Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava 4

Znalostný manažment na báze technológie .NET

Tímový projekt

Autori: Bc. Vladimír Mlynarovič
Bc. Michal Pažitný
Bc. Zdenko Porubčan
Bc. Daniel Princzkel
Bc. Lukáš Sim

Tím: FOOBE (tím č.19)
Vedúci tímu: Ing. Ivan Polášek, PhD.

Dátum: 14.11.2007

Obsah

0 Úvod

0.1 Úvod do problematiky

Znalostný manažment a znalostná ekonomika pútajú čím ďalej, tým väčšiu pozornosť modernej spoločnosti. Na základe toho sa objavujú zaujímavé projekty zamerané na túto tému, ako systémy znalostného riadenia podnikov, či celých spoločností. Podobná búrlivá diskusia sa vedie aj ohľadom sémantického webu. Aj napriek veľkej snahe v oblasti výskumu, stále chýbajú zdroje a prostriedky, pričom používateľsky vhodný a akceptovateľný systém neustále absentuje. Naďalej preto vzniká priestor pre vyvíjanie systémov, ktoré by boli schopné znalosti reprezentovať, spracovávať, vytvárať relácie s ostatnými znalosťami, modifikovať ich a pod.

0.2 Účel dokumentu

Snahou tohto projektu je preto vytvoriť dynamický sémantický model, ktorý by napomohol pri tvorbe nových dokumentov (zdrojových kódov, dokumentácií a manuálov a pod.) pomocou získaných znalostí. Touto témou sa už v predošlom roku zaoberal tím *Lucky Number 7*, ktorého výsledky by mali priniesť značnú pomoc a umožniť preniesť sa rýchlejšie cez už prekonané problémy. Úlohou nášho tímu je zároveň posunúť sa v tejto oblasti ďalej, teda predostrieť nové možnosti, myšlienky či nápady, o ktorý bude v konečnej fáze samotný systém obohatený.

0.3 Prehľad dokumentu

Celý dokument je členený do troch základných častí. Prvou je analýza, ktorá je venovaná znalostiam a znalostnému manažmentu. Po nej nasleduje analýza znalostného systému vytvoreného tímom *Lucky Number 7* minulý rok. Zvyšnú časť analýzy tvoria podrobnejšie úvahy o možných riešeniach a koncepciách v projekte. Špecifikáciu požiadaviek tvoria najmä dva modely. Prvým je model prípadov použitia, ktorého dopĺňa popis identifikovaných hráčov v systéme a tabuľky jednotlivých prípadov použitia. Druhým modelom je dátový model, konkrétne jeho logická úroveň. Kapitolu špecifikácie uzatvárajú nefunkcionálne požiadavky na systém. Posledná kapitola má názov návrh. V jej úvode je naznačený pohľad na celkovú architektúru systému a fyzický model údajov. Architektúra delí systém do jednotlivých modulov, ktorých činnosť je neskôr vysvetlená podrobnejšie.

0.4 Použité skratky

DPL – (Data Provider Layer) je vrstva poskytovania dát vyvinutý a štandardizovaný konzorciom W3C (World Wide Web Consortium)

OWL – (Web Ontology Language) jazyk na definovanie webových ontológií

RDF – (Resource Description Framework) je World Wide Web Consortium (W3C) špecifikácia formátu podporujúca popis zdrojov pomocou abstraktného modelu metadát, implementovaná v jazyku XML

UC – (Use Case) prípad použitia

VD – výber dokumentov

XML – (eXtensible Markup Language) v preklade rozšíriteľný značkovací jazyk, ktorý bol vyvinutý a štandardizovaný konzorciom W3C (World Wide Web Consortium)

1 Analýza

1.1 Úvod do manažmentu znalostí

Informačná spoločnosť, ako sa súčasná spoločnosť zvykne označovať, čelí nespočetnému množstvu informácií z veľkého množstva zdrojov. Prudký nárast množstva informácií za posledné obdobie so sebou však priniesol i nežiaduce problémy. Ťažkosti s prístupom, extrahovaním, interpretáciou, udržovaním informácií zostávajú na používateľovi. Myšlienka reprezentácie informácií vo forme znalostí nie je najmladšia, napriek tomu však táto téma ostáva stále otvorená a prebádaná iba čiastočne. Práve znalosti by mali posunúť informácie na vyššiu úroveň v reprezentácií, prehľadnosti či prístupe a vyhľadávaní.

Zavádzania manažmentu znalostí do praxe podnecujú hlavne tieto tri faktory [4]:

1. Od času potrebného na nasadenie výrobku na trh je závislý úspech či neúspech daného produktu. Pre spoločnosti je nevyhnutné využiť výhody prvotného iniciátora na trhu alebo aspoň rýchle vytvorenie konkurencie. Z tohto dôvodu je pre firmy nevyhnutné plne využitie ich znalostí týkajúcich sa trhu, zákazníkov, konkurenčných firiem a dostupných technológií.

2. Iná situácia vzniká pri fúzii alebo reštrukturalizácii firiem, ktoré hlboko závisia na nadobudnutých znalostiach. Pri týchto operáciách je veľmi dôležité vedieť preniesť znalosti z jednej organizácie do druhej a určiť, ktoré zo znalostí sú užitočné pre čo najrýchlejší profit firmy a čo najdlhšie prežitie na trhu.

3. Moderné informačné technológie poskytujú obrovské množstvo informácií a pomerne jednoduchý prístup k nim. Týchto informácií je veľké množstvo a je priveľmi zložité nájsť požadovanú informáciu dostatočne rýchlo na to, aby sme ju mohli využiť a aby námaha vynaložená na jej získanie neprekročila užitočnú hodnotu tejto informácie.

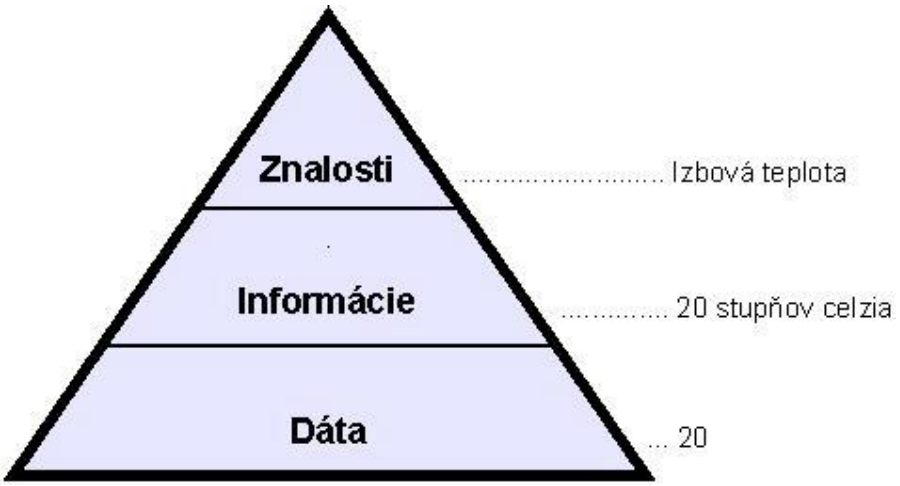
Manažment znalostí čerpá zo širokej škály disciplín ako napríklad [8]:

- Umelá inteligencia, expertné a znalostné systémy
- Počítačom podporovaná spolupráca (groupware)
- Informatika, kognitívne vedy, filozofia
- Správa dokumentov
- Systémy pre podporu rozhodovania
- Re-inžiniering firemných procesov
- Riadenie ľudských zdrojov
- Organizačná kultúra a pod.

Základné pojmy

Manažment znalostí
„Systémy pre vyhľadávanie a spracovanie znalostí a všeobecne prácu so znalosťami označujeme ako manažment znalostí. Aby systém vedel so znalosťami správne zaobchádzať, musí ich vedieť kategorizovať, analyzovať, dalo by sa povedať, že rozumieť im“.[3]

Na obr. 1 je zobrazená hierarchia pojmov v znalostnom manažmente. Pri každom pojme je uvedený aj príklad.



Obr. 1. Hierarchia pojmov v znalostnom manažmente

Dáta
Dáta sú súbor znakov predstavujúcich diskrétne objektívne fakty. Ako také majú dáta veľmi malú výpovednú hodnotu, ale dajú sa jednoducho uchovávať a spracovávať vo výpočtových systémoch.

Informácie
Informácie sú dáta ktoré sú vložené do kontextu. Informácie majú väčšiu hodnotu ako dáta no môžu spôsobovať dvojznačnosť alebo stratiť hodnotu, ak príjemca informácie nerozumie kontextu.

Znalosti
Znalosti sú informácie vložené do konkrétneho kontextu a sú hodnotné pre toho kto ich pozná. Umožňujú mu vykonávať činnosti, ktoré by bez týchto znalostí vykonávať nemohol.

1.2 Analýza znalostného systému vytvoreného tímom Lucky Number 7

Tím Lucky 7 sa zaoberal návrhom a implementáciou produktu, ktorý je založený na vrstvovej architektúre, kde každá vrstva má presne definovanú vlastnú funkcionalitu. Z projektovej dokumentácie však nie je veľmi jasné, z koľkých vrstiev táto architektúra pozostáva. Na lepšie porozumenie tohto návrhu je vhodné doplniť jednoduchý diagram, ktorý by znázornil jednotlivé úrovne a ich poradie. Vykonali sme analýzu jednotlivých vrstiev.

Databázová vrstva

Databázová vrstva slúži na ukladanie a získavanie dokumentov z externej databázy (MS SQL Server 2005 Express Edition). Tvorí rozhranie medzi vyššou vrstvou (Data Provider Layer) a touto databázou. Návrh databázovej vrstvy je rozdelený do dvoch častí.

Prvá časť opisuje návrh dátovej štruktúry, ktorá definuje uchovávaný dokument v databáze. Ide konkrétne o tieto údajové položky:

- identifikátor súboru
- meno súboru (meno súboru zadané pri ukladaní do úložišta)
- obsah súboru
- metadáta opisujúce súbor
- hash jednoznačne identifikujúci súbor

Druhá časť návrhu sa zameriava na opis služieb:

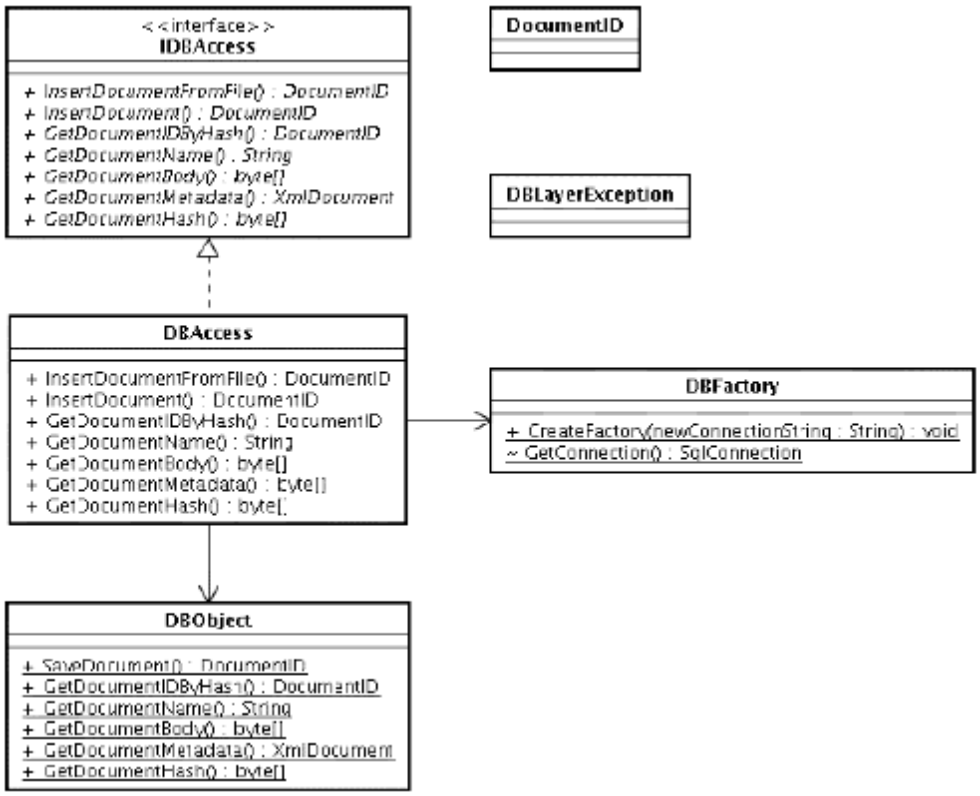
- pridanie dokumentu
- získanie ID dokumentu
- zistenie mena dokumentu
- získanie obsahu dokumentu
- získanie metadát
- získanie hashu

a s nimi súvisiacimi funkcií rozhrania *IDBAccess*:

- `DocumentID InsertDocumentFromFile(String name, String file, XmlDocument metadata, byte[] hash);`
- `String GetDocumentIDByHash(byte[] hash);`
- `String GetDocumentName(DocumentID ID);`
- `byte[] GetDocumentBody(DocumentID ID);`
- `byte[] GetDocumentMetadata(DocumentID ID);`
- `byte[] GetDocumentHash(DocumentID ID);`

ktoré poskytuje databázová vrstva. Tieto funkcie sú vysvetlené podrobne, a teda vieme presne povedať, čo ktorá funkcia vykonáva. Sú definované hlavičky metód, výnimky, vstupy a výstupy.

Pri práci so súbormi sa využíva trieda DocumentID, ktorá zapuzdruje identifikáciu dokumentu v rámci vrstvy. Chybové stavy vznikajúce na tejto vrstve sú obalené do výnimky *DBLayerException*. K pochopeniu celého návrhu a implementácie poslúži nasledovný diagram tried (obr. 2) [1].



Obr. 2. Diagram tried databázovej vrstvy

Vrstva poskytovania dát

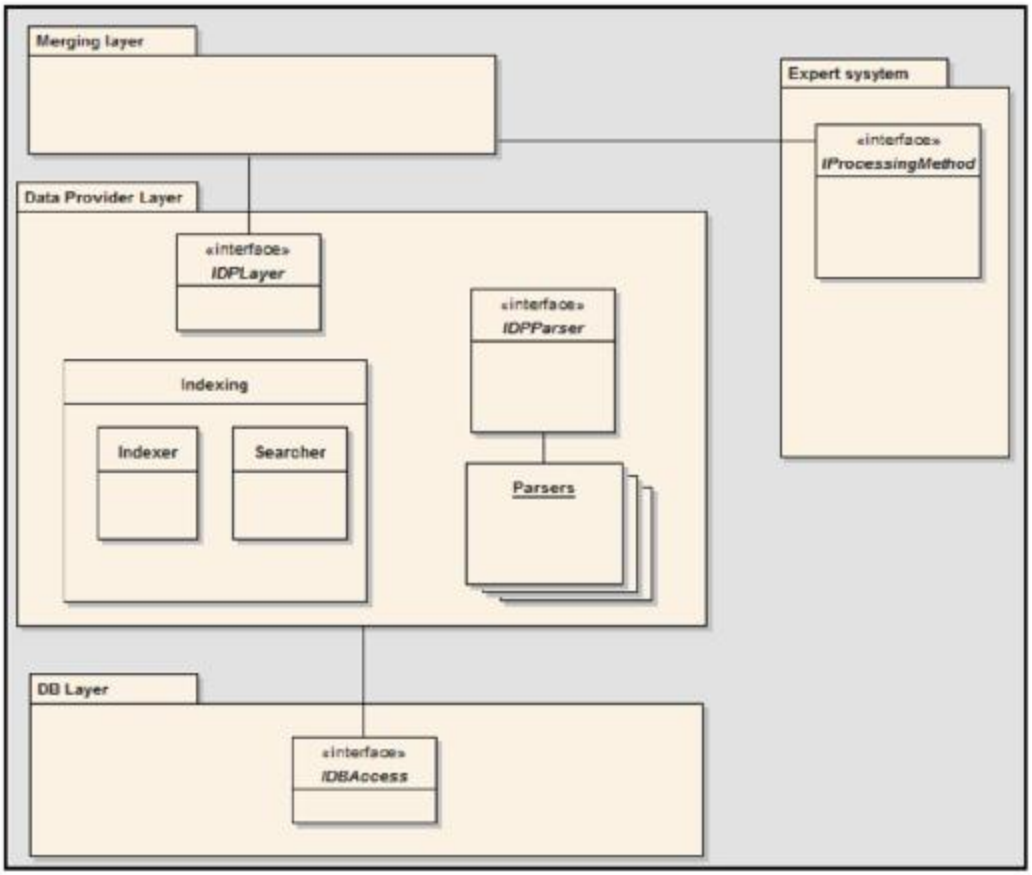
Ďalšou vrstvou, ktorú tím Lucky7 navrhol, je vrstva poskytovania dát (Data provider layer, DPL). Takisto ako pri databázovej vrstve autori sa venovali návrhu a implementácii tejto vrstvy, ako aj opisu vrstvy a jednotlivých požiadaviek.

