace je navržena tak, aby přidání dalších voleb bylo snadné. Všechna nastavení jsou totiž uložena v modulech (v aplikační vrstvě). Můžeme tak podobně jako u modulů přidat další, aniž bychom zasahovali do prezentační vrstvy.

Aktivita hledání kontaktů ke sloučení

Tato aktivita zobrazuje po dobu svého běhu dialog s informacemi o průběhu hledání. Proces hledání běží v samostatném vlákně, tedy nedochází k zamrznutí UI



Obrázek 4.5: Aktivita slučování kontaktů: 1) indikátor celkového postupu; 2) stavový řádek postupu - zobrazuje jméno běžícího modulu; 3) indikátor postupu modulu; 4) stavový řádek modulu - zobrazuje činnost kterou modul vykonává; 5) indikátor postupu právě vykonávané činnosti.

v době výpočtu. Pro multithreading je použitá metoda/třída asyncTask[4]. Realizace byla inspirována návodem na serveru codingforandroid.com.[8]. Uživatelské rozhraní aktivity vidíme na obrázku 4.5.

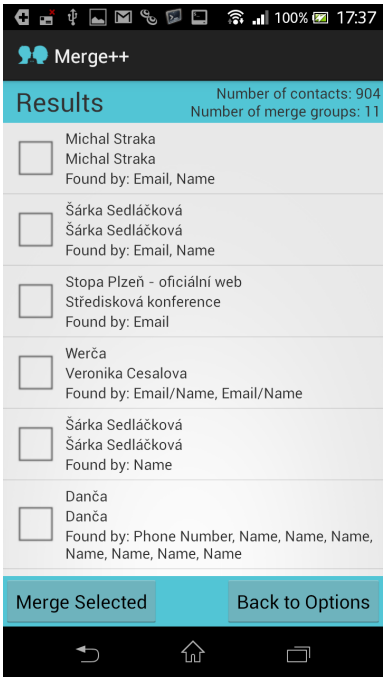
Aktivita výsledků hledání

V této aktivitě se zobrazují výsledky hledání. Celou aktivitu vidíme na obrázku 4.6. Checkbox vlevo určuje, zdali chceme, aby po stisku tlačítka v dolní části došlo ke spojení kontaktů.

V pravé části jsou pod sebou jména obou kontaktů určených ke spojení. Pokud by nám jména k identifikaci nestačila, dostaneme se po kliknutí k detailům obou kontaktů.

Aktivita detail kontaktu

V této aktivitě se zobrazuje detail jednoho kontaktu. Do aktivity se dostaneme po klepnutí na řádek seznamu výsledků. Celou aktivitu vidíme na obrázku 4.7. Po použití tlačítka *merge contacts* se dostaneme na aktivitu individuálního propojení



Obrázek 4.6: Aktivita výsledků hledání.

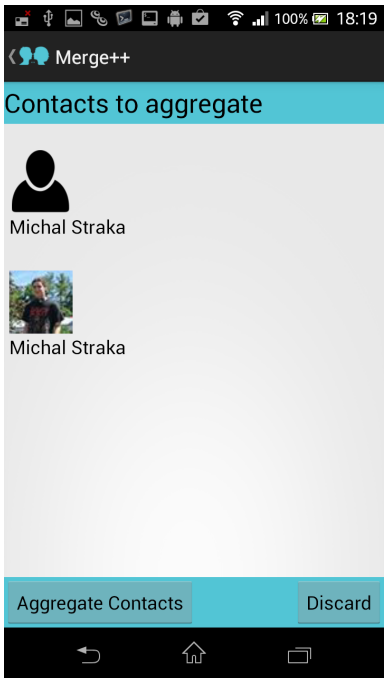
kontaktů, tlačítko *discard* nás vrátí zpět do seznamu kandidátů k propojení, z kterého bude odstraněna položka v jejímž detailu jsme se nacházeli.

Aktivita individuálního propojení kontaktů

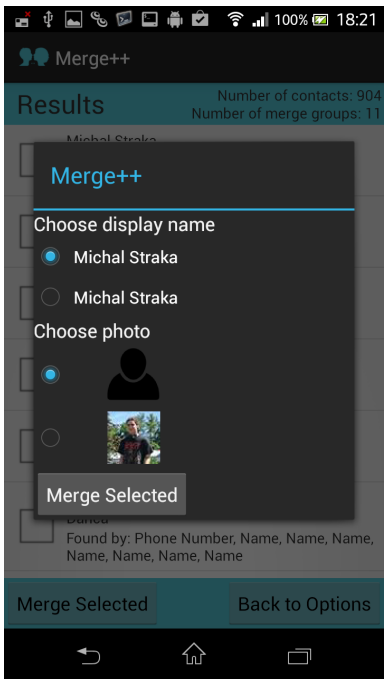
V této aktivitě dochází ke sloučení jedné dvojice kontaktů. Nejprve vybereme jméno a fotku kontaktu, pak klikneme na tlačítko *ok*. Vlastní slučování kontaktů probíhá v samostatném vlákně, nedochází proto ke zpomalení uživatelského rozhraní. Celou aktivitu vidíme na obrázku 4.8.

Aktivita slučování více kontaktů

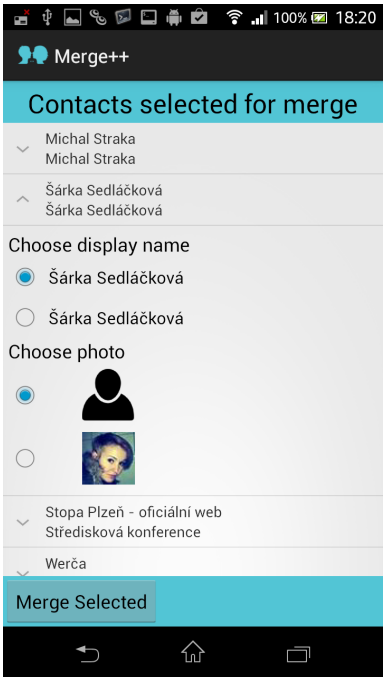
Do této aktivity se dostaneme po klepnutí na tlačítko *merge selected* v seznamu nalezených výsledků. Zobrazují se zde všichni kandidáti k propojení, které jsme zaškrtli v aktivitě výsledků. Po kliknutí na řádek v seznamu dojde k rozbalení detailů dvojice kontaktů. Zde máme možnost zvolit název kontaktu a fotku. Po kliknutí na tlačítko start merging dojde ke sloučení všech kontaktů v seznamu dle našeho nastavení. Celá operace probíhá ve zvláštním vlákně, nedochází tak ke zpomalování UI. Celou aktivitu vidíme na obrázku 4.9. Pro tvorbu rozbalovacího seznamu byl použit návod ze stránky androidhive.info[27].



Obrázek 4.7: Aktivita detail kontaktu.



Obrázek 4.8: Aktivita individuálního propojení kontaktů.



Obrázek 4.9: Aktivita propojování více kontaktů najednou.

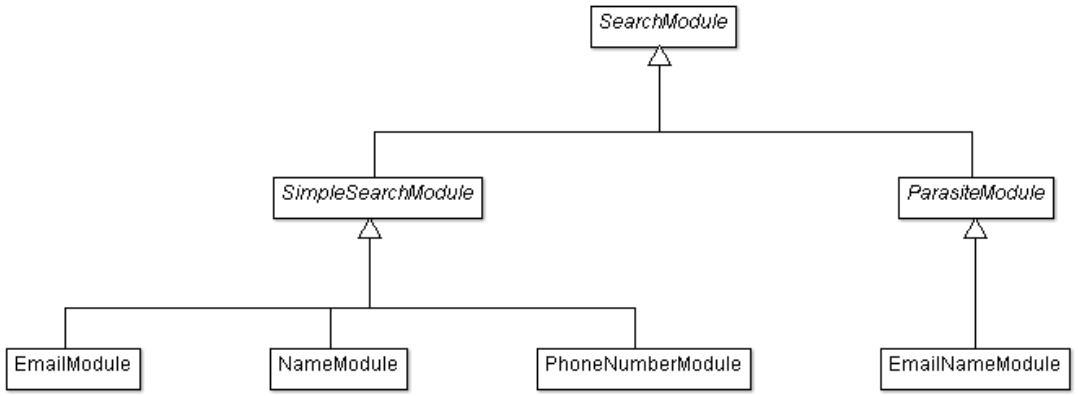
V obou aktivitách je použitý pro výběr názvu a jména prvek obecně známý jako *radio button*. Umístit do tohoto prvku místo textu obrázek, jak bylo potřeba u výběru fotografie se ukázalo jako náročný úkol. Nakonec byl problém vyřešen pomocí nastavení vlastnosti `drawableRight` třídy `radioButton`[28]. Dalším problémem byla skutečnost, že obrázky kontaktů jsou v zařízení uloženy ve velmi malých rozměrech. K jejich zvětšení byla použita metoda nalezená na stránce StackOverflov[25].

4.2 Aplikační vrstva

Aplikační vrstva se skládá ze třídy `ModulManager`, která řídí celý proces hledání kontaktů, a jednotlivých modulů, které hledání provádějí. Dále jsou zde vlákna řešící asynchronní procesy hledání a slučování kontaktů.

