MASARYKOVA UNIVERZITA
FAKULTA INFORMATIKY

Japonský slovník pro Android

BAKALÁŘSKÁ PRÁCE

Jaroslav Klech

Brno, 2013

Prohlášení

Prohlašuji, že tato bakalářská práce je mým původním autorským dílem, které jsem vypracoval samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používal nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

Jaroslav Klech

Vedoucí práce: Mgr. Bc. Jonáš Ševčík

Poděkování

Chtěl bych poděkovat mému vedoucímu, Jonáši Ševčíkovi, za jeho podporu, cenné rady a především za trpělivost. Poté bych chtěl poděkovat všem mým přátelům, kteří mi pomohli s touto prací. V neposlední řadě bych chtěl poděkovat svým rodičům, kteří mě podporují ve studiích a životě. Děkuji vám všem.

Shrnutí

Cílem této práce je navrhnout a vytvořit aplikaci představující japonský slovník pro mobilní telefony s operačním systémem Android. Aplikace si ukládá poslední prohlížené překlady a je schopna je prezentovat. K realizaci vyhledávání japonských výrazů je využit slovník JMDict, který je téměř denně doplňován o nové výrazy. Aplikace nabízí možnost aktualizovat slovník na nejnovější verzi bez nutnosti opětovné instalace. Dále je umožněno zobrazit informace o jednotlivých japonských znacích kanji poskytovaných slovníkem Kanjidic2.

Klíčová slova

Android, japonština, slovník, překlad, mobilní aplikace, Apache Lucene, JMDict, Kanjidict2

Obsah

1 Úvod

Být schopen komunikovat s ostatními je v dnešní době stěžejní a téměř samozřejmostí. S využitím výpočetní techniky se jazyková bariéra stále více ztenčuje. Nikoho již nepřekvapí hojný výskyt různých typů počítačů ve světě. Mezi nejpoužívanější zařízení patří chytré mobilní telefony, které můžeme využít při komunikaci v cizím jazyce.

Základním způsobem, jak se domluvit cizím jazykem, je prostý překlad, například pomocí různých automatických překladačů. Není novinkou používání překladatelských aplikací na počítačích nebo přenosných zařízeních. Překladačů pro běžné evropské jazyky je celá řada. Méně časté bývají aplikace, které umožňují překlad do jazyků využívajících jinou abecedu, než je latinka. Jedním z těchto jazyků je i japonština. Jedna z možností efektivního překladu z japonštiny je použití online překladačů, které v cizí zemi není vždy možné nebo žádoucí kvůli vyžadovanému internetovému připojení použít. V takové situaci je výhodné mít nainstalovanou aplikaci, která ke své činnosti nepotřebuje přístup k internetu.

Cílem této práce je navrhnout a vytvořit aplikaci pro mobilní telefony s operačním systémem Android, která dokáže překládat uživatelem zadané výrazy z japonštiny. Systém Android je v dnešní době široce rozšířen a vyskytuje se na mobilních telefonech malých rozměrů, ale i na tabletech a jiných zařízeních. Při tvorbě aplikace byl kladen důraz na příjemné uživatelské rozhraní a zachování funkčnosti na displejích s různým rozlišením a velikostí.

Aplikace je vytvořená tak, aby bezchybně fungovala od Androidu verze 2.2 (Froyo) až po současně nejnovější verzi 4.2 (Jelly Bean). Je používán kompatibilní balík, aby se i ve starších verzích systému daly využívat novější komponenty jako jsou fragmenty (viz 3.3). Po instalaci aplikace a stažení slovníků není k samotné funkčnosti potřeba

internetové připojení, což je užitečné, pokud uživatel nemá aktivovaný datový tarif nebo se nachází v cizí zemi. Aplikace využívá významový slovník Japanese-Multilingual Dictionary (JMDict) a znakový slovník Kanjidic2. Překlad je poskytován do jazyků obsažených ve slovníku JMDict. Mezi tyto jazyky patří angličtina, němčina, francouzština a holandština. Aplikace si uchovává poslední hledané výrazy a při spuštění je uživateli nabídne k zobrazení. Zadávání výrazů je prováděno psaním japonských znaků nebo převodem výrazu zapsaného mezinárodní transkripcí Hepburn.

Před samotnou prací na aplikaci bylo zapotřebí nastudovat návody a rady od společnosti Google a seznámit se s programováním pro systém Android. K vývoji aplikace bylo použito vývojové prostředí Eclipse, které je zaměřené na vývoj aplikací v jazyce Java. Pro možnosti vývoje bylo třeba nainstalovat Android Software development kit (SDK), který poskytuje potřebné knihovny pro fungování aplikace v prostředí operačního systému Android. Jednotlivé verze programu byly ukládány na vytvořený repozitář GITHub, který se široce používá k vývoji aplikací a k správě jejich verzí. Testování aplikace bylo provedeno na telefonu Samsung galaxy S s operačním systémem Android verze 4.1. Dále byl testován vzhled a správná funkčnost aplikace na různých typech zařízení s různými verzemi systému Android pomocí simulátoru Android Virtual Device (AVD).

2 Analýza

Aplikace využívá významový slovník JMDict [12] pro vyhledávání překladů a znakový slovník Kanjidic2 [9] pro dodatečné informace o jednotlivých znacích kanji.

2.1 Japonský slovník JMDict

Projekt JMDict je projekt, který má za cíl sestavit databázi překladů japonských výrazů. Projekt začal v roce 1999 jako odnož projektu Japanese-English Electronic Dictionary (EDICT). Slovník je nabízen ke stažení na internetových stránkách Jima Breema ve formátu XML [12].

Od roku 2006 je zdrojový soubor JMDict slovníku denně generován z databáze. Zdrojová databáze pro slovník je aktualizována a doplňována téměř každý den [12]. Aktuální slovník obsahuje více než 160 000 záznamů.