Úlohou DPL vrstvy je spracovanie údajov zo zadaných dokumentov, ich indexácia a vyhľadávanie. Boli identifikované dva hlavné moduly (parsovací a indexovací modul), ktorých funkciu opísali stručne, ale zato výstižne.

Ako parsovací modul bol vytvorený komponent, ktorý dokáže analyzovať súbory typu .doc a .docx. Bol vytvorený na základe volania metód menného priestoru Microsoft.Office.Interop a využíva pre svoju činnosť aplikáciu MS Word 2007. Preto je pre používanie systému potrebné, aby bola táto aplikácia dostupná na počítači tvoriacom server aplikácie.

Druhý modul slúži pre indexáciu dokumentov podľa ich kapitol v slovnom indexe a následné vyhľadávanie v tomto indexe na základe kľúčových slov. Na vytvorenie tejto funkcionality bola použitá dynamická knižnica pre indexovanie textu s názvom Lucene.Net.

Pre lepšie pochopenie štruktúry týchto modulov v rámci vrstvy, ale aj vrstvy v rámci celej architektúry systému, slúži nasledovný diagram komponentov (obr. 3) [1].



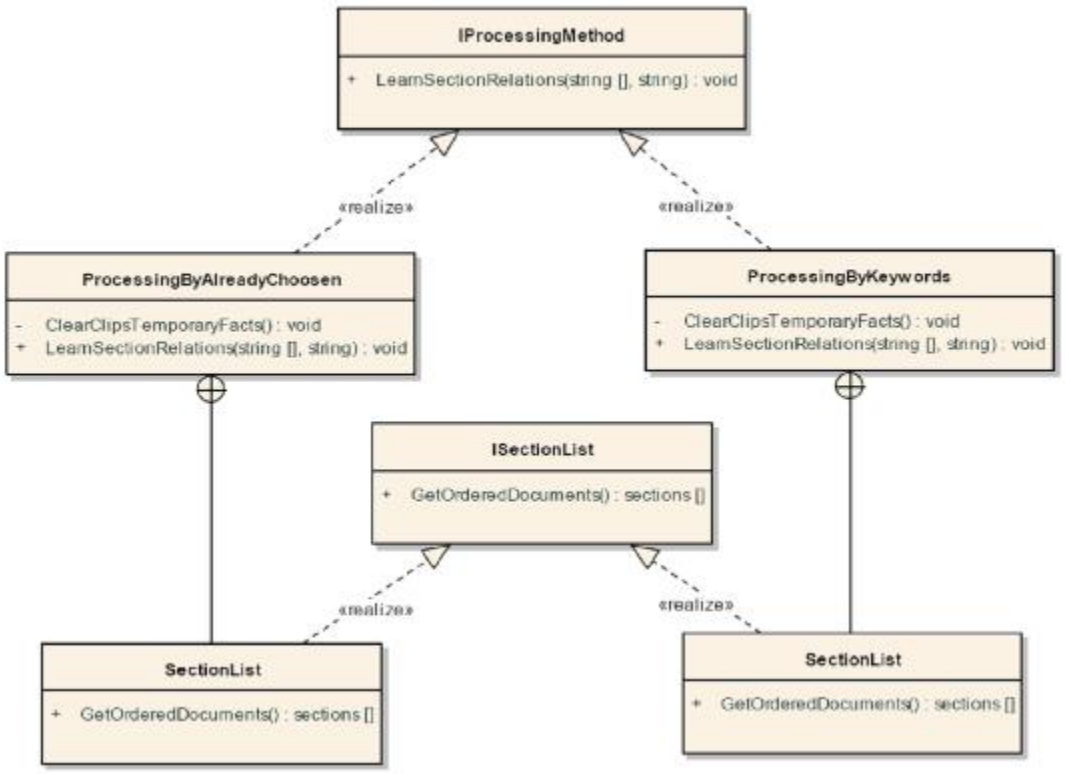
Obr. 3. Štruktúra vrstvy DPL

Komponent - expertný systém

Pre účely expertného systému si tím Lucky7 vybral systém Clips. Venovali sa implementácii jeho integrácie do systému, ako aj metódam vyhľadávania dokumentov. Z definovanej špecifikácie expertného systému sme sa dozvedeli úlohu komponentu v systéme:

- inteligentné vyhľadávanie na základe kľúčových slov
- inteligentné vyhľadávanie na základe kapitol už nachádzajúcich sa v dokumente
- učenie sa na používateľovom výbere kapitoly podľa kľúčových slov
- učenie sa na používateľovom výbere kapitoly na základe kapitol už nachádzajúcich sav dokumente

Návrh modulu je rozdelený na všeobecnú časť nezávislú od použitého expertného systému a časť implementujúcu techniky prehľadávania a doplňania znalostí v Clipse. Každá časť je opísaná samostatne na základe svojich tried. Diagram tried uľahčuje pochopenie architektúry daných častí (obr. 4) [1].



Obr. 4. Architektúra modulu expertný systém

Súčasťou návrhu je aj opis jednotlivých techník používaných na prehľadávanie dokumentov – technika prehľadávania na základe kľúčových slov a technika prehľadávania na základe už existujúcich kapitol. Každá z týchto techník pracuje s rozličnými pojmami a dátovými štruktúrami.

Plugin do MS Word 2007

Predstavuje prístup k integrácii projektu do programu MS Word. Plugin bol vyvinutý v prostredí .NET Framework. V rámci tohto prostredia vytvorili autori 4 projekty pri vývoji plugin-u. Každý z nich zohráva určitú úlohu pri vývoji a testovaní plugin-u – webové služby (napr. na testovanie), vytváranie inštalátora, nastavenie bezpečnosti a samotný projekt s plugin-om – GUIKrtko. O projekte GUIKrtko autori informujú

trochu viac ako o podporných projektoch, ale aj tak sa nedozvedáme informácie o návrhu a bližšom fungovaní samotného plugin-u. Samotnú funkcionalitu sa nám nepodarilo overiť.

1.3 Možnosti spracovania dokumentov

Táto časť projektu sa zaoberá analýzou súčasného stavu v oblasti získavania a vyhľadávania dokumentov. Pred samotným spracovaním dokumentov je nutné definovať, ako majú byť jednotlivé dokumenty reprezentované. Analyzovali sme preto uvedené často používané spôsoby reprezentácie jednotlivých dokumentov, ktoré sa využívajú v tejto oblasti a pokúsili sa o ich porovnanie.

Výber dokumentov

Výber dokumentov (VD) je výpočtový proces selekcie podmnožiny dokumentov z celkovej databázy dokumentov, ktorého výsledok je v sumárnej forme zobrazovaný užívateľovi, zvyčajne ako odozva na užıvateľovu požiadavku. VD systém teda rozdeľuje dokumenty do dvoch tried:

- dokumenty relevantné k užívateľovej požiadavke, ktorých zoznam sa užívateľovi zobrazuje v prijateľnej forme ako výstup VD procesu.

- dokumenty nespĺňajúce požiadavku užívateľa, ktoré nie sú užívateľovi zobrazované.

Našou úlohou nie je len navrhnúť VD systém vykonávajúci selekciu dokumentov, ale taktiež zotriedenie podľa stupňa relevancie k požiadavke. Vypočítavať teda stupeň príslušnosti jednotlivých dokumentov databázy k príslušným triedam. Vo všeobecnosti VD proces môžeme opísať 4 základnými fázami:

I. **indexovanie** – množina vstupných dokumentov je pri určitej textovej reprezentácii konvertovaná do výrazov tejto reprezentácie. Tieto výrazy nazývame ***reprezentanti dokumentov***, pričom ich štruktúra musí byť použiteľná pre ďalšie fázy VD procesu.

II. **formulácia požiadavky** - užívateľ musí zadať čiastočnú informáciu o relevantnosti dokumentov vo forme otázky (request), ktorá je interpretovateľná VD systémom. Otázka sa často zadáva vo forme zrozumiteľnej pre systém, pričom často obsahuje boolovské operátory (and, or, not), prípadne iné (proximity operátor - near). Ďalším spôsobom zadania otázky je zadanie priamou formou v prirodzenom jazyku, pri ktorom si systém vyextrahuje dôležité slová, frázy pre príznaky charakterizujúce požiadavku.

III. **porovnanie –** systém musí implicitne alebo explicitne porovnávať užívateľskú otázku s reprezentantmi dokumentov a vytvoriť tak klasifikačné rozhodnutie o tom, ktoré dokumenty vyberieme, t.j. vyhovujú danej požiadavke, a v akom poradí. Vybrané dokumenty sa potom zobrazia užívateľovi.

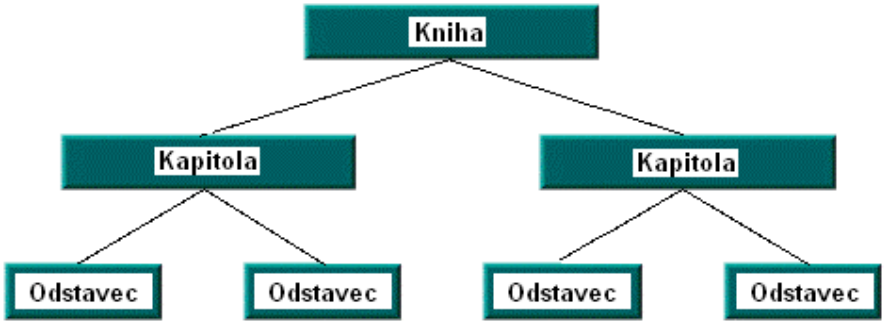
IV. **spätná väzba –** málokedy sa stáva, že prvotný, počiatočný výber dokumentov odpovedá potrebám používateľa. Zvyčajne je potreba niekoľkých iterácií, v ktorých sa požiadavka (otázka) používateľa modifikuje zadaním novej alebo úpravou starej otázky. Táto fáza sa zvyčajne označuje ako *relevance feedback*.

Reprezentácia textových dokumentov

Jednotlivé operácie pre spracovanie dokumentov, ku ktorým sa radí aj klasifikácia prebiehajú nad vstupným príkladovým priestorom tzv. *korpusom*. Korpusom teda nazývame vstupnú kolekciu dokumentov určenú na ďalšie spracovanie. Každý dokument je tvorený kolekciou termov z *univerza* termov. Pod pojmom *term* budeme rozumieť vybranú textuálnu jednotku z textu. Môže ňou byť napr. slovo, kmeň slova (stem), fráza, lema, a pod.

Z hľadiska štruktúry sa na každý dokument môžeme pozerať ako na:

- jednoduché textové pole, pri ktorom nezohľadňujeme štruktúru daného dokumentu
- štruktúrovaný celok tvorený na základe určitých pravidiel pre štruktúrovanie dokumentu, podľa ktorých je dokument rozčlenený do kapitol, odstavcov, a pod. Ukážka štruktúrovaného dokumentu je znázornená na nasledujúcom obrázku (obr. 5)



Obr. 5. Zobrazenie štruktúry dokumentu

Ľubovoľný textový dokument nie je len sústavou slov, fráz a viet v zmysle syntaktickej stavby daného textu, ale je tvorený taktiež sémantickými jednotkami popisujúcimi význam jednotlivých termov. Porozumenie textu si vyžaduje lingvistické znalosti týkajúce sa morfológie slov, sémantiky (významu) slov, štruktúry viet, susednosti slov, a pod. Tieto dodatočné informácie sa môžu zahrnuť do reprezentácie dokumentu, pre lepšie pochopenie jeho obsahu.

Medzi základné problémy, s ktorými sa stretávame pri tvorbe reprezentácie dokumentov, patria:

- **polysemický problém -** jedno slovo má viacero významov
- **synonymický problém -** ten istý pojem, resp. jeho význam sa dá vyjadriť viacerými slovami

Voľbou vhodnej reprezentácie dokumentov dokážeme tieto problémy vo väčšej, či menšej miere potlačiť. V nasledujúcich kapitolách sú uvedené najčastejšie spôsoby reprezentácie dokumentov.

Teoretické modely

Booleovský model reprezentuje dokumenty pomocou skupiny slov, pričom každé slovo predstavuje Booleovskú premennú. Jej hodnota je *Pravda,* ak sa slovo v dokumente nachádza. Váženie (váhovanie) slov, t. j. nakoľko dané slovo reprezentuje obsah dokumentu, nie je povolené. Pri vyhľadávaní dokumentov sa dopyty reprezentujú ako ľubovoľné Booleovské výrazy spojené štandardnými logickými operátormi: AND, OR a NOT.

Fuzzy model sa zakladá na teórii fuzzy množín, ktorá povoľuje čiastočné členstvo prvku v množine - v porovnaní s klasickou teóriou množín, v ktorej prvok buď patrí do množiny alebo nepatrí. Logické operátory sú náležite preddefinované tak, aby zahŕňali čiastočnú príslušnosť k množine. IR modely založené na teórii fuzzy množín sa ukázali ako neschopné rozlišovať relevantné dokumenty medzi vyhľadávanými [6].

Prísne Booleovské a fuzzy modely sú výhodnejšie ako iné modely v zmysle výpočtových nárokov. Malá pamäťová náročnosť na reprezentáciu dokumentu a malá algoritmická zložitosť indexovania a výpočtu podobnosti dopyt – dokument.

Vektorový model

Vector Space (VS) model je najčastejšou a zároveň najjednoduchšou reprezentáciou textových dokumentov. Príznakový priestor pre túto reprezentáciu je konštruovaný na základe množiny slov, pričom každý príznak korešponduje s textovou jednotkou v korpuse. Za textovú jednotku je považované slovo. To znamená, že dokument sa tu chápe ako sekvencia slov bez akýchkoľvek iných znalostí o štruktúre textu, pričom sa neberie ohľad na sémanticky význam jednotlivých slov.

Tento model vychádza len zo štatistických charakteristík dokumentu, a to najmä z distribúcie pravdepodobnosti jednotlivých slov extrahovaných z korpusu a dopytov. Každý dokument je vyjadrený ako vektor d_i v n-rozmernom priestore R^n, kde každá os daného priestoru predstavuje jedno slovo.

Môžeme teda písať pre dokument dok_i :

$$d_i = (w_{i1}, w_{i2}, .., w_{ij}, ..., w_{in}) \quad (1)$$

kde w_{ij} – funkčná hodnota vyjadrujúca váhu, resp. dôležitosť *j*-tého slova v *i*- tom dokumente dok_i.

n – rozmer indexovej množiny slov, t.j. tých slov, ktoré tvoria príznaky popisu jednotlivých dokumentov.

Vektor d_i sa v literatúre označuje aj ako *lexikálny profil* dokumentu.