Manažer modulů je zodpovědný za to, v jakém pořadí jsou moduly volány, a také například za to, že běží jen ty moduly, které uživatel zaškrtl v úvodní aktivitě. Třída je implementována jako singleton, aby nebylo třeba předávat odkaz na instanci napříč aplikací.

Jednotlivé moduly pak provádějí hledání, při kterém čerpají údaje z datové vrstvy, a následně do ní ukládají nalezené kandidáty na spojení.



Obrázek 4.10: Moduly hledání - Diagram tříd

4.2.1 Systém vyhledávacích modulů

Systém vyhledávacích modulů byl řešen s důrazem na snadnou rozšiřitelnost. Maximum možného kódu se nachází v nadřazených *abstraktních třídách* - pro vytvoření nového tak není třeba mnoho času. Strukturu modulů můžeme vidět na obrázku 4.10.

Základní třídou je třída `SearchModule`. Tato třída definuje základní vlastnosti které musí mít každý vyhledávací modul, stejně tak nabízí univerzální řešení tam, kde je jasné, že ani v budoucnu nebude třeba řešit věci jinak. Třída `SearchModule` tak funguje nejen jako mateřská třída nabízející zázemí pro svoje potomky, ale také jako rozhraní mezi moduly a třídou `ModulesManager` (správce modulů). Každý budoucí modul musí tuto třídu oddědit, buď přímo, nebo prostřednictvím jiné třídy.

Třída `SearchModule` řeší zejména:

- Zdali je modul zapnut nebo vypnut - dle nastavení uživatel v UI.
- Nastavení modulů - taktéž dle nastavení uživatel v UI.
- Pomocí abstraktních metod určuje základní metody potřebné pro každý modul, od inicializace po určení kontaktu k propojení.
- Zde se nachází metoda `run()`, kterou volá `ModulesManager` pokud je modul zapnutý, tato metoda je tak branou mezi správcem modulů a modulem.

Další důležitou třídou je třída `SimpleSearchModule`. Tato třída oddděná

od třídy `SearchModule` přináší zázemí pro jednoduché moduly. Struktura běhu takovýchto jednoduchých modulů je:

1. Načíst potřebná data z databáze.
2. Uložit data do datových struktur vhodných pro efektivní porovnávání.
3. Porovnat všechna data a uložit nalezené kandidáty k propojení do seznamu.

Běžný modul pro hledání kontaktů k propojení pak třídu `SimpleSearchModule` zdědí. Programátor nového modulu tak musí doplnit pouze:

1. enum `SearchSettings` seznam možných nastavení modulu.
2. metodu `getDisplayNameId()` která vrací id řetězce který obsahuje jméno modulu. (řetězce jsou deklarovány v XML a odkazovány pomocí id, takový postup je na platformě Android standardem)
3. metodu `createSearchOptions()` která vytvoří seznam možných nastavení modulu.
4. metodu `fillDataFromCursorLine(Cursor cursor)` která převede jeden řádek dat z databáze na data vhodná k porovnávání.
5. metodu `getProjection()` která určuje jaká data (sloupce) z databáze požadujeme.
6. metodu `getSelection()` která omezuje data z předchozí metody podle zadaných kritérií.
7. metodu `dataAreAggregationCandidates(Data dataObject1, Data dataObject2)` která určuje zdali patří data ke stejnému kontaktu.

4.2.2 Implementované moduly

V programu jsou vytvořeny vyhledávací moduly. Prvním modulem je telefonní modul (`PhoneNumberModule`). Tento modul porovnává telefonní čísla kontaktů. Nejprve uloží do třídy `PhoneNumber` číslo bez mezinárodní předvolby, následně čísla porovnává. Určování předvolby u čísel se ukázalo jako složitý úkol, neboť předvolba může v různých státech sestávat z jednoho až tří čísel[23], není proto jednoduché předvolbu od čísla oddělit.

Druhým modulem je emailový modul (`EmailModule`), který porovnává emailové adresy uživatelů. Při defaultním nastavení modul odděluje doménovou část.

Dále modul odděluje části emailové adresy pomocí teček a dalších znaků. Tyto části adresy pak mezi sebou porovnává, přičemž v nastavení lze určit kolik shodných částí je potřeba, či zda záleží na jejich pořadí.

Dalším modulem je modul porovnávající názvy kontaktů (`NameModule`). Tento modul je v mnohém podobný jako modul předchozí, porovnává spolu části názvu (oddělené mezerou) dle nastavení uživatele.

4.2.3 Parazitní moduly

Při navrhování posledního modulu, který měl porovnávat jména s emailovými adresami vyvstala důležitá otázka: Pokud pracuje modul se stejnými daty jako jiný modul, má data znovu načítat z databáze? Odpověď nebyla jednoduchá. Uživatel totiž může libovolný modul deaktivovat, na data z něj tak nelze spoléhat. Nakonec bylo řešení nalezeno formou *parazitních* modulů.

Tyto moduly využijí data z již proběhlých modulů. pokud byly potřebné moduly deaktivovány, zavolá parazitní modul metodu `parasiteRun()` (nachází se ve třídě `simpleSearchModule`), která spustí načtení dat v daném modulu, ale nepokračuje hledáním kontaktů k propojení jako metoda `run()` téhož modulu.

Jediným parazitním modulem v programu je modul pro porovnávání emailů s názvy kontaktu (`EmaiNameModule`). Tento modul porovnává části emailové adresy s části názvu kontaktu. Takový přístup je velkým posunem vpřed, neboť porovnávání různých dat kontaktu je něco, co nenabízí ani vestavěné řešení platformy Android, ani testované konkurenční aplikace.

4.3 Datová vrstva

Centrem datové vrstvy je třída `dataManager`. Tato třída uchovává seznam nalezených kandidátů na spojení. Navíc obsahuje vlastní content provider a slouží tak pro komunikaci s centrálním úložištěm dat v telefonu. Pro tuto komunikaci obsahuje mnoho metod, které odstiňují aplikační vrstvu od contact provideru. Pomocí těchto metod získávají vyhledávací moduly data potřebná pro běh aplikace.

4.3.1 Získávání kontaktů k porovnání

Tato část aplikace vyžadovala velice dlouhou analýzu. Nejprve byl zvažován způsob získání všech kontaktních informací z db pomocí tzv. *entit*[22] (datová struktura

obsahující data o kontaktu v podobě v jaké jsou potřeba v aplikacích, tedy všechna data o jednom kontaktu na jednom místě), a následné práci nad těmito daty. Data by se tak nahrála do paměti na začátku procesu hledání, v době práce vyhledávacích modulů už by pak nebylo třeba přistupovat do databáze. Toto řešení nabízelo přehledné oddělení procesů práce s databází a vyhledávání, naproti tomu ale znamená větší zátěž na databázi, neboť by byla získávána i zcela nepotřebná data.

Druhou zvažovanou variantou byla možnost získávat jen potřebná data, ale opět hned na začátku procesu hledání, před během vyhledávacích modulů. Tato varianta byla taktéž zavržena. Kritickým problémem se totiž stal výčet potřebných dat. V případě že by byl v budoucnu vytvořen vyhledávací modul potřebující nové typ dat, bylo by nutné zasahovat výrazně i do datové vrstvy. Proto byla i tato varianta zavržena.

Nakonec byl zvolen takový způsob, kdy každý modul získává data až na začátku svojí práce, sám si určí která data potřebuje. Způsob se ukázal jako velmi účinný, fáze získávání dat z databáze není nejpomalejší částí práce modulu, tak jak by se od databázových operací očekávalo, je výrazně rychlejší než následné srovnávání dat, obvykle se ani nezobrazí ve stavovém řádku modulu (obr. 4.5 č. 4). Důvodem je to, že vyhledávací modul obvykle potřebuje jen jeden či dva sloupce z tabulky *data*, což je jen nepatrný zlomek z informací které vrací *entita*.

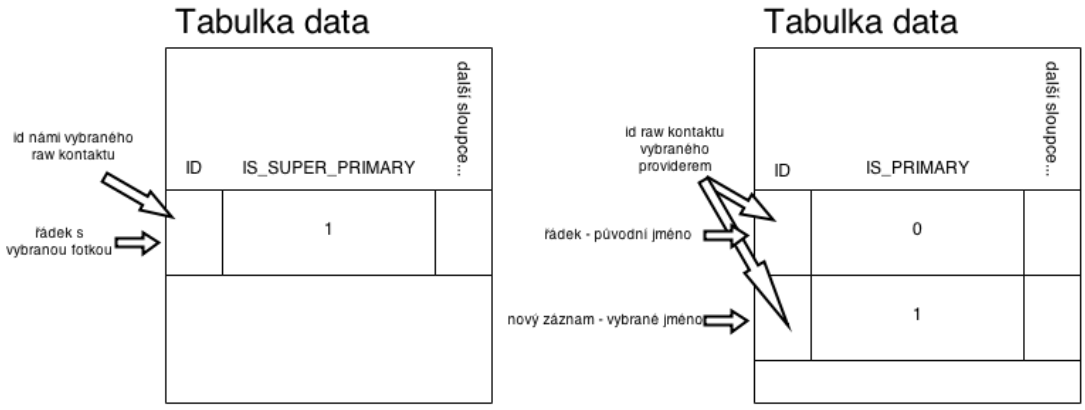
Záhy se ukázalo i jedno úskalí tohoto postupu, několik modulů totiž může pracovat se stejnými daty, tato data by pak byla získávána z databáze a uchovávána vícekrát. Tento problém ale řeší zavedení parazitních modulů popsaných v kapitole 4.2.

