Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ
Ilkovičova 3, 812 19 Bratislava

Tímový projekt

Grafická podpora vyhľadávania znalostí v dokumentoch

Tím 12: Šproty

Bc. Marian Beňo, Bc. Miloš Blaško, Bc. Ľubomír Eľko, Bc. Ján Kmeťko, Bc. Lukáš Lazarčík, Bc. Tomáš Mičko

Vedúci: Ing. Ivan Polášek PhD.

Kontakt:

sproty@googlegroups.com

História vývoja dokumentu

Dátum	Verzia dokumentu	Popis	Vykonal
1.11.2009	1.0	Vytvorenie dokumentu	Lukáš Lazarčík
2.11.2009	1.1	Pridanie kapitol od jednotlivých autorov (Autori jednotlivých kapitol sú rozpísaní v dokumentácii k riadeniu)	Lukáš Lazarčík
10.12.2009	1.2	Úpravy na formálnej stránke dokumentu	Lukáš Lazarčík
12.12.2009	1.3	Úpravy v dokumente, pridanie nových kapitol	Lukáš Lazarčík

Obsah

0. Úvod

Účelom tohto dokumentu je zdokumentovať 1. Iteráciu procesu vývoja projektu Grafická podpora vyhľadávania v dokumentoch, ktorý riešia študenti inžinierskeho štúdia na Fakulte informatiky a informačných technológií v rámci predmetu Tímový projekt pod vedením Ing. Ivana Poláška PhD. Jednotlivé kapitoly tohto dokumentu sa zaoberajú špecifikáciou požiadaviek na výsledný produkt, analýzou a výberom technológií použitých pri riešení projektu a návrhom tohto systému. Obsahom tejto kapitoly zadanie projektu a naša vízia jeho riešenia a slovník pojmov.

0.1 Zadanie

Úlohou tímu bude vytvoriť grafický modul pre zobrazenie prepojenia dokumentov, ktorý by sa neskôr stal súčasťou produktu Gratex Knowledge Office a napomáhal tvorbe nových dokumentov (analytickej a technickej dokumentácie, manuálov, zdrojových kódov a pod.) pomocou získaných znalostí.

Hrany medzi jednotlivými dokumentmi budú mať rôzny vzor, farbu, hrúbku a podobne podľa množstva odkazov, spoločných kľúčových slov, autorov, čitateľov/používateľov, typov dokumentov a podobne.

Jednotlivé algoritmy vyhľadávania a prepojení bude možné vypínať, ako aj vyberať v grafe vhodné dokumenty, vyraďovať nepotrebné cesty alebo vetvy. Na záver bude možné vybrať zaujímavé dokumenty, ktoré budú zdrojom pre tvorbu obsahu nového dokumentu.

0.2 Vízia projektu Grafická podpora vyhľadávania znalostí v dokumentoch

V prostredí softvérovej firmy ale aj v iných znalostne orientovaných firmách dochádza k vzniku veľkého množstva znalostí a vedomostí. Aby ich bolo možné v efektívnej miere zdieľať, je nielen potrebné tieto znalosti zachytiť, zvečniť v elektronických dokumentoch, ale aj vedieť medzi týmito dokumentmi rýchlo vyhľadávať. Touto problémovou doménou sa zaoberajú systémy pre správu bázy znalostí (knowledgebase). Manuálne udržiavanie bázy znalostí je náročné a vo firme s veľa znalosťami by si vyžiadalo veľa vynaloženého úsilia, preto je výhodné hľadať spôsoby, ako bázu znalostí udržiavať čo možno najviac automaticky.

Už na prvý pohľad sa naskytá nápad použiť už existujúce zápisy znalostí (rôzne dokumenty, atď.) a tieto dokumenty tak poprepájať, a umožniť v nich tak vyhľadávať, aby sa používateľ čo najrýchlejšie dostal ku hľadaným vedomostiam a informáciám.

Cieľom projektu Grafická podpora vyhľadávania znalostí v dokumentoch je vytvorenie takéhoto systému pre údržbu bázy znalostí a hlavne pre vyhľadávanie v nej. Kľúčovou myšlienkou je hľadanie väzieb medzi dokumentmi vloženými do systému a zobrazenie týchto väzieb. Väzby medzi dokumentmi je možné tvoriť a udržiavať na základe:

- Autorov
- Referencií medzi dokumentmi (jeden odkazuje druhý)
- Kľúčových slov
- Označenia používateľom

Používateľ sa v grafe dokumentov bude pohybovať cez existujúce väzby medzi dokumentmi, pričom tieto väzby môže aktívne mazať a upravovať. Zobrazenie grafu dokumentov musí byť veľmi interaktívne a prehľadné a kvalita zobrazenia tvorí podstatnú časť prínosu projektu. Vyvinutý systém musí byť schopný samostatne vyhľadať väzby medzi dokumentmi, čo je podmienkou aby nemuselo byť na údržbu bázy znalostí vynaložené priveľké úsilie.

Našou víziou je počas tímového projektu I. a II. Minimálne vytvoriť systém, ktorý bude schopný na reálnej množine dokumentov vyhľadať väzby aspoň na základe kľúčových slov a tieto väzby zobraziť v interaktívnom grafe. Tento systém chceme vylepšiť do takej miery, ako nám to umožní náš čas a situácia na projekte.

Chyba! Na použitie štýlu Heading 1 na text, ktorý sa má zobraziť na tomto mieste, použite kartu Domov.

0.3 Slovník projektových pojmov

Centrálny dokument – taký dokument, ktorý je momentálne zobrazený na ploche v jej strede. Na ploche sú zobrazené len dokumenty, ktoré sú nejakým spôsobom previazané, alebo tranzitívne previazané s centrálnym dokumentom.

Dokument – jednotka bázy znalostí obsahujúca nejaké informácie. Najčastejšie je dokumentom textový súbor, alebo súbor vo formáte .doc.

Doména – oblasť, alebo okruh znalostí. Napríklad znalosti okolo nejakého projektu, alebo okolo nejakej technológie, alebo problematiky. Dokument sa skladá zo súboru dokumentu a metadát (väzby atď.)

Odkazujúci dokument, odkazovaný dokument – pri jednosmernej väzbe je odkazujúci dokument taký z ktorého vychádza šípka. Odkazovaný je ten, do ktorého šípka smeruje. V odkazujúcom dokumente sú nejaké informácie. V odkazovanom dokumente sú tieto informácie:

Buď podrobnejšie vysvetlené (detailnejšie). Napríklad dokument MetodikaVývoja.doc odkazuje dokument PravidláPísaniaKomentárov.doc.

Alebo popísané na vyššej úrovni abstrakcie, popísané všeobecnejšie. Napríklad dokument BázaZnalostí.doc odkazuje dokument TeóriaGrafov.doc

Plocha - zobrazovacia komponenta používateľského rozhrania na ktorej sú zobrazované dokumenty a vzťahy medzi nimi.

Súbor dokumentu – samotný súbor dokumentu, súbor v súborovom systéme, alebo v databáze.

Trieda väzby – určuje za akých okolností bola väzba vytvorená. Väzby podľa triedy rozlišujeme na vytvorené používateľom, vymazané používateľom a vytvorené automaticky algoritmom na vyhľadávanie väzieb.

Typ väzby – určuje reálny význam väzby. Typ väzby napríklad určuje či ide o väzbu vytvorenú na základe rovnakého obsahu, rovnakého autora, alebo iné.

Virtuálny dokument – taký dokument, ktorý sám o sebe neobsahuje informácie, ale svojimi väzbami združuje dokumenty obsahujúce informácie o určitej doméne, alebo problematike.

Väzba medzi dokumentmi – vzťah medzi dokumentmi vyjadrujúci ich sémantickú, alebo inú previazanosť. Väzby môžu byť viacerých typov, napr. veľa spoločných kľúčových slov, spoločný autor, manuálne vytvorená väzba. Väzby môžu byť jednosmerné aj obojsmerné.

0.4 Slovník použitých cudzích pojmov

TODO

1. Špecifikácia

1.1 Používateľské príbehy

Načrtnime niekoľko príbehov, v akých prípadoch môže byť použitý vyvíjaný systém. Na základe nich budeme formulovať funkčné požiadavky na systém. Nasledovné príbehy obsahujú aj náznaky funkcionality, ktorá nebude z časových dôvodov ani analyzovaná ani implementovaná, ale napriek tomu bude figurovať ako požiadavka s nízkou prioritou.

Príbeh 01 – Zorientovanie sa v novom projekte

Michal pracuje v softvérovej firme ako programátor. Dostal úlohu vykonať pár zmien na projekte, ktorý sa robil vo firme pred dvoma rokmi. Človek, ktorý projekt v minulosti realizoval už vo firme nepracuje a tak Michal siahne po báze znalostí. Dá si vyhľadať všetky dokumenty, ktoré obsahujú názov projektu na ktorom má pracovať. Systém mu zobrazil všetky dokumenty, ako aj ich väzby na iné dokumenty. Postupným prezeraním dokumentov sa Michal dostal k informáciám, ktoré potrebuje, aby mohol dokončiť svoju prácu.

Príbeh 02 – Vyhľadanie informácií o firemnom frameworku

Štefan pracuje v softvérovej firme ako architekt jeden rok. Dostal sa na projekt, kde mu bolo navrhnuté, aby použil pri návrhu architektúry už existujúci firemný framework. Keďže sa s ním v praxi ešte nestretol, rozhodol sa vyhľadať si o ňom podrobné informácie vo firemnej báze znalostí. V systéme si vyhľadal dokumenty súvisiace s danou knižnicou. Systém mu zobrazil dokumenty podľa relevancie. Postupným prezeraním v grafe dokumentov sa Štefan dostal ku všetkým informáciám, ktoré boli v báze znalostí k frameworku k dispozícii.

Príbeh 03 – Pridanie dokumentu analýzy, úprava vzniknutých väzieb

Igor pracuje ako analytik na novom projekte. Po dokončení analýzy sa rozhodol, že tento dokument pridá do bázy znalostí, aby ho sprístupnil do budúcnosti. Po pridaní ho systém zindexoval a vytvoril väzby na iné dokumenty, ktoré sa už v báze nachádzali. Igor si pozrel vzniknuté väzby a zistil, že veľa väzieb vzniklo na dokumenty staršieho projektu, ktorý má podobné meno. Tieto väzby Igor zrušil a nechal iba tie, ktoré sú pravdivé.

