ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA ELEKTROTECHNICKÁ

Diplomová práce

Implementace pokročilé vstupní metody pro platformu Android

Bc. Jiří Bruchanov

Vedoucí práce: Ing. Tomáš Novotný

Studijní program: *Elektrotechnika a informatika strukturovaný magisterský*
Obor: *Výpočetní technika (magisterský)*

Praha 2011

Poděkování

Tímto bych chtěl poděkovat Ing. Tomáši Novotnému, vedoucímu diplomové práce, za všechny podněty a hodnotné připomínky k vylepšení, vstřícnost a ochotu při vypracování této diplomové práce.

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady v přiloženém seznamu.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 11. května 2011 ……

Abstract

This thesis deals with advanced input method removing disadvantages of current solutions. Document is split into several categories. Firstly, describes analysis and design of problematic parts. Then, it introduces comparison of the existing solutions and shows disadvantages in problematic parts. After, describes implementation of virtual keyboard and special input method. Finally, describes user testing and presenting thoughts of future development.

Abstrakt

Diplomová práce implementuje pokročilou vstupní metodu odstraňující nedostatky současných vstupních metod. Práce je rozdělena do několika kategorií. První část je zaměřena na analýzu a návrh řešení zmiňovaných problémů. Další porovnává existující řešení, kde jsou uvedeny nedostatky z pohledu analýzy. Dále se zabývá detaily samotné implementace, implementací speciální vstupní metody a zhodnocení testování. Na závěr jsou uvedeny myšlenky, jakým způsobem by mohl vývoj dále pokračovat.

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

1 ÚVOD

Tato práce se zabývá implementací pokročilé vstupní metody zadávání dat, neboli implementací virtuální klávesnice, která umožňuje uživateli jednoduše a rychle psát text pomocí dotykového displeje. Aplikace je tedy zaměřena primárně pro mobilní zařízení, která neobsahují fyzickou klávesnici. Aplikace je určena pro platformu Android.

1.1 Struktura dokumentu

- První kapitolou je úvod, specifikace cíle a motivace. Seznámení s vlastnostmi českého jazyka a s platformou Android.
- Druhá kapitola je věnována analýze a návrhu řešení pro důležité prvky při implementaci virtuální klávesnice.
- Třetí kapitola se zabývá porovnáním 3 existujících řešení a porovnání vůči analýze.
- Ve čtvrté kapitole je popis a řešení samotné implementace klávesnice, slovníku a speciální metody zadávání.
- Pátá kapitola popisuje testování.
- Šestá kapitola jako závěr hodnotí implementaci a snaží se nastínit možnou cestu budoucího vývoje této virtuální klávesnice.

1.2 Cíl a motivace

Cílem této práce je vytvořit funkční implementaci vstupní metody zadávání dat, odstraněním nedostatků současných řešení, případně vytvořením nových prvků užitečných pro psaní textu tak, aby byla velmi dobře použitelná mezi širokým spektrem uživatelů.

Mnohá současná řešení poskytují obdobné funkční prvky, ale nejsou vždy ideální pro použití v kombinaci s českým jazykem. Ať už je to psaní s diakritikou, nebo velikostí použitého slovníku, který je určen pro nápovědu při psaní. Moje implementace se tak snaží tyto nedostatky řešit.

1.3 Český jazyk

Čeština je západoslovanský jazyk patřící do rodiny indoevropských jazyků. Čeština je flektivní jazyk, který je typický pro systém skloňování a časování. Flektivní jazyk vyjadřuje gramatické funkce pomocí skloňování, časování, předpon a přípon. Pracuje s řadou vázaných morfémů. Morfém je nejmenší viditelná část slova, která je nositelem věcného nebo gramatického významu. Může to být předpona, vpona, přípona, koncovka nebo kořen slova. Z těchto důvodů je velmi složité určit alespoň přibližný počet slov českého jazyka. Nicméně se odhaduje cca 300 000 slovních kořenů. Toto číslo však nepojímá slova nespisovná, vulgarismy, které výsledné číslo ještě navýší [18].

Čeština používá upravenou latinku, doplněnou o tyto znaky s diakritikou:

Tab. 1 Znaky českého jazyka s diakritikou

Velké	Á	Č	Ď	É	Ě	Í	Ň	Ó	Ř	Š	Ť	Ú	Ů	Ý	Ž
Malé	á	č	ď	é	ě	í	ň	ó	ř	š	ť	ú	ů	ý	ž

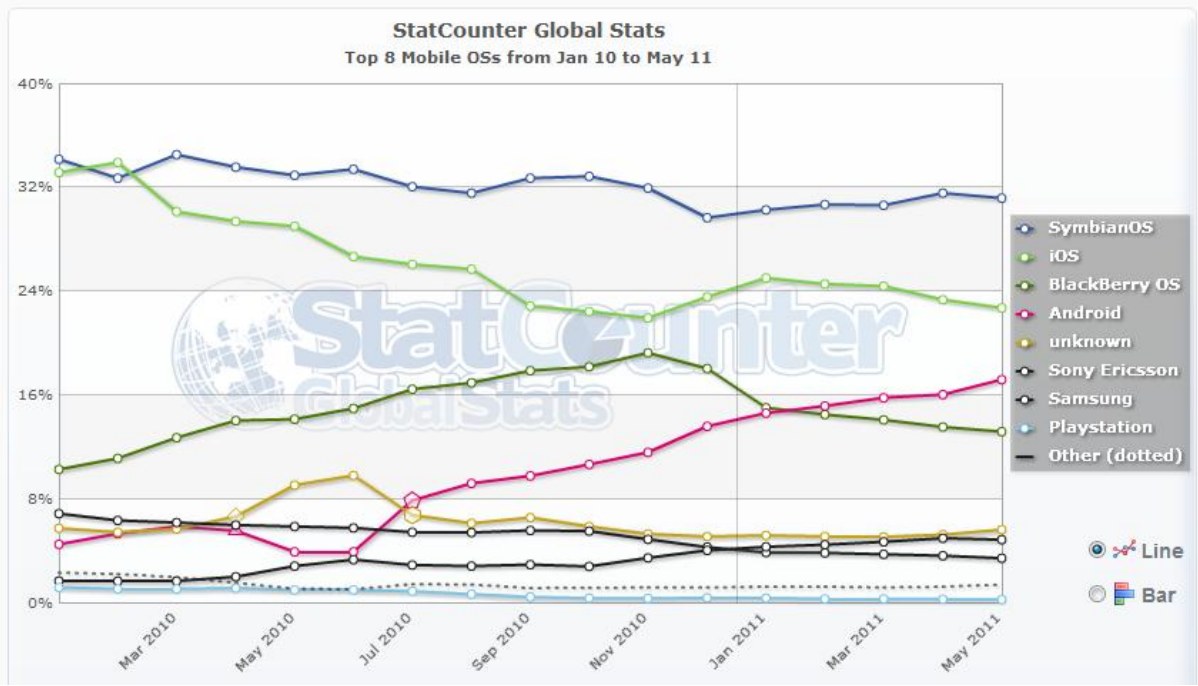
1.4 Platforma Android

Android je softwarová platforma primárně určená pro mobilní zařízení (mobily, tablety), která byla vyvíjena společností Android Inc., kterou Google akvizicí převzal v roce 2005. Platforma Android byla ohlášena koncem roku 2007 a počátkem roku 2008 byla vydána první veřejně dostupná verze platformy, která je šířená pod licencí Apache a GPL v2. Jedná se tak o open source software [17].

Vývoj aplikací je umožněn díky Android SDK. Ten umožňuje vývojářům využívat syntaxi jazyku Java s využitím knihoven od Googlu. Ačkoliv je Android postaven na jazyce JAVA, nevyužívá žádných ze standardů Java Standard Edition, případně Java Mobile Edition. Virtuální stroj (VM) pro běh kódu napsaných v tomto jazyce se nazývá Dalvik. Ten má odlišnou architekturu, proto principiálně nelze spouštět standardní byte code Javy. Dalvik VM vychází z podmnožiny knihoven projektu Apache Harmony, což je open source implementace jazyka Java. To umožňuje nezávislost na původních licencích, které s sebou jazyk Java nese [17].

Android za téměř 2,5 roku urazil obrovský kus cesty na poli mobilních zařízení. Jeho počet uživatelů stále roste, což zobrazuje následující graf.

Obr. 1 Celosvětový podíl mobilních zařízení na trhu 1. 1. 2010 – 1. 5. 2011



Zdroj: [15]

Jako jediný velký hráč na trhu platforem pro mobilní zařízení si udržuje vzrůstající tendenci od počátku roku 2010. Pokles v období duben až červen 2010, který kopíruje nárůst křivky pro neznámé operační systémy, byl způsoben chybou v aplikaci Android Market, která slouží k detekci počtu zařízení. Mnohá zařízení s platformou Android tak byla zařazena právě do skupiny neznámých zařízení. Pomalu ale jistě si ukrajuje větší a větší kus z koláče uživatelů [15].

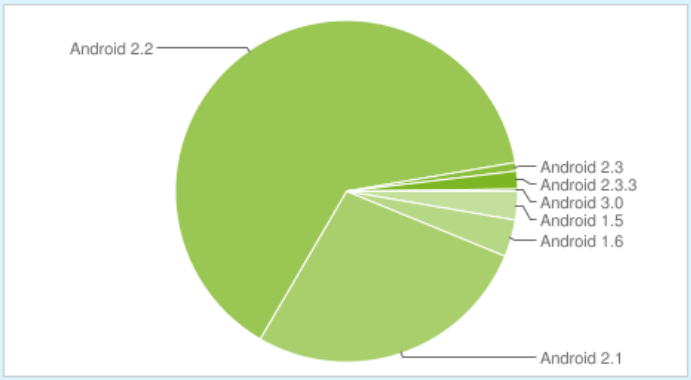
Android je v tuto dobu částečně rozdělen na 2 vývojové větve.

- Řada 2 Primárně pro chytré telefony
- Řada 3 Primárně pro tablety

Řada 1 je již v dnešní době spíše historická, uvedené řady z ní samozřejmě vycházejí, ale z následující tabulky je jasně vidět, že počet zařízení s touto řadou je velmi nízký a je jen otázkou času, kdy vymizí úplně.

Podíl jednotlivých verzí Androidu ukazuje následující tabulka.

Obr. 2 Podíl jednotlivých verzí Androidu



Platform	API Level	Distribution
Android 1.5	3	2.7%
Android 1.6	4	3.5%
Android 2.1	7	27.2%
Android 2.2	8	63.9%
Android 2.3	9	0.8%
Android 2.3.3	10	1.7%
Android 3.0	11	0.2%

Data collected during two weeks ending on April 1, 2011

Zdroj: [9]

Z výše uvedeného grafu je vidět, že k 1. dubnu 2011 je hlavní doménou pro mobilní zařízení verze 2.2. Novější verze si stále, ať už z jakéhokoliv důvodu, nenašla cestu k uživatelům.

2 ANALÝZA A NÁVRH IMPLEMENTACE

Tato kapitola shrnuje určité pohledy na používání virtuální klávesnice, které považuji za důležité, a proto na ně bude kladen důraz při implementaci.

2.1 Psaní

Psaní textu je hlavní účel implementované klávesnice. Vzhledem k omezenému prostoru displeje, je vidět pouze část klávesnice oproti standardní PC klávesnici. Klávesy musí být dostatečně veliké, aby byly použitelné. Návrh bude proto podobný jako u klasické PC klávesnice. Pro zadávání textu bude plná velikost textové části klávesnice, pro zadávání číselných hodnot bude numerická klávesnice. Díky většímu počtu znaků v českém jazyce je důležité, aby diakritická znaménka byla dobře přístupná, proto v kontextové nabídce jednotlivých znaků budou uvedeny pokud možno všechny existující znaky s diakritikou, nikoli jen české.

Dalším důležitým prvkem psaní textu jsou interpunkční znaky. Proto budou tři nejpoužívanější znaky (tečka, čárka, otazník) obsaženy v základním uspořádání kláves.

2.2 Ovládání

Hlavním prvkem ovládání mobilního zařízení je dotykový displej. Dalšími prvky pro ovládání jsou např. tzv. gesta. Jedná se o „namalování“ určitého obrazce, který může spustit určitou proceduru, případně napsat znak nebo slovo.

Způsob ovládání je proto vhodný přizpůsobit tak, aby nebylo nutné zbytečně klikat. Všechny ovládací prvky, které to budou z logiky věci umožňovat, naimplementuji tak, aby bylo možné iniciovat proces ovládání dotykem prstu, ovládání tahem prstu a ukončení, resp. potvrzení zvednutím prstu z displeje. Tato analogie mi přijde velmi vhodná, protože urychluje práci.

Ovládací prvek pro nápovědu je dalším důležitým prvkem pro použití, aby byl dobře použitelný. Proto budu implementovat ve formě panelu, jehož obsahem budou slova tak, aby se jich vešlo co nejvíce. V případě, že nabídka slov bude větší, než dokáže panel zobrazit, posuv bude umožněn tahem prstu.

2.3 Vzhled

Vzhled je dalším důležitým prvkem. V tomto případně nemyslím vizuální vzhled, který je spíše subjektivního charakteru, ale rozložení, případně velikost kláves, které je velmi důležité při použití. Platforma Android umožňuje definovat atributy u ovládacích prvků pro zadávání textu, podle kterého lze omezit data na určité typy textů. Jedná se tak například o telefonní čísla, číselné hodnoty apod. Tímto se jasně vymezuje množina znaků, které mohou být tímto ovládacím prvkem přijaty. Na základě toho je vhodné definovat vzhled klávesnice, např. pouze pro zadávání číselných hodnot apod.

Základní vzhled klávesnice bude implementován tak, aby umožňoval přepínání rozložení kláves QWERTY a QWERTZ a dále aby umožňoval měnit velikost kláves, příp. rozestup řádků kláves tak, jak to uživateli přijde vhodné.

Vzhled se bude automaticky měnit dle atributu vstupního pole. Implementovány budou varianty:

- Základní text
- Telefonní číslo, datum, numerická hodnota
- Email
- Web

Klávesy, které budou měnit svojí funkčnost, budou 3. Výše uvedené klávesy pro interpunkční znaky, tj. (tečka, čárka, otazník). V daném kontextu jsou potřeba většinou jiné znaky. Při psaní emailu je vhodnější nabízet znak zavináč namísto znaku otazník, který nepatří mezi povolené znaky emailu. Dále bude existovat speciální multifunkční klávesa, která bude umožňovat uživateli případné ruční přepnutí těchto režimů, která bude zobrazovat aktuální zvolený režim.

Další vzhled bude speciální klávesnice s obdobným rozložením kláves, kde budou definovány speciální znaky. Vzhled pro zadávání numerických hodnot bude implementován obdobně jako má telefonní klávesnice, nikoliv standardní numerická klávesnice. Základní textový vzhled bude mít implementován rozšiřující část klávesnice, která bude nahrazovat první textový řádek standardní PC klávesnice ve formě 2 řádků. První ve formě základních znaků, druhý ve formě znaků, které jsou zpřístupněny pomocí klávesy SHIFT. To znamená, např. pro českou klávesnici, řádek s číselnými hodnotami a řádek s diakritickými znaky. Tato klávesnice bude přístupná opět pomocí tahu prstu. Dalším vzhledem bude speciální typ klávesnice, který bude obsahovat emotikony ve formě ikon, příp. jejich textových reprezentací.

2.4 Slovník a nápověda slov

Nápověda při psaní textu je velmi výhodná funkcionalita, která umožňuje urychlovat psaní textu, opravovat psaný text, např. při překlepu. Díky existenci diakritických písmen, je v češtině další výhoda, a to doplňování diakritiky. Slovník, ačkoliv ne vždy uživateli využívaný, je tak velmi důležitá komponenta vstupní metody zadávání dat. Díky tomu, že český jazyk je flektivní, roste počet všech použitelných kombinací slov do poměrně vysokých čísel.

Například slovo *výhoda*, následující tabulka zobrazuje jeho slovní varianty.

Tab. 2 Varianty slov od slova výhoda

výhod	**výhod**ičkou	**výhod**ném
výhoda	**výhod**ičku	**výhod**nému
výhodách	**výhod**ičky	**výhod**ní
výhodám	**výhod**ně	**výhod**nosti
výhodami	**výhod**ného	**výhod**ností
výhodě	**výhod**něji	**výhod**nou
výhodičce	**výhod**nější	**výhod**ný
výhodiček	**výhod**nějšího	**výhod**ných
výhodička	**výhod**nějších	**výhod**ným
výhodičkách	**výhod**nějším	**výhod**nýma
výhodičkám	**výhod**nějšíma	**výhod**nými
výhodičkami	**výhod**nějšími	**výhod**o
výhodičko	**výhod**nějšímu	**výhod**ou

Jediný význam slova umožňuje kombinaci 39 slov. Tato problematika se rozrůstá ještě více, protože před slova lze vkládat předpony jako např. *ne, nej, od, do*. V případě slova *výhoda* mohu u každé použít předponu „ne“ a u některých „nej“. Celkový počet se pohybuje okolo stovky slovních variant. Z hlediska objemu dat jde o velkou redundanci dat, protože každé slovo obsahuje část „výhod“. Z tohoto hlediska by bylo vhodné zvolit určitý kompresní algoritmus, aby slovník nebyl zbytečně veliký a zároveň zůstala vysoká rychlost vyhledávání.

Základní jazykový slovník bude implementován tak, aby dokázal pojmout co největší množství slov, bude umět prioritizovat nalezené slova podle četnosti použití a bude umožňovat editaci uživatelských slov ve slovníku. Bude nezávislý na diakritických znacích, to znamená, že vyhledávání bude možné oběma způsoby. Konfigurace klávesnice bude umožňovat přepínání slovníků, které bude nezávislé na jazykovém prostředí telefonu. Vzhledem k výše uvedenému problému ohledně počtu kombinací slov, bude slovník navržen tak, aby byl dostatečně rychlý, ale neobsahoval všechny kombinace slov tak, jak jsou uvedeny ve výše uvedené tabulce, což bude mít za následek, že slovník bude umožňovat výběr predikovaných slov 2 způsoby. První standardní, kdy bude celé slovo vidět. Druhý kontextový, v němž bude zobrazena pouze část slova a její koncovky budou skryty do kontextu. Koncovky bude možné vybrat pomocí kontextového menu. Existence kontextových hodnot bude odlišena barvou.

Další slovník se bude skládat z kontaktů uložené v mobilním zařízení. Tj. jmen a telefonních čísel tak, aby uživateli byla nabízena slova v případě textového pole pro telefonní číslo. Hledání bude automaticky rozpoznáno podle jména nebo telefonního čísla. Výsledek bude zobrazen ve formě jmen v případě, že kontakt obsahuje více položek, bude takový kontakt barevně odlišen a telefonní čísla budou zpřístupněna vyvoláním kontextového menu.

Další varianta slovníku bude vycházet z emailových kontaktů daného zařízení. V případě textového vstupu pro email budou nabízeny emailové kontakty uložené v mobilním zařízení. Vyhledávání bude 3 stupňové. Na "local-part", tj. na hodnoty uvedené před znakem zavináč, hodnoty určující celý název domény a naposled hodnoty určující domény první třídy (com, net, cz,…).

Poslední slovník bude pro textové pole označené atributem URI. Tento slovník nebude určen k vyhledávání, ale pouze jako agregátor často používaných výrazů při psaní tohoto typu textu. Obsah bude uživatelem editovatelný a možnost definovat parametr pro pořadí, tak aby jej uživatel mohl měnit dle své potřeby.

