VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

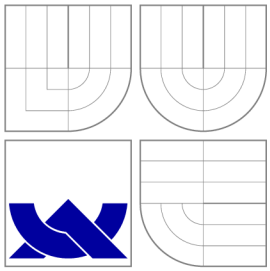
SLOVNÍK PRO MOBILNÍ ZAŘÍZENÍ S OS ANDROID

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

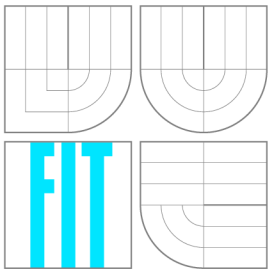
AUTOR PRÁCE
AUTHOR

TOMÁŠ DAMBORSKÝ

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

SLOVNÍK PRO MOBILNÍ ZAŘÍZENÍ S OS ANDROID

DICTIONARY FOR MOBILE DEVICES WITH OS ANDROID

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

TOMÁŠ DAMBORSKÝ

VEDOUCÍ PRÁCE
SUPERVISOR

Doc. RNDr. PAVEL SMRŽ, Ph.D.

BRNO 2014

Abstrakt

Tato práce se zabývá návrhem a implementací slovníkové aplikace pro mobilní zařízení s Android OS. Klíčovou vlastností aplikace je možnost provádět fulltextové vyhledávání. Toho je dosaženo použitím SQLite databáze s rozšířením Full Text Search. SQLite databáze jsou generovány ze zdrojových lexikálních dat ve formátu Lexical Markup Framework.

Abstract

This thesis deals with design and implementation of dictionary application for mobile device with an Android OS. The key feature of application is ability to perform fulltext search. This is achieved by using the SQLite database with Full Text Search extension. SQLite databases are generated from source lexical data in Lexical Markup Framework format.

Klíčová slova

Slovník, Android, mobilní zařízení, Python, SQLite, FTS, Full Text Search, databáze, LMF, Lexical Markup Framework

Keywords

Dictionary, Android, mobile device, Python, SQLite, FTS, Full Text Search, database, LMF, Lexical Markup Framework

Citace

Tomáš Damborský: Slovník pro mobilní zařízení s OS Android, bakalářská práce, Brno, FIT VUT v Brně, 2014

Slovník pro mobilní zařízení s OS Android

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Doc. RNDr. Pavla Smrže Ph.D.

.....
Tomáš Damborský
31. července 2014

Poděkování

Chtěl bych poděkovat docentu Pavlu Smržovi a inženýru Janu Kouřilovi za konzultace a vedení této práce.

Obsah

Kapitola 1

Úvod

Každý kdo čte tuto práci se jistě setkal se slovníky. S rozšířením informačních technologií se vyhledávání značně zjednodušilo a umožnilo nám najít překlady rychleji a pohodlněji. V současné době je možné mít celé knihovny v kapse v mobilu.

Pro uložení lexikálních dat je potřeba poznačit velké množství rozdílných vlastností. Za tímto účelem vznikl formát Lexical Markup Framework(LMF), který si klade za cíl obsáhnutí všech přirozených jazyků. Pro uchování a zpracování informací je vhodný, ale pro vyhledávání v něm obsažených informací je potřeba převedení do vhodnější podoby.

Cílem práce je v první řadě vytvořit systém pro převod ze zdrojových dat ve formátu LMF do databáze použitelné na systému Android. Je zvolena SQLite pro její snadnou konfiguraci.

Dalším problémem je zobrazení dat na mobilních zařízeních. Neposkytují tak velkou obrazovou plochu a proto je potřeba promyslet jak data získávat.

V kapitole 2 se seznámíme s existujícími slovníkovými formáty. Nastíníme si jejich strukturu s využití. Dále si popíšeme klady a zápory slovníkových aplikací.

V kapitole 4 je popsán návrh systému pro vytvoření SQLite databáze ze zdrojových dat. V sekci 4.3 je na předpokladech požadavků uživatelů a srovnáním s existujícími aplikacemi navržena slovníková aplikace pro vyhledávání a zobrazování.

Výsledná aplikace je testována a porovnána s konkurencí v kapitole 6.2. V závěru je nastíněno její možné pokračování a rozšíření.

Kapitola 2

Teorie

V následující kapitole jsou nejdřív prozkoumány existující slovníkové aplikace, jejich ovládání a formát dat která používají. Dále jsou zmíněny některé formáty uložení slovníkových dat. V další sekci je představen formát LMF, který je v projektu určen jako zdroj slovníkových dat.

2.1 Slovníkové aplikace

Na zařízení se systémem Android je spousta aplikací, ne jinak je tomu v oblasti slovníků. Většina nabízí stejné funkce - vyhledání a zobrazení. Rozdíly vznikají v rozsahu zdrojových dat. Slovníková data nemusí obsahovat jen dvoujazyčné překlady, mohou být i výkladové - objasňující. Mezi slovníky s řadí například WordNet.

2.1.1 HandyLex

Slovníky HandyLex jsou off-line aplikace určené k pohodlnému a rychlému vyhledávání hesel. Přinášejí spoustu užitečných funkcí uživatelům, kteří potřebují mít slovník stále u sebe. Hesla i významy byly vybírány tak, aby splňovaly praktické potřeby uživatelů při každodenní komunikaci a cestování. [4]

Takto se uvádí mobilní aplikace známé firmy Lingea s.r.o, která se věnuje tvorbě jazykových aplikací od roku 1997. Společnost existuje už více než patnáct let a tak by aplikace mohla sloužit jako dobrý příklad možných funkcí.

Je to placená aplikace a její free verzi je možné otestovat na Google Play [1].

Vestavěné funkce HandyLex

- Tvaroslovné hledání - slova se vyhledávají v základním tvaru, např. vyhledání slova *went* zobrazí i překlady slova *go*
- Fonetické hledání - umožní vyhledávat podle výslovnosti

V tabulce 2.1 je uvedeno, že fulltextové vyhledávání umí jenom slovníky pro počítač. Mobilní aplikace je schopná u některých slov zobrazit i použití ve větách. Je tedy možné prohledávat fulltext pomocí jednoho slova. Na obrázku 2.1 je vidět zvýraznění vyhledávaného slova v příkladech. Na levém panelu se seznamem Fulltext jsou zobrazena slova, u kterých se ve příkladech vyskytuje vyhledávané slovo. Vyhledání použití dvou a více slov v

[1] https://play.google.com/store/apps/details?id=cz.lingea.handylex&hl=cs

	Slovník zdarma	Kapesní slovníky Lingea	Slovníky Lingea Plus
Překlad slov	x	x	x
Morfologické hledání	x	x	x
Fulltextové hledání			x
Namluvená výslovnost			x
Počet hesel	8 000	35 000	100 000
Počet významů	14 000	42 000	160 000
Počet překladů	22 000	65 000	270 000

Tabulka 2.1: Popis funkcí slovníků od Lingea s.r.o.

jedné větě není možné. Při zadání dotazu "through customs"do vyhledávacího pole a nutného kliknutí na vyhledat se nezobrazí výsledek "clear sth through customs". Místo toho se zobrazí jen seznam nabízející výběr zobrazení jednoho ze dvou zadaných slov.

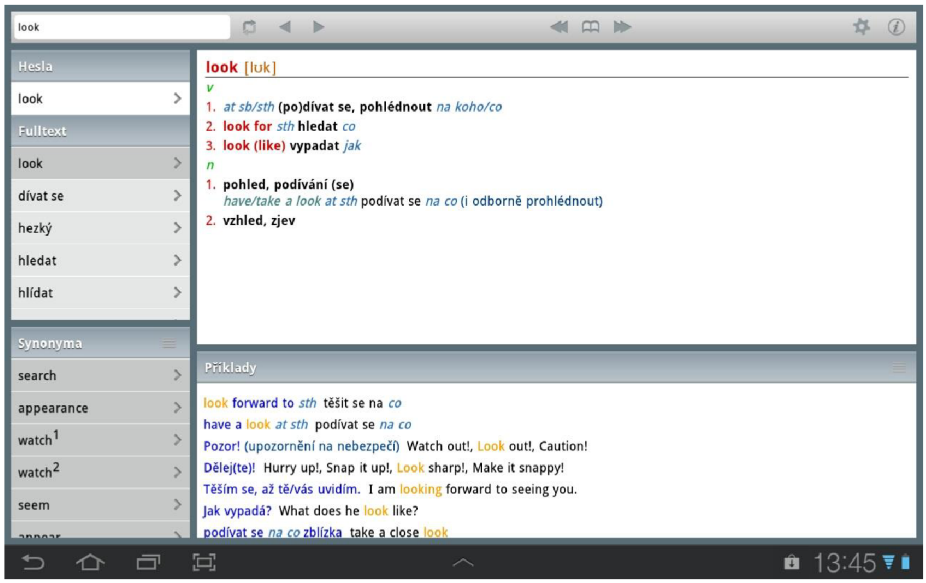
Dále je možné procházet slova ze stejné kategorie nebo výukové lekce jako vyhledané slovo. Příbuzná slova jsou snadnější pro zapamatování.

Aplikace ovšem působí pomalejším dojmem při ovládání. Swipe gesta ze strany na stranu pro procházení slovníku mají nastavenou až moc malou citlivost a tím pádem jsou takřka nepoužitelná. Vyhledávání fulltextu je možné pouze přes vedlejší obrazovky. Vyhledávání jako takové je rychlé.