2.1.1 Licence

JMDict soubor je chráněn autorským právem. Slovník je šířen pod licencí Commons Attribution-ShareAlike Licence. Dle licence má uživatel dovoleno kopírovat, šířit a prezentovat slovník za podmínky, že uvede autorství práce, jak je požadováno autorem. U mobilních aplikací skupina Electronic Dictionary research and Development Group požaduje uvést autorství na samostatné obrazovce přístupné z menu pod názvem „O aplikaci“, „Zdroje“ apod. [4] (viz 3.8).

2.1.2 Struktura

Struktura JMDict slovníku je zřejmá z ukázkového záznamu [8], společně s Document Type Definition (DTD) dodávaném přímo ve zdro-

jovém souboru slovníku [7].

Obrázek 2.1: Zobrazovaný záznam slovníku JMDict

V aplikaci jsou využity následující položky slovníku:

keb Skutečný zápis slova, který se může skládat ze znaků kanji, které jsou Japonci užívané čínské znaky. Může být doplněno slabičnou abecedou kana a latinskou abecedou (viz bod 1 v obr. 2.1). Jeden význam může mít více různých zápisů slova a jeden zápis může mít více různých čtení.

reb Jedná se o japonské čtení, kde obsah je omezen na kanu a příbuzné znaky jako chouon a kurikaeshi (viz bod 2 v obr. 2.1). Každé čtení má transkripci do latinské abecedy (viz bod 3 v obr. 2.1). V aplikaci je využita obvyklá anglická transkripce Hepburn. Ve slovníku JMDict nejsou jednotlivá čtení přiřazena zápisům. Jedná se o množinu alternativních zápisů a čtení (viz bod 4 v obr. 2.1).

sense Element, který obsahuje význam pro dané slovo v různých jazycích (viz bod 5 v obr. 2.1). Jeden záznam může obsahovat více významů. Samotný význam je uložen v elementu *gloss*, u kterého atribut *xml:lang* určuje jazyk překladu. Pokud není

atribut *xml:lang* uveden, použije se jako základní jazyk angličtina. Každý záznam obsahuje alespoň jeden význam, který není vždy dostupný ve všech jazycích.

2.2 Slovník japonských znaků Kanjidic2

Kanjidic2 je projekt, který má za cíl vytvořit databázi znaků kanji ve formátu XML. Znakový slovník byl vydán v roce 2008 a zůstává relativně konstantní. Kanjidic2 obsahuje přes 12 000 záznamů jednotlivých kanji znaků. Význam znaků je dostupný v angličtině, francouzštině, portugalštině a španělštině [9].

2.2.1 Licence

Licence pro slovník Kanjidic2 je shodná s licenčními podmínkami pro použití JMDict slovníku (viz 2.1.1).

2.2.2 Struktura

Kanjidic2 slovník je členěn do celků představujících záznamy pro jednotlivé kanji znaky. U jednotlivého znaku jsou uvedeny informace jako počet tahů, význam nebo odkazy do výkladových slovníků. V dostupném DTD [13] a ukázkovém záznamu [10] je uvedena kompletní struktura dokumentu.

V aplikaci jsou využity následující položky Kanjidic2 slovníku:

literal V tomto elementu je uložen vyhledávaný znak kanji.

radical Zde je zobrazen typ classical, který reprezentuje odkaz do *Modern Japanese-English Character Dictionary* slovníku.

grade Reprezentuje třídu znaků.

- Třída 1-6 odpovídá třídám specifikovaným japonským ministerstvem školství pro šestiletou základní školu. Jedná

se o 1 006 kanji znaků řazených mezi denně používané (nazývané Kyouiku).

- Třída 8 zastupuje zbývajících 1 130 Kyouiku znaků vyučovaných na tříleté nižší střední škole.
- Třída 9 a 10 zastupuje Jinmeiyou znaky používané pro jména.

stroke count Počet tahů potřebných k nakreslení znaku.

SKIP Používá se k identifikaci znaků na základě tří čísel. Kód je užitečný, pokud potřebujeme znak vyhledat, ale neznáme jeho výslovnost.

onyomi Sinojaponské čtení on'yomi.

kunyomi Japonské čtení kun'yomi.

meaning Obsahuje význam znaku. V aplikaci jsou využity významy v anglickém a francouzském jazyce v návaznosti na slovník JMDict.

nanori Zápis používaný ve jménech.

3 Implementace

Při implementaci byl brán zřetel na aktuální používané technologie a zvyklosti při vytváření aplikací pro platformu Android. Přestože starší verze Androidu nepodporují nejnovější technologie, bylo zapotřebí zajistit zpětnou kompatibilitu za pomoci podpůrného balíku společnosti Google [11] a knihovny ActionBarSherlock [2] vytvořené Jakem Whartonem (viz 3.3.1). V případě nemožnosti zajištění zpětné kompatibility se staršími verzemi systému, nebyly určité nejnovější technologie použity.

Pro realizaci vyhledávání je využíván vyhledávací stroj Apache Lucene [3]. Lucene je textový vyhledávač napsaný v programovacím jazyce Java. Předností je nízká náročnost na operační paměť a možnost uložení dokumentu na indexovaný klíč. Velikost indexu představuje 20 - 30 % velikosti původních indexovaných dat. V případě navržené aplikace, kde jsou indexovány pouze japonské výrazy, se jedná o minimální množství.