4.3.2 Propojování kontaktů

Propojování kontaktů probíhá pomocí přidávání řádek do tabulky `agg_exceptions` s identifikátory raw kontaktů které chceme propojit a s typem `TYPE_KEEP_TOGETHER` - obdobně jak to bylo ukázáno v kapitole 3.2.6. Po tom co jsou raw kontakty v aplikaci Contact Merge propojené, spustí se daleko náročnější proces. Je totiž třeba v databázi zohlednit volbu zobrazovacího jména a fotky kterou provedl uživatel.

Program nejdříve otestuje jestli se volba uživatele neshoduje s volbou provideru. Pokud se volba shoduje, není třeba dělat nic. Pokud se neshoduje, program volbu provideru *zastíní* svojí volbou (změnit/přepsat ji nelze, jak bylo vysvětleno v kapitole 3.3.2)

Pro zastínění jména je použita technika známá z dokumentace[13] a popsaná



Obrázek 4.11: Srovnání postupů při nastavení jména a fotky. V levé části je znázorněn postup pro nastavení fotky, v pravé pro nastavení jména.

kapitole 3.3.2. V téže kapitole je ale zmíněno že obdobný postup nelze použít pro určení názvu kontaktu. Po mnohých neúspěšných pokusech a dlouhém zvažování bylo nalezeno částečné řešení. Tím je vložení nového jména do dat raw kontaktu (staré jméno nemažeme), který byl vybrán contact providerem a nastavení tohoto jména jako primárního. Rozdílem je dále to, že u nového záznamu nastavujeme pouze sloupec `IS_PRIMARY`, neboť to ve kterém raw kontaktu se bude hledat primární jméno už za nás určil provider pomocí sloupce `NAME_RAW_CONTACT_ID`. Srovnání nastavení jména a fotky kontaktu je vidět na obrázku 4.11.

Tímto postupem je docíleno žádaného výsledku, kontakt se nyní zobrazuje v seznamu kontaktů pod vybraným jménem. Potíže nastávají pokud uživatel zařízení kontakt ručně rozdělí prostřednictvím seznamu kontaktů. V takovém případě budou mít oba oddělené kontakty stejné jméno.

Celá procedura zahrnuje několik operací modifikujících databázi. Provedení každého dotazu zvlášť by bylo pomalé. Proto byla zvolena metoda, při které se dotazy střádají do dávky, která je spuštěna pro všechny slučované kontakty najednou[24]. Pro uchovávání příkazů pro databázi se používá třída `ContentProviderOperation`, seznam operací je pak předán metodě `ContentResolver.applyBatch`.

4.3.3 Získávání facebookových kontaktů z databáze

Jak již bylo řečeno v kapitole 3.3.1, získání kontaktů ze sítě Facebook na standardním zařízení bez administrátorských práv není možné. Pokud však uživatel odemkl svůj telefon a užívá ho tedy jako administrátor, získání kontaktů možné je.

V prohlížení kontaktů z Facebooku nám brání přímo contact provider. Řešením je tedy provider nějakým způsobem obejít. V této části bude popsána technika *odemčení facebookových kontaktů* použitá v této aplikaci, jako i možné alternativní techniky.

Root na Androidu

Operační systém Android je z bezpečnostních důvodů k běžnému uživateli a aplikacím které spouští velmi restriktivní. Navíc nemá uživatel, na rozdíl od operačních systémů které známe ze stolních počítačů, zřízený administrátorský účet. Některé úkony, které známe ze světa Windows či Linuxu, jsou tak pro běžného vlastníka zařízení s OS Android zapovězena. Příkazy potřebné k zpřístupnění facebookových kontaktů tak, jak je to popsáno dále, bohužel patří do této množiny zapovězených úkonů. Proto je třeba provést takzvaný *root* zařízení, po kterém získá uživatel telefonu administrátorský přístup k zařízení, který aktivuje prostřednictvím příkazu `su` (podobně jako `sudo` na Linuxu) předcházejícím úkon, ke kterému je třeba administrátorských práv.

Způsob provedení rootu se liší podle typu zařízení, navíc se jedná o odbornou problematiku netýkající se tématu této práce, proto zde nebude probírán. Důležité je, že root lze provést téměř na každém zařízení, navíc aplikací které ho vyžadují je mnoho (bez rootu nebude na androidu plně fungovat ani běžný souborový manager typu Total Commander, nejdou odinstalovat některé aplikace...).

Technika použitá v aplikaci Contact Merge

Aplikace Contact Merge používá k přístupu ke kontaktům z Facebooku techniku odemykání kontaktů. Nejprve použijeme příkaz pomocí něhož celou databázi přesuneme do nezabezpečeného úložiště, například na SD kartu. Před přesunem je ještě potřeba vypnout contact provider, neboť budeme upravovat jeho databázi mimo něj, obejdeme ho. Po přesunu pak musíme změnit práva k souboru databáze, aby s ním mohla pracovat naše aplikace. Kód příkazů je vidět v kódu 4.2.

```
pm disable com.android.providers.contacts
cp /data/data/com.android.providers.contacts/databases/contacts2.db
   /data/data/fido.contacts.merge/contacts2.db
cd /data/data/fido.contacts.merge/
user=`ls -ld ../fido.contacts.merge | awk '{print $3}'`
chown $user:$user contacts2.db
```

Kód 4.2: Příkaz pro zkopírování databáze n kartu SD.

Následně můžeme manipulovat přímo se souborem databáze, neboť ho nechrání ani omezení na přístup do složek cizích aplikací, ani přístupová práva souborů. Databázi tedy upravíme pomocí SQL dotazu zadaného na nejnižší úrovni - dotazu u kterého se nekontroluje správnost, oprávnění, referenční integrita apod.. Takový dotaz můžeme zadat pomocí *raw query*[29]. Pro odemčení kontaktů stačí jednoduchý dotaz, jako v kódu 4.3.

```
UPDATE raw_contacts SET is_restricted = 0
')
```

Kód 4.3: Dotaz na databázi kontaktů pro odemčení kontaktů z Facebooku.

Po vykonání potřebného SQL dotazu přesuneme databázi zpět na místo. Opět je třeba změnit vlastníka souboru, tentokrát zpět na provider. Navíc nesmíme zapnout provider zapnout. Navíc je třeba vymazat soubor databáze a žurnálu ze složky naší aplikace. Vše je vidět v kódu 4.4.

```
cp -f /data/data/fido.contacts.merge/contacts2.db
   /data/data/com.android.providers.contacts/databases/contacts2.db
cd /data/data/com.android.providers.contacts/
user=`ls -ld databases | awk '{print $3}'`
chown $user:$user databases/contacts2.db
cd /data/data/fido.contacts.merge/
rm contacts2.db
rm contacts2.db-journal
pm enable com.android.providers.contacts
```

Kód 4.4: Příkaz pro zkopírování databáze zpět do adresáře contact provideru.

Jak již bylo řečeno, odemykání kontaktů funguje jen na zařízení kde má uživatel práva administrátora. Před vykreslením UI se aplikace Contact Merge přesvědčí, jestli má uživatel zařízení administrátorská práva (zdali jde o „rootnuté“ zařízení). Pokud práva nemá, tlačítko k odemknutí kontaktů se vůbec nezobrazí. K tomuto testu a k samotnému spuštění výše uvedených příkazů byl využit kód získaný ze stránky StackOverflow[21].

Další možnosti

Jednou z dalších možností jak se dostat ke kontaktům z Facebooku je SQL injektáž. Jedná se o techniku, kdy jsou provideru jako součást běžných SQL dotazů podstrčeny dotazy, které jinak provider nepovoluje. Tuto techniku jsem do detailu nezkoumal, neboť by taková práce nebyla programově čistá, použití takové techniky by mohlo mít nepříjemné právní důsledky, a v neposlední řadě by taková

aplikace byla téměř jistě stažena z obchodu Play.

Další zvažovanou variantou bylo použití programu sqlite3. Jedná se o konzolový program, který vykoná SQL dotaz na databázi, výstupem programu jsou pak navrácená data. Pokud před samotný program napíšeme v konzoli příkaz `su`, vykonáme příkaz s administrátorskými právy, díky tomu můžeme pracovat i s databází jiného programu. Tato varianta byla nakonec zavržena, program sqlite3 by musel být součástí distribuce programu Contact Merge, neboť již není součástí všech distribucí Android. Příklad příkazu který odemkne kontakty z platformy facebook můžeme vidět v kódu 4.5.

```
"su sqlite3
   /data/data/com.android.providers.contacts/databases/contacts2.db
   \"UPDATE raw_contacts SET is_restricted = 0\""
```

Kód 4.5: Příkaz pro odemčení facebookových kontaktů programem sqlite3[6].