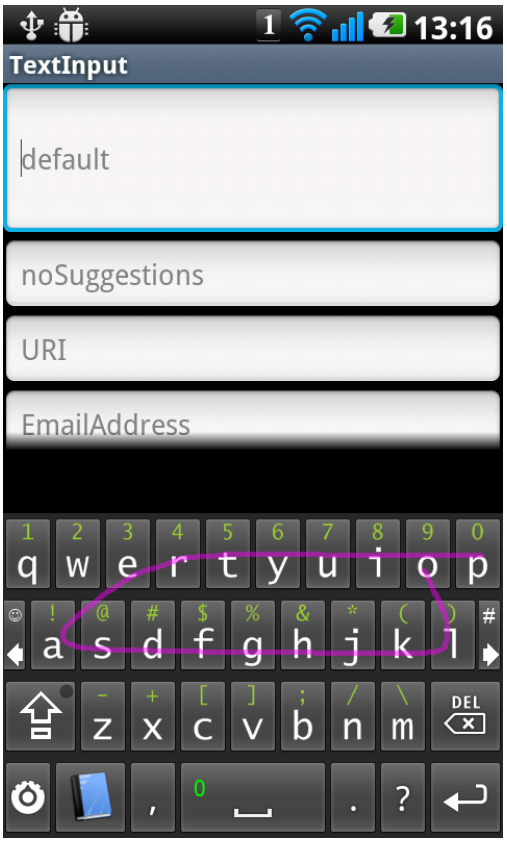
Slovníky se budou automaticky přepínat na základě zvoleného vzhledu klávesnice, případně je uživatel bude moci přepnout nezávisle na použitém vzhledu.

Důležitým prvkem použití slovníku je detekce chyb, proto se budu snažit implementovat tak, aby v případě překlepu nebo chybějícího písmena slovník nabízel uživatelem zamýšlené slovo.

2.5 Speciální metody zadávání textu

Jako speciální způsob zadávání textu se budu snažit implementovat metodu psaní tahem prstu. Ačkoliv je tato metoda alternativní a někteří uživatelé ji zatracují, jiní na ni nedají dopustit. Způsob psaní demonstruje následující obrázek, kde je znázorněn tah prstem přes znaky tak, aby uživatel napsal slovo *prádlo*.

Obr. 3 Příklad psaní pomocí speciální metody zadávání textu



Uživatel tak začne dotykem displeje na klávese P a postupně přes klávesy A,D,L pokračuje v tahu na klávesu O, kde prst uvolní.

Speciální metodu se pokusím naimplementovat tak, aby byla dobře použitelná a využívala navrženého slovníku. Dobrým předpokladem je přesný dotyk prstu na prvním písmenu. Následné hledání výrazných změn úhlu tahu nad klávesou. V jednoduchém případě z výše uvedeného příkladu slova *prádlo* a hledáním výrazných úhlových změn lze rozpoznat znaky PRALO. Kombinací předpokladu prvního správného písmene, výrazných úhlových změn se podařilo téměř správně detekovat všechny obsažené znaky ve slově. Díky tomu máme vymezovací předpoklady pro následný algoritmus, který implementuji pro následné vyhledávání.

Následující dvě uvedené metody, které uvedu, jsou tzv. kreslení gest a zadávání textu mluveným slovem. Obě tyto metody jsou standardně podporovány v Android API a v mé diplomové práci není úkolem implementovat některé z těchto metod. Proto se o nich zmíním jenom okrajově a v následujícím porovnání uvedu jenom, zdali je dané řešení umožňuje.

Kreslení gest spočívá v tom, že existuje předem definovaná množina gest, resp. ručně namalovaných obrazců. V případě, že uživatel nakreslí nějaké gesto na displej, dojde k porovnání a následně k vybrání nejvhodnějšího uloženého gesta. Což může být využito prakticky na cokoliv, psaní znaků, slov, nebo celých vět. Záleží čistě na uživateli, co si pod daným obrazcem uloží.

Mluvené slovo je metodou, která je asi ze všech nejpohodlnější. Implementace, která je poskytnuta v Android API ale zřejmě samotný problém neřeší, protože je nutné připojení na internet. Z tohoto důvodu předpokládám, že výkon dnešních mobilních zařízení není dostatečně silný ke zvládnutí tohoto problému. Toto omezení je tak i poměrně velkou nevýhodou, díky zmíněné nutnosti připojení na internet.

2.6 Ostatní

Do ostatních prvků zahrnuji určité prvky nebo komponenty, které pouze doplňují, příp. rozšiřují funkcionalitu, která nesouvisí přímo s klávesnicí jako takovou, ale obecně s psaním či editací textu. Jedná se např. o překlad textu, nahrazování textu, příp. různé úpravy.

Další prvky, které implementuji do své práce, budou následující:

- Ovládací panel kurzoru, který bude umožňovat jeho pohyb v základním módu, nebo text označovat v selekčním módu.
- Uživatelské šablony. To znamená, že s klíčovým slovem je spojena šablona textu. V případě napsání tohoto klíčového slova bude automaticky nahrazeno touto šablonou.

2.7 Požadavky

Následující body stručně shrnují stanovené požadavky.

2.7.1 Funkční

- Psaní znaků (diakritické, speciální, emotikony)
- Snadné psaní znaků tečka, čárka, otazník
- Změna vzhledu na základě kontextu
- Rozložení kláves QWERTY, QWERTZ
- Logika ovládání tahem prstu
- Slovní nápověda
- Definice textových šablon
- Ovládací panely (kurzor, překlad)

2.7.2 Nefunkční

- Platforma Android

3 SOUČASNÁ ŘEŠENÍ

V této kapitole porovnám vybrané virtuální klávesnice, které již existují a dají se používat a jsou šířeny zdarma nebo za peníze.

Pro porovnání jsem vybral následující hotová řešení.

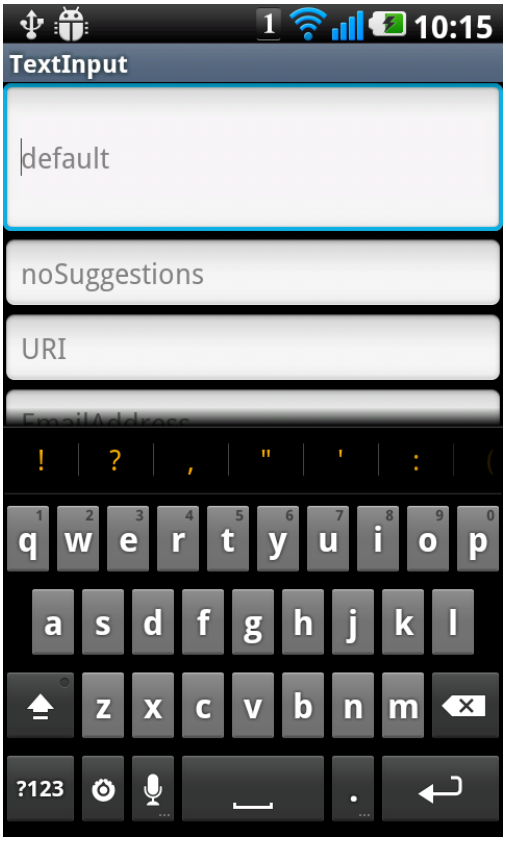
Název	**Šířeno**
Gingerbread keyboard (Výchozí klávesnice pro platformu Android 2.3)	zdarma
UltraKeyboard (trial verze)	komerčně
SlideIT (trial verze)	komerčně

Parametrů k porovnání je mnoho, já provedu porovnání na základě své analýzy z předchozí kapitoly.

3.1 Gingerbread Keyboard

Gingerbread keyboard je implementací od firmy Google. Jedná se o výchozí implementaci vstupní metody zadávání dat pro platformu Android verzi 2.3. Ačkoliv je výchozí, většina prodávaných telefonů jí zřejmě obsahovat nebude, protože prodejci si částečně modifikují jádro platformy Android a dodávají zařízení s vlastní implementací klávesnice. Gingerbread keyboard však jde stáhnout zdarma z tzv. Android marketu. Tuto verzi budu porovnávat, protože momentálně nevlastním žádné mobilní zařízení s verzí platformy 2.3. Následující obrázek zobrazuje prostředí platformy Android, kde je vidět zmiňovaná klávesnice.

Obr. 4 Gingerbread klávesnice



3.1.1 Psaní

Obecné psaní textu je bezchybné. Zcela jistě za výhodu považuji možnost zapsat znak otazník, vykřičník, čárka, které nejsou definovány ve výchozím vzhledu, ale v panelu pro slova nápovědy. Tyto znaky jsou však vidět pouze v případě, kdy není napovídáno žádné slovo, z toho vyplývá, že tyto znaky lze vidět pouze po ukončení psaného slova znakem mezera. Podpora diakritických znaků je z hlediska češtiny dostatečná, protože podporuje všechny diakritické znaky češtiny. Vzhledem k nižšímu počtu kontextových diakritických znaků postrádám možnost volby speciálního znaku, který může případně ušetřit přepínání vzhledů klávesnice do režimu speciálních znaků. Poslední věcí, která je spíše negativní je, že

v případě například klávesy Z je zobrazeno kontextové okno, která nabízí pouze znak *Ž*, což je poněkud zbytečné.

3.1.2 Ovládání

U ovládání je snaha vyhnout se analogii z PC, klikání na klávesy, ovládání je řešeno tahem prstu a dokončení procesu zvednutím prstu z displeje. Tento způsob ovládání je velmi příjemný oproti analogii z desktopového PC a každopádně urychluje psaní.

3.1.3 Vzhled

Vzhled využívá klávesnice standardní QWERTY. Ačkoliv nejsem zastáncem českého rozložení kláves ve stylu QWERTZ, spousta uživatelů je na toto rozložení zvyklá a může jim dělat problém s psaním. Proto považuji toto za poměrně zásadní problém pro použitelnost.

Na speciální atribut vstupního textového pole, který definuje typ požadovaných dat, klávesnice reaguje změnou 1 klávesy, kde je znak @ pro psaní emailů, případně / pro zadávání webové adresy. Tato funkčnost je jistě potěšující, bohužel mi např. při psaní webové adresy chybí znak '-', který je poměrně často používaný.

3.1.4 Slovník a nápověda slov

Vzhled nápovědy slovníku je bez připomínek. Díky proprietárnímu formátu slovníku, lze počet slov jenom těžko odhadovat. Kontextová nápověda zde téměř chybí. Ačkoliv je vidět, že klávesnice využívá slov z kontaktů, tj. nabízí jména kontaktů, nenabízí však např. maily nebo telefonní čísla. Implementace slovníku počítá i s překlepy. Například při psaní slova *autimobil* nabízí správně *automobil*, v případě slova *autumobil* nenabízí správnou korekci. Jedná se zde zřejmě o generalizaci znaků do nějaké obecnější abecedy, kde znak o a i je identický, avšak u a o už stejný není.

Hledání slov v případě vynechání znaku zde funguje také, při zadání *atomobil* nabízí správně *automobil*. Implementace slovníku v klávesnici Gingerbread je tak velmi dobrá, jediné negativum bude pravděpodobně počet slov, protože slovník jako takový má velikost cca 700kB.

Jedinou výtkou je zde absence prioritizace používaných slov. Slovník tak neupřednostňuje často používaná slova.

3.1.5 Speciální metody zadávání textu

Speciální metody zadávání tato klávesnice nemá implementovány žádné. Není zde podpora ani psaní tahem přes klávesnici ani podpora gest. Zmínit lze pouze zadávání textu mluveným slovem.

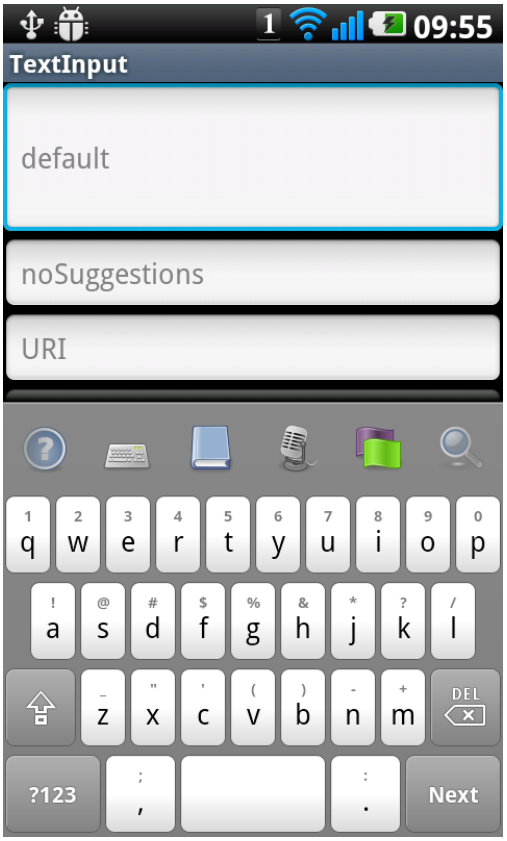
3.1.6 Ostatní

Klávesnice nenabízí žádnou rozšiřující funkčnost, kterou bych mohl uvést.

3.2 UltraKeyboard

UltraKeyboard dle Android marketu spadá do kategorie stažení 10000 – 50000, existují klávesnice i s vyšším ohodnocením, osobně jí však považuji za jednu z nejzdařilejších z hlediska funkcionality, i když na ní lze najít několik nevýhod, jednou z nich je absence českého slovníku. Následující obrázek zobrazuje vzhled UltraKeyboard implementace.

Obr. 5 UltreaKeyboard klávesnice



3.2.1 Psaní

Tato klávesnice také nijak nevybočuje ze standardů při psaní a lze jí také považovat za bezchybnou. I zde, tak jako v předchozím případě, považuji za výhodu existenci interpunkčních znamének tečka a čárka. Pro otazník je nutno vyvolávat kontextovou klávesnici znaku tečka. Pro zpřístupnění všech českých diakritických znaků je potřeba telefon nastavit do českého prostředí. Toto omezení mně příliš nevyhovuje, protože mám telefon ve výchozím jazykovém nastavení v angličtině. A ačkoliv to nebude častý jev, představa konfigurace jazykového prostředí mobilního telefonu pro zpřístupnění nějakého speciálního znaku, mě příliš nenadchla.

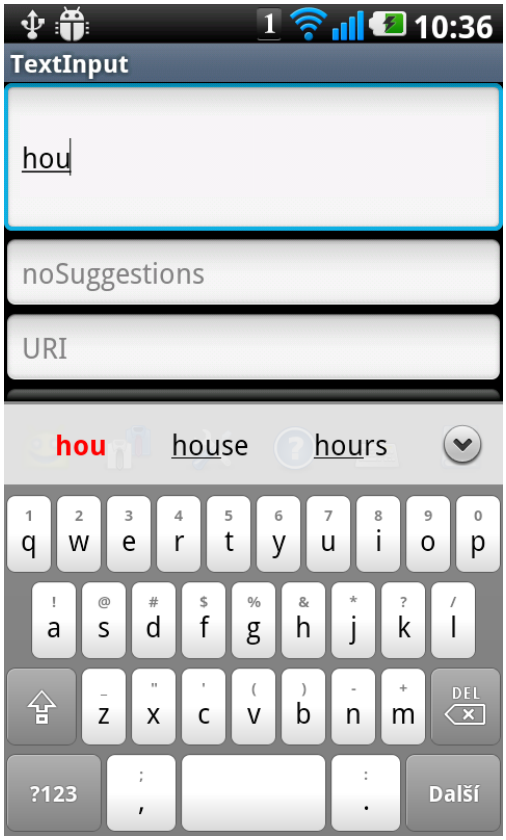
3.2.2 Ovládání

Z výše uvedeného obrázku je vidět, že nad klávesnicí může být zobrazen panel s tlačítky, kterými lze klávesnici konfigurovat, případně vyvolávat další funkcionální prvky klávesnice. Stejně jako v předchozím případě i zde je snaha zjednodušit ovládání tak, že je ovládání přizpůsobeno na tah prstu, nikoliv na „klikací“ analogii z PC. Velkou výhodou této

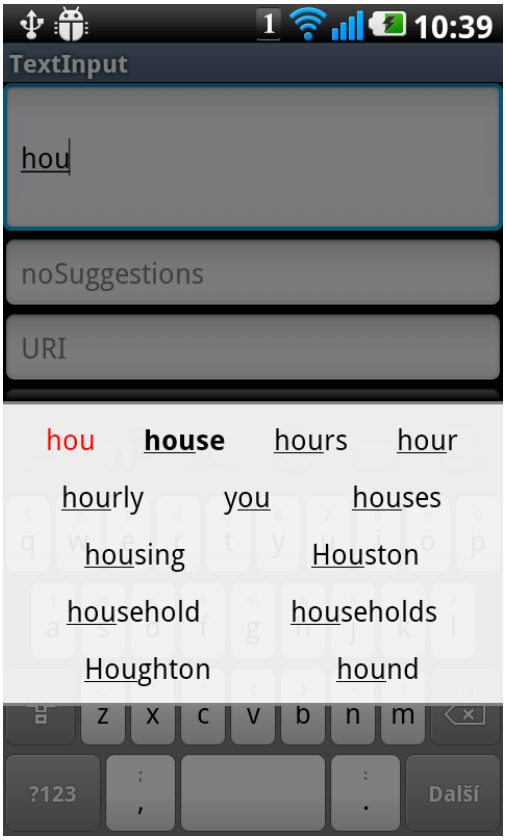
implementaci také považuji vysoký stupeň konfigurovatelnosti, takže si uživatel může z velké části ovlivnit chování klávesnice podle svých představ.

Dalším prvkem ovládání, které chci zmínit, je panel nápovědy slovníku, který je nepříliš vhodně zvolený. Jednak není ovládání přizpůsobeno na styl tahem prstu a druhou nevýhodou z mého pohledu je spíše upřednostňování vzhledu před funkcionalitou. Myslím si, že je lepší uživateli nabídnout největší počet možných variant. Následující obrázky ukazují použití panelu nápovědy při psaní slova *hourly*.

Obr. 6 UltraKeyboard - základní panel nápovědy



Obr. 7 UltraKeyboard - rozšířený panel nápovědy

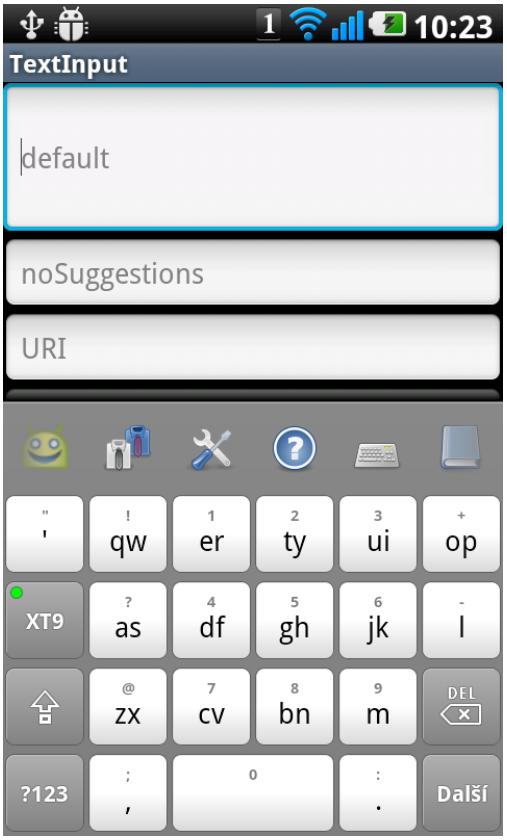


Panel tak nabízí aktuální psané slovo a 2 predikovaná. Po kliknutí na pravé tlačítko se šipkou se panel zvětší a jsou nabízena další predikovaná slova, která jsou díky zvolenému formátování poměrně chaoticky uspořádána. Zvýšením počtu nabízených slov se tento problém ještě prohlubuje.

3.2.3 Vzhled

Výchozí vzhled pro klávesnici po instalaci je kompaktní. Je podobný klávesnici, která je známa z dřívějších dob pro psaní pomocí režimu T9, který znázorňuje následující obrázek.

Obr. 8 UltreaKeyboard kompaktní klávesnice



Dále nabízí módy QWERTY, QWERTZ, AZERTY, což jistě uspokojí velkou většinu uživatelů. AZERTY však není implementována standardně, jsou zaměněny klávesy Q s A, W s Z, není však přesunuta klávesa se znakem M do pozice napravo od klávesy se znakem L. Klávesnice také umožňuje měnit velikost kláves, která je výhodná zvláště pro majitele zařízení s menším displejem. Uživatel si tak může nastavit velikost tak, jak mu vyhovuje pro psaní, případně pro dané okolnosti, kdy potřebuje vidět větší část používané aplikace nebo upřednostňuje pohodlnější psaní.