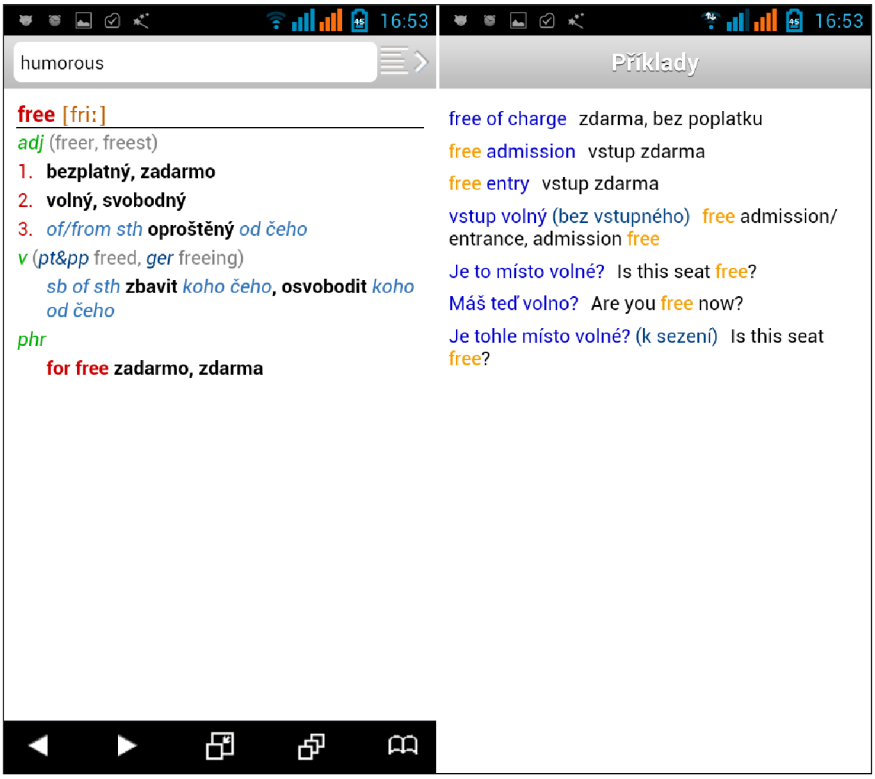
Vyzdvihl bych proměnu aplikace při spuštění na tabletech a dalších zařízeních s větší obrazovou plochou. Do dostupné plochy se rozestaví pohledy, ke kterým bylo potřeba se na mobilu dostat klikáním. Zobrazení vět s užitím hledaného slova je tak okamžité.

Jako ve většině slovníkových aplikací se při zadávání písmen do vyhledávání zobrazují návrhy. Po vybrání návrhu jsou na obrazovce zobrazeny překlady. Kliknutím na kterékoliv z přeložených slov se přepíše aktuální překlad novým překladem slova na které bylo kliknuto. V případě, že se nabízí víc možností nebo není jasné jak by se slovo mělo přeložit.

Slovník využívá proprietární systém uložení dat, kdy není možné ze souborů jednoduše přečíst slova.



Obrázek 2.1: HandyLex 4 spuštěný na tabletu (převzato z [5])



Obrázek 2.2: HandyLex - Zobrazení slova vlevo a ukázek použití vpravo

2.1.2 ColorDict

Aplikace ColorDict není zajímavá jen z hlediska podpory více slovníkových formátů. Mimo podpory formátů Lingvo, Babylon, StarDict, Lingoes a Dictd si zaslouží pozornost i použitím více slovníků zároveň. Mezi slovníky je zahrnuto i vyhledávání na Wikipedii. Aplikace má dobrou podporu módu zobrazení na šířku, kdy se změní rozvržení a zároveň zobrazí volitelně vyhledaná slova nebo historie a detail vybraného slova.

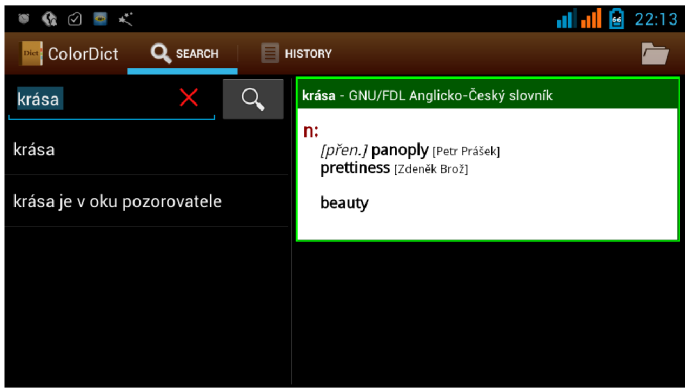
Aplikace podporuje tyto slovníky:

- StarDict
- Wordnet
- Thesaurus

2.1.3 dict.cc dictionary

Tento slovník je zajímavý díky své rozsáhlé databázi anglicko-německých překladů (240MB). Nabízí možnost vybrání dvojice jazyků a vyhledávání buď v každém zvlášť nebo v obou směrech zároveň. Po připojení k internetu nabízí možnost přehrát výslovnost vybraného slova. Díky webovému rozhraní na stránce [2] je možnost přidávat nová slovíčka do databáze, která se průběžně aktualizuje. Data jsou uložena v SQLite databázi a je možné si prohlížet její strukturu. Po provedení příkazu *.schema* a odstranění výpisu vytvářených tabulek rozšíření FTS s indexy, je vidět následující struktura:

[2] http://contribute.dict.cc/



Obrázek 2.3: Rozhraní aplikace ColorDict na mobilu na šířku

```
sqlite> .schema
CREATE TABLE "singlewords" (
        "id" INTEGER PRIMARY KEY NOT NULL,
        "colnum" INTEGER,
        "term" VARCHAR,
        "term4search" VARCHAR);

CREATE TABLE "subjects" (
        "subj_id" INTEGER,
        "lang_id" INTEGER,
        "abbr" VARCHAR,
        "description" VARCHAR);

CREATE INDEX "sw_term4search" on "singlewords" ("term4search" asc);

CREATE VIRTUAL TABLE "main_ft" using fts3(
        "id" INTEGER PRIMARY KEY NOT NULL ,
        "term1" VARCHAR,
        "term2" VARCHAR,
        "sort1" INTEGER,
        "sort2" INTEGER,
        "subj_ids" VARCHAR,
        "entry_type" VARCHAR,
        "vt_usage" INTEGER);

CREATE TABLE android_metadata (locale TEXT);
```

Zvláštní je, že přes použití virtuální tabulky rozšíření FTS, aplikace nenabízí fulltextové vyhledávání. V tabulce *singlewords* se nachází výrazy jak s diakritikou i bez diakritiky. Díky tomu podporuje vyhledávání slov s diakritikou bez jejího zadávání.

2.2 Slovníkové formáty

K reprezentaci slovníkových dat se používá spousta formátů. Popíšeme si několik hojně používaných.

2.2.1 WordNet

WordNet je lexikální databáze anglického jazyka vyvíjená od roku 1985 na Princetonské univerzitě. Sjednocuje synonyma a antonyma základního slova do celků. Těmto celkům s podobným oborem se říká synsety. Svojí strukturou využívání podobných slov je z něj cenný nástroj pro [2]. WordNet je volně dostupný ke stažení.

2.2.2 StartDict

Je schopný uchovávat dvojice *výraz - překlad* přičemž je možné aby překlad odkazoval na další informace, např. audio soubor s jeho výslovností. Slovníky ve formátu StarDict se skládají minimálně ze třech souborů.

Informace o souborech:

- .ifo – textový soubor obsahující informace o verzi slovníku, počtu jeho slov a velikosti .idx souboru
- .idx – binární soubor obsahující seřazený seznam záznamů, přičemž každý záznam obsahuje utf-8 řetězec, offset v .dict souboru a velikostí záznamu v .dict souboru
- .dict – soubor obsahující záznamy ve formátu DICT

Soubory s koncovkou .idx a .dict je možné komprimovat použitím gzip komprese [7].

Formát je využívaný řadou populárních slovníkových aplikací jako GoldenDict, Colordict a další. Tím že formát obsahuje jen indexy lemmat není v něm možné vyhledávat fulltextově.

2.2.3 XDXF

Cílem projektu XML Dictionary Exchange Format (XDXF) je sjednotit všechny existující volně dostupné slovníky a zpřístupnit uživatelům i vývojářům přístup pomocí formátu založeného na XML.[8] Formát má své využití také jako prostředek k převodu mezi ostatními slovníkovými formáty. Jsou mezi nimi populární StarDict, Mova, PtkDic a další. Svým typem uložení dat v XML souboru je podobný formátu LMF, který je standardizovaný a popsaný dále.

2.3 Lexical Markup Format (LMF)

LMF vznikl za účelem standardizace vytváření a využívání lexikálních zdrojů. Dále aby umožnil výměnu dat mezi těmito zdroji a zároveň slučování velkého počtu jednotlivých zdrojů k vytvoření rozšiřitelného zdroje.

Zahrnuje uložení morfologických, syntaktických i sémantických aspektů. Je vhodný pro obsáhnutí zdrojů využívajících jeden, dva nebo i více jazyků zároveň. K popisu vlastností využívá značkovací jazyk XML.

Je navržený s modulární strukturou podléhající standardu UML, kdy základním stavebním kamenem je Core package a na něj navazují rozšiřující Extensions.

2.3.1 Core package

Hlavní součásti LMF na kterém staví ostatní rozšíření jsou na obrázku 2.4. Je neoddělitelnou součástí LMF a každé rozšíření ho musí používat. Popisuje hierarchii jednoho záznamu - `Lexical Entry`.

Součásti jednoho záznamu jsou následující:

- Lexical Resource - třída označující celý zdroj, může obsahovat jeden nebo více lexikonů, ale v dokumentu se smí objevit pouze jednou
- Global Information - má povinný atribut *languageCoding* definující jazykové kódování celého dokumentu
- Lexical Entry - značí lexém daného jazyka
- Form - je abstraktní třída zastupující lexém nebo morfologickou variantu lexému
- Lemma - obsahuje vyhledávané slovo a je to podtřída třídy Form
- WordForm - určuje tvar uvedeného Lemma
- Sense - třída reprezentující význam lexikálního záznamu, povoluje podtřídy
- Definition - obsahuje podrobný popis Sense
- Statement - doplňuje popis Definition

2.3.2 LMF Extensions

V zobrazení síti na obrázku 2.5 jsou vidět vztahy mezi jednotlivými rozšířeními. Všechny využívají základní části a přidávají k nim svoje vlastní lexikální informace.