3.1 Struktura kódu aplikace

Zdrojový kód je reprezentován třídami členěnými do balíků podle jejich významu. Kód japonského slovníku je rozdělen do následujících logických celků:

cz.muni.fi.japanesedictionary.database Balík obsahuje třídy zaručující práci s databází SQLite. Databáze SQLite je využívána k ukládání zobrazených překladů, které jsou při startu aplikace nabídnuty uživateli k zobrazení.

cz.muni.fi.japanesedictionary.engine Balík obsahuje třídy realizující obslužné operace pro aktivity a fragmenty. Jedná se o adaptéry pro zobrazení dat a správu záložek. V tomto balíku se také nachází stěžejní komponenty aplikace, které provádí vyhledávání ve slovnících mimo uživatelské vlákno.

cz.muni.fi.japanesedictionary.entity Balík obsahuje třídy představující japonský překlad a znak kanji. Jedná se o základní nosné objekty dat v aplikaci.

cz.muni.fi.japanesedictionary.fragments Balík obsahuje třídy fragmentů. Fragmenty jsou načítány do aktivit a zobrazují data uživateli.

cz.muni.fi.japanesedictionary.interfaces Balík obsahuje rozhraní potřebné pro komunikaci mezi komponentami. Aktivity implementují rozhraní a poskytují funkcionalitu pro zobrazené fragmenty. Pro komunikaci mezi fragmenty je zapotřebí využít aktivity jako prostředníka.

cz.muni.fi.japanesedictionary.main Balík obsahuje aktivity, mezi které patří i *MainActivity*, která se zobrazuje při spuštění aplikace. Mimo jiné se zde nachází aktivity pro zobrazení překladu a informací o kanji znaku.

cz.muni.fi.japanesedictionary.parser Balík obsahuje službu vykonávající stažení a zpracování slovníků a komponenty potřebné pro její vykonání. Nachází se zde třídy zajišťující zpracování stažených XML souborů.

3.2 Zpracování dat slovníků

Nejdůležitější funkcionalitou aplikace je stažení a zpracování zdrojových slovníků. Při spuštění aplikace dochází k ověření přítomnosti slovníkových dat. Pokud nejsou slovníky nalezeny, je uživatel dotázán, zdali chce stáhnout aktuální slovníková data. Při dotazu je uživateli sdělena i přibližná velikost stahovaných dat za účelem zamezení nevědomého přenesení velkého objemu dat.

V případě nedostupnosti internetového připojení v době pokynu ke stažení dat, dochází po obnovení připojení k nové výzvě uživatele. V případě nedostupnosti velkokapacitního úložiště v telefonu

je uživatel vyzván ke vložení paměťové karty. Soubory jsou ze serveru staženy ve formátu gzip (GNU kompresní formát).

3.2.1 Vyhledávací stroj Apache Lucene

Jako první možnost pro uložení slovníku se nabízí vestavěná databáze systému Androidu SQLite. Vestavěnou databázi není doporučené využívat pro enormní množství dat. Při zpracování slovníků by bylo třeba vytvořit relační model, který by reflektoval potřebu indexovat velké množství japonských výrazů.

Z důvodu požadavků kladených na rychlé vyhledávání v textu byl místo systémové databáze použit externí projekt Apache Lucene, který je pro vyhledávání v textu vytvořen a optimalizován. Vyhledávací stroj Lucene je sestaven ze skupiny knihoven, které spolu spolupracují a vytvářejí požadovanou funkcionalitu.

Verze knihovny

Při použití projektu Lucene se vyskytl problém s kompatibilitou nejnovější verze. Přestože Android poskytuje téměř kompletní Java 6 prostředí, aktuální verze Lucene (verze 4.2.1) vyžaduje systémové třídy, které Android neposkytuje. Jedná se převážně o třídy provádějící správu operační paměti. Z tohoto důvodu bylo zapotřebí vyzkoušet různé verze Lucene, až byla zvolena verze 3.6, která je doporučená pro zpětnou kompatibilitu i společností Apache [3].

Zvolená verze Lucene je vysoce flexibilní a umožňuje programátorovi specifikovat prostředí, ve kterém bude projekt provozován. První konfigurace Lucene probíhá výběrem použitých knihoven. To umožňuje k aplikaci přibalit pouze potřebné součásti a tak limitovat velikost aplikace. Mezi základní knihovny potřebné pro správnou funkci patří jádro vyhledávače, knihovna analyzátorů a knihovna dotazů pro vyhledávání.

Analyzátor textu

Pro vyhledávání je důležitý zvolený analyzátor. Analyzátory se používají ke správnému indexování dat a také k vyhledávání v indexu. Je důležité, aby byl při indexování a poté při vyhledávání použit stejný analyzátor.

Pro indexování a vyhledávání japonských textů není vhodné použít základní analyzátor *SimpleAnalyzer*, který se používá na základní analýzu latinských jazyků. V aplikaci je využita třída *CJKAnalyzer*, která je určena pro čínštinu, japonštinu a korejštinu. Pro vyhledávací stroj Lucene je možné použít i analyzátory vyvinuté mimo společnost Apache.

V průběhu implementace došlo ke snaze využít i jiné analyzátory, než je *CJKAnalyzer* , ale většina z nich měla zásadní problém s nároky na operační paměť. Problém byl způsoben snahou sestrojit v paměti tabulky pro analýzu a konverzi japonských znaků.

Indexování

Lucene je určen pro vyhledávání celých slov v textu (fulltext). V aplikaci je vyhledávání založeno na základě části slova. Lucene podporuje hvězdičkovou notaci (Wildcard), kde znak hvězdička (*) zastupuje libovolný počet znaků při vyhledávání a znak otazník (?) zastupuje právě jeden znak. Vyhledávání pomocí hvězdičkové notace se značně projevuje na době potřebné k prohledání indexů. V případě výskytu hvězdičky na začátku výrazu (nastává při vyhledávání ve středu nebo na konci slova) musí Lucene prohledat všechny položky indexů.

Po zvážení výše uvedených informací byl zvolen přístup, kde se využívá vlastností vyhledávacího stroje Lucene pro vyhledávání v textu. Mezera mezi slovy slouží jako rozdělovač pro algoritmus analyzující text. Požadovaný indexovaný japonský výraz je prolo-

žen mezerami mezi znaky. Nově získaný výraz Lucene považuje za větu o slovech obsahujících jeden znak. Vyhledávání poté probíhá podobným způsobem. Hledané slovo je proloženo mezerami a uzavřeno do uvozovek. Uzavřením výrazu do uvozovek je vyhledávacímu stroji předán požadavek na přesnou shodu s částí textu. Nastává situace, na kterou je Lucene optimalizovaný, dochází k vyhledání věty v textu.

