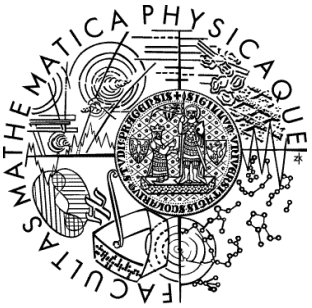
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Jakub Jelínek

Domácí gastronomický poradce

Katedra softwarového inženýrství

Vedoucí bakalářské práce: doc. RNDr. Tomáš Skopal, Ph.D.

Studijní program: Informatika, programování

2009

Na tomto místě bych chtěl poděkovat panu docentu Tomáši Skopalovi za odborné vedení a cenné rady při práci. Také bych chtěl poděkovat za zapůjčená skripta.

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne 4. srpna 2009

Jakub Jelínek

Obsah

Název práce: Domácí gastronomický poradce
Autor: Jakub Jelínek
Katedra (ústav): Katedra softwarového inženýrství
Vedoucí bakalářské práce: doc. RNDr. Tomáš Skopal, Ph.D.
e-mail vedoucího: Tomas.Skopal@mff.cuni.cz

Abstrakt: Předložená práce se zabývá sběrem receptů z Internetu, jejich uložením do databáze a následným vyhledáváním receptů podle uživatelských kritérií. Recepty mohou být získávány na základě šablon popsaných regulárními výrazy, nebo pomocí mechanizmu automatického rozpoznání. Pro získaný recept jsou rozpoznány jednotlivé ingredience, jejich množství a činnosti, kterých je potřeba pro jeho dokončení. Pokud není u receptu uvedena doba přípravy, je automaticky odhadnuta. Dále je vytvořen mechanizmus, který zabraňuje opakovanému uložení stejného receptu do databáze. Pro vyhledávání receptů jsou navržena pravidla pro párování ingrediencí, která podporují významovou příbuznost (rohlík – pečivo), významovou odlišnost pro podobné ingredience (pečivo – prášek do pečiva) a jednoduché zohlednění množství zadaného vzhledem k potřebnému. Na základě tohoto párování jsou vytvořena tři uživatelsky volitelná kritéria pro určení relevance. Součástí práce je uživatelská a programátorská dokumentace.

Klíčová slova: recept, text mining, vyhledávání informací, regulární výrazy

Title: Home gastronomic advisor
Author: Jakub Jelínek
Department: Department of Software Engineering
Supervisor: doc. RNDr. Tomáš Skopal, Ph.D.
Supervisor's e-mail address: Tomas.Skopal@mff.cuni.cz

Abstract: The propounded work deals with collecting recipes from the Internet, their storing into database and consequential retrieval of these recipes according to user's criteria. Recipes can be obtained based on patterns described by regular expressions, or by using automatic recognition mechanism. For each obtained recipe there are identified its ingredients, their amount and activities required to complete this recipe. Preparation time is automatically estimated, if not already present. There is also mechanism that prevents same recipe to be stored repeatedly into database. For recipe retrieval, there are designed rules for ingredient matching that supports semantic affinity (doughnut – pastry), semantic divergence for similar ingredients (ice – ice cream) and simple way of taking specified amount in relation to the required into account. On the basis of this matching rules are created three user selectable criteria for determining relevance. User and programmer documentation is part of this work.

Keywords: recipe, text mining, information retrieval, regular expressions

Kapitola 1
Úvod

Jídlo je pro žaludek a žaludek pro jídlo;
Bůh však jednou učiní konec obojímu.

Bible, 1 Korintským 6:13

Jídlo patří mezi základní lidské potřeby. Pokud má člověk dostatek a netrpí hladem, má zájem na tom, aby jeho jídelníček byl pestrý. Pro některé lidi je však obtížné hledat stále nové nápady, co připravit. Proto vznikají různé kuchařky, ve kterých můžeme nalézt inspiraci.

Kuchařky „první generace“ byly papírové. K vyhledání vhodného receptu nabízely nanejvýš abecední seznam receptů, případně rozdělovaly recepty do kategorií podle typu jídla (např. těstoviny, polévky, maso...)

„Druhá generace“ kuchařek jde cestou digitalizace receptů. K dispozici jsou buď aplikace pro osobní počítače nebo webová řešení. Vyhledávání je díky síle počítačů možné provádět fulltextově ve velmi krátkém čase.

Tato práce přináší řešení, které jde ještě o krok dále. Vychází z různých situací, které mohou člověka potkat a ve kterých ocení dobrou radu. Uveďme dva příklady:

1. Student Adam pozve několik svých přátel na návštěvu. Když po nějaké době začnou mít jeho přátelé hlad, rozhodne se, že pro ně něco rychle připraví. Adam bude potřebovat najít recept, na který může použít nejlépe jenom suroviny, které má k dispozici. Navíc bude chtít přípravu stihnout co nejdříve.
2. Rodina paní Báry se stěhuje a je potřeba dojíst potraviny, které doma zůstaly. Paní Bára je ochotná nějaké suroviny přikoupit, ale hlavní je, aby se spotřebovaly ty, které má doma. Také už má sbalený mixér v krabici, takže recepty, ke kterým by jej potřebovala, ji nezajímají.

Řešení má dvě části. První část se zabývá získáváním receptů z Internetu, druhá část zajišťuje „chytré“ vyhledávání receptů s důrazem na suroviny.

K dispozici jsou dva způsoby získávání receptů – z jiných webových serverů pomocí šablon definovaných regulárními výrazy a z náhodných zdrojů. V druhém případě není možné získat recept pomocí předem definované šablony, ale je potřeba jeho umístění ve stránce a strukturu odhadnout. Tento způsob je spíše dodatkový.

Kapitola 2 analyzuje problémy řešené touto prací, představuje hlavní koncepty. Kapitola 3 obsahuje popis použití jednotlivých částí programu z hlediska uživatele. Ukazuje všechny důležité funkčnosti a možnosti. Kapitola 4 obsahuje programátorskou dokumentaci nad rámec referenční dokumentace. Popisuje důležité části programu a důležité problémy, které bylo potřeba při implementaci vyřešit. Kapitola 5 srovnává toto řešení s jinými již existujícími implementacemi. Kapitola 6 ukazuje, jak lze pomoci Adamovi a Báře při hledání vhodného receptu.

1.1 (Nejen) typografické konvence

V rámci této práce jsou pro zvýšení srozumitelnosti dodržovány zde uvedené konvence. První důležitou konvencí je zápis souborů, případně adresářů. Není-li v textu řečeno jinak, lomítko na začátku cesty k souboru odpovídá kořenovému adresáři přiloženého CD. Další konvence se týkají použitých fontů:

Kurzívou se vyznačuje zejména první výskyt nějakého pojmu v textu. Dále je kurzíva použita pro odlišení názvů ovládacích prvků.

Bezpatkovým písmem se vyznačují názvy souborů a adresářů, tříd, rozhraní a metod, speciálních hodnot nebo výrazů v textu, které je potřeba odlišovat.

`Neproporcionálním písmem` se uvádějí části textových souborů, zdrojového kódu nebo vstupy a výstupy na příkazové řádce.

1.2 Slovníček pojmů

Tento oddíl se zaměřuje na slova, která jsou v textu použita a jejichž význam se může lišit od běžného pojetí. Jedná se nejprve tři pojmy z lingvistiky – kmen, kořen a lemma. Práce používá tyto pojmy jako synonyma, která odpovídají pojmu *kmen*. Další dvě slova souvisejí s recepty – ingredience a množství.

Kmen. Ta část slova (jména nebo slovesa), která vznikne po oddělení koncovky. Kmen sestává z kořene, případně též jedné nebo více předpon či přípon. [1]

Kořen. Morfém slova, který nese základní význam. Nelze ho dále členit na menší významové jednotky. Přidáváním předpon a přípon a spojováním s jinými kořeny se vytvářejí nová slova a modifikuje původní význam. Z jednoho nebo více kořenů, případně i z jedné nebo více předpon a přípon se skládá kmen slova. [2]

Lemma. Označuje základní podobu lexému (tedy slova nebo fráze), která se uvádí jako reprezentativní ve slovnících (slovníkový tvar). [3]

Ingredience. Ingrediencí se označuje jedna přísada do receptu – např. mouka. Ve spojení s množstvím tvoří ingredience *položku receptu*. Někdy je pojem ingredience použit na místě, kde by logicky správně měla být uvedena položka receptu.

Množství. Určuje kvantitu ingredience v rámci položky receptu. Je-li množství ve tvaru číselná hodnota a jednotka (např. 100g), používá se někdy označení množství pouze pro číselnou část.

Nakonec je potřeba zmínit zkratkové slovo DOGAPO, které vzniklo ze slov „DOmácí GAstronomický POradce“, tedy z oficiálního názvu této práce.

Kapitola 2
Analýza problému

Problémy řešené v této práci se dají rozdělit do dvou oblastí. Nejprve je potřeba získávat recepty a následně mezi těmito recepty vyhledávat. Následující oddíly analyzují jednotlivé otázky z těchto oblastí a nastiňují, jakým způsobem byly řešeny.

2.1 Zdroje receptů

Recepty je obecně možno získávat různě. Někdo může něco dobrého uvařit a nám to chutná, tak si řekneme o recept. Nebo můžeme čerpat se sešitu, kam si recepty zapisovala naše babička. V obou případech bude výsledkem pouze malá sbírka receptů. Pokud bychom jich chtěli řádově více, můžeme buď od někoho jejich databázi koupit nebo si takovou databázi vytvořit automatizovaným sběrem receptů z webových stránek. To je cesta, kterou se ubírá tato práce.

Recepty lze na Internetu najít ve dvou formách. Jednak na webových portálech poskytujících recepty, jednak pomocí vyhledávačů typu google. V prvním případě se jedná o data s pravidelnou strukturou, kterou je možné popsat např. regulárním výrazem. Proto se také tento typ zdroje označuje jako regex. V druhém případě jsou nalezené recepty v předem neznámé a vždy jiné struktuře. V takovém případě není možné strukturu předem popsat, ale je potřeba ji odhadnout. Tento typ zdroje se označuje jako google-like. Za zdroj receptů se označí webový portál nebo vyhledávač, pro který se zadefinuje způsob získávání receptů.

2.1.1 Extrakce strukturovaných dat

Liu [4] ukazuje tři možnosti extrakce dat ze zdrojů s pravidelnou strukturou:

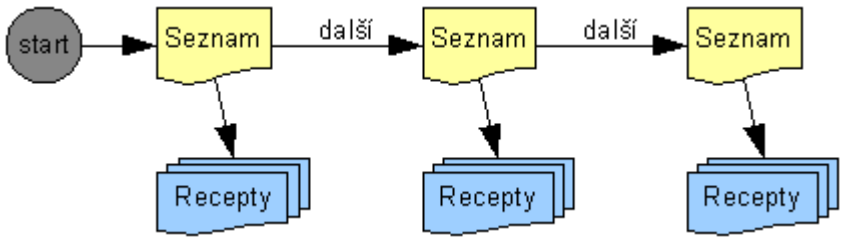
1. **Manuální přístup.** Pozorováním webové stánky a jejího zdrojového kódu nalezne lidský programátor určité vzory a pak napíše program, který extrahuje cílová data. Pro usnadnění tohoto procesu bylo vytvořeno několik jazyků a uživatelských rozhraní pro specifikaci vzorů. Tento přístup však není škálovatelný pro velké množství stránek.
2. **Indukce obalu** (*Wrapper induction*). Jde o přístup učení se s dohledem, který je poloautomatický. Práce na něm začaly kolem let 1995-1996. V tomto přístupu se odvozuje sada pravidel pro extrakci z kolekce manuálně značkovaných stránek nebo datových záznamů. Tyto pravidla jsou pak použita pro extrakci datových položek z podobně formátovaných stránek.

3. **Automatická extrakce.** Jde o přístup bez dohledu, který začal kolem roku 1998. V jedné nebo více stránkách automaticky nalézá vzory nebo gramatiky pro extrakci dat. Protože tento přístup eliminuje úsilí vynaložené na ruční značkování, může rozšířit extrakci dat na velké množství stránek.

Pro extrakci receptů ze zdrojů s pravidelnou strukturou byl pro účely této práce zvolen manuální přístup. Lze jej totiž poměrně jednoduše implementovat. Navíc nevzniká velká zátěž na lidský faktor, neboť takových zdrojů receptů není mnoho (řádově jednotky) a vytvoření potřebného regulárního výrazu není obtížné.

2.1.2 Struktura zdroje

Při pozorování různých potenciálních zdrojů receptů se ukázalo, že obecný koncept struktury je vždy stejný. V každém zdroji existují dva typy stránek – seznamy obsahující odkazy na konkrétní recepty a stránky s těmito recepty.



Obrázek 1: Obecný koncept struktury zdroje.

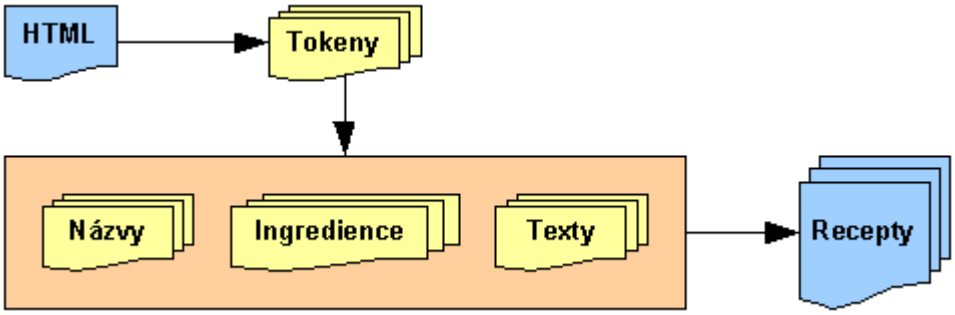
Zdroj je tedy možné popsat výchozím seznamem a popisem, jak získávat ze seznamů odkazy na recepty a další seznamy. Jde-li o zdroj s pravidelnou strukturou, je zapotřebí ještě popis uložení receptu ve stránce. Pro popis odkazů i popis uložení receptu jsou použity regulární výrazy.