4.4 Kompatibilita

Aplikace by měla být kompatibilní s jakýmkoliv zařízením s operačním systémem Android od verze 3.0 - Honeycomb (API level 11). Úspěšné testování proběhlo na mobilních telefonech Sony Xperia V, Sony Xperia L, Samsung GT-S762, Samsung Galaxy S3, dále pak na tabletech Prestigio 5580C duo a Samsung Tab 3.

5 Testování aplikace

V poslední kapitole před závěrem se zaměříme na testování aplikací. Nejprve srovnáme aplikaci Contact Merge s aplikacemi představenými v kapitole 2.1. Následně pak srovnáme nalezené výsledky s různým nastavením programu na telefonu Sony Xperia V.

5.1 Porovnání aplikací

Níže uvedené tabulka nabízí srovnání funkcí a schopností aplikace Contact Merge s dalšími testovanými aplikacemi. Nutno poznamenat že funkce v tabulce byly vybrány podle zadání bakalářské práce, proto se zde velká část funkcí ostatních testovaných aplikací vůbec nenachází. Dále je nutno říci, jak už bylo naznačeno v úvodu, že se testované aplikace k problému kontaktů staví jinak než aplikace Contact Merge. Z testování tedy nevyplynul závěr, že ostatní aplikace jsou horší než aplikace Contact Merge, ale spíše to, že nebyla nalezena žádná aplikace, jejíž účel a filozofie práce by odpovídali aplikaci Contact Merge.

Všechny aplikace byly testovány s defaultním nastavením. Testovacím zařízením byl telefon Sony Xperia V. Celkový počet kontaktů v telefonu byl 887. Srovnání aplikací vidíme v tabulce 5.1.

Z tabulky je vidět že v počtu nalezených výsledků aplikace Contact Merge s přehledem vede. Navíc selhávají ostatní aplikace i v dalších testovaných kritériích, neumějí pracovat s facebookovými kontakty, neumějí pracovat s kontakty z více zdrojů, není u nich ani možnost vybrat fotku či název výsledného kontaktu. Celkově lze s jistotou říci že aplikace Contact Merge zvítězila ve všech testovaných kritériích.

5.2 Porovnání různých nastavení

V následujících sekcích jsou vypsány výsledky modulů s různým nastavením. Celkově se jako nejúčinnější ukázal modul pro srovnávání jmen, srovnávání telefonních čísel pak bylo účinné nejméně. Moduly pro srovnání jmen a srovnávání jmen s emailovými adresami ukázaly prudký nárůst počtu výsledků, pokud srovnávaly různě dlouhá jména/maily. S tímto nastavením se ale také objevily falešné nálezy - v případě prvního jmenovaného jedna polovina, v druhém případě jedna třetina falešných nálezů.

Název	počet nálezů	propojuje kontakty[1]	výběr jména a fotky	analýza[2]	Facebook[3]	slučuje kontakty z více účtů
Contact Merge	15	ano	ano	ano	ano	ano
Merge +	6	ano	ne	ne	ne	ne
Contact optimizer	nelze určit[4]	ano[6]	ne	ne	ne	ne
Duplicate Contacts Manager	2	ano[6]	nelze zjistit[5]	ne	ne	nelze zjistit[5]

1. Pokud ne, tak pouze maže duplicitní kontakty.
2. Pracuje s kontakty ze sítě Facebook - nutno root zařízení.
3. Určuje, zda byly shodné kontakty nalezené dle analýzy, nebo přesné shody databázového pole (jméno, tel. číslo...). Tento údaj je třeba brát s rezervou, byl určen na základě nalezených výsledků. Pokud se údaje u všech nalezených výsledků přesně shodovaly, je zvolena hodnota ne.
4. Nálezů bylo více než 40, ale z pěti testovaných nálezů byly všechny v telefonu již propojeny.
5. nelze zjistit, neboť zdarma je pouze detekce duplicit, funkce propojování kontaktů je placená.
6. ano, ale většina funkcí aplikace směřuje k mazání duplicit, nikoliv k jejich propojování.

Tabulka 5.1: Srovnání dostupných aplikací s aplikací Contact Merge v oblastech zadání práce.

Defaultní nastavením je vždy zvýrazněno tučně. Testovaným telefonem byl opět Sony Xperia V s 887 kontakty.

5.2.1 Email module

Email module má jen průměrné výsledky. Je to především proto, že byl implementován jako první, nebyly proto vytvořeny žádné možnosti hlubší analýzy. Na druhou stranu ale několik shod našel, navíc se nevyskytly žádné falešné nálezy.

Nastavení:

- vše vypnuto - 3 nálezy
- **zapnuta možnost include similar emails (testuje adresy bez domény) - 5 nálezů.**

5.2.2 Phone number module

Phone number module mnoho nálezů nemá. Je to proto, že čísla kontaktů porovnává už operační systém Android, kontakty se stejnými čísly se tak sloučí automaticky.

Nastavení:

- vše vypnuto - 1 nález
- **test number without country calling code (testování bez předvolby čísla) - 2 nálezy**

5.2.3 Name module

Největší počet nálezů má modul pro srovnávání jmen. Devět nálezů s defaultním nastavením je slušný výsledek. S rozšířeným nastavením najde modul ještě daleko větší množství výsledků, více než polovina nálezů je ale falešných, výsledky takového hledání proto musíme pečlivě projít.

Nastavení:

- **vše vypnuto - 9 nálezů**

- ignore name order (záměna pořadí jméno příjmení) - 9 nálezů
- compare names with different length (porovnávání různě dlouhých jmen, např. jedno a dvouslovných) - 71 nálezů (z toho 37 falešných nálezů)

5.2.4 Email/Name module

Výsledky modulu pro srovnání jmen s defautním nastavením nejsou oslnivé. Při spuštění s rozšířeným nastavením ale nalezl přes dvacet výsledků, z nichž pouhá třetina byly falešné nálezy.

Nastavení:

- **vše vypnuto - 3 nálezy**
- ignore name order (záměna pořadí částí mailu/jména) - 3 nálezy
- compare names with different length (porovnávání různě dlouhých mailů/jmen) - 22 nálezů (z toho 7 falešných nálezů)

6 Závěr

Tato práce se zabývá problematikou práce s kontakty na platformě Android. Na tomto nejrozšířenějším operačním systému na mobilních zařízeních je většinou uloženo mnoho kontaktů nejen zadaných uživatelem přímo do telefonu, ale také synchronizovaných z různých sítí a služeb. Často je pak v telefonu uloženo více kontaktů patřících ke stejné osobě. Cílem práce bylo takové kontakty najít a propojit.

Prvním úkolem této práce bylo najít a zhodnotit existující aplikace s podobným zaměřením jako tato práce na platformě Android. Byly vybrány a zhodnoceny tři aplikace, z nichž ani jedna nesplňovala cíle, které si klade tato práce. Nalezených duplicit bylo málo, některé aplikace se snažily takových kontaktů zbavit, nikoliv je propojit, v žádné aplikaci pak nebylo možné pracovat s kontakty ze sítě Facebook. Zhodnocení aplikací tedy potvrdilo důležitost vytvoření vlastní aplikace, neboť jejich výsledky zaostávaly za cíli práce.

V další části práce se podařilo analyzovat práci s kontakty na platformě Android. Většina této problematiky je pokrytá v oficiální dokumentaci, k některým ale bylo nutné hledat informace jinde a vyskytla se i taková témata, u kterých bylo nutné najít odpovědi na otázky vlastním výzkumem. K takovým oblastem patřila především práce s facebookovými kontakty nebo změna názvu a fotky kontaktu.

Aplikace dle zadání práce byla navržena a realizována, cíle práce se podařilo splnit. Práce s kontakty ze sítě Facebook je sice omezená - funguje jen na zařízeních s administrátorským přístupem, tato situace je však způsobena záměrným zablokováním těchto kontaktů proti přístupu aplikací. Je tak velkým úspěchem, že se podařilo s těmito kontakty pracovat alespoň na zařízeních s administrátorským přístupem. Velkým přínosem aplikace je také možnost určit název a fotku výsledného kontaktu.

Vyhledávací část aplikace byla realizována pomocí vyhledávacích modulů. Tyto moduly byly navrženy pro snadnou rozšiřitelnost, není tak problém vytvořit moduly další. Díky Parazitním modulům je pak možné využít data z již existujících modulů a implementovat nově jen porovnávání kontaktních dat.

Nakonec byla aplikace otestována, byla provedena testovací hledání kontaktů k propojení s různými moduly a různým nastavením modulů. Testy ukázaly užitečnost aplikace, i při defaultním nastavení, kde se nevyskytují falešné nálezy, bylo nalezeno více než deset dvojic kontaktů k propojení.

Celkově se zadání práce podařilo splnit, a to jak v teoretické tak praktické rovině. Praktickou část - aplikaci je možné v budoucnu rozšířit o další moduly

a zvýšit tak účinnost hledání. Z teoretické části je pak možné čerpat při dalších projektech zabývajících se prací s kontakty na platformě Android. Celkově je tak, dle mého názoru moje práce přínosem v teoretické i praktické rovině.