2.3.3 Použité tagy v dokumentu

Dále si popíšeme značky které obsahuje zdrojový LMF dokument

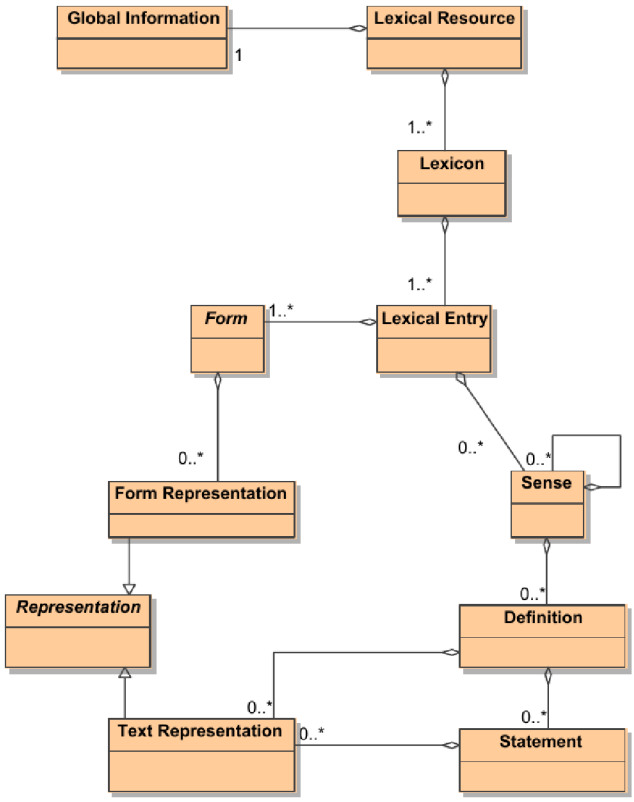
Začátek dokumentu vždy označuje

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE LexicalResource SYSTEM "http://www.tagmatica.fr/lmf/DTD_LMF_REV_16.dtd">
<LexicalResource dtdVersion='16'>
       <GlobalInformation>
              <feat att='languageCoding' val='ISO 639-3'/>
</GlobalInformation>
<Lexicon>
```

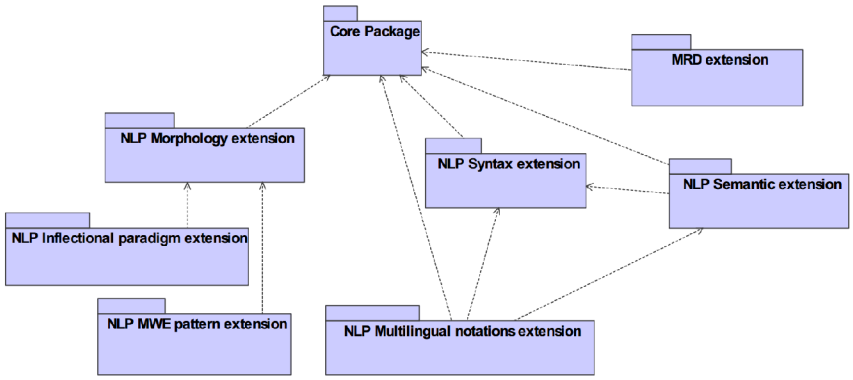
ISO 639-3 definuje standard pro reprezentaci názvů jazyků v třípísmenném kódu. Jeho cílem je obsáhnout všechny známé jazyky.

Dále si uvedeme jak vypadá záznam jednoho slova v anglicko-českém zdroji dat.

```
<LexicalEntry id='e17797g'>
  <feat att='partOfSpeech' val='noun'/>
  <Lemma>
    <feat att='writtenForm' val='carabao'/>
```



Obrázek 2.4: LMF Core, zobrazení základních prvků(převzato z [10])



Obrázek 2.5: LMF Extensions, vztahy mezi rozšiřujícími balíčky (převzato z [10])

```
</Lemma>
<WordForm>
     <feat att='phoneticForm' val=',kHrE&apos;beIEU'/>
     <feat att='writtenForm' val='carabao'/>
</WordForm>
<Sense>
```