Klávesnice reaguje na atribut definující typ vstupního pole a přepíná automaticky vzhled podle typu a mění 2 znaky, resp. klávesy, případně celý vzhled pro zadávání numerických hodnot nebo telefonních čísel

3.2.4 Slovník a nápověd slov

Jak jsem již zmínil v úvodu, porovnání této klávesnice chybí zcela český slovník. Budu tak porovnávat nabídku anglických slov. Slovník bohužel také není v otevřeném formátu, proto počet slov lze pouze odhadovat. Pozitivním prvkem slovníku je jeho upřednostňování často používaných slov, která jsou pak podle toho řazena při predikci. Uživateli tak mohou být nabídnuta v menší velikosti panelu nápovědy. Bohužel, jak jsem zmínil v předchozím

odstavci, pouze 2 slova jsou nabízena v kompaktním režimu panelu nápovědy. Při zobrazení celého panelu jsou slova řazena opět dle počtu použití. Slova nepoužívaná zřejmě podle interního indexu. Zde bych spíše volil variantu abecední, protože například slovo *hound* bylo umístěno „někde“ uprostřed množiny slov a nebylo příliš pohodlné jej hledat. Slovník počítá i s překlepy, případně s vynechaným znakem, takže při slovu *hone* správně nabízí slovo *home*.

Slovník jako v předchozím případě využívá pro nápovědu slova uvedená jako jména v telefonním seznamu, nabízeny jsou i částečně maily. Bohužel však pouze tzv. "local-part" část, tedy ta, co je před znakem zavináč. Při vstupu na textové pole s definicí vstupu email se predikce zcela vypíná a není nabízeno nic, což je škoda. Stejně tak při psaní telefonních čísel se nápověda zcela vypíná.

3.2.5 Speciální metody zadávání textu

UltraKeyboard implementuje speciální metodu psaní pomocí tahu prstu. V případě zodpovědného tahu prstu zde není co dodat, vše funguje velmi dobře a klávesnice je použitelná. V případě rychlého a nedbalého tahu prstu pro slovo *assertion* se mi nikdy nepodařilo zajistit, aby se toto slovo dostalo mezi první 2 predikovaná slova. Z 10 pokusů se však 7x objevilo v dalších predikovaných slovech. 70% úspěšnost je tak velmi dobrá. Klávesnice umožňuje pozdržením tahu prstu potvrdit danou klávesu, tímto uživatel může zpřesnit hledaný výraz. Pro slovo *assertion* je to užitečné z hlediska tahu prstu, protože znaky ERTIO jsou všechny v 1. řádku a jdou za sebou při tahu prstu. Ačkoliv je toto jistě užitečný způsob pro heuristiku algoritmu, pro slovo *assertion* bylo potřeba min dvou takto potvrzených znaků. Toto potvrzovaní však poměrně prodlužuje tah prstu, protože minimální doba pro toto potvrzení je nastavitelná na hodnotu 300ms.

Jedinou výtkou k implementaci metody psaní tahu prstem je poněkud horší detekce tahu prstu oproti standardnímu vyťukávání kláves, protože při rychlém psaní dochází k inicializaci této metody. To má za následek, že dochází k nesmyslnému zadávání slov.

3.2.6 Ostatní

I tato klávesnice podporuje zadávání textu hlasem. Další využitelná funkčnost této klávesnice je podpora překladů, kde jednoduše pomocí některé z existujících webových služeb odešle text k překladu, což samozřejmě vyžaduje existenci připojení k internetu. Zajímavou možností na mě působilo to, že lze definovat tzv. zakázaná slova. Při psaní takového slova je uživatel upozorněn v panelu s nápovědou, že toto slovo patří mezi definovaná zakázaná slova.

Klávesnice také umožňuje nahrazování slov. To znamená, že lze definovat nějakou zkratku, která je pak nahrazena podle definované šablony. Lze si tak tedy nadefinovat často používané věty nebo slovní kombinace jako jméno a příjmení apod.

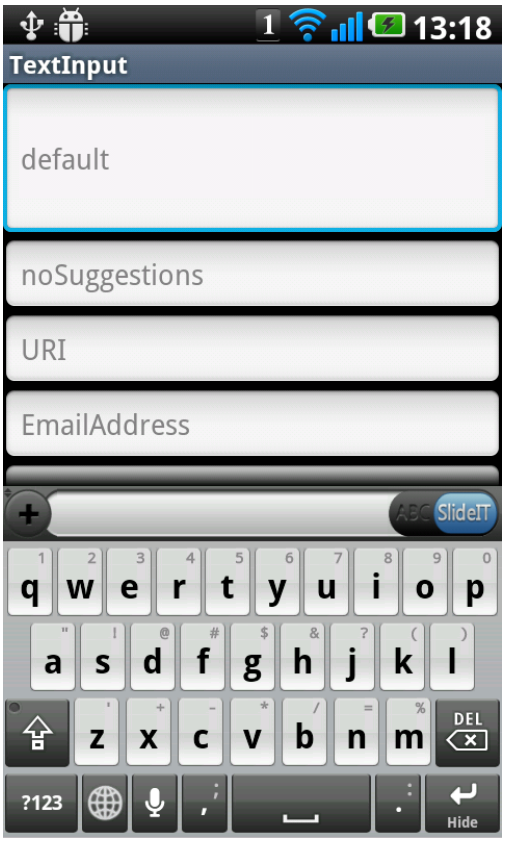
UltraKeyboard má klávesy pro práci se schránkou, takže příkazy COPY, CUT, PASTE jí nejsou cizí. Zajímavý je ovládací prvek, který simuluje tzv. DPad, pomocí kterého je možno klikáním na tlačítka pohybovat kurzorem, což je zvláště užitečné u zařízení, které tímto ovládacím prvkem nedisponují. Zajímavý je také prvek, který dokáže měnit velikost psaného textu, což můžou využít uživatelé se zhoršeným zrakem.

Další zajímavou funkcionalitou u této klávesnice je podpora psaní při chůzi využitím kamery ze zadní strany telefonu. Při tomto módu se zobrazí částečně průhledná klávesnice a jako pozadí je promítán obraz z kamery. O použitelnosti tohoto módu lze polemizovat, protože já osobně při chůzi a psaní na mobilu nemám ruce předpažené, takže bych viděl akorát chodník, což mi při případné kolizi s předmětem před sebou příliš nepomůže.

3.3 SlideIT

SlideIT je další implementace vstupní metody zadávání dat pro platformu Android, kterou jsem vybral pro porovnání. Je také šířena komerčně. Ačkoliv nemám s touto klávesnicí dlouhodobé zkušenosti, zařadil jsem jí do porovnání také z důvodu, že má implementovanou speciální metodu zadávání textu tahem prstu, které je už patrné z názvu. Vzhled klávesnice je uvedený v následujícím obrázku.

Obr. 9 SlideIT klávesnice



3.3.1 Psaní

Ani v tomto případě nelze nic vytknout této implementaci pro obecné psaní. Znaky pro ukončování vět jsou ve výchozím vzhledu pouze dva a to znak tečka a čárka. Pro znaky otazník resp. vykřičník je potřeba již vyvolávat kontextové hodnoty tlačítka se znakem J resp. S. Diakritická znaménka jsou obsažena v dostatečném množství pro pokrytí českého jazyka.

3.3.2 Ovládání

Ovládání této klávesnice je nejhorší z testovaných variant. Bohužel neumožňuje ovládání tahem prstu. Horní panel, který je určen pro nabídku slov, je zbytečně zmenšen o boční tlačítka, proto mám i zde dojem, že je spíše upřednostněn vzhled před funkcionalitou. Panel pro nápovědu nemá žádný režim pro větší zobrazení, díky tomu jsou vždy vidět maximálně 4 slova, pro další varianty je nutné posouvat, příp. přepínat slova pomocí bočního tlačítka.

3.3.3 Vzhled

SlideIT podporuje standardní rozložení kláves QWERTY, QWERTZ, AZERTY také i ty méně standardní jako Dvorak nebo Colemak. Podporuje také navíc znakovou sadu Cyrilice. Výhodná je také, jako u předchozí varianty, možnost měnit velikost tlačítek.

Pro speciální varianty textového vstupu klávesnice reaguje změnou vzhledu pouze pro číselné hodnoty. V případě psaní internetové adresy, příp. emailů nedochází k žádné změně a uživatel je tak nucen speciální znaky pro daný kontext použití vyvolávat z kontextových znaků jednotlivých kláves.

3.3.4 Slovník a nápověda slov

Klávesnice SlideIT podporuje variantu českého slovníku a podporuje i velké množství slovníků jiných jazyků. Bohužel se jedná opět o proprietární formát, proto lze počet slov pouze odhadovat. Kontextová nápověda je zde obsažena pouze jako v předchozích případech. Jsou nabízena pouze jména a příjmení. Slovník se i v této variantě učí často používaná slova, a proto je výsledek nápovědy řadí tak, aby tyto slova byla na začátku. Zvláštní je, že predikce slov je funkční pouze v režimu pro metodu psaní tahem prstu, nikoliv však pro normální psaní. Další nevýhodou je neexistence tzv. composing režimu. Jedná se o režim, kdy uživatel vidí psané slovo zvýrazněné podtržením. Je tedy informován, které slovo se snaží napsat. V případě této implementace tomu tak není, což je poměrně matoucí v případě opravy nějakého slova, kdy klávesnice nabízí nové možnosti nápovědy. Pokud ale uživatel začal opravovat slovo např. vymazáním jeho konce, navrhované možnosti jsou tak zcela nesmyslné. Za chybu také považuji fungující nápovědu pro textové pole určené pro hesla. Tato funkčnost sice může být někomu ku prospěchu, ale určitě by měla být nastavitelná tak, aby skutečně nedocházelo k optickému zobrazení hesla a případnému zcizení. Další nevýhodou je rozdílnost slovníků na základě vzhledu klávesnice. Naučená slova pro češtinu v profilu QWERTY se nezobrazují pro češtinu profilu QWERTZ. Naopak výhodná je možnost volby 2 slovníků. V případě, že slova se neshodují s dostatečným počtem variant prvního slovníku, prohledává se i druhý.

Z hlediska překlepů se i tato varianta chová obdobně jako předchozí 2. Např. při psaní slova *autpnobil* nabízí správně *automobil*. Opravování chybějících písmen zde zřejmě implementováno není, protože ani jedna z mnou zkoušených variant nebyla správně opravena.

3.3.5 Speciální metody zadávání textu

SlideIT již z názvu napovídá, že podporuje speciální metodu psaní pomocí tahu prstu. Ani zde není při pečlivém psaní co vytýkat a jak v anglickém, tak i v českém jazyce funguje vše bez problémů. V případě rychlého a nepečlivého tahu prstu v případě anglického *assertion* se slovo objevilo vždy, ať už jako v ihned viditelné nápovědě, anebo v množině dalších. Pro ledabylé psaní českého slova *pouta* jsem dosáhl nalezení 7x z 10 pokusů, což je také velmi slušný výsledek. Vzhledem ale k velkému počtu možných variant, které může toto slovo splnit, předpokládám, že tento i předchozí úspěch jsou kompenzací za nižší celkový počet slov.

Jako v předchozím případě zde musím uvést problém, který vzniká při rychlém psaní, kdy je špatně detekován start procesu psaní tahem a dochází tak k nesmyslnému psaní textu. V případě zkusmo rychlého napsání slova *halenka* proběhne 2x nesprávná detekce a výsledkem je text „*H alen já*“.

Tato klávesnice podporuje ještě zadávání textu hlasem a využití gest. Bohužel zde chybí jejich editor, takže se uživatel musí spokojit pouze s předdefinovanými gesty.

3.3.6 Ostatní

Klávesnice má speciální tlačítka pro práci se schránkou. Podporuje také režim, který zamezuje vložení diakritických znaků, což může být na jednu stranu užitečné, na druhou stranu praktická využitelnost mi uniká, protože psaní bez diakritiky je snazší, neboť není nutné vyvolávat kontextovou nabídku jednotlivých kláves pro diakritické znaky.

4 IMPLEMENTACE

Tato kapitola je věnována implementaci hlavních komponent diplomové práce. Nejprve je uvedena implementace základní klávesnice, poté implementace slovníku a na závěr kapitoly je popsána implementace speciální vstupní metody zadávání dat.

4.1 Použité technologie

4.1.1 SQLite

SQLite je platformě nezávislá knihovna implementující SQL databázový engine. Díky tomu můžeme pro operace nad daty používat SQL jazyk, který umožňuje velmi rychle provádět různé operace jako filtrace, řazení a další výhody plynoucí z použití relační databáze. Jedná se o jeden z nejrozšířenějších databázových enginů na světě z hlediska používání. Používají ho nejen ostatní výrobci platforem pro mobilní zařízení, ale také pro interní použití aplikace jako Mozilla Firefox, Skype a mnohé další [3].

V následujících dvou bodech uvedu několik, z mého pohledu výhod a nevýhod SQLite databáze, avšak pouze s obdobnými produkty. Nemyslím si, že má smysl porovnávat SQLite s databázovými enginy jako Oracle Database apod., protože se jedná o zcela jinou úroveň použití, co do velikosti množství dat a také do robustnosti aplikace. Jak je uvedeno na http://www.sqlite.org, SQLite se nesnaží nahradit Oracle, ale snaží se nahradit metodu *fopen*() [3].

Výhody z mého pohledu

- Platformě nezávislé
- Kompaktní knihovna, nemusí se instalovat (velikost cca 275 kB)
- Rychlé zpracování dat
- Jednoduché použití
- Umožňuje fulltextové vyhledávání
- Triggery
- Transakční zpracování
- Podpora FullText vyhledávání

Výhody, jak je vidí autoři.

- Žádné externí knihovny
- Žádný server
- Žádné konfigurace
- Transakční

Nevýhody

- Typeless
- Zjednodušená implementace normy SQL92

Ačkoliv SQLite umožňuje definici datových typů, bohužel při ukládání neprobíhá kontrola. Z mého pohledu je toto velká nevýhoda, protože tak umožňuje uložit např. textovou hodnotu do pole s definicí číselného typu. Pokud mají být data konzistentní, musí být kontrola typů provedena explicitně programátorem. Databáze má, alespoň podle mého názoru, jeden z úkolů, kontrolovat konzistenci vstupních dat při ukládání.

Implementace SQLite pro platformu Android není úplně kompletní. Chybí např. možnost vytvořit si vlastní funkce pro zpracování při spuštění SQL dotazu, nebo využití vnořených dotazů pomocí operátoru IN.

4.1.2 Full Text vyhledávání

FullTextové vyhledávání je metoda prohledávání textových dat, která má analogii s rejstříkem v knize. Pro rychlejší vyhledání určitého výrazu je lepší vytvořit index (rejstřík) a při hledání se do něj podívat, kde je dané slovo uvedeno. Tímto je dosaženo mnohonásobného zrychlení oproti lineárnímu prohledávání celého textu. SQLite databáze FullTextové prohledávání na platformě android je implementováno pomocí FTS3 modulu.

4.1.3 Paralelní programování

V dnešní době se již začínají objevovat první zařízení, která mají více jádrové procesory, a proto se problematika paralelního programování bude zcela jistě zohledňovat při vývoji nových mobilních aplikací. Vstupní metoda zadávání dat je na platformě android implementována ve formě služby s viditelným prvkem (klávesnicí) a není nijak speciálně přizpůsobena pro využití paralelního zpracování. Ve své implementaci jsem však tuto metodiku využil při hledání slov, které jsou prediktivně nabízeny uživateli. Ačkoliv paralelní zpracování přináší výhody z hlediska výkonu, při vývoji je na to potřeba brát ohled, aby nedocházelo k nedeterministickému chování programu.

4.2 Klávesnice

4.2.1 Základní vztahy

Základem virtuální klávesnice na platformě Android jsou 3 třídy.

- InputMethodService
- KeyboardView
- Keyboard

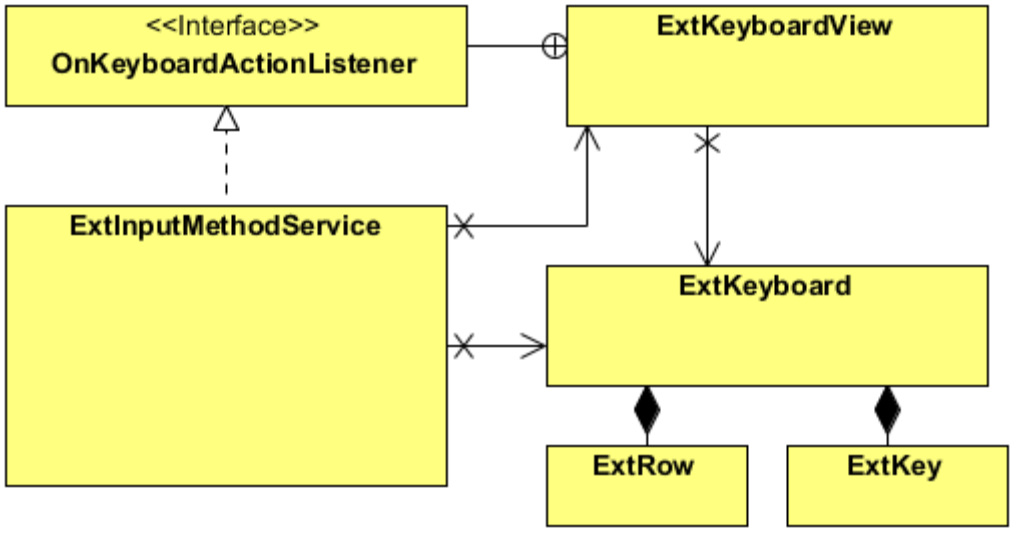
InputMethodService je třída, jejíž instancí je služba běžící na pozadí, která obstarává veškeré procesy spojené s klávesnicí, např. její vytváření, zobrazování, interakci s textovými ovládacími prvky apod.

KeyboardView je třída, která slouží k vykreslování klávesnice na displej, reaguje na vnější události uživatele. Tyto události zpracovává a odesílá zprávy objektu, jenž implementuje rozhraní *KeyboardView.OnKeyboardActionListener*, což je v mém případě právě objekt *InputMethodService*, který na ně reaguje patřičným způsobem.

Keyboard je spíše kontejner definující klávesnici, mnoho funkčnosti tato třída sama o sobě neposkytuje.

Celá diplovomá práce se převážné zabývá rozšiřováním těchto 3 tříd o další funkcionalitu. Následující diagram popisuje vztahy mezi těmito třídami.

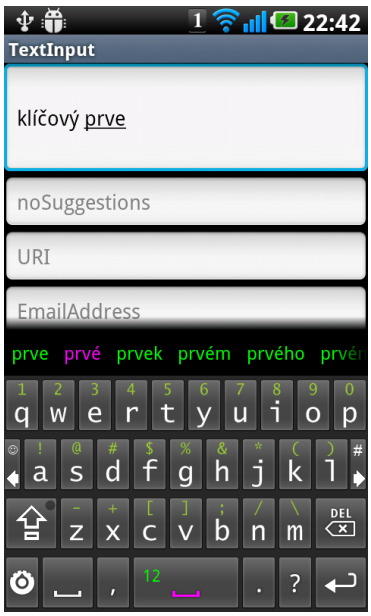
Obr. 10 Základní vztahy objektů



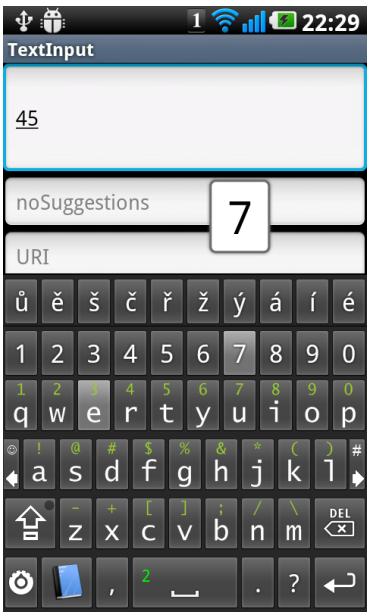
4.2.2 Náhled

Vizuální výsledek diplomové práce částečně prezentují následující 4 obrázky klávesnice, které zobrazují určité módy.