Bibliografie

[1] Google Inc., Open Handset Alliance. *Activity* [online]. URL: http://developer.android.com/reference/android/app/Activity.html [cit. 2014-04-02] .

[2] Allen, Grant. *Beginning Android 4*. New York: Apress, 2012. Kap. Handling Rotation. 582 s. ISBN: 978-1-4302-3984-0.

[3] andr...@JRAF.org. *Restricted contacts limit external Android app integration* [online]. 28. srp. 2010. URL: http://code.google.com/p/android/issues/detail?id=5176 [cit. 2013-12-26] .

[4] Google Inc., Open Handset Alliance. *AsyncTask* [online]. URL: http://developer.android.com/reference/android/os/AsyncTask.html [cit. 2013-12-08] .

[5] Aydin, Murat. *Android 4: New features for Application Development*. Birmingham: Packt Publishing, 2012. Kap. Fragments. 149 s. ISBN: 978-1-84951-952-6.

[6] Biber. *Open sqlite-database of another app on Android with root-access* [online]. 23. dub. 2012. URL: http://stackoverflow.com/questions/16142844/open-sqlite-database-of-another-app-on-android-with-root-access [cit. 2014-01-17] .

[7] Boyle, Chris. *ContactsProvider: Aggregated contact's display name should respect IS_SUPER_PRIMARY* [online]. 9. ún. 2010. URL: http://code.google.com/p/android/issues/detail?id=6545 [cit. 2014-03-15] .

[8] Cervilla, Jose. *Basic AsyncTask with a progress bar widget* [online]. 16. červ. 2011. URL: http://www.codingforandroid.com/2011/06/basic-asynctask-with-progress-bar.html [cit. 2013-12-08] .

[9] Google Inc., Open Handset Alliance. *Contacts Provider* [online]. URL: http://developer.android.com/guide/topics/providers/contacts-provider.html [cit. 2013-11-12] .

[10] DesignerzBase. *Contacts3 icon* [online]. URL: https://www.iconfinder.com/icons/216476/contacts3_icon#size=128 [cit. 2014-05-07] .

[11] Google Inc., Open Handset Alliance. *ContactsContract.AggregationExceptions* [online]. URL: http://developer.android.com/reference/android/provider/ContactsContract.AggregationExceptions.html [cit. 2013-12-22] .

[12] Google Inc., Open Handset Alliance. *ContactsContract.Contacts* [online]. URL: http://developer.android.com/reference/android/provider/ContactsContract.Contacts.html [cit. 2014-02-22] .

[13] Google Inc., Open Handset Alliance. *ContactsContract.data* [online]. URL: http://developer.android.com/reference/android/provider/ContactsContract.Data.html [cit. 2014-03-08] .

[14] Google Inc., Open Handset Alliance. *Content Provider Basics* [online]. URL: http://developer.android.com/guide/topics/providers/content-provider-basics.html [cit. 2013-11-05] .

[15] Google Inc., Open Handset Alliance. *Content Providers* [online]. URL: http://developer.android.com/guide/topics/providers/content-providers.html [cit. 2013-11-05] .

[16] DPLOTNIKOV. *How access restricted contacts in third party app* [online]. 26. bře. 2010. URL: https://groups.google.com/forum/#!topic/android-developers/u2TdwbBUpBI [cit. 2013-12-26] .

[17] DPLOTNIKOV. *How to get a contact's facebook id or url from native contacts / content resolver?* [online]. 22. pros. 2010. URL: https://groups.google.com/forum/#!topic/android-developers/lREN16Hh4LQ [cit. 2013-12-28] .

[18] Google Inc., Open Handset Alliance. *Fragment* [online]. URL: http://developer.android.com/reference/android/app/Fragment.html [cit. 2013-11-10] .

[19] Google Inc., Open Handset Alliance. *Fragments* [online]. URL: http://developer.android.com/guide/components/fragments.html [cit. 2013-11-10] .

[20] I., Roger. *Cannot find Facebook contacts in RawContacts* [online]. 28. květ. 2011. URL: http://stackoverflow.com/questions/6038290/cannot-find-facebook-contacts-in-rawcontacts [cit. 2013-12-28] .

[21] KEVIN. *Determine if running on a rooted device* [online]. 11. lis. 2011. URL: http://stackoverflow.com/questions/1101380/determine-if-running-on-a-rooted-device [cit. 2014-03-16] .

[22] KOMATINENI, Satya – MACLEAN, Dave. *Pro Android 4: Android 4 platform SDK techniques for developing smartphone and tablet apps*. 4th edition. New York: Apress, 2012. Kap. Exploring the Contacts API. 991 s. ISBN: 978-1-4302-3930-7.

[23] *List of ITU-T recommendation E.164 assigned country codes*. Tech. zpr. 991. International telecomunication union, 2011. URL: http://www.itu.int/dms_pub/itu-t/opb/sp/T-SP-E.164D-11-2011-PDF-E.pdf.

[24] MANITOBA. *Android - Update a contact* [online]. 29. břez. 2012. URL: http://stackoverflow.com/questions/9907751/android-update-a-contact [cit. 2014-03-29] .

[25] . *Resize Drawable in Android* [online]. 2. ún. 2014. URL: http://stackoverflow.com/questions/7021578/resize-drawable-in-android [cit. 2014-05-06] .

[26] SAMUEL. *How do you get contacts to aggregate properly when programmatically adding them?* [online]. 23. ún. 2012. URL: http://stackoverflow.com/questions/9419305/how-do-you-get-contacts-to-aggregate-properly-when-programmatically-adding-them [cit. 2013-12-22] .

[27] TAMADA, Ravi. *Android Expandable List View Tutorial* [online]. 27. čvc 2013. URL: http://www.androidhive.info/2013/07/android-expandable-list-view-tutorial/ [cit. 2014-02-05] .

[28] VARUN. *How to programatically set drawableLeft on Android button?* [online]. 21. pros. 2010. URL: http://stackoverflow.com/questions/4502605/how-to-programatically-set-drawableleft-on-android-button [cit. 2014-05-06] .

[29] VOGEL, Lars. *Android SQLite database and content provider - Tutorial* [online]. Ver. 4.9. 19. srp. 2013. URL: http://www.vogella.com/tutorials/AndroidSQLite/article.html [cit. 2014-03-02] .

Přílohy

A Instalační příručka

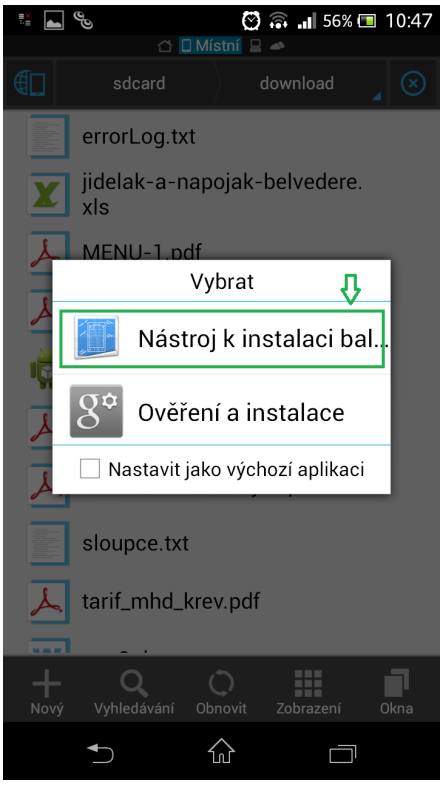
Zde se nachází stručný postup instalace aplikace pomocí instalačního balíčku (souboru .apk). Body označené hvězdičkou nastanou jen v některých případech či konfiguracích telefonu

1. Odinstalujte předchozí verze aplikace.*
2. Nahrajte instalační balíček do nechráněné části souborového systému na zařízení Android - např. na kartu SD.
3. Poklepněte na instalační balíček.
4. Pokud se zobrazí menu dostupných aplikací, zvolte *Nástroj k instalaci balíčků*, jak je ukázáno na obrázku A.1a.*
5. Pokud se zobrazí informace že instalace je zablokována protože je její zdroj neznámý, je nutné povolit instalaci z neznámých zdrojů. Zvolte *Nastavení* (obrázek A.2a), dále sekci zabezpečení A.2b) zde pak zaškrtněte možnost *Neznámé zdroje* (obrázek A.2c). Celý postup je vidět na obrázku A.2.*
6. Odsouhlaste seznam práv která aplikace vyžaduje zvolením tlačítka instalovat - obrázek A.1b.