Celkový korpus *m* dokumentov môžeme tak vyjadriť maticou rozmeru *m x n*. Takáto matica sa častokrát v literatúre označuje ako *term-dokument* matica. Voľba jednotlivých slov opisujúcich korpus dokumentov by mala byť zvolená tak, aby dané slová mali čo najväčšiu rozlišovaciu schopnosť.

Potom je možné definovať operáciu podobnosti

$$sim(d_i,q) = (d_i * q) / (|d_i| \times |d_i|), \quad (2)$$

kde q je vektor reprezentujúci vyhľadávaciu požiadavku.

Možnosť, ako sa dá ovplyvniť reprezentácia dokumentu a tým aj výsledky vyhľadávania je ovplyvnenie váhy jednotlivých kľúčových slov rôznymi spôsobmi ich výpočtu:

- *TF-IDF váhová schéma*

  Wi,j = tfi,j x log (N/ni),

  kde

  N – celkový počet dokumentov

  ni – počet dokumentov, v ktorých sa ki vyskytuje

  fi – frekvencia výskytu ki

  tfi,j = fi,j / (maxi fi,j) je normalizovaná frekvencia pre dokument dj

- *Schéma podľa Saltona a Buckleyho*

  Wi,j=(0,5 + (0,5 x fi,q)/max(Ifi,q)) x log(N/ni),

  kde

Ifi = log (N/ni) je inverzná frekvencia dokumentu pre kľúčové slovo ki

- *Asociačnými pravidlami* [2]

    1. vygenerovanie množiny z kľúčových slov,
    2. vygenerovanie asociačných pravidiel spĺňajúcich minimálne kritériá.

Pravdepodobnostné modely

Pravdepodobnostný model uvažuje závislosti medzi slovami v dokumente a používa váhy slov zadaných v dopyte používateľa. Model je založený na dvoch hlavných parametroch – P_{rel} *a* P_{irel}, pravdepodobnosť relevantnosti a irelevantnosti dokumentu k dopytu, ktoré sa počítajú pomocou pravdepodobnostných váh slov a prítomnosti slov v dokumente. Pravdepodobnostný prístup vyžaduje trénovaciu množinu dokumentov, ktorú získa od používateľa tak, že používateľ poskytne rozhodnutie o relevantnosti dokumentu vzhľadom na výsledok dopytu.

Predpokladáme, že v množine máme N dokumentov, z ktorých je R relevantných pre používateľský dopyt, R_t relevantných dokumentov obsahuje slovo t a t sa vyskytuje v f_t dokumentoch. Potom môžeme pre každé slovo t vypočítať dve podmienené pravdepodobnosti:

P (t je v dokumente | dokument je relevantný) = R_t / R ;

P (t je v dokumente | dokument je irelevantný) = (f_t - R_t) / (N - R).

Z týchto odhadov sa pomocou ***Bayesovej teorémy*** odvodí váha slova t ako:

$$w_t = \log \frac{R_t / (R - R_t)}{(f_t - R_t) / [N - f_t - (R - R_t)]} \qquad (3)$$

Čitateľ (menovateľ) zlomku vyjadruje pravdepodobnosť, že slovo t sa nachádza v relevantnom (irelevantnom) dokumente. Váha $w_t > 0$ indikuje relevantnosť dokumentu k dopytu za podmienky, že sa slovo t vyskytuje v dokumente. Váha menšia ako nula indikuje jeho irelevantnosť.

Porovnanie modelov

Nasledujúca tabuľka (tab. 1) slúži pre porovnanie 3 klasických modelov, slúžiacich na reprezentáciu dokumentov.

Model	Výhody	Nevýhody
Boolovský	• jasný formalizmus • jednoduchosť	• presná zhoda výskytu termov otázky v dokumente môže viesť k príliš veľkému alebo naopak príliš malému počtu dokum. v odpovedi • dokumenty nemožno usporiadať podľa stupňa relevancie k otázke • neberie sa do úvahy frekvencia výskytu jednotlivých termov otázky v dokumente

Vektorový	• schéma váženia termov podľa frekvencie ich výskytu zvyšuje výkonnosť vyhľadávania • vyhľadá aj dokumenty, ktoré len čiastočne vyhovujú zadanej otázke usporiadania nájdených dokumentov podľa stupňa ich relevancie	• predpoklad nezávislosti indexových termov síce neplatí, ale prakticky ide väčšinou iba o lokálne závislosti malých skupín termov
Pravdepodobnostný	• usporiadanie nájdených dokumentov podľa pravdep. ich relevancie k otázke	• nutnosť počiatočného odhadu niektorých pravdepodobností • neberie sa do úvahy frekvencia výskytu jednotlivých termov otázky v dokumente • predpoklad nezávislosti indexových termov síce neplatí, ale prakticky ide väčšinou iba o lokálne závislosti malých skupín termov

Tab. 1. Porovnanie modelov

1.4 Vyhľadávanie informácií v dokumentoch, indexovanie

Keďže sekvenčné vyhľadávanie slov a fráz v texte je pri objemnej databáze veľmi neefektívne, vznikali metódy, ako hľadanie urýchliť. Zistilo sa, že je dobré vytvárať indexy slov, ktoré sa v dokumente vyskytujú. Indexovanie dokumentov značne urýchľuje vyhľadanie požadovaných slov v dokumente. Index neobsahuje všetky slová, ktoré sa v dokumente vyskytujú. Takzvané „stop slová“, ktoré sú v jazyku najčastejšie používané a/alebo nevytvárajú obraz o obsahu dokumentu sú z indexu vynechané. Patria tu napríklad slová *a, alebo, pre, od, do, iné*. Indexy majú zvyčajne veľkosť 20-30% z veľkosti dokumentu.

Inverzný index

Najbežnejšia metóda indexovania je tzv. invertované indexovanie [1]. Je to metóda, pri ktorej sa postupne vykonávajú tieto kroky:

- dokument sa rozdelí na slová
- vytvorí sa zoznam týchto slov
- ku každému slovu sa priradí zoznam pozícií jeho výskytov v dokumente

Pozície slov v takomto indexe môžu byť buď znakové, alebo slovné. Slovné pozície sú výhodnejšie pri vyhľadávaní frázy s približnou vzdialenosťou slov. Aby sme vedeli rozoznať, v ktorom dokumente a na akej pozícii sa hľadané slovo nachádza, musí mať slovo priradený jednoznačný identifikátor dokumentu a pozíciu slova. Pokiaľ index neobsahuje slovnú pozíciu, trvá vyhľadávanie frázy oveľa dlhšie.

Znakové pozície sú užitočné pri nachádzaní textu v dokumente. Inak je potrebné prehľadať celý dokument a nájsť hľadané slová. Je potrebné zvážiť, či potrebujeme obidva druhy pozícií a či sme ochotní zväčšiť tak výsledný indexový súbor.

Tretím parametrom môže byť číslo bloku. Dokumenty môžu byť delené na bloky s pevnou dĺžkou (10,64 alebo 256 tisíc znakov/slov). Použitím takýchto blokov sa veľkosť indexu zmenšuje.

Hľadanie v indexoch

Cieľom indexovania je teda zvýšenie rýchlosti vyhľadávania v dokumente. Vyhľadávanie môžeme rozdeliť na tri základné kroky:

1. Rozdelenie hľadanej požiadavky na slová a vyhľadanie každého slova v indexe.
2. Získanie zoznamu dokumentov, v ktorých sa slovo nachádza.
3. Skombinovanie zoznamov dokumentov podľa operátorov použitých v požiadavke.

V prvom kroku sa z požiadavky odstránia „stop slová". Je možné, že ak sa požiadavka skladá iba z takýchto slov, po prvom kroku nemáme čo hľadať. V druhom kroku sa vytvoria zoznamy dokumentov prislúchajúce jednotlivým slovám. Podľa operátorov v požiadavke tieto zoznamy skombinujeme a dostaneme výsledný zoznam hľadaných dokumentov.

Príklad
Slová, ktoré hľadáme: *Jablko, Hruška*

Zoznamy dokumentov, ktoré obsahujú jednotlivé hľadané slová:
Jablko 21,68,125,348
Hruška 25,68,97,125,336,412

Výsledný zoznam dokumentov, ktoré obsahujú hľadané slová:
Pri použití operátora AND: 68,125
Pri použití operátora OR: 21,25,68,125,336,348,412

Správnosť a priepustnosť

Dve základné metriky pre vyhľadávanie slov v dokumente sú:

správnosť - pomer nájdených relevantných dokumentov ku všetkým nájdeným dokumentom

priepustnosť - pomer nájdených relevantných dokumentov ku všetkým relevantným dokumentom v báze dokumentov

Príklad:
Máme daný stav znázornený v tabuľke:

	Relevantné	Irelevantné	Spolu
Nájdené	40	80	120
Nenájdené	10	870	880
Spolu	50	950	1000

Správnosť je 40/120 = 0,33
Priepustnosť je 40/50 = 0,8

Ak by sme chceli zvýšiť správnosť z 0,33 na 0,5, upravili by sme podmienky vyhľadávania tak, aby sa znížil počet nájdených dokumentov zo 120 na 60. Tým by sa znížil počet relevantných nájdených dokumentov z 40 na 30.

Dostávame:
Správnosť 30/60 = 0,5
Priepustnosť 30/50 = 0,6

Pri tvorení podmienok vyhľadávania musíme určiť pomer medzi správnosťou a priepustnosťou tak, aby sa nám nestalo že nenájdeme veľa relevantných dokumentov alebo nájdeme veľa irelevantných.

Lucene.Net – systém pre indexovanie a vyhľadávanie

Lucene.Net je fulltextový indexujúci a vyhľadávací systém. Bol vyvinutý z projektu Jakarta Lucene implementovaného v Jave a je určený pre jazyk C# na platforme .Net. Poskytuje výkonné funkcie cez jednoduché používateľské rozhranie. Je to Open Source projekt a je šírený pod licenciou (Apache Software License 2.0). Okrem oficiálneho názvu Lucene.Net sa používajú aj DotLucene, dotLucene, Lucene.NET. V tejto dokumentácii budeme tento systém nazývať dotLucene.

Charakteristiky dotLucene [1,11]

- Použiteľnosť v ASP.NET, Win Forms alebo konzolových aplikáciách
- Veľmi dobrý výkon
- Klasifikované výsledky hľadania
- Možnosť zvýrazniť hľadaný výraz vo výsledkoch hľadania
- Prehľadávanie štruktúrovaných aj neštruktúrovaných dát
- Vyhľadávanie v metadátach (podľa dátumu ,hľadanie vo voliteľných poliach indexu a iných)
- Veľkosť indexu je približne 20-30% z veľkosti indexovaného textu
- Vie uchovávať indexované celé dokumenty
- Čistý a jednoduchý kód pod platformou .NET v jednej knižnici (244 KB)
- Licencia Apache Software License 2.0, ktorá umožňuje komerčné aj Open Source používanie
- Lokalizovateľné (Angličtina, Čeština, Francúzština a iné)
- Rozšíriteľnosť (voľný zdrojový kód)
- Možnosť dopĺňať index (pri pridávaní nových dokumentov nie je nutné vytvoriť celý index nanovo)
- Viacindexové vyhľadávanie so spojením výsledkov
- Možnosť indexovať rôzne typy dokumentov (indexovať sa dá akýkoľvek dokument, ktorý sa dá previesť na text)
- Možnosť indexovať dokumenty z rôznych druhov zdrojov (web, databáza, lokálne uložené dokumenty)
- Súčasne je možné spustiť optimalizovanie indexu a vyhľadávanie
- Indexy sú kompatibilné s Lucene pre iné platformy

Hlavné funkcie dotLucene

Nad knižnicou dotLucene boli vytvorené jednoduché nástroje na sprístupnenie poskytovaných funkcií [11]. Popíšme si indexovanie dokumentov a vyhľadávanie kľúčových slov.

Indexovanie

dotLucene poskytuje na indexovanie dokumentov buď triedu, ktorú je možné používať v zdrojovom kóde, alebo jednoduchú konzolovú aplikáciu konfigurovateľnú pomocou XML súboru. Parametre v konfiguračnom súbore sú nasledovné:

- Meno výsledného indexu a cesta k adresáru, v ktorom bude uložený
<index name="index_A" indexFolderUrl="C:\TestIndexA">

- Indexované atribúty
<field name="Nazov" isStored="true" isIndexed="true" isTokenised="false" />

Jednotlivé atribúty majú nasledujúci význam:

- **isStored** - určuje, či bude daný atribút ukladaný do indexu (kedykoľvek sa dá k nemu pristúpiť - je to vhodné pre kratšie texty ako meno autora alebo názov)

- **isIndexed** - určuje, či sa bude dať podľa tohto atribútu vyhľadávať
- **isTokenised** - nastavením toho parametra sa indexovaná časť pred indexovaním rozdelí na slová

- **Zdroj dát** (databáza, súbor uložený na internetovej adrese alebo lokálnom súborovom systéme)

dotLucene vytvorí index a uloží ho v špecifikovanom adresári, kde je možné k nemu pristupovať a predkladať mu požiadavky na vyhľadávanie v dokumentoch.

Vyhľadávanie

Pri vyhľadávaní dotLucene-u predložíme kľúčové slová, ktoré v indexe vyhľadá. Ako výstup dostaneme zoznam identifikátorov dokumentov, v ktorých sa zadané slová nachádzajú. Taktiež dotLucene vyhodnotí relevanciu nájdených dokumentov.

Uveďme niekoľko základných požiadaviek pre vyhľadávanie dokumentov, v ktorých sa vyskytujú kľúčové slová v istých vzťahoch:

Požiadavka	**Príklad**	**Význam**
Slovo	jablko	Hľadá slovo „jablko“
Výraz	“zelené jablko“	Hľadá výraz „zelené jablko“
V poli	ovocie: jablko	Hľadá slovo „jablko“ v poli „ovocie“
Nahradenie jedného znaku	jablk?	Hľadá slová začínajúce na „jablk“ a končiace ľubovoľným znakom (napr. jablko, jablká)
Nahradenie viac znakov	jablk*	Hľadá slová začínajúce na „jablk“, za ktorými nasleduje ľubovoľný reťazec (napr. jablkový)
Operátor OR	jablko OR med	Alebo – nájde dokumenty obsahujúce slovo „jablko“ alebo „med“
Operátor AND	jablko AND med	A – nájde dokumenty obsahujúce slovo „jablko“ aj slovo „med“
Operátor NOT	NOT jablko	Nájde dokumenty, v ktorých sa nevyskytuje slovo „jablko“
Rozsah	autor: Berg TO Roy	Nájde dokumenty od autorov, ktorí sa abecedne nachádzajú medzi Berg-om a Roy-om

Tab. 2. Porovnanie modelov

Spracovanie iných ako textových formátov

dotLucene vie priamo spracovávať iba textové formáty a ak chceme spracovávať iné typy súborov, musíme ich previesť do textového formátu [5] alebo ich indexovať pomocou nejakej služby.

Jedným z dostupných nástrojov na indexovanie iných formátov je Seekafile Server [10], ktorý dokáže vytvoriť index z dokumentov vo formáte DOC, PDF, XLS, PPT, RTF, HTML, TXT, XML.

Ďalšou možnosťou je pretransformovať dokument do formátu XML, ktorý je možné indexovať ako je spomenuté v predchádzajúcej časti. Dokumentom, ktoré sa členia na kapitoly, obsahujú nejaké špecifické pole (napr. autor, názov) a pod., môžeme vymyslieť štruktúru, ktorá sa dá previesť do formátu XML.