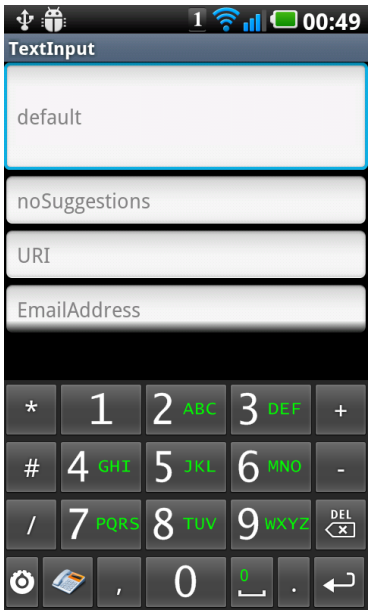
Obr. 11 Nápověda slova



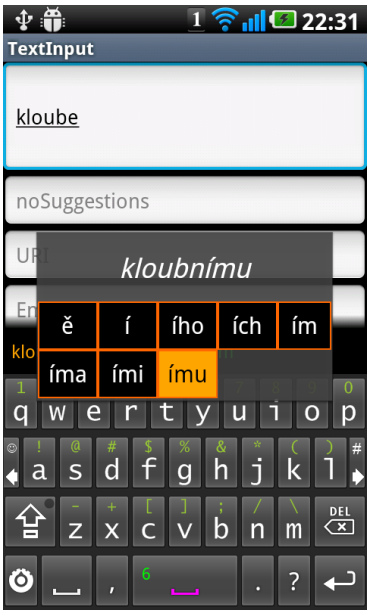
Obr. 12 Rozšiřující část klávesnice



Obr. 13 Vzhled klávesnice pro zadávání numerických hodnot



Obr. 14 Výběr slova pomocí kontextového menu



4.2.3 Implementace

Implementace klávesnice je rozvržena do několika základních stavů, podle atributu vstupního pole, které je klávesnicí právě obsluhováno. Jedná se o následující stavy:

- Obecný text
- Mailová adresa
- URI identifikátor
- Numerický text

Každý z těchto stavů má v určitých situacích odlišné chování. Vzhledem k dalším možnostem konfigurace se tak celkový počet všech možných konfigurací stavů rozrůstá. Z tohoto důvodu jsem implementoval jednotlivé třídy pro každý výše uvedený základní stav.

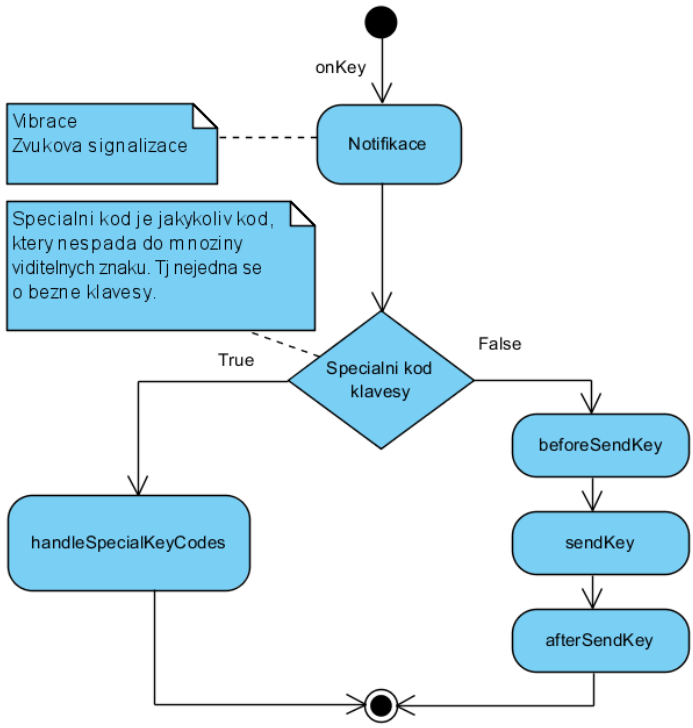
Základní obsluha samotné události kliknutí na klávesu je zobecněna do procesu, který je rozdělen na následujících 5 částí.

Tab. 3 Obsluha události kliknutí na klávesu

1	Spuštění událostí, která jsou shodná pro obecné kliknutí na klávesu. Jedná se např. o vibraci, tónovou odezvu apod. Tato část je spuštěna vždy.
2	Spuštění procesů, které jsou definovány nějak speciálně. Jedná se o události jako přepínání kontextu klávesnice, zmáčknutí např. klávesy SHIFT apod. Jsou to všechny klávesy, které nereprezentují alfanumerické znaky a jiné další speciální znaky, které jsou běžně využívané.
Následující 3 části jsou spuštěny pouze v případě, když nebyla spuštěna 2. část	
3	Událost, která se spouští před odesláním znaku textovému poli, tj. před tím, než se stane viditelným.
4	Samotné odeslání znaku textovému poli. To znamená, že v tomto bodě se znak stává viditelným v textovém poli.
5	Poslední částí je operace, která je spuštěna po zobrazení znaku v textovém poli.

Následující diagram zobrazuje výše popsaný proces.

Obr. 15 Obsluha události kliknutí na klávesu

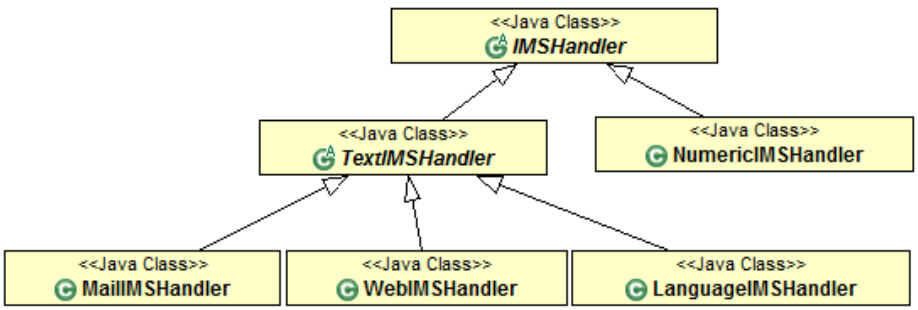


Jednotlivé třídy reprezentující výše zmíněné stavy tak převážně specializují metody

- *beforeSendKey*
- *sendKey*
- *afterSendKey*

Další specializací metody, která není zmíněna v diagramu, je *onUpdateSelection*, která slouží k obsluze při změně polohy kurzoru. Např. pro skokovou změnu polohy kurzoru a následné obnovy slova do režimu nápovědy.

Obr. 16 Struktura objektů implementující chování na základě stavů

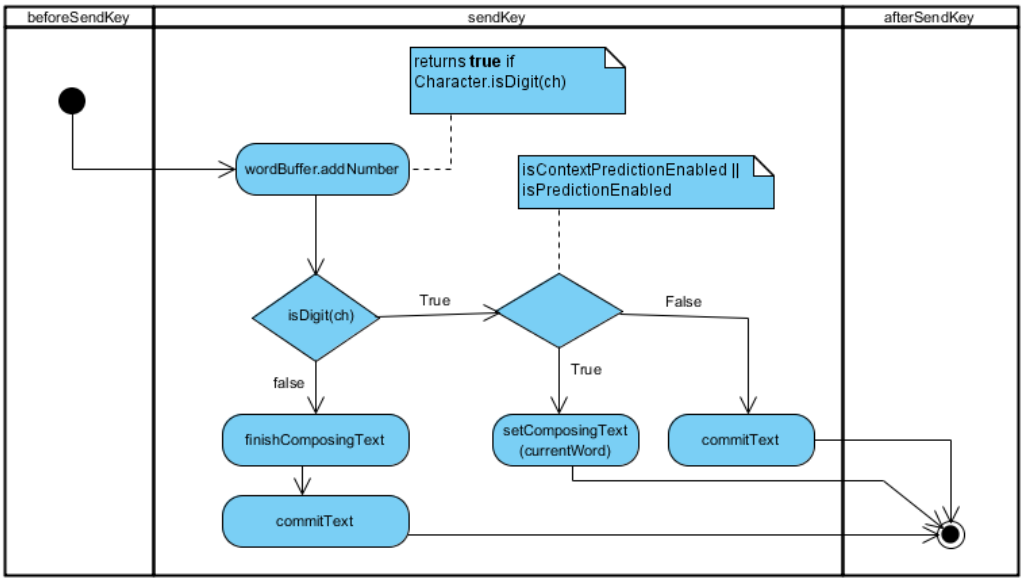


4.2.4 Numerická klávesnice

Numerická klávesnice je zobrazena na *Obr. 13 Vzhled klávesnice pro zadávání numerických hodnot*. Slouží primárně k zadávání numerických hodnot. V případě, že je textové pole označeno pro vstup telefonního čísla, je v horním panelu nabízena nápověda z telefonního seznamu.

V tomto případě je pouze specializována metoda *sendKey*. Samotný proces obsloužení události je zobrazen na následujícím diagramu.

Obr. 17 Obsluha události kliknutí - Numerická klávesnice

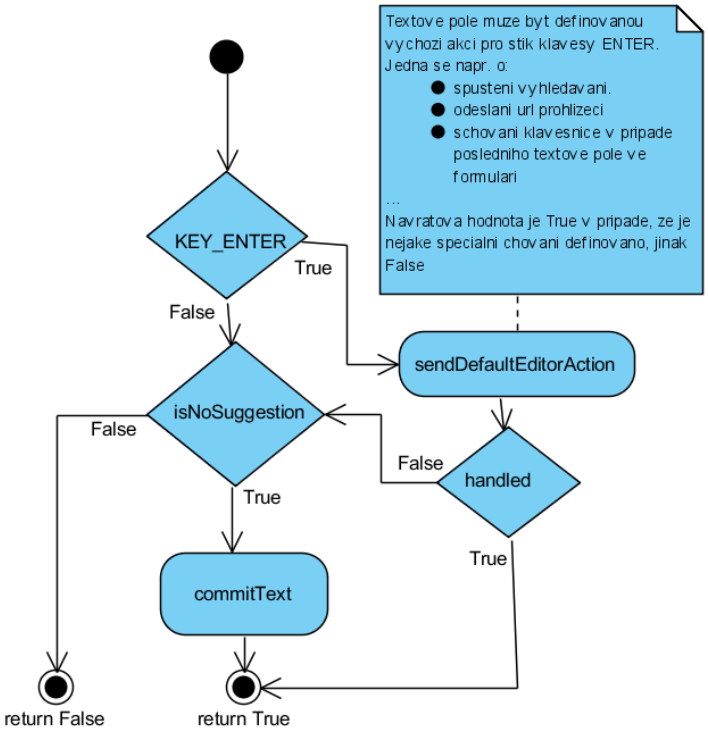


4.2.5 Textová klávesnice

Pro zbylé 3 typy jsem zvolil další stupeň specializace ve formě další třídy. Tato třída je obecně určena pro všechny následující specializace, které jsou určeny obecně pro text.

Tato třída implementuje logiku metody *sendKey*, kterou v tuto chvíli využívají všechny její podtřídy. Následující diagram zobrazuje aktivitu odeslání znaku objektu zprostředkovávající komunikaci s textovým polem.

Obr. 18 Implementace části *sendKey* pro text. klávesnice

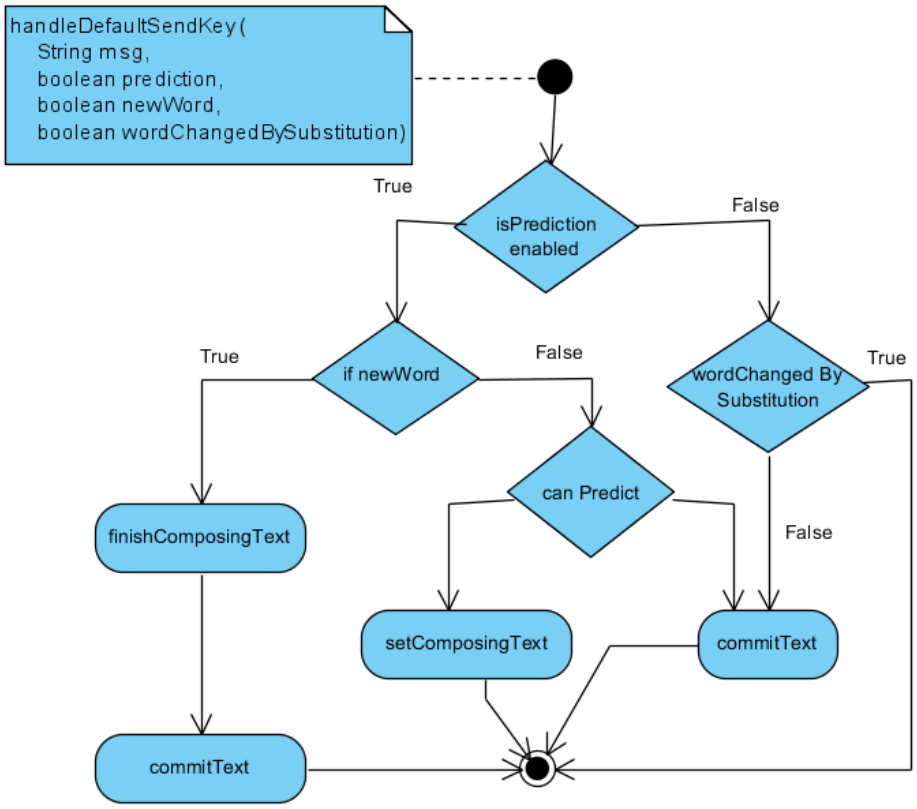


V případě stisku klávesy ENTER je spuštěna metoda *sendDefaultEditorAction*, která může vyvolat určitou definovanou událost, jako např. spuštění vyhledávání, opuštění textového pole, příp. schování klávesnice apod. V takovém případě vrací hodnotu *true*, jinak *false*.

V případě, že textové pole má atribut, který definuje možnost *nenapovídat*, odešle požadovaný znak do textového pole a vrací také hodnotu *true*, jinak *false.* Poté proběhne implementace metody *sendKey,* potomka této třídy.

Logika zobrazení aktuálně psaného znaku, resp. slova, je implementována také v této třídě. Samotné využití je však až v podtřídách, pro které se tato aktivita shoduje. Následující diagram zobrazuje potvrzení současného znaku, resp. znaků (např. v případě emotikon).

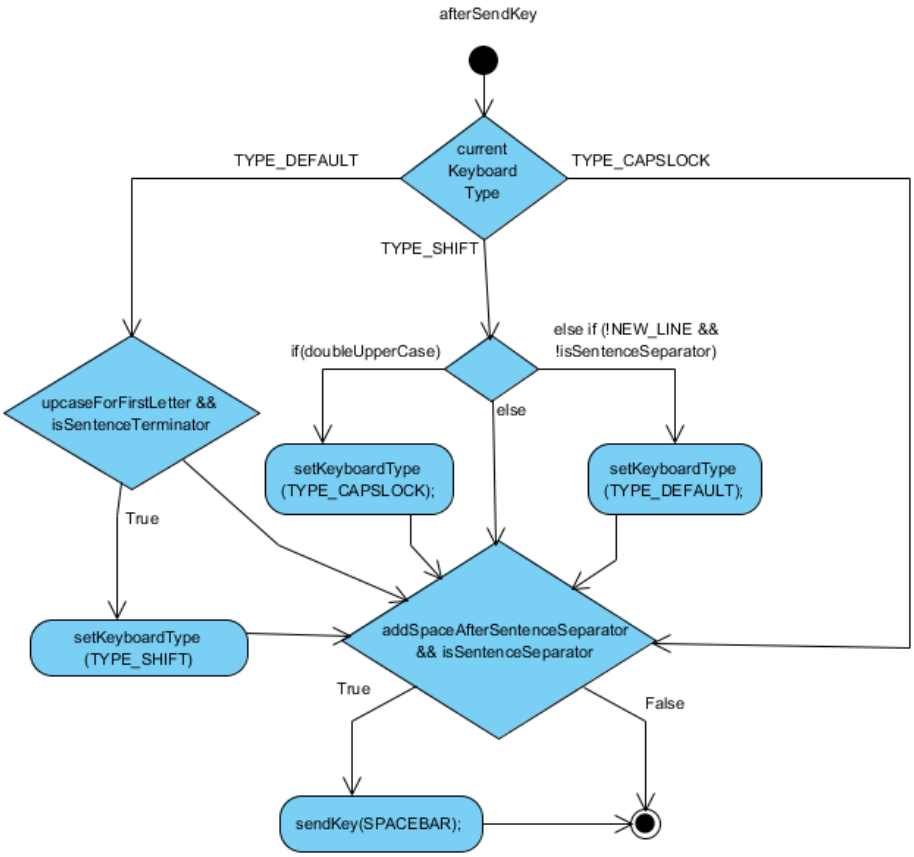
Obr. 19 Diagram odeslání psaného znaku textovému poli



Na základě 3 vstupních parametrů je tak rozhodnuto, jak je s aktuálně psaným znakem naloženo.

Konfigurační parametry, které umožňují automaticky přidávat znak mezera za vybraná slova nápovědy, případně automaticky měnit SHIFT mód klávesnice po stisknutí klávesy se znakem pro ukončení věty apod. obsluhuje metoda *afterSendKey*. Následující diagram znázorňuje metodu *afterSendKey*.

Obr. 20 Implementace metody *afterSendKey* pro obecnout text. klávesnici



Implementace tak na základě aktuálního znaku, resp. 2 posledních znaků, přepíná režimy výchozí, SHIFT, CAPS-LOCK, případně dle uživatelského nastavení přidává navíc za znaky ukončující větu znak mezera.

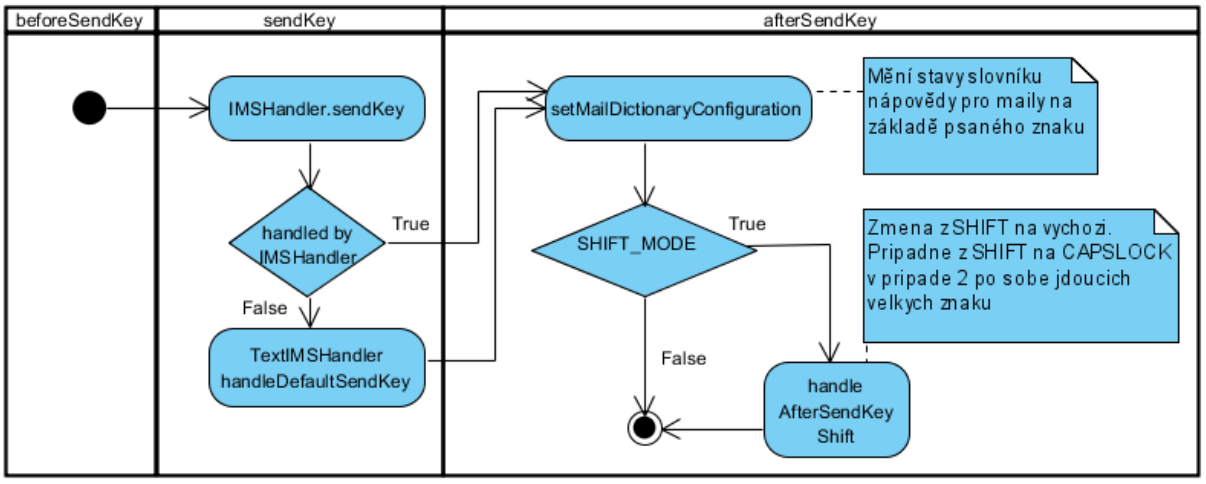
4.2.6 Klávesnice určená pro psaní e-mailových adres

Tato klávesnice vychází implementačně z obecné textové klávesnice. Vzhled je přizpůsoben pro psaní mailů. To znamená, že spodní 3 speciální klávesy jsou definovány pro znaky zavináč, pomlčka a tečka, které jsou při psaní adres často využívány.

Z hlediska chování je klávesnice zjednodušeně definována následujícími body:

- Při vstupu na toto pole se nastavuje výchozí klávesnice (tj. malá písmena).
- Neprobíhá žádná úprava předchozího textu.
- Numerické znaky, tečka, pomlčka a další, které jsou umožněny v mailové adrese, neukončují slovo nápovědy.
- Nápověda je ukončena (rozdělena) znakem zavináč.
- Znaky, které běžně ukončují větu, nepřidávají navíc mezeru, automaticky nezapínají klávesnici do stavu SHIFT.
- Automatická změna ze SHIFT stavu na výchozí, případně do CAPS-LOCK stavu při detekci 2 po sobě jdoucích velkých písmen.