```
    <feat att='senseNumber' val='1'/>
    <Equivalent>
         <feat att='language' val='ces'/>
         <feat att='writtenForm' val='buvol indický/vodní'/>
    </Equivalent>
    <Equivalent>
         <feat att='language' val='lat'/>
         <feat att='writtenForm' val='Bubalus arnee'/>
    </Equivalent>
    <SubjectField>
         <feat att='label' val='zool.'/>
    </SubjectField>
  </Sense>
</LexicalEntry>
```

Pro popis vlastností jednotlivých tříd je popsán ve dvojicích *att* - *val*. Atributy vlastností jsou čerpány z Data Category Registry (DCR)[3]

V DCR je možné najít použité prvky, ale ne u všech je detailní popis. Například v tagu *phoneticForm* se vyskytuje výslovnost, ale v registru se o způsobu zápisu více nenalézá.

[3] https://catalog.clarin.eu/isocat/interface/index.html

Kapitola 3

Použité technologie

Pro vytvoření navrhnutého V následující kapitole jsou popsány nástroje použité pro vývoj aplikace.

3.1 Vývojové prostředí Eclipse

Eclipse cílí na vývoj aplikací v jazyce Java. Existuje pro něj množství pluginů. Jedním z nich je Android Developer Tools (ADT), který zprostředkovává rozhraní pro používání Android Software Development Kitu (SDK).

3.2 Android OS

Android je v současné době nejrozšířenější mobilní platformou. Jeho pole působnosti se stále rozšiřuje. První a stále primární zařízení je mobilní telefon. Pro vývoj jsou uvolněny nástroje v Android Software Development Kitu (SDK).

3.2.1 Programovací jazyk

Primárním programovacím jazykem pro vývoj aplikací je jazyk Java. Jaká verze a tak

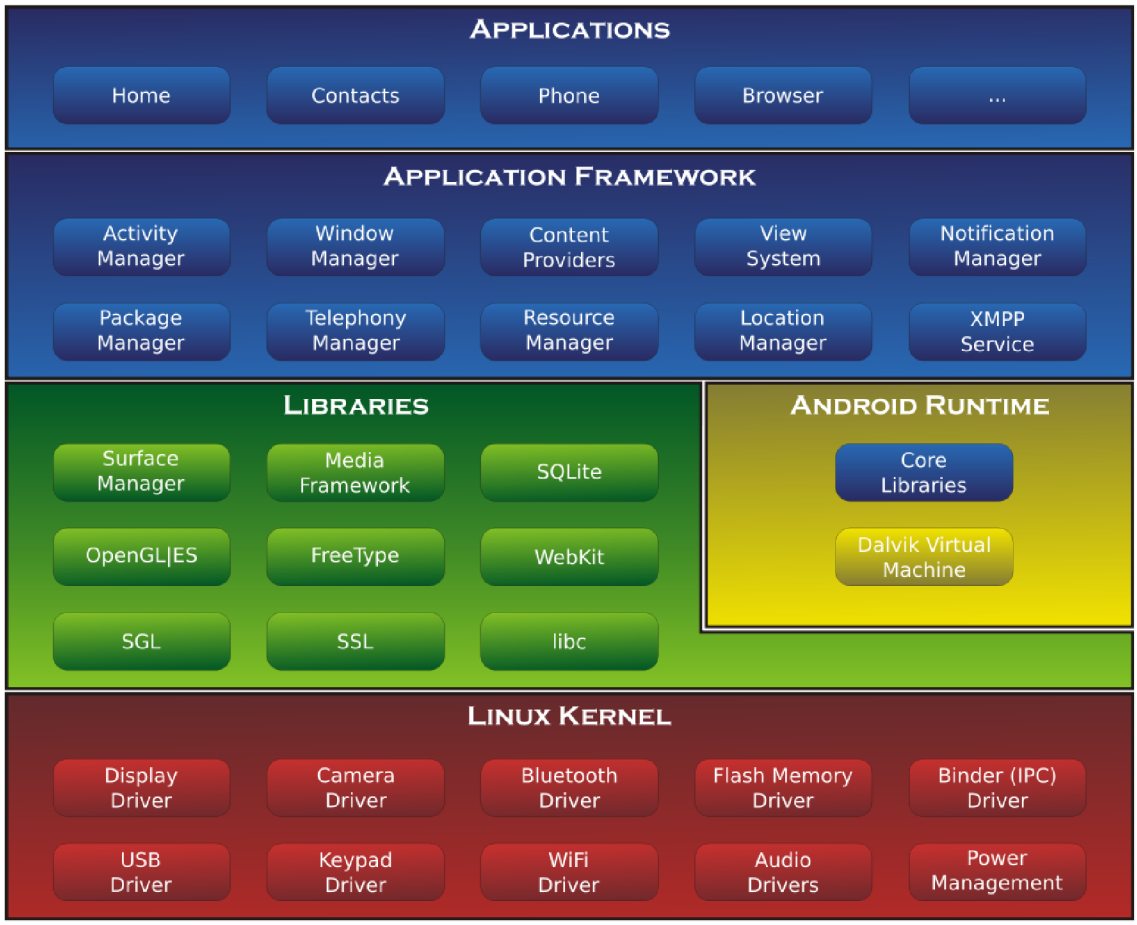
Kód napsaný v C, C++ a dalších jazycích může být zkompilován do ARM, MIPS nebo x86 nativního kódu a nainstalován použitím Android Native Development Kit (NDK). https://developer.android.com/tools/sdk/ndk/index.html

3.2.2 Android SDK

Součástí Android Software Development Kitu (SDK) je sada nástrojů pro vývoj aplikací. Je možné jej stáhnout jako součást ADT balíčku včetně vývojového prostředí Eclipse.

Součásti SDK Bundle jsou

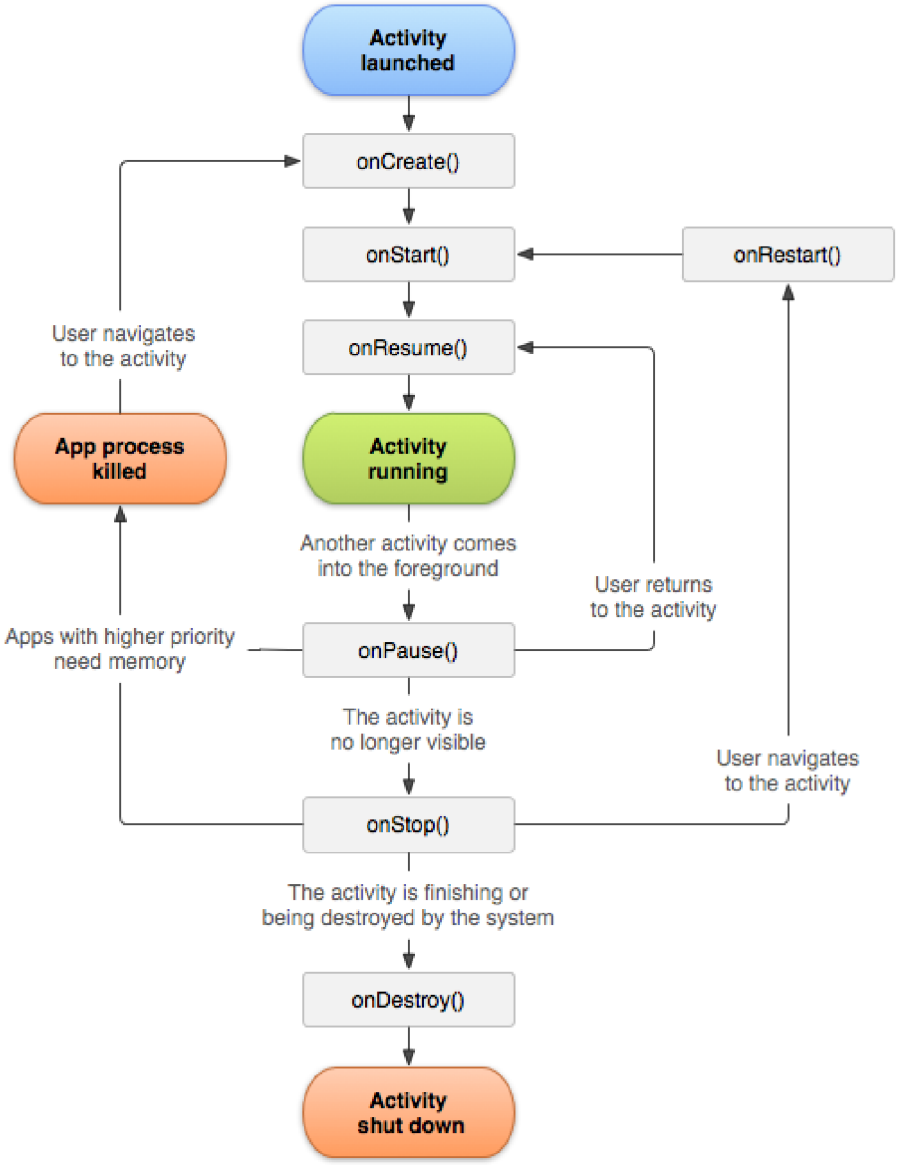
- Eclipse + ADT plugin
- Android SDK Tools
- Android Platform-tools
- Verze platformy Android
- Verze systémového obrazu Androidu pro emulátor



Obrázek 3.1: (převzato z http://commons.wikimedia.org/wiki/File:Android-System-Architecture.svg)

Komponenty, z kterých se skládá každá aplikace:

- Activity (Aktivity) - je základní stavební součátka aplikace, která zprostředkovává obrazovku s kterou uživatel aplikace může pracovat, aplikace po startu zobrazí hlavní aktivity, která je nadefinovaná v souboru `AndroidManifest.xml`
- View [ViewGroup, TextView, ...] je základní zobrazovací částí aplikace, jsou na něm postaveny. Je základní třídou pro `widgety`, které jsou použity pro vytváření prvků uživatelského rozhraní (tlačítka, textová pole, atd.)
- Service (Služby) Služby jsou procesy běžící na pozadí, které nemají grafické rozhranní.
- Content Provider (Poskytovatel obsahu) Poskytovatelé obsdahu poskytují úroveň abstrakce pro jakákoliv data uložená v zařízení, ke které lze přistupovat z více aplikací. Umožňuje zpřístupnění dat své aplikace i ostatním aplikacím.
- Intent (Záměr) Záměry jsou systémové zprávy, které kolují v zařízení a upozorňují aplikace na různé události, např. příchozí volání, vložení SD karty. Oznamuje tedy změny stavu hardwaru, příchozí data a další spoustu událostí. Systém pomocí intentů oznamuje události, například příchozí volání, na které mohou aplikace reagovat -



Obrázek 3.2: Activity lifecycle (převzato z [3])

v aplikaci na přehrávání hudby se příchozím intentem oznamujícím hovor pozastaví přehrávání skladby a podobně

Kořenový adresář aplikace obsahuje tyto položky: [9]

- AndroidManifest.xml - soubor XML popisující aplikaci, použité komponenty, potřebná oprávnění, Záměry které umí zpracovat, verze SDK na které je určena služby, poskytovatele obsahu a další [1]
- bin/ – složka se zkompilovanou aplikací
- gen/ – vygenerovaný kód vytvoření při sestavování aplikace

[1] http://developer.android.com/guide/topics/manifest/manifest-intro.html

- libs/ – knihovny, které aplikace používá
- res/ – obsahuje ikony, rozvržení rozhraní, řetězce a další
- src/ – zdrojové soubory aplikace
- assets/ – obsahuje soubory, které aplikace může využít po spuštění

K debugování aplikace při vývoji je možné použít slouží nástroj monkey, který provádí náhodné nebo definované činností. Je uvedena ukázka jeho použití na balík `cz.vut.fit.DDict`.

```
$ adb shell monkey -p cz.vut.fit.DDict -v 500
```

3.3 SQLite

SQLite je relační databázový systém obsažený v malé knihovně. Knihovna je vytvořena v jazyce C. Rozhraní pro práci s SQLite je zpřístupněno v mnoha programovacích jazycích. [1] Je k dispozici i v jazyce Python a v SDK Androidu.

Pro použití v zařízení systému Android je vhodná z více důvodů:

- není potřeba žádná konfigurace
- není potřeba server
- splňuje kritéria ACID

Používá tyto datové typy :

- NULL
- INTEGER
- REAL
- TEXT
- BLOB

Vytvořené databáze je možné prohlížet v prohlížeči a zároveň editoru databází sqlitebrowser[2]. Jeho nevýhodou je chybějící podpora rozšíření FTS, ale ta se projeví jen při provádění dotazů nad virtuální tabulkou. V zobrazení to nevadí.

Pro urychlení vyhledávání v textu bylo pro SQLite vytvořeno rozšíření FTS [6].

3.3.1 Vytváření FTS tabulek

Při vytváření tabulek FTS je potřeba počítat s tím, že indexy se vytváří nad celou tabulkou. Vyhledávání v určitém sloupci tabulky je pak možno stanovit v příkazu `MATCH` pomocí zápisu sloupec:heslo ale je výhodnější oddělit použití FTS indexu opravdu jen na informace ve kterých bude nutné vyhledávat fulltextově. V případě umístění veškerého obsahu do jedné tabulky by se vytvořily indexy na všechny termíny.

Vytvořením virtuální tabulky použitím FTS rozšíření se vytvoří pět stínových tabulek s informacemi a indexy. Je reprezentována přeti stínovými tabulkami s přídavkem content, segdir, segments, stat, docsize. [6]

Virtuální tabulka se vytvoří příkazem

[2] http://sqlitebrowser.org/

```
CREATE VIRTUAL TABLE enrondata1 USING fts4(obsah TEXT); /* FTS4 tabulka */
CREATE TABLE enrondata2 (obsah TEXT); /* Obyčejná tabulka */
```

Vyhledávání v obsáhlém textu je o mnoho rychlejší než pomocí výrazu `%LIKE`