Príbeh 04 – Odstránenie chybného dokumentu z bázy znalostí

Martin pridal do bázy znalostí omylom iný dokument ako chcel (pošmykla sa mu myška a nevšimol si to). Po týždni to zistil, keď sa ho kolega pýtal, čo tam robí tento nevhodný dokument. Martin zistil svoj omyl a rozhodol sa dokument zo systému vymazať. Aj tak urobil.

Príbeh 05 – Nastavenie prístupových práv

Oddelenie obchodná skupina Verejné zákazky vytvorila smernicu jednania so zákazníkmi z radov štátnych inštitúcii. Táto smernica má byť známa a prístupná všetkým členom oddelenia, ale nie ostatným zamestnancom, lebo sa jedná o dôverný dokument. Preto referentka Andrea pridala dokument do bázy znalostí, ale ten sa bude zobrazovať len výlučne členom spomínaného oddelenia. Po čase však chcel dokument vidieť generálny riaditeľ, ale nevedel ho nájsť. Preto Andrea rozšírila viditeľnosť dokumentu aj na generálneho riaditeľa a zvyšné dôležité pozície vo vedení firmy.

Príbeh 06- Vytvorenie virtuálneho dokumentu

Personalistka Janka dostala podnet od nových zamestnancov, že im dlho trvá zorientovanie sa vo firme. Preto sa rozhodne vytvoriť dokument na ktorý ich bude odkazovať. Tento dokument chce mať rýchlo vytvorený, preto sa rozhodne použiť už existujúce smernice. Vytvorí preto v báze znalostí iba virtuálny dokument, ktorý bude mať priame referencie na už existujúce dokumenty. Nový zamestnanci potom začínajú práve na tomto dokumente a cez väzby sa dostanú k všetkým potrebným informáciám.

Vedúci programátorov Miro však zistil, že v tomto dokumente pre nových zamestnancov nie sú žiadne informácie pre programátorov. Preto vytvoril ďalší virtuálny dokument pre nových programátorov a ten zviazal s dokumentom pre nových zamestnancov. Tiež mu nastavil viditeľnosť len pre rolu programátor, aby nezaťažoval tých nových zamestnancov, ktorí nie sú programátormi.

1.2 Požiadavky

Triedy aktérov

V systéme rozlišujeme nasledovné triedy používateľov:

Bežný používateľ

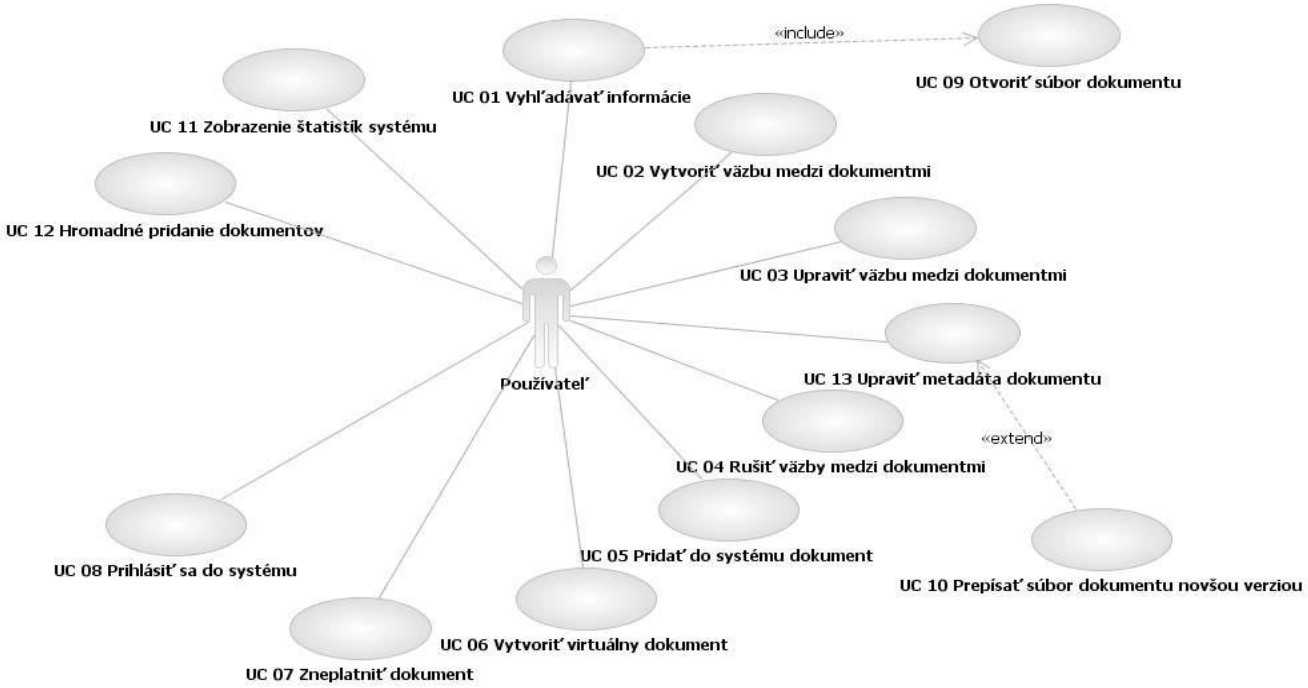
Bežný používateľ bázy znalostí, ktorého cieľmi je zveľaďovanie bázy znalostí a vyhľadávanie informácii v nej. Pod zveľaďovaním rozumieme hlavne pridávanie vlastných znalostí (dokumentov) a pomoc pri identifikovaní väzieb v doménach znalostí. Pri vyhľadávaní v báze znalostí získava znalosti iných používateľov, ktorí ich predtým do bázy znalosti vložili.

Administrátor

Používateľ systému, ktorého zodpovednosťou je udržať systém v chode. Zodpovedá za automatické funkcie systému (automatické hľadanie väzieb), ako aj za správu používateľov. Administrátor dedí práva od bežného používateľa. Z dôvodu jednoduchosti systému bude všetka funkcionalita, ku ktorej má možnosť pristupovať administrátor realizovaná pomocou konfigurácie .xml súbormi a **nebude ani analyzovaná ani implementovaná.** V neskorších verziách je možné pre túto funkcionalitu zriadiť databázu, vystavať model a používateľské rozhranie.

1.3 Prípady použitia

Prípady použitia sú zamerané na vyhľadávanie v báze znalostí, jej rozširovanie a udržiavanie. Konfigurácia systému (algoritmy vyhľadávania väzieb, atď.) a jeho údržba bude v prvej verzii systému riešená cez statickú konfiguráciu a operácie priamo v databáze. V neskorších verziách je možné rozanalyzovať, navrhnúť a implementovať aj administračné rozhranie.



Obr. 1 – Prípady použitia aktéra Používateľ

UC 01 – Vyhľadávať informácie

Identifikátor	UC01		
Názov	Vyhľadávať informácie		
Opis	Používateľ využíva používateľské rozhranie na prezeranie dokumentov a vzťahov medzi nimi. Systém reaguje na používateľove podnety podľa špecifikácie funkcionality komponenty plocha.		
Priorita	nevyhnutná	**Frekvencia**	Niekoľko krát za deň, až niekoľko krát za sekundu
Vstup. podm.	Používateľ je prihlásený do systému		
Výstup. podm.	Používateľ videl väzby medzi dokumentmi, dokumenty, niektoré z nich má, alebo mal otvorené a videl ich obsah a mohol si ich stiahnuť, alebo uložiť.		
Aktér	Bežný používateľ		

Základná postupnosť	**Krok**	**Činnosť**
	1	Používateľ vyplní a odošle formulár pre vyhľadanie informácií (autor a/alebo názov a/alebo keywords...)
	2	Systém zobrazí zoznam dokumentov, ktoré vyhovujú vyhľadávacím pravidlám
	3	Používateľ si zvolí jeden dokument zo zoznamu a nechá ho zobraziť na ploche
	4	Systém zobrazí plochu. Na nej zobrazí zvolený dokument ako centrálny dokument, ako aj všetky defaultne dosiahnuteľné dokumenty z centrálneho dokumentu.
	5	Používateľ využíva plochu (viď jej funkcionalitu) na prezeranie súvisiacich dokumentov a väzieb medzi nimi. (Include UC 09 Otvoriť súbor dokumentu)

	6	Používateľ ukončil prezeranie
	7	Systém zatvorí / vyčistí plochu
Alternatívna postupnosť	**Krok**	**Činnosť**
	3.a	Používateľ si zvolí možnosť zobraziť všetky dokumenty.
	4.a	Systém zobrazí plochu. Na nej vytvorí jeden neperzistentý virtuálny dokument, ktorý predstavuje zadané vyhľadávanie. Od neho budú zobrazené väzby k vyhľadaným dokumentom a od nich budú zobrazené ďalšie väzby, ktoré od nich existujú.
Rozširujúce body	5	Tento bod je bodom rozšírenia pre iné prípady použitia

Tab. 1. UC01- vyhľadať informácie

UC 02 – Vytvárať manuálne väzby medzi dokumentmi

Identifikátor	UC02		
Názov	Vytvárať manuálne väzby medzi dokumentmi		
Opis	Používateľ identifikuje väzbu medzi dokumentmi a vytvorí ju, aby zachytil tento vzťah.		
Priorita	Stredná až vysoká	**Frekvencia**	Niekoľko krát za deň
Vstup. podm.	Používateľ je prihlásený do systému. Používateľ má na ploche zobrazené obidva dokumenty, medzi ktorými chce vytvoriť väzbu		
Výstup. podm.	V systéme je vytvorená a uložená väzba medzi dokumentmi		
Aktér	Bežný používateľ		
Základná postupnosť	**Krok**	**Činnosť**	
	1	Používateľ vyvolá akciu vytvoriť väzbu medzi dokumentmi	
	2	Systém vyzve používateľa, aby na ploche klikol na odkazujúci dokument	
	3	Používateľ klikne na odkazujúci dokument.	
	4	Systém na ploche vyznačí odkazujúci dokument a vyzve používateľa na označenie odkazovaného dokumentu	
	5	Používateľ klikne na ploche na odkazovaný dokument	
	6	Systém označí odkazovaný dokument a vyzve používateľa na doplnenie ostatných parametrov väzby (typ, sila, obojsmernosť, atd.)	
	7	Používateľ vyplní ostatné parametre väzby a potvrdí	