Obr. 21 Zpracování události stisku klávesy pro mailovou klávesnici

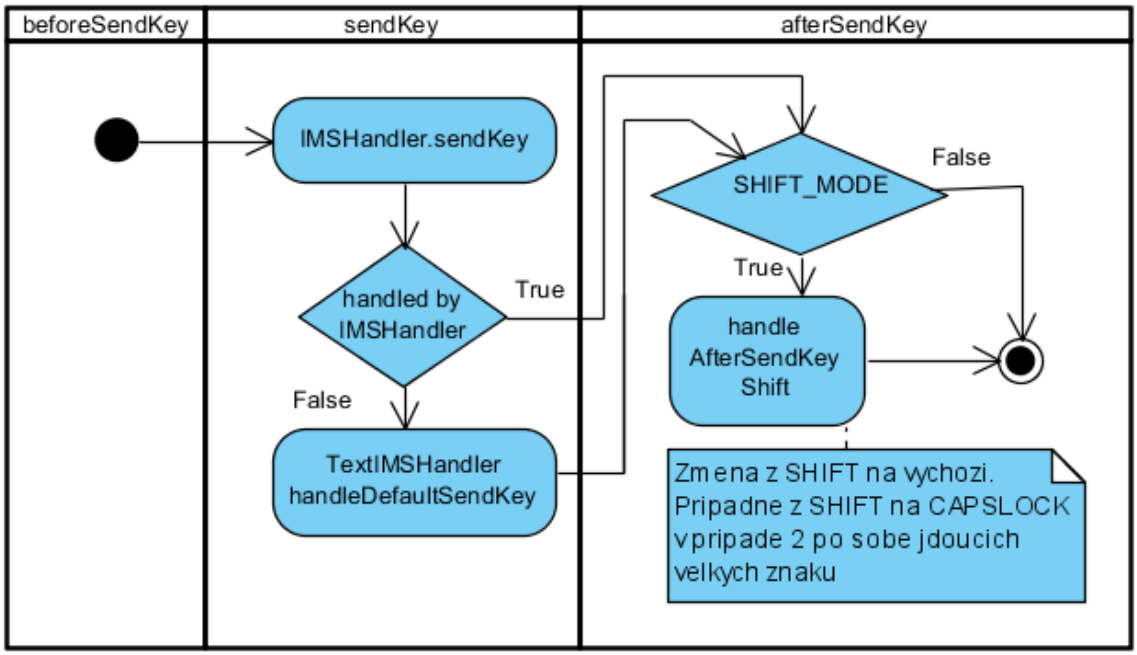


4.2.7 Klávesnice určená pro URI identifikátory

Tato varianta také vychází implementačně z obecné textové klávesnice. 3 spodní klávesy jsou uzpůsobeny psaní tak, že jsou definovány jako znaky pomlčka, tečka, lomeno. Chování je zjednodušeně definováno následujícími body:

- Při vstupu na toto pole se nastavuje výchozí klávesnice (tj. malá písmena).
- Neprobíhá žádná úprava předchozího textu.
- Nápověda je ukončena jakýmkoliv znakem, který nespadá do množiny alfanumerických znaků.
- Znaky, které běžně ukončují větu, nepřidávají navíc mezeru, automaticky nezapínají klávesnici do stavu SHIFT.
- Automatická změna ze SHIFT stavu na výchozí, případně do CAPS-LOCK stavu při detekci 2 po sobě jdoucích velkých písmen.

Obr. 22 Zpracování události stisku klávesy pro URI klávesnici



4.2.8 Klávesnice určená pro obecný text

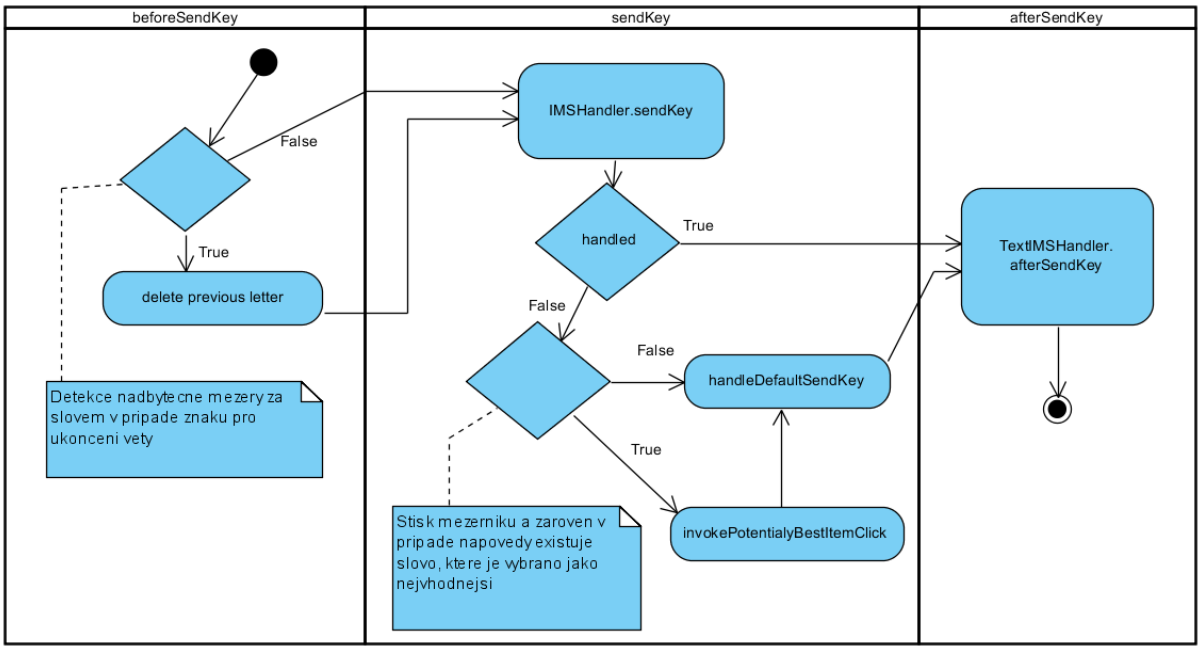
Implementace třídy, která zajišťuje obsluhu událostí při obecném psaní textu, je ze všech 3 variant nejsložitější. Využívá všech částí, tj. *beforeSendKey*, *sendKey* i *afterSendKey*.

Z hlediska chování je klávesnice zjednodušeně definována následujícími body:

- Při vstupu je nastavena výchozí textová klávesnice, případně je nastavena do stavu SHIFT, pokud je textové pole prázdné.
- Předchozí znak může být smazán, pokud se jedná o mezeru a je stisknuta klávesa ukončující větu.
- Nápověda je ukončena jakýmkoliv znakem, který nespadá do množiny znaků abecedy.
- Znaky, které běžně ukončují větu, můžou přidat mezeru, automaticky změnit klávesnici do stavu SHIFT.
- Automatická změna ze SHIFT stavu na výchozí, případně do CAPS-LOCK stavu při detekci 2 po sobě jdoucích velkých písmen.
- V případě existence vhodného slova z nápovědy, stisk klávesy mezera vyvolá událost kliknutí na toto slovo. Pro tento stav multifunkční klávesa mění chování a simuluje chování normální mezery.

Následující diagram zobrazuje aktivitu zpracování události stisku klávesy pro textovou klávesnici.

Obr. 23 Zpracování události stisku klávesy pro textovou klávesnici



4.3 Jazykový slovník

Implementace jazykového slovníku byla poměrně časově náročná záležitost. Ačkoliv s relačními databázemi mám poměrně velké zkušenosti z praxe, s vědomím, že mobilní

zařízení je výkonnostně zcela na jiné úrovni než běžné databázové servery, jsem věděl, že řešení nebude snadné. Vývojovým cyklem jsem tak prošel několikrát, protože jsem v prvních verzích narážel na problém rychlosti nápovědy. K nápovědě docházelo s výrazným zpožděním, často až po samotném napsání slova.

Při návrhu jsem zvolil jeden předpoklad, a to takový, že uživatel 1. písmeno psaného slova nevynechá a ani nedojde k překlepu. Vycházím tak z vlastní zkušenosti psaní textu a také částečně z důvodu rychlejšího zpracování dotazů. Toto omezení lze jednoduše vyřešit dodatečnými dotazy v případě překlepu. V případě úplně vynechaného znaku by se musely procházet všechny existující tabulky, což by bylo časově příliš náročné při velkém počtu slov.

Pro vyhledávání slov, která obsahují chybu ve formě překlepu, jsem zvolil předpoklad, že případná stisknutá klávesa překlepu bude v blízkém okolí správné klávesy. Proto jsem navrhl zobecnění abecedy na vybrané znaky na základě rozvržení klávesnice QWERTY. Následující tabulka zobrazuje toto zobecnění.

Tab. 4 Generalizované znaky

Generalizace	Znaky
q	q w
r	e r t
u	y u i
p	o p
s	a s d
g	f g h
k	j k l
z	z x
v	c v b
n	n m

Tuto variantu zobecnění jsem využil jak pro pojmenování tabulek, tak i pro data. Detaily jsou uvedeny v následujících odstavcích. V následujícím textu má generalizace význam právě tohoto zobecnění znaků.

4.3.1 Zdroj dat

Slovník z produktu OpenOffice jsem využil jako zdroj dat pro implementaci jazykového slovníku. Bohužel, nikde není volně dostupná databáze, případně analýza, která slova jsou často používaná a která nikoliv. Tento slovník také neobsahuje nespisovná slova, která jsou ve velké míře využívána v textové komunikaci na mobilních zařízeních.

4.3.2 Implementace

Pro slovník jsem implementoval následující třídu *LanguageDictionary*. Hledání je možné spustit asynchronně. Procedura je tedy spuštěna v novém vlákně, proto jsou všechny podpůrné komponenty, které to vyžadují, navrženy jako *ThreadSafety*, aby nedocházelo k nedeterministickému chování kódu.

Návratovou hodnotou hledání je kolekce vyhledaných dat. Rozdílem však je, zdali to jsou jednotlivá kompletní slova, nebo základy slov se svými koncovkami. V případě, že je zvolena varianta základů slov se svými koncovkami, je to výhodnější z hlediska viditelného množství slov v panelu pro nápovědu, protože jsou vidět pouze tyto základy, uživatel tak nemusí v takové míře posouvat náhled na výsledek hledání.

Důležitým prvkem pro využití slovníku je zaznamenávat často používaná slova, návrh s tímto kritériem počítá a v implementaci je každé použití slova zaznamenáno. Po delším čase používání by se tak mohla provést určitá analýza často používaných slov a mohl by se vytvořit derivát původní databáze slov, což by jistě vedlo k celkovému zmenšení velikosti slovníku.

4.3.3 Generování dat

Vzhledem k vysokému počtu slov díky možnostem českého jazyka jsem navrhl strukturu tak, aby se co největší opakující se část slova uložila pokud možno pouze jednou. Koncovky slov jsou pak vloženy do samostatné tabulky. Důsledkem zvoleného formátu uložení dat dochází k efektu komprese. Generování dat zjednodušeně popisuje následný pseudoalgoritmus.

```
function generateWords(allWords)
{
   allWords.sort();
   for-each word in allWords
   {
      save(word);
      wordsWithSuffixes = allwords.getWordsStartsWith(word);
      if(words.size <= 25)
      {
         suffixes = getSuffixes(word, wordsWithSuffixes);
         saveSuffixes(word, suffixes);
         skip-next(wordsWithSuffixes);
       }
    }
}
```

Limit 25 koncovek je zvolen čistě z praktického důvodu, aby v případě vyvolání kontextového panelu pro koncovky nebylo toto okno příliš veliké a nedošlo tak k „přetečení“ mimo viditelnou část displeje. Ačkoliv to není žádný sofistikovaný přístup pro kompresi textu, jeho efektivita je nad očekávání dobrá.

Následující tabulka udává výsledek pro generování českého slovníku, kde je vybráno 1 705 918 slov.

Tab. 5 Statistika generování slovníku

Prefix	Počet slov	Počet základů slov	Úspora
a	34377	7812	77%
á	182	50	73%
b	48919	12551	74%
c	27097	7330	73%
č	14344	3774	74%
d	110038	28109	74%
ď	250	65	74%
e	16076	3570	78%
é	77	16	79%
f	18862	4797	75%
g	10729	2560	76%
h	39179	10368	74%
i	14295	3113	78%
í	74	20	73%
j	15067	3811	75%
k	81032	21888	73%
l	31412	8449	73%
m	53794	13846	74%
n	165103	100658	39%
ň	164	50	70%
o	124291	32771	74%
ó	38	6	84%
p	271355	72682	73%
q	79	16	80%
r	80377	21139	74%
ř	4708	1357	71%
s	121013	32560	73%
š	23945	6698	72%
t	45270	11839	74%
ť	477	145	7%
u	49398	13216	73%
ú	6799	2004	71%
v	141689	37530	74%
w	745	156	79%
x	605	123	80%
y	293	65	78%
z	143500	36862	74%
ž	10265	2820	73%
	1 705 918	**504 826**	**74%**

Celková úspora je tedy cca 74%. Slabé místo algoritmu je u slov začínající písmenem N, což jsou z velké části slova začínající předponou (ne, nej…). To jsou převážně přídavná jména třetího stupně, slova začínající předponou *nej* a končící *ejší*. Obdobných hodnot nabývá úspora i pro slovenský a anglický jazyk.

4.3.4 Rychlost vyhledávání

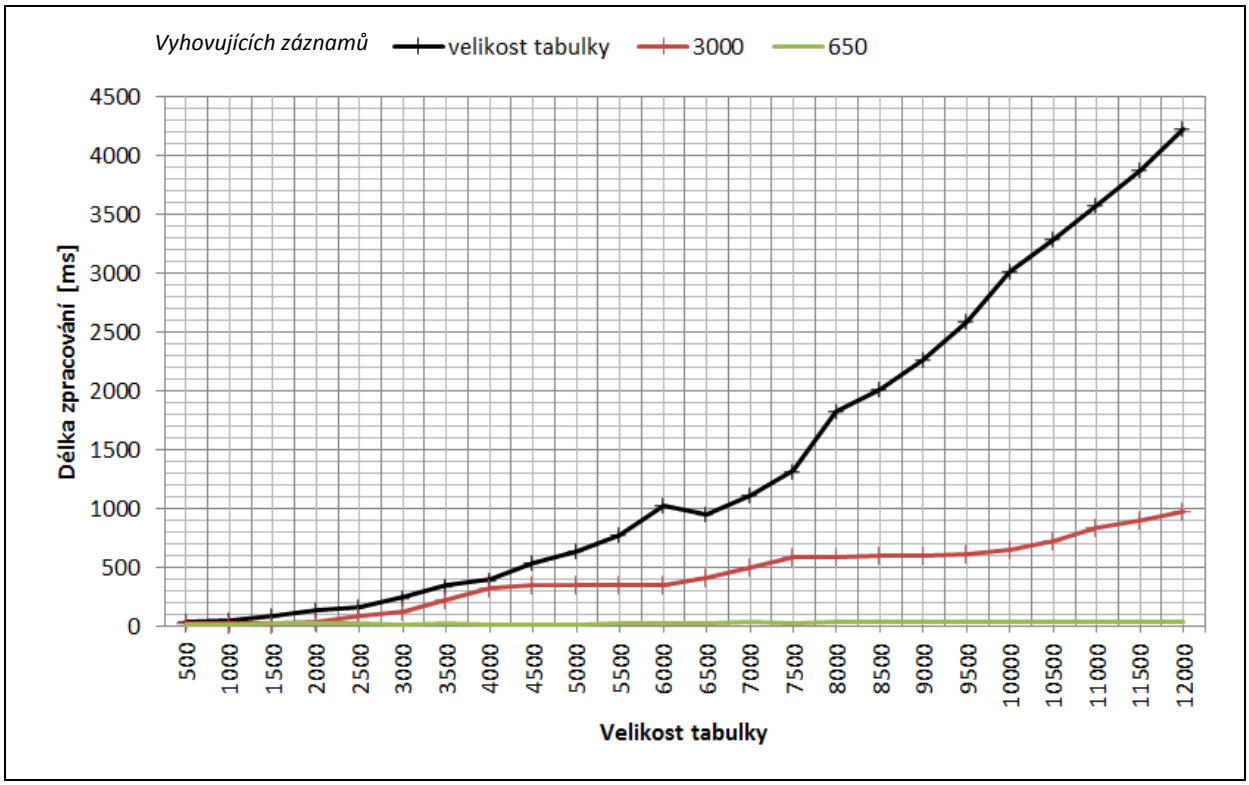
Pro udržení rychlosti vyhledávání jsem empiricky došel k závěru, že maximální rozumný počet záznamů v tabulce by neměl překročit hodnotu 5000. Tato hodnota se bude zřejmě s výkonností mobilního zařízení značně měnit. Zařízení HTC Magic je poměrně vhodné pro výchozí hodnoty, protože výkonnost tohoto zařízení je v dnešní době spíše horší. Následující tabulka zobrazuje závislosti délky zpracování dotazu vzhledem k velikosti tabulky a množiny splňující podmínku. Všechny hodnoty byly naměřeny na výše uvedeném zařízení při spuštění dotazu

```
SELECT *
FROM […]
WHERE normwordbase MATCH "na*" LIMIT 10
```

Tab. 6 Závislost časové délky dotazu na velikosti tabulky a množiny splňující podmínku

Vyhovujících záznamů	Dle velikosti tabulky	Max 3000	Max 650
Velikost tabulky	**Délka zpracování** [ms]		
500	47	32	20
1000	51	24	22
1500	89	24	31
2000	140	48	24
2500	171	92	28
3000	249	125	22
3500	349	232	27
4000	399	332	11
4500	538	352	22
5000	640	355	20
5500	781	360	32
6000	1030	352	33
6500	954	417	32
7000	1117	505	36
7500	1324	590	34
8000	1831	587	40
8500	2023	605	42
9000	2266	602	44
9500	2593	620	41
10000	3021	657	42
10500	3292	735	45
11000	3576	840	40
11500	3875	902	38
12000	4237	975	42

Obr. 24 Závislost čas. délky dotazu na velikosti tabulky a množiny splňující podmínku



Z výše uvedené tabulky, resp. grafu je vidět, že dotazy, které vracejí potenciálně velké množství záznamů, mají výraznější změnu okolo počtu 7 – 8 tisíc záznamů. Tento "skok" se více či méně projevoval u jakéhokoliv dotazu, který potenciálně vrací více než 6000 záznamů. Vždy po tomto skoku dochází k výraznějšímu nárůstu časové prodlevy.

V případě konkrétnějšího filtru, 3 znaky "`nad*`", se rozdíl zmenšuje. Červená čára grafu zobrazuje časovou náročnost pro dotazy, kterým vyhovuje cca 3000 záznamů. Pro tabulky s menším počtem než 3000 záznamů tedy vyhovují všechny. Rozdíl mezi minimem a maximem již není tak markantní, v tomto případě necelá 1 vteřina.

Pro filtr "`nast*`" o délce 4 znaků, který reprezentuje zelená čára, kterému vyhovuje 650 záznamů, je časový rozdíl pro vyhodnocení dotazu minimální. V tomto konkrétním případě 25ms. Z tohoto důvodu jsem zvolil maximální limit 5000 záznamů na tabulku, aby i obecné dotazy o velikosti 2 znaků, které mohou vracet potenciálně velké množství záznamů, byly dostatečně rychlé.

Ačkoliv je v dotazu omezení na vrácení 10 záznamů, zřejmě při spuštění nedochází k žádné výrazné optimalizaci. V porovnání se stejným dotazem, který není omezen tímto limitem, se čas neliší nijak výrazně. Následující tabulka zobrazuje měření stejných dotazů bez limitu (v dotazu není uvedena část *LIMIT 10*). Uvedeny jsou pouze časové přírůstky.