Pro možnost rychlého a přesného vyhledávání v jednotlivých částech slova je využit přístup, kdy je na začátek a konec každého indexovaného výrazu přidána specifická značka. Při indexování japonských výrazů je použita značka „lucenematch“, která vymezuje začátek a konec výrazu. Pro vyhledávání v indexu je sestaven dotaz na základě těchto pravidel:

přesná shoda *"lucenematch výraz lucenematch"*

začátek slova *"lucenematch výraz"*

střed slova *"výraz"*

konec slova *"výraz lucenematch"*

Pod pojmem *výraz* je myšleno vyhledávané slovo proložené mezerami.

3.2.2 Služba pro zpracování

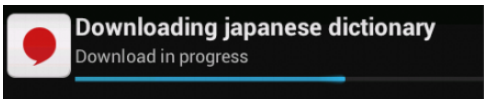
Pro stažení a zpracování slovníků se využívá služba (*Service*). Služba běží na pozadí aplikace. Využívá se převážně pro náročné a dlouhotrvající operace. Služba je spuštěna v aplikačním vlákně. V případě provádění náročných operací ve službě, je potřeba pro tyto operace spustit nové vlákno.

V aplikaci zastává roli služby třída *ParserService* z balíku *parser*. Služba je spuštěna v samostatném vlákně, aby neblokovala aplikaci. Bylo zapotřebí nastavit spuštění služby na popředí. Pokud je služba

nastavena na běh na popředí, systém ji považuje za komponentu, které si je uživatel vědom, a ukončí ji pouze v případě nedostatku prostředků. V případě násilného ukončení služby systémem je po uvolnění dostatečného množství prostředků služba restartována. Dochází k restartování postupu a začínají se stahovat slovníky. Běh na popředí umožňuje zpracovávání slovníku i v případě, že uživatel zavře aplikaci a uzamkne telefon. Uživatel je informován o postupu pomocí notifikace (obr. 3.1).

Celková doba zpracování slovníků na současných zařízeních zabere přibližně pět minut. Tato doba je způsobená nutností přečíst a zpracovat celý obsah slovníku za použití malého množství operační paměti. Výhodou zpracování dat přímo v mobilním zařízení je nezávislost na externím serveru, který by data zpracoval a nabídl je mobilnímu zařízení ke stažení.

V případě, že se v telefonu nachází platná slovníková data, jde tedy o aktualizaci slovníku, služba využívá dočasné úložiště pro ukládání nových dat. Tento přístup umožňuje využití aplikace i v průběhu aktualizace slovníků. Po úspěšném ukončení aktualizace služba nahradí původní slovníková data novými.



Obrázek 3.1: Notifikace zobrazená uživateli při stahovaní slovníku

3.2.3 Stažení souborů

Ke stažení souborů se využívá oficiálních odkazů na aktuální verze slovníků. Tím je při každé aktualizaci zaručeno stažení nejnovější verze. Služba stahuje soubory na externí úložiště, kterým je nejčastěji Secure Digital (SD) karta. Pokud telefon neumožňuje připojení

SD karty, využívá se vysokokapacitního vestavěného úložiště telefonu.

U nestabilního připojení k internetu služba při ztrátě spojení stahovaní ukončí a po obnovení připojení na něj naváže. Služba bude čekat na připojení do doby, než ji uživatel nebo systém ukončí. Po úspěšném zpracování jsou stažené soubory smazány.

3.2.4 Zpracování stažených souborů

Po rozbalení by soubor JMDict zabíral přibližně 70 MB místa v zařízení. Proto je čtení dat prováděno za pomoci třídy *GZIPInputStream*, která realizuje rozbalování souboru při čtení.

Stažené soubory jsou po rozbalení ve formátu XML (viz kapitoly 2.1.2 a 2.2.2). Základním způsobem zpracování XML dat je využití Document Object Model (DOM) přístupu. Jedná se o případ, kdy se v operační paměti sestaví hierarchie objektů odpovídající původnímu XML. Při použití DOM je potřeba načíst celý soubor do operační paměti. Tento přístup není vhodný pro mobilní zařízení.

V aplikaci je ke zpracování XML souborů využit nástroj Simple API for XML (SAX). SAX zpracovává XML při čtení souboru a v paměti sestavuje pouze právě přečtené části dokumentu. Je založen na principu volání metod začátek dokumentu, počáteční element, obsah textového uzlu, koncový element a konec dokumentu. Výhoda nástroje SAX je převážně v nízké náročnosti na operační paměť.

Při otevření elementu *entry* se vytvoří nový Lucene dokument, do kterého se postupně ukládají přečtená data. Jedinými indexovanými položkami jsou elementy představující japonské výrazy. Po každých pěti zpracovaných procentech dochází k uložení indexu za účelem zamezení udržování nadměrného množství objektů v paměti.

V JMDict slovníku je třeba indexovat elementy pro zápis i čtení (viz 2.1.2) japonských výrazů. Při vyhledávání se pak využívá index pro čtení, při transkripci z latinky, nebo všeobecný japonský index

pro celý slovník (viz 3.4.4).

V Kanjdic2 slovníku je třeba indexovat pouze znak kanji. Kanjidic2 obsahuje i alternativní zápisy pro kanji, jedná se ovšem o znakový slovník a vyhledává se pouze na základě znaku (viz 3.5.3).

K ukládání struktury dat je využita konstrukce JavaScript Object Notation (JSON). Jedná se o formát pro výměnu dat, který je jednoduše čitelný a zapisovatelný jak pro člověka, tak pro stroj [6]. Použití JSON je především vhodné pro ukládání významů překladu, kdy jeden překlad může mít více významů v jednom jazyce. Nejčastěji je využita třída *JSONArray*, která umožňuje ukládat a jednoduše rekonstruovat více logicky oddělených prvků.

