

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

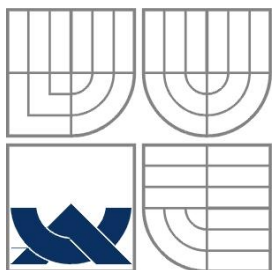
VYHLEDÁVÁNÍ HLASEM NA INTERNETU

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

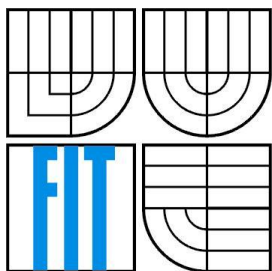
AUTOR PRÁCE
AUTHOR

MICHAL BELOBRAD

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

VYHLEDÁVÁNÍ HLASEM NA INTERNETU

INTERNET VOICE SEARCH

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MICHAL BELOBRAD

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PETR SCHWARZ, Ph.D.

BRNO 2011

Abstrakt

Tato bakalářská práce se zabývá tvorbou aplikace pro dotykové telefony s operačním systémem Bada. Cílem této aplikace je umožnit uživatelům vyhledávat na internetu pomocí svého hlasu. Seznámíme se s telefonem Samsung Wave, pro který byla tato aplikace vyvíjena. Dále blíže se podíváme na zpracování výsledků rozpoznávače, našeptávače a jejich kombinace.

Abstract

This thesis is concerned with creating applications for touchscreen phones with the operating system Bada. The objective of this application is to allow users to search the web using their voice. We introduce with Samsung Wave for which this application was developed. In addition we look at the results of the recognizer processing, auto-complete, and their combination.

Klíčová slova

Samsung Wave, Bada, Bada SDK, aplikace pro Bada, rozpoznávání hlasu, vyhledávání na internetu.

Keywords

Samsung Wave, Bada, Bada SDK, application for Bada, voice recognizer, search over the internet.

Citace

Belobrad Michal: Vyhledávání hlasem na internetu, bakalářská práce, Brno, FIT VUT v Brně, 2011

Vyhledávání hlasem na internetu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Petra Schwarze, Ph.D.

Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Michal Belobrad

19. 4. 2011

Poděkování

Tímto bych chtěl poděkovat panu Ing. Petru Schwarzovi, Ph.D. za odbornou pomoc při práci na tomto projektu a poskytnutí licence k rozpoznávací hlasu.

© Michal Belobrad, 2011

Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů..

Obsah

Obsah	1
1 Úvod.....	3
1.1 Přehled kapitol	3
2 Zadání práce.....	4
2.1 Cíle projektu	4
2.2 Aplikace pro osobní počítač	4
2.3 Aplikace pro mobilní telefon	4
3 Použité technologie.....	5
3.1 Mobilní telefon	5
3.1.1 Samsung S8500 Wave	6
3.1.2 Operační systém Bada	7
3.2 Vývojové prostředí	7
3.2.1 Eclipse.....	7
3.2.2 Bada SDK.....	8
4 Aplikace pro mobilní telefon	9
4.1 Návrh uživatelského rozhraní	9
4.1.1 Požadavky.....	9
4.1.2 Realizace.....	9
4.1.3 Testování rozhraní	11
4.1.4 Konečný vzhled	11
4.2 Implementace.....	12
4.2.1 Pozadí vývoje.....	12
4.2.2 Formuláře.....	12
4.2.3 Záložky	15
4.2.4 Komunikace v rámci aplikace.....	15
4.2.5 Nahrávání hlasu	16
4.2.6 Přenos souborů.....	16
4.2.7 Otáčení displeje	17
4.2.8 Uchování dat.....	17
4.3 Distribuce aplikace	18
5 Server	19
5.1 Princip činnosti	19
5.2 Zpracování zvuku	20
5.3 Rozpoznání hlasu.....	21

5.3.1	Výstupy rozpoznávače.....	21
5.3.2	Ohodnocení rozpoznávačem.....	24
5.4	Našeptávač.....	24
5.4.1	Google Suggest.....	24
5.4.2	Nejčastěji hledané výrazy.....	26
5.4.3	Ohodnocení našeptávačem.....	27
5.5	Výsledné ohodnocení a tvorba vět.....	27
5.5.1	Sumarizace pravděpodobností.....	27
5.5.2	Spojení slov do vět.....	27
6	Testování.....	29
6.1	Samotný rozpoznávač hlasu.....	29
6.2	Aplikace.....	30
6.3	Zhodnocení testování.....	31
7	Závěr.....	32
7.1	Možná rozšíření do budoucna.....	32
7.2	Podobné projekty.....	33
	Literatura.....	34
	Příloha A.....	35

1 Úvod

Tato bakalářská práce vznikla na půdě Fakulty informačních technologií Vysokého učení technického v Brně. Práce je zajímavá v tom, že se zaměřuje na poměrně nové odvětví vývoje aplikací – aplikací pro *chytré mobilní telefony*. Jak již název této práce vypovídá, jeho hlavním tématem bylo vytvořit software pro vyhledávání hlasem na internetu. Aplikace byla vytvořena pro operační systém Bada od firmy Samsung. Testování a vývoj probíhaly na mobilním telefonu Samsung S8500 Wave.

1.1 Přehled kapitol

V kapitole 2 se dozvíme, jaké bylo zadání práce, jaké se nabízeli možnosti jeho vypracování a výhody/nevýhody jednotlivých variant. Kapitola 3 se také zabývá obecnými prvky, které byly potřeba, od mobilního telefonu na kterém byla aplikace testována, až po vývojové prostředí Bada SDK. Kapitola 4 pojednává o návrhu výsledné aplikace a jejím vývoji. Dozvíme se, jak jsou navrženy jednotlivé prvky operačního systému Bada a jak jsou poté využity v rámci tohoto projektu. Cílem kapitoly 5 je popis serverové části aplikace. Na serveru probíhají veškeré ohodnocující výpočty a přepis hlasu na text. Kapitola 5.3 se věnuje té části serveru, která má na starosti právě rozpoznání hlasu a ohodnocení výsledků pro další použití. V části 5.4 je naopak popsána ta část, která se zabývá ohodnocením výsledků z hlediska vyhledávání na internetu, konkrétně o ohodnocení našeptávačem. Kapitola 6 pak nabízí porovnání výsledků od samotného rozpoznávače a výsledků z aplikace.

2 Zadání práce

2.1 Cíle projektu

Cílem této práce bylo vytvořit software, pomocí kterého by uživatel mohl vyhledávat na internetu mluveným slovem bez nutnosti psát hledaný výraz ručně na klávesnici. Aplikace využívá rozpoznávače řeči k přepisu hlasu do textové formy a seznamem nejčastěji hledaných výrazů v internetových vyhledávačích. K dosažení lepších výsledků tyto informace vhodně spojuje a vyvažuje tak chyby rozpoznávače, kdy se nabízí více možností pro překlad daného výrazu, nebo kdy není slovo zcela zřetelné.

Toto zadání nabízelo více různých podob výsledného produktu. Nicméně po konzultaci s panem Ing. Petrem Schwarzem, Ph.D. připadaly v úvahu dvě možnosti, jak by měla výsledná aplikace vypadat.

2.2 Aplikace pro osobní počítač

První varianta se zaměřovala na využití v osobních počítačích, například ve formě přídatku do některého z internetových prohlížečů (Firefox, Chrome, Opera atd.) nebo jako miniaplikace běžící na pozadí operačního systému. Tato forma by byla užitečná například pro dotykové terminály v knihovnách nebo pro osobní užití těch, kteří by si chtěli udělat vyhledávání na internetu pohodlnějším.

2.3 Aplikace pro mobilní telefon

Druhou možností bylo vytvoření aplikace pro *chytré mobilní telefony*. Ta by bylo vhodná pro rychlé vyhledání informací na místě, kde je ruční zapsání hledaného výrazu nepraktické a zdlouhavé. Což platí zejména právě pro dotykové mobilní telefony bez hardwarové klávesnice. Představte si situaci, kdy jdete po ulici a potřebujete si rychle něco vyhledat na internetu a někam spěcháte. Bez této aplikace by uživatel byl nucen na pár okamžiků zastavit a text v klidu napsat, nebo zkusit psát za chůze a výraz několikrát opravovat a přepisovat. Pozitivum pro tuto variantu je i možnost masového rozšíření mezi koncové uživatele díky obchodům s aplikacemi přímo v mobilním telefonu.

Tato forma nakonec zvítězila i díky tomu, že vývoj aplikací pro mobilní telefony spojuje něco, co mě zajímá s něčím zajímavým, neokoukaným.

3 Použité technologie

3.1 Mobilní telefon

V dnešní době se mobilní telefony těší masivnímu rozmachu a trendy ukazují, že růst bude pokračovat. Novinkou posledních několika let jsou tzv. *smartphony*, česky *chytré telefony*. Takový telefon se od toho klasického liší tím, že mj. poskytuje některé pokročilé funkce, jako jsou např. video hovory nebo gps navigace. Hlavní výhodou těchto zařízení je aplikační rozhraní, které umožňuje instalaci, nebo úpravy nových programů. Dle agentury IDC (International Data Corporation¹) trh s mobilními telefony ve třetím čtvrtletí roku 2010 vykázal téměř 90% meziroční nárůst. V počtech prodaných telefonů to znamená, že se v roce prodalo 81 miliónů přístrojů, oproti necelým 42 miliónům za rok 2009. Za celý rok 2010 pak počet prodaných mobilů s lehkostí přesahuje hranici 200 miliónů prodaných kusů.