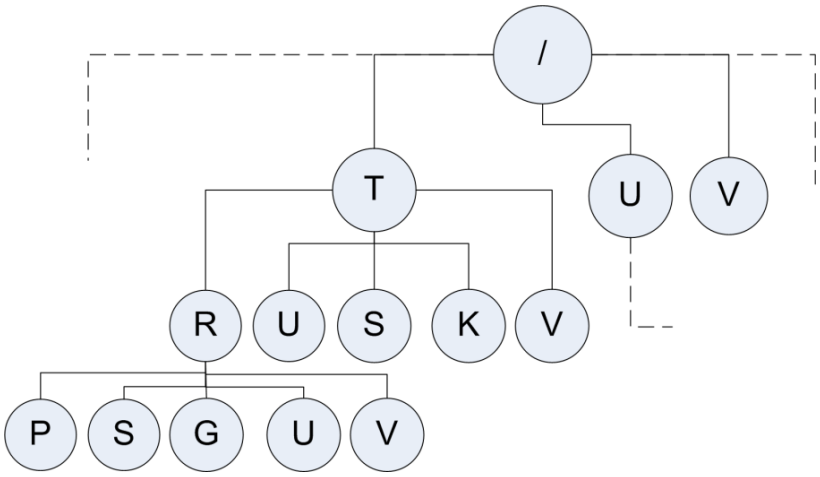
Tab. 7 Časové přírůstky dotazu bez použití limitu

Vyhovujících záznamů	Dle velikosti tabulky	Max 3000	Max 655
Velikost tabulky	**Časový nárůst zpracování** [ms]		
500	4	4	1
1000	39	8	1
1500	84	12	0
2000	119	3	7
2500	152	66	5
3000	175	65	13
3500	214	62	9
4000	319	21	26
4500	323	32	18
5000	419	62	21
5500	423	151	10
6000	373	226	10
6500	747	168	21
7000	758	98	12
7500	773	146	21
8000	706	171	15
8500	729	212	14
9000	782	163	13
9500	883	123	14
10000	755	281	19
10500	734	139	18
11000	716	138	29
11500	715	270	32
12000	772	239	34

4.3.5 Struktura uložených dat

Dalším krokem je vhodně tato data uložit. Ačkoliv jsou data uložena ve formě relační databáze, jednotlivé datové kontejnery (tabulky) jsou pojmenovány tak, že je struktura podobná stromu. Část výsledného stromu může vypadat jako např. na níže uvedeném obrázku.

Obr. 25 Příklad části stromu datových kontejnerů



První písmeno je vždy uloženo v normalizovaném (bez diakritiky) tvaru, virtuální kořen (reprezentovaný uzlem `/`) má vždy 26 uzlů. To je dáno počtem znaků abecedy.

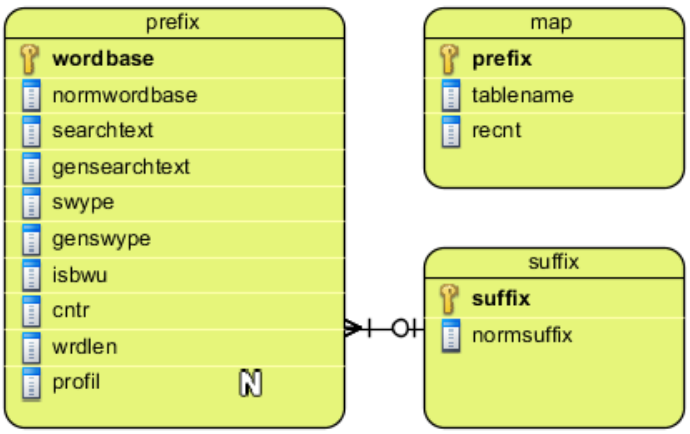
Každý potomek takového uzlu je v generalizovaném tvaru (jako je uvedeno v Tab. 4 Generalizované znaky), tzn. že má maximálně 10 potomků. Každý list má odkaz na datový kontejner, ostatní uzly mohou mít odkaz na datový kontejner.

V případě hledání např. slova *třída* je výraz převeden do výrazu `TRUSS`. První znak je normalizovaný, ostatní generalizované. A postupně se prochází strom až k cíli. V tomto konkrétním případě `/-T-R-U`. Tento list již má odkaz na tabulku. Tím je nalezen datový kontejner, resp. fyzická tabulka relační databáze.

V případě, že výraz není dostatečný pro nalezení datového kontejneru, (tudíž neexistuje slovo skládající se z dané cesty stromu a které je dlouhé jako hloubka daného uzlu), je hledáno v následujícím nejlevějším listu. To má za následek výběr slov dle abecedního řazení.

Zjednodušený model databáze je zobrazen na následujícím obrázku. *Prefix* reprezentuje entitu s jediným prefixem (ve stromě se jedná o jediný uzel, který má odkaz na tabulku). Výsledný počet bude vždy závislý na celkovém počtu slov slovníku.

Obr. 26 Zjednodušený datový model



Suffix reprezentuje koncovky pro daná slova s vazbou na všechny datové tabulky. *Map* je už pouze uložená struktura navigačního stromu.

Pro mnou vygenerovaný slovník obsahující 1 705 918 slov je vygenerován „strom“ odkazující na 457 datových úložišť, resp. tabulek. Počet záznamů je pro většinu tabulek dostatečně vzdálen limitu 5000 záznamů, k němu se blíží pouze 7 tabulek.

Takto zvolená struktura uložení je vhodná pro hledání překlepů. Díky generalizování znaků jsou uložená slova ve shodném kontejneru. Problém nastává, pokud dojde k záměně písmen, která jsou na hranici tohoto rozdělení a také v případě chybějícího písmene v části slova, které definuje cestu k datovému kontejneru. Toto omezení lze vyřešit dodatečnými dotazy, bohužel je to náročné z časového hlediska, proto není tato funkcionalita v jazykovém slovníku podporována.

4.3.6 Detailní popis struktury tabulky

Následující tabulka detailně popisuje jednotlivá pole tabulek s jednoduchým příkladem pro uložení základu slova *vhodně* s možnými koncovkami. Normalizovaný výraz je zde ve významu, že je výraz bez diakritiky, generalizovaný ve významu jako je uveden v Tab. 4 Generalizované znaky.

Pole	Popis	Příklad
Tabulka jako datový kontejner (prefix)		
wordbase	Základ slova s diakritikou	vhodně
normwordbase	Normalizovaný výraz wordbase	vhodne
searchtext	Výraz k vyhledání skládající se ze základu slova a všech znaků koncovek.	v h o d n e j i s h o c m a u
gensearchtext	Generalizovaný výraz searchtext	c g p s n e k u s g p c n s u
swype	Předgenerované výrazy pro speciální metodu psaní Swype.	vodne
genswype	Generalizovaný výraz swype	cpsne
isbwu	Identifikace, zdali je základ slova samostatně použitelný. Nabývá pouze hodnot 1 a 0.	1
cntr	Čítač použití, používaný pouze v případě uživatelského slovníku	0
wrdlen	Délka základu slova	6
profil	V tuto chvíli nepoužívané. Určeno pro budoucí použití.	
Tabulka suffix		
suffix	Jednotlivé koncovky k základu slova	ji jší jšího jších…
normsuffix	Normalizovaný výraz suffix	ji jsi jsiho jsich…
Tabulka map		
prefix	Cesta ve stromu k datovému kontejneru	Vg
table	Název datového kontejneru (tabulky)	Vg
recnt	Počet záznamů v tabulce. Pouze statistický význam.	…

4.3.7 Optimalizace struktury uložených dat

Při generování dat jsem kvůli rychlostní optimalizaci způsobil částečnou redundanci dat. Protože je dotazování do tabulek poměrně časově drahá záležitost, některá slova jsou uložena navíc i v kontejnerech, které jim přímo nenáleží. Což je částečně v rozporu se strukturou, kterou jsem popisoval v předchozím odstavci.

Optimalizace probíhá tak, že jsou na konci procesu generování dat analyzovány jednotlivé tabulky a pokud je základ slova stejně dlouhý jako cesta ve stromě k tomuto uzlu, je takovéto slovo zkopírováno i do rodičovského uzlu (pokud neexistuje, tak se vytvoří).

Uvedená optimalizace je např. vhodná pro slovo *něco*, v datovém uložišti je uloženo v kontejneru `NECU`, při této optimalizaci je tak zkopírováno i do kontejneru `NEC` tak, aby při vyhledávání výrazu "nec" bylo nabídnuto i slovo *něco*. To se významně podílí na rychlosti a uživatel nemusí napsat celé slovo, aby mu bylo nabídnuto.

Tato optimalizace způsobuje redundanci řádově stovek slovních základů, což má nepatrný význam na celkovou velikost souboru, ale markantní význam při rychlosti vyhledávání a úspěšnosti nápovědy.

4.3.8 Ukládání uživatelských slov a četnosti použití

Datové uložiště slovníku je navrženo tak, aby se výchozí uložiště neměnilo. Veškerá dodatečná data, jako uživatelem definovaná slova, jsou uložena do externího uložiště. To může být výhodné např. pro synchronizaci tohoto uložiště mezi více místy, protože bude mnohonásobně menší než samotný datový soubor slovníku.

Struktura uživatelského datového uložiště je v podstatě shodná jako u výchozího slovníku. Rozdíl je pouze ve stromové struktuře, protože předpokládám, že uživatel bude definovat vlastní slova maximálně v řádu stovek. Proto má stromová struktura uložení dat hloubku 1.

Ukládání četnosti použití je vyřešeno tak, že v případě prvního využití dojde k uložení daného slova do uživatelského slovníku. Pokud se jedná o uživatelské slovo, dochází k inkrementaci hodnoty čítače, která je uložena společně s daty v příslušném datovém kontejneru. Tato varianta není příliš vhodná z hlediska objemu dat, protože se ukládá kompletně celá kopie záznamu. Významnou výhodou je však to, že slovo je nalezeno s mnohem menším prefixovým výrazem, protože je uloženo v uživatelské databázi a může být uživateli nabídnuto mnohem dříve, než kdyby bylo nalezeno ve výchozím slovníku.

Možnost ukládání slov je v mé implementaci poněkud problematičtější. Uživatel samozřejmě může uložit jednoduše pomocí dlouhého kliku na nově zvolené slovo v panelu nápovědy, tímto způsobem se však ochuzuje o výhody kontextového uložení slova. Pro ukládání slov v kontextovém režimu je implementován vlastní formulář, kde může uživatel jakkoliv upravovat slovník. Pro případné vygenerování vlastního slovníku bude sloužit speciální program.

4.3.9 Ukládání výsledků do dočasné paměti

Důležitým prvkem, který umožňuje urychlit hledání je tzv. cache, neboli dočasná paměť. Každý výsledek je uložen a při každém dotazu se ověřuje, zdali již výsledek není v této

paměti uložen a až poté se iniciuje proces vyhledávání. Toto je zvláště výhodné při upravování již napsaného slova, kdy postupným umazáváním znaků od konce slova dochází k opětovnému dotazování slovníku pro možnosti nápovědy. Po dosažení určitého limitu se nejméně používaná data mažou.

Díky způsobu uložení dat se pro základy slov musí také dohledávat samotné koncovky, proto jsou i tyto výsledky ukládány do své vlastní dočasné paměti. Protože je až cca 75% slov uloženo ve stavu, že je rozděleno na základ a koncovky, zvolil jsem velikost této paměti 3násobně vyšší než velikost dočasné paměti pro výsledky hledání.

Pro zjištění efektivity dočasné paměti jsem provedl následující jednoduchý test. Z internetového portálu www.novinky.cz jsem si vypůjčil část textu, který jsem se pokusil napsat pokud možno bez chyb a měřil čas potřebný k nalezení výsledků. Pro test jsem zkusil varianty bez použití dočasné paměti a velikosti pro 50, 100, 150 položek.

Text pro testování byl následující:

> Z měsíce na měsíc zaznamenala důvěra ve vládu v očích veřejnosti silný propad, a po krizi je na tom nejhůře za poslední rok.

Minimální délkou pro vyhledávání jsou 2 znaky, proto je maximální ideální počet vyhledávání roven počtu 77.

Tab. 8 Měření použití dočasné paměti

Velikost cache	**Celkový čas pro vyhledávání** [s]	**Počet uskutečněných hledání.**	**Časová úspora oproti nepoužití dočasné paměti**
0	37,127	80	0%
50	25,109	77	33%
100	21,198	84	43%
150	22,601	82	40%

Počet uskutečněných hledání je vyšší díky způsobeným překlepům a opravám nutných pro napsání textu. Z tabulky každopádně vyplývá, že využití dočasné paměti se vyplatí a úspora je, v tomto jednoduchém měření, okolo 40%. Na základě tohoto měření jsem zvolil hodnotu velikosti dočasné paměti na 100 položek, protože tato hodnota není příliš vysoká a zároveň poskytla nejlepší výsledek z měření.

4.3.10 Vyhledávání slov

Vyhledávání a porovnávání probíhá vždy bez ohledu na diakritiku, jak v uživatelské databázi, tak i ve výchozí databázi slovníku, několikastupňově, a to vždy na základě počtu nalezených výsledků. Minimální délka výrazu pro vyhledávání jsou 2 znaky. Tuto hodnotu jsem zvolil záměrně, protože 1 znak je až příliš obecný a zároveň nápovědu pro slova o délce

1 znaku, jako může být spojka nebo předložka, nepovažuji příliš za smysluplnou. Vyhledávání je rozděleno na 2 základní typy:

- Hledání s předpokladem správně zapsaného slova uživatelem.
- Hledání s předpokladem nějaké chyby (překlep, chybějící znak).

Hledání správně zapsaného slova je ještě rozděleno na 2 části. Základní a rozšířená, vyhledávající slova rozdělená výše uvedeným způsobem na základ a koncovku.

Pro lepší znázornění uvedu následující příklad pro hledání slova *zhasnutými*. Slovo je uloženo ve tvaru *zhasnut* a k němu jsou v relaci následující koncovky "*ě i í ích ím ími o u y*". V tomto případě dochází k hledání podle dat uložených v poli *searchtext*, kde se nachází výraz "*z h a s n u t e i c h m o u y*". Výraz je kombinací základu slova a kombinací znaků koncovek, které jsou odděleny mezerou tak, aby mohly být nalezeny pomocí *FullTextu*. Jakmile uživatel překročí délku základního slova, tj. hledaný výraz nebyl nalezen při hledání v 1. stupni, vybírá se následující šablona:

- Slovo začíná na výraz „zhas“ (jednoznačně určené uloženými daty, 4 znaky jsou minimum pro rozdělená slova).
- Pole searchtext obsahuje znaky, resp. slova „z h a s n u t y m i“.
- Vzhledem k nenalezení v předchozím případě, musí být délka základu slova menší než hledaný výraz.

Na základě těchto 3 filtračních parametrů je vyhledán, v tomto konkrétním případě, právě 1 základ, a to právě *zhasnut*. V případě nalezení více záznamů probíhá další krok porovnání shodně jako pro 1 záznam. A to provedením iterace přes nalezená slova a koncovky a porovnání normalizovaných (bez diakritiky) hodnot.

Další způsoby vyhledávání probíhají obdobným způsobem jako v předchozím příkladu. Zásadním rozdílem je však porovnání na základě generalizovaných znaků obdobně jako v případě definice názvů tabulek, a to vždy tak, že následující způsob vždy určitým způsobem zobecňuje hledání tak, aby byla zahrnuta do porovnání větší skupina slov. Pokud ani v jednom případě není hledání dostatečné, dojde k poslední variantě. Ta vyhledá slova, která se skládají z posloupnosti znaků hledaného výrazu. Znaky mohou být i prohozené mezi sebou. Tato varianta je však uplatňována pouze pro výchozí slovník, protože při malém počtu slov uživatelského slovníku mohou být ve výsledku slova velmi „vzdálená“ od hledaného.

Každé další prohledávání probíhá pouze tehdy, pokud právě aktuální nevrátilo dostatečný počet vhodných slov. Z toho vyplývá, že pokud uživatel píše přesně, vyhledávání je rychlejší, protože nedochází k dalším stupňům prohledávání.

Za velmi výhodné považuji zobecnění znaků Y a I, protože budou tato slova ve shodném datovém kontejneru. V případě napsaní špatného Y, resp. I jim slovník nabídne správnou variantu, a to v případě že „špatná“ varianta neexistuje. Nevýhodné je to bohužel v případě použití rozložení kláves QWERTZ, kde může dojít k překlepu znaků X a Y. V takovém případě bohužel nedojde k správnému nalezení hledaného výrazu.

Detailní diagram aktivity vyhledávání výrazu je v příloze Obr. 31 Diagram aktivity vyhledání v jazykovém slovníku.

4.3.11 Optimalizace vyhledávání

Částečná optimalizace pro zúžení množiny potenciálních výsledků je pomocí definice maximální délky základu slova. Maximální délka je definována na základě délky vyhledávaného prefixu slova:

- Prefixy kratší než 3 znaky včetně, je limit délka prefixu + 2 znaky.
- Delší slova jsou bez limitu.

Tato optimalizace se poměrně významně podílí na rychlosti vyhledávání při zadání krátkého prefixu, protože podmínku krátkého prefixu obecně splňuje velká množina slov.

Nevýhodou však může být existence "hluchých míst" pro určitý prefix, kterým vyhovují převážně dlouhá slova, resp. základy slov (základ slova je delší než 6 znaků včetně) a minimum příp. žádné z krátkých slov, resp. základů slov (základ je kratší než 5 znaků včetně), tím by došlo k zobrazení až po zadání minimálně 4 znaků.

4.3.12 Předchozí vývojový cyklus

Jak jsem zmínil na začátku celé kapitoly, implementace slovníku nebyla jednoduchou záležitostí, protože jsem stále narážel na určité problémy. Po úspěšném nalezení řešení jak a v jakém množství ukládat vhodně data, jsem narážel na problém, jak správně vyhledávat slova, která jsou rozdělena na základ a koncovky a vyhledávaný výraz je již delší než onen slovní základ. Což jsem řešil pomocí dočasné paměti.

Předpokládal jsem, že pokud uživatel píše slovo a dojde k hranici, že psaný výraz je delší než slovní základ, je takové slovo hledáno v dočasné paměti, protože již toto slovo muselo být v jednom z předchozích hledání. Toto řešení se rychlostí nelišilo příliš od současného řešení. Problém však nastal, pokud uživatel začal např. opravovat text, který se z nějakého důvodu nevyskytoval v dočasné paměti (uživatel ho např. nenapsal, ale opravuje uložený text apod.).

Pokud uživatel slovo mazal od konce, nedocházelo k nalezení vhodných variant, které jsou rozděleny na základ a koncovky, dokud slovo nesmazal až na onu hranici rozdělení základu a koncovek. Tento problém, ačkoliv se nevyskytoval příliš často, byl poněkud nepříjemný pro použití a matoucí, protože docházelo k nápovědě pouze "někdy". Proto jsem toto řešení zavrhl a implementoval současné, které tímto problémem netrpí.

4.4 Slovník telefonních kontaktů

Další varianta implementace slovníku je pro telefonní seznam. Tento slovník je specializací abstraktní třídy *Dictionary*. Hledání je rozděleno na 2 kritéria:

- Hledání podle telefonu
- Hledání podle jména

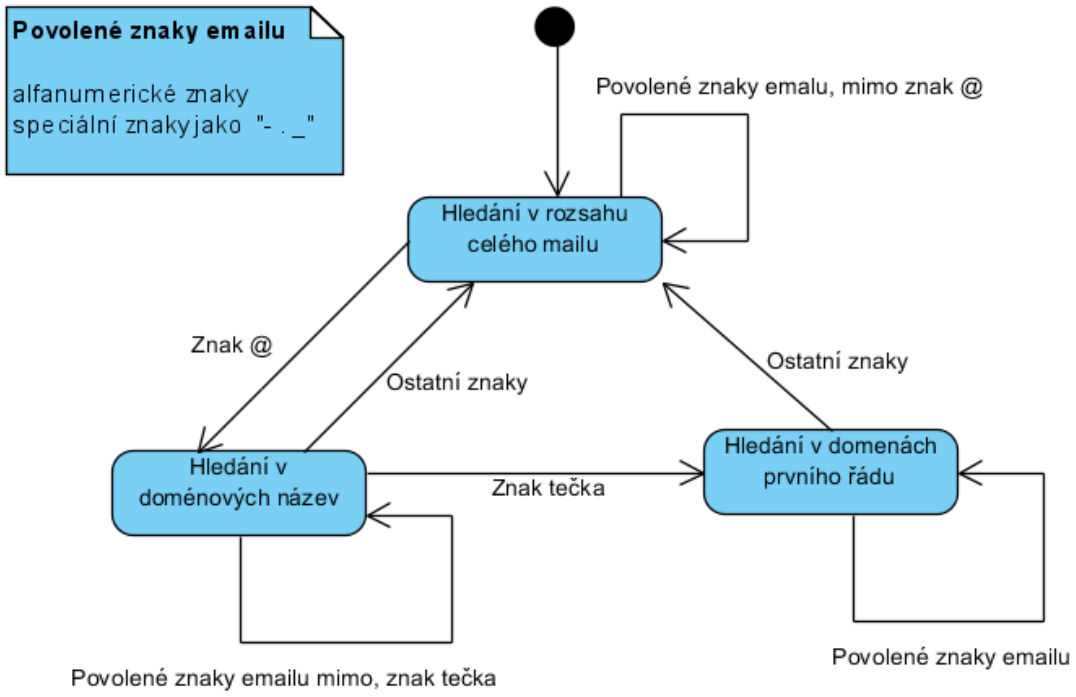
Rozdělení probíhá automaticky na základě hledaného výrazu. Je-li první znak hledaného výrazu + nebo numerická hodnota, vybírá se varianta hledání podle telefonu, jinak podle jména. Pro vyhledávání se porovnává, zdali jméno, resp. telefonní číslo obsahuje hledaný výraz, nikoliv jestli jím pouze začíná. Výsledkem jsou vždy nalezená jména a jejími kontextovými hodnotami jsou telefonní čísla.