3.3 Zobrazení dat

Aktivita (*Activity*) je nejčastěji užívaný prvek v aplikacích. Jedná se o základní komponentu, která zobrazuje informace uživateli a komunikuje s ním. Každá aplikace, která má uživatelské rozhraní (UI), obsahuje alespoň jednu aktivitu. Aktivita obsahuje komponenty nazývané pohledy (*View*), které rozložením a obsahem vytvářejí UI aktivity [1]. Od verze Androidu 3.0 (Honeycomb) přibývají v SDK komponenty nazývané fragmenty, které jsou s aktivitou úzce spjaty.

Jedna aktivita může obsahovat více fragmentů. Na fragment v aktivitě se dá nahlížet jako na samostatnou sekci, která má svůj vlastní životní cyklus a své vlastní vstupní události. Fragment může být přidán či odebrán při běhu aktivity, a tak tvoří pružnou komponentu UI. Viditelný fragment musí být vždy vložen v aktivitě [5].

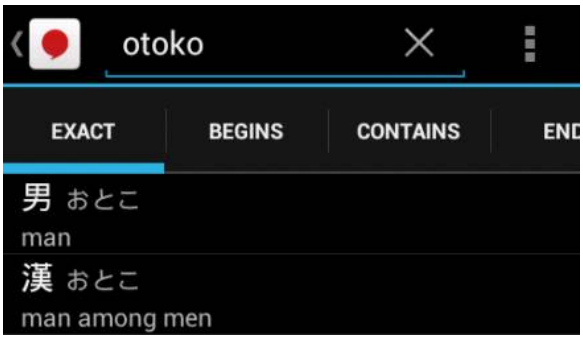
Fragmenty byly navrženy převážně kvůli zavedení platformy Android i na tablety, které mají větší displeje než mobilní telefony. Na mobilním telefonu se většinou fragmenty nezobrazují současně, ale spouští se nové aktivity, které zobrazují jednotlivé fragmenty. Na displeji tabletu lze mnohdy zobrazit více fragmentů najednou.

3.3.1 Zpětná kompatibilita

Aplikace je konstruována tak, aby fungovala na verzi Androidu 2.2 (Froyo) a výše. Zpětnou kompatibilitu umožňuje podpůrná knihovna *Support Library*, kterou je možno zahrnout přímo do projektu aplikace. Knihovna obsahuje balíky přidávající funkční celky, které by jinak na starších verzích Androidu nebylo možno použít [11].

V současných aplikacích se běžně využívají fragmenty. Fragmenty jsou dostupné v Androidu od verze 3.0 (Honeycomb). Podpůrný balík umožňuje využívat fragmenty i v aplikaci japonského slovníku. Kromě nastavení aplikace (viz 3.7) a aktivity „O aplikaci“ (viz 3.8) jsou veškeré informace v aplikaci zobrazeny uživateli skrze fragmenty.

Podpůrná knihovna dále umožňuje využít komponentu *ViewPager* používanou ve spolupráci se záložkami. *ViewPager* umožňuje přesun mezi záložkami posunem prstu po obrazovce a plynulou změnu fragmentů zobrazovaných pod záložkami. Tato funkcionalita je použita v aplikaci k realizaci zobrazení vyhledaných překladů (viz 3.4.1).



Obrázek 3.2: Vyhledávání pomocí SearchView v ActionBaru

Další knihova, která je využita za účelem zajištění zpětné kompatibility, je *ActionBarSherlock* [2]. Jedná se o knihovnu, která poskytuje Application Programming Interface (API) pro práci s komponentou *ActionBar*. Ten představuje navigační prvek zavedený od verze 3.0

(Honeycomb). Při horním okraji obrazovky zobrazuje lištu s možnostmi pro navigaci a ovládání aplikace. Součástí knihovny *ActionBarSherlock* je i prvek *SearchView*, který poskytuje uživatelské rozhraní pro vyhledávání v aplikaci. V aplikaci japonského slovníku je využit *SearchView* k vyhledávání překladů (obr. 3.2).

3.4 Hlavní aktivita

Jako úvodní obrazovka při spuštění aplikace se zobrazuje aktivita *MainActivity*. Jedná se o aktivitu, která zajišťuje zobrazení vyhledávacího pole a implementuje *ViewPager* pro zobrazení fragmentů s výsledky hledání.

Obrázek 3.3: Hlavní aktivita pracující v duálním režimu

Tato aktivita pracuje ve dvou režimech, a to v jednoduchém nebo duálním (obr. 3.3). Jednoduchým režimem se myslí aplikace spuštěná na telefonu, tedy na malém displeji. V duálním režimu aktivita funguje v případě tabletů, kde zobrazuje, kromě fragmentu s vyhledanými výsledky, také fragment s informacemi o posledním vybraném překladu. Zaručuje se tak optimální využití zobrazovací plochy přístroje.

3.4.1 ViewPager

ViewPager je komponenta umožňující napojení fragmentů na záložky aktivity. Obsahuje *FragmentPagerAdapter*, který je v aplikaci nadefinován jako třída *MainPagerAdapter* z balíku *engine*. Cílem třídy MainPagerAdapter je zajistit správu fragmentů pro *ViewPager*.

MainPagerAdapter si udržuje odkaz na právě zobrazený fragment a na fragmenty sousedících záložek. Pokud má být zobrazena záložka, pro kterou ještě nebyl vytvořen fragment, adaptér zavolá metodu *getItem (int index)*. Metoda vrací novou instanci fragmentu specifickou pro danou záložku. V případě zobrazení záložky si adaptér zároveň vytváří instance fragmentů pro bezprostředně sousedící záložky, čímž dosahuje plynulého přechodu při přesunu mezi záložkami.

3.4.2 ListFragment

ListFragment představuje posouvací seznam položek. Jedná se o potomka třídy *Fragment*, který zajišťuje správu *ListView*.

V aplikaci se nachází třída *ResultFragmentList* (dále jen „seznam“) z balíku *fragments*, která rozšiřuje *ListFragment* a poskytuje aplikační logiku pro vyhledávání a zobrazení výsledků. Při vytváření instance třídy se jako argument předává klíč záložky, pod kterou fragment patří. Po vytvoření fragmentu se již jeho klíč nemění.