2.2 Automatické rozpoznání receptu

Zdroje typu google-like z principu neobsahují informaci o uložení receptu na stránce. Je tedy zapotřebí pro danou HTML stránku určit, zda obsahuje nějaký recept. Stránka může také obsahovat více než jeden recept.

Pro tyto účely se recept chápe jako trojice název – ingredience – text. Textem se rozumí postup přípravy. Rozpoznávání probíhá podle následujícího postupu:

1. Vytvoření tzn. tokenů z HTML kódu.
2. Určení, které tokeny nebo skupiny tokenů mohou plnit funkci názvu, ingrediencí nebo textu.
3. Spojení nalezených částí tak, aby tvořily celé recepty. Kontrola receptů.



Obrázek 2: Postup rozpoznávání receptů v HTML stránce.

2.2.1 Vytvoření tokenů z HTML stránky

Text HTML stránky, který je viditelný pro uživatele, je rozdělen na části, které spolu strukturně souvisejí – např. odstavec, nadpis, řádek tabulky nebo prvek seznamu. Taková část textu je označována jako *token*. Může-li patřit text do více tokenů – např. je v buňce tabulky a dále ještě v nadpisu – přiřadí se do nejmenšího možného tokenu. Pro účely dalšího zpracování se ke každému tokenu přiřadí také jeho typ odvozený od HTML značek (PARAGRAPH, HEADLINE, LISTITEM, TABLEROW nebo OTHER).

Následující úryvek HTML kódu ilustruje možný vstup pro tvorbu tokenů. Výsledkem je pět tokenů: „Nejlepší recepty“, „Dnes má svátek Karel“, „Špagety“, „Dáme do vody“ a „a vaříme.“ První a třetí token jsou typu nadpis, ostatní typu odstavec. Je možné si všimnout, že formátovací tagy a nejsou součástí tokenu.

```
<table><tr><td>
  <h1>Nejlepší recepty</h1><p>Dnes má svátek <b>Karel</b>
</td><td>
  <h2>Špagety</h2><p>Dáme do vody <br /> a vaříme.
</table>
```

Výpis 1: Příklad HTML kódu pro vytvoření tokenů. Je možné si všimnout, že HTML kód není validní. To je pro web běžné, ovšem pro účel vytvoření tokenů to nevadí.

2.2.2 Rozpoznání názvu

Rozpoznání názvu spočívá v určení, zda nějaký token může plnit funkci názvu receptu či nikoliv. Pro název bylo stanoveno pravidlo, že musí začínat velkým písmenem (přesněji, že nesmí začínat malým písmenem, ale může číslicí nebo např. uvozovkou) a dále nesmí obsahovat některé vybrané znaky – závorky, tečku, čárku, dvojtečku, pomlčku a znak nového řádku. Toto pravidlo zajistí odfiltrování velkého množství tokenů, které funkci názvu neplní, ale vyskytují se kolem něj (např. „tiskni recept“, „recept si oblíbilo 8 návštěvníků“ nebo „Od uživatele: Lada Kovářová“). Zároveň toto pravidlo neeliminuje skutečné názvy receptů, protože ty vetšinou této podmínce vyhovují.

2.2.3 Rozpoznání ingrediencí

Tokeny popisující ingredience rozpoznáváme po skupinách, tzn. že předpokládáme, že text popisující ingredience je obsažen v několika (nebo jen jednom) po sobě jdoucích tokenech.

Pro každý token lze odhadnout, zda popisuje ingredience – použitím mechanizmu, který později rozpoznává jednotlivé části ingrediencí (tj. s přesností na množství, jednotku, název), rozdělíme daný token a zkoumáme každou ingredienci, která takto vznikla. Na tu je kladen požadavek, aby název ingredience nebyl prázdný nebo delší jako čtyři slova, také nesmí obsahovat číslo. Pokud není uvedeno množství, požaduje se dále, aby alespoň jedno slovo názvu bylo v *referenčním seznamu ingrediencí* (obsahuje přes 1000 slov, v souboru config\item-names.txt). Splní-li všechny ingredience tuto podmínku, má se za to, že token popisuje ingredience.

Za token(y) popisující ingredience se považuje:

1. maximální souvislá posloupnost tokenů typu LISTITEM nebo TABLEROW, ve které minimálně 70% tokenů považovat za popis ingrediencí podle výše uvedeného postupu, nebo
2. souvislá posloupnost tokenů typu PARAGRAPH, ve které každý token lze považovat za popis ingrediencí podle výše uvedeného postupu.

Hodnota minimálního požadovaného podílu tokenů v bodě jedna (70%) byla určena empiricky. Lze ji chápat následujícím způsobem: Nemá-li recept více než tři ingredience (3% případů), musí být rozpoznány všechny. Má-li recept čtyři až šest ingrediencí (21% případů), může zůstat jedna ingredience nerozpoznána. Má-li recept sedm až devět ingrediencí (37% případů), mohou být nerozpoznány dvě ingredience. Při deseti až třinácti ingrediencích (31% případů) se tolerují tři nerozpoznané a podobně pro více ingrediencí ve zbylých 8% případů.

Nakonec dojde ke spojení takových dvojic skupin tokenů, u kterých bylo určeno, že mohou popisovat ingredience a zároveň nejsou vzdáleny více než o jeden token (tzn. že mezi nimi je pouze jeden jiný token). Motivací pro tuto konečnou úpravu byla situace, kdy každá ingredience byla v samostatném odstavci a pokud nebyla nějaká rozpoznána (např. tymián – pokud by nebyl v referenčním seznamu ingrediencí), považovaly se ingredience před a za touto nerozpoznanou jako různé skupiny, které k sobě nepatří.

2.2.4 Rozpoznání textu (postupu přípravy)

Za text popisující postup přípravy receptu se označí token, nebo skupina po sobě jdoucích tokenů, z nichž každý splňuje následující podmínky:

1. Po odstranění bílých znaků platí, že začíná velkým písmenem a končí tečkou.

2. Je „dostatečně podobný“ *referenčnímu dokumentu* (obsahuje 1000 textů popisujících přípravu receptu, v souboru config\ref-recipe-text.txt). „Dostatečnou podobnost“ popisuje oddíl *2.3 Vektorový model a referenční dokument*.

2.2.5 Sestavení receptu a jeho kontrola

Po identifikaci možných rolí jednotlivých tokenů nebo jejich skupin přichází na řadu sestavení jednotlivých receptů. Recept vznikne tak, že se ke každému rozpoznanému textu přiřadí nejbližší (směrem nahoru) rozpoznané ingredience a dále nejbližší (opět směrem nahoru) rozpoznaný název. Nejsou-li ingredience nebo název k dispozici, recept se nesestaví.

Po tomto sestavení je potřeba provést kontrolu, zda nedošlo k situaci, kdy dvěma textům byly vybrány stejné ingredience (a tedy i stejný název). Pokud k takové kolizi dojde, je diskvalifikován recept, jehož text je uvedený níže.

2.2.6 Efektivita automatického rozpoznávání receptů

Získávání receptů ze zdrojů typu google-like s sebou nese riziko zanesení chybných dat. Pro zjištění efektivity automatického rozpoznávání receptů během extrakce receptů z těchto zdrojů byl sestaven následující test.

Vstupem tohoto testu je 100 receptů, které byly získány pomocí nějakého zdroje receptů typu regex. Během testu se na zdrojové URL těchto receptů aplikuje automatické rozpoznávání a porovnají se recepty získané oběma metodami. Pro podrobný výstup testu viz soubor \test-output\ExtractorPrecisionTest\index.htm na přiloženém CD. Výsledek testu se dá shrnout v následujících bodech:

- 33 receptů nebylo rozpoznáno vůbec.
- U rozpoznaných receptů byl správně rozpoznán
  - název v 83% případů,
  - všechny ingredience v 70% případů.
- U receptů, kde nebyly rozpoznány všechny ingredience, bylo správně rozpoznáno v průměru 69% ingrediencí.
- Text rozpoznaného receptu sice v žádném případě neodpovídal přesně původnímu receptu, většinou však sdílel podstatnou část. Chybělo například závěrečné přání „Dobrou chuť!“ nebo jiná nepodstatná část.

Test ukazuje, že automatické rozpoznávání sice v mnoha případech přináší dobrý výsledek, má však také značnou míru nepřesnosti. V praxi by bylo pravděpodobně potřeba rozpoznaná data ručně zkontrolovat, příp. porovnat se zdrojem, aby se snížilo množství chybných dat.

2.3 Vektorový model a referenční dokument

Při určování míry podobnosti nějakého textu s textem referenčním je použita reprezentace dat pomocí *vektorového modelu*, jak jej popisuje Pokorný a kol. [5].

Při popisu vektorového modelu se jednotlivým textům říká *dokumenty*. Slova nebo slovní spojení, která jsou zajímavá z hlediska podobnosti, se označují jako *termy*. Při očíslování termů $t_1..t_n$ může být každý dokument reprezentován vektorem

$$D_x = (w_{x1}, w_{x2}, ..., w_{xn}), \text{ kde } w_{xi} \in \langle 0 ; 1 \rangle$$

kde w_{xi} jsou váhy náležející termu t_i v dokumentu D_x. Váha ohodnocuje důležitost jednotlivých termů pro identifikaci dokumentu. Váha rovna nule představuje nejnižší důležitost, váha rovna jedné pak důležitost nejvyšší. V realizovaném řešení vzniká vektor vah normalizací vektoru četností jednotlivých termů v daném dokumentu.

Pro dva dokumenty D_1 a D_2 se definuje tzn. *koeficient podobnosti*. Hodnoty koeficientu podobnosti se pohybují v rozsahu od nuly do jedné. Hodnota nula představuje naprosto různé dokumenty (tj. nemají žádný společný term), hodnota jedna pak odpovídá situaci, kdy jsou vektory odpovídající oběma dokumentům totožné.

Tento koeficient lze vypočítat různě, v realizovaném řešení byl zvolen vztah označovaný jako *kosinová míra*:

$$\mathrm{Sim}(D_1, D_2) = \frac{\sum_{k=1}^{n} w_{1k} w_{2k}}{\sqrt{\sum_{k=1}^{n} w_{1k}^2} \sqrt{\sum_{k=1}^{n} w_{2k}^2}}$$

Jsou-li vektory vah *normalizované* (tzn. jejich euklidovská délka je rovna jedné), lze dělitele vynechat a psát zjednodušeně

$$\mathrm{Sim}(D_1, D_2) = \sum_{k=1}^{n} w_{1k} w_{2k}$$

Uvedený vztah je skalárním součinem vektorů D_1 a D_2.

Nakonec je potřeba stanovit způsob výběru termů. V nejjednodušším pojetí je termem každé slovo, které je v textu obsaženo. Pro zvýšení efektivity se zavádí následující dvě pravidla:

1. Všechna slova, která mají stejný kmen (zpravidla jsou jiným tvarem téhož slova), jsou považovány za stejný term.
2. Za term se nepovažují tzn. *nevýznamová slova*, tj. frekventovaná slova s malou rozlišovací schopností (spojky, částice, zájmena...) Použitý seznam nevýznamových slov je v souboru config\stoplist.txt, obsahuje 150 slov.

2.3.1 Lemmatizátor

Při volbě termů je potřeba pro každé slovo určit jeho kmen, tj. tu část slova, která vznikne po oddělení koncovky. Hledání kmene se označuje jako *lemmatizace*. Nutno podotknout, že toto označení může být poněkud zavádějící, protože výraz lemma označuje základní podobu určitého slova, nikoliv jeho kmen.

Pro lemmatizaci byl zvolen jednoduchý algoritmus, který popisuje Žemlička [6]. Tento algoritmus odtrhne nejdelší možnou koncovku z níže uvedeného seznamu tak, aby výsledné lemma bylo dlouhé alespoň tři znaky. Jde o tyto koncovky:

íma	ýma	ata	a	ová	á	ete
ěte	e	ové	é	ách	atech	ech
ích	ých	ami	emi	ěmi	ími	ými
mi	eti	ěti	ovi	i	ě	í
ám	etem	ětem	em	ém	ím	atům
ům	ým	ého	ího	o	at	ému
ímu	ou	u	ů	aty	y	ý

Tabulka 1: Koncovky odstraňované lemmatizátorem.

2.3.2 Funkce M: Alternativní přístup

Pro určení podobnosti s referenčním textem byl v rámci této práce navržen ještě jeden přístup. Vychází ze znalosti situace, do jaké bude aplikován. Na jedné straně je relativně rozsáhlý referenční soubor, na druhé straně krátký text, pro který chceme rozhodnout o podobnosti. Tento přístup právě využívá této asymetrie.

Pro každý term t z referenčního textu nechť je $p(t)$ jeho relativní četnost. Zřejmě platí

$$\sum_{t \in T} p(t) = 1, \text{ kde } T \text{ je množina všech termů}$$

Jako *skóre termu* se označí funkce $m(t)$ definovaná předpisem

$$m(t) = \frac{1}{p(t)|T|}$$

Pro porovnávaný text se pak funkce $M(t_1,\ldots,t_n)$ nazývá *skóre textu* složeného z termů t_1 až t_n a definuje se jako aritmetický průměr hodnot $m(t_i)$. Není-li pro některý term hodnota $m(t_i)$ definována, dodefinuje se dostatečně velkou konstantou (např. 10). Očekává se, že je-li text podobný textu referenčnímu, bude hodnota funkce M rovna přibližně jedné (střední hodnota funkce m je jedna a průměr zachovává střední hodnotu). Čím více se bude text odlišovat, tím více se bude projevovat vliv dodefinované konstanty a hodnota funkce M bude růst.