4.5 Slovník mailových kontaktů

Slovník obsahující mailové kontakty je další implementací z kategorie slovníků. Jako v předchozím případě k úspěšnému vybrání hodnoty do výsledku se dojde pomocí porovnání obsahu, nikoli jen začátku.

Mailový slovník vyhledává na základě definovaných stavů podle psané části emailové adresy. Mailová adresa se skládá ze 2 částí. Tzv. *local-part* část, tedy ta před znakem zavináč a na *domain-part*, což je část za znakem zavináč. Doménovou část lze ještě rozdělit na třídy domén. Ve své implementaci jsem je rozdělil na 2 typy. První jsou domény 1. řádu (cz, sk, com…) a ostatní. Podle daného stavu jsou tak vyhledávány vždy určité množiny výrazů. Následující diagram zobrazuje jednoduchý stavový automat, dle kterého se rozhoduje, jaká část mailu bude prohledávána.

Obr. 27 Diagram stavů - MailDictionary



4.6 Slovník pro URI identifikátory

Tento slovník má oproti ostatním slovníkům odlišné chování. Je to dáno z důvodu nápovědy pro tzv. URI klíčová slova. Objem tohoto slovníku je uživatelem definovaný a předpokládá se, že bude velmi malý. Řádově do 10 výrazů, které budou používány pro zadávání např. webové adresy. Slovník nemá nahradit např. uložiště pro záložky apod., ale sloužit pouze jako kontejner pro často používané výrazy jako *www*, *http://*, případně definovat nejčastěji používané domény první třídy. Slovník proto nepodporuje možnost hledání a je naimplementován tak, že vrací vždy celý svůj obsah slov. Uživatel má možnost určit si nejenom tato klíčová slova, ale i jejich pořadí, které se nemění na základě počtu využití.

4.7 Speciální metody zadávání textu

Implementovaná speciální metoda zadávání textu je tahem prstu.

4.7.1 Předpoklady

V mé implementaci jsem zvolil následující 2 předpoklady:

- Uživatel začíná na správné klávese.
- Uživatel končí na téměř správné klávese.

Téměř správnou klávesou myslím takovou, která spadá do skupiny generalizovaných znaků, jako v případě jazykového slovníku. Toto jsou 2 základní pravidla, která musí být splněna pro nalezení vhodných kandidátů do nápovědy.

4.7.2 Implementace

Pro daný problém jsem implementoval třídu, která řeší následující problémy:

- Detekuje start procesu tohoto způsobu psaní.
- Sbírá všechny body tahu.
- Vykresluje opisovanou křivku prstem.
- Vyhodnocuje křivku, kterou převádí do textové podoby.

Výsledkem vyhodnocení křivky jsou vždy 2 slova:

- SwypeSimpleText
- SwypeComplexText

SwypeSimpleText je posloupnost znaků, na kterých byla tahem prstu provedena výrazná změna tahu. Tato změna je detekována pomocí rozdílů úhlů pomyslných čar mezi body nad jednou klávesou. Tyto znaky jsou brány jako klíčové a předpokládám, že budou obsaženy v uživateli psaném slově.

SwypeComplexText je posloupnost všech znaků kláves, které byly součástí tahu prstu.

4.7.3 Hledání extrémů

Hledání extrémů jsem implementoval následujícím způsobem, který popisuje následující zjednodušený pseudoalgoritmus.

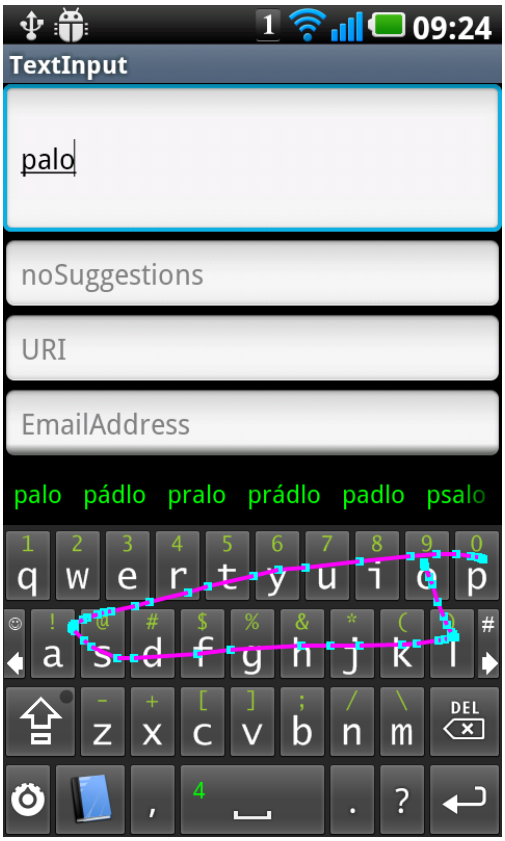
```
function generateSwypeSimpleText(allPoints)
{
	limit = 110;
	key = getKey(allPoints[0]);
	swypeSimpleText = key.char;
	angle = 0;

	for i = 1 to (allPoints.size - 1)
	{
		if(getKey(allPoints[i])) == getKey(allPoints[0]))
			continue;
		angle += getAngle(allPoints[i-1],allPoints[i]);
		if(key != getKey(allPoints[i])){
		{
			if(angle > limit)
				swypeSimpleText += key.char;
			angle = getAngle(allPoints[i-1],allPoints[i]);
			key = getKey(allPoints[i]);
		}
	}
	return swypeSimpleText;
}
```

Limit úhlu 110° je zvolen empiricky a také s ohledem k předgenerovaným *SwypeSimpleText* výrazům ve slovníku. Vzhledem k předpokladu, že znaky *SwypeSimpleTextu* mají být obsaženy i ve výsledném slově, je z tohoto pohledu lepší znaky vybírat spíše v menším měřítku.

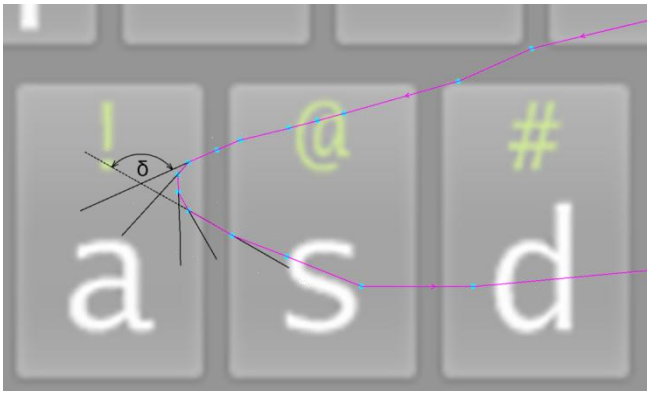
Pro každou dvojici je dle výše uvedeného algoritmu spočítán úhel. Pro každou klávesu jsou tak sečteny rozdíly těchto úhlů, kde na základě limitní hodnoty mohou být vyhodnoceny jako dostatečné a znak klávesy je přidán do výsledku *SwypeSimpleTextu* a nebo je ignorován. Pro generování *SwypeComplexTextu* se jedná pouze o iteraci přes všechny body a vygenerování textu tak, aby neobsahoval vícenásobně znak klávesy v dané části křivky.

Příklad na zapsání slova *prádlo*. Následující obrázek znázorňuje klávesnici po skončení tahu prstem. Azurovou barvou jsou znázorněny zachycené body tahu křivky tak, jak docházelo k volání systémových událostí. Purpurová barva je křivka spojující jednotlivé body.

Obr. 28 Příklad psaní tahem prstu



Obr. 29 Hledání úhlového extrému nad klávesou A



Šipky znázorňují směr tahu prstu. V tomto konkrétním případě je výsledek následující. (První a poslední znaky P a O jsou na základě předpokladů přidány vždy, proto nejsou v tabulce uvedeny).

Tab. 9 Výsledek analýzy křivky

Znak	Součet diferencí úhlů
o	9
i	9
u	5
y	13
t	5
r	2
e	4
d	2
s	17
a	140
s	14
d	2
f	5
g	7
h	1
j	9
k	9
l	130

Z tabulky lze vidět, že pouze znaky A a L přesahují limit, proto jsou zařazeny do *SwypeSimpleTextu*. Výsledkem celého procesu analýzy křivky je:

Tab. 10 Textový výsledek analýzy křivky

SwypeSimpleText	palo
SwypeComplexText	**P**oiuyt**R**eds**A**s**D**fghjk**LO**

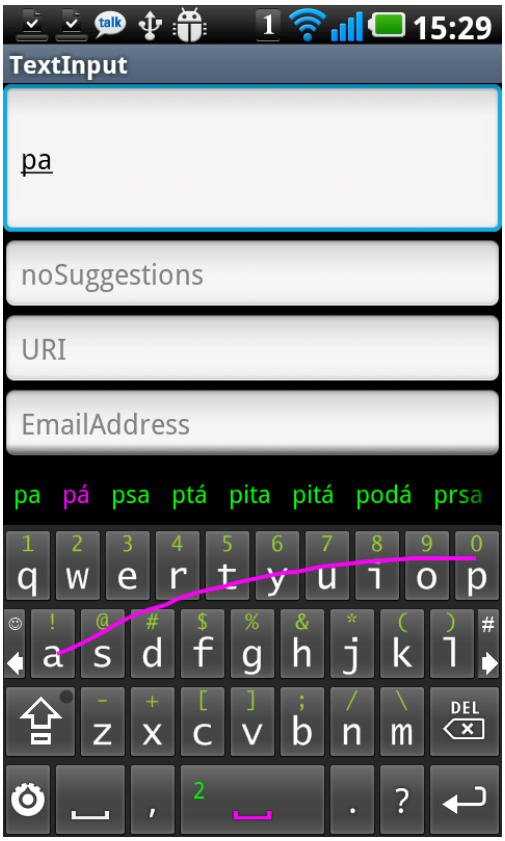
Tento výsledek je následně předán slovníku, který na základě těchto 2 výrazů hledá vhodné kandidáty.

4.7.4 Problematika rychlosti a velikosti slovníku

Vyhledávání slov je v tomto případě výrazně náročnější oproti jazykovému slovníku. Zde již musí probíhat prohledávání všech datových kontejnerů příslušného podstromu, protože nelze jednoznačně určit přesnou lokalitu datového kontejneru ve stromové struktuře. Tento problém z části naráží na výkonnost mobilních zařízení, protože takové vyhledávání není levná záležitost. Proto je pro rychlost, při tomto způsobu zadávání dat, poměrně zásadní míra „kvality“ nakreslené křivky.

Velikost slovníku se zde podepisuje ještě na jedné věci, a to je počet nabízených variant. Z důvodu toho, že *SwypeSimpleText* se může výrazně lišit od uživatelem myšleného slova, bude takovéto slovo v případě nepřesné nápovědy problematické opravovat a bude jej lepší napsat normálním způsobem. Při pokusu napsat například slovo *pouta* znázorňující následující obrázek.

Obr. 30 Příklad psaní tahem slova pouta



Relativně rovná křivka je analyzována do podoby, že *SwypeSimpleText* jsou znaky PA a *SwypeComplexText* jsou znaky POIUYTRDSA. Tato kombinace vrací poměrně velký počet slov, která jsou následující

Tab. 11 Nalezené kombinace při hledání slova "pouta"

pa	potra	pta
pá	pouta	ptá
pita	poutá	půda
pitá	prsa	pusa
podá	psa	

Kritéria pro porovnání jsou v tomto případě pouze ta s předpokladem ideálního vstupního výrazu, v případě hledání s předpokládanou nepřesností se výsledek rozroste na téměř 3násobek. Z tohoto úhlu pohledu je vyšší počet obsažených slov ve slovníku spíše jeho nevýhodou.

4.7.5 Hledání vhodných kandidátů

Pro vyhledání vhodných kandidátů jsem implementoval další variantu třídy slovník. Vychází z jazykového slovníku a vstupem pro vyhledávání je využívána dvojice výrazů *SwypeSimpleText* a *SwypeComplexText*.

Vzhledem k vysokému počtu slov ve slovníku a poměrně časové náročnosti porovnávat textové řetězce jsem zvolil variantu takovou, že ve slovníku jsou vygenerované možné varianty SwypeSimpleTextu pro daná slova. Díky rozložení kláves, je velká část těchto výrazů shodných, takže mohou být uloženy pouze jednou. Jsou to 3 varianty s limitním úhlem 55°, 90° a 120°. To má výhodu z hlediska rychlosti vyhledávání daného slova, pokud uživatel provádí tah prstu dostatečně přesně. Nevýhodou je větší velikost slovníku a také nepřímá závislost na vzhledu klávesnice. Slova jsou v tuto chvíli vygenerovány pro vzhled QWERTY a QWERTZ. V případě implementace dalšího vzhledu by bylo nutné slovník vygenerovat znova, aby byly varianty *SwypeSimpleTextu* zohledněny i pro ostatní rozložení kláves.

Prohledávání probíhá, stejně jako u jazykového slovníku, a to několikastupňově na základě předchozího výsledku s tím rozdílem, že slovo splňující kritéria nalezení musí splňovat navíc podmínku, že ho lze složit z výrazu *SwypeComplexText* postupným odebíráním skupiny znaků. Tak jak je zobrazeno v Tab. 10 Textový výsledek analýzy křivky. Tato kontrola je jedním z limitů určující přesnost nalezeného výsledku, bohužel nesmí být absolutním rozhodovacím kritériem.

Vzhledem k příkladu z předchozího odstavce se slovem pouta, v případě, že by uživatel zamýšlel napsat slovo *pouhá*, výsledek může být obdobný při nedbalém tahu prstu. A takové slovo by nesplňovalo výše uvedené kritérium.

5 TESTOVÁNÍ

Jedním z úkolů této diplomové práce je nechat otestovat implementaci uživateli. Testování bylo zprostředkováno vedoucím práce. Bohužel se z časových důvodů nepodařilo implementaci otestovat v takové míře, jak bylo původně plánováno. V následujících odstavcích jsou uvedeny uživatelské připomínky ke klávesnici s tím, jak byly mnou implementovány a jaké byly uživatelské připomínky a následně také implementovány.

5.1 Funkcionalita

5.1.1 Automatické přepínání módu SHIFT, CAPS-LOCK

Navrženo	Shift se automaticky zapíná po znacích ukončující větu (. ? ! …).
Změna	2 po sobě jdoucí velké znaky zapínají CAPS-LOCK režim
Změna	Po ukončení slova dochází k přepnutí do výchozího módu, resp. do SHIFT módu v případě, že za slovem následuje znak ukončující větu.
Změna	Po mazání dochází k detekci, zdali je kurzor před znakem ukončující větu a v takovém případě přepnout do SHIFT modu

5.1.2 Mazání slov pomocí tahu prstu

Navrženo	Nenavrženo
Požadavek	Mazání slov pomocí tahu prstu na spodním řádku směrem zprava doleva.
Změna	Umožnit definovat stav, zdali mazání maže poslední znak nebo celé slovo.

5.2 Vzhled

5.2.1 Multifunkční klávesa

Navrženo	Klávesa zobrazuje následující stav.
Změna	Klávesa zobrazuje aktuální stav.

5.2.2 Rozvržení kláves

- **Speciální klávesy**
  Jedná se o 3 klávesy, které se mění vzhledem k aktuálnímu kontextu. Jsou umístěny na spodním řádku klávesnice. Ve výchozím stavu obsahují znaky tečka, čárka, otazník.

Navrženo	Bez speciálního rozvržení.
Změna	Rozložení kláves tak, aby jich co nejvíce neměnilo svojí pozici.

- **Kontextové klávesy**
  Klávesy, které jsou vyvolány událostí *LongClick*. Obsahují speciální znaky a znaky s diakritikou.

Navrženo	Speciální výchozí kontextový znak je umístěn vlevo. Diakritických znaků je co nejvíce vůči znaku aktivační klávesy.
Změna	Speciální výchozí kontextový znak je umístěn pokud možno ve stejné horizontální pozici jako aktivační klávesa.
Změna	Kontextové diakritické znaky omezit na znaky používané v daném jazyce.

- **Klávesy pro emotikony**

Navrženo	Klávesy jsou reprezentovány ikonami.
Změna	Konfigurovatelné emotikony tak, aby byly zobrazeny i ve své textové podobě.
Změna	Konfigurovatelná položka, pro definici nosu emotikony. (:) :–) :o))

- **Klávesy měnící vzhled klávesnice na speciální znaky a emotikony**

Navrženo	Klávesy jsou úzké, piktogram je ve formě šipky. Logika ovládání umožňuje „rotaci“ mezi stavy.
Změna	Definovat piktogram tak, aby ukazoval, jaký typ klávesnice se zobrazí po stisku.

- **Numerická klávesnice**

Navrženo	Navržena tak, aby se spodní řádek klávesnice neměnil. Na něm je např. multifunkční tlačítko, enter, mezera. Klávesnice byla tak o 1 řádek vyšší než ostatní textové.
Změna	Změnit rozvržení tak, aby numerická klávesnice byla stejně vysoká jako ostatní.

- **Klávesa pro schování celé klávesnice**

Navrženo	Byla umístěna na spodním řádku klávesnice mezi dalšími speciálními klávesami.
Změna	Klávesa zcela odstraněna, protože existují další způsoby, jak provést schování klávesnice.

5.3 Slovník

- **Datové uložiště**
  Fyzický soubor s daty

Navrženo	Vše je ukládáno v tomto souboru, tj. i změny a čítače použití.
Změna	Výchozí databázi neměnit a veškerá uživatelsky definovaná data ukládat do dalšího souboru, který může být synchronizován s dalšími službami.

- **Kontextový výběr**
  Kontextové menu slouží k výběru koncovek slova

Navrženo	V případě výběru potvrzení kontextového slova se potvrdí vždy i v textovém poli.
Změna	Potvrzení proběhne pouze pro slova, která spadají do množiny existujících český slov, v opačném případě se zobrazí kontextové menu.

- **Výběr vhodného slova z nápovědy**
  Při použití nápovědy dochází k zobrazení napovídaných slov v panelu nápovědy. Pokud je nastavena nějaká položka jako vhodný kandidát, tato položka změní barvu. Chování klávesy mezera se změní, takže v případě stisku vyvolá událost stisku onoho vhodného kandidáta z panelu nápovědy. Stejně tak se změní chování multifunkční klávesy, která změní své chování na standardní chování klávesy mezera.

Navrženo	Vždy je vybráno nějaké slovo tak, že se neshoduje se psaným. Tj. například slovo psané bez diakritiky vybírá slovo s diakritikou. Pokud je toto slovo nežádoucí, provede se mezera pomocí MF klávesy, která je příslušně označena.
Změna	Vybírat kandidáty pouze tehdy, pokud je slovo shodné na úrovni znaků bez diakritiky a zároveň slovo bez diakritiky není v množině vhodných kandidátů. Tj. že psané slovo neexistuje ve slovníku.