	8	Systém vytvorí a uloží väzbu podľa zadaných parametrov.
Alternatívna postupnosť	**Krok**	**Činnosť**
Rozširujúce body		

Tab. 2. UC02- vytvárať manuálne väzby medzi dokumentmi

UC 03 Upraviť väzbu medzi dokumentmi

Identifikátor	UC03		
Názov	Upraviť väzbu medzi dokumentmi		
Opis	Používateľ zistí, že niektorá väzba nemá správne atribúty a preto jej ich zmení.		
Priorita	Stredná až vysoká	**Frekvencia**	Niekoľko krát za deň
Vstup. podm.	Používateľ je prihlásený do systému. Používateľ má na ploche zobrazenú väzbu, ktorú chce upraviť		
Výstup. podm.	Väzba má zmenené atribúty a tieto zmeny sú trvalo uložené		
Aktér	Bežný používateľ		
Základná postupnosť	**Krok**	**Činnosť**	
	1	Používateľ označí upravovanú väzbu. Používateľ vyvolá akciu upraviť väzbu	
	2	Systém zobrazí dialóg na úpravu väzby.	
	3	Používateľ upraví atribúty väzby a vyvolá akciu uložiť	
	4	Systém uloží zmenenú väzbu	
Alternatívna postupnosť	**Krok**	**Činnosť**	
Rozširujúce body			

Tab. 3. UC03- upraviť väzbu medzi dokumentmi

UC 04 Rušiť väzby medzi dokumentmi

Identifikátor	UC04		
Názov	Rušiť väzby medzi dokumentmi		
Opis	Používateľ zistí, že niektoré väzby nie sú opodstatnené. Boli napríklad vytvorené automaticky, ale systém na automatizované vyhľadávanie väzieb ich neurčil správne. Používateľ túto väzbu vymaže a systém si to zapamätá.		
Priorita	Stredná až vysoká	**Frekvencia**	Niekoľko krát za týždeň
Vstup. podm.	Používateľ je prihlásený do systému. Používateľ má na ploche zobrazené väzby, ktoré chce vymazať.		
Výstup. podm.	Väzby sú vymazané a systém si pamätá, ktoré väzby boli vymazané		
Aktér	Bežný používateľ		
Základná postupnosť	**Krok**	**Činnosť**	
	1	Používateľ označí tie väzby, ktoré chce vymazať. Používateľ vyvolá akciu vymazať väzby.	
	2	Systém sa spýta používateľa, či chce naozaj vymazať väzby	
	3	Používateľ potvrdí	
	4	Systém vymaže vyznačené väzby. Systém si zapamätá, že takéto väzby boli vymazané, aby ich nevytváral automaticky v budúcnosti.	
Alternatívna postupnosť	**Krok**	**Činnosť**	
Rozširujúce body			

Tab. 4. UC04- rušiť väzby medzi dokumentmi

UC 05 Pridať do systému dokument

Identifikátor	UC05		
Názov	Pridať do systému dokument		
Opis	Používateľ vloží do systému dokument s informáciami, ktoré chce zdieľať v báze znalostí.		
Priorita	Stredná až vysoká	**Frekvencia**	Niekoľko krát za týždeň
Vstup. podm.	Používateľ je prihlásený do systému. Súbor s dokumentom je na používateľovom počítači.		
Výstup. podm.	V systéme (báze znalostí) je pridaný dokument		
Aktér	Bežný používateľ		

Základná postupnosť	**Krok**	**Činnosť**
	1	Používateľ označí vyvolá akciu pridať dokument
	2	Systém zobrazí dialóg na pridanie dokumentu.
	3	Používateľ vyberie súbor, s dokumentom. Používateľ vyplní ostatné atribúty dokumentu. Používateľ vyberie, aké typy väzieb sa majú nad dokumentom vyhľadať hneď po pridaní.
	4	Systém uloží dokument do databázy a uloží si o ňom všetky informácie. Systém sa pokúsi vyhľadávať väzby práve pridaného dokumentu. Systém vyhľadané väzby vytvorí a uloží v databáze. Systém sa spýta sa používateľa, či si praje zobraziť na ploche práve pridaný dokument aj s vyhľadanými väzbami.
	5	Používateľ potvrdí a systém na ploche zobrazí pridaný dokument ako

		centrálny dokument aj s práve vytvorenými väzbami.
Alternatívna postupnosť	**Krok**	**Činnosť**
	4.a	Systém zistil, že súbor nie je v podporovanom formáte. Systém o tom používateľa upovedomí a beh pokračuje krokom 3
	5.b	Používateľ zamietol a beh končí....t.j. obsah plochy ostáva rovnaký
Rozširujúce body		

Tab. 5. UC05- pridať dokument do systému

UC 06 Vytvoriť virtuálny dokument

Identifikátor	UC06		
Názov	Vytvoriť virtuálny dokument		
Opis	Používateľ vytvorí virtuálny dokument, ktorý sám o sebe neobsahuje informácie, ale svojimi väzbami združuje dokumenty, ktoré obsahujú informácie o určitej doméne. Tento virtuálny dokument sa uloží do databázy.		
Priorita	Nízka	**Frekvencia**	Niekoľko krát za mesiac
Vstup. podm.	Používateľ je prihlásený do systému		
Výstup. podm.	V systéme (báze znalostí) je vytvorený a zapamätaný virtuálny dokument		
Aktér	Bežný používateľ		

Základná postupnosť	**Krok**	**Činnosť**
	1	Používateľ vyplní formulár na vyhľadanie informácií a potvrdí
	2	Systém zobrazí zoznam vyhľadaných dokumentov
	3	Používateľ si zvolí možnosť zobraziť všetky dokumenty.
	4	Systém zobrazí plochu. Na nej zobrazí ako centrálny dokument jeden neperzistentý virtuálny dokument, ktorý predstavuje zadané vyhľadávanie. Od neho budú zobrazené neperzistentné väzby k vyhľadaným dokumentom a od nich budú zobrazené ďalšie väzby, ktoré od nich existujú.
	5	Používateľ označí neperzistentný virtuálny dokument, ktorý je v tom čase centrálnym dokumentom a vyvolá akciu uložiť virtuálny dokument
	6	Systém zobrazí dialóg pre uloženie virtuálneho dokumentu
	7	Používateľ vyplní nasledovné a potvrdí

		1.	Názov virtuálneho dokumentu
		2.	Kľúčové slová (tie, čo pochádzali z vyhľadávania opäť vypĺňať nemusí)
		3.	Popis virtuálneho dokumentu, akú doménu zhromažďuje
	8	Systém uloží nový virtuálny dokument, ako aj všetky neperzistentné väzby, ktoré vznikli vyhľadávaním (stanú sa z nich väzby perzistentné).	
	9	Používateľ môže s dokumentom narábať ako s normálnym dokumentom (pridávať a editovať väzby atď.)	
Alternatívna postupnosť	**Krok**	**Činnosť**	
	1.a	Používateľ vyvolá akciu vytvoriť nový virtuálny dokument	
	2.a	Systém zobrazí dialóg na pridanie nového virtuálneho dokumentu	
	3.a	Používateľ vyplní názov, kľúčové slová a popis domény a vyvolá akciu uložiť	
	4.a	Systém uloží nový virtuálny dokument a zobrazí ho na ploche ako jeden z centrálnych dokumentov. K tomuto dokumentu ešte neexistujú žiadne väzby.	
	5.a	Používateľ môže s dokumentom narábať ako s normálnym dokumentom (pridávať a editovať väzby, atď.)	
	5.b	Používateľ môže mazať neperzistentné väzby tak, že na ne klikne a stlačí klávesu delete.	

Tab. 6. UC06- vytvoriť virtuálny dokument

UC 07 Zneplatniť dokument

Identifikátor	UC07		
Názov	Zneplatniť dokument		
Opis	Používateľ nájde neaktuálny, alebo inak nevhodný dokument a zneplatní ho, čím sa tento bude zobrazovať len za špeciálnych okolností		
Priorita	Stredná	**Frekvencia**	Niekoľko krát za mesiac
Vstup. podm.	Používateľ prihlásený do systému Mazaný dokument je v systéme a zobrazený na ploche		
Výstup. podm.	Mazaný dokument, ani žiadne väzby s ním súvisiace nie je v systéme		
Aktér	Bežný používateľ		
Základná postupnosť	**Krok**	**Činnosť**	
	1	Používateľ označí na ploche zneplatňovaný dokument a vyvolá akciu Zneplatniť dokument	
	2	Systém sa spýta používateľa, či si je istý zneplatnením	
	3	Používateľ potvrdí	
	4	Systém nastaví atribút isValid daného dokumentu na false a rovnako označí aj všetky väzby, ktoré do a z tohto dokumentu vedú.	
Alternatívna postupnosť	**Krok**	**Činnosť**	
Rozširujúce body			

Tab. 7. UC07- zneplatniť dokument

UC 08 Prihlásiť sa do systému

Identifikátor	UC08		
Názov	Prihlásiť sa do systému		
Opis	Používateľ použije svoje prihlasovacie údaje na prihlásenie sa do systému		
Priorita	Stredná	**Frekvencia**	Niekoľko krát za hodinu
Vstup. podm.	Používateľ je registrovaný v systéme		
Výstup. podm.	Používateľ je prihlásený do systému		
Aktér	Bežný používateľ		
Základná postupnosť	**Krok**	**Činnosť**	
	1	Používateľ zadá do prehliadača URL systému	
	2	Systém sa načíta a zobrazí prihlasovací formulár	
	3	Používateľ vyplní prihlasovacie údaje a potvrdí	
	4	Systém overí zadané prihlasovacie údaje a pustí používateľa ďalej	
Alternatívna postupnosť	**Krok**	**Činnosť**	
	4.a	Prihlasovacie údaje sú nesprávne. Systém o tom dá vedieť používateľovi a beh pokračuje krokom 3	
	4.b	Systém overí prihlasovacie údaje voči podporovanému single sign in systému a v prípade úspechu pustí používateľa ďalej	
Rozširujúce body			