2.3.3 Určení limitů a porovnání metod

Obě uvedené metody je potřeba vyzkoušet na skutečných datech, aby bylo možné rozhodnout, pro jaké hodnoty koeficientu podobnosti resp. hodnoty funkce M bude ještě daný text považován za dostatečně podobný.

V následujících testech bude vždy pro obě metody použit referenční text obsahující 500 textů popisujících postup receptu. Každá metoda pak dostane padesát jiných textů popisujících postup receptu (pozitivní vzorky) a padesát zcela jiných textů (negativní vzorky), pro které určí podobnost s referenčním textem. Bude také ukázáno, jak výsledky ovlivňuje použití lemmatizátoru (stemmer) a odstínění nevýznamových slov (stoplist).

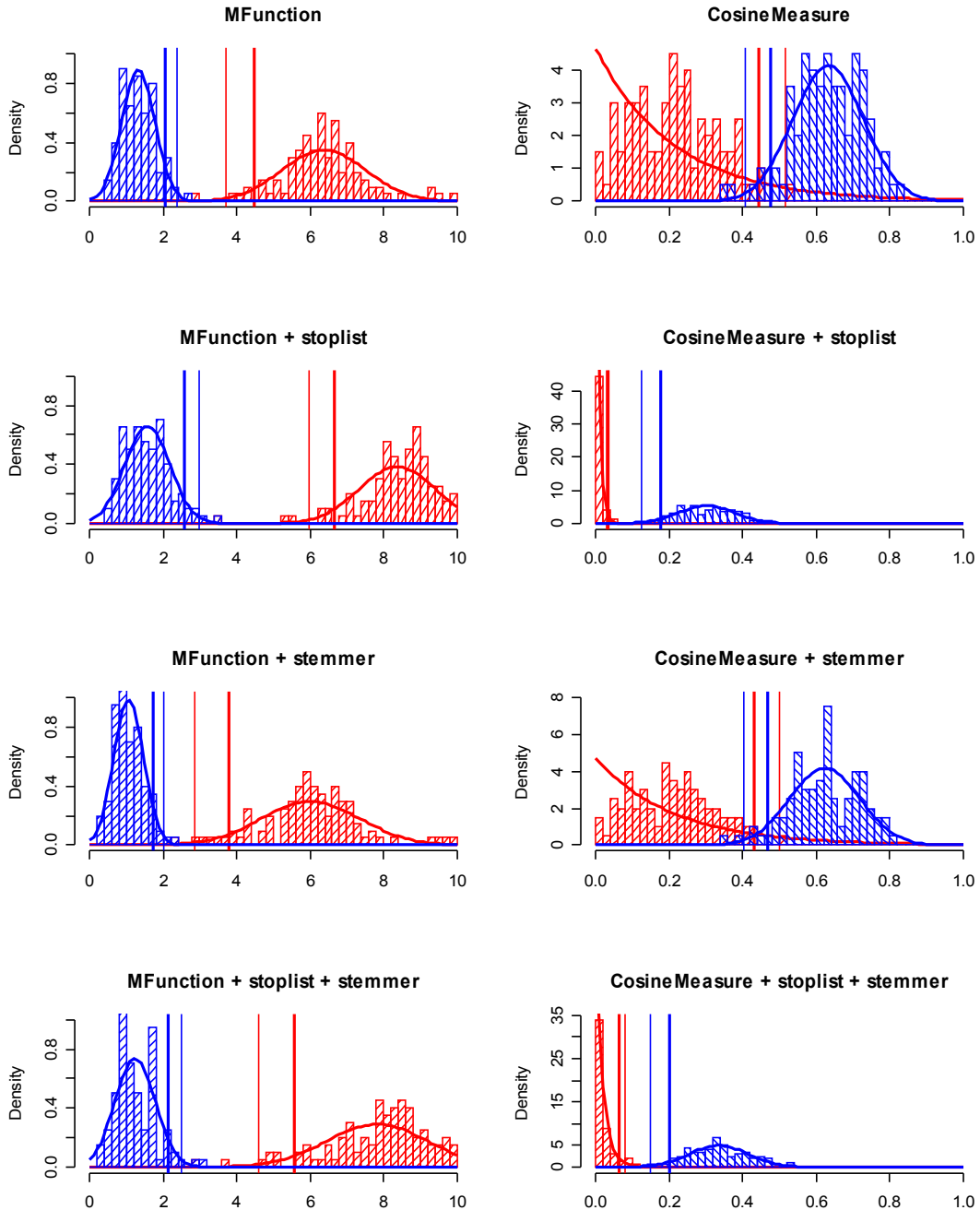
Následující obrázek ukazuje výsledky pro jednotlivé metody. Na ose x je pro funkci *M* (MFunction) vyneseno skóre jednotlivých vzorků, pro vektorový model pak hodnota koeficientu podobnosti vypočítaná pomocí kosinové míry (CosineMeasure). Histogram pozitivních vzorků je vyznačen modrou barvou, negativní vzorky jsou vyznačeny červenou barvou. Dále je do grafů pro každý histogram zakreslena normální aproximace (tj. hustota normálního rozdělení se střední hodnotou určenou jako výběrový průměr a výběrovým rozptylem). Nakonec svislé čáry v grafu reprezentují 0,95-kvantil (tlustá čára) a 0,99-kvantil (tenká čára) vzhledem k zmíněnému normálnímu rozdělení.

Z uvedených grafů vyplývá, že funkce *M* rozlišuje pozitivní a negativní vzorky již v základní variantě téměř jistě. Při použití stoplistu (bez ohledu na to, zda je ještě použita lemmatizace) je možné volit hodnotu 4 jako akceptační limit. Je-li tedy skóre textu nejvýše 4, bude považován za podobný referenčnímu textu.

V případě kosinové míry není její rozlišovací schopnost bez použití stoplistu tak přesvědčivá, jako u funkce *M*. Je-li však stoplist použit, jsou výsledky již přijatelné. Za akceptační limit (opět pro variantu jak s lemmatizací, tak i bez ní) lze stanovit jednu desetinu, tzn. že pokud je hodnota kosinové míry pro nějaký text alespoň jedna desetina, bude považován za podobný referenčnímu textu.

Na základě těchto pozorování lze říci, že použití stoplistu má obecně větší vliv na rozlišovací schopnost než lemmatizace.

Poznámka. Popsaný test je implementován ve třídě SimilarityTestSuite. Výstupem tohoto testu je skript pro statistický program R [7]. Pomocí tohoto skriptu byly vytvořeny grafy na obrázku 3.



Obrázek 3: Srovnání metod pro určování podobnosti s referenčním dokumentem.

2.4 Proces sběru

Je-li k dispozici zdroj receptů, definice jeho struktury a způsob získávání jednotlivých receptů, je možné stanovit proces, jakým budou recepty získávány. Následující diagram naznačuje čtyři fáze sběru.

Obrázek 4: Jednotlivé fáze sběru.

Sběrač receptů postupuje přímočaře. Začne zpracovávat výchozí seznam zdroje, rozpozná všechny recepty, na které tento seznam odkazuje a pokračuje zpracováním dalšího seznamu. Takto pokračuje, dokud není přerušen nebo již neexistuje další seznam. Nalezené recepty pokračují do další fáze.

Rozpoznání činností potřebných ke zdárnému dokončení receptu spočívá v hledání klíčových slov v textu popisujícím postup přípravy. Podle takto nalezených slov se receptu přiřadí, které činnosti vyžaduje.

Časový odhad má smysl v okamžiku, kdy čas potřebný k přípravě nebyl získán přímo ze zdrojové HTML stránky. Způsob, jakým se odhad provádí, je popsán dále.

Uložením do databáze končí proces sběru. Pro každý recept se před jeho uložením vypočítá tzn. *kontrolní součet*, který se uloží spolu s receptem. Pokud nastane situace, že stejný recept přijde znovu do fáze uložení, bude vypočten jeho kontrolní součet a po zjištění, že recept s takovým kontrolním součtem již v databázi existuje, nebude znovu uložen. Tímto mechanizmem je umožněno opakované spouštění sběru nad nějakým zdrojem a zároveň získávání pouze těch receptů, které ve zdroji přibyly.

2.4.1 Časový odhad

Pro účely časového odhadu bylo vybráno více než 12 000 receptů z takového zdroje, který pro každý z nich uvádí také čas potřebný k přípravě. Na tomto vzorku pak byly testovány metody odhadu. Úvodem je potřeba poznamenat, že čas k přípravě je do jisté míry subjektivní záležitost – co jednomu trvá půl hodiny, může druhému trvat třeba hodinu.

Nejjednodušší je odhad průměrem. Průměrná doba byla 46 minut. Při tomto odhadu je směrodatná odchylka ± 29,3 minut. Pro zlepšení odhadu je zapotřebí uvažovat některé parametry receptu, jako např. počet surovin. Dá se předpokládat, že čím více surovin je v receptu obsaženo, tím déle bude trvat jeho příprava. Za použití lineárního modelu matematické statistiky vznikl následující vzorec (v minutách):

$$\text{čas} = 3{,}44 * \text{počet ingrediencí} + 14{,}9$$

Směrodatná odchylka tohoto odhadu je ± 27,2 minut. Přidáním dalších vysvětlujících proměnných do lineárního modelu lze odhad ještě zpřesnit. Ukazuje se, že délka textu a časové údaje v textu mají kladnou korelaci (0.403 a 0.336) s odhadovanou dobou. Po zahrnutí těchto veličin do lineárního modelu vznikl následující vzorec:

čas = 1,90 * počet ingrediencí + 0,0305 * délka textu + 0,373 * minut v textu + 9,51

Tento odhad dosahuje směrodatné odchylky ± 25,6 minut. Tento odhad je také použit v realizovaném řešení. Odhadnutý čas je následně standardizován, tj. je nahrazen tou z hodnot 10, 15, 20, 25, 30, 45, 60, 75, 90, 120 a 150, které je nejblíže.

2.5 Vyhledávání receptů

Uživatel může přes webové rozhraní vyhledávat v receptech, které již byly uloženy do databáze. Vyhledávání probíhá na základě zadaných kritérií. Kritéria určující čas přípravy a činnosti, které je uživatel ochoten vykonat pro dokončení receptu, omezují výslednou množinu receptů. Zadané ingredience, příp. spolu s množstvím, se podílejí na stanovení relevance nalezených receptů (řazení výsledků se řídí touto prioritou). Pro vyhledávání je potřeba vyřešit dva problémy – způsob, jakým se budou porovnávat ingredience zadané uživatelem a ingredience receptu v databázi, a způsob určení relevance.

2.5.1 Párování ingrediencí

Párování ingrediencí spočívá v rozhodnutí, zda ingredience, kterou má uživatel, se dá použít na místě ingredience uvedené v receptu. Nejjednodušší kritérium pro párování je rovnost. Má-li uživatel „vejce“ a potřebuje-li „vejce“, pak tyto ingredience tvoří pár. Pro odstínění flexe se uvažují pouze kmeny slov, které se získávají dříve popsaným lemmatizátorem. Pak platí, že má-li uživatel „polohrubou mouku“ a potřebuje-li „(100g) polohrubé mouky“, pak tyto ingredience tvoří pár, protože obě mají kmen „polohrub mouk“. Odstraněním diakritiky z kmenů se lze zbavit např. rozdílu mezi „citronem“ a „citrónem“.

Další pravidlo pro párovaní říká, že jsou-li všechny kmeny jedné ingredience obsaženy v druhé, nebo naopak, pak tyto ingredience tvoří pár. Následující tabulka ukazuje možné příklady využití tohoto pravidla:

Uživatel má	kmen	V receptu je	kmen
pepř	pepr	pepř černý	pepr cern
hladká mouka	hladk mouk	mouka	mouk

Tabulka 2: Příklady párování.

Uvedené pravidlo zajišťuje, že významově si odpovídají ingredience budou spárovány. Může ovšem nastat situace, kdy díky tomuto pravidlu budou spárovány takové

ingredience, které k sobě nepatří. Příkladem je „pečivo“ s kmenem „peciv“ a „prášek do pečiva“ s kmenem „prasek do peciv“. Pro takové případy existuje mechanizmus, kterému lze říci, že dvě ingredience netvoří pár, pokud obě obsahují kmen X, ale pouze jedna z nich obsahuje kmen Y. Pro uvedený příklad odpovídá X kmenu „peciv“ a Y kmenu „prasek“. Tím jednak zamezíme párování „pečiva“ s „prášem do pečiva“, jednak zachováme možnost párovat např. „staré pečivo“ s „pečivem“.

Posledním vylepšením párování ingrediencí je mechanizmus, který umožňuje párovat ingredience s naprosto odlišnými kmeny. Lze např. definovat, že má-li uživatel „vejce“ a potřebuje-li „bílek“, budou tyto ingredience tvořit pár. Podobně např. má-li uživatel „chleba“ a potřebuje-li „pečivo“, je možné definovat, že mají vytvořit pár.

Poslední dvě pravidla se definují v databázi v tabulkách ROOT_MATCHES_EXCLUDE a ISA. Více podrobností lze nalézt oddíle *4.5.1 Datový model*.

2.5.2 Určení relevance

Způsob výpočtu relevance receptu pro zadaný dotaz určuje řazení receptů ve výsledcích vyhledávání. Existují tři možnosti určení relevance, všechny jsou založeny na párování ingrediencí:

1. Maximální „pokrytí“ receptu.
2. Maximální využití zadaných ingrediencí.
3. Nejlepší shoda zadaných ingrediencí s ingrediencemi receptu.