Použitie dotLucene v našom projekte

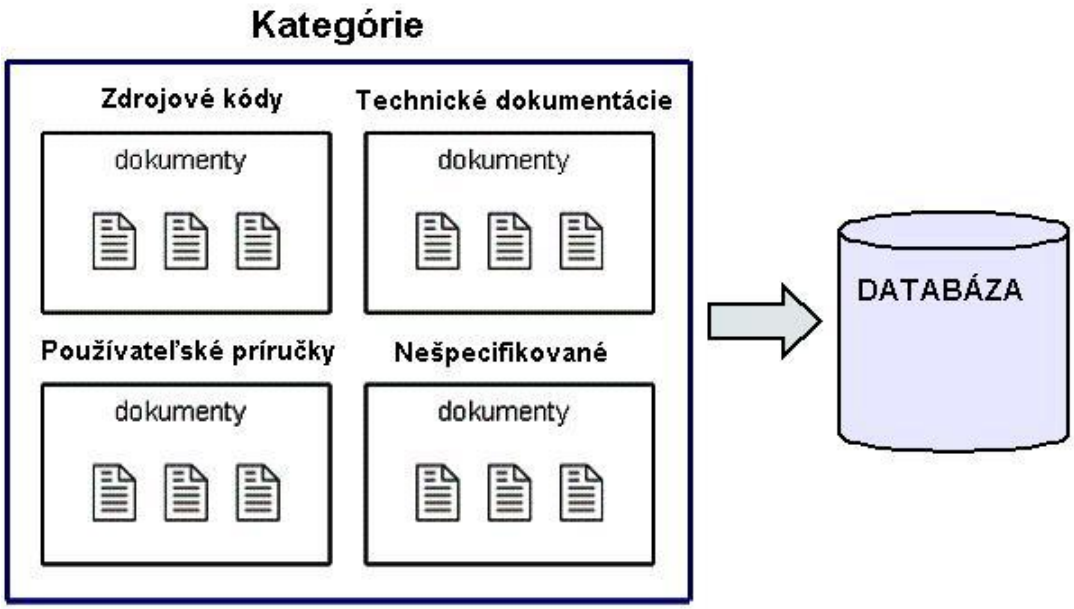
Tento vyhľadávací systém nám ponúka všetky požadované funkcionality (indexácia a vyhľadávanie v rámci databázy alebo lokálnych zdrojov), je výkonný, dá sa používať ako Open Source a ku knižnici je dostupná kompletná API dokumentácia. Preto sa javí ako veľmi dobrý nástroj pre indexovanie a vyhľadávanie v databáze dokumentov a znalostí v našom projekte.

1.5 Kategórie dokumentov

Pri dnešných kapacitných možnostiach databáz je možné uchovávať nesmierne množstvo dokumentov rôzneho druhu. Ak sa používateľ snaží vyhľadať informácie v podobe určitého dokumentu, je veľmi dôležité, aké vyhľadané typy dokumentov mu systém predostrie. Ľahko sa dá predstaviť situácia, kedy by používateľ hľadal informácie podľa určitej skupiny kľúčových slov v dokumente a systém by mu ponúkol dokument, ktorý by síce dané slová obsahoval, ale v úplne inom kontexte.

Týmto problémom sa v súčasnej dobe výskum zaoberá veľmi intenzívne. Viacznačnosť jazyka ako aj slabá reprezentácia znalostí zatiaľ neumožňuje tento problém vyriešiť úplne. Snahou tohto projektu by malo byť eliminovať tento problém čo najviac a poskytovať používateľovi dokumenty poskytujúce informácie, ktoré si z čo najväčšou pravdepodobnosťou vyžiadal.

Aby sa predišlo predkladaniu navzájom nesúvisiacich dokumentov používateľovi, je vhodné dokumenty roztriediť do jednotlivých skupín, čiže kategórií dokumentov. Pod pojmom kategória dokumentu treba rozumieť doplňujúcu informáciu, ktorá prezradí, či sa jedná napríklad o používateľskú príručku, technickú dokumentáciu, dokument zdrojového kódu a podobne (obr. 6). Táto informácia je ďalším kritériom, ktoré môže mať veľký význam pri vyhľadávaní dokumentov. Používateľovi sa budú predkladať iba tie dokumenty, ktoré spolu súvisia čo najviac.



Obr. 6: Schéma rozdelenia dokumentov do kategórií

Matica mier podobnosti medzi kategóriami

Jednotlivé kategórie dokumentov je potrebné zadefinovať a uchovávať medzi nimi určité váhy súvislosti, ktoré budú symbolizovať mieru podobnosti kategórií. Možný spôsob uchovávania váh podobnosti kategórií zobrazuje nasledovná tabuľka.

	Používateľská príručka	Zdrojový kód C#	Zdrojový kód Java	Technická dokumentácia	Nešpecifikovaná kategória
Používateľská príručka	1.0	0.1	0.1	0.3	0.5
Zdrojový kód C#	0.1	1.0	0.75	0.75	0.5
Zdrojový kód Java	0.1	0.75	1.0	0.75	0.5
Technická dokumentácia	0.3	0.75	0.75	1.0	0.5
Nešpecifikovaná kategória	0.5	0.5	0.5	0.5	0.5

Tab. 3. Matica mier podobnosti medzi kategóriami dokumentov

Údaje o vzťahoch medzi jednotlivými typmi dokumentov je vhodné uchovávať v matici, ktorá pomocou indexov zabezpečí rýchly prístup k údajom (tab. 3). Miera podobnosti medzi dvomi kategóriami je vždy rovnaká, matica bude teda symetrická. Medzi rovnakými kategóriami je miera podobnosti 1.0, čiže najvyššia možná. Ostatné miery podobnosti sú v tabuľke zadané iba približne, je ich možné nastaviť a prispôsobiť podľa potreby. Rovnako aj uvedené kategórie a ich počet je len ilustračný.

Ak sa používateľ snaží nájsť iba určitú kategóriu dokumentov, pochopiteľne najviac uprednostňované budú dokumenty práve z tejto kategórie. Na ďalších miestach sa však môžu objaviť aj dokumenty, ktoré sú z inej kategórie blízkej jemu vybranej a obsahujú zadané kľúčové slová. V prípade, že používateľ kategóriu bližšie nešpecifikuje, výber dokumentov sa môže uskutočňovať podľa jeho profilu (analytik, programátor a pod.), prípadne ním často používaných dokumentov. Vtedy sa opäť môžu použiť vzťahy medzi kategóriami.

Maticu, uchovávajúcu miery podobnosti medzi kategóriami, nie je možné vytvárať automatizovane a to z dôvodu, že pri pridaní novej kategórie, počítač nedokáže rozhodnúť, ako sa podobá s ostatnými. Dokonca aj pre rôznych jednotlivcov môžu mať podobnosti kategórií rôznu váhu. Z týchto dôvodov je nutné v prípade novej kategórie jednotlivé miery manuálne zadefinovať človekom (administrátorom) podľa potreby

Automatické zisťovanie kategórie dokumentu

Osobitným problémom je rozpoznanie kategórie dokumentu v prípade, že nie je definovaná. Vtedy by bolo vhodné, aby systém dokázal typ dokumentu zistiť samostatne. To však prestavuje náročnú činnosť. Aby to systém mohol urobiť, musel by vedieť rozoznať kategóriu dokumentu podľa určitých pravidiel, alebo porovnávať dokument z ostatnými a z určitou pravdepodobnosťou kategóriu danému dokumentu priradiť. Je dôležité, aby systém priraďoval kategóriu iba s určitou pravdepodobnosťou, prípadne aby prideľoval miery podobnosti s ostatnými kategóriami. Ak by tak neurobil, riskovalo by sa priradenie zlej kategórie dokumentom, čo by mohlo spôsobiť nežiadané následky.

Spôsobov, akým je možné kategórie zisťovať, je mnoho. Jeden z nich by mohol byť postavený na určitých formách vzorov, podľa ktorých by sa dokumenty kategorizovali. Vytvoriť takýto vzor je veľmi náročné, nakoľko dokumenty spadajúce pod určitú kategóriu sa môžu diametrálne odlišovať. Rovnako by sa na tieto účely dala využiť neurónová sieť, ktorá by sa po predložení veľkého množstva rôznych dokumentov istej kategórie natrénovala a jej výstupom by bola pravdepodobnosť spadania dokumentu pod danú kategóriu. Stále však ostáva problém s dokumentom, ktorý by nespadal ani pod jeden zo spomínaných vzorov kategórie. Systém by mu samozrejme musel priradiť nesprávnu kategóriu, na základe najväčšej podobnosti. Iný spôsob, ktorým by sa dal tento problém obísť, je pri každom dokumente udržiavať všetky

pravdepodobnosti, s ktorými dokument spadá do jednotlivých kategórií. Na základe nich by bolo možné vyrátavať vzťahy medzi dokumentmi a predkladať ich používateľovi. Netreba zabúdať na to, že možných kategórií môže byť veľmi veľa a vytváranie pravdepodobnosti podobností je opäť nesmierne náročné.

Automatické zisťovanie kategórie dokumentu je každopádne zložitá činnosť, ktorej úspešná realizácia by však projekt veľmi obohatila. Na automatické zisťovanie by bolo vhodné vytvoriť samostatný nezávislý modul, ktorý by sa dal kedykoľvek vylepšiť, či vymeniť.

1.6 Modelovanie používateľov v znalostnom manažmente

Modelovanie používateľov má svoj vplyv na informačné systémy hlavne z hľadiska nastavenia vlastností a správania sa systému. Vychádzame z faktu, že každý používateľ je jedinečný, má iné očakávania od spôsobu reprezentácie znalostí a od ich parametrov vyhľadávania.
Navyše systémy znalostného manažmentu môžu byť použité aj pre vytváranie nových znalostí (pridanie nových dokumentov), ich prenos a zdieľanie. Systémy takého typu neslúžia iba pre čiru administráciu elektronickej informácie, ale sú schopné učiť sa, napr. procesy používateľov, vzťahy medzi nimi alebo poskytujú štatistiky a rôzne pravidlá pri vyhľadávaní znalostí.

Samotný model pre používateľov je nutný z dvoch dôvodov:

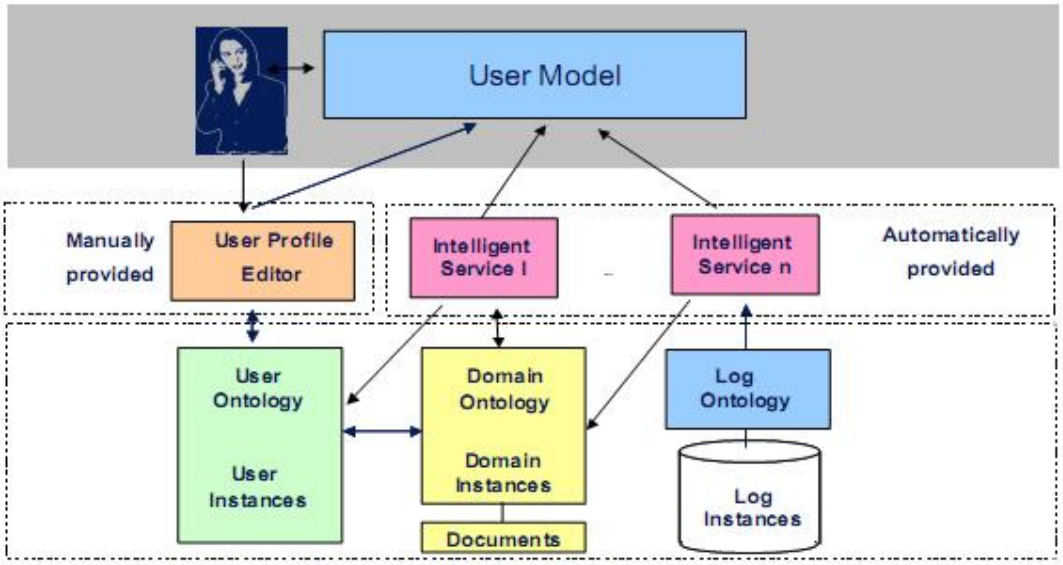
1. Rozdiely medzi jednotlivými individuálnymi potrebami
2. Heterogenita medzi rôznymi skupinami používateľov

Architektúra používateľského modelu založená na ontológiách

Používateľský model je vlastne samostatný modul, ktorý má vlastnosti univerzálneho komponentu

- Adaptácia – podpora konfigurovateľnosti aplikácie na základe charakteristík používateľov, ktoré sú opísané modelom
- Interoperabilita - výmena informácií medzi ďalšími komponentmi
- Znovupoužiteľnosť – opakované použitie v iných systémoch

Bez ujmu na všeobecnosť používateľský model rozdeľujeme na dve časti: manuálne definovaná(explicitná) a automatická časť(implicitná) zabezpečená inteligentnými službami. Manuálna časť je založená na profiloch používateľov, ktoré sa získavajú formou dotazníkov, formulárov. Automatická časť je založená na ďalšom modeli, ktorý obsahuje údaje ktoré sa zbierajú počas interakcie používateľa so systémom, napr. aké typy dokumentov vyhľadával, koľkokrát si ich zobrazil, ktoré navrhnuté dokumenty akceptoval atď. Inteligentné služby majú za úlohu aktualizovať model používateľa a poskytovať služby vzťahujúce sa na práve aktuálneho používateľa. Tu spomenutý koncept integrovania používateľského modelu do systému je zobrazený na obr. 7.[9]



Obr. 7. Model používateľa použitím ontológií

Na obr. 7 sú zobrazené aj tri rôzne ontológie:

- Ontológia používateľov – definuje charakteristiky používateľov a vzťahy medzi nimi
- Ontológia domény – opisuje doménu a koncepty aplikácie
- Ontológia záznamov – definuje sémantiku používateľa so systémom, dáta záznamov sa získavajú monitorovaním tejto interakcie

Pri architektúre sa spomínajú ontológie, ktoré slúžia hlavne pre definovanie domén a vzťahov medzi nimi. Máme však rôzne možnosti ako tieto informácie uchovávať a spracovať. Pri implementácii sa dajú použiť tieto spôsoby:

- Jazyk založený na XML
- Relačná databáza
- Sémanticky silnejšie prostriedky(RDF/OWL)
- Vytvorenými procedúrami rôznych programovacích jazykov(objektovo orientované, procedurálne, logické, ...)

Jazyk založený na XML nám dáva voľnú ruku pri definovaní modelu, čo sa javí ako veľká výhoda, lenže nezabezpečuje univerzálnosť s ďalšími systémami. Relačná databáza už v sebe zahŕňa opis charakteristík a vzťahov medzi nimi, navyše poskytuje prostriedky pre bezpečnosť dát. Databázu v podstate potrebujeme pri ktorejkoľvek reprezentácii modelu.
Ak máme dobre definované charakteristiky používateľov a záznamov interakcie so systémom, môžeme použiť jazyky pre definovanie ontológií. Tieto jazyky sú vytvorené pre definovanie vlastností domén, vzťahov medzi nimi, podmienok a rôznych reštrikcií.

Manuálne definovaná časť modelu používateľa

Na začiatku práce používateľa so systémom ešte nemáme o ňom žiadne informácie, ktoré získame inteligentnými službami. Je žiaduce, aby používateľ poskytol niektoré základné údaje o sebe.

Pretože predpokladáme, že náš systém bude súčasťou nejakého podnikového (enterprise) systému, medzi najdôležitejšie informácie o používateľov by sme zaradili:

- Osobné údaje
- Jeho pozíciu v organizačnej štruktúre
- Aktuálny projekt/y, ktorým/i sa zaoberá
- Skúsenosti a vedomosti
- Oblasti záujmov

Ďalšie údaje, ktoré by sa uchovávali pre jednotlivcov, by mohli byť preferencie týkajúce sa grafického zobrazenia a rozhrania systému. To však neuvažujeme v našom projekte za také dôležité.

Explicitne zadané údaje používateľom sa prejavia pri použití systému - hľadaní znalostí a to hlavne pri ponúknutí alternatívnych možností k výsledkom hľadania získaným iba pomocou kľúčových slov. Z implementačného hľadiska sa táto súčasť najľahšie realizuje pomocou formulárov. Pričom používateľ sa najprv zaregistruje do systému, vytvorí sa mu profil, ktorý si neskôr môže aj modifikovať. Tento spôsob autentifikácie zaručí aj istú úroveň bezpečnosti.