Tab. 8. UC08- prihlásiť sa do systému

UC 09 Otvoriť súbor dokumentu

Identifikátor	UC09		
Názov	Otvoriť súbor dokumentu		
Opis	Používateľ otvorí súbor dokumentu (samotný dokument) v systémovej aplikácii určenej na daný typ súborov (napr. Word, alebo Acrobat Reader)		
Priorita	Vysoká	**Frekvencia**	Niekoľko krát za hodinu
Vstup. podm.	Používateľ je prihlásený v systéme. Dokument, ktorého súbor chce zobraziť, je zobrazený na ploche		
Výst. podm.	Súbor dokumentu je otvorený v aplikácii na to určenej		
Aktér	Bežný používateľ		
Základná postupnosť	**Krok**	**Činnosť**	
	1	Používateľ vyvolá akciu otvoriť súbor dokumentu	
	2	Systém vyvolá funkciu operačného systému pre otvorenie súboru (alebo funkciu prehliadača)	
	3	Používateľ pokračuje dialógom pre otvorenie súboru podľa prehliadača, alebo operačného systému.	
	4	Používateľ ďalej manipuluje so súborom podľa uváženia (ukladá ho, atď.)	
Alternatívna postupnosť	**Krok**	**Činnosť**	
	2.a	Ak dokument nemá súbor (virtuálny dokument), systém na to upozorní používateľa a beh končí	
	2.b	Ak sa systému nepodarí vyvolať funkciu operačného systému, alebo prehliadača, upovedomí o tom používateľa a beh končí	
Rozš. body			

Tab. 9. UC09- otvoriť súbor dokumentu

UC 10 Prepísať súbor dokumentu novšou verziou

Identifikátor	UC10		
Názov	Prepísať súbor dokumentu novšou verziou		
Opis	Používateľ nahradil starú verziu súboru dokumentu novšou verziou a systém preindexoval tento nový súbor		
Priorita	Stredná	**Frekvencia**	Niekoľko krát za deň
Vstup. podm.	UC rozširuje UC 13 Upraviť metadáta dokumentu v bode 3		
Výstup. podm.	Súbor dokumentu je nahradený novou verziou a korektne zindexovaný		
Aktér	Bežný používateľ		
Základná postupnosť	**Krok**	**Činnosť**	
	3	Používateľ vyvolá akciu nahratie novšej verzie súboru dokumentu. Systém otvorí dialóg a používateľ vyberie zo súborového systému.	
	4	Systém vylúči z indexov starý súbor a zindexuje nový súbor. Systém uloží nový súbor do databázy a nahradí ním starý súbor. Systém dá pokyn podsystému na vyhľadávanie väzieb, aby overil nový súbor a prípadne vyhľadal na ňom väzby (asynchrónne)	

Tab. 10. UC10- prepísať súbor dokumentu novšou verziou

UC 11 Zobrazenie štatistík systému

Identifikátor	UC11		
Názov	Zobrazenie štatistík systému		
Opis	Systém zobrazí používateľovi štatistiky o počte súborov, väzieb, atď.		
Priorita	Nízka	**Frekvencia**	Niekoľko krát za týždeň
Vstup. podm.	Používateľ je prihlásený v systéme.		
Výstup. podm.	Používateľ získal informáciu o štatistikách systému		
Aktér	Bežný používateľ		
Základná postupnosť	**Krok**	**Činnosť**	
	1	Používateľ vyvolá akciu zobrazenie štatistík	
	2	Systém zistí, kedy boli naposledy prepočítané štatistiky. Ak je to po kritickom dátume, tak prepočíta štatistiky.	
	3	Systém zobrazí vybrané štatistiky	
	4	Používateľ si prezrie štatistiky a beh končí	
Alternatívna postupnosť	**Krok**	**Činnosť**	
Rozširujúce body			

Tab. 11. UC11- zobrazenie štatistík systému

UC 12 Hromadné pridanie dokumentov

Identifikátor	UC12		
Názov	Hromadné pridanie dokumentov		
Opis	Používateľ pridá hromadne veľké množstvo dokumentov		
Priorita	Stredná	**Frekvencia**	Niekoľko krát za mesiac
Vstup. podm.	Používateľ je prihlásený v systéme.		
Výstup. podm.	Pridané súbory sú vložené v báze znalostí ako dokumenty a korektne zindexované		
Aktér	Bežný používateľ		
Základná postupnosť	**Krok**	**Činnosť**	
	1	Používateľ vyvolá akciu hromadné pridanie súborov	
	2	Systém otvorí dialóg na hromadné pridanie súborov	
	3	Používateľ vyberie adresár, kde sa hromadne pridávané súbory nachádzajú . Používateľ vyberie, aké typy väzieb sa majú nad dokumentmi vyhľadať hneď po pridaní a potvrdí	
	4	Systém pre každý dokument - Pomocou predindexačných algoritmov sa pokúsi domyslieť si metadáta (autor, kľúčové slová, ...) - Pridá dokument do databázy aj s jeho súborom a uloží ho - Zindexuje súbor dokumentu v indexačnom podsystéme - Systém sa pokúsi vyhľadávať väzby práve pridaného dokumentu podľa zvolených algoritmov vyhľadávania väzieb. - Systém vyhľadané väzby vytvorí a uloží v databáze.	
	5	Systém zobrazí potvrdenie o pridaní dokumentov do bázy znalostí. Systém sa spýta používateľa, či si praje zobraziť pridané dokumenty na ploche	

	6	Používateľ potvrdí a systém zobrazí na ploche pridané dokumenty, pričom každý bude centrálnym dokumentom
Alternatívna postupnosť	**Krok**	**Činnosť**
Rozširujúce body	4.a	Akúkoľvek chybu si systém poznačí (napr. nepodporovaný) a pridávanie chybného dokumentu do bázy znalostí preruší
	5.a	Systém zobrazí zoznam chýb a zoznam dokumentov, ktoré neboli do bázy znalostí pridané.

Tab. 12. UC12- hromadné pridanie dokumentov

UC 13 Upravenie metadát dokumentu

Identifikátor	UC13		
Názov	Upraviť metadáta dokumentu		
Opis	Používateľ upravil metadáta dokumentu, ako názov, meno autora, kľúčové slová a iné.		
Priorita	Stredná	**Frekvencia**	Niekoľko krát za deň
Vstup. podm.	Používateľ je prihlásený v systéme. Dokument, ktorý chce používateľ upraviť je zobrazený na ploche		
Výstup. podm.	Dokument je upravený		
Aktér	Bežný používateľ		
Základná postupnosť	**Krok**	**Činnosť**	
	1	Používateľ vyvolá akciu upraviť dokument	
	2	Systém zobrazí dialóg na úpravu dokumentu	
	3	Používateľ upravuje informácie o dokumente (názov, meno autora, atď.).Používateľ vyvolá akciu uložiť	
	4	Systém uloží zmeny, ktoré zadal používateľ.	
Alternatívna postupnosť	**Krok**	**Činnosť**	
Rozširujúce body	3.	Možné rozšírenie o uploadnutie novšej verzie súboru dokumentu.	

Tab. 13. UC13- upravenie metadát dokumentu

1.4 Funkcionalita UI komponenty plocha

Systém bude zobrazovať v používateľskom rozhraní identifikované väzby medzi dokumentmi. Dokumenty aj ich väzby sú zobrazované na **ploche**. Plocha je dvojrozmerný priestor na ktorom sú rozložené dokumenty a väzby medzi nimi. Zobrazovanie dokumentov je kľúčovou funkcionalitou systému. Nasledovná tabuľka popisuje vlastnosti používateľskej komponenty **plocha.**

Číslo	Vlastnosť	Popis	Priorita
01.1	Automatické rozmiestnenie dokumentov	Dokumenty sa na ploche usporiadajú tak, aby boli podľa možností viditeľné všetky dokumenty aj všetky väzby medi nimi.	vysoká
01.2	Približovanie, vzďaľovanie a posun po ploche	Používateľ nemusí nevyhnutne vždy vidieť celú plochu. Môže sa od plochy vzďaľovať, približovať, ako aj posúvať sa. Požiadavka umožní zobrazenie väčšieho množstva dokumentov.	Stredne vysoká
01.3	Posun a manuálne rozmiestnenie dokumentov	Ak používateľovi nevyhovuje rozmiestnenie dokumentov tak, ako boli automaticky usporiadané, mal by mať možnosť dokumenty dodatočne usporiadať.	stredná
01.4	Otvorenie dokumentu dvojklikom	Ak používateľ dvojklikom klikne na nejaký dokument, ten sa otvorí v aplikácii preddefinovanej operačným systémom	vysoká
01.5	Zobrazenie sily väzby	Každá väzba medzi dokumentmi má určenú svoju silu. Tá musí byť zobrazená vizuálne.	vysoká
01.6	Filtrovanie väzieb podľa sily	Používateľ môže skryť väzby určitej sily. Potom tie, ktoré nedosahujú určitú silu, nebudú zobrazené podobe, ako ani dokumenty týmito väzbami odkazované.	nízka
01.7	Expand a collapse uzlov	Používateľ môže pomocou tlačidla pri dokumente skryť, alebo rozbaliť jeho vetvu. Skrytím vetvy sa skryjú všetky dokumenty odkazované od hlavného	vysoká

		dokumentu iba ním. Rozbalením jeho vetvy sa zobrazia všetky dokumenty ním odkazované. Požiadavku treba spresniť v tom zmysle, či pri expande sa rozbalí iba jedna úroveň, alebo všetky úrovne, alebo len uzly pripojené k expandovanému uzlu určitou silou.	
01.8	Popisy hrán	Jednotlivým hranám by mal systém zobrazovať popisy akého je typu a iné dáta, ktoré sa vzťahujú k vlastnostiam hrany (spresní sa pre každý typ hrany).	stredná
01.9	Zmena centrálneho dokumentu	Používateľ môže zmeniť dokument, od ktorého sú zobrazované všetky väzby. Po zmene centrálneho dokumentu sa zobrazia nové dokumenty, ktoré sú dosiahnuteľné od nového centrálneho dokumentu. Z už zobrazených dokumentov sa stratia len tie, ktoré nie sú dosiahnuteľné od nového centrálneho dokumentu.	nízka
01.10	Explicitné pridanie dokumentu na plochu	Používateľ môže na plochu pridať dokument, ktorý už je v báze znalostí aj bez toho, aby k nemu viedli väzby z centrálneho dokumentu. Toto môže slúžiť napríklad na grafické vytvorenie väzby medzi týmito dokumentmi.	nízka
01.11	Viac centrálnych dokumentov	Používateľ pridá na plochu viac centrálnych dokumentov. Tým pádom sa zväčšia zobrazovacie možnosti plochy a umožní to zoptimalizovať množstvo zobrazených dokumentov.	stredná
01.12	Označenie viacerých väzieb a dokumentov	Používateľ môže označiť na ploche aj viacero väzieb odrazu a to tak, že drží klávesu ctrl.	stredná