Jednotlivé možnosti označujeme jako kritéria K_1 až K_3. Každé kritérium může nabývat hodnot z intervalu <0;1>, kde nula označuje nejnižší relevanci a jednička relevanci nejvyšší. Jednotlivé kritéria jsou definována následujícími rovnostmi:

$$K_1 = \frac{m_H}{H} \qquad K_2 = \frac{m_N}{N} \qquad K_3 = \frac{m}{H + N - m}$$

kde m_H … počet spárovaných ingrediencí uživatele
H … počet všech ingrediencí uživatele
m_N … počet spárovaných ingrediencí receptu
N … počet všech ingrediencí receptu

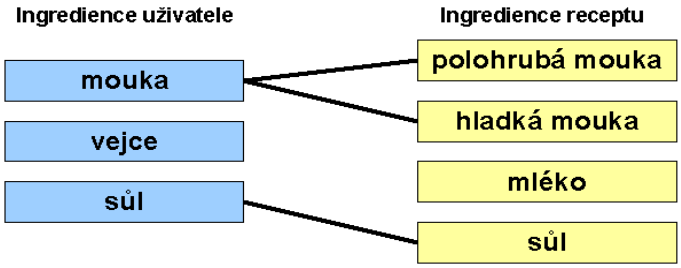
Zvláštní pozornost si zaslouží proměnná m. Kritérium K_3 je určeno podílem velikosti průniku a velikosti sjednocení množin ingrediencí uživatele a receptu. Na rozdíl od normálních množin se však může stát, že nějaká ingredience bude v průniku vícekrát. K tomu dojde v okamžiku, kdy se např. uživatelova ingredience spáruje s více než jednou ingrediencí v receptu. Pak se velikost průniku nedá určit počtem spárovaných ingrediencí uživatele nebo receptu, ale je potřeba tuto velikost zadefinovat jako průměr těchto hodnot, tedy $m = (m_H + m_N) / 2$. Kritérium K_3 pak lze zapsat takto:

$$K_3 = \frac{\frac{m_H + m_N}{2}}{H + N - \frac{m_H + m_N}{2}} = \frac{m_H + m_N}{2\mathrm{H} + 2\mathrm{N} - m_H - m_N}$$

Příklad. Pro ilustraci předpokládejme, že uživatel má ingredience naznačené na obrázku níže a chceme určit relevanci postupně podle všech kritérií pro naznačený recept. Čáry mezi ingrediencemi znázorňují, které ingredience tvoří páry. Je vidět, že „mouka“ je obsažena ve dvou párech. Pro tuto situaci nabývají jednotlivé proměnné a kritéria následujících hodnot:

$$m_H = 2,\ H = 3,\ m_N = 3,\ N = 4,\ m = 2{,}5$$

$$K_1 = 2/3,\ K_2 = 3/4,\ K_3 = 5/9$$



Obrázek 5: Příklad na kritéria výpočtu relevance.

Pozorování. Je-li $m_H = m_N$, pak je kritérium K_3 menší rovno kritériím K_1 a K_2. V takovém případě platí také $m = m_H = m_N$. Vzhledem k symetrii stačí ukázat pro jedno z nich, např. tedy K_1:

$$K_3 = \frac{m_H}{H + N - m_H} \leq \frac{m_H}{H} = K_1$$

Po vydělení nerovnice m_H a převrácení (obě strany nerovnice jsou kladné, nerovnost se tedy musí otočit) dostaneme

$$H + N - m_H \geq H \text{ , po úpravě } N \geq m_H = m_N$$

což je jistě pravda, protože počet spárovaných ingrediencí receptu nemůže být vyšší než počet všech ingrediencí v receptu. Při vynechání předpokladu rovnosti $m_H = m_N$ tvrzení již neplatí. Protipříklad:

$$m_H = 1,\ H = 4,\ m_N = 3,\ N = 3$$
$$K_3 = 2/5,\ K_1 = 1/4$$

2.5.3 Zohlednění množství ingredience při vyhledávání

Uživatel může při vyhledávání specifikovat množství, ve kterém má ingredienci k dispozici, a to v hmotnostních (g, dkg, kg) a objemových (ml, cl, dl, dcl, l) jednotkách. V souvislosti s množstvím vzniká ještě jedno pravidlo pro párování ingrediencí: Pokud je množství uvedeno jak uživatelem, tak v receptu, pár se vytvoří, pouze pokud má uživatel alespoň takové množství, jaké vyžaduje recept.

Toto jednoduché pravidlo s sebou přináší jistou nepřesnost. Např. pokud uživatel má 200g cukru a v receptu je uvedeno 250g, je to jako by uživatel neměl žádný cukr. Naopak pokud má uživatel stejných 200g cukru, a v receptu je dvakrát 150g cukru, spáruje se uživatelův cukr s oběma, i když tolik cukru nemá.

Při snaze vymyslet sofistikovanější způsob jak zohlednit množství se velmi rychle dojde k nejednoznačným situacím v souvislosti s určováním relevance, zejména proto, že ingredience se mohou vyskytovat ve více párech a že ne každá ingredience má uvedeno množství.

Kapitola 3
Uživatelská dokumentace

Cílem této kapitoly je seznámit uživatele s postupem instalace aplikace a ovládáním jejích jednotlivých částí, tzn. webového rozhraní a sběrače receptů. Webové rozhraní je navrženo tak, aby jej uživatel mohl používat intuitivně bez nutnosti předchozí přípravy. Na opačné straně stojí definice zdroje, která je popsána v posledním oddíle, což je činnost, kterou může dělat jen zkušený uživatel (nejlépe také programátor).

3.1 Instalace a konfigurace

Aplikace sestává ze tří částí, z nichž každou je potřeba nainstalovat. Nejprve se instaluje databáze, následně webové rozhraní a sběrač receptů. Konfigurace znamená zejména nastavení databázového připojení v obou později jmenovaných částech.

3.1.1 Instalace databáze

Databáze se instaluje na databázový server MSSQL 2005. Pro instalaci a také pro následný chod aplikace je potřeba mít na daném serveru uživatele a také databázové schéma. Instalační skript předpokládá, že schéma, do kterého se bude instalovat, se jmenuje DOGAPO. Pokud tomu tak není, je potřeba před provedením kroku tři změnit příkaz „use DOGAPO" na první řádce skriptu run-install.sql na odpovídající schéma. Vlastní instalace pak znamená:

1. Nastavit připojovací údaje k databázi v souboru run-install.bat. Jedná se o jméno databázového serveru, uživatelské jméno a heslo.
2. Změnit pracovní adresář na \install\db\create.
3. Spustit run-install.bat.

Pokud dojde během kroku tři k chybě (např. nedostatečná práva), je možné databázi vyčistit skriptem \install\db\clean\clean.bat, opravit příčinu chyby a krok tři zopakovat.

3.1.2 Instalace webového rozhraní

Pro webové rozhraní je potřeba server umožňující běh ASP.NET 2.0 aplikace. Instalace spočívá v nakopírování souborů ze složky \install\web do složky wwwroot na daném serveru. Následuje nastavení připojení k databázi (connectionString) v souborech \wwwroot\Web.Config a \wwwroot\Bin\config\config.xml.

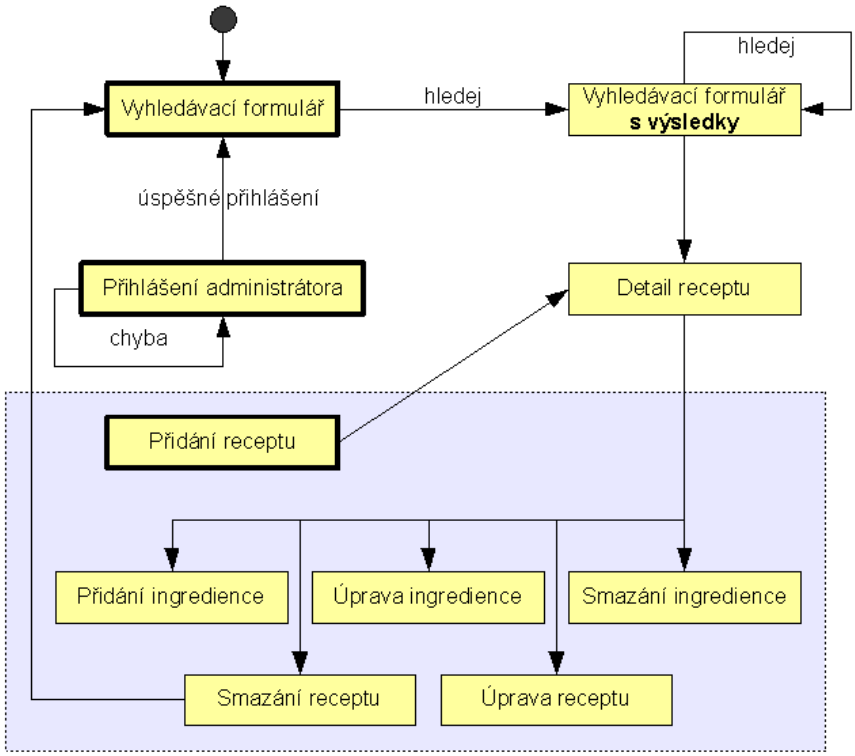
3.1.3 Instalace sběrače receptů

Sběrač receptů je desktopová aplikace, pro svůj běh vyžaduje .NET Framework 2.0. Instalace se spouští souborem \install\collector\setup.exe. Testováno na Windows XP, Windows Vista a Windows 7 v 32bitových variantách, Vista i v 64bitové variantě.

Po nainstalování se nastaví databázové připojení v souboru config\config.xml (cesta je relativní vzhledem k exe souboru nainstalovaného programu).

3.2 Ovládání webového rozhraní

Webové rozhraní je určeno jednak pro uživatele, kteří chtějí vyhledávat recepty podle zadaných kritérií, a jednak pro administrátora, který kromě vyhledávání může modifikovat obsah databáze – změnou stávajících receptů, jejich přidáváním nebo mazáním. Následující obrázek zobrazuje strukturu webového rozhraní:



Obrázek 6: Obrazovky webového rozhraní a přechody mezi nimi. Silnějším rámečkem jsou označeny obrazovky, na které je možné přistoupit z libovolné obrazovky za použití menu. V modrošedém obdélníku jsou obrazovky, pro které je potřeba předchozí úspěšné přihlášení administrátora.

3.2.1 Vyhledání receptu

Vyhledávání receptů je jediná činnost běžného návštěvníka. Pro vyhledání receptu není nutné přihlášení. Formulář pro zadání kritérií hledání se zobrazí jako první obrazovka při přístupu na webové rozhraní. Uživatel může vyplnit následující pole:

Seznam ingrediencí. Jednotlivé ingredience se oddělují *čárkou*. Před ingrediencí je možné uvést také množství. V tom případě není nutné zadávat ingredience v prvním pádě, je tedy možné např. psát „100g hovězího masa“. Jsou rozpoznávány jednotky hmotnosti (g, dkg, kg) a objemu (ml, cl, dl, dcl, l), číslo je možné zadat pouze celé. V případě, že je množství ingredience uvedené v seznamu ingrediencí menší než množství uvedené v nějakém receptu, chápe se to stejně, jako by taková ingredience vůbec nebyla k dispozici.

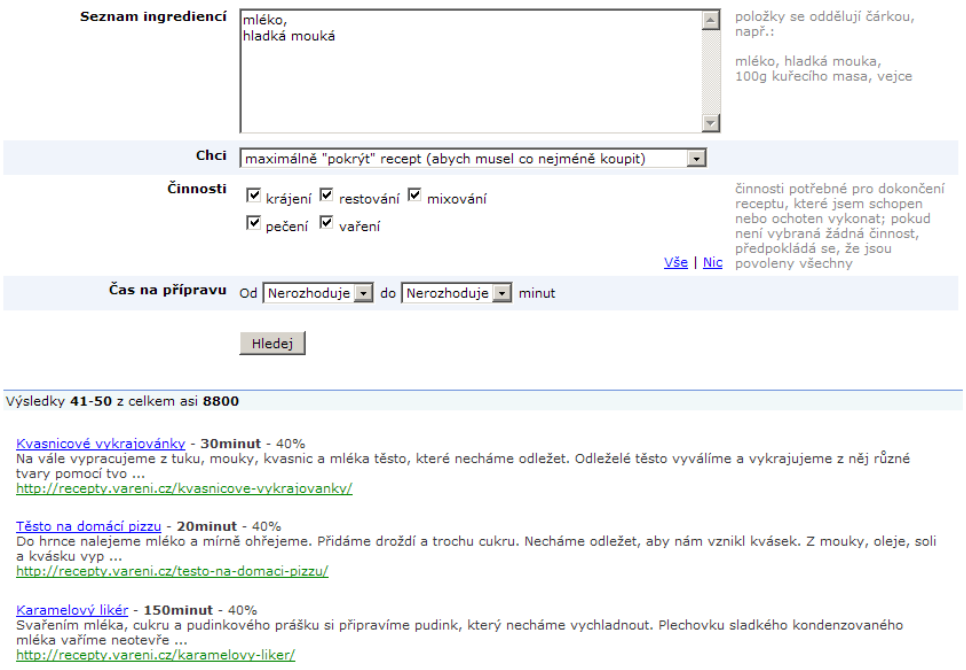
Chci. Určení způsobu výpočtu relevance a z toho plynoucího řazení receptů. Jsou k dispozici tři možnosti:

1. Nejlepší shoda zadaných ingrediencí s ingrediencemi receptu.
2. Maximální využití zadaných ingrediencí (aby mi zbylo co nejméně).
3. Maximální „pokrytí“ receptu (abych musel co nejméně nakoupit).

Činnosti. Činnosti potřebné pro dokončení receptu, které je uživatel schopen nebo ochoten vykonat; pokud není vybraná žádná činnost, předpokládá se, že jsou povoleny všechny.