Chytré mobilní telefony postupně nahrazují starší, *hloupé telefony*, a trh s aplikacemi pro ně určených nabírá na obrátkách. Číslo stažených aplikací už nejsou v řádech miliónů, dokonce ani stovek miliónů, nýbrž miliard. Například operační systém Android očekává v měsíci květnu (rok 2011) překročení celých čtyř miliard stažených aplikací a to má nyní na trhu operačních systémů zhruba 30% podíl.

V pozadí těch nejrozšířenějších operačních systémů, jako jsou již zmíněný **Android** od Google Inc., **Symbian** vyvíjený společností Symbian Foundation (původně Symbian Ltd.) nebo **iOS**, operačního systému firmy Apple Inc., se ukrývá nový člen do této rodiny – **Bada**. Ten tak nabízí další alternativu operačního systému pro chytrý mobilní telefon. A právě pro tento systém byla vyvinuta aplikace, která je předmětem této zprávy.

Aplikace byla vyvíjena na mobilním telefonu Samsung S8500 Wave, vyráběným společností Samsung Electronics. Tento telefon byl s nadsázkou označován jako „zabiják“ iPhonů. Hlavním tahákem pro pořízení tohoto telefonu byl hlavně poměr cena/výkon, kdy neměl na trhu konkurenci. Tento telefon byl ve své době vlajkovou lodí platformy Bada a prodalo se ho přes jeden milion kusů během prvních čtyř týdnů prodeje a nyní ho prý vlastní 15 miliónů uživatelů.

¹ <http://www.idc.com>

3.1.1 Samsung S8500 Wave

Dotykový telefon Samsung S8500 Wave (obrázek 3.1) byl uveden na trh k 1. červnu 2010 jako vlajková loď nového operačního systému Bada.

Wave má 3,3 palcový kapacitní displej o rozlišení 480 na 800 pixelů a jako jeden z prvních telefonů vůbec, využívá technologie Super AMOLED², která se staví vedle Retina displeje (iPhone 4) ke špičce současných displejů. SAMOLED displej je zajímavý tím, že nemá žádnou podsvětlučící vrstvu, jako např. LCD displeje, ale jednotlivé obrazové body svítí samy. Tímto řešením dosáhl Samsung téměř dokonalé černé barvy.

Wave je poháněn cpu s kódovým označením „*Hummingbird*“ (S5PC110) od společnosti Samsung. „*Hummingbird*“ obsahuje 1GHz ARM Cortex-8 cpu a výkonné GPU Power VR SGX 540, které je schopné generovat 90 MTPS (milionů trojúhelníků za vteřinu). Samsung Wave má 512MB RAM, z kterých je pouze 256MB využito pro aplikace uživatele, což občas vedlo k nedostatkům volných zdrojů a následným problémům (nejnovější firmware už s pamětí pracuje efektivněji a k těmto problémům už nedochází tak často).

Přímým následovníkem Wave je model Samsung S8530 Wave II, který se Samsung rozhodl uvést zejména kvůli nedostatku SAMOLED displejů. Tyto dva modely jsou prakticky identické, jedinou odlišností tvoří displej, kdy se musel Samsung u Wave II vrátit zpět k LCD displejům. Zdroj informací poskytl [6].



Obr. 3.1 Samsung S8500 Wave

² AMOLED je označení pro aktivní matici organických svítivých diod

3.1.2 Operační systém Bada

Jak již bylo zmíněno výše, operační systém Bada je dílem Samsung Electronics. Ačkoli byla Bada uvedena už 10. listopadu 2009, prvním telefonem s tímto systémem se stal až Samsung Wave v polovině roku 2010. Od té doby se Bada postupně rozšiřuje na celou škálu telefonů od levných low-end telefonů až po high-end.

Samsung Wave při svém uvedení na trh obsahoval Badu ve verzi 1.0. Nedlouho poté vyšla verze 1.0.2, která upravovala několik menších chyb. Nyní (od prosince 2010) je dostupná Bada ve verzi 1.2, která obsahuje mj. technologii swype³. Jedná se o psaní textu, kdy se prstem pohybujete nad písmeny z výsledného slova a swype tuto trasu vyhodnocuje a podobně jako T9 nabízí odpovídající slova.

Jak Samsung uvádí, Bada není pouze operační systém, ale platforma s konfigurovatelným jádrem, což umožňuje použití jiného vhodného RTOS (real-time operační systém) jádra, nebo linuxového jádra. Bada využívá části z FreeBSD, NetBSD a OpenBSD a poskytuje vývojářům API v jazyku C++.

Bada nabízí celou škálu základních UI kontrolérů, které tak usnadňují programátorům při vývoji aplikací práci. Bada obsahuje webový prohlížeč založený na open-source webKit technologii, který podporuje Adobe Flash 9. Jak WebKit, tak Flash je dále využíván v interních Bada aplikacích (Facebook, Twitter klient atd.).

Samsung v Badě využívá všechny dostupné technologie, od detektoru obličeje např. ve fotoaparátu pro ostření, přes akcelerometr užívaný zejména pro ovládání her, až po GPS využitelném v celé škále aplikací. Jako zdroj informací sloužil [3] a [4].

3.2 Vývojové prostředí

3.2.1 Eclipse

Eclipse⁴ je vývojářská open-source platforma určená zejména pro vývoj aplikace v programovacím jazyce Java. Díky její flexibilitě je však možné toto prostředí pomocí pluginů používat i pro jiné jazyky, jakou jsou například C, C++, Perl, PHP, Python, Ruby a další.

Největší výhodou Eclipse oproti ostatním vývojovým prostředím je právě jeho rozšiřitelnost pomocí pluginů. Na základě této volnosti začali pod platformou Eclipse různé její odnože, která pak

³ <http://www.swypeinc.com/>

⁴ <http://www.eclipse.org/>

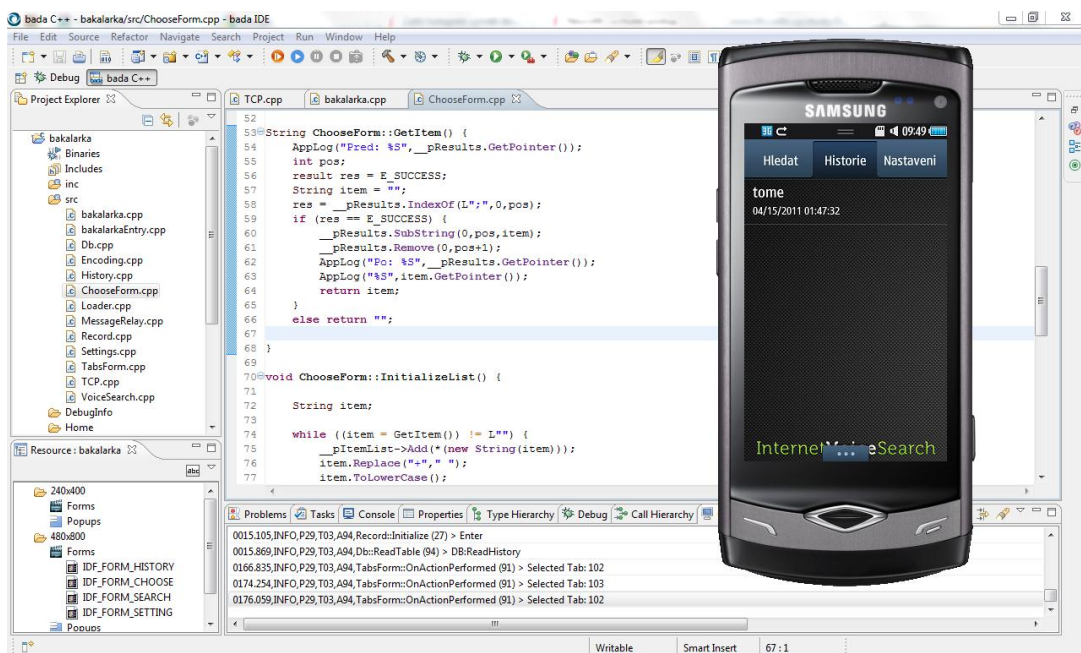
usnadňují integraci všech potřebných rozšíření pro různé oblasti vývoje Java aplikací (jako např. aplikační server, nebo nástroj pro vizuální návrh uživatelských rozhraní).

Původně Eclipse (Eclipse 1.0) vznikl jako projekt společnosti IBM, vyvinutý firmou Object Technology International (OTI). V listopadu 2001 uvolnil IBM Eclipse pod CPL licenci (později přelicencován na EPL) a dnes se tento příspěvek do open-source rodiny odhaduje na 40 milionů dolarů.

Uvedení Eclipse 3.0 dodalo této platformě ještě více lesku. V této verzi se Eclipse adaptoval na standart OSGi R4. Díky této architektuře Eclipse dynamicky nahrává pluginy až v okamžiku jejich potřeby, čímž se minimalizuje potřeba systémových prostředků a zkracuje se čas startu aplikace. Informace v této kapitole čerpají z [5].

3.2.2 Bada SDK

Bada SDK (Software Development Kit, 3.2) je IDE založené na Eclipse CDT. CDT značí C/C++ Development Toolkit. Je to kolekce Eclipse rozšíření umožňující vyvířet, editovat, překládat a debugovat projekty psané v C a/nebo C++ jazyku. Bada SDK obsahuje nástroje pro vývoj aplikací, pro jejich simulaci a nástroje pro návrh uživatelských rozhraní (podporující UI Samsung TouchWiz - grafickou nádstavbu mobilních telefonů od společnosti Samsung). Jednou z věcí, která se mi na Bada SDK líbí nejvíc je integrovaná Bada API dokumentace, která je propojená přímo s editorem. Tato kapitola čerpala z [4].