01.13	Filtrovanie zobrazených dokumentov	Plocha umožňuje nastaviť filtrovanie zobrazených dokumentov. Filter je možné nastaviť na zobrazené väzby (len určité typy, alebo triedy väzieb, väzby určitej sily), alebo aj obmedziť množinu zobrazených dokumentov na také, ktoré majú na centrálne dokumenty dostatočne silnú väzbu. Väzby a dokumenty nevyhovujúce filtru nebudú na ploche zobrazené, alebo jednoducho zobraziť aj neplatné dokumenty.	stredná

Tab. 14. Funkcionalita UI komponenty plocha

1.5 Automatické vytváranie väzieb medzi dokumentmi

Dôležitá funkcionalita systému funguje na pozadí bez vedomia a dohľadu používateľa. Systém neustále vo voľnom čase analyzuje dokumenty a snaží sa vyhľadávať medzi nimi väzby. Vyhľadané väzby pridáva do systému a ohodnocuje. Systém väzby vyhľadáva najmä na základe

- Autorov
- Referencií medzi dokumentmi (jeden odkazuje druhý)
- Kľúčových slov
- Predchádzajúcich akcií používateľa (vytvorenie väzby, vymazanie väzby)

Systém môže niektoré väzby, ktoré vytvoril v minulosti na základe novej analýzy zrušiť, respektíve prehodnotiť ich silu. Predmetom prípadu použitia „Konfigurácia automaticky vytváraných väzieb“ je nastavenie, ktoré algoritmy sa majú použiť, v akom rozsahu a aké sú parametre týchto algoritmov. Jednotlivé algoritmy, ich parametre a parametre väzieb, ktoré budú nimi vyhľadané budú vyšpecifikované po podrobnejšej analýze tejto oblasti.

1.6 Požiadavky, ktoré nebudú v projekte riešené

Prístupové práva k dokumentom	Prístupové práva k dokumentom by používateľovi filtrovali iba tie dokumenty, ktoré má právo vidieť. Jednoduché prístupové práva nedávajú pri takomto systéme zmysel (práva prideľované na osobu používateľa). Aby sme mohli nasadiť zložitejší systém prístupových práv bolo by nutné buď ho vytvoriť (veľký projekt sám o sebe), alebo ho s nejakým existujúcim systémom prideľovania práv zintegrovať. Napríklad systém, ktorý zachytáva organizačnú štruktúru organizácie tak, aby sme mohli práva prideľovať na celú organizačnú jednotku (oddelenie) a nie iba pre jednotlivcov.
Vlastníctvo dokumentov, práva na editáciu a mazanie dokumentov a väzieb	Vlastnícke práva, alebo aspoň viacero rolí používateľov by zabránilo prípadným neodborným zásahom do bázy znalostí zo strany nekompetentných používateľov. Problém je, že práve činnosť všetkých používateľov je potrebná na efektívne vytvorenie bázy znalostí. Riešenie tejto problematiky je tak pravdepodobne nad rámec projektu.
Správa verzií	Dokumenty sa priebežne v reálnom svete menia – vytvárajú sa ich nové verzie. Každá nová verzia by mala byť uploadnutá do bázy znalostí.

dokumentov	Pamätanie starších verzií dokumentov a vôbec čo i len analýza zložitej logiky, ako sa správať k verziám dokumentov je nad rámec tímového projektu, preto v našom projekte nové verzie súborov jednoducho nahradia staršie verzie bez zisťovania zmien a pamätania verzií.
3D zobrazenie dokumentov a ich väzieb	Experimentovanie s 3D zobrazením je nad rámec projektu, ale môže byť zaujímavým námetom pre samostatný projekt v budúcnosti.
Zobrazenie, počítanie a evidovanie informačnej hodnoty dokumentu	Každému dokumentu by malo byť možné vypočítať jeho informačnú hodnotu- rating. Tento by ovplyvňovali najmä používatelia tým, že by prečítané dokumenty hodnotili z pohľadu hodnoty ich obsahu. Takáto funkcionalita nebude ani analyzovaná ani implementovaná.

Tab. 15. – neimplementované požiadavky

2. Analýza

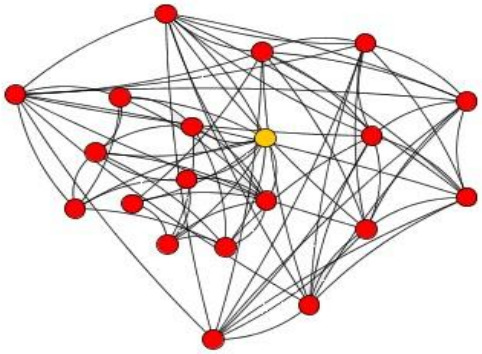
2.1 Analýza vizualizačných knižníc

Úvod

Náš projekt má grafické zobrazenie vzťahov medzi dokumentmi priamo vo svojom názve a preto sa tejto problematike musíme dôkladne venovať. Za najprehľadnejší spôsob zobrazenia vzťahov považujeme graf, či už orientovaný alebo neorientovaný. Tento musí rozlišovať typy a silu asociácií medzi uzlami, ktoré predstavujú spracovávané dokumenty, napríklad rôznym tvarom alebo farbou hrán. Aby sme užívateľovi obohatili prácu s dokumentmi, poskytovaná vizualizácia musí byť interaktívna. K dispozícii by mala byť možnosť otvárania dokumentov, ich pridávanie a odoberanie alebo napríklad zmena vzťahu medzi dokumentmi. Pre náš projekt sme zvažovali viacero knižníc:

- JUNG
- Prefuse
- JGraph
- JGraphT

Knižnica JUNG



Grafická knižnica vytvorená v akademickom prostredí.

Obr.2. - Príklad jednoduchej vizualizácie knižnicou Jung

+	-
Pravdepodobne prepracovanejšia ako Prefuse, čo sa týka algoritmov.	Neestetická vizualizácia.
Najnovšia verzia je 2.0	
Je zadarmo	

Tab. 16. - Plusy a mínusy JUNG

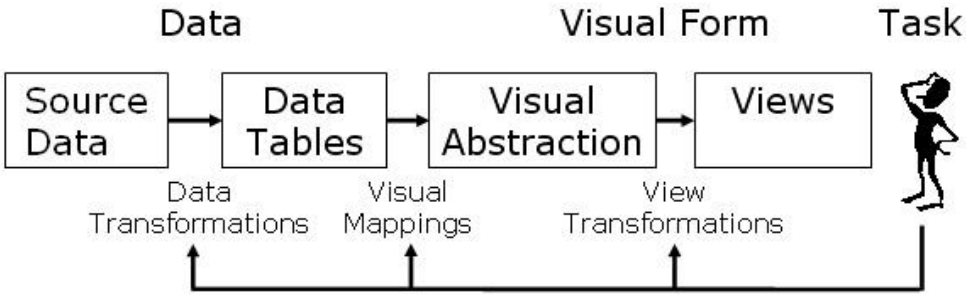
Knižnica Prefuse

Knižnica Prefuse obsahuje súbor nástrojov pre tvorbu interaktívnych vizualizácií, ktorých zdrojom môžu byť ľubovoľné dáta. Je určená pre platformu Java, v ktorej je ľahko integrovateľná do swing aplikácií alebo appletov. Hlavným cieľom Prefuse je v čo najväčšej miere zjednodušiť a zefektívniť proces reprezentácie spracovávaných dát a vzťahov medzi nimi.

Pre potreby nášho projektu ponúka nasledujúcu funkcionalitu:

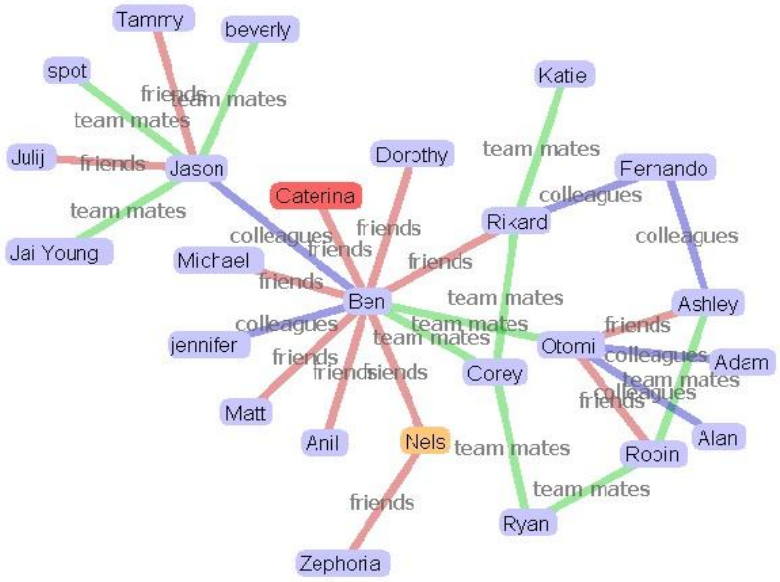
- vizualizácia prostredníctvom grafu
- použitie hrán rôznej hrúbky, farby a tvaru
- meniteľné popisy k hranám aj uzlom grafu
- interaktívne prvky grafu – odchytávanie eventov
- simulácia pôsobenia síl medzi uzlami grafu –> prehľadnejšie zobrazenie grafu
- zmena pohľadu zobrazenia, približovanie a oddiaľovanie zobrazenia
- viacero spôsob zobrazenia – napríklad celok + detail
- prehľadávanie v zobrazovanom texte
- podpora pre vykonávanie SQL dotazov a následné mapovanie výsledkov na objekty knižnice

Prefuse funguje na základe vzoru „*information visualization reference model*“, ktorý rozdeľuje proces vizualizácie na viacero krokov. Zdrojové dáta sa mapujú do tzv. dátových tabuliek (z angl. data tables), ktoré tvoria základ vizualizácie. Z týchto zdrojov je následne vytvorená vizuálna abstrakcia modelujúca zobrazované vlastnosti objektov ako sú pozícia, tvar alebo farba. Táto abstrakcia slúži na vytvorenie interaktívneho zobrazenia zdrojových dát, v ktorom môže užívateľ meniť vlastnosti ľubovoľnej z predošlých úrovní.