Vytváranie záznamov na základe interakcie používateľa so systémom

Systém znalostného manažmentu sleduje proces jeho požívania jednotlivými používateľmi. Jadro analytického nástroja čerpá z rôznych oblastí výskumu: strojové učenie a štatistika, teóriu agentov a efektívne a rýchle manipulácie s dátami. Na základe poznatkov ako systém pracuje, aký je tok údajov a aké sú dostupné procesy a ako sa pritom používateľ správa, sa vytvorí ontológia záznamov.

Vlastnosťami správania sa môžu byť napr. úroveň aktivity, typ aktivity, zdieľané údaje a štatistické informácie získané pri reakciách na výsledky vyhľadávania.

Inteligentné služby majú zabezpečiť:
- Ponúknuť znalosti podobné aktuálne prezentovanému
- Vybrať znalosti, ktoré zapadajú do oblasti záujmu používateľa, alebo ktoré sú k nemu príbuzné
- Odporučiť členov spoločnosti s podobnými preferenciami

Ak chceme nájsť používateľov, ktorí sú sebe blízky, musíme porovnať ich oblasti záujmu a porovnať ich charakteristiky správania sa. Predpokladajme, že máme dvoch používateľov A a B, skúmame ich „príbuznosť“. Tento vzťah budeme nazývať vzdialenosťou a označovať ako D(A,B). Nech A_d je množina dokumentov, ktoré zobrazil používateľ A pri vyhľadávaní, analogicky aj B_d. Množina $C_{A,B} = A_d \cap B_d = \{Z_1, Z_2, \dots Z_n\}$ je množina dokumentov Z_i, ktoré si vybral aj A aj B. Nech $V_{X,Y}$ označuje hodnotenie dokumentu Y používateľom X, pričom hodnotenie sa zvyšuje pri každom zobrazení daného dokumentu. Vzdialenosť medzi A a B bude vyjadrený vzťahom (4):

$$D(A,B) = \sqrt{\sum_{i}^{C(A,B)} \left(V_{A,Z_i} - V_{B,Z_i}\right)^2} \qquad (4)$$

Teda používatelia A a B sú si blízky ak vyberali podobné dokumenty s približne rovnakou frekvenciou. Čím nižšiu hodnotu dostaneme pri vypočítaní vzdialenosti, tým sú bližší A a B. Najnižšou hodnotou je nula.
Vlastnosť (4) sa dá dobre využiť pri predpovedaní, či sa bude páčiť dokument používateľovi B, na ktorý ešte nehlasoval, ale A už áno. Ak je vzdialenosť používateľov A a B malá, pravdepodobne pri vysokom hodnotení dokumentu používateľom A je logické ponúknuť ju aj druhému. Vzdialenosť dvoch používateľov je však relevantná iba pri vyššom počte prezretých dokumentov.

Vyhľadanie podobných dokumentov je založená na týchto vstupných parametroch: kľúčové slová hľadania, charakteristika používateľa, systémom vytvorené záznamy o jeho aktivite a činnostiach. Najlepším riešením by bolo dokumenty rozčleniť na rôzne sekcie/kapitoly a potom by sa hľadali kapitoly s takým istým typom. Podobne sa dá jednotlivé dokumenty priradiť projektom, čím zvyšujeme organizáciu v báze znalostí.

Teda sa ľahšie nájdu podobné dokumenty, napr. s rovnakými typmi kapitol, alebo na základe používateľa, ktorý pracuje na podobnom oddelení podniku atď.

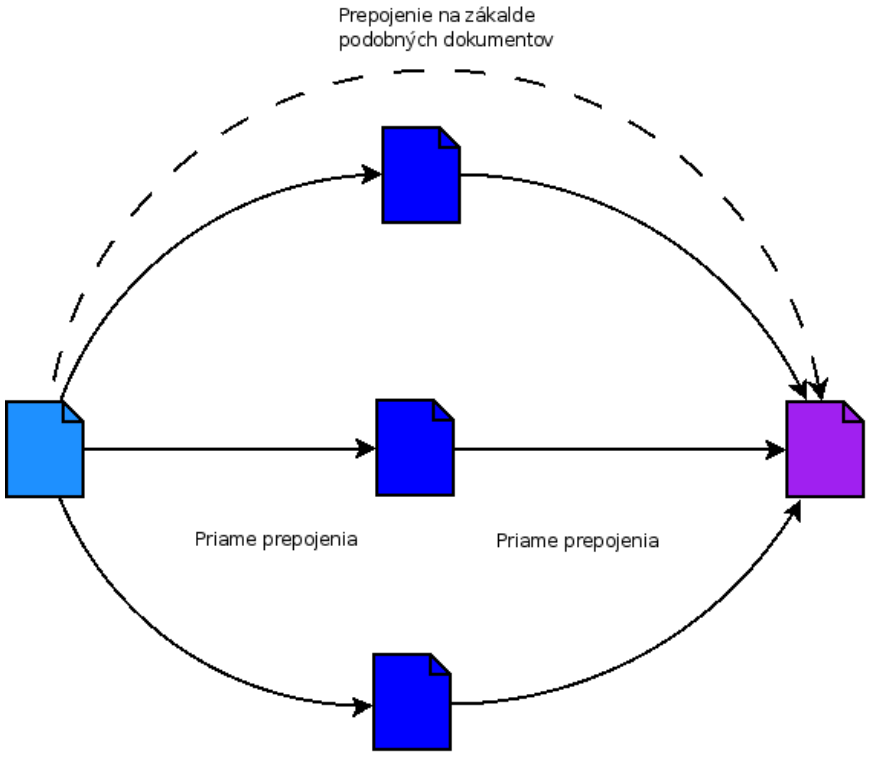
1.7 Určenie relevancie dokumentov založené na topológii grafu

Pri určovaní relevancie dokumentu môžeme vychádzať z niekoľkých možných princípov. Jeden z týchto prístupov vychádza z predpokladu, že dokumenty sú navzájom poprepájané odkazmi a to buď priamymi alebo nepriamymi. Je teda možné nájsť dokument ktorý súvisí s iným dokumentom na základe týchto prepojení.

Výhodou využitia takýchto prepojení je fakt že sa nám môže podariť nájsť dokument ktorý síce priamo nesúvisí s nami vyhľadávaným výrazom, ale odkazujú sa naň dokumenty ktoré s našou požiadavkou priamo súvisia a teda je veľká pravdepodobnosť, že takto nájdený dokument bude relevantný

Princíp spočíva v predstavení si problémového priestoru ako grafu, v ktorom dokumenty tvoria uzly a hrany sú tvorené prepojeniami medzi týmito dokumentmi. Každý z vrcholov je ohodnotený určitým hodnotením, ktoré predstavuje jeho dobrú povesť alebo jeho relevanciu. Čím je táto hodnota vyššia tým je dokument pre nás relevantnejší.

V nasledujúcom texte budú popísané spôsoby ako vytvoriť takýto graf a taktiež možnosti jeho využitia pre potreby nášho projektu.



Obr. 8. Prepojenie dokumentov

Vytváranie grafu

Vytvorenie grafu reprezentujúceho väzby medzi dokumentmi je pravdepodobne najproblematickejšou časťou celého procesu. Obnáša totiž identifikáciu jednotlivých prepojení medzi dokumentmi a taktiež určenie typu týchto prepojení.

Identifikácia uzlov

V princípe existujú dva spôsoby určenia uzlov.

- uzly tvoria dokumenty
- uzly tvoria kapitoly dokumentov

V prípade tvorenia uzlov dokumentmi sa nám značne zjednodušuje situácia pri určovaní spojení medzi nimi, no redukujú sa možnosti pre určenie relevancie jednotlivých spojení. Preto sa ako lepšie riešenie javí použiť ako uzly kapitoly.

Identifikácia prepojení

Prepojenia medzi dokumentmi môžeme klasifikovať do nasledujúcich kategórií.

- Priamy odkaz na dokument: Predstavuje cestu na diskovú jednotku alebo URL dokumentu s ktorým je daný dokument prepojený. Vzhľadom na typ (technická, používateľská dokumentácia, zdrojové kódy) dokumentov, ktorý je v našom projekte spracovávaný je pravdepodobnosť výskytu tohto prepojenia pomerne malá.
- Prepojenie na základe názvov dokumentov a kapitol: Jedná sa o prepojenie v ktorom jeden z dokumentov vo svojom tele obsahuje názov iného dokumentu alebo názov kapitoly obsiahnutej v inom dokumente. Tento typ prepojenia nám umožňuje identifikovať pomerne silne relevantné prepojenia medzi dvoma dokumentmi. Problém môže nastať v prípade negatívneho odkazu na názov (Tento dokument NEsúvisí s dokumentom XY. NErobte to ako je popísané v YZ). Pri použití tohto prepojenia je potrebné uchovávať názvy všetkých dokumentov a názvy všetkých ich kapitol a tie následne porovnávať s textom v dokumente.
- Prepojenie na základe kľúčových slov: Prepoja sa dokumenty, ktoré majú rovnaké kľúčové slová alebo aspoň vysoké percento rovnakých kľúčových slov. Vyžaduje sa vytvoriť zoznam kľúčových slov a porovnávať ho s kľúčovými slovami v danom dokumente.
- Prepojenie na základe výskytu slov: Prepoja sa dva dokumenty ktoré majú vysoký výskyt rovnakých slov.

Všetky tieto prepojenia dokážu určiť súvislosti medzi dvoma dokumentmi. Každému takémuto typu prepojenia môže byť priradená hodnota určujúca jeho dôležitosť a tak ovplyvniť výsledky určenie relevancie dokumentu.

Ďalším spôsobom delenia je delenie na základe toho, či sa jedná o prepojenie medzi kapitolou alebo celým dokumentom. V tomto prípade dokážeme identifikovať tieto druhy prepojení.

1. kapitola - kapitola
2. kapitola - dokument
3. dokument - kapitola
4. dokument - dokument
5. kapitola - kapitola (v rámci dokumentu)

Každému z týchto prepojení môžeme priradiť určitú hodnotu dôležitosti. Napríklad prepojenie medzi dvoma kapitolami môžeme považovať za dôležitejšie ako prepojenie medzi dvoma dokumentmi. Dôležitosť

prepojenia môžeme určovať na základe princípu, v ktorom prepojenie konkrétnejšieho(kapitola) má prednosť pred prepojením všeobecného (celý dokument). Je taktiež podstatné poznamenať že medzi sebou sú prepojené aj kapitoly v jednom dokumente. Toto prepojenie je rovnocenné prepojeniu typu kapitola - kapitola

Prehľadávajúce algoritmy

Pokiaľ existuje graf ktorý máme možnosť prehľadávať je potrebné vybrať vhodný algoritmus, ktorý dokáže určiť hodnoty relevancie jednotlivých uzlov. V našom prípade sa ako vhodný kandidáti javia algoritmy *Pagerank* a *NodeRank.*

PageRank
Algoritmus ohodnocovania vrcholov grafu *pagerank* sa zakladá na jednoduchom princípe, ktorý predpokladá že uzol grafu je tak kvalitný ako vrcholy ktoré sú s ním prepojené. Teda pokiaľ má uzol veľa prepojení s inými uzlami ktoré majú vysoké hodnotenie, tak aj on má vysoké hodnotenie. Princíp je zobrazený na obr. 9.

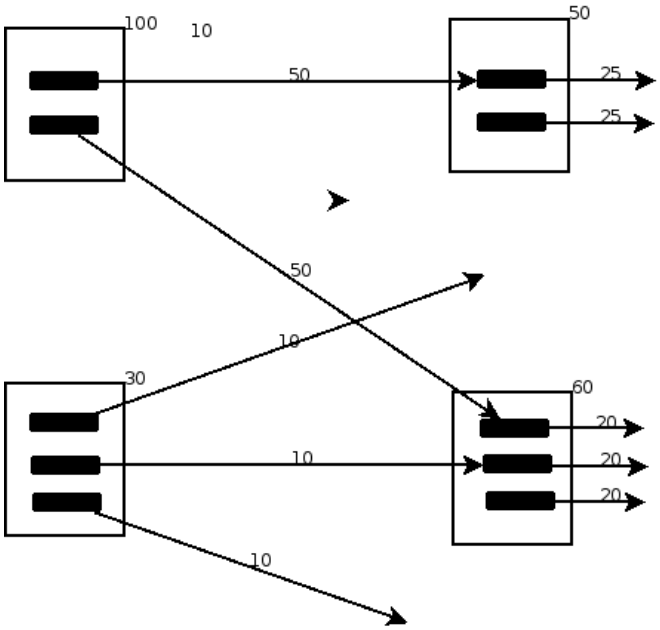
Pri postupnom prechode cez všetky uzly však môže dôjsť k situácii keď sú uzly opakovane ohodnocované a tak môže dôjsť k situácii keď hodnotenie niektorých uzlov rastie do nekonečna, a dôjde k takzvanému "zahlteniu ohodnotenia".

Pre odstránenie tohto prípadu je používaná stratégia náhodného chodca (*random walker*). Pri použití tejto stratégie sa prechádza grafom a s určitou pravdepodobnosťou sa preskočí do iného uzla. Týmto spôsobom dokážeme predísť zahlteniu.

NodeRank
Na rozdiel od algoritmu *PageRank* dokáže pracovať s grafom ktorého hrany sú ohodnotené. Pracuje na podobnom princípe ako *PageRank.* Teda posúva hodnotenie uzlu grafu do jeho nasledovníkov, no využíva pri tom aj hranu ktorá tieto dva uzly spája. Teda hodnotenie nasledovníka uzla je tvorené jednak predchodcom ohodnotením uzla samotného ako aj váhou hrany ktorá uzol a jeho nasledovníka spája. V prípade rozhodovania do ktorého ďalšieho uzlu sa pri prechode grafom vyberieme slúži váha prepojení ktoré z uzla vychádzajú.

V algoritme je taktiež použitá stratégia náhodného chodca. Táto stratégia je však modifikovaná tým spôsobom, že v prípade návštevy toho istého vrcholu niekoľkokrát po sebe, dostáva vrchol stále menší a menší prírastok ohodnotenia.

Pre potreby nášho projektu sa ako výhodnejšie javí použitie algoritmu *NodeRank*, ktorý berie do úvahy aj váhy prepojení medzi dokumentmi.