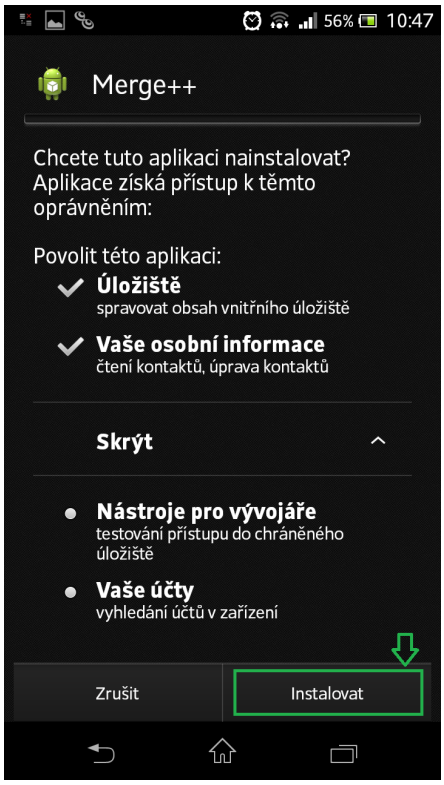
B Uživatelská příručka

Celé schéma aplikace z pohledu uživatele vidíme na obrázku B.1. Nejprve spustíme aplikaci poklepáním na její ikonu v menu. Objeví se nám úvodní aktivita (obrázek 4.3). Zde je možné aktivovat/deaktivovat vyhledávací moduly klepnutím na checkbox napravo vedle modulu (obrázek B.2a číslo 1), po klepnutí na modul (viz obrázek B.2a číslo 2) mimo checkbox se dostaneme do nastavení modulu (viz. dále). Na zařízeních s administrátorskými právy je navíc možné odemknout kontakty ze sítě facebook klepnutím n tlačítko Unlock Facebook contacts (viz. obrázek B.2a číslo 3). Stejným tlačítkem je možné následně kontakty zamknout zpět.

V aktivitě nastavení modulu můžeme klepnutím na checkbox aktivovat/deaktivovat příslušné nastavení (obrázek B.2b). Po nastavení všech modulů a jejich

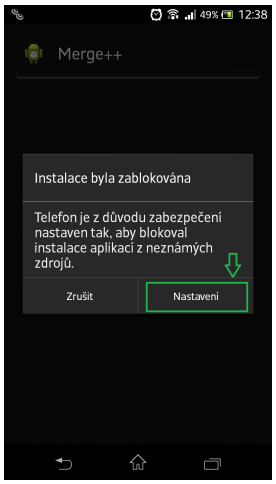


(a) Výběr nástroje pro instalaci balíčků

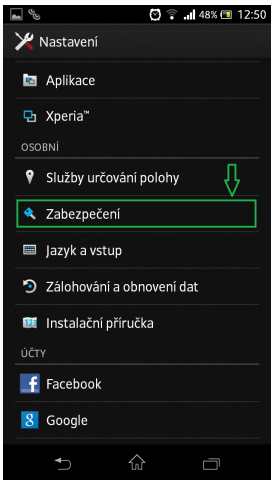


(b) Schválení oprávnění

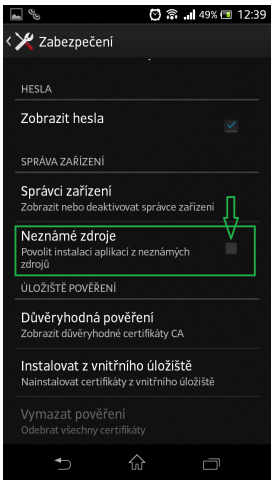
Obrázek A.1: Návod na instalaci



(a) Dialog zablokování instalace

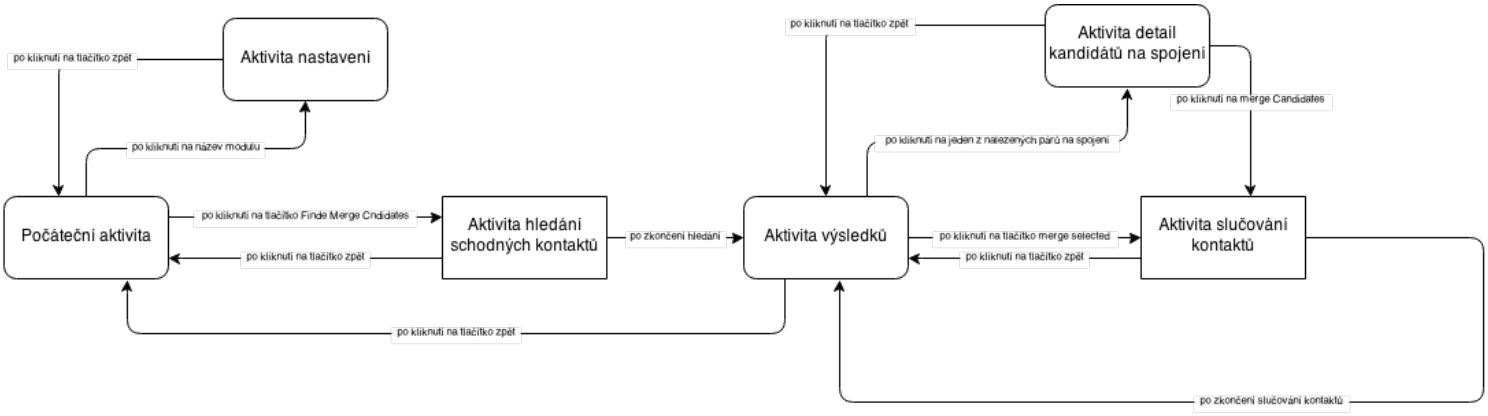


(b) Sekce zabezpečení



(c) Možnost instalovat aplikace neznámých zdrojů

Obrázek A.2: Instalace aplikací z neznámých zdrojů



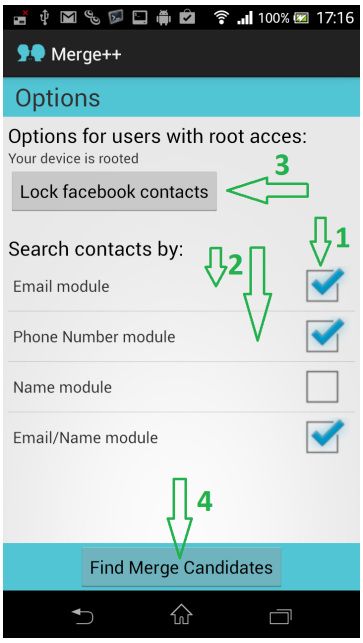
Obrázek B.1: Schéma fungování aplikace z pohledu uživatele

aktivování dle našich představ pokračujeme kliknutím na tlačítko Find merge candidates (obrázek B.2a číslo 4). Následně se zobrazí dialog z indikátorem stavu hledání, který již známe z obrázku 4.5.

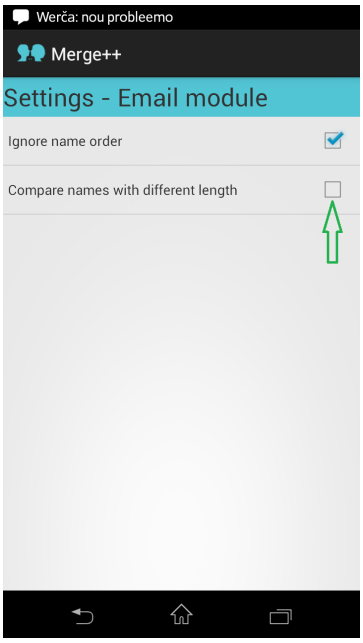
Po skončení hledání se zobrazí aktivita výsledků (obrázek 4.6). Zde vidíme dvojice kontaktů, u kterých program zhodnotil, že patří ke stejné osobě. Zbývá kontakty spojit. Pro spojení kontaktů má uživatel dvě možnosti.

První možností je individuální propojení kontaktů. Nejprve přejdeme na aktivitu detailu kandidátů ke spojení(viz. obrázek 4.7), a to tak že klepneme na řádek v seznamu na kterém jsou jména kontaktů které chceme spojit (obrázek B.3a číslo 1). Pokud kontakty nepatří k sobě, můžeme použít tlačítko discard (obrázek B.3b číslo 1), které nás vrátí zpět na seznam výsledků, ze kterého bude prohlížená dvojice kontaktů vyřazena. Pokud k sobě kontakty patří, klepneme na tlačítko Aggregate contacts (obrázek B.3b číslo 2). Zobrazí se dialog s výběrem jména (obrázek 4.8). Zde pomocí radio buttonů vybereme název a fotku kontaktu (obrázek B.4a číslo 1). následně stiskneme tlačítko Merge Selected (obrázek B.4a číslo 2), čímž dojde ke sloučení kontaktů, a my se vrátíme do aktivity výsledků.

Druhá možnost je sloučit kontakty hromadně. Nejprve zaškrtneme kontakty, které chceme sloučit pomocí checkboxů v aktivitě výsledků (obrázek B.3a číslo 2). Pak klepneme na tlačítko Merge selected (obrázek B.3a číslo 3). Tím se dostaneme do aktivity sloučení více kontaktů (obrázek 4.9). Zobrazí se nám seznam vybraných dvojic kontaktů k propojení. Pokud chceme rozhodnout, jakou fotku a název bude mít výsledný kontakt, rozbalíme položku seznamu pomocí poklepání. V rozbalené dvojici kontaktů vybereme jméno a fotku kontaktu, obdobně jako v případě individuálního propojení kontaktů (obrázek B.4b číslo 1). Po klepnutí na tlačítko merge dojde ke sloučení všech kontaktů ze seznamu a k návratu do aktivity výsledků (obrázek B.4b číslo 2).