- **Vyhledávání ve slovníku pro mailové kontakty**

Navrženo	Vyhledávání probíhá vždy porovnáním s celou adresou, nikoliv pouze podle počátku.
Změna	Vyhledávání je rozděleno na 3 stavy. Hledání podle mailu, hledání podle doménového názvu a hledání podle domény prvního řádu.

6 ZÁVĚR

Věřím, že jsem úspěšně splnil stanovené cíle, vytvořil použitelnou virtuální klávesnici a založil slušný základ pro případné další pokračovatele této práce.

Díky této práci jsem se naučil API platformy Android. Zjistil jsem, že při programování aplikací pro platformu Android je potřeba přizpůsobit myšlení, protože je třeba brát ohledy na mnoho věcí, která při vývoji pro standardní PC nejsou tolik potřeba. Mnohem více je zde potřeba asynchronního volání, protože hrubá výkonnost je oproti běžnému PC zcela na jiné úrovni. Časově náročné operace by měly být spouštěny v jiném vlákně, aby aplikace stále reagovala na události od uživatele a nedocházelo k nežádoucímu zasekávání. Nejrychlejší kód velmi často znamená problematický, protože nedochází ke správnému uvolňování paměti, případně dalším problémům. A paměti je zde velmi málo, oproti běžnému PC. Výchozí limit haldy je pro proces v rozmezí 12 – 32 MB, který je většinou stanoven dle hardwarové konfigurace zařízení. S výkonem také souvisí spotřeba, což je pro mě zcela nová zkušenost. Existují doporučené postupy pro vývoj aplikací tak, aby nespotřebovávaly zbytečně energii a nedocházelo tak k rychlému vybíjení baterie.

Zklamáním pro mě byl, bohužel, Android emulátor. Jediný nástroj, jak vyzkoušet zdrojový kód i na hardwarově rozdílných zařízeních. Rychlost emulátoru je velmi tristní, s vyšším rozlišením se výkon razantně zhoršuje a stává se téměř nepoužitelný pro použití. Složitější odlaďování chyb je nemyslitelné. Veškeré ladění kódu jsem proto musel provádět přímo na fyzickém telefonu.

Testování bohužel neproběhlo v tak velké míře, jak se původně předpokládalo. V každém případě byly uživatelské připomínky a požadavky přínosné pro použití. V některých pohledech ale velmi odlišné od mých představ. Čímž jsem si opět ověřil, že pohled vývojáře a pohled uživatele na použitelnost a funkce, může být velmi odlišný.

6.1 Pokračování vývoje

Dalších cest k rozšiřování implementace se jistě najde velké množství. Zajímavým prvkem by bylo umožnit uživateli si klávesnici dodatečně definovat vlastním způsobem. Např. rozvržení klávesnice pro speciální znaky, příp. definici těchto kláves. Tato funkcionalita by si však žádala kompletní přepsání třídy *Keyboard*.

Ačkoliv se jedná převážně o datový kontejner, třída např. generuje obrázky ve formě popisků kláves, pokud jsou definovány textově, počítá rozvržení kláves na displeji apod. Třída bohužel při zpracování XML souboru, který v tuto chvíli slouží jako definiční zdroj pro klávesnici, nenabízí žádné rozhraní tak, aby se do tohoto procesu dalo zasáhnout. Ačkoliv lze třídu rozšířit a jednotlivě měnit klávesy, nepovažuji toto z hlediska návrhu za šťastné řešení, protože tak dochází ke zbytečnému spouštění kódu, který vygeneruje klávesnici, která následně bude dalším kódem “přepsána“.

Za vhodné také považuji implementovat řešení pro definování tzv. skinů pro klávesnici. Klávesnice v původní implementaci umožňuje definici pouze jediného pozadí pro klávesy. Speciální klávesy jako SHIFT, ENTER, DELETE a další je podle mého názoru vhodné barevně odlišit. O vykreslování se stará objekt *KeyboardView*, bohužel, ani zde nejde do procesu vykreslování nijak zasahovat. Bohužel s vykreslováním souvisí i zpracování

dotykových událostí od uživatele. Z tohoto důvodu, aby byla v rozumné míře tato funkcionalita zajištěna, by bylo potřeba implementovat zcela nově zpracování těchto událostí.

Další funkcionalitou, která by mohla být užitečná, je implementace multi-dotykového zpracování událostí. Uživatel by tak nemusel např. přepínat klávesnici do stavu SHIFT, ale současným dotykem na klávesu SHIFT a další klávesy by došlo k zapsání velkého písmene, stejně jako je tomu na běžné PC klávesnici. Obdobně by se toho dalo využít u znaků s diakritikou.

Rozšíření slovníku by bylo vhodné např. o modul, který by postupně analyzoval všechny možné textové zdroje, jako SMS zprávy, emaily apod. Tato slova pak zpracoval a zapsal do slovníku, aby byla použitelná při nápovědě. Slovník v tuto chvíli nemá žádný zdroj např. pro nespisovná slova, která jsou často v mobilní komunikaci využívána.

LITERATURA

Použitá literatura

[1] ABLESON W. Frank, SEN Robi, KING Chris. *Android in Action, Second Edition*. Manning Publications Co. January, 2011. 592 pages. ISBN: 9781935182726

[2] CROCHEMORE M, RYTTER W., Text Algorithms, Oxford University Press, New York, 1994, 412 pages. ISBN 0-19-508609-0.
Dostupné na WWW: <http://www-igm.univ-mlv.fr/~mac/REC/B1.html>

Použité internetové zdroje

[3] *About SQLite*. [online]
Dostupný na WWW: <http://www.sqlite.org/about.html>

[4] Eclipse Foundation, The. *Eclipse*. [online]
Dostupný na WWW: <http://www.eclipse.org/>

[5] Google. *Android SDK | Android Developers*. [online]
Dostupný na WWW: <http://d.android.com/sdk/>

[6] Google. *Designing for Accessibility*. [online]
Dostupný na WWW:
<http://developer.android.com/guide/practices/design/accessibility.html>

[7] Google. *Designing for Performance*. [online]
Dostupný na WWW:
<http://developer.android.com/guide/practices/design/performance.html>

[8] Google. *Designing for Responsiveness*. [online]
Dostupný na WWW:
<http://developer.android.com/guide/practices/design/responsiveness.html>

[9] Google. *Platform Versions | Android Developers.* [citováno 20. 4. 2011]
Dostupný na WWW:
<http://developer.android.com/resources/dashboard/platform-versions.html>

[10] Google*. Testing Fundamentals | Android Developers*. [online]
Dostupný na WWW:
<http://developer.android.com/guide/topics/testing/testing_android.html>

[11] *ObjectAid UML Explorer – Home*. [online]
Dostupný na WWW: <http://www.objectaid.com/>

[12] Oracle. *Dictionaries – OpenOffice.org Wiki* [online] Stránka byla naposledy editována 24. 4. 2011 v 22:03 [citováno 11. 5. 2011].
Dostupný na WWW: <http://wiki.services.openoffice.org/wiki/Dictionaries>

[13] Romain Guy. *Avoiding memory leaks*. [online]
Dostupný na WWW:
<http://android-developers.blogspot.com/2009/01/avoiding-memory-leaks.html>

[14] *SQLite Documentation*. [online]
Dostupný na WWW: <http://www.sqlite.org/docs.html>

[15] StatCounter. *Top 8 Mobile OSs from Jan 10 to Mar 11 | StatCounter Global Stats*. [online]. [citováno 20.4.2011]
Dostupný na WWW:
<http://gs.statcounter.com/#mobile_os-ww-monthly-201001-201103>

[16] *Tech Team Lead News: Attacking memory problems on Android*. [online]
Dostupný na WWW:
<http://ttlnews.blogspot.com/2010/01/attacking-memory-problems-on-android.html>

[17] Wikipedie. *Android (operační systém)* - *Wikipedie* [online]. Stránka byla naposledy editována 24. 4. 2011 v 22:32. [citováno 1. 5. 2011].
Dostupný na WWW:
<http://cs.wikipedia.org/wiki/Android_%28opera%C4%8Dn%C3%AD_syst%C3%A9m%29>

[18] Wikipedie. *Čeština* - *Wikipedie* [online]. Stránka byla naposledy editována 15. 4. 2011 v 19:21 [citováno 20. 4. 2011].
Dostupný na WWW: <http://cs.wikipedia.org/wiki/%C4%8Ce%C5%A1tina>

PŘÍLOHA A

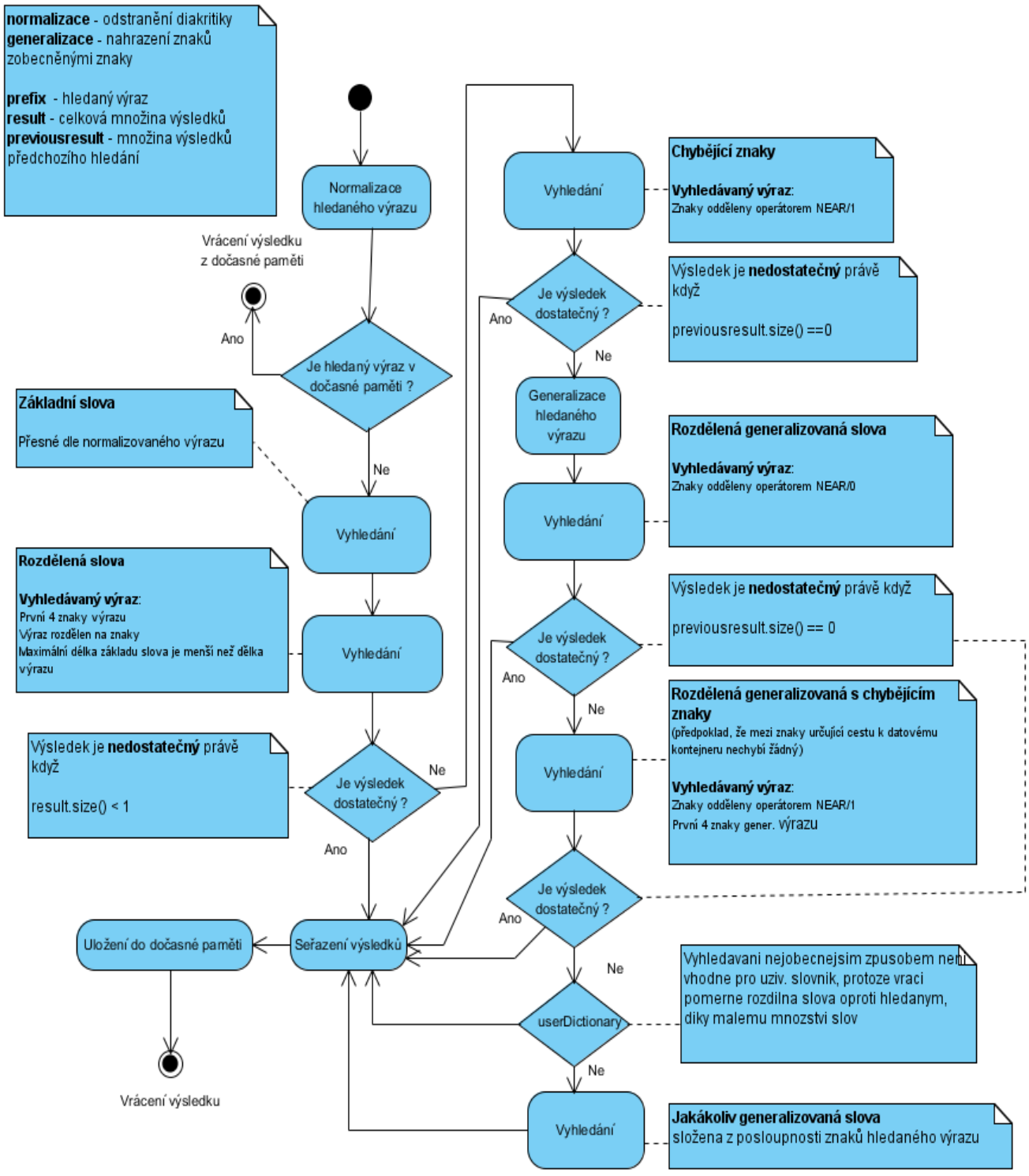
Použité zkratky

API	Application programming interface. Rozhranní pro programování aplikací.
AZERTY	Rozložení kláves klávesnice, kde první řádek písmenných kláves začíná znaky AZERTY.
DPad	Directional pad. Ovládací prvek pro definování směru pohybu.
GPL	General Public License. Licence pro svobodný software
kB	Kilobyte, jednotka množství informace.
QWERTY	Rozložení kláves klávesnice, kde první řádek písmenných kláves začíná znaky QWERTY.
QWERTZ	Rozložení kláves klávesnice, kde první řádek písmenných kláves začíná znaky QWERTZ.
SDK	Software Development Kit. Balík nástrojů určený pro vývoj aplikací.
SQL	Structured Query Language. Strukturovaný dotazovací jazyk.
URI	Uniform Resource Identifier. Jednotný identifikátor zdroje.
VM	Virtual machine. Virtuální stroj.
XML	Extensible Markup Language Značkovací jazyk.

PŘÍLOHA B

Diagramy

Obr. 31 Diagram aktivity vyhledání v jazykovém slovníku

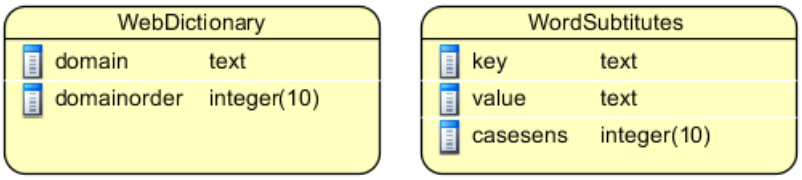


PŘÍLOHA C

Struktura obecného datového uložiště ExtKeyboard.sqlite

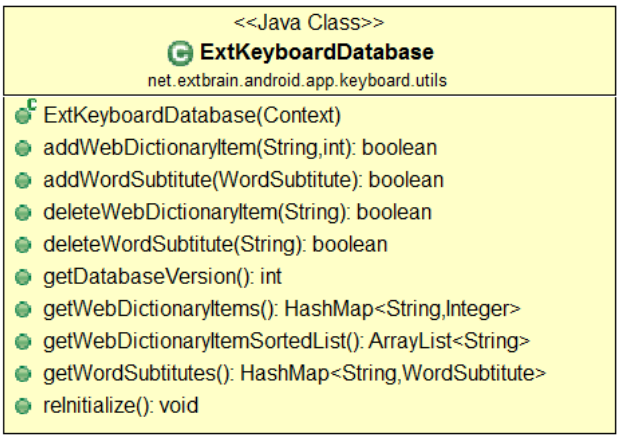
Tento soubor slouží jako externí datové uložiště. V současnosti se používá pro uložení webového slovníku a položek pro nahrazování slov. Následující diagram zobrazuje strukturu databáze. V dalším vývoji by měl být tento soubor využíván pro ukládání dat.

Obr. 32 Model databáze ExtKeyboard



Implementovaná třída poskytující přístup k tomuto databázovému soboru je *ExtKeyboardDatabase.*

Obr. 33 Diagram třídy ExtKeyboardDatabase



PŘÍLOHA D

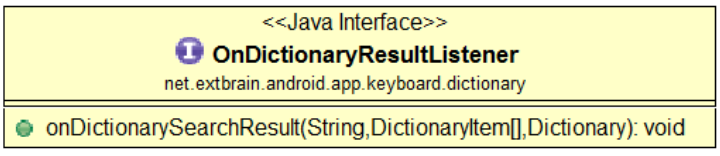
Vlastní implementace slovníku

Vlastní implementaci slovníku lze provést implementací abstraktní třídy *Dictionary*. Tato abstraktní třída má v základu implementovanou metodu *getDictionaryItemsAsync.* Tato metoda spouští v novém vlákně metodu *getDictionaryItems. O*statní neabstraktní metody jsou pouze k základní funkčnosti a předpokládám, že v samotné implementaci bude jejich funkce nově implementována.

Obr. 34 Abstraktní třída Dictionary

Pro asynchronní volání je nutné implementovat rozhraní *OnDictionaryResultListener.*

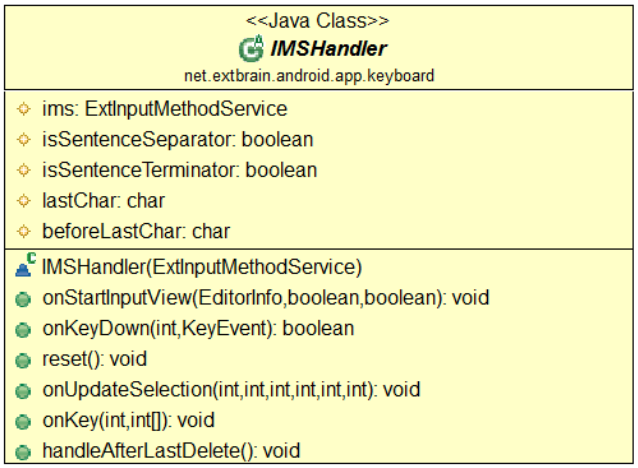
Obr. 35 Rozhraní OnDictionaryResultListener



Vlastní implementace klávesnice

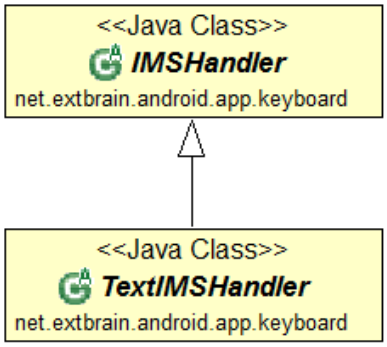
V případě vzniku nové klávesnice je možno definovat nové chování implementací abstraktní třídy *IMSHandler*. Tato třída má implementované pouze velmi základní funkce, které jsou v mém případě shodné pro všechny typy klávesnic (Kapitola 4.2 Klávesnice).

Obr. 36 Diagram abstraktní třída IMSHandler



V případě vzniku nové klávesnice, která bude určena pro nějakou omezenou množinu zadávání textu, je vhodnější implementovat abstraktní třídu *TextIMSHandler*, která je rozšířena o různé prvky vhodné při psaní textu.

Obr. 37 Diagram třídy IMSHandler a TextIMSHandler



PŘÍLOHA E

Obsah přiloženého CD

Název	Popis
/XD36DIP_JBruchanov.pdf	Diplomová práce
/UzivatelskyManual.pdf	Uživatelský manuál
/ReferencniManual.pdf	Referenční manuál
/src/*	Zdrojový kód klávesnice
/bin/ExtKeyboard.apk	Binární soubor klávesnice
/dictionary/*.db	Vygenerované slovníky
/support/DBGenerator/	Generátor databáze
dbgen.jar	JAR soubor pro generování databáze
gen-default-db-from-OO.cmd	Skript s příkladem pro vygenerování výchozího slovníku ze souborů zpracovaných pomocí nástroje OpenOfficeParser
gen-user-db-empty.cmd	Skript s příkladem pro vygenerování prázdné uživatelské databáze
gen-user-db-and-add-from-userdbtxt.cmd	Skript s příkladem pro vygenerování uživatelské databáze ze souboru definovaný uživatelem
userdb/userdb.txt	Příklad textového souboru pro import do uživatelské databáze
src/*	Zdrojový kód pro dbgen.jar
openofficeparser/*	2 soubory, které jsou výstupem nástroje OpenOfficeParser
/support/KeyIconMaker	Aplikace určená pro generování ikon pro jednotlivé klávesy
KeyMaker.exe	Aplikace
*.xml	Definiční soubory klávesnice stejné jako využívá Android.
src/*	Zdrojový kód pro KeyMaker.exe
/support/OpenOfficeParser	Nástroj pro vygenerování slov ze slovníku OpenOffice
OOWordGenerator3.exe	Aplikace
parse-english.cmd	Skript s příkladem pro vygenerování výstupu pro dbgen.jar z anglického slovníku
parse-english-limit4.cmd	Skript s příkladem pro vygenerování výstupu pro dbgen.jar z anglického slovníku. Maximální délka slova z OpenOffice slovníku je 4 znaky.
parse-wordslist.cmd	Skript s příkladem pro vygenerování výstupu pro dbgen.jar z jednoduchého seznamu slov.
src/*	Zdrojový kód pro OOWordGenerator3.exe