Obr. 3. - Diagram procesu vizualizácie prostredníctvom knižnice Prefuse

Ukážka vizualizácie:



Obr. 4 - Obrázok znázorňuje rôzne označenia aj zafarbenie hrán

+	-
kvalitne spracovaná vizualizácia	beta verzia, nepredpokladá sa ďalší vývoj
možnosť simulácie pôsobenia síl medzi uzlami	veľmi nízka úroveň dokumentácie – len javadoc

	k používanému API + fórum s nízkou návštevnosťou
jednoduché používanie zapracovaných súčastí	menšia množina použiteľných vzorov pre hrany aj uzly
interaktívne prvky grafu	pomerne náročná modifikácia existujúcich komponentov
prístup k zdrojovým kódom	

Tab. 17 – Plusy a mínusy Prefuse

Knižnica JGraph

JGraph je silná open-source knižnica vytvorená v Jave. Je vyvíjaná už osem rokov a jej najnovšia verzia je 5.9, takže možno predpokladať, že je na vysokej úrovni. Donedávna bol JGraph komerčnou knižnicou, avšak v súčasnosti zmenil na BSD licenciu, čo znamená že je zadarmo. Je určená na vizualizáciu a spracovanie grafov. Je to Swing kompatibilný komponent takže je ľahko integrovateľný do Java aplikácií. Obsahuje API na vizualizáciu, interakciu, automatický layout a analýzu grafu.

Pre naše potreby poskytuje nasledovnú funkcionalitu:

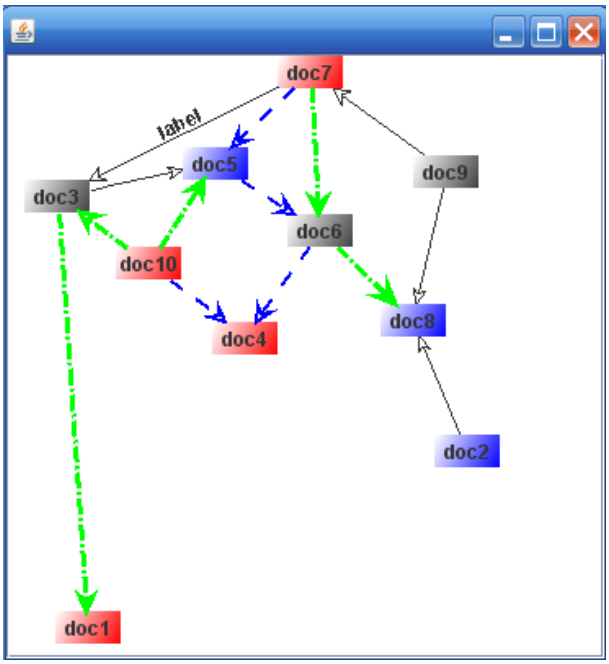
- spracovanie grafu a jeho následne upravovanie
    - umožňuje editovať zobrazený graf – pridávanie, úprava a tribútov a mazanie hrán
- vizualizáciu grafu
    - rôzne tvary a farby uzlov a hrán
    - rôzne patterny hrán, rôzna hrúbka hrán
    - umožňuje popis (label) hrán a vrcholov
- zooming, drag and drop, pohybovanie grafu, undo, redo, selekciu v grafe, grid
- umožňuje automatické layouty

+	-
kvalitná a rozsiahla dokumentácia	nemá automatický pohyblivý layout

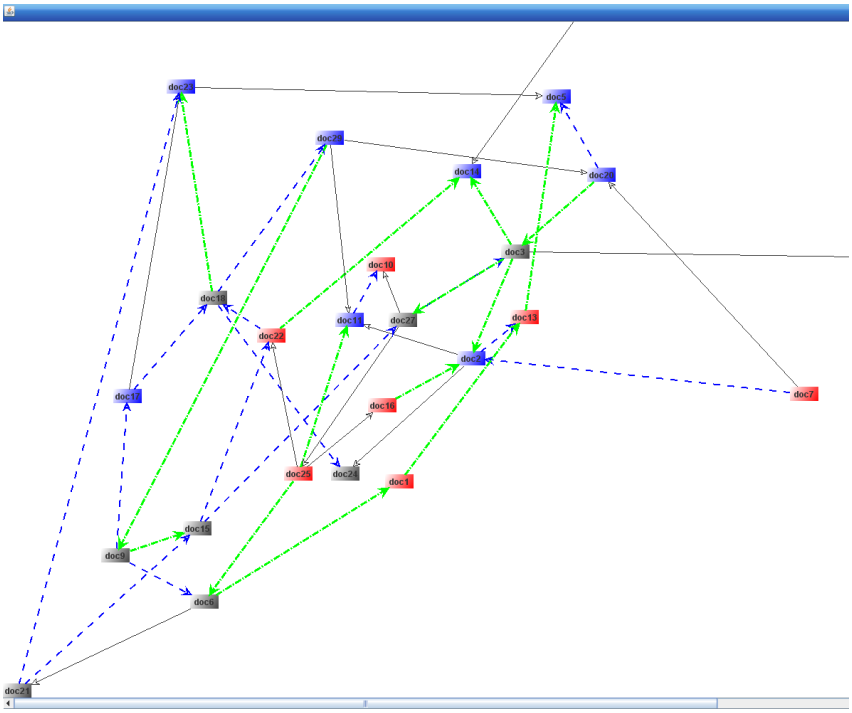
bohaté API (zooming, undo, redo, drag and drop, ...)	len statické layouty (dajú sa parametrizovať)
dokáže takmer všetko, čo potrebujeme (farby a patterny hrán, rôzne hrúbky hrán, labely, editáciu grafu, ...)	oproti knižnici Prefuse layout nevyzerá až tak dobre
	pravdepodobne bude náročné nájsť vhodné parametre na optimálny layout pre rôzne grafy, ktoré budeme chcieť zobrazovať

Tab. 18 – Plusy a mínusy JGraph

Statické layouty sa dajú parametrizovať, avšak zistili sme, že tieto parametre sú pomerne citlivé na zmenu veľkosti grafu. My budeme chcieť vizualizovať grafy rôznych veľkostí, preto bude pravdepodobne náročné nájsť vhodné parametre, aby rozloženie vrcholov a hrán v grafe nepôsobilo neesteticky. Na obrázkoch nižšie sú zobrazené dva rôzne veľké grafy s použitím tých istých parametrov layoutu. Na prvom obrázku je vizualizácia krajšia ako na druhom, kde sú zbytočne predĺžené dĺžky hrán.



Obr. 5 - Rozloženie grafu pri 10 vrcholoch použitím JGraph



Obr. 6 - Rozloženie grafu pri 30 vrcholoch s tými istými parametrami layoutu

Knižnica JGraphT

Grafická knižnica určená hlavne na spracovanie grafov, grafové algoritmy a dátové štruktúry. Vizualizácia je jej slabšou stránkou – využíva prvky JGraph.

+	-
Rýchlosť, zvládne spracovať niekoľko miliónov uzlov a hrán.	Určená hlavne na grafové algoritmy, slabšia podpora vizualizácie.
Je zadarmo.	Najnovšia verzia je ešte len 0.8.

Tab. 19. - Plusy a mínusy JGraphT

2.2 Analýza algoritmov a nástrojov na hľadanie väzieb medzi dokumentmi

Úvod

Cieľom nášho tímového projektu je prehľadne zobraziť sémantické vzťahy medzi dokumentmi. Nachádzanie týchto vzťahov je netriviálna činnosť, ktorá je zároveň kľúčová pre náš projekt. Problém je, že väčšina dokumentov v rámci bázy znalostí je neštruktúrovaného alebo slabo štruktúrovaného charakteru, čo znemožňuje hľadanie väzieb priamo nad dokumentmi. Je potrebné všetky dokumenty v báze znalostí spracovať a väzby vyhľadávať až nad takto spracovanou bázou znalostí. Na spracovanie dokumentov do formy, v ktorej je už možno vyhľadávať vzťahy a väzby existuje viacero nástrojov.

Analýza indexovacích nástrojov

Na prepojenie dokumentov potrebujeme poznať ich charakteristické dáta. Jedná sa metadáta ako sú autor, názov dokumentu, kľúčové slová a referencie na iné dokumenty. Na získanie týchto dát sa používajú rôzne algoritmy a nástroje, pomocou ktorých je potom vyhľadávanie dokumentov omnoho jednoduchšie a rýchlejšie.

Lucene

OpenSource projekt Apache Lucene poskytuje indexovanie a vyhľadávanie súborov. Je založený na Jave. Lucene sa skladá z niekoľkých častí. Najdôležitejšou je indexátor

(IndexWriter), ktorý pomocou vybraného analyzátora získa potrebné parametre o dokumente, ktorý chce používateľ zindexovať a naplní týmito metadátami určené polia (Fields). Každému poľu samozrejme môžeme určiť názov a obsah, no dôležitejšie je, že môžeme aj zadefinovať čo sa má spraviť s jeho obsahom (tokenizovanie, indexovanie, celkové uloženie). Na analyzovanie textu sa používajú 3 základné druhy analyzátorov:

- *SimpleAnalyzer* – používa iba Tokenizer, ktorý skonvertuje všetok vstup do malých písmen (angl. lower case).
- *StopAnalyzer* – používa rovnaký konvertor do malých písmen, no zároveň aj filtruje vstup podľa tzv. stopových slov, teda slov, ktoré nechceme aby sa indexovali (napr. the, that, a, b, use, atď.).
- *StandardAnalyzer* – používa konvertor do malých písmen aj filtrovanie stopových slov, navyše sa však snaží o prenesenie slova do základného tvaru (napr. vynecháva dokončenie apostrofom [‘s]).