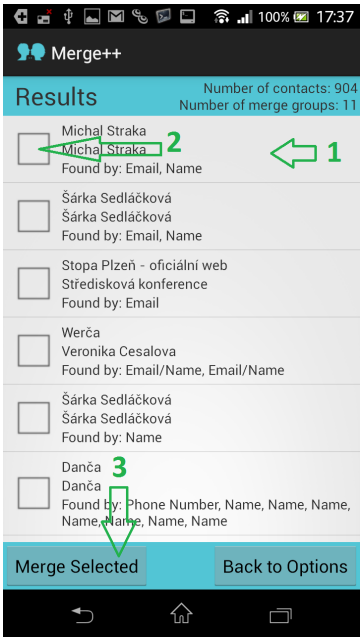


(a) úvodní aktivita

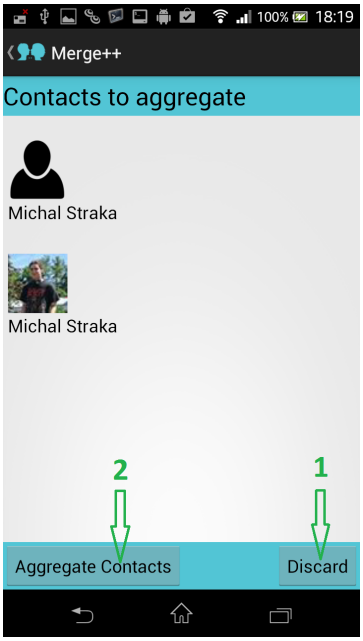


(b) aktivita nastavení

Obrázek B.2: Příručka - úvodní aktivita a aktivita nastavení.

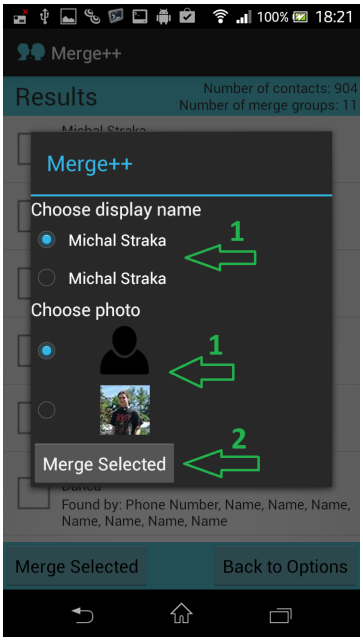


(a) aktivita výsledků

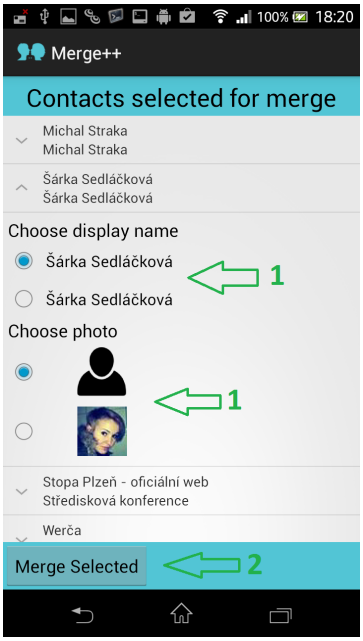


(b) aktivita detailu

Obrázek B.3: Příručka - aktivita výsledků a aktivita detailu.



(a) aktivita individuálního slučování



(b) aktivita hromadného slučování

Obrázek B.4: Příručka - aktivita individuálního a aktivita hromadného slučování.

Program je možné v kterékoliv fázi bezpečně opustit, v nejhorším případě ztratíme nalezené výsledky, které můžeme ovšem snadno opět vyhledat.

C Sloupce tabulky kontaktů

* Hodnota konstanty v kontraktní třídě = skutečné jméno tabulky.

C.1 Základní

Typ	Jméno	hodnota*	význam
long	_ID	_id	identifikátor řádku
String	LOOKUP_KEY	lookup	klíč sloužící k nalezení kontaktu, pokud se jeho id změní v důsledku sloučení kontaktů, nebo synchronizace
long	NAME_RAW_CONTACT_ID	name_raw_contact_id	klíč k raw kontaktu který poskytuje název kontaktu
String	DISPLAY_NAME_PRIMARY	display_name	zobrazené jméno kontaktu. Je to primární jméno raw kontaktu, na který ukazuje NAME_RAW_CONTACT_ID
long	PHOTO_ID	photo_id	klíč k řádku v tabulce ContactsContract.Data kde je uložena fotka kontaktu.
long	PHOTO_URI	photo_uri	Odkaz na fotku sloužící ke zobrazení
long	PHOTO_THUMBNAIL_URI	photo_thumb_uri	Odkaz na náhled fotky sloužící ke zobrazení
int	IN_VISIBLE_GROUP	in_visible_group	Indikuje zdali má být kontak zobrazen. ”1”pokud má kontakt alespoň jeden viditelný raw kontakt, jinak ”0”.
int	HAS_PHONE_NUMBER	has_phone_number	Indikuje zdali má kontakt telefoní číslo. ”1”pokud ano, jinak ”0”
int	TIMES_CONTACTED	times_contacted	Kolikrát byl kontak použit.
long	LAST_TIME_CONTACTED	last_time_contacted	Datum a čas posledního použití kontaktu.
int	STARRED	starred	Indikuje zdali se jedná o oblíbený kontakt: ’1’ značí oblíbený kontakt, jinak ’0’.
String	CUSTOM_RINGTONE	custom_ringtone	Odkaz na melodii specifickou pro kontakt.
int	SEND_TO_VOICEMAIL	send_to_voicemail	Indikuje, zdali má kontakt při volání okamžitě spadnout do hlasové schránky’1’ pro ano, jinak ’0’.
int	CONTACT_PRESENCE	contact_presence	Status pro IM.
String	CONTACT_STATUS	contact_status	Status pro IM.
long	CONTACT_STATUS_TIMESTAMP	contact_status_ts	Čas v milisekundách od poslední změny statusu pro IM
String	CONTACT_STATUS_RES_PACKAGE	contact_status_res_package	Balíček poskytující zdroje pro status IM: popiska a ikona
long	CONTACT_STATUS_LABEL	contact_status_label	IDd popisku statusu pro IM
long	CONTACT_STATUS_ICON	contact_status_icon	ID ikony statusu pro IM.

C.2 Dostupné přes implicit-join

Typ	Jméno	hodnota*	význam
String	DISPLAY_NAME_ALTERNATIVE	display_name_alt	Alternativní zápis zobrazovaného jména
String	DISPLAY_NAME_SOURCE	display_name_source	Typ dat použitý jako zobrazované jméno
String	PHONETIC_NAME	phonetic_name	Výslovnost jména pomocí fonetické abecedy specifikované ve sloupci PHONETIC_NAME_STYLE.
String	PHONETIC_NAME_STYLE	phonetic_name_style	fonetická abeceda pomocí níž je zakódováno PHONETIC_NAME.
String	SORT_KEY_ALTERNATIVE	sort_key_alt	klíč řazení dle sloupce DISPLAY_NAME_ALTERNATIVE.
String	SORT_KEY_PRIMARY	sort_key	klíč k řazení dle zobrazovaného jména
String	CONTACT_CHAT_CAPABILITY	contact_chat_capability	Schopnosti kontaktu v souvislosti s IM
String	IS_USER_PROFILE	is_user_profile	Značí jestli je tento kontakt uživatelským profilem
String	PHOTO_FILE_ID	photo_file_id	identifikátor souboru fotky v plné velikosti

D Sloupce tabulky raw kontaktů

D.1 Základní

Typ long	Jméno _ID	hodnota* _id	význam vlastní klíč raw kontaktu
long	CONTACT_ID	contact_id	klíč ukazující na tabulku ContactsContract.Contacts - ukazuje na kontakt pod který raw kontakt patří.
int	AGGREGATION_MODE	aggregation_mode	Pomocí konstant nastavuje automatické slučování kontaktů.
int	DELETED	deleted	značí zdali je kontakt určen k vymazání ”0”pokud ne, ”1”pokud ano.
int	TIMES_CONTACTED	times_contacted	Počet použití kontaktu.
long	LAST_TIME_CONTACTED	last_time_contacted	Časová známka posledního použití kontaktu
int	STARRED	starred	Určuje zdali patří raw kontakt mezi oblíbené - ’1’ pokud ano, jinak ’0’
String	CUSTOM_RINGTONE	custom_ringtone	Speciální vyzvánění pro raw kontakt.
int	SEND_TO_VOICEMAIL	send_to_voicemail	Určuje, zdali spadne volající raw kontakt rovnou do hlasové schránky (’1’) nebo ne (’0’).
String	ACCOUNT_NAME	account_name	Jméno účtu ze kterého raw kontakt pochází (například gmail pro účet na googlu)
String	ACCOUNT_TYPE	account_type	Typ účtu ze kterého pochází raw kontakt (pro účet na googlu com.google)
String	DATA_SET	data_set	Využívá se pro identifikaci rw kontaktů patřících ke stejnému účtu, ale k různým adaptérům pro synchronizaci.
String	SOURCE_ID	sourceid	identifikátor vzhledem k uživatelskému účtu. Vytváří se na straně servru.
int	VERSION	version	značí kolikrát byl kontakt změněn
int	DIRTY	dirty	značí že VERSION se změnila, a kontakt je třeba synchronizovat se serverem.
String	SYNC1	sync1	Generické sloupce pro potřeby sync adaptérů.
String	SYNC2	sync2	Generické sloupce pro potřeby sync adaptérů.
String	SYNC3	sync3	Generické sloupce pro potřeby sync adaptérů.
String	SYNC4	sync4	Generické sloupce pro potřeby sync adaptérů.