```
SELECT count(*) FROM enrondata1 WHERE content MATCH 'linux'; /* 0.03 seconds */
SELECT count(*) FROM enrondata2 WHERE content LIKE '%linux%'; /* 22.5 seconds */
```

Ukázky jsou převzaty z [6].

3.4 Python

Skriptovací jazyk Python se používá pro mnoho činností a je vhodný i pro zpracování textu. Je široce rozšířený a je pro něj vytvořeno mnoho knihoven. Mimo jiné má zabudovanou knihovnu pro práci s SQLite. Tu stačí importovat, popis práce s ní je k nalezení v dokumentaci[3].

Pro zpracování XML souborů je možné nabízí jednoduché API jejíž použití je také blíže popsáno v dokumentaci [4].

[3] https://docs.python.org/3/

[4] https://docs.python.org/3.3/library/xml.etree.elementtree.html

Kapitola 4

Návrh řešení

V této kapitole jsou stanoveny cíle řešeného systému. V další části je Detailnějším rozebráním vlastností existujících slovníkových aplikací v kapitole 2 byly nalezeny výhody i nevýhody daných řešení.

4.1 Cíle

V první řadě je cílem vytvořit systém který převede data ve formátu LMF do podoby vhodné pro zobrazení. Pobráním dostupných metod se jeví nejschůdnější řešení převedením do databáze SQLite.

Cílem aplikace bude zpřístupnit uživateli rozhraní pro převod slovníkových dat ve formátu LMF do databáze použitelné v Android aplikaci.

Jako referenční aplikaci, jejíž funkcionalitě bych se chtěl přiblížit je kombinace rozmanitosti funkcí slovníku HandyLex4 s jednoduchostí a plynulostí ovládání aplikace ColorDict, kde se mi líbí i podpora pro zobrazení na šířku.

4.2 Vytvoření databáze z LMF

Zdrojová data je potřeba převést do podoby vhodné pro použití na mobilním zařízení. V kapitole 2 byly popsána existující řešení. Mezi nejschůdnější řešení považuji převedení na relační databázi, která byla vytvořena v aplikaci dict.cc 2.1.3.

U každého slova se vyskytuje pár opakujících se vlastností. Z důvodu úspory paměti jsem se rozhodl databázi rozčlenit na více malých tabulek s výčtem těchto opakujících se vlastností (například z tabulky `lemmaTypes` se získávají slovní druhy)

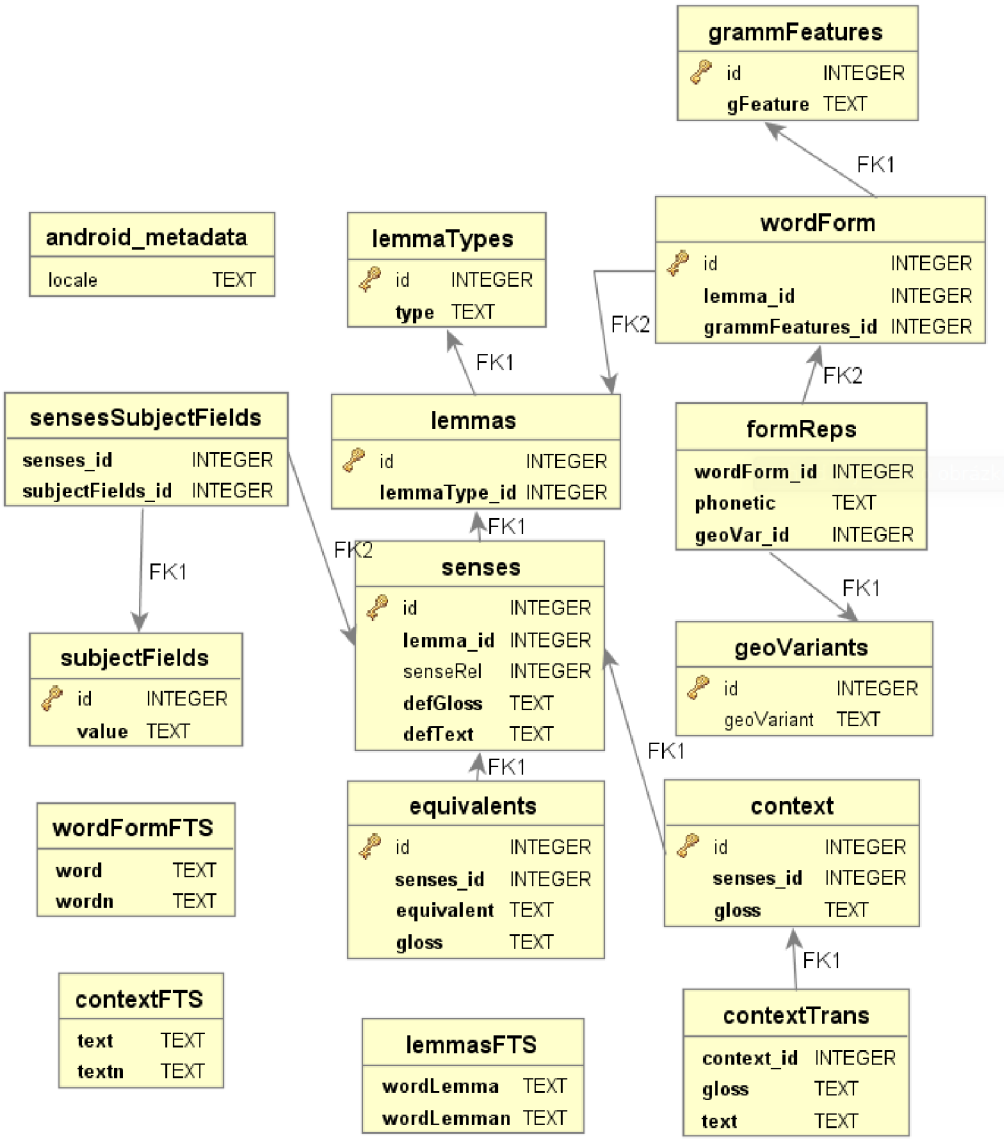
např. slovní druh - podstatné jméno, přídavné jméno a další vlastnosti které se často opakují. Tímto způsobem předcházíme duplicitě dat

Ze zdrojových dat je nejdřív potřeba získat jednotlivé informace. K tomu se vytvoří SAX parser XML dat schopný zpracovat značky použité ve zdrojových datech.

Většina tabulek bude vytvořených normálním způsobem. Tabulky obsahující informace potřebné pro fulltextové vyhledávání je potřeba indexovat. Je tím myšleno

Splňuje požadavky na jednoduchou konfiguraci i převod všech důležitých informací záležejích na návrhu databáze.

Pro spojování normálních tabulek se na všech cizích klíčích vytvoří indexy, aby se urychlilo jejich spojování.



Obrázek 4.1: Návrh struktury databáze

Kvůli vyhledávání slov s diakritikou bude potřeba vytvořit u fulltextových tabulek dva sloupce. Jeden s diakritikou, druhý bez. Při vyhledávání v aplikaci se poté bude zadávaný text převádět na text bez diakritiky, vyhledávání proběhne a najde výraz bez diakritiky, ale vrácená data se vyberou ze sloupce s diakritikou. Struktura výsledné databáze je znázorňena na obrázku 4.1. Uvedeným návrhem databáze je dosaženo uchování téměř všech dat zdrojového LMF souboru.

Standardní zápis výslovnosti je podle formátu IPA (International Phonetic Aphabet [1]) Je to akademický standard vytvořený Mezinárodní Fonetickou Organizací (International Phonetic Association). X-Sampa and CXS are ways of writing IPA in plain ASCII messages, as used in mails or newsgroups. The few modifications CXS introduces are marked in bold and in green color and are additionally mentioned in a note in that section.

[1] http://www.internationalphoneticalphabet.org/

4.3 Návrh aplikace slovníku

Při návrhu slovníkové aplikace je potřeba zajistit tyto hlavní činnosti:

- překopírování slovníkových databází do interní paměti
- přepínání aktuální slovníkové databáze
- vyhledávání v databázi
- zobrazení výsledků vyhledávání

Funkce vylepšující základní funkčnost:

- použití fulltextového vyhledávání
- návrhy vyhledávání při zadávání do vyhledávacího pole
- zobrazení detailu slova na vedlejší obrazovce
- vyhledávání při kliknutí na libovolné z vyhledaných slov
- ukládání nastavení
- historie vyhledávání

Kopírování slovníkové databáze musí proběhnout při prvním spuštění aplikace, protože samotný slovník žádnou databázi neobsahuje. Proto se prohledá interní paměť.

Přepínání aktuální databáze bude řešeno tlačítkem blízko pole pro zadávání vyhledávacího textu. Klasické umístění je v pravém horním rohu.

Pro výběh mezi vyhledáváním slov a fulltextu bude použito dvou přepínacích tlačítek pod textem pro vyhledávání.

Po kliknutí řádek s vyhledaným slovem nebo frází se pohled přesune na obrazovku vpravo. Mezi těmito pohledy je možné rychle přepínat pomocí táhnutí prstu po obrazovce ze strany na stranu.

4.3.1 Uživatelské rozhraní

V grafickém návrhu je potřeba dbát na navození dobrého prvního dojmu uživatelů. Je potřeba mít dostupné všechny hlavní funkce. Ovládání by mělo být intuitivní.

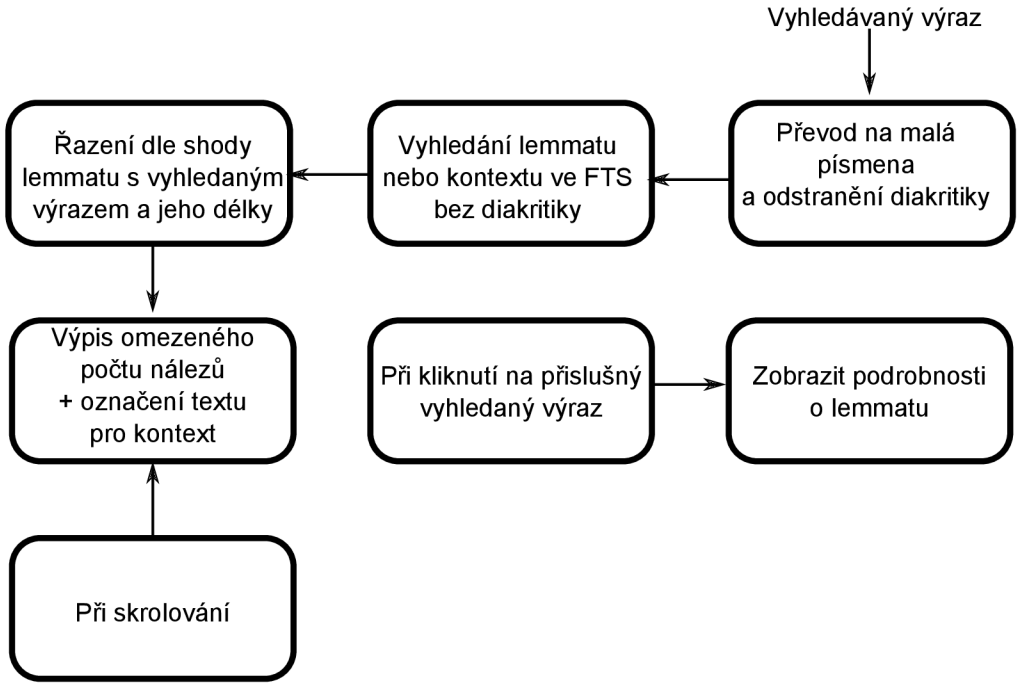
Kapitola 5

Implementace

V následující kapitole si popíšeme postupy, které vedly k vytvoření databáze a jejímu zobrazení na mobilním zařízení systému Android.

Práce probíhaly na zařízení s procesorem Intel Core i5-4200M, pamětí 8GB a v prostředí Linuxu Sabayon s kernelem 3.15.0 a telefonu Prestigio PAP4500T s verzí Androidu 4.1.1.

Postup implementace je následující. Vstupem jsou lmf soubory. Každý soubor představuje jednu slovníkovou databázi. Tyto soubory se převádí pomocí python3 skriptu do jednotlivých SQLite3 databází. Dále se za pomocí editoru Eclipse verze Juno 4.2.1.v20130118 vytvořila aplikace s názvem DDict (minimální požadavky na sdk jsou od verze 14 až po verzi 21). Dále se práce zaměřuje na samotnou implementaci aplikace, která se skládá z SQLite3 databází, jenotlivých widgetů Androidu, volání dotazů nad databází, výpis nálezů do rozhraní, užití Fultextového vyhledávání, kopírování a výběr databází a dalších.



Obrázek 5.1: Výběr hlavních z funkcí aplikace.

5.1 Vytvoření databáze

Pro převod LMF souborů do SQLite databáze byl vytvořen skript v Python 3 s názvem `xmltosqlite.py`. Tento skript přečte data z LMF souborů, vytvoří databáze a tabulky a následně vloží přečtená data do již vytvořených databází.

V převodním skriptu je využita Třída `XMLParser` ze standardní knihovny `xml.etree.ElementTree`, která slouží pro parsování xml souborů.

Protože pro vyhledávání ve Fulltextových tabulkách je použito vyhledávání bez diakritiky, musí se do každé FTS dabulky uložit 2 sloupce. Jeden bez diakritiky a druhý s diakritikou. K samotnému odstranění diakritiky byla použita metoda `string.translate(s, table[, deletechars])`.

Převodní řetezce pro odstranění diakritiky

- ěščřžýáíéúůďťňĚŠČŘŽÝÁÍÉÚŮĎŤŇ
- escrzyaieuudtnESCRZYAIEUUDTN

Před samotným použitím XMLParseru je potřeba implementovat vlastní metody pro startovací a ukončující tagy. Protože v LMF souborech není použit textový obsah, nemá smysl implementovat metodu `XMLParser.data`. Pro implementaci třídy je použita nová třída s názvem `MyXMLParser`. Jejím výstupem jsou `csv` soubory, které korespondují s názvy tabulek. Dále se tyto soubory ukládají do dočasného adresáře a importují do databáze specifikované názvem, který se zadává jako výstupní parametr do skriptu `xmltosqlite.py`. Pro práci s databázemi v `Python 3` je použita knihovna `sqlite3`.

Zde je příklad spuštění skriptu pro převod LMF do SQLite3:

```
xmltosqlite.py --input="czen-lmf.xml" --output="encz.db" --from=ces --to=eng
```

5.2 Aplikace slovník DDict

V této kapitole bude popsána implementace aplikace DDict pro operační systém Android pro sdk 14 a vyšší.

V této aplikaci bylo aplikováno fulltextové vyhledávání v nyní již vytvořených databázích. Fulltextové vyhledávání je velice efektivní, a proto nemusí uživatel dlouho čekat na vypsání výsledků.

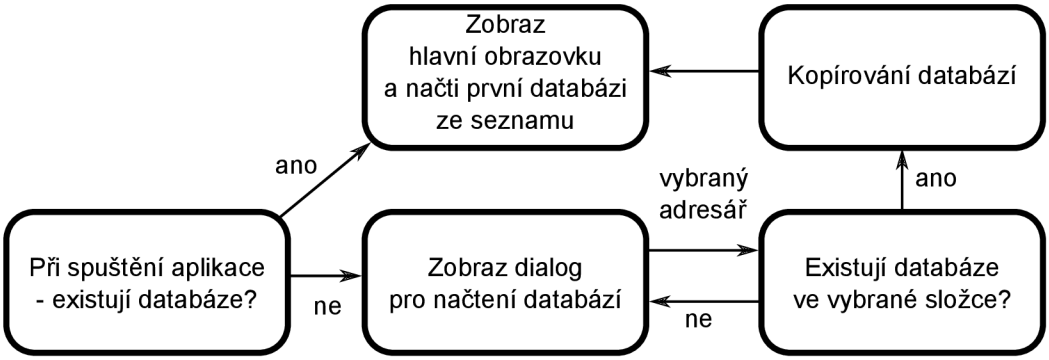
Během vyhledávání je použito asynchronních vláken (běžících na pozadí), která zabraňují krátkému zablokování aplikace. Pokud je nálezených výsledků i tak příliš mnoho, výpis netrvá dlouho právě díky dynamickému načítaní obsahu spuštěnému až při skrolování dosažení konce aktuálně načtených výsledků. Tato funkce nebyla popsána v návrhu. Vznikla na základě potřebu řešit situaci při načtení velkého počtu záznamů.

Nejprve však je potřeba popsat implementaci kopírování databází, která se provádí z pravidla při prvním spuštění aplikace.

5.2.1 První spuštění

Při prvním spuštění aplikace nejsou k dispozici zdrojová data (slovníkové databáze) v domovském adresáři `/data/data/cz.vut.fit.ddict/databases/`. Proto se v hlavní aktivitě `DDictActivity` kontroluje zda jsou databáze v uvedeném adresáři a pokud ne, spustí se

vedlejší aktivita `StartUpMsg` ve které je pouze text s instrukcemi a tlačítko OK. Po opuštění vedlejší aktivity se zobrazí dialog, ve které je uživatel nucen vybrat spravný adresář s potřebnými databázemi.



Obrázek 5.2: Diagram běhu aplikace při nenalezení databází.

Protože v rozhraní Android SDK není implementován jednoduchý univerzální prohlížeč složek, bylo použito jednoduchý dialogu se seznamem složek, který lze nálézt zde: [1]. Ten je nastaven tak, aby po vybrání a potvrzení zkontroloval, jestli obsahuje potřebné databáze. Pokud není vybrán správný adresář, zobrazí se dialog s chybovým hlášením. V opačném případě pokud je potvrzena přítomnost těchto databází, spustí se nová úloha na pozadí, která zkopíruje databáze do domovského adresáře. Pro implementaci této úlohy je použita třída `AsyncTask`. Po dobu kopírování je zobrazen dialog s hlášením o stavu aplikace a uživateli není dovoleno provádět žádné další akce. Po dokončení je pomocí metody `ProgressDialog.dismiss()` odstraněn dialog o stavu kopírování a aplikace je připravena k použití.

5.2.2 Hlavní aktivita DDictActivity

Hlavní activita `DDictActivity` je potomkem `ActionBarActivity`, která zprostředkovává komunikaci s `ActionBar` tlačítky a jeho menu. `DDictActivity` obsahuje jeden `DrawerLayout`, který spravuje vysunovací menu nalezající se na levé straně obrazu. Tento widget se skládá ze dvou dalších widgetů. Jeden z nich je `ViewPager` a druhý je `ListView`.

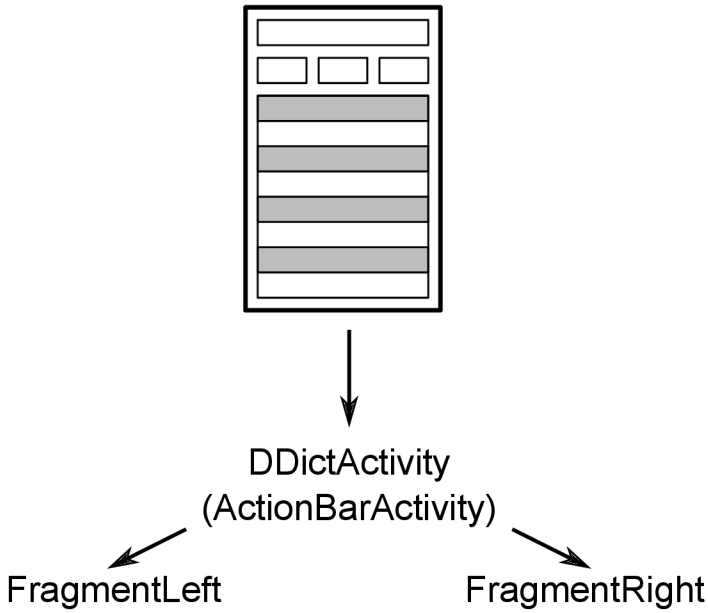
`ViewPager` zprostředkovává dynamické přepínání fragmentů pomocí pohybu prstu po obrazovce. O naplnění a správu fragmentů se v tomto widgetu stará třída `MyPagerAdapter`, která je potomkem `FragmentPagerAdapter`. Zde bylo potřeba implementovat metody `Fragment getItem(int pos)` (pro vytvoření a vrácení instance určitého fragmentu) a `int getCount()` (pro vrácení celkového počtu použitých fragmentů).

Celkově byly použity 2 `Fragmenty`. Pro jednoduchou orientaci se jmenují `FragmentLeft` a `FragmentRight`. `FragmentLeft` slouží k hledání a zobrazení nalezených výsledků a je to také hlavní fragment, který se zobrazí při spuštění. `FragmentRight` pak zobrazuje detailnější informace o lemmatu, který se zobrazí při kliknutí na dané lemma. Použití fragmentů má tu výhodu, že lze plynule přepínat mezi výsledky vyhledávání a detailním výpisem lemmatu.

Oba tyto fragmenty obsahují `ScrollView`, který obsahuje jeden `LinearLayout`, do kterého se vykreslují jednotlivá nalezená data.

`ListView` nalézající se uvnitř `DrawerLayoutu` nám slouží jako rozhraní pro jeho menu. O naplnění a správu tohoto menu (`ListView`) se stará `SideMenuAdapter`, což je zděděná

[1] http://www.codeproject.com/Articles/547636/Android-Ready-to-use-simple-directory-chooser-dial



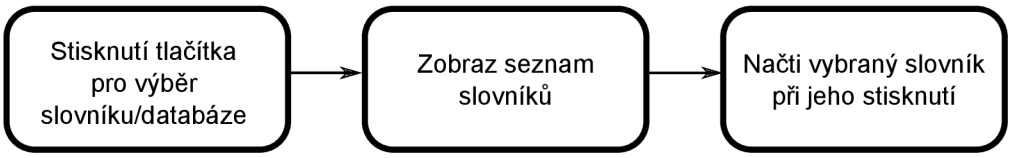
Obrázek 5.3: Rozdělení hlavní aktivity na 2 fragmenty.

třída od třídy `BaseAdapter`.

5.2.3 Vybírání aktuální databáze slovníku

Vybírání databázi se provádí přes menu `DrawerLayoutu`, které je implementovaté přes `ListView`. Toto menu se zobrazí buď kliknutím na ikonu aplikace nebo na tlačítko `ActionBaru`, které nese jméno aktuální databáze. V tom měnu je jedno tlačítko které zobrazuje aktuálně vybraný slovník.

Samotný výběr se provádí přes `OnItemClickListener` (implemetuje se u `ListView`), kde při kliknutí na položku seznamu v menu se přepne databáze, smažou se výsledky hledání a schová se menu. Dále se označí barevným pozadím vybraná položka a změní se tlačítko `ActionBaru` s názvem nového slovníku. Změna databáze se provádí zkrze `SQLiteHandler` a jeho metodu `setDb`.



Obrázek 5.4: Zjednodušený diagram změny slovníku databáze.

5.2.4 Vyhledávání

Vyhledávání se dělí na 2 části. Vyhledávání lemmat, neboli slov, a vyhledávání fulltextové v kontextu databáze.

Hledaný výraz se zadává do textového pole `AutoCompleteTextView`. Tento widget má tu výhodu, že při psaní zobrazuje napovídaná slova/věty (v závisloti na vybraném způsobu hledání), která se nachází v databázi. Aby `AutoCompleteTextView` správně pracoval, musí

se na něm nasatavit adapter, který naplňuje `ListView`, které se uvnitř něj schovává. V tomto případě byl použit
`SimpleCursorAdapter`, u kterého se musí nastavit filtrování dat a to v závisloti jestli se hledají lemmata nebo kontext (fulltext). Dynamické přepínaní mezi těmito dvěma způsoby hledání se provádí metodou `SimpleCursorAdapter.changeCursorAndColumns(...)`.

Filtrování je v podstatě jednoduchý dotaz na databázi.