Po vytvorení indexu môžeme zadať vyhľadávací dotaz (angl. query), ktorý rovnaký analyzátor upraví do rovnakého tvaru ako bol upravený vstup do indexátora. Lucene vyhľadá (IndexSearcher) zodpovedajúce dokumenty a vráti nám tzv. hity (angl. Hits), čo je vlastne zoznam dokumentov zoradený podľa ich ratingu relevancie (angl. Score).

Lucene Apache Tika

OpenSource podprojekt projektu Apache Lucene. Tika je toolkit na detekciu a extrakciu metadát a štruktúrovaný text z rôznych dokumentov, ktorý používa existujúce knižnice na parsovanie. Podporuje formáty ako PDF, DOC, XLS, PPT, HTML, XML, JAR, RTF, TAR, ZIP, MP3, MIDI, WAV atď.

Compass

OpenSource projekt postavený ako nadstavba nástroja Lucene, ktorý uľahčuje a zjednodušuje integráciu vyhľadávania do ľubovoľnej java aplikácie. Oproti nástroju Lucene má mnoho výhod. Napríklad má jednoduchšie API, dokáže vykonať vyhľadávanie nad všetkými poliami, má podporu transakcií s externými transakčnými manažérmi ako JTA, Spring alebo ORM, má podporu technológie Hibernate a dokáže uchovávať index v databáze. Z týchto mnohých výhod však samozrejme vyplývajú aj nevýhody, ktorými sú veľkosť a zložitosť.

Zilverline Search Engine

Zilverline by sa dal označiť ako Reverse Search Engine. Zilverline je indexátor a vyhľadávač, ktorý ponúka webový prístup k osobným dátam alebo dátam intranetu. Je dosť podobný Google Desktop, no pracuje na základe Lucene. Zilverline podporuje formáty PDF, Word,

Excel, Powerpoint, RTF, TXT, java, CHM, rovnako ako aj zip, rar, a mnoho ďalších archívov. Zilverline je postavený na Jave, Lucene a Spring. K jeho spusteniu potrebujete Servlet Engine, ako je napr. Tomcat. Základom je vytváranie kolekcií (sada súborov a adresárov v adresári), ktoré sa indexujú a vyhľadáva sa v nich.

BDDBot

BDDBot je webový robot, vyhľadávací nástroj a webový server napísaný v Jave. Je to príklad k výučbe ako tvoriť veľmi jednoduchý vyhľadávací nástroj. Dokáže indexovať len súbory typu HTML a plain/text. BDDBot prichádza s vlastným zabudovaným webovým serverom. Jeho indexy sú veľmi malé (cca 10% obsahu textu). Napriek tomu je veľmi malý (len niečo okolo 200 KB). Tento projekt skončil a už dlhšiu dobu sa nevyvíja ďalej.

Analýza algoritmov

Esenciálnou časťou celého systém je nachádzanie sémantických väzieb medzi jednotlivými dokumentmi. Počas analýzy sme identifikovali tri rôzne typy väzieb. A to väzby medzi dokumentmi založené na zhodnosti niekoľkých s množiny kľúčových slov každého dokumentu, ďalej väzby založené na spoločnom autorovi dokumentov a väzby založené na referenciách medzi dokumentmi. Toto sú typy väzieb, ktoré sú vytvárané automaticky pomocou algoritmov, ktoré budú implementované. Ale je vhodné vytvoriť aj ďalší typ väzby a to väzbu vytváranú používateľom. Tento typ väzby je vhodný najmä z dôvodu, že algoritmy môžu prehliadnuť dôležitý a významný sémantický vzťah medzi dvojicou dokumentov. Alebo v prípade, že užívateľ považuje dva dokumenty za navzájom relevantné napriek tomu, že nemajú spoločné žiadne z vlastností použitých na identifikáciu väzieb. Je vidno že sily jednotlivých typov identifikovaných väzieb sú rozdielne, preto je vhodné zaviesť určité pomery, ohodnotenie našich väzieb. Napríklad väzba založená na piatich zhodných kľúčových slovách je totožná s väzbou založenou na jednom spoločnom autorovi, respektíve na jednej referencii medzi dokumentmi. Toto je iba príklad, sily jednotlivých väzieb bude môcť v systéme nastavovať používateľ typu administrátor.

Typy väzieb

Keyword

Väzba typu keyword, je vytvorená na základe určitého počtu rovnakých kľúčových slov v dokumentoch. Je dôležité, aby v zozname kľúčových slov každého dokumentu boli naozaj kľúčové a relevantné slová, ktoré jasne definujú obsah tohto dokumentu. Teda treba definovať

nejakú reštrikciu, a iba slová, ktoré prejdú touto reštrikciou, môžu byť považované za kľúčové. Jednoduchý spôsob ako obmedziť množstvo kľúčových slov je pomocou white list a black list zoznamov. Tento typ väzby je obojsmerný, to znamená, že netreba rozlišovať referencujúci a referencovaný dokument.

Author

Väzba typu author je vytvorená na základe jedného alebo viacerých rovnakých autorov dvoch rozličných dokumentov. Pri tomto type väzby je dôležité rozlišovať ak dvaja rôzni autori majú rovnaké mená. A takisto treba vyriešiť problém s rôznym zápisom mena, napríklad zápisy Jožko Mrkvička, Mrkvička Jožko, J. Mrkvička a pod. sú všetky rôzne spôsoby zápisu mena tej istej osoby. Väzba typu author je takisto ako väzba typu keyword obojsmerná.

Reference

Väzba typu reference je vytvorená na základe odkazov v jednom dokumente na druhý dokument napríklad v časti Použitá literatúra. Tento typ väzby je iba jednosmerný, teda treba ho zobraziť iným spôsobom ako dva predchádzajúce.

Spracovaný dokument

Algoritmy, pomocou ktorých vyhľadávame väzby medzi dokumentmi nemožno použiť priamo nad dokumentom vo formáte v akom sa nachádza v súborovom systéme, ale treba ho spracovať do formy, ktorá bude vhodná ako vstup pre algoritmy. Na základe typov väzieb, ktoré sa budú vyhľadávať potrebujeme minimálne pre každý dokument poznať jeho autora, kľúčové slová, a dokumenty na ktoré sa odkazuje. Formátom spracovania dokumentov sa bližšie venujeme v časti Lucene.

Algoritmus

Prvá a najjednoduchšia forma algoritmov, ktorú implementujeme v rámci prototypu je postavená na jednoduchom porovnávaní textových reťazcov. Avšak už v prototype bude vytvorený kontajner na algoritmy, ktorý bude poskytovať vhodné rozhranie ostatným častiam systému takže bude jednoduché pridávať nové a zložitejšie algoritmy.

Analýza toolkitov text/data miningu

Spomínané nástroje dokážu iba zindexovať dokumenty a full textovo ich prehľadávať. Samotné analyzovanie dokumentov a hľadanie vzťahov medzi nimi je realizované pomocou rôznych metód text miningu alebo data miningu. Tieto metódy sú už implementované v rôznych toolkitoch. Ale v niektorých prípadoch je vhodné implementovať ich ručne.

Lingpipe

Lingpipe je súprava nástrojov na spracovanie prirodzeného jazyka, je napísaná v jave. Medzi jej hlavné funkcie patria:

- Tokenizácia
- Nachádzanie viet
- Nachádzanie entít
- Klasifikácia
- Zhlukovanie
- POS tagging

Je to jeden z najvyspelejších a často používaných nástrojov na spracovanie prirodzeného jazyka (NLP). Jeho prednosťami je najmä rýchlosť, stabilita a škálovateľnosť. Ďalšou výhodou jeho použitia je množstvo tutoriálov na stránke tohto projektu. Technicky nie je open source, ale je šírená aj so zdrojovým kódom.

Gate

Tiež jeden z často používaných toolkitov na text mining a získavanie informácií. Je možné ho používať ako samostatný nástroj, čo mi nechceme. Ale možno ho používať aj ako SDK napísane v jave. Je to komplexný nástroj. Je open source.

Weka

Je kolekcia algoritmov strojového učenia sa používaných v data miningu. Je to jeden z najpopulárnejších frameworkov na klasifikáciu textov. Obsahuje implementácie rôznych algoritmov, ako napríklad Naive Bayes, SVM. Má dobrú dokumentáciu, a široké využitie.

Jtmt

Je to projekt jedného človeka, jedinou dokumentáciou je iba blog autora. Jedinou výhodou tohto projektu by mohlo byť, že nie je veľmi rozsiahly, takže zvládnuť ho by nemal byť väčší problém. Je takisto open source.

3. Architektúra a návrh

3.1 Zdôvodnenie navrhnutej architektúry

Systém si architektonicky môžeme rozdeliť na viacero vrstiev. Najpodstatnejšie je oddelenie zobrazovacej logiky od celého systému, aby bolo možné vyvíjať a aj nasadiť viacero nezávislých používateľských rozhraní. Z tohto dôvodu je zvolená klient-server architektúra. Interafce medzi používateľským rozhraním a serverom musí byť veľmi dobre definovaný a zdokumentovaný.

Na serveri bude bežať objektový model, ktorý zachytáva dokumenty a vzťahy medzi nimi. Samotné dokumenty (súbory) musia byť zindexované, aby bolo možné v nich fulltextovo vyhľadávať

Na serveri na pozadí budú bežať skryté vlákna, ktoré sa celý čas venujú analyzovaniu dokumentov a vyhľadávaniu väzieb medzi nimi. Tieto algoritmy sú dôležitou súčasťou systému a ich zdokonaľovanie v budúcnosti bude dôležité pre zvyšovanie pridanej hodnoty systému. Preto aj ich správa a nasadzovanie nových algoritmov a nových verzií starých algoritmov by malo byť nezávislé od ostatných častí systému.

3.2 Zdôvodnenie výberu vizualizačnej knižnice

Knižnice JGraph a Prefuse majú svoje výhody aj nevýhody. JGraph má na rozdiel od Prefuse kvalitnú dokumentáciu, takže nebude problém sa naučiť s ňou pracovať. Prefuse má zase kvalitnejšie spracovanú vizualizačnú stránku. Výber správnej knižnice je pre náš projekt kľúčovou záležitosťou, preto sme sa rozhodli pri vytváraní prototypu použiť obidve knižnice, pričom sa neskôr na základe získaných skúseností rozhodneme, ktorú použijeme vo finálnej verzii aplikácie.