Čas na přípravu. Je možné nastavit horní nebo dolní limit, případně oba.

Po zvolení všech kritérií se odešle formulář kliknutím na tlačítko *Hledej*. Pod vyhledávacím formulářem se zobrazí stránkované výsledky vyhledávání. V záhlaví výsledků je možné vidět celkový počet nalezených receptů. Detail receptu se zobrazí po kliknutí na název receptu (modrou barvou). Navigace na zdrojovou stránku daného receptu je možná po kliknutí na odkaz (zelenou barvou).



Obrázek 7: Ukázka vyhledávacího formuláře s výsledky.

3.2.2 Přihlášení administrátora

Všechny činnosti, které mění obsah webu, jsou podmíněny přihlášením administrátora. Přihlašovací formulář je dostupný přes položku menu *Administrace*. Po zadání uživatelského jména a hesla se formulář odešle kliknutím na tlačítko *Přihlásit*. Pokud zadané údaje odpovídají, je uživatel přesměrován na vyhledávací formulář, přihlášení je indikováno nápisem „Administrátor přihlášen“ v záhlaví stránky. Vedle tohoto nápisu je také odkaz *Odhlásit*, pomocí kterého uživatel může přejít zpět do režimu bez administrátorských práv.

Výchozí uživatelské jméno je admin, výchozí heslo password. Obě tyto hodnoty je možné změnit v souboru wwwroot\Bin\config\config.xml.

3.2.3 Úprava ingrediencí

Přihlášený administrátor může ingredience přidávat, měnit a mazat. Všechny tyto úpravy se vyvolávají z obrazovky detailu receptu. Pro přidání a úpravu platí, že ingredience musí být zadána, vyplnění množství a jednotky jsou nepovinná.

Přidat. Kliknutím na odkaz *Přidat ingredienci* pod receptem.

Upravit. Kliknutím na ikonu  na stejném řádku, jako upravovaná ingredience.

Smazat. Kliknutím na ikonu  na stejném řádku, jako mazaná ingredience.

3.2.4 Úprava receptů

Přihlášený administrátor může kromě ingrediencí upravovat vlastnosti vztahující se k celému receptu, recepty přidávat a mazat.

Přidat. Kliknutím na položku menu *Přidej nový recept*. Ingredience se zadávají včetně případného množství oddělené čárkou. Po přidání je zobrazen detail nově přidaného receptu. Příklad ingrediencí: „100g hladké mouky, 1/2 kostky másla“. Při uložení receptu systém rozpozná uvedené množství i jednotky.

Upravit. Kliknutím na odkaz *Upravit* pod receptem. Lze upravit název a text receptu, čas na přípravu a činnosti nutné k dokončení receptu.

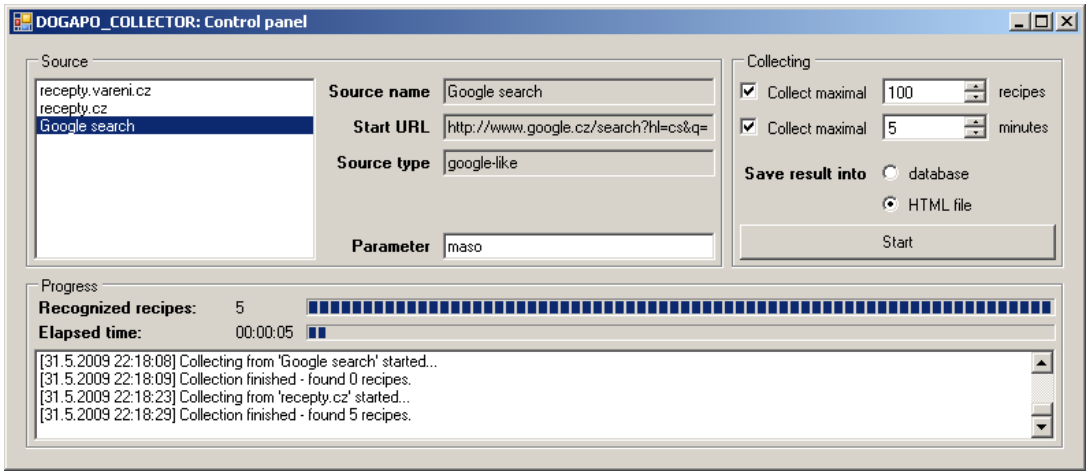
Smazat. Kliknutím na odkaz *Smazat recept* pod receptem.

3.3 Ovládání sběrače receptů

Sběrač receptů je aplikace, která má na starost získávat data o receptech a ukládat je do databáze. Jsou k dispozici dvě rozhraní – grafické uživatelské rozhraní (GUI) určené pro uživatelskou interakci a rozhraní na příkazové řádce (CLI) určené především pro automatizované spouštění. Činnost a principy sběrače popíšeme na GUI, následně ukážeme, v čem se liší použití CLI.

3.3.1 Grafické uživatelské rozhraní

Grafické uživatelské rozhraní sestává z jediného okna, ve kterém uživatel může zvolit parametry sběru, spustit jej a sledovat jeho průběh. Rozhraní je lokalizováno do anglického jazyka. Jednotlivé části budou popsány v následujících odstavcích.



Obrázek 8: Grafické uživatelské rozhraní sběrače receptů.

Source. Zdroj, odkud se budou získávat recepty. Jednotlivé zdroje jsou uvedeny v seznamu. Po kliknutí se zobrazí jejich detail v zašedlých textových polích napravo od seznamu zdrojů. Má-li zdroj parametr, je zde také textové pole pro jeho zadání. Návod, jak zdroje definovat, je pro svou náročnost uveden v samostatné kapitole.

Collecting. Nastavení parametrů sběru. Je možné zvolit maximální počet získaných receptů a maximální čas sběru. Dále je možné zvolit, zda se nalezené výsledky mají ukládat do databáze nebo do souboru (má smysl např. při testování definice nového zdroje). Sběr se spouští tlačítkem *Start*.

Progress. Zobrazuje se počet již nalezených receptů a uplynulý čas. Pokud byly stanoveny limity, zobrazuje se také poměrná část vzhledem k těmto limitům. Ukončení sběru je signalizováno výpisem do textové oblasti v dolní části okna. Předčasné ukončení sběru je možné provést kliknutím na tlačítko *Stop*, které během sběru nahradí tlačítko *Start*.

3.3.2 Rozhraní na příkazové řádce

Rozhraní na příkazové řádce (CLI) sdílí s GUI stejný exe soubor. Pokud je tomuto exe souboru při startu poskytnut alespoň jeden parametr, bude se chovat jako CLI.

Jediný vždy povinný parametr je source-name. Pokud zvolený zdroj vyžaduje parametr, je povinný také parametr param. Volitelně je možné zadat také limity pro počet receptů a čas v minutách parametry max-recipes a max-minutes. Nalezené recepty se vždy ukládají do databáze.

Příklady, jak je možné uvedené parametry použít, ukazuje následující obrázek. Po spuštění již není možné do chodu programu zasahovat.

```
DOGAPO_COLLECTOR.exe --source-name="recepty.cz"
DOGAPO_COLLECTOR.exe --source-name="recepty.cz" --max-minutes=60
DOGAPO_COLLECTOR.exe --source-name="Google search" -param=maso
```

Výpis 2: Příklady parametrů při spuštění CLI.

3.3.3 Definice zdroje

Definice zdroje receptů je bezpochyby nejnáročnější činnost, kterou uživatel může vykonat. Zabývat se touto činností je potřeba buď v okamžiku, kdy se nějaký zdroj změní a současná definice mu již neodpovídá, nebo při přidávání nového zdroje. Předpokládá se, že uživatel je schopen pracovat s regulárními výrazy.

Zdroje jsou definovány v souboru config\sources.xml. Každý zdroj je popsán elementem source. Jednotlivé atributy jsou popsány pod příkladem, pro hlubší pochopení jejich významu viz kapitolu Analýza problému.

```
<source
    name  ="recepty.cz"
    type  ="regex"
    url   ="http://recepty.cz/recept_all.asp"
    listRE="recepty.cz/list"
    nextRE="recepty.cz/next"
    pageRE="recepty.cz/page" />
```

Výpis 3: Příklad popisu zdroje v souboru config\sources.xml.

Atribut name obsahuje unikátní název zdroje, který je důležitý pro spouštění z příkazové řádky. Atribut type určuje typ zdroje (regex nebo google-like). Atribut url udává výchozí URL zdroje. Zdroj má parametr, právě když toto url obsahuje {0} jako podřetězec. Tento podřetězec bude při vyhledání nahrazen aktuální hodnotou parametru. Parametr lze použít např. pro hledané slovo v případě google-like zdroje.

Poslední tři atributy jsou odkazy na soubory s regulárními výrazy. Cesty k regulárním výrazům jsou relativní vzhledem ke složce config\RE. Za uvedený název souboru se připojuje koncovka .re. Cesta tedy nakonec vypadá např. takto: config\RE\recepty.cz\list.re.

Při zpracování regulárních výrazů jsou použity přepínače RegexOptions.Singleline a RegexOptions.IgnorePatternWhitespace. Praktickým důsledkem druhého přepínače je potřeba escapovat mezeru v případě, že má být součástí regulárního výrazu.

Jednotlivé regulární výrazy obsahují tzn. named capturing groups – pojmenované skupiny. Na tyto skupiny bude v textu odkazováno zkratkou CG. Následuje popis jednotlivých regulárních výrazů uvedených v definici zdroje:

- **listRE** – položka seznamu, CG Link obsahující url, do výsledku se zahrnou všechny výskyty. Např. <a\ class='hyperlink3'\ href='(?<Link>[^']*)'.
- **nextRE** – odkaz na další seznam receptů, CG NextPage, do výsledku se zahrne pouze první výskyt.
- **pageRE** – konkrétní recept na dané stránce. Uvádí se pouze pro pro typ zdroje regex. Povinné CG jsou Name (název receptu), HowTo (postup přípravy), dále popis ingrediencí buď jako jeden kus textu (CG Ingredients) nebo jednotlivě jako dvojice CG IngredientName (název ingredience) a IngredientAmount (množství včetně jednotky). Volitelně může být také CG Time (čas v minutách). Do výsledku se zahrne pouze první výskyt.

Nejobtížnější část těchto regulárních výrazů je popis ingrediencí jako dvojic název – množství. Tento požadavek nejlépe vynikne na následujícím příkladě (jde o zjednodušený fragment ze skutečného regulárního výrazu):

```
#Ingredience
(
  ( #Ingredience item
    <tr>\s*
      <td>
        (?<IngredientName>[^<]*)
      </td>\s*
      <td\ class="tdl1">
        (?<IngredientAmount>[^<]*)
      </td>\s*
    </tr>\s*
  )
)* #next ingredience
```

Výpis 4: Regulární výraz obsahující CG IngredientName a IngredientAmount. Zvýrazněná hvězdička na posledním řádku zajišťuje, že do výsledku budou zahrnuty všechny ingredience.

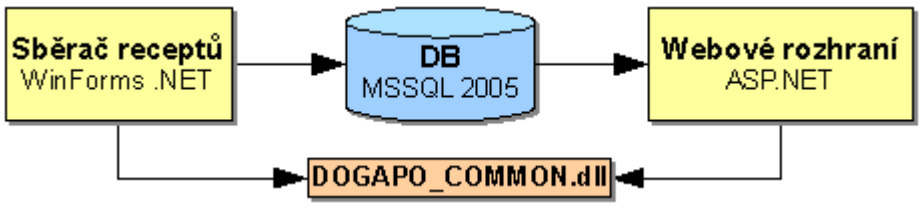
Na uvedeném příkladě je také možné si všimnout, že regulární výrazy mohou obsahovat komentáře uvozené znakem #, platné do konce řádku.

Kapitola 4
Programátorská dokumentace

Programátorská dokumentace, obsažená v této kapitole, nejprve nastíní základní strukturu řešení, použité technologie a nástroje. Pak se bude detailně zabývat jednotlivými částmi aplikace. Bude také ukázáno, jaké problémy během implementace vznikly a jak byly vyřešeny. Referenční dokumentace vygenerovaná na základě zdrojových kódů může sloužit jako zdroj dodatečných informací, je k dispozici na přiloženém CD ve složce \ref_doc.

4.1 Struktura programu

Struktura programu na nejvyšší úrovni je naznačena na následujícím schématu. Sběrač receptů představuje část programu, která dodává data do databáze. V ní pak vyhledává recepty webové rozhraní. Sběrač receptů a webové rozhraní spolu sdílí některé funkcionality, které jsou sdruženy do knihovny DOGAPO_COMMON.dll.



Obrázek 9: Struktura programu.

Tato struktura se odráží v rozdělení do projektů a jmenných prostorů:

- projekt DOGAPO_COLLECTOR, jmenný prostor Dogapo.Collector
- projekt DOGAPO_WEB, jmenný prostor Dogapo.Web
- projekt DOGAPO_COMMON, jmenný prostor Dogapo.Common
- projekt DOGAPO_DB, obsahuje pouze SQL

4.2 Použité technologie a nástroje

Technologie, na kterých je řešení postaveno, jsou naznačeny v uvedeném schématu. Sběrač receptů je koncipován jako desktopová aplikace na technologii WinForms, vyžadující .NET Framework 2. Webové rozhraní je postaveno na technologii ASP.NET 2.0. Roli databáze zastává Microsoft SQL Server 2005.