Obr. 3.2 Bada SDK

4 Aplikace pro mobilní telefon

4.1 Návrh uživatelského rozhraní

Návrh uživatelských rozhraní je často opomíjen a odsouvá se až na druhou kolej vývoje softwaru spolu s jinými, spíše kvalitativními aspekty. Přitom je vhodný návrh uživatelského rozhraní jedním z klíčových aspektů, které ovlivňují úspěšnost a použitelnost výsledného systému. V praxi návrh uživatelského rozhraní řeší některé softwarové firmy pouze tím, že si stanoví vlastní interní pravidla a specifikace a výsledné produkty pak bývají často nepřehledné a firma pak musí pořádat pro klienty různá školení.

4.1.1 Požadavky

Při navrhování uživatelského rozhraní byl kladen důraz zejména na jednoduchost a přehlednost. Jelikož je aplikace určena pro dotykové ovládání, bylo potřeba ji i tomu uzpůsobit. Bylo nezbytné zamyslet se nad tím, jak se aplikace bude používat (při chůzi, ovládání jednou rukou atd.) a jednotlivé ovládací prvky rozmístit tak, aby byly lehce stisknutelné a přesně tam, kde to uživatel čeká.

V neposlední řadě jsem se zaměřil také na grafický vzhled celé aplikace. Z vlastní zkušenosti soudím, že aplikace, u které se grafický vzhled zcela opomenul, nemá takový úspěch, jako aplikace na které si dal někdo záležet a podle toho i vypadá. To že design často nepřímou znamená úspěch, dokazuje např. iOS (ať už verze pro iPhone, nebo iPad), kde je skloubena využitelnost a „krása“. Tento systém tak v dnešní době slouží jako inspirace pro ostatní mobilní operační systémy.

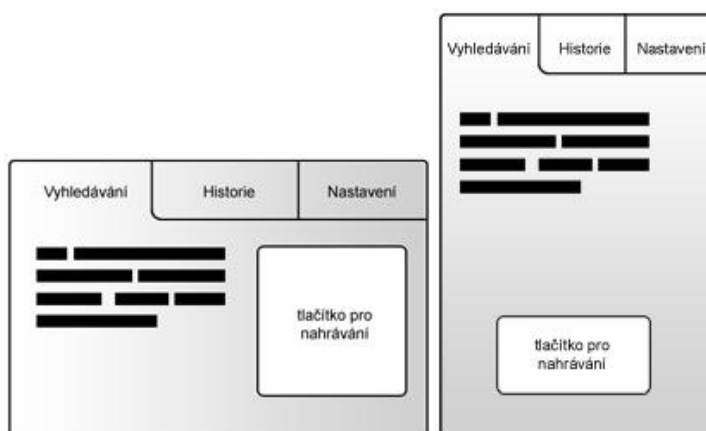
4.1.2 Realizace

Aby bylo možné aplikaci snadno ovládat i s většími prsty nebo za nepříznivých podmínek a zároveň nedocházet k překlepům, umožňuje aplikace otočení displeje na šířku a tím zvětšení všech ovládacích prvků na více než dvojnásobek. Po otočení displeje se rozhraní samo uzpůsobí a uživatel tak není omezen jen na jednu polohu (návrh rozhraní ba obrázku 4.1).

Aby nebyl uživatel po spuštění aplikace přesycen informacemi, je samotné rozhraní rozděleno do tří kategorií, které odpovídají třem hlavním částem aplikace. Každá kategorie má vlastní záložku, která navozuje pocit jakéhosi pořadače. Mezi tyto hlavní části patří:

- Vyhledávání
- Historie dotazů
- Nastavení programu

Vyhledávání, jak již název napovídá, je jakousi hlavní obrazovkou celé aplikace. V této záložce se bude uživatel vyskytovat nejčastěji a také proto jsem se zaměřil na její jednoduchost. Podle aktuálního stavu aplikace mění tlačítko svůj popis a ve fázi kdy se nahrávka vyhodnocuje, je tlačítko zcela zablokované. Dále se uživateli na obrazovce ukazuje, jakou akci zrovna aplikace vykonává a po jejím ukončení také její stav. Ukončit nahrávání lze buď opětovným stisknutím hlavního tlačítka, nebo vyčkáním 10 vteřin (což je maximální doba nahrávání). O tom kolik času do ukončení nahrávky zbývá je uživatel informován prostřednictvím jednoduchého progress baru těsně nad tlačítkem. Po ukončení všech uživatelských aktivit a úspěšného zaznamenání hlasu dojde k přenosu dat na server. Tento přenos je realizován taktéž pomocí progress baru, ovšem tentokrát ve vyskakovacím okně, aby měl uživatel na očích, že program pracuje správně a něco se v něm děje.



Obr. 4.1 Návrh uživatelského rozhraní

Záložka historie obsahuje všechny předchozí úspěšné dotazy, které uživatel zvolil za správné. Ty jsou přehledně seřazeny podle data jejich vytvoření a uspořádány do seznamu. V případě že seznam obsahuje více položek než by se vešlo na obrazovku, může uživatel seznam procházet jednoduchým pohybem prstu po displeji v požadovaném směru. Pokud se uživateli zdá, že už je seznam příliš obsáhlý, může vyvolat kontextovou nabídku a seznam promazat. Tato nabídka se skrývá pod posuvníkem na spodní straně obrazovky.

Poslední záložka se věnuje nastavení důležitých informací pro připojení serveru. Stav nastavení lze snadno rozpoznat dle barvy písma u příslušného popisku. Pokud je zadaná hodnota špatná a neodpovídá požadovanému tvaru, je popisek zvýrazněn červeně. Pokud je hodnota správná, avšak ještě nebylo nastavení uloženo, je popisek bílý. Zároveň je zobrazeno upozornění, že je třeba celé nastavení uložit, to se provede stisknutím tlačítka „Uložit“. Pokud jsou data validní a jsou uložena, mají popisky barvu zelenou. Zároveň je uživatel na úspěšné uložení informován textově.

4.1.3 Testování rozhraní

Testování je důležitou součástí návrhu každého uživatelského rozhraní. Vývojář může navrhnout rozhraní aplikace sebelépe, avšak dokud si to neověří u potenciálních uživatelů, zůstává to pouze jeho domněnkou.

Můj návrh není svým rozsahem nijak extrémní, jedná se pouze o jednoduchou aplikaci bez nějakých revolučních prvků, ale i přesto jsem se rozhodl rozhraní otestovat a zkusit i to málo nějak vylepšit. Testování jsem provedl formou pozorování. To v praxi znamená, že jsem telefon s aplikací zapůjčil několika vyvoleným 5 kamarádům a přítelkyni s rodinou. Všichni kamarádi jsou studenty informačních technologií a pohybují se v oboru IT. Rodina v rámci toho testování zaujímá pozici laické veřejnosti.

Každému jsem dal jednoduché tři úkoly, bez toho abych jim rozhraní nějak vysvětloval. Poté jsem pozoroval jak tyto úkoly zvládnout. Jednalo se o tyto úkoly:

- Vyhledej automobilový klub poblíž Prahy
- Vymaž historii
- Změň cílovou IP adresu serveru

Nebylo překvapením, že podstatně větší úspěšnost byla na straně studentů IT. Ti zvládli všechny úkoly velice rychle, víceméně bez větších zádrhelů. Jedinou problémovou částí bylo vymazání historie, kdy většina z nich nepřišla nikdy dřív se systémem Bada do styku a tak chvíli hledali to správné tlačítko, které tuto nabídku vysunulo. Odezva na aplikaci byla kladná a jediná připomínka padla k části s nastavením. Šlo o to, že nebylo zcela jasně patrná nutnost toto nastavení uložit a zda se uložení podařilo. Docházelo pak k tomu, že uživatel nastavení změnil, ale už data neuložil. To mě přivedlo k myšlence barevného označení všech popisků (červená – chyba, bílá – neuloženo, zelená – uloženo, vše v pořádku) a přidání informativního textu.

Rodina už si nevedla zdaleka tak dobře. Všichni zvládli nahrát hledaný výraz a vyčkat do otevření okna vyhledávače, ale další dva body, zejména nastavení, dopadly špatně. Je však třeba přihlídnout ke skutečnosti, že nikdo z nich neměl představu co je to IP adresa. Většina uživatelů z řad laiků by ani tyto kroky nemusela podstupovat (měnit nastavení) a pokud ano, v budoucnu bych tuto situaci řešil přidáním textové nápovědy v popisu aplikace v SA.

4.1.4 Konečný vzhled

Vzhled aplikace se v průběhu vývoje prakticky neměnil, od začátku jsem se držel návrhem uživatelského rozhraní. Po provedení drobných úprav po testování rozhraní jsem dosáhl její finální podoby, která se líbila i vedoucímu mé práce Ing. Petru Schwarzovi Ph.D. Konečný vzhled aplikace je vidět na obrázku 4.2.



Obr. 4.2 Finální vzhled aplikace

4.2 Implementace

4.2.1 Pozadí vývoje

Při vývoji aplikace jsem se potýkal se spoustou problémů, spojenou zejména s nedostatkem naučných materiálů a tak jsem zkoumáním pozadí jednotlivých funkcí Bady strávil podstatně více času než samotným programováním. Vývojem softwaru pro operační systém Bada, bohužel na rozdíl od konkurence, se zabývá zatím pouze jediný knižní titul a to [3]. Dokonce ani na internetu nenajdete kromě oficiálního fóra⁵ a jednoho komunitního webu⁶ prakticky nic a jste tak odkázáni jen na pár ukázkových příkladů z SDK.