D.2 Dostupné přes implicit-join

Typ	Jméno	hodnota*	význam
String	DISPLAY_NAME_ALTERNATIVE	display_name_alt	Alternativní zápis zobrazovaného jména
String	DISPLAY_NAME_PRIMARY	display_name	Zobrazované jméno kontaktu
String	DISPLAY_NAME_SOURCE	display_name_source	Typ dat použitý jako zobrazované jméno
String	PHONETIC_NAME	phonetic_name	Výslovnost jména pomocí fonetické abecedy specifikované ve sloupci PHONETIC_NAME_STYLE.
String	PHONETIC_NAME_STYLE	phonetic_name_style	fonetická abeceda pomocí níž je zakódováno PHONETIC_NAME.
String	SORT_KEY_ALTERNATIVE	sort_key_alt	klíč řazení dle sloupce DISPLAY_NAME_ALTERNATIVE.
String	SORT_KEY_PRIMARY	sort_key	klíč k řazení dle zobrazovaného jména
String	RAW_CONTACT_IS_USER_PROFILE	raw_contact_is_user_profile	Označuje jestli je tento raw kontakt součástí uživatelského profilu.

E Sloupce tabulky dat

E.1 Základní

Typ	Jméno	hodnota*	význam
String	_ID	_id	identifikátor řádku
String	MIMETYPE	mimetype	MIME typ záznamu
String	RAW_CONTACT_ID	raw_contact_id	Klíč k raw kontaktu ke kterému data patří.
String	IS_PRIMARY	is_primary	Určuje, zdali je tento záznam primárním záznamem tohoto druhu v raw kontaktu.
String	IS_SUPER_PRIMARY	is_super_primary	Určuje, zdali je tento záznam primárním záznamem tohoto druhu v kontaktu - tedy ve všech spojených raw kontaktech.
String	DATA_VERSION	data_version	verze záznamu.
String	DATA1	data1	Generický sloupec, jeho obsah záleží na MIMETYPE dat.
String	DATA2	data2	Generický sloupec, jeho obsah záleží na MIMETYPE dat.
String	DATA3	data3	Generický sloupec, jeho obsah záleží na MIMETYPE dat.
String	DATA4	data4	Generický sloupec, jeho obsah záleží na MIMETYPE dat.
String	DATA5	data5	Generický sloupec, jeho obsah záleží na MIMETYPE dat.
String	DATA6	data6	Generický sloupec, jeho obsah záleží na MIMETYPE dat.
String	DATA7	data7	Generický sloupec, jeho obsah záleží na MIMETYPE dat.
String	DATA8	data8	Generický sloupec, jeho obsah záleží na MIMETYPE dat.
String	DATA9	data9	Generický sloupec, jeho obsah záleží na MIMETYPE dat.
String	DATA10	data10	Generický sloupec, jeho obsah záleží na MIMETYPE dat.
String	DATA11	data11	Generický sloupec, jeho obsah záleží na MIMETYPE dat.
String	DATA12	data12	Generický sloupec, jeho obsah záleží na MIMETYPE dat.
String	DATA13	data13	Generický sloupec, jeho obsah záleží na MIMETYPE dat.
String	DATA14	data14	Generický sloupec, jeho obsah záleží na MIMETYPE dat.
String	DATA15	data15	Generický sloupec, jeho obsah záleží na MIMETYPE dat.
String	SYNC1	data_sync1	Generický sloupec pro sync adaptéry.
String	SYNC2	data_sync3	Generický sloupec pro sync adaptéry.
String	SYNC3	data_sync2	Generický sloupec pro sync adaptéry.
String	SYNC4	data_sync4	Generický sloupec pro sync adaptéry.

E.2 Dostupné přes implicit-join

Typ	Jméno	hodnota*	význam
String	DISPLAY_NAME_ALTERNATIVE	display_name_alt	Alternativní zápis zobrazovaného jména
String	DISPLAY_NAME_PRIMARY	display_name	Zobrazované jméno kontaktu
String	DISPLAY_NAME_SOURCE	display_name_source	Typ dat použitý jako zobrazované jméno
String	PHONETIC_NAME	phonetic_name	Výslovnost jména pomocí fonetické abecedy specifikované ve sloupci PHONETIC_NAME_STYLE.
String	PHONETIC_NAME_STYLE	phonetic_name_style	fonetická abeceda pomocí níž je zakódováno PHONETIC_NAME.
String	SORT_KEY_ALTERNATIVE	sort_key_alt	klíč řazení dle sloupce DISPLAY_NAME_ALTERNATIVE.
String	SORT_KEY_PRIMARY	sort_key	klíč k řazení dle zobrazovaného jména
String	CUSTOM_RINGTONE	custom_ringtone	Odkaz na melodii specifickou pro kontakt.
String	LAST_TIME_CONTACTED	last_time_contacted	Datum a čas posledního použití kontaktu.
String	SEND_TO_VOICEMAIL	send_to_voicemail	Indikuje, zdali má kontakt při volání okamžitě spadnout do hlasové schránky'1' pro ano, jinak '0'.
String	STARRED	starred	Indikuje zdali se jedná o oblíbený kontakt: '1' značí oblíbený kontakt, jinak '0'.
String	TIMES_CONTACTED	times_contacted	Kolikrát byl kontak použit.
String	CONTACT_CHAT_CAPABILITY	contact_chat_capability	Schopnosti kontaktu v souvislosti s IM
String	CONTACT_PRESENCE	contact_presence	Status pro IM.
String	CONTACT_STATUS	contact_status	Status pro IM.
String	CONTACT_STATUS_ICON	contact_status_icon	ID ikony statusu pro IM.
String	CONTACT_STATUS_LABEL	contact_status_label	IDd popisku statusu pro IM
String	CONTACT_STATUS_RES_PACKAGE	contact_status_res_package	Balíček poskytující zdroje pro status IM: popiska a ikona
String	CONTACT_STATUS_TIMESTAMP	contact_status_ts	Čas v milisekundách od poslední změny statusu pro IM
int	HAS_PHONE_NUMBER	has_phone_number	Indikuje zdali má kontakt telefoní číslo. "1"pokud ano, jinak "0"
int	IN_VISIBLE_GROUP	in_visible_group	Indikuje zdali má být kontak zobrazen. "1"pokud má kontakt alespoň jeden viditelný raw kontakt, jinak "0".
String	LOOKUP_KEY	lookup	klič sloužící k nalezení kontaktu, pokud se jeho id změní v důsledku sloučení kontaktů, nebo synchronizace
String	PHOTO_FILE_ID	photo_file_id	identifikátor souboru fotky v plné velikosti
long	PHOTO_ID	photo_id	klíč k řádku v tabulce ContactsContract.Data kde je uložena fotka kontaktu.
long	PHOTO_THUMBNAIL_URI	photo_thumb_uri	Odkaz na náhled fotky sloužící ke zobrazení
long	PHOTO_URI	photo_uri	Odkaz na fotku sloužící ke zobrazení

long	CONTACT_ID	contact_id	klíč ukazující na tabulku ContactsContract.Contacts - ukazuje na kontakt pod který raw kontakt patří.
String	DATA_SET	data_set	Využívá se pro identifikaci rw kontaktů patřících ke stejnému účtu, ale k různým adaptérům pro synchronizaci.
String	RAW_CONTACT_IS_USER_PROFILE	raw_contact_is_user_profile	Označuje jestli je tento raw kontakt součástí uživatelského profilu.
String	CHAT_CAPABILITY	chat_capability	Schopnosti kontaktu (audio, video. . .)
String	PRESENCE	mode	Poslední úroveň přítomnosti kontaktu
String	STATUS	status	Poslední aktualizace statusu
String	STATUS_ICON	status_icon	id na ikonu statusu
String	STATUS_LABEL	status_label	id textu statusu
String	STATUS_RES_PACKAGE	status_res_package	zdroj pro ikonu a text statusu
String	STATUS_TIMESTAMP	status_ts	čas poslední změny statusu
String	ACCOUNT_NAME	account_name	Jméno účtu ze kterého raw kontakt pochází (například gmail pro účet na googlu)
String	ACCOUNT_TYPE	account_type	Typ účtu ze kterého pochází raw kontakt (pro účet na googlu com.google)
int	DIRTY	dirty	značí že VERSION se zmněnila, a kontakt je třeba synchronizovat se servrem.
String	SOURCE_ID	sourceid	identifikátor vzhledem k uživatelskému účtu. Vytváří se na straně servru.
int	VERSION	version	značí kolikrát byl kontakt změněn
String	GROUP_SOURCE_ID	group_sourceid	sourceid skupiny ke které patří tento člen. Musí být specifikováno, nebo musí být specifikováno GROUP_ROW_ID.