Seznam klíčů záložek:

exact přesná shoda

beginning začátek slova

middle střed slova

end konec slova

Při vytvoření si fragment v metodě *onCreate()* vyžádá aktuální vyhledávané slovo od aktivity (*MainActivity*) a provede vyhledávání

pro získané slovo a záložkou určenou část slova. Pokud dojde ke změně záložky a fragment již existuje, zavoláním metody *search (String slovo)* se provede aktualizace seznamu a patřičné vyhledání výrazu. Pokud se slovo z posledního hledání shoduje s aktuálním slovem předaným do metody *search*, neprobíhá nové vyhledávání a zobrazí se výsledky předešlého hledání.

3.4.3 ListAdapter

ListAdapter tvoří prostředníka mezi *ListView* a daty, které se v něm mají zobrazit. V aplikaci je využita třída *TranslationsAdapter*, která je potomkem systémové třídy *ArrayAdapter*. Hlavním důvodem využití vlastního adaptéru je potřeba zobrazit vlastní prvek seznamu. Prvek seznamu symbolizuje heslo slovníku, které obsahuje zápis, výslovnost slova a jeden z významů překladu (viz obr. 3.4).

Při zobrazení prvku seznamu na tabletu bylo zapotřebí zajistit, aby po výběru překladu zůstal zvolený prvek seznamu zvýrazněn. Aplikace obsahuje dva rozdílné XML soubory udávající vzhled a chování prvku, které jsou podle typu zařízení používány v seznamu.

Obrázek 3.4: Prvek seznamu zobrazující možný překlad

3.4.4 Vyhledávání ve slovníku JMDict

Vyhledávání dat je považováno za blokující operaci, a proto by mělo být prováděno v samostatném vlákně. Pokud by bylo prováděno v uživatelském vlákně, mohla by aplikace přestat odpovídat a systém Android by ji mohl násilně ukončit (Force Close).

V aplikaci načítání dat provádí třída *FragmentListAsyncTask* (dále jen „vyhledávač“). Jedná se o potomka třídy *AsyncTask*. *AsyncTask* je

komponenta Androidu, která provádí právě jednu operaci v samostatném vlákně a neblokuje uživatelské rozhraní aplikace.

Spuštění a přerušení načítaní řídí seznam. V případě změny hledaného výrazu nebo absence načtených dat, vytváří novou instanci vyhledávače, předává mu svůj klíč (požadovanou část slova) a požadovaný výraz. Pokud v době vytváření nového vyhledávače stále vyhledává předchozí vyhledávač, je předchozí přerušen a nahrazen novým. Rušením starého vyhledávače se předchází zobrazení nežádoucích položek v seznamu výsledků a šetří se systémové zdroje zařízení.

Jestliže vyhledávaný výraz předaný do vyhledávače obsahuje hodnotu *null*, jedná se o první vyhledávání po spuštění aplikace. Dochází k načtení deseti naposledy zobrazených překladů z databáze (viz 3.5.1). Vybírá se posledních deset odlišných překladů v pořadí jejich zobrazení od nejnovějšího. Databáze posledních překladů zůstává platná i po aktualizaci slovníků nebo aktualizaci aplikace (pokud v novější verzi aplikace nedošlo ke změně struktury databáze). Databáze je uložena v paměti telefonu.

Vyhledávač při spuštění zkontroluje, zdali existuje složka se slovníkem JMDict. Pokud složka neexistuje, je uživateli zobrazena informace o nedostupnosti slovníku. Jestliže složka existuje, je vytvořena nová instance vyhledávacího stroje Lucene a sestavuje se dotaz pro vyhledávání. V případě, že hledaný výraz je zadán pouze latinskou abecedou (ověřováno regulárním výrazem), dojde ke konverzi zadaného výrazu do japonské slabičné abecedy hiragana. Pro převod se využívá Hepburnova transkripce. Pokud je převáděna transkripce, k vyhledávání je použit pouze index pro japonské čtení.

Seznam implementuje rozhraní *SearchListener*, který specifikuje metody volané z vyhledávače při nalezení překladu a při dokončení načítání. Jednotlivé nalezené překlady jsou ihned přidány do seznamu a zobrazeny uživateli pomocí volání metody *publishPro-*

gress(), která provádí synchronizaci a je prováděna zpět v uživatelském vlákně. Uživatel tak nemusí čekat na dokončení celého vyhledávání.

3.5 Zobrazení překladu

Při zobrazení překladu mohou nastat dvě možnosti v závislosti na velikosti displeje. V případě tabletu je načteno rozložení obrazovky, které obsahuje zobrazení podrobností o překladu přímo v hlavní aktivitě napravo od seznamu s výsledky. V případě zobrazení na telefonu, dochází ke spuštění nové aktivity. V obou případech je do aktivity vložen fragment zobrazující informace o překladu. V aplikaci se jedná o třídu *DisplayTranslation*.

Aktivita, která zobrazuje *DisplayTranslation* fragment, musí implementovat rozhraní *OnCreateTranslationListener*. Rozhraní zaručuje implementaci metod pro získání instance databáze a zobrazení informací o jednotlivých znacích kanji.

3.5.1 Uložení překladu do databáze

Při zobrazování detailu překladu je právě zobrazovaný překlad uložen do databáze. K uložení se využívá databáze SQLite zabudovaná přímo v systému Android. Struktura databáze je navržena tak, aby využívala metody pro práci s překladem využívané pomocí vyhledávacího stroje Apache Lucene (viz 3.2.1).

Práce s databází je považována za blokující operaci, a proto je třeba ji provádět ve vlastním vlákně. V aplikaci je ukládání překladů zajištěno třídou *DBAsyncTask*. Jako argument vyžaduje instanci databáze, která je realizována třídou *GlossaryReaderContract* z balíku *database*, a právě zobrazovaný překlad, který následně uloží do databáze. Využití nového vlákna umožňuje uložení překladu, aniž by uživatel musel čekat na dokončení ukládání.