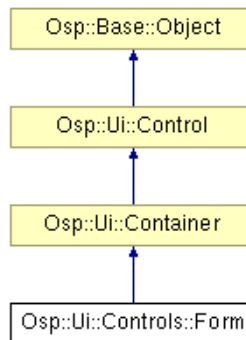
4.2.2 Formuláře

Formuláře jsou v podstatě tou nejdůležitější částí aplikace vyvíjené pro OS Bada. Zjednodušeně můžeme říci, že jsou to „okna“ programu, skrz které může uživatel aplikaci ovládat a být informován o jejím stavu. Do formulářů můžeme přidávat různé kontrolery (jako jsou tlačítka, panely, textová a editovatelná pole atd.), tvoří tak univerzální kontejnery pro všechny UI prvky aplikace. Čím je vyvíjená aplikace složitější, tím více formulářů by měla obsahovat. Dá se říci, že je lepší mít pár ovládacích prvků rozdělených do vícera formulářů dle jejich funkcionality, než mít jeden formulář těmito prvky přeplněný.

⁵ <http://developer.bada.com>

⁶ <http://www.badadev.com/>

V daném okamžiku může být zobrazen pouze jeden formulář, jehož data mohou záviset na ostatních formulářích (předávání informací mezi formuláři řeší kapitola 4.2.4). Formulář je třída Bada API, která se nachází pod jmenným prostorem `Osp::Ui::Controls::Form` (Obr. 4.3).



Obr. 4.3 Diagram dědičnosti `Osp::Ui::Controls::Form`. Převzato z [4].

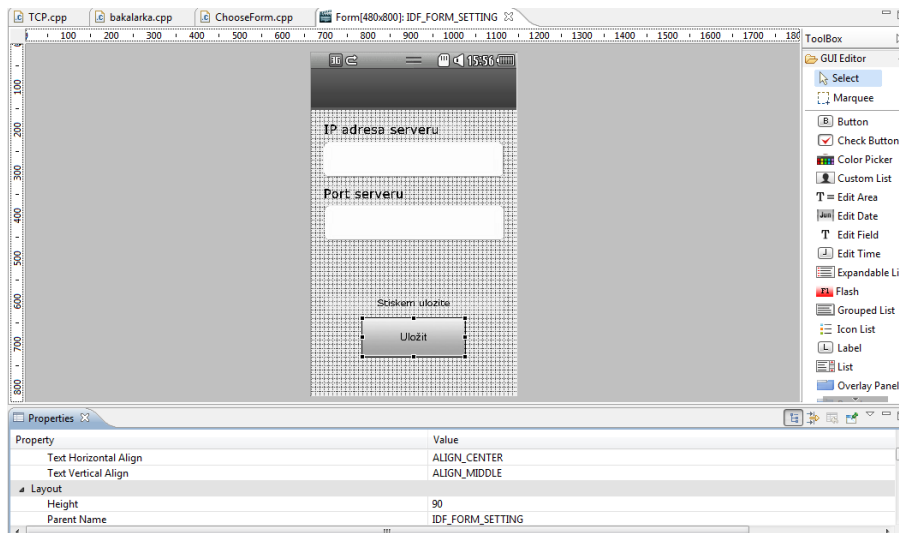
Vývojář má k dispozici dva zcela odlišné způsoby, kterými lze navrhovat, nastavovat rozhraní a poté vytvářet formuláře:

- Využití integrovaného UI designeru z SDK - pomocí tohoto nástroje (obrázek 4.4) můžete jednoduše a přehledně rozmístit prvky do formuláře tak, že si v seznamu *ToolBox* vyberete libovolný prvek a jeho přetáhnutím myší ho umístíte do vámi zvolené pozice. V okně *Properties* je pak možné nastavit prvku jeho identifikátor, jeho rozměry a další spoustu různých vlastností. Ve zdrojovém kódu se pak pomocí identifikátoru prvek přidělí zvolenému kontroleru (ukázka v kódu 4.1).

```
Label* __Search_pLabel = static_cast<Label *> (GetControl(L"IDC_LABEL2"));
```

Kód 4.1 Ukázka implementace ovládacího prvku z UI designeru

Samotný návrh v designeru je velice pohodlný. Vývojář má k dispozici různá pravítka a mřížky, pomocí kterých může jednotlivé prvky zarovnávat. Zároveň může vytvářet návrhy formulářů pro různé rozlišení a umožnit tak kompatibilitu s jinými zařízeními. Aplikace by si přitom měla sama volit, podle nativního rozlišení telefonu, jaký druh formuláře zvolit.



Obr. 4.4 Integrovaný nástroj pro tvorbu UI

- Tvorba UI přímo v kódu aplikace – pokud je potřeba, aby byl formulář plně dynamický, je vhodnější tvořit UI přímo ve zdrojovém kódu (ukázka v kódu 4.2). Je to sice trochu náročnější na představivost a celý návrh je trochu obtížnější, ale výsledek je stejně kvalitní jako u prvního zmiňovaného způsobu.

```

__pHistoryListFormat = new CustomListItemFormat();
__pHistoryListFormat->Construct();

__pHistoryListFormat->AddElement(ID_FORMAT_TEXT1,
    Rectangle(20, 10, 450, 60), 38);
__pHistoryListFormat->AddElement(ID_FORMAT_TEXT2,
    Rectangle(20, 60, 450, 25), 27);

__pHistoryList = static_cast<CustomList *>
    GetControl(L"IDC_HISTORY_LIST");
__pHistoryList->AddCustomItemEventListener(*this);

```

Kód 4.2 Ukázka implementace ovládacího prvku přímo ve zdrojovém kódu

Po správném nastavení, je potřeba formulář také vykreslit a zobrazit. K tomu slouží dvě jednoduché metody Draw a Show. Pro hlavní formulář se tyto metody volají při spuštění aplikace (ukázka v kódu 4.3). Je nutné si uvědomit, že formulář je i bez vykreslení aktivní a stále zabírá systémové prostředky. Vývojář by proto neměl zapomínat tyto prostředky uvolňovat, pokud již daný formulář nebude dále využívat.

```

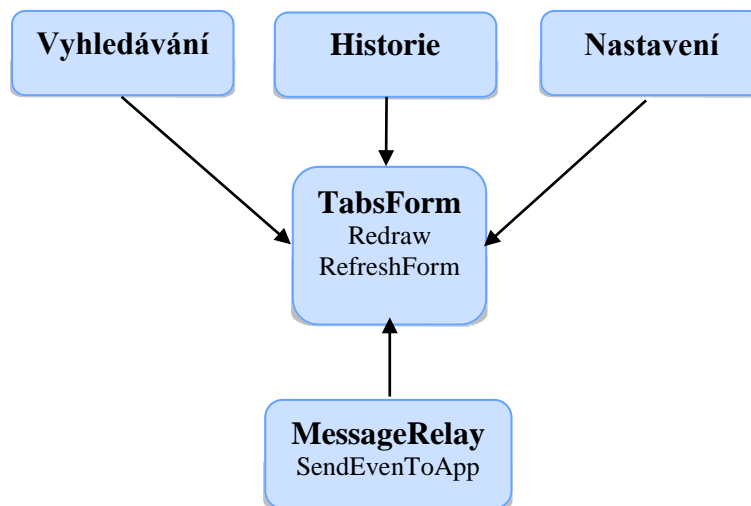
VoiceSearch pSearch = pSearch = new VoiceSearch();
pSearch->Initialize();
pSearch->Draw();
pSearch->Show();

```

Kód 4.3 Ukázka vykreslení a zobrazení formuláře

4.2.3 Záložky

Důležitou částí mé aplikace jsou záložky (diagram dědičnosti lze vidět na obrázku 4.5), skrze které se přistupuje k hlavním částem aplikace a dochází k přepínání formulářů. Všechny záložky se dědí z třídy `TabsForm`, která zastřešuje veškeré aktivity spojené s přepínáním záložek, jejich překreslováním a otáčením displeje. Způsob, kterým se aplikaci sdělí informace o tom, jaký formulář se má vykreslit, se podíváme v následující kapitole.



Obr. 4.5 Diagram dědičnosti pro záložky

4.2.4 Komunikace v rámci aplikace

Veškerá komunikace mezi formuláři probíhá přes třídu `MessageRelay`, která umožňuje odesílání zpráv instancí celé aplikace. Hlavní dvě funkce zprostředkovávající komunikaci jsou:

- `SendEventToApp(RequestId requestId, const Osp::Base::Collection::IList *pArgs);`
- `OnUserEventReceivedN(RequestId requestId, IList *pArgs);`

Pro praktickou ukázkou uvedu příklad, kdy si uživatel žádá přepnutí do záložky nastavení. Po klepnutí na záložku `TabsForm` zavolá funkci `SendEventToApp`, jejichž argumenty budou ID formuláře (v tomto případě `ID_TAB_SETTINGS`) a ID označující událost „změň záložku“ (`ID_CHANGE_TAB`). Hlavní třída aplikace tuto zprávu přijme funkcí `OnUserEventReceivedN`, dle ID události zjistí, že se žádá změna formuláře a změnu provede.