```
SELECT MIN(docid) AS _id, wordLemma
FROM lemmasFTS
WHERE wordLemman MATCH 'auto'
GROUP BY wordLemman /*odstraneni duplicit*/
ORDER BY LENGTH(wordLemman) ASC
```

Listing 5.1: Příklad hledání (filtrování) lemmat v databázi pro `AutoCompleteTextView`.

Dále je na widgetu `AutoCompleteTextView` nastaveno naslouchání na položku při kliknutí, aby se text vepsal nejenom do textového pole, ale i automaticky začalo vyhledávat. To nám zajišťuje naslouchač `AutoCompleteTextView.OnItemClickListener`, který naslouchá a při kliknutí provede námi naprogramovanou akci, tedy odeslaní dat k hledání.

Před samotným hledáním, se převede hledaný výraz na malá písmena a odstraní diakritika (`public static String SQLiteOutput.cutDiacritics (String str)`). Dále, aby nám aplikace nezamrzla, se vyhledávání provádí na pozadí pomocí třídy `AsyncTask`. V průběhu zpracování se zobrazuje dialog se zprávou. Běh na pozadí zprostředkovává metoda `AsyncTask.doInBackground`, ve které se spustí dotaz na databázi. O zpracování a komunikaci s databází se stará třída `SQLiteHelper` (potomek `SQLiteOpenHelper`), konkrétně třída `SQLiteHelper.findLemmas(String s)` nebo `SQLiteHelper.findFulltext(String s)` (použití metody opět závisí na výběru způsobu hledání). Výstupem obou metod je třída `SQLiteOutput`, která se stará o zpracování dat z databáze, jeho zformátování a výpis.

```
SELECT l.id, lFTS.wordLemma, lt.type,
CASE WHEN e.equivalent is null THEN
    ee.id ELSE e.id
END AS eid,
CASE WHEN e.equivalent is null THEN
    ee.equivalent ELSE e.equivalent
END AS eq
FROM
(
    SELECT lFTS.docid, lFTS.wordLemma
    FROM lemmasFTS lFTS
    WHERE lFTS.wordLemman MATCH ?
) AS lFTS
LEFT JOIN lemmas l ON l.id = lFTS.docid
JOIN lemmaTypes lt ON l.lemmaType_id = lt.id
JOIN senses s ON s.lemma_id = l.id
LEFT JOIN equivalents e ON e.senses_id = s.id
LEFT JOIN senses ss ON ss.id = s.senseRel
LEFT JOIN equivalents ee ON ee.senses_id = ss.id
```

Listing 5.2: Příklad vyhledávání kotextu.

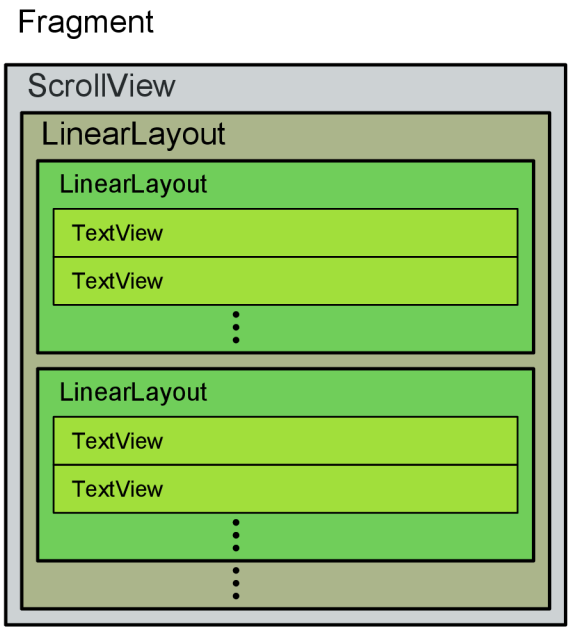
5.2.5 Řazení a výpis

K formátování, řazení a výpisu dat z databáze slouží třída `SQLiteOutput`. Uvnitř této třídy se nacházejí `HashMapy` a další `Collections` pro ukládání vyfiltrovaných výsledků.

Tyto výsledky hledání se dále řadí určitým způsobem. Nejprve se vyberou výsledky, které mají shodné lemma s vyhledávaným výrazem a teprve potom následuje zbytek. Vzniknou tak 2 nezávislé seznamy, konkrétně `LinkedHashMap` (díky použití linků bude zaručeno pořadí položek), které se dále setřídí dle délky lemmatu od nejkratšího po nejdelší. Řazení provádí metoda `SQLiteOutput.sortLemmasBy(String sortBy)`.

Samotný výpis dat provádí metody `SQLiteOutput.fillLeftFragment(...)` a `SQLiteOutput.fillRightFragment(...)`. Jak název napovídá, `SQLiteOutput.fillLeftFragment(...)` vypisuje data do levého fragmentu a `SQLiteOutput.fillRightFragment(...)` do pravého fragmentu. Vstupem těchto metod je počet výpisů a `LinearLayout`, do kterého se data mají vypsat.

V obou případech se pro kontextová data provádí zvýrazňování textu dle hledaného výrazu. Před samotným zvýrazňováním se musí hledaný výraz rozdělit na tokeny (slova), pro které se jednotlivě prohledává text a při shodě se označí jeto shodná část. Změna barvy textu se provádí pomocí `SpannableString` třídy. Pro rozdělení výrazu na tokeny se používá metoda `SQLiteOutput.findlemmaToTokens(String findlemma)` a na vyhledání a zvýraznění textu pak metoda `SQLiteOutput.highlightLemmas(List<String> tokens, String text)`.



Obrázek 5.5: Jednotlivé `TextView` jsou řádky s barevným kolečkem na levé straně a jednotlivé `LinearLayout` znázorňují jednu položku lemmatu.

V pravém fragmentu se pak navíc pro každé slovo vytváří klikací span, tzn., že při kliknutí na jednotlivá slova se pomocí `ClickableSpan` zachytí tato událost a prove se námi naprogramovaná činnost. V tomto případě se spustí dialog s dotazem, zda uživatel chce provést prohledávání daného slova. Pokud ano, stiskem tlačíka `OK` se tato činnost provede. Pro opačný názor je zde tlačítko `Cancel`.

5.2.6 Zobrazení výsledků hledání

Jednotlivý nález záznamu se vypisuje do widgetu `LinearLayout` (co záznam to nový layout). Každý takový `LinearLayout` se vkládá do rodičovského `LinearLayout`, který je obalen ve `ScrollView`. Ten nám možňuje skrolovat výsledky, které se nevlezou na obrazovku, tak jak ne popsáno na obr. 5.5.

Protože tento kořenový `ScrollView` nemá přístupný `MyScrollView.OnScrollListener`, bylo potřeba vytvořit si vlastní. Ten nám umožnuje zachytávat událost při skrolování a vypočítávat kdy se zaskrolovalo nakonec widgetu. Pro tento účel byla vytvořena metoda `MyScrollView.isBottomReached (int offset)`, která vrací `true`, pokud se dosáhlo konce widgetu o daný offset. Můžeme tedy zjisti jestli jsme nejen dosáhli už konce, ale i ho téměř dosáhli. V kombinaci s `MyScrollView.OnScrollListener` a `MyScrollView.isBottomReached (int offset)` lze načítat postupně výsledky hledání a nečekat tak na zdlouhavé vypisování dat v závisloti na množství nalezených záznamů.

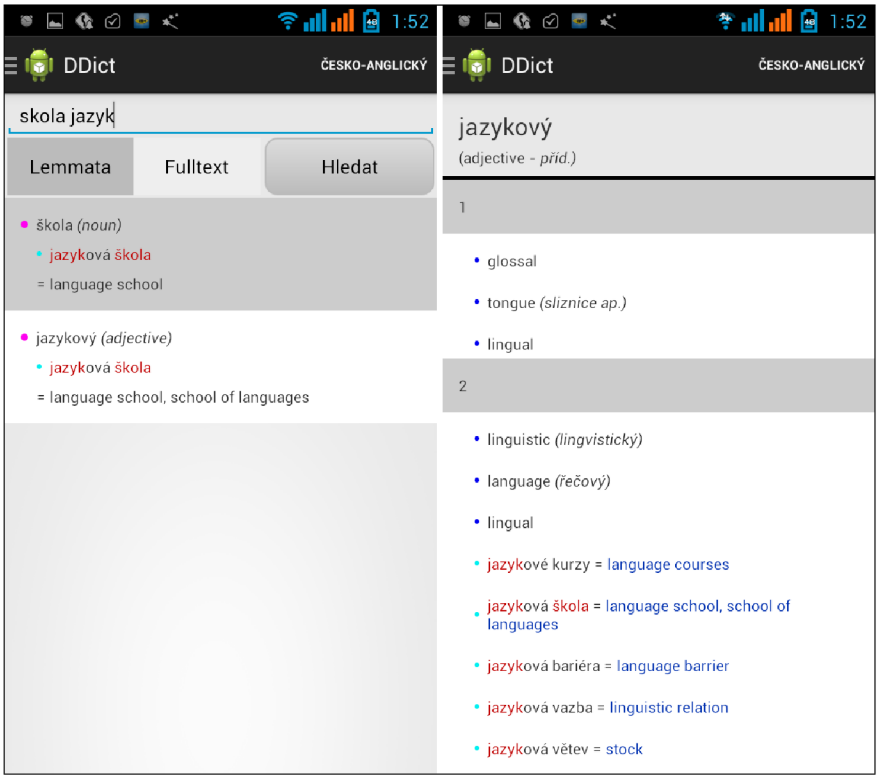
Dále je každému `LinearLayoutu`, který obsahuje dané lemma, přiřazen `OnTouchListener` a `OnClickListener`. `OnTouchListener` změní barvu pozadí při stlačení daného layoutu a `OnClickListener` provede zobrazení datailního výpisu lemmatu na pravém fragmentu. Přepnutí na pravý fragment zajišťuje metoda `PagerView.setCurrentItem(int pos)`.

Kapitola 6

Testování a vyhodnocení

Testování probíhalo průběžně při vývoji. Na fyzickém zařízení Prestigio PAP4500T s verzí Androidu 4.1.1

Výsledná aplikace je na obrázku 6.1.



Obrázek 6.1: Výsledná aplikace DDict

Na uživatele působí aplikace už od spuštění pozitivně. Při prvním spuštění se zeptá kde se nachází databáze a nikomu nedělalo problém ji vybrat a tím nahrát do aplikace.

Ocenili velké množství zobrazených dat společně se zvýrazněním hledané části.

Při dalším vývoji jsem do testování zapojil členy různě technicky zdatných skupin. Technicky zdatnější ocenili možnost rychlejšího prohlížení nalezených dat. V raných fázích vývoje objevili také nepříjemné vlastnosti, které bylo třeba opravit.

6.1 Zobrazení výslovnosti

Zobrazování výslovnosti ve výsledcích není vždy dokonalé. Závisí na převodu informací z tagu `<phoneticForm>` ve zdrojové databázi. Některá slova například `[zIm'bA:bwIEn]` jsou správně převedena na `[zɪm'bɑ:bwɪɛn]`. Správné převedení je možné ověřit na stránkách s online konvertorem [1] nebo v LaTeXu použitím balíčku `tipa`.

6.1.1 Rychlost vyhledávání v databázi

Pro testování rychlosti vyhledávání je potřeba zapnout zobrazování doby provedení dotazu. To se provádí příkazem `.timer ON` v jednoduché utilitě sqlite3 pro příkazový řádek. Nejdříve je nutné připojit testovanou databázi příkazem `.attach czen.db`. Dále už je možné zadat jakýkoli dotaz a po zobrazení výsledků dotazu je vypsána doba jeho provedení. Tato utilita je dostupná i v telefonu.

Pro spuštění utility je potřeba přístup do shellu pomocí `adb` nacházejícím se ve složce `platformtools` v SDK. Bude použit dotaz pro vyhledání fulltextu 5.2 a vyhledávaný výraz bude *"p*"*. Je to jeden z nejhorších případů, protože se vyhledávají výskyty ve všech větách, které obsahují slovo začínající na *p*.