Obr. 9. Princíp ohodnocovania vrcholov algoritmom PageRank

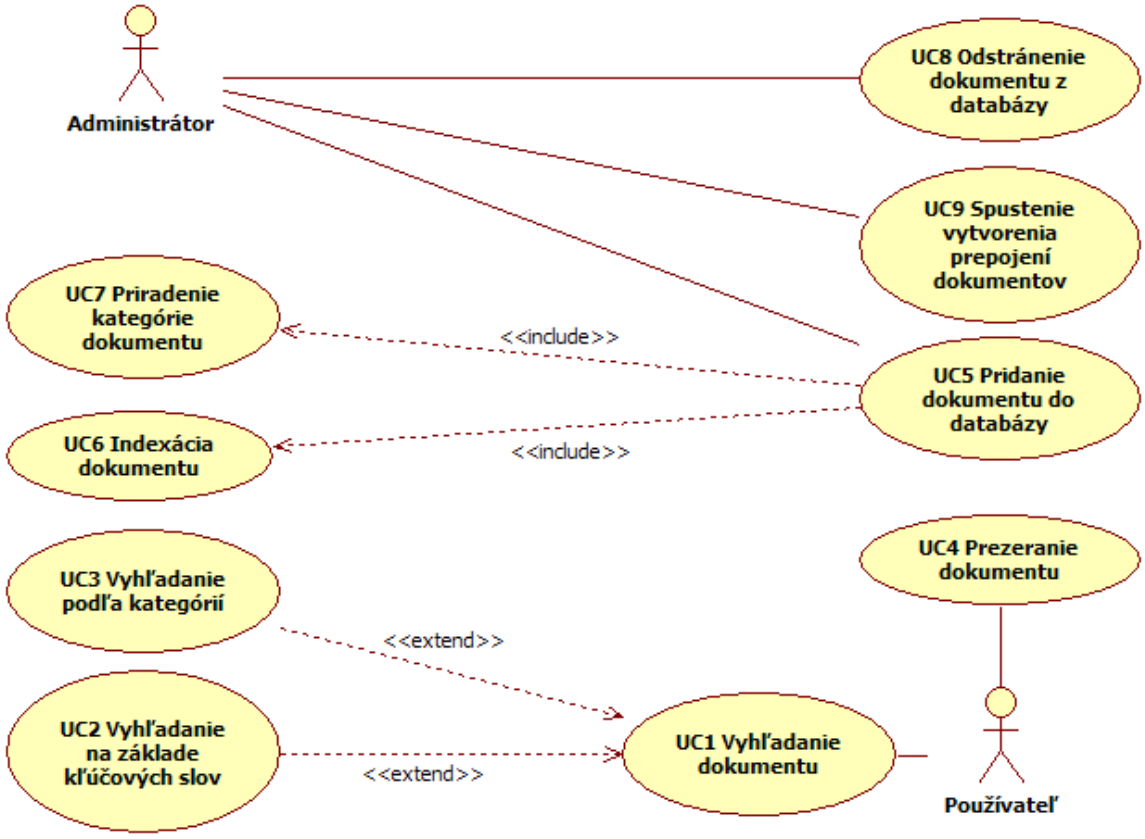
Vhodnosť algoritmov

Pre potreby nášho projektu sa ako vhodnejšie javí použitie algoritmu *NodeRank* vzhľadom na to že dokáže pracovať aj s váhami medzi jednotlivými uzlami v grafe. Je taktiež nutné využitie stratégie náhodného chodca kvôli predídeniu zahlteniu ohodnotenia.

2 Špecifikácia

2.1 Špecifikácia požadovaného riešenia

Na obr. 10 je diagram prípadov použitia, v ktorom sú znázornené funkcionality systému poskytnuté jednotlivým hráčom.



Obr. 10. Diagram prípadov použitia

2.2 Identifikácia hráčov

Systém bude používaný dvomi kategóriami hráčov.

Administrátor - bude spravovať databázu, bude do nej pridávať nové dokumenty, priraďovať kategóriu dokumentu, odstraňovať dokumenty. Po pridaní nových dokumentov do databázy je potrebné aktualizovať bázu znalostí, tak aby sa neporušila konzistencia dát. Keďže tieto operácie môžu byť náročné na čas a počas ich vykonávania sa nesmie s databázou pracovať, spúšťať ich bude administrátor (napr. v noci).

Používateľ - je mu umožnené vyhľadávať dokumenty v systéme. Môže zadať kľúčové slová alebo kategóriu dokumentu, podľa ktorých znalostný systém vyhodnotí vhodnosť dokumentov pre požiadavku používateľa. Následne mu systém poskytne zoznam kandidátov, ktorých vyhodnotil ako najvhodnejších a používateľ si môže podľa krátkej časti textu dokumentu

vybrať ten dokument, ktorý má tendenciu byť skutočne relevantným vzhľadom na jeho požiadavky.

Keďže jeden modul v systéme bude analyzovať aké dokumenty používateľ často hľadá, resp. používa, bude vedená evidencia používateľov. Ak sa používateľ prihlási ako už evidovaný používateľ, systém na základe podobnosti hľadaní a výberov dokumentov s inými používateľmi poskytne výbery iných používateľov, ktoré by mohli byť pre hľadajúceho zaujímavé. Ak sa používateľ nehlási ako evidovaný, táto funkcionalita systému bude nefunkčná.

2.3 Prípady použitia

Prípady použitia sú popísané v nasledujúcich tabuľkách.

Identifikátor	UC1		
Názov	Vyhľadanie dokumentu		
Opis	Používateľ vyhľadá dokument podľa svojich preferencií		
Priorita	vysoká	**Frekvencia**	Denne, rádovo stovky
Vstup. podm.	Požiadavka na vyhľadanie dokumentu		
Výstup. podm.	Zobrazenie výsledkov vyhľadávania		
Používatelia	Používateľ		
Základná postupnosť	**Krok**	**Činnosť**	
	1	Používateľ: Spustenie grafického rozhrania programu	
	2	Používateľ: Výber typu vyhľadávania	
	3	Používateľ: Zadanie kľúčových slov pre vyhľadávanie (UC2) a/alebo kategórie vyhľadávania(UC3)	
	4	Systém: Spracovanie požiadavky	
	5	Systém: Zobrazenie výsledkov vyhľadávania používateľovi	
Alternatívna postupnosť	**Krok**	**Činnosť**	
	-	Používateľ: Zrušenie vyhľadávania	

Identifikátor	UC2		
Názov	Vyhľadanie na základe kľúčových slov		
Opis	Používateľ vyhľadá dokument podľa kľúčových slov		
Priorita	vysoká	**Frekvencia**	Denne, rádovo stovky
Vstup. podm.	Zobrazenie formulára na zadanie kľúčových slov		
Výstup. podm.	Zobrazenie výsledkov vyhľadávania		
Používatelia	Používateľ		
Základná postupnosť	**Krok**	**Činnosť**	
	1	Používateľ: Zadanie kľúčových slov	
	2	Systém: Spracovanie požiadavky	
	3	Systém: Zobrazenie výsledkov vyhľadávania používateľovi	
Alternatívna postupnosť	**Krok**	**Činnosť**	
	-	Používateľ: Zrušenie vyhľadávania	

Identifikátor	UC3		
Názov	Vyhľadanie podľa kategórií		
Opis	Používateľ vyhľadá dokument podľa kategórií, do ktorých dokument patrí		
Priorita	vysoká	**Frekvencia**	Denne, rádovo stovky
Vstup. podm.	Zobrazenie formulára na zadanie kategórií		
Výstup. podm.	Zobrazenie výsledkov vyhľadávania		
Používatelia	Používateľ		
Základná	**Krok**	**Činnosť**	

postupnosť	1	Používateľ: Zadanie kľúčových slov
	2	Systém: Spracovanie požiadavky
	3	Systém: Zobrazenie výsledkov vyhľadávania používateľovi
Alternatívna postupnosť	**Krok**	**Činnosť**
	-	Používateľ: Zrušenie vyhľadávania

Identifikátor	UC4			
Názov	Prezeranie dokumentu			
Opis	Používateľ si prezrie dokument, ktorý mu poskytol systém vo výsledkoch			
Priorita	vysoká		**Frekvencia**	Denne, rádovo stovky
Vstup. podm.	Zobrazené výsledky vyhľadávania			
Výstup. podm.	Zobrazenie zvoleného dokumentu			
Používatelia	Používateľ			
Základná postupnosť	**Krok**	**Činnosť**		
	1	Systém: Zobrazenie výsledkov vyhľadávania		
	2	Používateľ: Výber dokumentu, ktorý chce používateľ prezerať		
	3	Systém: Zobrazenie zvoleného dokumentu		
Alternatívna postupnosť	**Krok**	**Činnosť**		
	-	Používateľ: Zrušenie zobrazovania výsledkov		

Identifikátor	UC5			
Názov	Pridanie dokumentu do databázy			
Opis	Administrátor pridá nový dokument do databázy			
Priorita	vysoká		**Frekvencia**	Mesačne, rádovo desiatky
Vstup. podm.	Nový dokument			
Výstup. podm.	V databáze je uložený nový dokument, jeho meno, metadáta.			
Používatelia	Administrátor			
Základná postupnosť	**Krok**	**Činnosť**		
	1	Administrátor: Požiadavka na systém, aby zobrazil formulár pre pridanie dokumentu do databázy		
	2	Systém: Zobrazí administrátorovi formulár na pridanie dokumentu do databázy		
	3	Administrátor: Zadá cestu k dokumentu		
	4	Systém: Uloží do databázy meno dokumentu, binárne dáta, a metadáta vo formáte XML (okrem iného musia byť úspešne vykonané UC6 a UC7)		
	5	Systém: Zobrazí administrátorovi správu o úspešnom vykonaní operácie		
Alternatívna postupnosť	**Krok**	**Činnosť**		
	-	Administrátor: Zrušenie pridávania dokumentu do databázy		
	4.a	Systém: Dokument nebude uložený do databázy		
	5.a	Systém: Vypísanie dôvodu neúspešného pridania dokumentu do databázy		

Identifikátor	UC6			
Názov	Indexácia dokumentu			
Opis	Systém pridelí dokumentu jedinečné identifikačné číslo a prepíše index			
Priorita	vysoká		**Frekvencia**	Mesačne, rádovo desiatky
Vstup. podm.	Existujúci dokument			
Výstup. podm.	Pridelený identifikátor dokumentu, prepísaný indexu			
Používatelia				
Základná	**Krok**	**Činnosť**		

postupnosť	1	Systém: Pridelenie identifikátora novému dokumentu
	2	Systém: Aktualizácia indexu
	3	Systém: Zobrazenie správy o úspešnosti vykonanej operácie

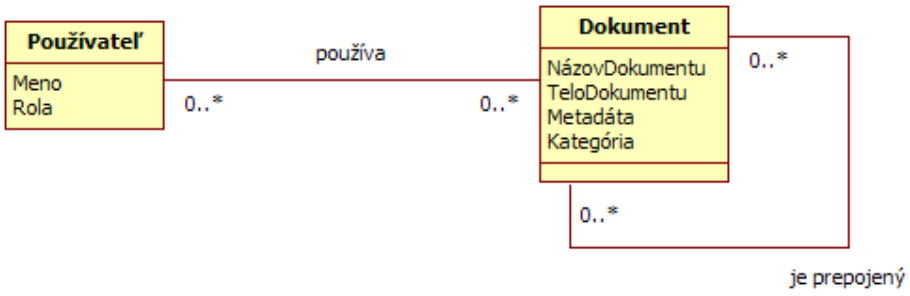
Identifikátor	UC7		
Názov	Priradenie kategórie dokumentu		
Opis	Administrátor		
Priorita	stredná	**Frekvencia**	Mesačne, rádovo desiatky
Vstup. podm.	Existujúci dokument		
Výstup. podm.	Priradená kategória dokumentu		
Používatelia	Administrátor		
Základná postupnosť	**Krok**	**Činnosť**	
	1	Systém: Poskytnutie rozhrania pre zadanie kategórie dokumentu	
	2	Administrátor: Zadanie kategórie (existuje aj kategória „bez kategórie“)	
	3	Systém: Priradenie zadanej kategórie dokumentu	
	4	Systém: Zobrazenie správy o úspešnosti vykonania operácie	
Alternatívna postupnosť	**Krok**	**Činnosť**	
	-	Administrátor: zrušenie priradenia kategórie (systém priradí kategóriu „bez kategórie“)	

Identifikátor	UC8		
Názov	Odstránenie dokumentu z databázy		
Opis	Administrátor		
Priorita	vysoká	**Frekvencia**	Ročne, rádovo jednotky
Vstup. podm.	Dokument uložený v databáze		
Výstup. podm.	Dokument sa v databáze ďalej nenachádza, je odstránený z indexu aj tabuliek jednotlivých modulov		
Používatelia	Administrátor		
Základná postupnosť	**Krok**	**Činnosť**	
	1	Systém: Poskytnutie rozhrania pre výber dokumentu na odstránenie	
	2	Administrátor: Zvolenie dokumentu, ktorý má byť odstránený	
	3	Systém: Odstránenie dokumentu z databázy	
	4	Systém: Odstránenie všetkých výskytov odstráneného dokumentu z indexu a zo všetkých tabuliek v databáze	
Alternatívna postupnosť	**Krok**	**Činnosť**	
	-	Administrátor: zrušenie odstránenia dokumentu	

Identifikátor	UC9		
Názov	Spustenie vytvorenia prepojení dokumentov		
Opis	Administrátor		
Priorita	vysoká	**Frekvencia**	Mesačne, rádovo desiatky
Vstup. podm.	Databáza dokumentov, pridané nové dokumenty do databázy		
Výstup. podm.	Znalostným systémom vytvorené požadované prepojenia medzi novými a starými dokumentmi v databáze		
Používatelia	Administrátor		
Základná postupnosť	**Krok**	**Činnosť**	
	1	Administrátor: Spustenie operácie	
	2	Systém: Modul pracujúci s node-rankingom vytvorí prepojenia medzi novými dokumentmi a ostatnými dokumentmi uloženými v databáze	
	3	Systém: Informovanie o úspešnosti vykonania operácie	

2.4 Model údajov

Model údajov použitý v systéme sa nachádza na obr. 11.



Obr. 11. Logický model údajov

Entity modelu údajov:

Dokument – predstavuje dokument uložený v databáze. Pozostáva z jeho názvu, metadát vo formáte XML (kvôli manažmentu znalostí), kategórie a samotného dokumentu v binárnych dátach (poskytnutého používateľovi v prípade výberu daného dokumentu). Dokumenty sú medzi sebou poprepájané (priame odkazy, podobnosť kľúčových slov, podobnosť názvov kapitol, podobnosť indexovaného obsahu dokumentu).

Používateľ – vystupuje v systéme ako niekto, kto vyhľadáva dokumenty. Evidovaný používateľ má svoje meno, ktoré jednoznačne určuje jeho identitu a rolu, v akej v systéme vystupuje (napr. analytik). Systém si uchováva informácie o tom, ktorý dokument používateľ vyhľadá a/alebo použije a poskytne mu dokumenty, ktoré vyberal iný používateľ s podobným výberom.

2.5 Nefunkcionálne požiadavky

Systém bude implementovaný v jazyku C# na platforme .NET verzia 2.0. Aplikácia bude poskytovať grafické používateľské rozhranie, ktoré umožní používateľovi vyhľadávať v báze znalostí a administrátorovi pridávať a odstraňovať dokumenty z databázy, pričom databáza bude relačná.

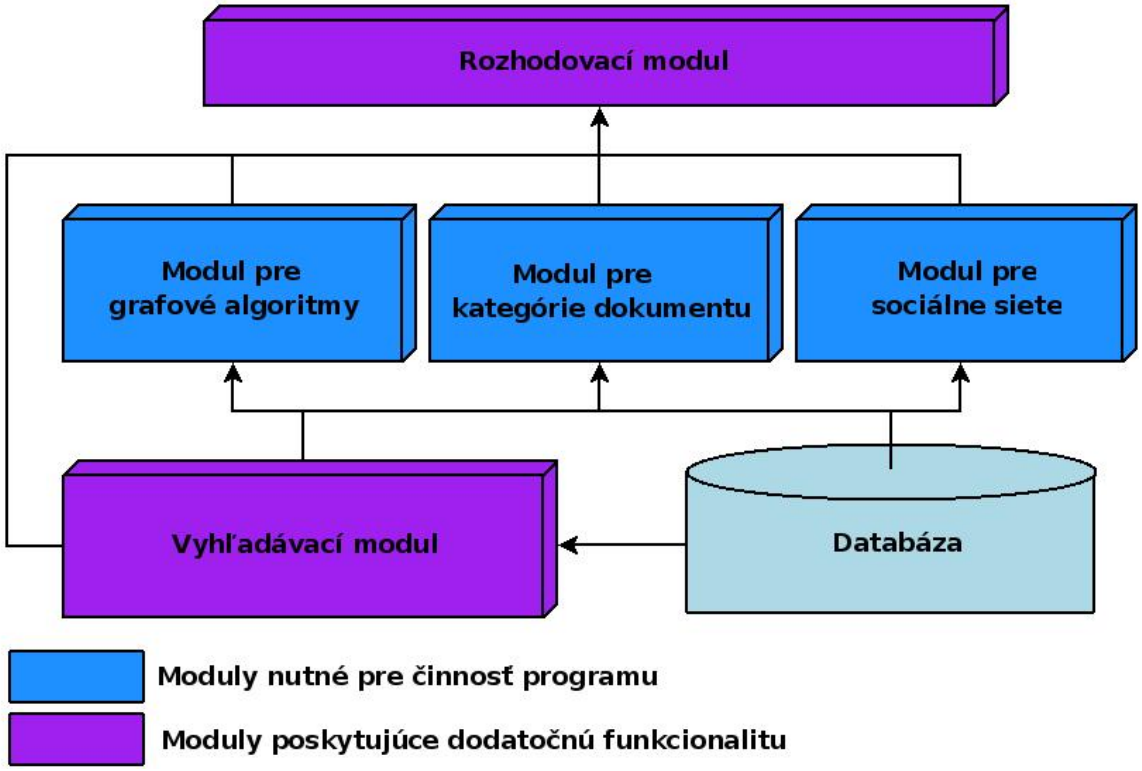
3 Návrh

3.1 Architektúra systému

V nasledujúcej kapitole bude popísaná architektúra systému, jeho časti a spôsob prepojenia medzi nimi (obr. 12).