Mezi další takto přenášené informace patří:

- Aktuální stav TCP spojení
- Aktuální stav nahrávání z mikrofону
- Žádost o uložení dat do databáze aplikace

Pomocí této třídy lze instanci aplikace zaslat prakticky libovolnou informaci libovolného datového typu. Při odeslání se tato hodnota uloží ve formě seznamu hodnot (získání času ze seznamu v kódu 4.4). Jediné co musí adresát této funkce vědět je, jaký datový typ má na kterém místě seznamu očekávat.

```
DateTime time = *(static_cast<DateTime *> (pArgs->GetAt(2)));
```

Kód 4.4 Ukázka získání časové informace ze seznamu

4.2.5 Nahrávání hlasu

Původně byla k nahrávání použita třída `Osp::Media::AudioRecorder`. Ta však bohužel nenabízí možnosti jak nastavit výstupní formát, přitom rozpoznávač požaduje audio se vzorkovací frekvencí 8KHz s 1 kanálem. Bylo tedy nutné sestoupit o úroveň níž a využít třídu, která je `AudioRecorderem` nejspíš zapouzdřena. Jedná se o `Osp::Media::AudioIn`, která nabízí nastavení jak vzorkovací frekvence, tak i počtu kanálů. Je však daleko rozsáhlejší a složitější na integraci do aplikace.

`AudioIn` ukládá nahrané informace do přidělených bufferů, které je pak nutno zpracovat a uložit do souboru. Aby při nahrávání nevznikaly nějaké prodlevy vznikající v místě, kde se buffer ukládá do souboru, bylo nutné vytvořit bufferů několik a ty pak vhodně proházovat.

Jelikož není vhodné a ani se nedá očekávat, že by nahrávaný výraz měl délku větší než 10 vteřin, je nastaven limit. Každou vteřinu je nastaven a spuštěn nový časovač a hlídá se, kolikrát už byl vyvolán. Pokud nahrávací čas přesáhl limit, nahrávání se zastaví.

4.2.6 Přenos souborů

Přenos souborů se zvukovou nahrávkou na server je uskutečněn pomocí protokolu TCP. Klient se připojuje k serveru a pomocí předem dohodnutého komunikačního protokolu zahajuje přenos. Příkaz pro server že se má začít přijímat data má tvar `__STARTFILE;název_souboru;délka_souboru__`.

Čtení ze souboru je uskutečněno také pomocí bufferů. Buffer je naplněn daty ze souboru a odesílán pomocí funkce `Send` na server. Přitom se využívá šikovnosti této funkce, která nastavuje ukazatel v rámci bufferu podle toho, jaké množství dat se podařilo odeslat. Na straně klienta se tedy plní jeden konec TCP a čeká se, než se uvolní místo pro další data (to se uvolní tím, že server data na druhé straně přijme). Po úspěšném odeslání souboru odešle klient zprávu o dokončení odesílání `__ENDFILE__` a potom už očekává odpověď serveru s výsledky.

4.2.7 Otáčení displeje

Otáčení displeje je zrealizováno přetížením metody `OnDraw`, ta se volá při požadavku na vykreslení obrazovky. A právě této vlastnosti, kdy se při otočení telefonu znovu vyžádá překreslení celé obrazovky, je využito pro přetransformování UI na šířku/výšku. Pomocí `GetClientAreaBounds` lze snadno zjistit rozměry obrazovky a metodou `GetOrientationStatus` zjistíme, v jaké poloze se telefon nachází. A jak je uvedeno v ukázce kódu 4.5, můžeme rozdělit funkci na dvě části, jednu pro *landscape* mód (na šířku) a jednu pro *portrait* mód (klasicky na výšku).

```
if (status == ORIENTATION_STATUS_PORTRAIT || status
    == ORIENTATION_STATUS_PORTRAIT_REVERSE) {
    pCanvas->DrawBitmap(Point(0, 0), *__pBackground);
}
else if (status == ORIENTATION_STATUS_LANDSCAPE || status
    == ORIENTATION_STATUS_LANDSCAPE_REVERSE) {
    pCanvas->DrawBitmap(Point(0, 0), *__pBackground_horiz);
}
```

Kód 4.5 Ukázka implementace rotace displeje

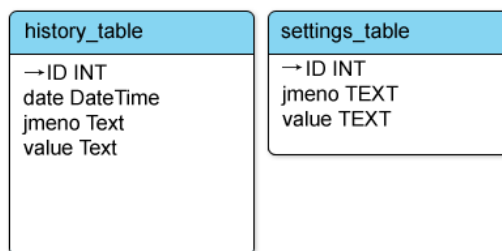
4.2.8 Uchování dat

Operační systém Bada nabízí dvě zcela odlišné možnosti, jak bezpečně uložit informace i při vypnutí aplikaci. Lze tedy uložit nastavení, stav aplikace, nebo libovolný typ informací, aplikaci vypnout a data si znovu načíst.

Databáze

Jako první z variant pro uchování dat aplikací pro operační systém bada, je použití třídy `Osp::Io::Database`. Jedná se o klasickou databázi, která se ovládá pomocí SQL příkazů. Samotná databáze je uložena v souboru přímo u aplikace a s telefonem není nijak spojena. Není tedy možné nešetrným zacházením nijak ohrozit chod samotného telefonu.

Tuto variantu jsem si zvolil pro uložení dat z historie hledaných příkazů a nastavení programu zejména kvůli možnosti s daty dále pracovat. Například data z historie potřebují řadit podle data jejich vložení atd. Pro historii je určena tabulka *history_table* a jak je uvedeno na obrázku 4.6, každá její položka má své ID, časovou značku, jméno a hodnotu. Jméno odpovídá textu, který je zobrazen v historii dotazů u každé položky jako hledaný výraz. Zatímco položka hodnota slouží pro uložení hledaného výrazu v syntaxi určené pro vyhledávače. Pro uložení IP adresy a portu serveru je určena tabulka *settings_table*.



Obr. 4.6 Tabulky databáze

Registr aplikace

Další možnost uložení dat je použití registrů aplikace. Její funkci bych přirovnal ke Cookies v jazyce HTML. Umožňuje tak rychlé uložení jednoduchých informací (typu název-hodnota), ale oproti databázi neumožňuje data jakkoli třídit a filtrovat. Bada počet hodnot v registru neomezuje, což by se dle mého názoru mělo ze shora omezit. Nešikovný programátor by mohl lehce zahltit paměť telefonu.

4.3 Distribuce aplikace

Pro distribuci aplikací pro operační systém Bada slouží jedno centrální místo - SamsungApps⁷ (dále SA), odkud ostatní uživatelé můžou nakupovat a stahovat všechny aplikace. SA je přístupný buď prostřednictvím zabudované aplikace přímo v telefonu, nebo přes desktop aplikaci Samsung KIES. Tento obchod zatím (ke dni 20. 4. 2011) v České republice nabízí pouze možnost stahovat aplikace, které jsou zdarma. Nicméně možnost nakupování placených aplikací by se měla v ČR, podle oficiálních zpráv, objevit na přelomu dubna/května.

Každá aplikace je před uvedením do SA nezávisle otestována a uživatelé tak mají zajištěnou určitou kvalitativní úroveň a nemusí se bát, že by aplikace telefon nějak poškodila, nebo odesílala jejich osobní informace bez jejich vědomí. Proces vývoje aplikace je zjednodušeně uveden na obrázku Obr. 4.7.



Obr. 4.7 Proces vývoje aplikací pro systém Bada. Převzato z [3].

⁷ <http://www.samsungapps.com>

5 Server

Server byl vyvíjen zprvu pro OS Windows, později jsem však přepsal zdrojové kódy pro Linux. Důvodem byla hlavně snazší práce s regulárními výrazy a celkově lepší zázemí pro vývoj. Jako vývojové prostředí mi posloužil nástroj Codeblocks⁸, se kterým pracuji prakticky celé studium na fakultě.

5.1 Princip činnosti

Hlavním úkolem serveru je zpracování audio nahrávky a získání těch co možná nejpravděpodobnějších výsledků. Na obrázku 5.1 můžete vidět, jak server pracuje. Pokud to vezmeme chronologicky, server začíná tím, že vytvoří socket a poté už na předem nastaveném portu čeká na připojení klientů. Pokud se někdo pokusí připojit, server tento požadavek přijme a vytvoří nový proces, který bude obsluhovat pouze jednoho klienta. Vytváření nových procesů s každým příchozím požadavkem dosáhneme konkurentnosti serveru, čili možností zpracovávat více požadavků naráz. Pokud server kdykoli během komunikace s klientem obdrží neočekávaná data, komunikaci ukončí. Tím se předejde možnému pokusu připojení nechtěných zařízení na server a jeho následnému poškození nebo zastavení.

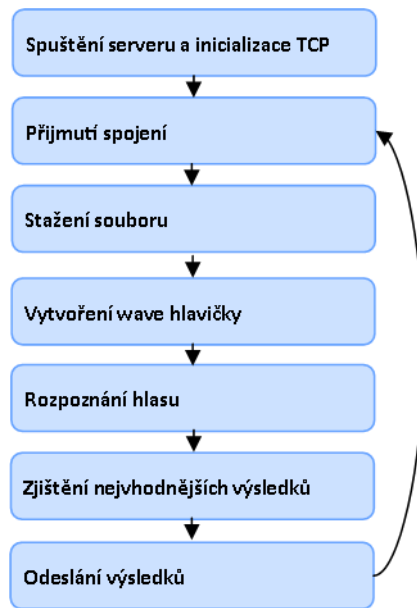
Pokud je vše v pořádku, z klienta se přenese zvukový soubor na server. Ten pak server předá rozpoznávacímu a počká, než se data zpracují. Server čeká na existenci *log* souboru ve výstupním adresáři. Tento soubor vzniká jako záznam informací (čas provedení operace, její délka atd.) o provedeném přepisu a vzniká vždy po jeho ukončení. Poté co rozpoznávač vytvoří soubor s výsledky, přesune je do výstupního adresáře, kde už server soubory očekává. Výsledky mají formát *confusion network*, nicméně existují i jiné možnosti výstupů rozpoznávače, ty jsou později popsány v kapitole 5.3.1.