Během vývoje byly použity zejména následující vývojové a podpůrné nástroje:

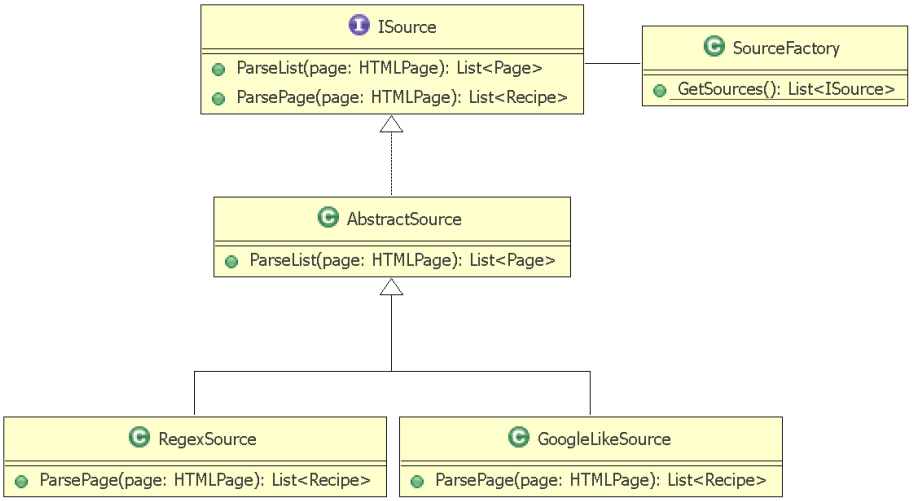
- vývojové prostředí Microsoft Visual Studio 2005
- databázový klient SQL Server Management Studio Express
- profiler nprof v0.9-alpha [8]
- nástroj pro generování dokumentace NDoc2 Alpha [9]

4.3 DOGAPO_COLLECTOR

Projekt DOGAPO_COLLECTOR řeší implementaci sběrače receptů, má závislost na projektu DOGAPO_COMMON. Po kompilaci vzniká soubor DOGAPO_COLLECTOR.exe.

4.3.1 Zdroje receptů

Zdroje receptů jsou popisovány třídami, které splňují rozhraní ISource. Toto rozhraní má kromě jiných metod také metody ParseList a ParsePage. Metoda ParseList dostane HTML stránku, která obsahuje odkazy na stránky s konkrétními recepty, a vrátí seznam těchto stránek. Tato metoda funguje stejně bez ohledu na typ zdroje, je proto implementována již v abstraktní třídě AbstractSource. Naproti tomu metoda ParsePage, která získává konkrétní recepty z HTML stránky, je závislá na typu zdroje, proto má každý typ svou implementaci.



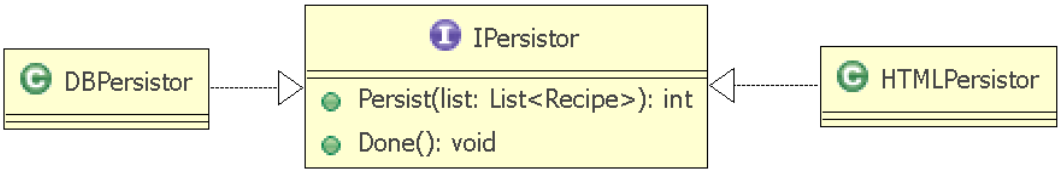
Obrázek 10: Hierarchie tříd popisujících zdroje receptů.

Zdroje, které bude mít sběrač k dispozici, jsou uloženy v souboru config/sources.xml. Načtení tohoto souboru do objektové reprezentace zajišťuje třída SourceFactory a její statická metoda GetSources.

4.3.2 Persistory

Persistor je komponenta, která má za úkol ukládání receptů. Její kontrakt definuje rozhraní IPersistor. Použití persistoru má tři kroky – vytvoření nové instance, pře-

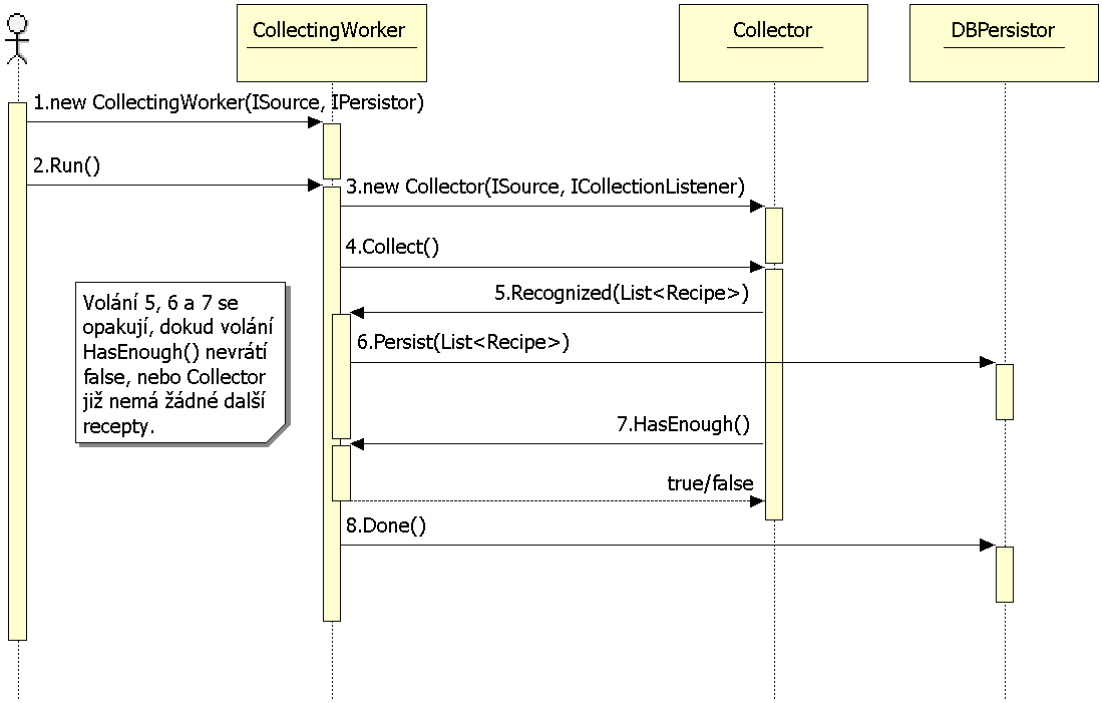
dávání receptů metodou Persist a ukončení činnosti metodou Done. Po volání této metody se již instance nesmí dále používat. Recepty je možné ukládat do souboru ve formátu HTML (HTMLPersistor), nebo do SQL databáze (DBPersistor).



Obrázek 11: Persistory.

4.3.3 Třída CollectingWorker

Proces sběru implementuje třída CollectingWorker. Při vytváření instance se předá konstruktoru zdroj receptů (ISource), persistor (IPersistor) a limity (maximální čas v minutách a maximální počet nalezených receptů). Sběr provádí metoda Run. Ta nejprve vytvoří novou instanci třídy Collector, které předá zdroj receptů a sebe v roli ICollectionListener. Vždy, když Collector nalezne recept (jeden nebo více), zavolá metodu Recognized na svém posluchači, kterým je CollectingWorker. Pro získané recepty se dopočítá informace o aktivitách a příp. potřebném čase. Takto upravené recepty se předají persistoru metodou Persist. Collector se během své práce dotazuje svého posluchače, zda už „má dost“ metodou HasEnough. Po dokončení sběru je persistor informován o ukončení voláním metody Done. Popsaný proces zachycuje následující obrázek.



Obrázek 12: Sekvenční diagram třídy CollectingWorker.

4.3.4 Kontrolní součet receptu

Při ukládání receptu do databáze používá DBPersistor kontrolní součet receptu pro zjištění, zda je tento recept v databázi již obsažen.

Původní návrh výpočtu kontrolního součtu přetížením metody GetHashCode skrývá riziko nesprávného chování, pokud použijeme na řetězce jejich původní implementaci. Podle dokumentace [10] je totiž chování GetHashCode závislé na své implementaci, která se může lišit mezi jednotlivými verzemi CLR. Také hodnota vrácená metodou GetHashCode pro určitý řetězec se zpravidla liší pro 32-bitovou a 64-bitovou verzi .NET Frameworku.

K problému by tedy mohlo dojít v okamžiku, kdy nějaký recept získáme a uložíme s kontrolním součtem vypočteným nad určitou verzí .NET Frameworku a následně spustíme sběrač receptů nad jinou verzí .NET Frameworku, nalezneme stejný recept, ale jeho kontrolní součet bude odlišný od toho předchozího. Pak dojde k opětovnému uložení do databáze.

Proto se kontrolní součet vypočítává metodou Recipe.GetCheckSum, která pro řetězce místo jejich metody GetHashCode využívá metodu Recipe.GetCheckSum(string).

4.3.5 Zjištění ID vloženého záznamu

Při ukládání receptu se nejprve vkládá záznam do tabulky RECIPE, následně ingredience do tabulky RECIPE_ITEM. Než začneme ukládat ingredience, musíme znát id, které dostal záznam v tabulce RECIPE, abychom na něj mohli odkazovat. Databáze MSSQL nabízí tři způsoby, jak získat id posledního vloženého záznamu.

1. IDENT_CURRENT('jméno tabulky') – vrací poslední id vygenerované pro danou tabulku, pro libovolnou session a modul[1].
2. SCOPE_IDENTITY() – vrací poslední id vygenerované pro libovolnou tabulku v současné session a současném modulu.
3. @@Identity – vrací poslední id vygenerované pro libovolnou tabulku v současné session, přes všechny moduly.

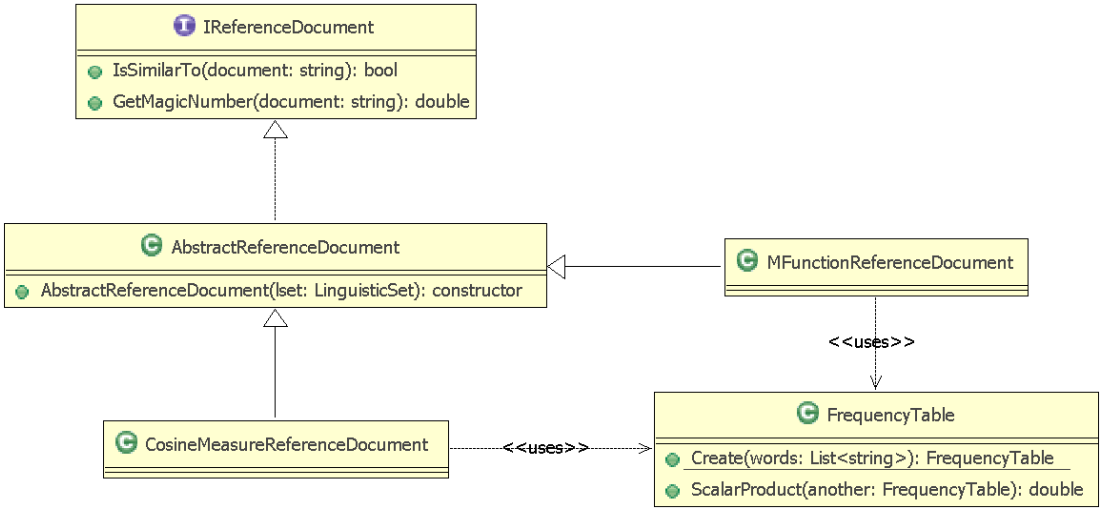
Původně byla použita první varianta. Při souběžném sběru však docházelo k situaci, kdy dva sběrače vložily záznam do tabulky RECIPE, a až poté se dotazovaly na id, čímž oba dostaly stejné. Pro zajímavost šlo o 326 kolizí na 13 223 receptech, tj. podíl chybných receptů 2,47%.

Druhá varianta nemůže být použita, protože každý příkaz je chápán jako samostatná dávka, je tedy použita varianta 3.

1 Session pro srozumitelnost uvedeno v anglickém jazyce. Modulem je zde označena uložená procedura, trigger, funkce nebo dávka. Dva příkazy jsou tedy ve stejném modulu, jsou-li ve stejné uložené proceduře, funkci nebo dávce.

4.3.6 Referenční dokument

Referenční dokument splňuje rozhraní IReferenceDocument. Jeho metoda IsSimilarTo pro každý poskytnutý řetězec rozhodne, zda je dostatečně podobný tomu referenčnímu či nikoliv. Metoda GetMagicNumber je pro testovací účely, vrací číslo, které má smysl pouze ve spojení s konkrétní implementací (např. hodnotu kosinové míry). Třída AbstractReferenceDocument sjednocuje práci s lemmatizátorem a stoplistem (tuto dvojici zastupuje třída LinguisticSet). Obě implementace referenčního dokumentu používají třídu FrequencyTable. Pomocí této třídy je možné reprezentovat vektor vah jednotlivých termů ve vektorovém modelu. Třída také nabízí metodu ScalarProduct, která vypočítá hodnotu skalárního součinu s jinou frekvenční tabulkou.



Obrázek 13: Třídy a rozhraní referenčních dokumentů.

4.3.7 Třída ActivityManager

Třída ActivityManager je zodpovědná za určování aktivit potřebných pro dokončení receptu na základě analýzy textu popisujícího postup přípravy receptu. Toto určení probíhá v rámci statické metody AddActivities(List<Recipe>).

Aktivity jsou ukládány v tabulce ACTIVITY. Každá aktivita má název (name) a tzn. *masku*. Maska je číslo ve tvaru 2^k, pro hodnoty k od jedné do 31. Každá aktivita má svou masku. Všechny aktivity daného receptu se pak ukládají do jediného sloupce jako bitový (a zároveň aritmetický) součet masek jednotlivých aktivit. Tento způsob uložení později umožňuje klást podmínky na aktivity v SQL pomocí bitových operací. Slova, podle kterých se rozpoznávají aktivity popisované v textu, jsou uloženy v tabulce ACTIVITY_SYNONYM.

4.4 DOGAPO_WEB

Projekt DOGAPO_WEB implementuje webové rozhraní. Projekt sestává z jednotlivých aspx stránek. Stránky používají společnou šablonu template\Main.master. Styly jsou definovány v souboru template\Main.css.