3.5.2 Fragment DisplayTranslation

Fragment *DisplayTranslation* provádí zobrazení překladu. Po vytvoření fragment sestrojí ze zaslaného argumentu (*Bundle*) třídu překladu *Translation* z balíku *entity* a provede vykreslení uživatelského rozhraní uživateli.

Obrázek 3.5: Obrazovka zobrazující jeden překlad

Při otočení obrazovky je zapotřebí právě zobrazovaný překlad uložit a po znovu vytvoření fragmentu překlad sestrojit. Systém Android po otočení poskytuje automatickou obnovu uživatelského rozhraní u prvků, které mají řádně nastavený identifikátor. Vzhled uživatelského rozhraní je nadefinován v XML souboru, který se poté načítá do fragmentu. Prvky, jako jsou jednotlivé překlady, se sestavují při vykreslování fragmentu a nejsou součástí hlavního XML sou-

boru fragmentu. Po otočení není systém schopen automaticky sestrojit uživatelské rozhraní obsahující dynamické prvky. Je zapotřebí obnovit vzhled fragmentu ručně.

Uživateli je zobrazeno psaní a čtení vyhledaného překladu. Dalším prvkem zobrazených informací je zobrazení přepisu kany do latinky nazývané nanori. K přepisu je použita Hepburnova transkripce. Následuje zobrazení významů v požadovaných jazycích. Pokud zobrazený překlad význam v požadovaném jazyce neobsahuje, jazyk je zcela vynechán. Pokud uživatel nezvolil ani jeden požadovaný jazyk, zobrazuje se automaticky angličtina. Jako poslední se zobrazuje seznam znaků kanji nalezených ve slovníku Kanjidic2 (obr. 3.5).

3.5.3 Vyhledávání znaků kanji ve slovníku Kanjidic2

Z důvodu urychlení vyhledávání překladů (viz 3.4.4) pro zadaný výraz nejsou informace o jednotlivých znacích kanji vyhledávány společně s překladem. Vyhledání informací k zobrazovaným znakům kanji probíhá po zobrazení překladu. Vyhledávání probíhá v samostatném vlákně za pomoci třídy *CharacterLoader*. Jsou vyhledávány pouze znaky kanji obsažené v hlavním zápise slova.

Při sestavování uživatelského rozhraní je spuštěno vyhledávání znaků pomocí třídy *CharacterLoader*. Do třídy je poslán celý zápis slova. Zápis je při zpracování rozdělen na jednotlivé znaky a u každého znaku je ověřeno, jestli se jedná o kanji. Následně je sestrojen dotaz pro vyhledávácí stroj Apache Lucene (viz 3.2.1) obsahující všechny vyhovující kanji znaky. Formát dotazu odpovídá předpisu „znak_1 znak_2 ...“. Mezera v dotaze je reprezentována jako logická spojka nebo. Uložený Kanjidic2 slovník obsahuje pouze jeden index sestrojený na základě jednotlivých znaků. Vyhledávání probíhá pouze mezi těmito znaky.

Po dokončení vyhledávání jsou znaky zapsány do fragmentu a následně zobrazeny uživateli. Před zápisem je ověřena aktuálnost frag-

mentu. Zápis znaků se provede pouze v případě, že je pro uživatele fragment stále viditelný. Pokud znak ze zápisu nebyl dohledán nebo se nejedná o znak kanji, je tento znak vynechán a není zobrazen uživateli.

3.6 Zobrazení informací o znaku

Pokud je aplikace spuštěna na telefonu, dochází k odstartování aktivity *DisplayCharacterInfoActivity*. V případě tabletu dochází k zobrazení informací o znaku nahrazením detailu o překladu. Fragment překladu je při transakci uložen na zásobník. Při navigaci zpět (tlačítko zpět) je fragment překladu zrekonstruován a zobrazen uživateli. V obou případech aktivita zobrazuje fragment *DisplayCharacterInfo*.

Zobrazovaný znak je do fragmentu předáván ve formě argumentu pro spuštění (*Bundle*) a je při spuštění fragmentu rekonstruován do entitní třídy *JapaneseCharacter*. Aktivita pro zobrazení fragmentu *DisplayCharacterInfo* nemusí implementovat žádné pomocné rozhraní.

DisplayCharacterInfo je nejnižší možná úroveň, do které se aplikace může zanořit. Pokud je fragment zobrazen pomoci samostatné aktivity *DisplayCharacterInfoActivity*, dochází k zaktivnění tlačítka „Domů“. Tlačítko umožňuje navigaci na hlavní aktivitu aplikace.

Struktura zobrazených dat odpovídá struktuře Kanjidic2 slovníku uvedené v kapitole 2.2.2.

3.7 Nastavení aplikace

Spuštění aktivity s nastavením aplikace probíhá přes menu aplikace. Na zařízeních s fyzickým tlačítkem se menu zobrazí po stisknutí tlačítka. Zařízení, která fyzické tlačítko nemají, zobrazí menu po stisku tří teček v ActionBaru (viz obr. 3.2).

Nastavení pro aplikaci obstarává třída *MyPreferencesActivity* rozšiřující třídu *PreferenceActivity*. Vzhled a funkcionalita pro nastavení jsou uloženy v XML souboru, který se poté do aktivity načítá pomocí metody *addPreferencesFromResource()*. Tato metoda je od Androidu 3.0 (Honeycomb) označena jako zastaralá (deprecated). Je doporučeno využít *PreferenceFragment*, který má stejnou funkcionalitu, ale je založen na principu fragmentů.

Podpůrný balík pro Android (viz Zpětná kompatibilita 3.3.1) neobsahuje třídu *PreferenceFragment*, která není podporována na zařízeních s operačním systémem Android verze nižší než 3.0. Jednou z možností, jak se vyhnout použití zastaralé aktivity na novějších zařízeních, je rozhodnout se na základě nainstalované verze systému, jestli použít fragment nebo aktivitu. Pro zobrazení nastavení, by kromě třídy správce fragmentů *SupportFragmentManager* (dodávané s podpůrným balíkem) bylo také zapotřebí využívat třídy systémového správce fragmentů *FragmentManager*. Po důkladném zvážení byla použita aktivita *PreferenceActivity* samotná.