Server tedy má výsledky rozpoznávače a dalším milníkem je vlastní ohodnocení jednotlivých slov a vytvoření vět na základě těchto výpočtů. Celkové ohodnocení slova se skládá ze dvou položek:

1. Ohodnocení rozpoznávačem (viz kapitola 5.3.2)
2. Ohodnocení našeptávačem (viz kapitola 5.4.3)

Ze všech slov se poté vytvoří jejich permutace a vzniknou různě ohodnocené věty. Tento postup je popsán v kapitole 5.5.2.

⁸ <http://www.codeblocks.org/>



Obr. 5.1 Časový diagram běhu serveru

5.2 Zpracování zvuku

Jelikož server obdrží audio nahrávku ve formátu bez-hlavičkového RAW PCM WAVE, musí tuto hlavičku (viz tabulka 5.1) sám vytvořit na základě předem známých hodnot o kvalitě zvukové stopy (8000 Hz, 16bit, mono) a informace o délce souboru.

Offset	Velikost	Popis	Hodnota
0x00	4	Chunk ID	RIFF
0x04	4	Chunk data size	(velikost souboru) - 8
0x08	4	Typ RIFF	WAVE
0x10	4	Chunk ID	fmt
0x14	4	Chunk data size	16 + bity navíc pro formát
0x18	2	Kód komprese	1-65,536
0x1a	2	Počet kanálů	1-65,536
0x1c	4	Vzorkovací frekvence	1-0xFFFFFFFF
0x20	4	Průměrný počet byte/s	1-0xFFFFFFFF
0x24	2	Velikost bloku vzorku	1-65,536
0x26	2	Počet bitů na vzorek	2-65,536
0x28	2	Další bity pro formát	0-65,536

Tabulka 5.1 Tvar hlavičky wave souboru

Pro tyto účely byla vytvořena speciální datová struktura, která uchovává data přesně dle daných velikostí jednotlivých prvků. Pro každý audio soubor je struktura znovu naplněna tak, aby odpovídala

parametrům daného souboru a jeho délce. Struktura je po jejím naplnění zapsána do souboru před samotná zvuková data a poté již lze tento soubor přehrát v obyčejném přehrávači, nebo právě zpracovat rozpoznávačem.

5.3 Rozpoznání hlasu

Pro účely přepisu mluvené řeči na text byl využit rozpoznávač hlasu vyvíjený skupinou Speech@FIT⁹. Tímto děkuji Ing. Petr Schwarzovi Ph.D. za poskytnutí licence k tomuto rozpoznávači.

5.3.1 Výstupy rozpoznávače

Technologie přepisu řeči stále není na takové úrovni, aby rozpoznávala slova se 100% úspěšností. Proto rozpoznávač poskytuje několik možných druhů výstupu, které lze dále zpracovat a případně poskytnout jiným technologiím ke zpřesnění výsledků. Což pro tento projekt znamená vyšší ohodnocení podmnožiny slov, používané pro vyhledávání na internetu (viz kapitola 5.4).

Rozpoznávač poskytuje tyto typy výstupu:

- Nejlepší přepis
- Laticci
- Confusion network

5.3.1.1 Nejlepší přepis

Jak již název této podkapitoly napovídá, jde o spojení slov, které rozpoznávač ohodnotil jako nejvíce pravděpodobné. Výstupem této metody jsou pak oddělené věty rozpoznávané řeči, složené pouze z nejvhodnějších slov. Nejedná se tedy o množinu možných výsledků, ale o jeden konkrétní přepis. Konkrétní výstup této metody je zobrazen v tabulce 5.2 (jedná se o přepis zvukové nahrávky, která je součástí demo balíčku rozpoznávače skupiny Speech@FIT).

<s> ŽE MI SI MYSLEL ŽE NEDOSTUDUJE SKONČILO TO VŠAK DOCELA DOBRĚ </s>
<s> DOST MOŽNÁ ŽE NĚMEC SE NECHAL PROSTĚ SKAPCE </s>
<s> DO PRAHY PŘILETĚL PRAVIDELNOU LINKOU Z PAŘÍŽE </s>

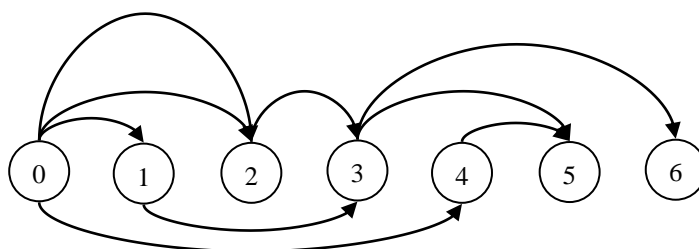
Tabulka 5.2 Výstup rozpoznávače ve formátu "nejlepší přepis"

Metoda nejlepšího přepisu byla pro tento projekt nevhodná, protože nenabízela alternativní možnosti přepisu.

⁹ <http://speech.fit.vutbr.cz/>

5.3.1.2 Laticci

Lattice je acyklický orientovaný graf. Rozpoznávač ho využívá k rozdělení časového bloku na N menších úseků, mezi které rozprostře možné výsledky přepisu. Jak lze vidět na obrázku 5.2, cesty nemusí jít z uzlu na uzel, ale můžou je libovolně přeskakovat. Každá cesta pak reprezentuje jedno slovo (nebo může být i prázdná - ticho), ohodnocené pravděpodobností z rozpoznávače. Výslednou větu pak získáme tak, že projdeme orientovaný graf po jeho hranách. Projítím všech dostupných cest, dostaneme množinu vět, kde součet pravděpodobností jednotlivých cest tvoří výsledné ohodnocené věty.



Obr. 5.2 Lattice

Pro tuto práci nabízí lattice zbytečně mnoho možnosti, jejichž zpracování by zabralo při výpočtu hodně času navíc. Výstup rozpoznávače je v textové formě, jeho formát je uveden v tabulce 5.3, kde:

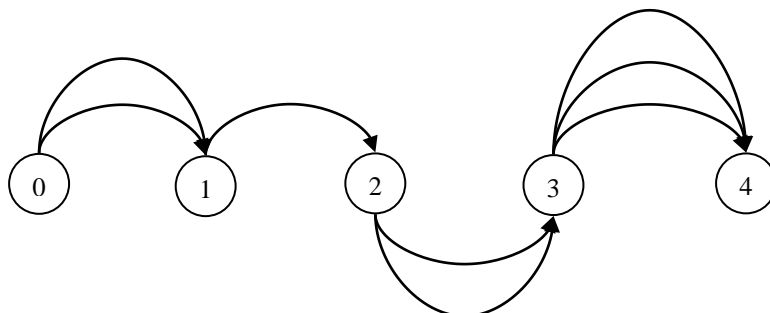
- SN - počáteční uzel slova (start node)
- EN - koncový uzel (end node)
- ST – počáteční čas slova (start time)
- ET – koncový čas (end time)
- W – hypotéza slova
- P – posteriorní pravděpodobnost slova
- I - příspěvek věrohodnosti slova z jazykového modelu
- L - příspěvek věrohodnosti slova z akustického modelu
- A – věrohodnost slova

SN	EN	ST	ET	W	P	I	L	A
2	11	0,98	1,22	ŽA	2,46953e-112	3221,804688	17,624889	-2655,433105
2	8	0,98	1,19	JÁ	4,59395e-108	2768,865967	-4,273459	2616,388672
2	16	0,98	1,20	JO	4,10315e-133	2859,857910	-3,645589	-2726,844727

Tabulka 5.3 Část výstupu rozpoznávače ve formátu lattice

5.3.1.3 Confusion network

Confusion network (na obrázku 5.3) vychází ze zjednodušeného modelu lattice. V tomto případě však vedou hrany grafu mezi právě dvěma sousedními uzly, kde uzel představuje jeden okamžik v časovém bloku, který se odvíjí od délky řečeného slova. Každá hrana grafu reprezentuje právě jedno slovo (i prázdné slovo), které bylo vysloveno v čase určeném uzly, které hrana spojuje. Takovýchto hran může být mezi dvěma sousedními uzly neomezený počet, pro každé slovo však jedna.



Obr. 5.3 Confusion network

Kombinace výsledných vět pak dostaneme permutací všech množin slov mezi jednotlivými uzly, kde ta nejpravděpodobnější vznikne průchodem po hranách s nejvyšším ohodnocením.

Výstup rozpoznávače ve formátu confusion network je taktéž v textové formě, její formát je pak uveden v tabulce 5.4, kde:

- T – číslo uzlu
- ST – počáteční čas
- ET – koncový čas
- W – slovo
- P - pravděpodobnost

T	ST	ET	W	P
4	1,29	1,37	MI	2, 22632e-029
4	1,29	1,37	MY	1,22008e-029
5	1,37	1,48	SI	1,00012
5	1,37	1,48	SE	8,5712e-028
6	1,48	1,49	MYSLEL	1,00049

Tabulka 5.4 Část výstupu rozpoznávače ve formátu confusion network

5.3.2 Ohodnocení rozpoznávačem

Pro tento projekt jsem zvolil výstup confusion network. Z rozpoznávače dostaneme seznam slov (jeden řádek – jedno slovo) ve formátu uvedeném v tabulce 5.4. Ten je třeba zpracovat pro jednotlivé informace, které jsou odděleny mezerou. Každý řádek je zpracován a jeho hodnoty (kromě počátečního a koncového času, které nejsou potřeba) jsou uloženy do struktury, které jsou dále ukládány do pole.