4.4.1 Uložená procedura SEARCH_RECIPE

Nejzajímavější částí implementace webového rozhraní je vyhledávání receptů podle zadaných kritérií. Samotné vyhledávání receptů vykonává uložená procedura SEARCH_RECIPE. S ní souvisí dva problémy, které bylo potřeba vyřešit. Jednak způsob předání parametrů vyhledávání, jednak efektivní způsob párování ingrediencí.

Předání parametrů. Parametry jednoduchých datových typů se předávají snadno. Aktivity, které je uživatel ochoten vykonat, se díky reprezentaci pomocí masek předávají také snadno v jediném čísle. Největší problém je předání ingrediencí a příp. jejich množství. Nejlepším řešením se ukázalo předání v podobě dvou řetězců (varchar(8000)) obsahujících jednotlivé hodnoty oddělené znakem #. Na začátku procedury SEARCH_RECIPE jsou tyto řetězce převedeny do dočasné tabulky @USER_HAS pomocí funkce fn_Split:

```
declare @USER_HAS TABLE (root varchar(255), amount int)

insert into @USER_HAS select r.value, a.value
from dbo.fn_Split(@rootList, '#') r join
     dbo.fn_Split(@amountList, '#') a on r.position = a.position;
```

Výpis 5: Vytvoření tabulky ingrediencí, které má uživatel, v proceduře SEARCH_RECIPE

Párování ingrediencí. Vzhledem ke složitosti podmínek, které definují párování ingrediencí, není možné spojit (join) tabulku @USER_HAS s tabulkou RECIPE pomocí podmínky na rovnost. Pokud se logika párování implementuje pomocí SQL funkce, která se následně v podmínce spojení použije, je vyhodnocení dotazu nepřijatelně zdlouhavé – prakticky vyzkoušeno s funkcí MATCHES. Tento přístup totiž vyžaduje porovnání s každým záznamem v tabulce RECIPE_ITEM, kterých je řádově stovky tisíc. Pro zvýšení efektivity je potřeba párování vypočítat dopředu. Během dotazu pak lze použít již spočtené párování, rovnost v podmínce spojení tabulek a sílu databázových indexů. Každé dva kořeny ze sloupce RECIPE_ITEM.root, které tvoří pár, jsou uloženy v tabulce ROOT_MATCHES. Pak je možné psát:

```
from
     @USER_HAS uh join
     ROOT_MATCHES rm on uh.root = rm.have join
     RECIPE_ITEM ri on rm.need = ri.root join
     RECIPE r on ri.recipe = r.id
```

Výpis 6: Použití tabulky ROOT_MATCHES.

4.4.2 Tabulka ROOT_MATCHES

Tabulka ROOT_MATCHES obsahuje dvojice kořenů, které tvoří pár ve smyslu oddílu *2.5.1 Párování ingrediencí*. Její konstrukce vyžaduje porovnání každé dvojice kořenů, má tedy kvadratickou složitost vzhledem k počtu kořenů.

Tabulka vzniká postupně během sběru receptů. Jak jsou recepty přidávány, přibývají páry, které v tabulce ještě nejsou. Aktualizaci tabulky provádí uložené procedury UPDATE_ROOT_MATCHES a UPDATE_ROOT_MATCHES_USING_ISA. Pro zjištění, kolik nových kořenů přibylo, a rozhodnutí, zda spustit aktualizaci, slouží pohled ROOT_MATCHES_TO_BE_UPDATED, který má jediný záznam a jediný sloupec newRoots obsahující počet nových kořenů.

Volání uložených procedur uvedených v předchozím odstavci z kódu zajišťuje třída RootManager a její statické metody UpdateRootMatches a UpdateRootMatchesUsingISA. Kromě toho nabízí RootManager také metodu UpdateRootMatchesInMemory, jejíž efekt je stejný jako volání uložené procedury UPDATE_ROOT_MATCHES. Rozdíl je v tom, že tato metoda provádí rozhodování o párech v rámci C# kódu a do databáze pouze uloží výsledky. Tento způsob je přibližně dvakrát rychlejší než čistě databázová implementace.

4.4.3 Třída AdminManager

Pro řízení přístupu k funkcionalitám, které jsou k dispozici pouze v režimu administrátora, slouží třída AdminManager. Přihlášení probíhá pomocí metody Login. Pokud jsou poskytnuté jméno a heslo správné, metoda Login vrátí hodnotu true a do HTTP session uloží informaci o úspěšném přihlášení. Pro odhlášení slouží metoda Logout.

Každá stránka, která má být přístupná pouze v režimu administrátora, musí ve své metodě Page_Load zavolat metodu AdminManager.AssertAdminMode. V případě, že uživatel na tuto stránku přistupuje bez řádného přihlášení, bude přesměrován na stránku pages/admin/denied.aspx.

Stránky, které jsou k dispozici jak pro návštěvníka, tak pro administrátora, mají vedle společných částí také části, které nejsou pro návštěvníka viditelné. Pro zjištění režimu je možné použít metodu IsAdminMode a psát např.:

```
<asp:Panel
      ID="editRecipePNL"
      runat="server"
      Visible='<%# Dogapo.Web.AdminManager.IsAdminMode() %>'
>
```

Výpis 7: Příklad použití metody AdminManager.IsAdminMode.

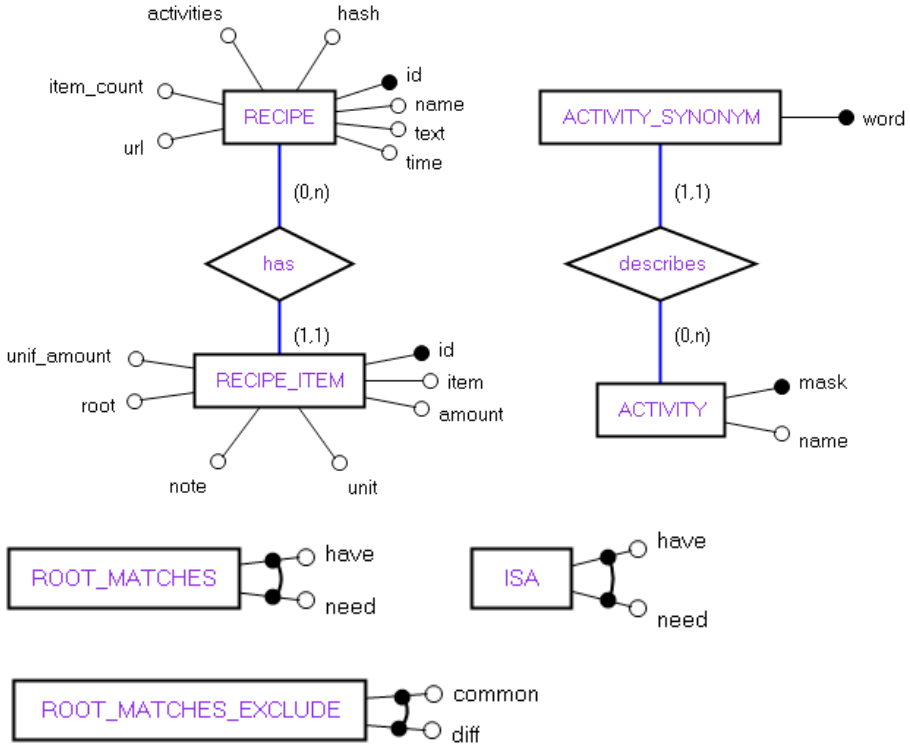
4.5 DOGAPO_DB

Projekt DOGAPO_DB soustřeďuje na jedno místo SQL skripty pro

- založení databázového schématu – create\create.sql
- vložení iniciálních dat – create\data.sql
- vyčištění databáze – clean\clean.sql

4.5.1 Datový model

Datový model sestává ze sedmi databázových tabulek. Dvě dvojice z nich mají mezi sebou relační vztah – RECIPE a RECIPE_ITEM, ACTIVITY a ACTIVITY_SYNONYM. Zbylé tři tabulky (ROOT_MATCHES, ISA a ROOT_MATCHES_EXCLUDE) se vyskytují izolovaně.



Obrázek 14: Datový model.

Tabulka ISA definuje párování významově si odpovídajících ingrediencí. Sloupec have obsahuje kořen ingredience, kterou má uživatel (např. „vejc“), sloupec need obsahuje kořen ingredience, kterou vyžaduje recept (např. „bilek“). Vztah definovaný touto tabulkou je nesymetrický. Pro symetrický vztah je potřeba vložit dva záznamy.

Tabulka ROOT_MATCHES_EXCLUDE definuje vyjímky z obecného párování. Společná část kořene je ve sloupci common, rozlišovací část ve sloupci diff.

4.6 DOGAPO_COMMON

Projekt DAGAPO_COMMON obsahuje knihovnu funkcionalit, které je potřeba sdílet mezi projekty DOGAPO_COLLECTOR a DOGAPO_GUI. Jde o pomocné třídy, třídy pro přístup k datům (DAO), konfiguraci a některé další třídy.

4.6.1 Konfigurace

Konfigurací se chápe složka config a její obsah. Složka musí být vždy umístěna ve stejné složce jako DOGAPO_COMMON.dll. Jedná se zejména o soubor config.xml, který obsahuje základní nastavení jako např. připojení k databázi. Dále je důležitý soubor sources.xml, který definuje jednotlivé zdroje dat. Za součást konfigurace jsou považovány i soubory s regulárními výrazy odkazované z tohoto souboru. Dále jsou součástí konfigurace následující soubory:

- item-names.txt
- ref-recipe-text.txt
- stemmer-endings.txt
- stoplist.txt
- units.txt

Pro přístup k nastavení slouží třída ConfigProvider. Jednotlivé hodnoty ze souboru config.xml se získávají pomocí metody Get – pro zadaný klíč vyhledá hodnotu. K obsahu jednotlivých souborů se přistupuje pomocí metody GetFileContent.

4.6.2 Implementace lemmatizátoru

Lemmatizace je velice častou operací, proto je důležité, aby její implementace byla rychlá. Každá implementace musí splňovat rozhraní IStemmer. Společným předkem všech implementací je třída AbstractStemmer, která poskytuje implementaci metody GetStemList. Instance lemmatizátoru se získává pomocí statické metody GetStemmer třídy StemmerFactory.

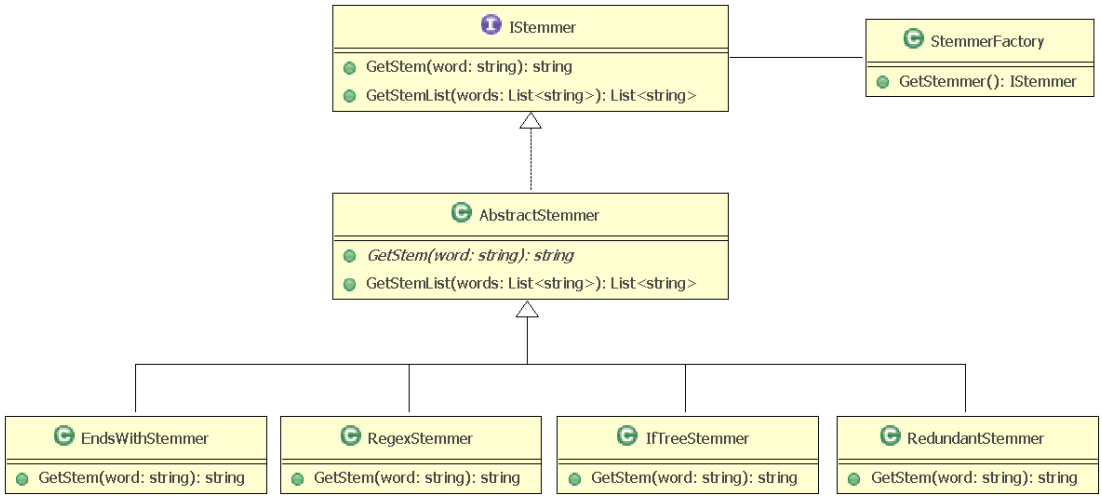
První implementace, která se nabízela, bylo použití metody EndsWith třídy string – postupné zkoušení koncovek od nejdelší po nejkratší, kterou koncovkou slovo končí. Třída s touto implementací se jmenuje EndsWithStemmer.

Další varianta využívala regulárních výrazů – třída RegexStemmer. Ukázalo se však, že tato implementace je ještě pomalejší než předchozí.

Třetí varianta, která se nakonec ukázala jako dostatečně rychlá a použitelná, využívá generování kódu. Na základě koncovek se vygeneruje rozhodovací strom (if-then-else). Generování zajišťuje třída IfTreeStemmerGenerator, kód generuje do připravené šablony IfTreeStemmer.cs.template. Výsledkem je pak třída IfTreeStemmer.

Třída RedundantStemmer sice vypadá jako další implementace, slouží však pouze pro kontrolu správnosti. Ve skutečnosti obaluje více lemmatizátorů. Pro každé volání metody GetStem zavolá RedundantStemmer tuto metodu na všech svých lemmatizátorech. Pokud nevrátí všechny stejnou hodnotu, vyvolá se výjimka.

Výhoda implementací EndsWithStemmer a RegexStemmer spočívá v tom, že jejich chování je možné změnit v konfiguračním souboru stemmer-endings.txt, zatímco IfTreeStemmer by bylo nutné při změně koncovek přegenerovat.



Obrázek 15: Různé implementace rozhraní IStemmer.

Na obrázku 15 jsou vidět třídy, které souvisí s implementací lemmatizátoru. Následující tabulka ukazuje srovnání rychlostí jednotlivých implementací. Hodnota ve sloupci *rychlost* je poměrná vzhledem k rychlosti třídy EndsWithStemmer, hodnoty jsou určeny pomocí profileru na základě 2000 volání metody GetStem.