Časti z ktorých s systém skladá môžeme rozdeliť do nasledujúcich kategórií:

1. Databáza
2. Moduly nutné k činnosti programu
3. Moduly poskytujúce dodatočnú funkcionalitu



Obr. 12. Architektúra systému

Databáza

V databáze sú uchovávané všetky informácie potrebné pre činnosť systému a jeho jednotlivých modulov. Znamená to, že sú tu uchovávané údaje ako dokumenty, prepojenia medzi dokumentmi a ich kategóriami, zoznamy toho kedy kto použil ktorý dokument.

Moduly nutné pre činnosť systému

Sú to moduly ktoré poskytujú svoje výstupy iným modulom alebo priamo používateľovi. Vzhľadom na fakt že ostatné moduly sú závislé na výstupe týchto modulov , funkčnosť systému v prípade vynechanie jedného z nich nie je možná.

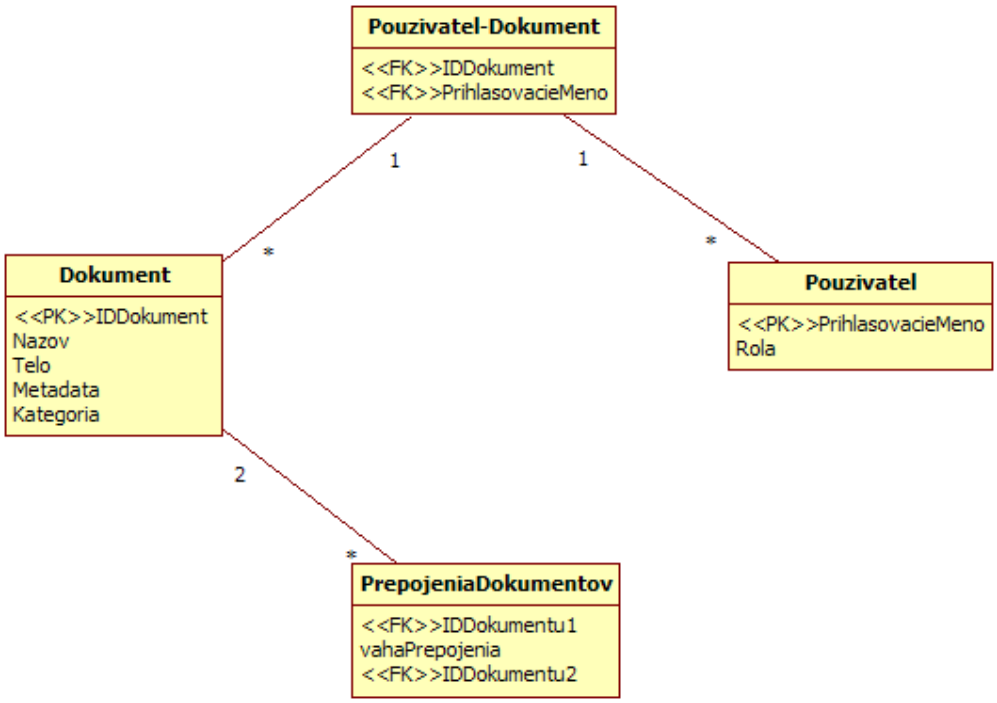
V prípade existencie len týchto modulov, je systém schopný poskytovať výstupy, no ich kvalita je otázna, lebo pôjde len o výstupy na základe vyhľadávanie zhody slov v texte.

Moduly poskytujúce dodatočnú funkcionalitu

Tieto moduly, ako je zrejmé z názvu poskytujú dodatočnú funkcionalitu pre vyhľadávanie znalostí. Ich použitím bude vyhľadávanie znalostí obohatené o výstupy ktoré b pri použití len základných modulov neboli nájdené.

3.2 Fyzický model

Táto kapitola obsahuje fyzický model údajov uložených v databáze.



Obr. 13. Fyzický model údajov uložených v databáze

Dokument - atribútmi tabuľky sú názov dokumentu, kategóriu dokumentu, metadáta vo formáte XML, binárnu formu súboru, ktorá bude poskytnutá používateľovi pri požiadavke na prezeranie

Pouzivatel - tabuľka evidovaných používateľov, ktorí majú jedinečné prihlasovacie mená a role, v akých vystupujú (napr. analytik, dokumentarista)

Pouzivatel-Dokument - tabuľka uchováva údaje o tom, ktoré dokumenty používatelia prezerali

PrepojeniaDokumentov - údaje o prepojeniach dokumentov podľa algoritmu ohodnocovania uzlov (*NodeRank*)

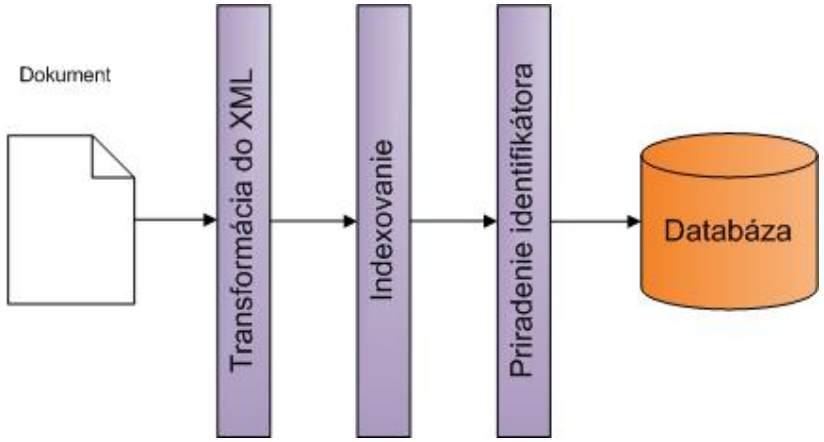
3.3 Vyhľadávací modul

Tento modul bude pevnou súčasťou celého systému. Keďže vyhľadávanie podľa kľúčových slov je základom pre dolovanie v dátach, modul bude poskytovať indexáciu dokumentov uložených v databáze a vyhľadávanie zadaných slov v indexovaných dokumentoch.

Je potrebné uvedomiť si, že dokumenty musia byť jednoznačne identifikovateľné vo všetkých moduloch. V databáze budú ukladané pod jednoznačným identifikátorom, ku ktorému budú prislúchať položky *názov* dokumentu, *metadáta* dokumentu vo formáte XML a *telo* dokumentu. Tento jednoznačný identifikátor bude vygenerovaný a priradený dokumentu pri jeho pridaní do databázy.

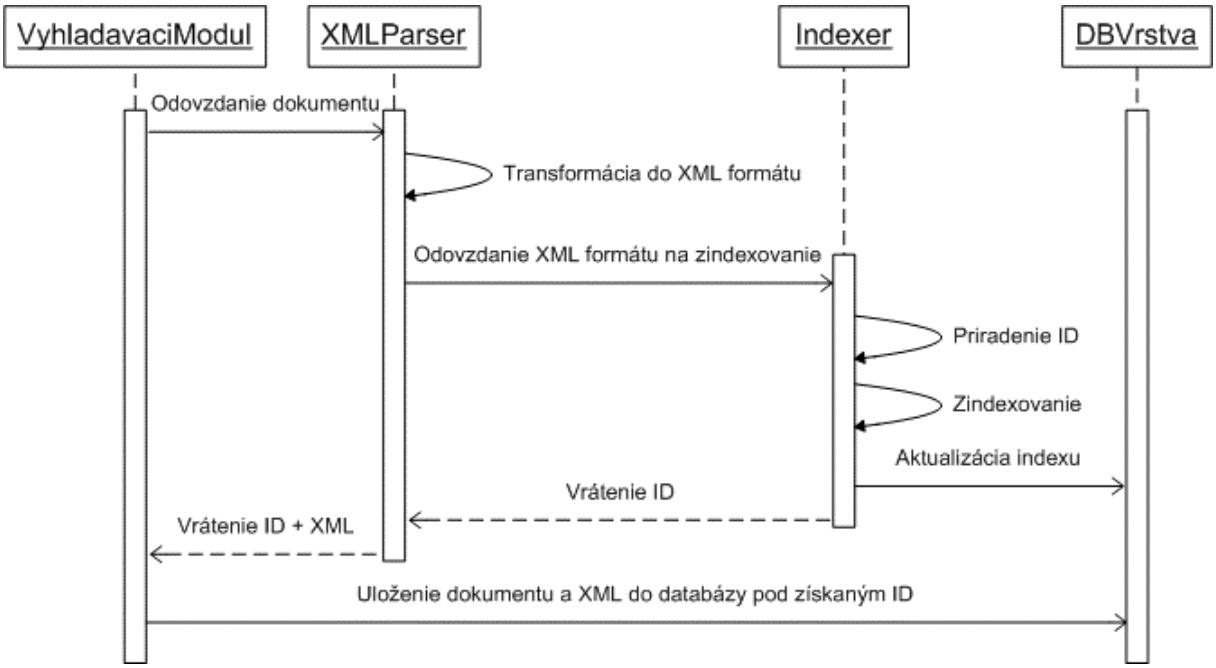
Pridanie dokumentu do databázy

Pridanie dokumentu do databázy je znázornené na obr. 14.



Obr. 14. Pridanie dokumentu do databázy

Najskôr dokument pretransformujeme do formátu XML. dotLucene z tejto reprezentácie pridá potrebné informácie do indexu dokumentov, pričom vytvorí dokumentu jedinečný identifikátor. Tento identifikátor sa použije ako identifikátor dokumentu v databáze! Následne budú samotný dokument, jeho XML forma a názov uložené do databázy. Na nasledujúcom obrázku je sekvenčný diagram pridania dokumentu do databázy.



Obr. 15. Sekvenčný diagram pridania dokumentu do databázy

Takúto databázu indexovaných dokumentov môžeme považovať za bázu znalostí, v ktorej môžeme vyhľadávať.

Odstraňovanie dokumentu z databázy

Pri odstraňovaní dokumentu z databázy musí systém odstrániť nielen samotný dokument vrátanie XML formátu, názvu a kategórie v databáze, ale aj všetky prepojenia vo všetkých moduloch a všetky výskyty v indexoch. Aby sme mohli odstrániť tieto doplňujúce údaje z bázy dát, musíme poznať identifikátor referencií, ktorým je identifikátor dokumentu pridelený pri pridávaní do databázy.

V dotLucene je implementovaných niekoľko spôsobov na odstraňovanie dokumentov z indexov. Napr. trieda *IndexModifier*, ktorá je súčasťou balíka *Lucene.Net.Index*, poskytuje funkciu *DeleteDocument*, ktoré z indexov odstránia všetky výskyty dokumentu, ktorého identifikátor zadáme ako argument funkcii.

Vyhľadanie dokumentov podľa kľúčových slov

Je zrejmé, že najdôležitejšou úlohou tohto modulu je vyhľadávanie dokumentov na základe kľúčových slov. Knižnice dotLucene ponúkajú kompletnú podporu vyhľadávania v indexoch. Je preto veľmi jednoduché nájsť všetky dokumenty, ktoré obsahujú zadané kľúčové slová. Tieto funkcie poskytuje trieda *Net.Lucene.Search.IndexSearcher*.

Vyhľadávací modul na vyhľadávanie používa metódu:

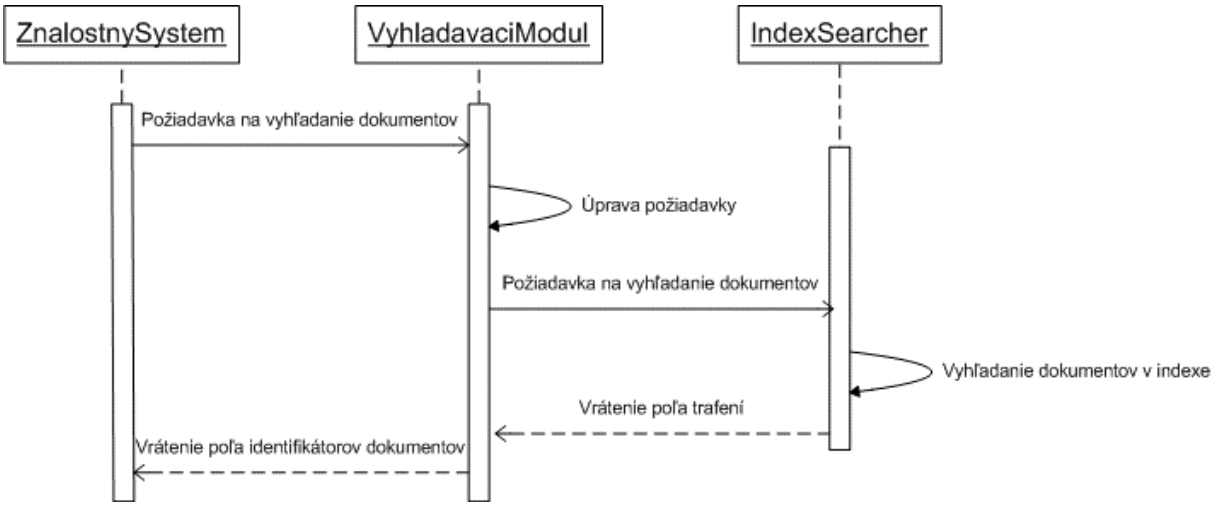
public Hits search(Query query, Sort sort)

- Metóda vracia množinu tzv. „trafení“ (trafenie obsahuje identifikátor dokumentu, relevanciu, a pod.), ktorou je možné prostredníctvom iterátora prechádzať výsledkami zoradenými podľa relevancie.
- Argumentmi sú požiadavka (na úpravu z textového vstupu zadaného používateľom do formátu očakávaného funkciou použijeme triedu *QueryParser*) a metóda usporiadania výsledkov (podľa

relevancie alebo podľa usporiadania v indexe). Modul používa usporiadanie podľa relevancie (RELEVANCE).

V prípade potreby je možné použiť metódu *public Hits search(Query query, Filter filter, Sort sort)*, ktorá medzi výsledky zaradí iba dokumenty splňujúce podmienky dané argumentom *filter*.

Výsledky budú spracované hlavným vyhodnocovacím systémom, ktorý ich poskytne používateľovi a ponúkne prezretie najvhodnejších dokumentov. Sekvenčný diagram vyhľadania dokumentov v indexoch podľa požiadavky hlavného vyhodnocovacieho systému je na obr. 16.