Jelikož jsou u slov uvedeny posteriorní pravděpodobnosti (pravděpodobnost daného slova, která se vztahuje ke všem ostatním slovům ve stejném čase) a můžou mít velikou ostrost, bylo třeba dynamiku zmenšit. Toho lze dosáhnout tak, že se pravděpodobnost převede do logaritmické oblasti, vynásobí konstantou menší než jedna, převede se nazpět a renormuje tak, aby všechny pravděpodobnosti v daném čase sumovaly do jedné. Přepočet je proveden pomocí následující rovnice:

$$p_i' = \frac{\exp(k \cdot \log p_i)}{\sum \exp(k \cdot \log p_j)} ; k < 1$$

Projdeme tedy celý seznam a pro každý uzel (záznamy se stejným číslem) přepočteme pravděpodobnost jednotlivých hypotéz. Převod čerpá z *readme* souboru, který je přiložen k rozpoznávači.

5.4 Našeptávač

Našeptávač naleznete ve většině dnešních nejrozšířenějších vyhledávačů na internetu. Jedná se o pomocnou funkci, která vám v průběhu zadávání vámi hledaného výrazu „našeptává“ různé možnosti dle znaků, které jste již napsali. Jednotlivé možnosti jsou většinou seřazeny pomocí různě složitých algoritmů, které berou v potaz vše co je možné. Od počtu vyhledání za měsíc, přes efektivní hodnotu (číslo značící úspěšnost kdy si uživatel skutečně vybere nějaký odkaz) až po aktuální oblíbenost. Tyto algoritmy si většina prohlížečů drží jako tajemství a není se čemu divit. Stejně jako algoritmy počítající co nejrelevantnější výsledky hledání, tak i samotný našeptávač v jisté míře přispívá ke kvalitě daného vyhledávače.

5.4.1 Google Suggest

Jedním z prvních průkopníků „našeptávání“ byla experimentální služba Google Suggest a nyní ji často nevědomky využíváme při používání webového vyhledávače Googlu, nebo při vyhledávání emailových adres ve vašem Gmailu.

Google Suggest není tak jednoduchý jak by se na první pohled mohlo zdát, nejde tedy jen o pouhé doporučování výsledků dle počtu jejich hledání. Suggest spojuje několik různých faktorů

(kompletní výčet Google samozřejmě nezveřejnil) a jejich sumarizací dává relevantnější výsledky. Na toto téma by se dala napsat rozsáhlá diplomová práce, avšak bez konkrétních podkladů od Googlu by to pořád byly jen odhady.

Existují dvě možnosti jak si funkci Google Suggest vyzkoušet v praxi:

- využít přímo jejich vyhledávač¹⁰, jak lze vidět na obrázku 5.4. Jelikož dnes vyhledávač Google poskytuje možnost dynamického vyhledávání (výsledky se aktualizují průběžně s psaným textem), je nutné tuto funkci napřed vypnout.



Obr. 5.4 Google Suggest ve vyhledávači Google

- položení dotazu na Google Suggest API prostřednictvím URL adresy prohlížeče. Tuto možnost, i když je veřejnosti přístupná, není dovoleno využívat pro komerční účely a jedná se pouze o neoficiální přístup k této službě.

Dotaz je ve tvaru „`google.com/complete/search?output=toolbar&q=bakalá`“, kde slovo „bakalá“ může být nahrazeno libovolným hledaným výrazem. Na první pohled po zadání dotazu to vypadá, jakoby se nic nestalo, je totiž nutné zobrazit si zdrojový kód stránky, protože výsledek nemá formu HTML stránky, ale XML. Jak lze vidět na výsledku pro dotaz „bakalá“, uvedeném v kódu 5.1, dostaneme neseřazený seznam možných výsledků s číselným ohodnocením. Toto ohodnocení je cílem mnoha spekulací, z čehož ta nejpravděpodobnější varianta je, že se jedná o počet výskytů hledání tohoto výrazu za jednotku času.

¹⁰ <http://www.google.com>

```
<?xml version="1.0"?>
<toplevel>
  <CompleteSuggestion>
    <suggestion data="bakalářská práce"/>
    <num_queries int="255000"/>
  </CompleteSuggestion>
  <CompleteSuggestion>
    <suggestion data="bakaláři"/>
    <num_queries int="61300"/>
  </CompleteSuggestion>
  ...
  ...
</toplevel>
```

Kód 5.1 Ukázka výsledku od Google Suggest API

5.4.2 Nejčastěji hledané výrazy

Jelikož by bylo časově náročné odesílat stále požadavky na Google Suggest, rozhodl jsem pro tento projekt využít nějakou lokální databázi hledaných výraz. Dostal jsem se k databázi našeptávače od nejmenovaného tuzemského vyhledávače. Databáze obsahuje výčet přes 14,000 nejčastěji hledaných výrazů a jejich pozici v našeptávači, počet výskytů za měsíc a za den.

Získanou databázi bylo třeba projít filtrem a odstranit výrazy, které by nebylo možné získat z rozpoznávače. Jednalo se zejména o slovní spojení, která obsahovala speciální znaky a čísla. Pro zvýšení efektivnosti vyhledávání v tomto seznamu jsem předpokládal, že čím vyšší počet výskytu hledání, tím častěji se bude dané slovo v seznamu vyhledávat. Seznam jsem tedy seřadil sestupně od nejčastěji hledaných.

Ve výsledku zůstalo 7483 položek, které odpovídají požadavkům. Některé z nich jsou tvořeny směsicí písmen, která netvoří skutečné slovo, k jejich vyfiltrování by však bylo potřeba celý seznam projít manuálně, nebo jednotlivá slova srovnávat s nějakým slovníkem.

Seznam ani zdaleka nedosahuje rozměrů skutečných databází vyhledávačů a proto se do budoucna nabízí možnost spolupráce s některým z tuzemských vyhledávačů. Ti by poskytli kompletní databázi všech hledaných výrazů a snížil by se tak výskyt případů, kdy by se slovo v seznamu vůbec nebylo.

5.4.3 Ohodnocení našeptávačem

Ohodnocení dat z našeptávače probíhá pro každé slovo zvlášť. Každé slovo se server pokusí vyhledat v seznamu a najít mu odpovídající hodnotu. Pokud server slovo najde, ohodnotí ho jednoduchou rovnicí, která je následující:

$$p = \frac{\text{počet hledání za měsíc hledaného slova}}{\text{počet hledání za měsíc nejčastěji hledaného slova}}$$

Tuto pravděpodobnost, stejně jako u ohodnocování rozpoznávačem, přepočítáme tak, aby všechny pravděpodobnosti stejného uzlu sumovaly do jedné.

Při vyhledávání je nyní seznam procházen kontinuálně řádek po řádku a čas potřebný pro nalezení posledního (nejméně často hledaného) výrazu se pohybuje kolem 0,083ms. To je pro účely této práce více než dostačující a je nutno také podotknout, že tento čas byl dosažen ve virtualizovaném prostředí a na případném serveru by byl čas daleko nižší. Pokud by však byla použita databáze přímo od některého z vyhledávačů a aplikace by byla používána ve velkém měřítku, takovýto přístup by nebyl vhodný. Při 10000 požadavcích za minutu by ve virtualizovaném prostředí bylo potřeba 830 vteřin strojového času k vyřízení požadavku, to je v průměru 0,77 vteřinový nárůst vyřizovací doby (téměř 1000% oproti původní hodnotě).

5.5 Výsledné ohodnocení a tvorba vět

Pomocí postupů z předchozích kapitol jsme tedy získali seznam slov s jejich časovou značkou (číslem uzlu) a pravděpodobností z rozpoznávače a našeptávače. Dalším úkolem je sumarizace těchto hodnot a získání množiny nejlépe ohodnocených vět.

5.5.1 Sumarizace pravděpodobností

Aby bylo možné tyto pravděpodobnosti nějak spojit, bylo třeba určit jejich váhy. Při testování rozpoznávače s jednoduchými dotazy jsem dosahoval velmi dobrých výsledků. Poměr ohodnocení rozpoznávače : našeptávače jsem tedy stanovil na 70 : 30. Při testování s nižším poměrem rozpoznávače se stávalo, že přednost dostávala slova, která s větou nesouvisela, avšak byla dobře ohodnocena našeptávačem. Obě pravděpodobnosti jsem vynásobil daným poměrem a získal tak pravděpodobnost konečnou.

5.5.2 Spojení slov do vět

Ke složení vět jsem vytvořil funkci, která pomocí rekurze vytvoří permutace všech možných výsledků. Její princip je takový, že každé první slovo všech uzlů a u posledního uzlu všechny. Poté se

vrátí k předchozímu uzlu. Tam vypíše aktuální slovo tolikrát, kolik slov je v posledním uzlu. Poté vezme další slovo a znovu se vrátí k poslednímu uzlu a znovu vypíše všechny jeho slova. Funkce funguje zjednodušeně následovně:

1. Vezmeme první slovo prvního uzlu.
2. Procházíme další slova, dokud nenarazíme na další uzel.
3. Pokud funkce dosáhla konce souboru, končí.
4. Funkce zavolá rekurzivně sama sebe, s informacemi o dalším uzlu. Očekává návrat hodnoty, kolikrát proběhl zápis slov v rekurzivním volání.
5. Funkce vypíše aktuální slovo, přejde na slovo následující a pokračuje bodem 4.

S každým zapsáním slova do příslušného řádku funkce zároveň inkrementuje jeho hodnotu. Ve výsledku dostaneme neseřazený seznam vět ohodnocených součtem pravděpodobností slov, ze kterých je složena.

Abychom získali nejlépe ohodnocené věty, je potřeba tento seznam seřadit. K tomu je použit řadící algoritmus *selection sort*, který je popsán v [1].