3.8 O aplikaci

Informace o tvůrci aplikace, její aktuální nainstalované verzi a licenčních ujednáních jsou zobrazeny pomocí aktivity *AboutActivity*. Aktivita „O aplikaci“ stejně jako aktivita „Nastavení“ se spouští z menu aplikace. Aktivita obsahuje požadované licenční ujednání týkající se slovníků JMDict (viz 2.1.1) a Kanjidic2 (viz 2.2.1).

V licenčních ujednáních bylo zapotřebí vytvořit odkazy na internetové stránky slovníků. Je využita třída *Linkify* a pomocí regulárních výrazů cílených na jednotlivé části ujednání jsou přidány odkazy na internetové stránky. Odkazy se otevírají v základním prohlížeči přístroje.

3.9 Překlad aplikace

Aplikace plně využívá možností poskytovaných systémem Android pro výběr jazykové verze aplikace. Všechny textové informace zobrazené uživateli jsou uloženy v samostatném XML souboru, ze kterého jsou pak načítány nebo odkazovány v aplikaci. Volbu jazykové verze provádí systém Android na základě právě zvoleného jazyku v zařízení. V případě nepřítomnosti systémem požadované jazykové verze se jako základní jazyk aplikace zvolí angličtina.

V současné době aplikace poskytuje dva jazyky překladu. Jedná se o angličtinu a češtinu. Další jazykové verze je možno vytvořit přeložením jazykových souborů pro angličtinu nebo češtinu a přiložením k projektu.

4 Závěr

Výsledkem bakalářské práce je funkční aplikace pro operační systém Android realizující japonský slovník. Aplikace využívá slovníků JMDict (viz 2.1) a Kanjidic2 (viz 2.2). Před započetím vývoje bylo zapotřebí seznámit se se zvyklostmi při vývoji aplikací pro platformu Android. Bylo navrženo a implementováno zpracování (viz 3.2) a následné vyhledávání ve slovnících (viz 3.4.4). Zdrojové kódy aplikace jsou uloženy v repozitáři dostupném na adrese: `https://github.com/xklech/japanese_dictionary`.

V textu byly popsány jednotlivé funkční celky aplikace a práce s nimi. Dále jsou v práci popsány klíčové technologie a důvody, proč byly vybrány, a také v čem spočívají jejich výhody či nevýhody. Je zde rozebrána práce s vyhledávácím strojem Apache Lucene (viz 3.2.1), který provádí indexování a vyhledávání ve slovnících. V neposlední řadě jsou v textu popsány problémy při používání japonských znaků a jejich převod z latinské abecedy s využitím Hepburnovy transkripce.

Výsledná aplikace má konstantní uživatelské rozhraní na různých zařízeních a reaguje pohotově i na méně výkonných mobilních telefonech. Aplikace by se dala rozšířit o propojení s online slovníkem pro možnost využití alternativních překladů. Jako další možností rozšíření je přidání ukázek použití vyhledaných výrazů ve větách. Další oblast vývoje by mohla být bližší interakce aplikace se systémem Android a možnost začlenění slovníku do ostatních aplikací. Aplikaci by bylo možné doplnit o sdílení vybraných překladů a propojení se sociálními sítěmi.

Literatura

[1] ABLESON, W. *Android in action.* 3rd ed. Greenwich, Conn.: Manning, c2012, xxviii, 632 p. ISBN 16-172-9050-5.

[2] ActionBarSherlock. WHARTON, Jake. *ActionBarSherlock* [online]. 2012 [cit. 2013-04-05].
Dostupné z: `http://actionbarsherlock.com/`

[3] Apache Lucene. *Apache* [online]. 2011 [cit. 2013-04-12].
Dostupné z: `http://lucene.apache.org/`

[4] ELECTRONIC DICTIONARY RESEARCH AND DEVELOPMENT GROUP GENERAL DICTIONARY LICENCE STATEMENT. *EDRDG* [online]. 2000 [cit. 2013-04-12].
Dostupné z: `http://www.edrdg.org/edrdg/licence.html`

[5] Fragments. *Android Developers* [online]. 2013 [cit. 2013-04-05].
Dostupné z: `http://developer.android.com/guide/components/fragments.html`

[6] Introducing JSON. *JSON* [online]. 2013 [cit. 2013-04-05].
Dostupné z: `http://www.json.org/`

[7] JMdict DTD. BREEN, Jim. *JMDict* [online]. 1999 [cit. 2013-04-28].
Dostupné z: `http://www.csse.monash.edu.au/~jwb/jmdict_dtd_h.html`

[8] JMDict sample. BREEN, Jim. *JMDict* [online]. 2008 [cit. 2013-04-08].
Dostupné z: `http://www.csse.monash.edu.au/~jwb/jmdict_sample.html`

[9] KANJIDIC2 HOME PAGE. BREEN, Jim. *KANJIDIC2* [online]. 2008, 2012 [cit. 2013-04-01].
Dostupné z: `http://www.csse.monash.edu.au/~jwb/kanjidic2/`

[10] KANJIDIC2 XML EXAMPLE. BREEN, Jim. *KANJIDIC2* [online]. 2008 [cit. 2013-04-28].
Dostupné z: `http://www.csse.monash.edu.au/~jwb/kanjidic2/kd2examph.html`

[11] Support Library. *Android Developers* [online]. 2013 [cit. 2013-04-05].
Dostupné z: `http://developer.android.com/tools/extras/support-library.html`

[12] The JMDict Project. BREEN, Jim. *JMDict* [online]. 1999 [cit. 2013-03-25].
Dostupné z: `http://www.csse.monash.edu.au/~jwb/jmdict.html`

[13] THE KANJIDIC2 DTD. BREEN, Jim. *KANJIDIC2* [online]. 2008 [cit. 2013-04-29].
Dostupné z: `http://www.csse.monash.edu.au/~jwb/kanjidic2/kanjidic2_dtdh.html`

Seznam obrázků