Obr. 16. Sekvenčný diagram vyhľadania dokumentov v indexe podľa požiadavky

Metódy zavolavšie inými modulmi

Každý modul v systéme má špecifické úlohy. Vďaka kooperácii modulov systém vyhodnotí a poskytne používateľovi najlepších nájdených kandidátov (dokumentov) podľa jeho charakteristík a požiadaviek. Napriek tomu, že moduly sú na sebe nezávislé, vyhľadávací modul musí niektorým z nich poskytnúť dokumenty s istými vlastnosťami. Preto budú v module implementované nasledujúce funkcie:

- *getDocIdsFromQuery(String[] keywords)*
  Vrátenie množiny identifikátorov dokumentov, v ktorých sa vyskytujú kľúčové slová.

- *getDocIdsFromQuery(String[] keywords, int maxCount)*
  Vrátenie množiny identifikátorov dokumentov, v ktorých sa vyskytujú kľúčové slová, pričom maximálna veľkosť množiny je daná druhým argumentom.

- *getDocIdsFromQueryWithRelevance(String[] keywords, float relevance)*
  Vrátenie množiny identifikátorov dokumentov, v ktorých sa vyskytujú kľúčové slová s relevanciou vyššou ako je hodnota druhého argumentu.

- *getDocIdsFromQueryWithRelevance(String[] keywords, int maxCount, float relevance)*
  Vrátenie množiny identifikátorov dokumentov, v ktorých sa vyskytujú kľúčové slová s relevanciou vyššou ako je hodnota tretieho argumentu. Maximálna veľkosť množiny je daná druhým argumentom.

Všetky tieto funkcie vracajú pole identifikátorov.

3.4 Modul pre kategórie dokumentu

Reprezentácie matice mier podobnosti medzi kategóriami

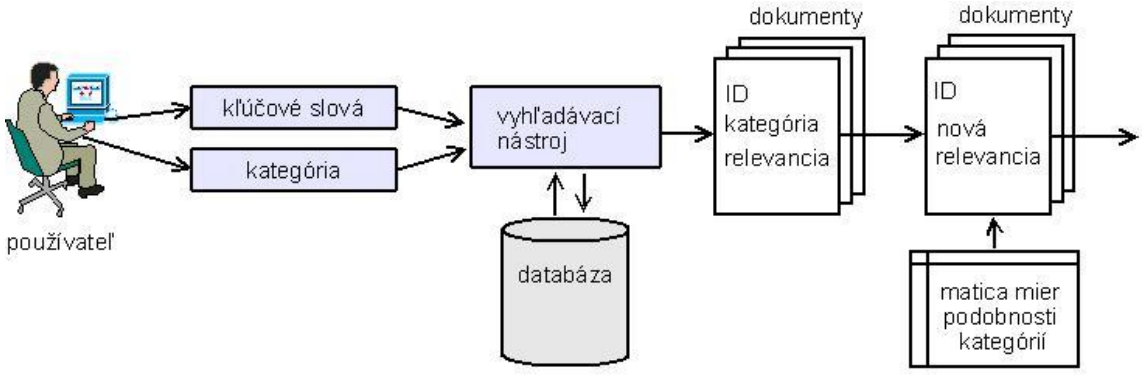
Táto matica uchováva reálne čísla - miery podobnosti medzi kategóriami. Čísla môžu nadobúdať hodnoty z intervalu <0.0, 1.0>, pričom ak dve kategórie spája hodnota 1.0, znamená to, že sa kategórie podobajú na 100%. V matici je potrebné mať zadefinované všetky kategórie dokumentov. Tieto kategórie, ako aj potrebné hodnoty, je nutné zadať manuálne, napríklad do konfiguračného súboru, odkiaľ budú do matice načítané. Keďže je matica symetrická, stačí zadať iba jednostranné prepojenie. Čo sa týka kategórií netreba zabúdať aj na možnosť, že súbor do žiadnej z nich spadať nemusí. Vtedy treba do konfiguračného súboru napísať novú kategóriu a hodnoty, udávajúce prepojenia s ostatnými, alebo mu prideliť kategóriu „nešpecifikovaná kategória“.

Kategorizácia dokumentov

Kategóriu každého dokumentu je potrebné špecifikovať už pri jeho vkladaní do databázy, pričom kategória, ktorú administrátor priradí dokumentu, sa musí nachádzať v už spomínanej matici mier podobnosti. Následne dokument putuje do databázy, kde sa mu v tabuľke „Dokument“ do atribútu „Kategoria“ zapíše kategória.

Vyhľadávanie dokumentov

Postupnosť krokov vyhľadávania dokumentov začína u používateľa, ktorý zadá kľúčové slová a špecifikuje kategóriu dokumentov, alebo sa kategória bude považovať za bližšie nešpecifikovanú. Systém na vyhľadávanie dokumentov potom podľa kľúčových slov vyhľadá skupinu dokumentov a vráti ich ID spolu s relevanciou a kategóriou, do ktorej dokument patrí. Kategória, ktorú používateľ zadal sa následne porovná podľa matice mier podobnosti s kategóriami dokumentov, ktoré vyhľadávací nástroj ponúkol. Miery pre každý dokument sa prenásobia s jeho relevanciu, čím sa získa nová relevancia. Takto ohodnotené dokumenty sa potom poskytujú na ďalšie spracovanie. Celý postup je zachytený na obr. 17.



Obr. 17. Proces vyhľadávania a ohodnocovania dokumentov na základe kategórií

Výpočet relevancie dokumentu na základe kategórie

Ako už bolo spomenuté, modul, ktorý bude mať na starosti výpočet relevancie dokumentov na základe kategórií, dostáva na vstupe dokumenty vyhľadané z databázy vyhľadávacím nástrojom, ktorý im priradí aj určitú relevanciu, podľa výskytu kľúčových slov. Ďalšie dôležité vstupy sú matica mier podobnosti kategórií dokumentov a samotná kategória špecifikovaná používateľom. Na základe kategórií dokumentov vyhľadaných v databáze a kategórie špecifikovanej používateľom sa z matice priradia jednotlivým dokumentom miery podobnosti. Každý dokument bude mať teda priradenú mieru podobnosti a relevanciu. Tieto dve čísla sa medzi sebou vynásobia, čím vzniknú nové relevancie priradené dokumentom. Tieto dokumenty, ako aj ich relevancie sa potom ďalej spracovávajú.

Príklad výpočtu relevancie:

Vstupy:

Kategória zadaná používateľom:
Zdroj. kód C#

Matica mier podobnosti medzi kategóriami:

	ZKC#	*ZKJava*	*PP*	*TD*	*ND*
ZKC#	1.00	0.75	0.10	0.30	0.50
ZKJava	0.75	1.00	0.10	0.30	0.50
PP	0.10	0.10	1.00	0.30	0.50
TD	0.30	0.30	0.30	1.00	0.50
ND	0.50	0.50	0.50	0.50	0.50

Legenda:
ZKC# – Zdroj. kód C#
ZKJava – Zdroj. kód Java
PP – Používateľská príručka
TD – Technická dokumentácia
ND – Nešpecifikovaný dokument

Dokumenty:
Dokument A {ID = 1, kategória: *ZKC#*, relevancia: 0.6}
Dokument B {ID = 3, kategória: *ZKJava*, relevancia: 0.5}
Dokument C {ID = 4, kategória: *PP*, relevancia: 0.4}
Dokument D {ID = 7, kategória: *ND*, relevancia: 0.3}
Dokument E {ID = 9, kategória: *TD*, relevancia: 0.2}

Výpočet:

Dokument A: 0.6 * 1.0 = 0.6
Dokument B: 0.5 * 0.75 = 0.375
Dokument C: 0.4 * 0.1 = 0.04
Dokument D: 0.3 * 0.5 = 0.15
Dokument E: 0.2 * 0.3 = 0.06

Výstup:

1.) Dokument A {ID = 1, relevancia: 0.6}
2.) Dokument B {ID = 3, relevancia: 0.375}
3.) Dokument D {ID = 7, relevancia: 0.15}
4.) Dokument E {ID = 9, relevancia: 0.06}
5.) Dokument C {ID = 4, relevancia: 0.04}

3.5 Modul pre grafové algoritmy

Úlohou tohto modulu je prehľadávanie stavového priestoru tvoreného dokumentmi a prepojeniami medzi nimi. Modul sa snaží nájsť dokumenty relevantné ku kľúčovým slovám zadaným používateľom. Na určenie relevantnosti jednotlivých dokumentov používa algoritmus NodeRank.

Vstupy Modulu

Modul dostane na vstupe zoznam kľúčových slov ktoré boli zadané používateľom.
Maximálny počet relevantných dokumentov, ktoré budú na výstupe modulu.

Výstup modulu

Zoznam relevantných dokumentov k zadaným kľúčovým slovám.

Komunikácia s okolím

V rámci systému modul komunikuje s Vyhľadávacím modulom, ktorý mu poskytuje zoznam dokumentov ktoré obsahujú najväčší počet kľúčových slov zadaných používateľom. V databáze má modul uložené prepojenie medzi jednotlivými dokumentmi , to znamená ID jednotlivých dokumentov a taktiež typ prepojenie ktorý je medzi nimi. Modul poskytuje svoje výstupy Rozhodovaciemu modulu.

Požiadavky používateľa sú modulu zadávané prostredníctvom grafického rozhrania priamo používateľom.

Činnosť modulu

Obr. 18. Graf činnosti modulu

Činnosť modulu spočíva v určení zoznamu dokumentov ktoré sú relevantné ku kľúčovým slovám.

Prvým krokom je získanie kľúčových slov, ktoré zadáva priamo používateľ. Po získaní týchto kľúčových slov modul pošle požiadavku na získanie ID dokumentu ktorý má najväčší počet podobných kľúčových slov ako v zadanom zozname. Toto ID získa od Vyhľadávacieho modulu.

Na základe tohto ID module prehľadá graf tvorený dokumentmi a prepojeniami medzi nimi a určí dokumenty ktoré sú s ním relevantné. Na určenie relevancie sa použije algoritmus NodeRank, ktorý je bližšie popísaný v časti analýza.

Vytváranie grafu

Pri vytváraní grafu z dokumentov a prepojení medzi nimi vychádzame z predpokladu, že pokiaľ je dokument A prepojený z dokumentom B je aj dokument B prepojený z dokumentom A. Po pridaní nového dokumentu do systému sa tento automaticky indexuje za použitia vyhľadávacieho modulu.

Následne sa v ňom určia priame odkazy na iné dokumenty a ich názvy. To znamená, že pokiaľ nový dokument obsahuje názov iného dokumentu, prípadne cestu na diskovú jednotku s týmto súborom identifikuje sa prepojenie medzi týmito dvoma dokumentmi.

Po vytvorení priamych prepojení dôjde k vytvoreniu prepojení na základe kľúčových slov v kapitolách, názvoch a celých dokumentov. Pre vytvorenie týchto prepojení bude potrebná spolupráca s Vyhľadávacím modulom. Vyhľadávací modul poskytne zoznam dokumentov, ktoré sa na určitý počet percent podobajú na nový dokument. To znamená že obsahujú určitý počet podobných slov. Pre začiatok bude použitá hodnota 80%, ktorá sa môže v procese implementácie zmeniť. Následne sa zaznamenajú nájdené prepojenia do databázy.

Obr. 19. Vytváranie prepojení dokumentu

3.6 Modul pre sociálne siete

Pri analýze tejto problematiky sa hovorilo o dvoch súčastiach vytvárania používateľského modulu: explicitné a implicitné. V tejto kapitole sa uvádza hrubý návrh týchto dvoch komponentov.

Manuálny komponent, slúžiaci na vytváranie profilov bude mať tieto hlavné funkcie:

- Vytvorenie nového profilu
- Zmena údajov v danom profile
- Prípadne vymazanie konta(najlepšie ak túto funkciu priradíme administrátorovi)

Editovanie profilu bude možné po prihlásení sa do systému, pričom prvé prihlásenie sa do systému je podmienené vytvorením konta. Tieto funkcie sú už všeobecne známe z webových portálov, stačí si napríklad zobrať vytváranie konta pre e-mail. Dôraz pri návrhu kladieme na navrhnutie údajov, ktoré vytvárajú profil používateľa. Navrhujeme tieto položky:

- prihlasovacie meno
- prihlasovacie heslo
- osobné údaje(meno, titul, kontakt)
- oddelenie, v ktorom používateľ pracuje(zadefinované administrátorom)
- témy, ktoré požívateľa zaujímajú
- oblasti v ktorých má dobré skúsenosti

V profile sa zachovajú aj údaje na akom aktuálnom projekte používateľ pracuje. A budú sa pamätať aj údaje, ktoré dokumenty pridal do systému. To sa však viacej týka explicitného komponentu, ktorý zaznamenáva interakciu používateľa so systémom.

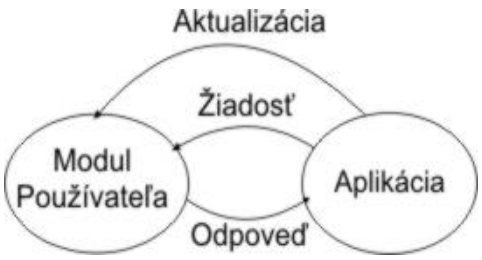
Dynamický komponent umožní sledovať tieto akcie systému:

- Pridávanie dokumentov do bázy dát
- Vyhľadávanie dokumentov, zadávanie parametrov vyhľadávania
- Vyberanie dokumentov z alternatívne poskytnutých

Údaje(získané z interakcie), ktoré sa uložia do modelu používateľa:

- Dokumenty, ktoré používateľ pridal do databázy
- Dokumenty, vytvorené autorom
- Ktorý dokument koľkokrát vyhľadával
- Počet využitých alternatív, tým sa dá overiť nakoľko spĺňa systém používateľove požiadavky a meniť údaje na základe týchto dát

Samotná komunikácia aplikácie s týmto modulom bude založená na posielaní správ a získavaní odpovedí. Z aplikácie sa pošle žiadosť na údaje o používateľovi, tieto sa zahrnú do vyhľadávania dokumentov. Po vyhľadaní sa aktualizuje profil používateľa na základe jeho správania sa. Znázornenie tejto myšlienky je zobrazené na obr. 20. [5]



Obr. 20. Komunikácia modulu s aplikáciou

4 Použitá literatúra

1. Znalostný manažment na báze technológie .NET (Tímový projekt) – dokumentácia k projektu Lucky Number 7, http://www2.dcs.elf.stuba.sk/TeamProject/2006/team07/public_html/ (14.11.2007)
2. Agrawal, R., Srikant, R.: Fast algorithms for mining associations rules. In: J.B. Bocca, M.Jarke. C.Zaniolo (eds.), Proc. of 20th VLDB Conference, Santiago de Chile, 12-15 September 1994, Morgan Kaufmann, pp. 487-499.
3. Bieliková, M., Návrat, P., a kol.: Štúdie vybraných tém softvérového inžinierstva 2. Slovenská technická
univerzita, 2006.
4. Duffner, S.: A Knowledge Portal for Multi-Project Management. Karlsbad, 2000. 90 s. Diplomová práca na Univerzite Karlshure. Vedúci prace: prof. Dr. Studer.
5. Fröschl, Christoph: User Modeling and User Profiling in Adaptive E-learning Systems, diplomová práca na Technickej Univerzite Graz, November 2005
6. Gudivada, V. N. et al.: Information Retrieval on the World Wide Web, IEEE Internet Computing, September-October 1997, pp. 58-68.
7. Konchady, M.: Text Mining Application Programming. Charles River Media, Boston, 2006.
8. Paralič, J.: Manažment znalostí – podklady k prednáškam, http://www.tuke.sk/paralicj/mz.html (15.11.2006)
9. Razmerita, L., Angehrn, A., Maedche, A.: Ontology-based User Modeling for Knowledge Management Systems, INSEAD, CALT-Centre of Advanced Learning Technologies
10. Seekafile - flexible indexing server, http://sourceforge.net/project/showfiles.php?group_id=143466/ (10.11.2007)
11. http://lucene.apache.org/java/docs/features.html
12. http://www.codeproject.com/aspnet/DotLuceneSearch.asp