6 Testování

Pro testování jsem zvolil 10 náhodně vybraných slovních spojení, které by potenciální uživatel aplikace mohl použít. Ty jsem pak jednu podruhé předával nejprve rozpoznávači (kapitola 6.1) a shromažďoval výsledky do tabulky. Stejný postup jsem opakoval s mou aplikací (kapitola 6.2).

6.1 Samotný rozpoznávač hlasu

Pokud se podíváte na tabulku s výsledky (tabulka 6.1), najdete pár chyb:

- u pokusu č. 3, chybí předložka „v“.
- u pokusu č. 5 dal rozpoznávač přednost slovu „nabídla“ před „pravidla“.
- chybu pokusu č. 6 přisuzuju slovníku rozpoznávače, kde slovo „bary“ nejspíš chybí.
- pokus č. 8 vznikl nejspíš také absencí slova „hokeji“ ve slovníku.
- u č. 9 rozpoznávač vynechal slovo „lístky“.

Jak je vidět, samotný rozpoznávač rozpoznává velmi dobře, nicméně stále není dokonalý a chyby v přepisu se najdou.

Číslo pokusu	Mluvené slovo	Přepis rozpoznávače
1	Automobilový klub Praha	AUTOMOBILOVÝ KLUB PRAHA
2	Mobilní telefony do tisíce korun	MOBILNÍ TELEFONY DO TISÍCIKORUN
3	Tenisové kurty v okolí Brna	TENISOVÉ KURTY OKOLÍ BRNA
4	Benzínová stanice	BENZÍNOVÁ STANICE
5	Pravidla českého pravopisu	NABÍDLA ČESKÉHO PRAVOPIS
6	Bary a restaurace	BARRY RESTAURACE
7	Recept na dobrou večeři	RECEPT NA DOBROU VEČEŘI
8	Program mistrovství světa v hokeji	PROGRAM MISTROVSTVÍ SVĚTA HOLKY
9	Lístky do Národního divadla	DO NÁRODNÍHO DIVADLA
10	Taxislužba nonstop	TAXISLUŽBA NONSTOP

Tabulka 6.1 Výsledky rozpoznávače

6.2 Aplikace

Stejně jsem postupoval se samotnou aplikací a do výsledků jsem zařadil pouze první tři možnosti., výsledky jsou uvedeny v tabulce 6.2. Můžete si všimnout zejména pokusu č. 3 a 5, kde ke kompenzaci chyby z vyhledávače pomohl právě našeptávač.

Číslo pokusu	Mluvené slovo	Výsledek aplikace
1	Automobilový klub Praha	AUTOMOBILOVÝ KLUB PRAHA AUTOMOBILOVÝ PRAHA AUTOMOBILOVÝ KLUB
2	Mobilní telefony do tisíce korun	MOBILNÍ TELEFONY DO TISÍCIKORUN MOBILNÍ TELEFONY DO TISÍCE MOBILNÍ TELEFONY DVA TISÍCE
3	Tenisové kurty v okolí Brna	TENISOVÉ KURTY V OKOLÍ BRNA TENISOVÉ KURTY V OKOLÍ TENISOVÉ KURTY OKOLÍ BRNA
4	Benzínová stanice	BENZÍNOVÁ STANICE BENZÍNOVÁ STANICE
5	Pravidla českého pravopisu	PRAVIDLA ČESKÉHO PRAVOPIS NABÍDLA ČESKÉHO PRAVOPIS ČESKÉHO PRAVOPIS
6	Bary a restaurace	BARRY A RESTAURACE BARRY A RESTAURACI BARRY RESTAURACE
7	Recept na dobrou večeři	RECEPT NA DOBROU VEČEŘI NECHCE NA DOBROU VEČEŘI RECEPT NA DOBROU
8	Program mistrovství světa v hokeji	PROGRAM MISTROVSTVÍ SVĚTA HOLKY PROGRAM MISTROVSTVÍ SVĚTA MISTROVSTVÍ SVĚTA HOLKY
9	Lístky do Národního divadla	LÍSTKY DO NÁRODNÍHO DIVADLA LÍSTKY DO NÁRODNÍHO LÍSTKY NÁRODNÍHO DIVADLA
10	Taxislužba nonstop	TAXISLUŽBA NONSTOP

Tabulka 6.2 Výsledky aplikace

6.3 Zhodnocení testování

Testování dokázalo, že pomocí našeptávače opravdu lze v jisté míře kompenzovat chybu při automatizovaném přepisu hlasu na text. Samozřejmě tohle platí jen v případě vyhledávání na internetu, kdy jsou používány určité množiny slov specifické právě pro internet. Nelze tak využívat této funkce pro odstranění chyb při přepisu nahrávek z konferencí, nebo jinak specifických záznamů.

Samotná aplikace běží na mobilním telefonu velice svižně a telefon nijak nebrzdí. Největší prodleva při používání aplikace vzniká právě u rozpoznávače, když server čeká na výsledek přepisu. Nicméně Ing. Petr Schwarz Ph.D. mě ujistil, že na urychlení přepisu usilovně pracují, čímž je zaručeno další zrychlení celé aplikace do budoucna.

7 Závěr

Cílem této práce bylo vytvořit aplikaci pro mobilní telefon, pomocí které by uživatel mohl jednoduše vyhledávat na internetu svým hlasem. Na začátku bylo nutné, abych se seznámil s funkcí rozpoznávače hlasu a jeho výstupy. Stejně tak jsem zjišťoval, jak pracuje Google Suggest (i když nakonec bylo využito dat z jiného našeptávače) a jak by se dal využít pro tento projekt. Po získání přehledu o těchto aplikacích jsem konzultoval s vedoucím mé práce další postup a získal tak ucelenou představu o aplikaci.

Má práce pokračovala návrhem uživatelského rozhraní, abych získal představu o vzhledu aplikace, a teprve poté jsem začal pracovat na samotné aplikaci. Jelikož je projekt cílen na používání na dotykových mobilních telefonech, bylo třeba dbát zejména na jednoduchost ovládání pomocí prstů rukou. Během práce na tomto projektu, jsem vývoj často konzultoval s Ing. Petrem Schwarzem Ph.D., který mi poskytl několik důležitých rad při vývoji.

Velké množství času jsem věnoval studiu jazyka C++ a jeho použitím na platformě Bada. Mezi hlavní problémy, se kterými jsem se potýkal, byl přenos souborů na server, nahrávání audia v telefonu a již zmíněná nedostupnost materiálů pro vývoj aplikací na operační systém Bada. Mnohokrát se stávalo, že aplikace fungovala v simulátoru, ale přímo v telefonu pak vykazovala chybu.

Při práci na projektu jsem navštěvoval předmět „Seminář Java“, který i když nijak s touto prací přímo nesouvisí, mi poskytl základy důležité pro vývoj objektově orientovaných aplikací.

Výsledkem mé práce je funkční aplikace, která spojuje rozpoznávání hlasu, našeptávač internetového vyhledávače a dotykové prostředí systému Bada.

7.1 Možná rozšíření do budoucna

Protože mě vývoj aplikací pro mobilní telefony zaujmul, chtěl bych se mu věnovat i v příštích letech a dále rozvíjet a zdokonalovat tuto aplikaci. Už během práce na tomto projektu mě napadali další možná rozšíření, mezi která například patří:

- Použití kodeku Speex (nebo jiného) pro zmenšení objemu dat přenášených mezi aplikací a serverem. Během vývoje jsem projevil snahu tento kodek do aplikace aplikovat, avšak neúspěšně. Jako základ tohoto rozšíření mi může sloužit knihovna Speex, která se mi podařila pro Badu přeložit.
- Přidání dalších funkcí spojených s vyhledáváním na internetu (nejen hlasovému) a vytvořit tak aplikaci sloužící jako centrum pro vyhledávání.
- Rozšíření kompatibility pro ostatní telefony systému Bada

- Přejít aplikace na systém Bada 2.0, jehož spuštění Samsung plánuje na 3. čtvrtletí roku 2011
- Využití obsáhlejší databáze pro našeptávač

7.2 Podobné projekty

Jediná existující alternativa (kterou jsem našel) k tomuto projektu je aplikace Voice Search od firmy Google Inc., která je integrovaná do operačního systému Android a ke stažení pro systém iOS.

Literatura

- [1] ASHOK, Kamthane. *Programming and Data Structures*. Singapore : Pearson Education, 2004. 579 s. Dostupné z WWW: <<http://books.google.cz/books?id=6vNIE8WNOjQC>>. ISBN 9788131724224.
- [2] DAVIS, Stephen. *C++ For Dummies*. 6th Edition. New York : John Wiley & Sons, 2009. 432 s. ISBN 0470317264.
- [3] MORRIS, Ben, et al. *Introduction to bada : A Developer's Guide*. First edition. Chichester : John Wiley & Sons, 2010. 504 s. ISBN 9780470974018.
- [4] *Bada Developers* [online]. 2011 [cit. 2011-05-8]. Bada documentation. Dostupné z WWW: <<http://developer.bada.com/library/help>>.
- [5] *Eclipse - The Eclipse Foundation open source community website* [online]. 2011 [cit. 2011-05-7]. About the Eclipse Foundation. Dostupné z WWW: <http://www.eclipse.org/org/>
- [6] Samsung Wave S8500. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 14 February 2010, last modified on 10 May 2011 [cit. 2011-05-14]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Samsung_Wave_S8500>

Příloha A

Obsah DVD

- Zdrojové kódy serveru: složka `src/server`
- Zdrojové kódy aplikace: složka `src/voiceSearch`
- Programová dokumentace: složka `doc`
- Elektronická podoba této práce: složka `text`