Implementace	Rychlost
EndsWithStemmer	1
RegexStemmer	5,5
IfTreeStemmer	0,04

Tabulka 3: Srovnání rychlostí lemmatizátorů.

Kapitola 5
Konkurenční aplikace

Cílem této kapitoly je představit existující aplikace řešící podobný problém jako tato práce. Autor si není vědom žádné implementace, která by vykonávala podobnou činnost jako sběrač receptů. Je však velké množství serverů, které umožňují vyhledávání receptů podle zadaných kritérií. Největší dva z nich budou popsány a porovnány s webovým rozhraním DOGAPO. Při porovnávání bude věnována zvláštní pozornost možnostem při vyhledávání.

5.1 Server recepty.cz

Server recepty.cz obsahuje přes 17 000 receptů rozdělených do kategorií. K receptu je možné přidat fotografii. Uživatelé mohou sami recepty vkládat a zapojit se do diskuze k již publikovaným receptům. Ingredience jsou chápány jako souvislý text. Vyhledávání je možné podle

- kategorie
- autora
- klíčového slova, které se vyhledává v názvu a ingrediencích

5.2 Server recepty.vareni.cz

Server recepty.vareni.cz obsahuje přes 17 000 receptů rozdělených do kategorií podle více kritérií – podle chodu, země původu a příležitosti. K receptu je možné přidat fotografii. Uživatelé mohou sami recepty vkládat a zapojit se do diskuze k již publikovaným receptům. Také mohou hodnotit recept pomocí pětihvězdičkové stupnice. Ingredience jsou strukturované – dvojice název ingredience a její množství. Vyhledávání je možné podle

- všech kategorií
- klíčového slova v názvu a postupu
- ingrediencí – je možné zadávat více oddělených čárkou
- je možné vyhledávat recepty pouze s fotografií či pouze pro vegetariány

Ve výsledku jsou všechny recepty, které obsahují alespoň jednu zadanou ingredienci. Výchozí řazení receptů je podle hodnocení uživatelů (pět hvězdiček), dále je možné řádit podle chodu nebo doby přípravy.

5.3 Srovnání

Níže uvedená tabulka poskytuje názorné srovnání uvedených implementací a programu vytvořeného v rámci této práce. Zatímco ostatní implementace kladou důraz na rozdělení receptů do kategorií, uživatelské účty a vkládání receptů či možnost přiložit fotografie, DOGAPO přenáší důraz na schopnost vyhledávat podle ingrediencí a řadit výsledky podle různých kritérií relevance. Absence rozdělení do kategorií je kompenzována možností vyhledávat podle činností potřebných pro recept.

Vlastnost	recepty.cz	recepty.vareni.cz	DOGAPO
Rozdělení receptů do kategorií	ANO	ANO	—
Možnost přiložit fotografii	ANO	ANO	—
Uživatelské vkládání receptů	ANO	ANO	pouze admin
Vyhledávání podle doby přípravy	—	ANO	ANO
Vyhledávání v názvu	ANO	ANO	—
Vyhledávání v postupu	—	ANO	—
Vyhledávání v ingrediencích	ANO	ANO	ANO
Vyhledávání více ingrediencí	—	ANO	ANO
Vyhledávání podle množství	—	—	ANO
Vyhledávání podle činností	—	—	ANO
Relevance výsledků podle kritérií	—	—	ANO

Tabulka 4: Srovnání s existujícími implementacemi.

Kapitola 6
Případová studie

Cílem této kapitoly je ukázat na konkrétních případech, které byly popsány v úvodu, možnosti a schopnosti vyhledávání receptů ve webovém rozhraní. Databáze, nad kterou se budou dotazy pouštět, obsahuje přes 30 000 receptů.

6.1 Student Adam

Situace. Student Adam pozve několik svých přátel na návštěvu. Když po nějaké době začnou mít jeho přátelé hlad, rozhodne se, že pro ně něco rychle připraví. Adam bude potřebovat najít recept, na který může použít nejlépe jenom suroviny, které má k dispozici. Navíc bude chtít přípravu stihnout co nejdříve.

Předpokládejme, že Adam bude formulovat následující dotaz:

Ingredience: párek, chleba, kečup, špagety
Chci: maximálně „pokrýt“ recept
Čas na přípravu: do 30 minut

Výsledkem tohoto dotazu je (pouze první čtyři výsledky):

Název receptu	Čas	Relevance	Ingredience
Chlebová strouhanka	15minut	100%	chleba
Studentská omáčka	15minut	67%	cibule, párek, kečup
Rychlé pikantní tousty	15minut	50%	eidam, kečup, pečivo, ďábelská směs na topinky
Studentská večeře	15minut	50%	arabský chléb, tvrdý sýr, olej, kečup

Tabulka 5: Výsledky vyhledávání pro studenta Adama.

Adam může připravit pouze chlebovou strouhanku. Je možné si všimnout, že u „rychlých pikantních toustů“ se spároval chleba a pečivo. Pokud by se Adam rozhodl, že je ochoten jít nakoupit, mohl by formulovat dotaz takto:

Ingredience: párek, chleba, kečup, špagety
Chci: nejlepší shodu
Čas na přípravu: do 30 minut

Výsledek upraveného dotazu ukazuje následující tabulka. Tři recepty z předchozích výsledků jsou stále přítomny, zmizela však chlebová strouhanka.

Název receptu	Čas	Relevance	Ingredience
Studentská omáčka	15minut	40%	cibule, párek, kečup
Rychlé pikantní tousty	15minut	33%	eidam, kečup, pečivo, ďábelská směs na topinky
Studentská večeře	15minut	33%	arabský chléb, tvrdý sýr, olej, kečup

Tabulka 6: Výsledky upraveného vyhledávání pro studenta Adama.

6.2 Paní Bára

Situace. Rodina paní Báry se stěhuje a je potřeba dojíst potraviny, které doma zůstaly. Paní Bára je ochotná nějaké suroviny přikoupit, ale hlavní je, aby se spotřebovaly ty, které má doma. Také už má sbalený mixér v krabici, takže recepty, ke kterým by jej potřebovala, ji nezajímají.

Paní Bára bude formulovat následující dotaz:

Ingredience: olej, mrkev, česnek, sůl, slanina, fazole
Chci: maximálně využít suroviny, které mám
Činnosti: ne mixování

Výsledkem tohoto dotazu jsou:

- Dva recepty, které využívají všechny ingredience („Špagety s fazolemi a slaninou“ a „Pikantní fazolová polévka“),
- 42 receptů, které využití pět ze šesti surovin,
- stovky receptů, které využijí čtyři ze šesti surovin.

Kapitola 7
Závěr

Cílem závěrečné kapitoly je shrnout všechny důležité problémy, které byly vyřešeny, poskytnout kritický pohled na nevyřešené části a nastínit možnosti dalšího rozvoje.

První část práce – sběrač receptů – je nástroj, který umožňuje definovat zdroje receptů na Internetu a z nich pak recepty získávat. Zdroje mohou mít buď pravidelnou strukturu (pak lze pro jejich popis použít regulárních výrazů), nebo mají strukturu náhodnou (pak se uplatní mechanizmus automatického rozpoznání receptu). Recepty se po získání analyzují, rozpoznávají se jednotlivé ingredience, jejich množství, jejich počet, dále se rozpoznávají činnosti, které je potřeba pro jejich dokončení. Pokud není u receptu uvedena doba přípravy, je automaticky odhadnuta. Dále je vytvořen mechanizmus, který zajišťuje, že stejný recept nebude do databáze uložen dvakrát. Tím je umožněno opakovaně spouštět sběrač receptů na stejný zdroj pro získání pouze nových receptů. Sběrač receptů nabízí dvě uživatelská rozhraní – grafické pro interaktivní použití a na příkazové řádce pro automatické spouštění.

Druhá část práce – webové rozhraní – poskytuje možnost vyhledávat v databázi receptů. Vyhledávat je možné podle doby přípravy a činností, které je uživatel ochoten vykonat. Největší důraz je však kladen na vyhledávání podle ingrediencí. Zadané ingredience se párují s ingrediencemi v receptech. Párování podporuje i složitější mechanizmy jako významovou příbuznost (rohlík – pečivo), významovou odlišnost pro podobné ingredience (pečivo – prášek do pečiva) nebo jednoduché zohlednění množství zadaného vzhledem k potřebnému. Na základě tohoto párování jsou vystavena tři uživatelsky volitelná kritéria pro určení relevance. Recepty ve výsledném seznamu jsou seřazeny podle této relevance (sestupně). Administrátor může přes webové rozhraní recepty přidávat, upravovat a odstraňovat.

7.1 Možnosti rozšíření

Přes mnohé úspěchy, kterých bylo dosaženo, zůstávají věci, které by mohly být přidány či udělány lépe. Uveďme nejdůležitější z nich:

- **Vylepšené možnosti sběru** – přerušení probíhajícího sběru a pokračování jindy. Dále možnost více paralelních sběrů v rámci jedné instance programu. V současné době lze docílit paralelizmu jen spuštěním více instancí.
- **Orientace na uživatele –** umožnit uživatelům vkládat vlastní recepty, evidovat si oblíbené recepty, vkládat fotografie, hodnocení a diskuzní příspěvky.

- **Rozšíření poskytovaných informací –** poskytnout informace o druhu (nápoj, hlavní chod, polévka) nebo kategorii (maso, těstoviny, zelenina)**.**
- **Kalorická kalkulačka –** systém eviduje množství u ingrediencí, nabízí se tak možnost dopočítávat k receptu jeho kalorickou hodnotu.
- **Zobrazování podobných receptů** v detailu receptu. Podle ingrediencí, které obsahuje právě zobrazovaný recept by se vyhledaly jemu podobné a jejich náhledy by se zobrazily např. na konci stránky.

Reference

[1] *Wikipedie*. Dostupné na http://cs.wikipedia.org/wiki/Kmen_(mluvnice).

[2] *Wikipedie*. Dostupné na http://cs.wikipedia.org/wiki/Ko%C5%99en_(mluvnice).

[3] *Wikipedie*. Dostupné na http://cs.wikipedia.org/wiki/Lemma_(lingvistika).

[4] Liu B.: *Web Data Mining: Exploring Hyperlinks, Contents and Usage Data.* Springer, 2007.

[5] Pokorný J., Snášel V., Húsek D.: *Dokumentografické informační systémy.* Karolinum, Praha, 1998, s. 89–94.

[6] Žemlička M.: *Signaturové metody hledání v kolekcích textů.* Dostupné na http://www.ksi.mff.cuni.cz/~zemlicka/pdf/sigtext.pdf. Citováno 12. července 2009.

[7] *The R Project for Statistical Computing.* Dostupné na http://www.r-project.org/.

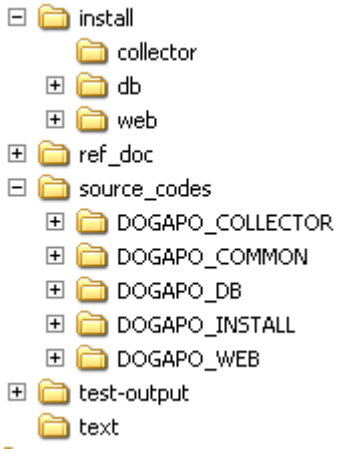
[8] *NProf – statistický profiler pro .NET aplikace.* Dostupný na http://code.google.com/p/nprof/.

[9] *NDoc 2.0 Alpha – nástroj pro generování dokumentace pro .NET 2.0.* Dostupné na http://www.kynosarges.de/NDoc.html.

[10] *Microsoft Developer Network.* Dostupné na http://msdn.microsoft.com/.

Příloha A: Obsah přiloženého CD

Obsahem této přílohy je popis adresářové struktury přiloženého CD. Tuto strukturu zobrazuje níže uvedený obrázek.



Obrázek 16: Struktura CD.

Ve složce install jsou instalační soubory. Jsou rozděleny do tří podsložek – složka collector obsahuje instalátor sběrače receptů, složka db skripty pro založení databázového schématu a vložení iniciálních dat a složka web obsahuje soubory webového rozhraní. Postup instalace je popsán v oddíle *3.1 Instalace a konfigurace*.

Referenční dokumentace, vygenerovaná na základě XML komentářů ve zdrojových kódech, je uložena ve složce ref_doc. Navigaci lze začít souborem index.html.

Zdrojové kódy jsou uloženy ve složce source_codes jako řešení (solution) pro MS Visual Studio 2005. Projekt DOGAPO_INSTALL, který nebyl dříve zmíněn, je projekt instalátoru pro sběrač receptů.

Složka test-output obsahuje výsledek testu automatického rozpoznávání receptů, odkazovaný v oddíle *2.2.6. Efektivita automatického rozpoznávání receptů.*

Text této práce ve formátu PDF je umístěn ve složce text.

Příloha B: Testovací prostředí

Možnost testovat aplikaci bez nutnosti instalace umožňuje testovací prostředí, které bude k dispozici nejméně do 18. září 2009. V rámci tohoto prostředí je k dispozici databáze a webové rozhraní. Následují přístupové údaje k jednotlivým částem:

Databáze

Server:	sportka.ms.mff.cuni.cz
Databáze:	dogapo
Uživatel:	jelij5am
Heslo:	dogapo3

Webové rozhraní

Web:	http://dogapo.asp2.cz/
Admin login:	admin
Admin password:	password
FTP server:	ftp://dogapo.asp2.cz
FTP login:	dogapo
FTP heslo:	3PB5XwwSr

Upozornění. Platnost hesla do databáze čas od času vyprší, pak je potřeba jej změnit. V takovém případě se heslo mění postupně na dogapo4, dogapo5 … Tato změna se musí reflektovat do dvou konfiguračních souborů webového rozhraní:

- \wwwroot\Web.Config
- \wwwroot\Bin\config\config.xml

Pro správnou funkci sběrače se musí nové heslo vložit do souboru config\config.xml.