```
102651|zemědělsko-potravinářský|adjective|355488|food farming
102651|zemědělsko-potravinářský|adjective|355489|farming and food
Run Time: real 0.179 user 0.167000 sys 0.012000

102651|zemědělsko-potravinářský|adjective|355488|food farming
102651|zemědělsko-potravinářský|adjective|355489|farming and food
CPU Time: user 2.110000 sys 0.080000
```

Tímto náročným dotazem se zjistilo, že vyhledávání v databázi je na mobilním zařízení zhruba 12x pomalejší. Použití více než sta tisíc nalezených záznamů je v aplikaci zbytečné. Při provádění stejného dotazu a použití nejméně tří písmen je výsledek dotazu na mobilu získán do deseti milisekund.

6.2 Srovnání funkcí výsledné aplikace

Pro porovnání s testovanými aplikacemi byla zvolena kritéria uvedená z tabulce.

	Fulltext	Diakritika	Skloňování	Návrhy	Odkazy	Historie
DDict	x	x	x	x	x	
ColorDict				x	x	x
HandyLex			x	x	x	x
dict.cc		x		x	x	x

Tabulka 6.1: Tabulka porovnávající funkce aplikací

Vytvořená aplikace DDict jako jediná zprostředkovává fulltextové vyhledávání a společně s dict.cc se v ní dá vyhledávat i bez zadávání diakritiky.

[1] http://www.lfsag.unito.it/ipa/converter_en.html

Aplikace má větší možnosti v zobrazení a rychlého procházení většího množství vyhledaných dat. V rámci vyhledání jednoho hesla aplikace označuje už prohlédnuté prvky.

V tabulce se nedají porovnat možnosti zobrazení. Ty jsou také těžko posuzovatelné, protože uživatelé mohou mít jiné preference.

Uživatelům chyběla historie vyhledávání a nepřesné zobrazení výslovnosti. Při rychlém procházení velkého počtu slov, by se mohla historie naplnit moc daty a nebyla by přehledná. Řešení například v podobě důkladných statistik je ponecháno do dalších verzí.

Kapitola 7

Závěr

Cílem práce bylo vytvořit systém pro vyhledání a zobrazení rozsáhlých slovníkových dat na mobilním zařízení. Součástí řešení se byl i převod zdrojových dat v XML strukturovaných podle formátu LMF do databáze SQLite s rozšířením FTS. Převodní skript byl implementován v jazyce Python. Výhodou zvoleného řešení oproti většině volně dostupných slovníkových aplikací je velký zdroj slovníkových dat včetně příkladů užití ve větách.

Problém zobrazení většího množství informací byl vyřešen rozdělením do dvou oken mezi kterými je možné rychle přejíždět tahem prstu ze strany na stranu. V prvním jsou výsledky vyhledání s náhledem na překlad. Ve druhém je zobrazení všech informací vybraného slova. V aplikaci je na rozdíl od porovnávaných řešení možné vyhledávat jak pomocí zadání jednoho slova, tak fulltextově. Návrhy vyhledávání aplikace nabízí podle aktivovaného módu vyhledávání. V okně s výsledky vyhledávání je možné použít vyhledaná slova k vytvoření nového vyhledávání pouhým kliknutím na libovolné slovo. Aplikace rozpozná v jakém jazyce má slovo vyhledávat a také podporuje vyhledávání bez zadávání diakritiky.

V budoucích verzích přibude chybějící historie a předělání zobrazení v režimu na šířku. Současné řešení je schopné zobrazit překlady při překlopení na stranu, ale neprovádí úpravu v zobrazení jako třeba aplikace ColorDict. Použitím fragmentů je implementace této možnosti usnadněna. Možné pokračování bych viděl také v zabudování předávání slovíček z jiných aplikací.

Literatura

[1] About SQLite. [online], [cit. 2014-07-24].
URL https://www.sqlite.org/about.html

[2] About WordNet - WordNet. [online], [cit. 2014-07-24].
URL http://wordnet.princeton.edu/

[3] Activity — Android developers. [online], [cit. 2014-07-24].
URL http://developer.android.com/reference/android/app/Activity.html#ActivityLifecycle

[4] Android — Lingea s.r.o. [online], [cit. 2014-07-24].
URL http://mobile.lingea.eu/android

[5] Aplikace pro Android ve službe Google Play. [online], [cit. 2014-07-24].
URL https://play.google.com/store/apps

[6] SQLite FTS3 and FTS4 Extensions. [online], [cit. 2014-07-24].
URL https://www.sqlite.org/fts3.html

[7] StarDict format. [online], [cit. 2014-07-24].
URL https://code.google.com/p/babiloo/wiki/StarDict_format

[8] XDXF - XML Dictionary Exchange Format. [online], [cit. 2014-07-24].
URL http://xdxf.sourceforge.net/

[9] Allen, G.: *Android 4 Průvodce programováním mobilních aplikací*. Computer Press, 2001, iSBN 978-80-251-3782-6.

[10] Language resource management - Lexical markup framework (LMF). Technická zpráva, International Organization for Standardization, 2 Park Street, London W1A 2BS, UK, 2008.