3.3 Návrh algoritmu vyhľadávania dokumentov

Na zindexovanie dokumentov a full text prehľadávanie sa v prototype bude používať samostatný nástroj Lucene. Ten vo svojej základnej verzii dokáže indexovať len súbory typu plain/text. V prototype teda vytvoríme vlastné API, ktoré využijeme pre vývoj ostatných častí

projektu a ich vzájomnú integritu. V zimnom semestri sa zameriame na indexovanie HTML a TXT dokumentov a následne na ich vyhľadávanie pomocou jednoduchých dotazov (autor, kľúčové slovo a názov dokumentu).

Na vyriešenie problémov týkajúcich sa diakritiky sme mali na výber metódy dvoch tried (ISOLatin1AccentFilter a ASCIIFoldingFilter). Použijeme metódy triedy ASCIIFoldingFilter nakoľko ISOLatin1AccentFilter sa už neodporúča používať a v novej verzii nástroja Lucene sa už ani nebude nachádzať.

Indexovanie DOC a PDF súborov pomocou nástroja Lucene docielime použitím knižníc, ktoré sú odporúčané priamo na stránkach tohto nástroja. Na spracovanie dokumentov Word DOC použijeme knižnicu Apache POI a na PDF súbory knižnicu PDFBox.org. Pomocou týchto knižníc (ktoré majú podporu integrácie Lucene Search Engine) bude môcť aplikácia získať metainformácie z jednotlivých typov dokumentov a plnohodnotne ich tak zindexovať.

V prototype sa pri chýbajúcich metadátach HTML súboru nepoužíva žiaden text mining toolkit. Je však potrebné uviesť, že v ďalších fázach vývoja ich pravdepodobne použijeme, pretože dokument musíme zindexovať na základe nejakých informácií.

V letnom semestri sa tiež zameriame na zakomponovanie nástroja Compass, ktorý nám ako nadstavba Lucene-u poskytne mnoho výhod (spomínaná podpora transakcií, Hibernate, Spring a indexovania do databázy).

3.4 Návrh kontajnera algoritmov vyhľadávania väzieb

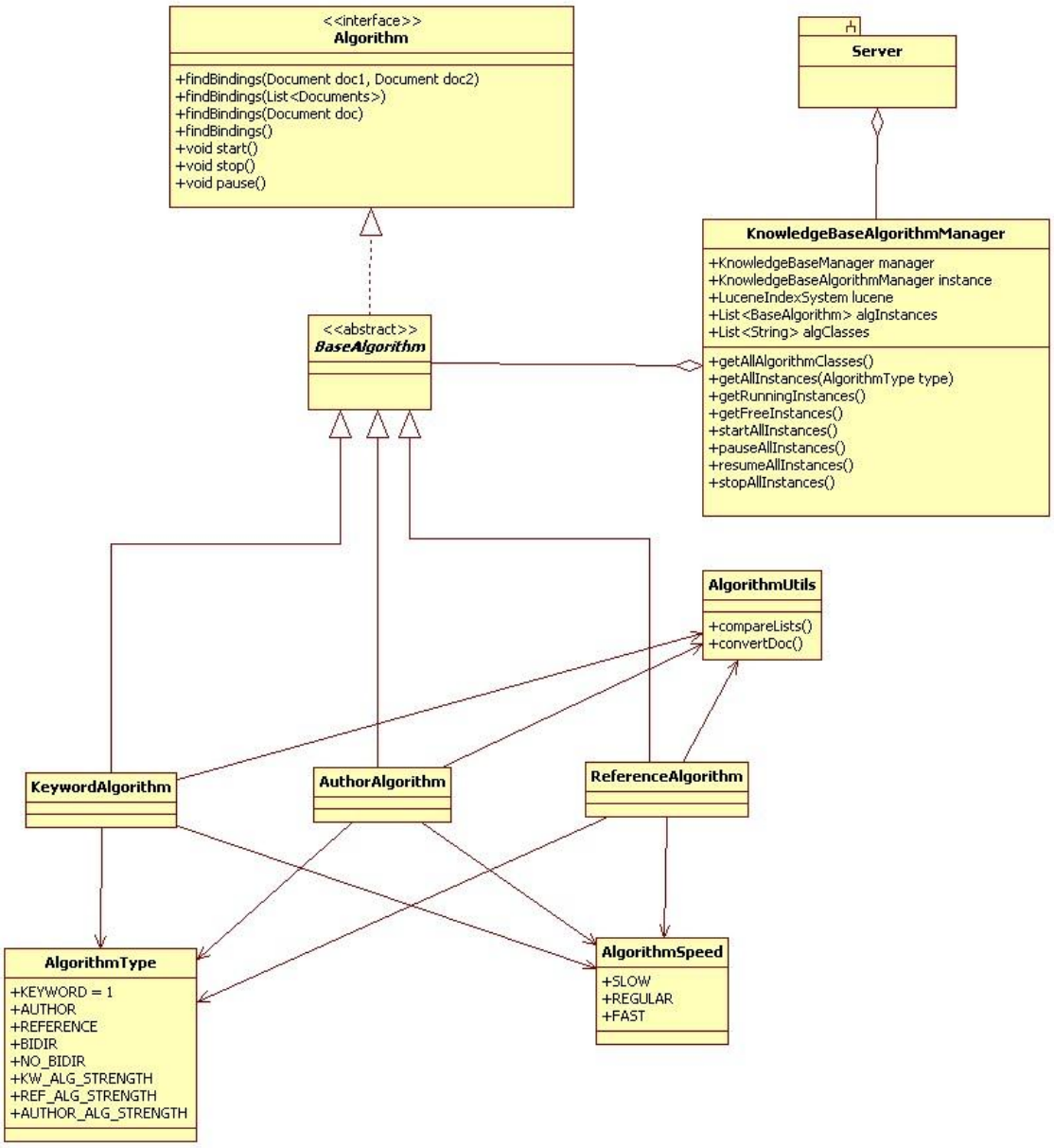
Reštrikcia keywordov - pomocou white list zoznamov a black list zoznamov, ktoré podporuje priamo Lucene a Compass.

Mená pomocou objektov typu User, teda každý autor musí byť zároveň používateľom systému. Alebo pomocou hashovacích máp.

Je rozdiel medzi silou väzby vytvorenej na základe jedného autora, alebo jednej referencie a jedného kľúčového slova. Teda treba definovať, akú silu má každá z typov väzieb. Zároveň si treba uvedomiť že sila väzieb musí byť tranzitívna. To znamená, že ak sila väzby na základe jedného autora je definovaná ako rovná sile väzby na základe troch kľúčových slov. Sila väzby jednej referencie je rovná sile väzby dvoch kľúčových slov. Potom sila väzby dvoch referencií je rovná sile väzby na základe troch autorov.

Všetky jednotlivé implementácie algoritmov budú zabalené do kontajneru, ktorý bude poskytovať jednotné API pomocou ktorého bude možné ovládať a nastavovať jednotlivé inštancie a triedy algoritmov. Takýto kontajner nám takisto umožní pridávať nové implementácie algoritmov bez zmeny kódu v iných častiach systému.

V rámci doménového modelu z algoritmami súvisí trieda väzby- Binding a jej podtriedy. Tá je popísaná v časti návrhu doménového modelu.



Obr. 7 Diagram tried Kontajnera algoritmov

Interface Algorithm

Je rozhranie, od ktorého dedia všetky algoritmy. Sú v ňom deklarované metódy, pomocou ktorých sa bude k algoritmom pristupovať. Dôležitosť jednotného rozhrania spočíva práve v tom, že ku všetkým algoritmom sa bude pristupovať rovnakým spôsobom. Všetky metódy sú popísane neskôr v dokumente.

Abstract class BaseAlgorithm

Abstraktná trieda BaseAlgorithm takisto dedí z rozhrania Algorithm, pričom niektoré z metód, spoločné pre viacero algoritmov, aj implementuje. Napríklad je v nej implementovaný prechod zoznamom dokumentov v báze znalostí a podobne. Takisto je v nej implementovaný aj všeobecný lifecycle algoritmu.

Class KeywordAlgorithm

Trieda KeywordAlgorithm je konkrétna implementácia algoritmu na vyhľadávanie väzieb, táto je konkrétne založená na podobnosti dokumentov vzhľadom na kľúčové slová.

Class ReferenceAlgorithm

Trieda KeywordAlgorithm je konkrétna implementácia algoritmu na vyhľadávanie väzieb, táto je konkrétne založená na vzťahu medzi dokumentmi vzhľadom na referencie medzi nimi.

Class AutorAlgorithm

Trieda KeywordAlgorithm je konkrétna implementácia algoritmu na vyhľadávanie väzieb, táto je konkrétne založená na podobnosti dokumentov založenej na autoroch dokumentov.

Class AlgorithmType

Obsahuje konštanty, ktoré s týkaju typu algoritmov.

Class AlgorithmSpeed

Obsahuje konštanty, ktoré sa týkajú rýchlosti algoritmov.

Class AlgorithmUtils

Obsahuje metódy, ktoré využíva viacero implementácií algoritmov, ale konceptuálne sa nehodia do abstraktnej triedy BaseAlgorithm. Napríklad konvertovanie dokumentu a podobne.

Class KnowledgebaseAlgorithmManager

Základná trieda kontajneru algoritmov. Táto trieda je zodpovedná za manažovanie a poskytovanie logiky ohľadom algoritmov, ktoré hľadajú väzby medzi dokumentmi. Manažment obsahuje štartovanie, zastavovanie algoritmov, ktoré sú spustené na pozadí a podobne. Trieda takisto obsahuje API na priame volanie algoritmov. Viac k tejto triede je popísané ďalej v tomto dokumente.

Algorithm

V nasledovnej kapitole sú popísane jednotlivé metódy rozhrania Algorithm.

List<Bindings> findBindings()

Metóda, ktorá hľadá väzby medzi všetkými dokumentmi, ktoré sa nachádzajú v báze znalostí, navzájom. Jej návratová hodnota je list nájdených väzieb. V prípade, že žiadne väzby neboli nájdene je to null.

List<Binding> findBindings(List<Document> doc)

Metóda, ktorá hľadá väzby medzi dokumentmi navzájom, ktoré dostane ako parameter vo forme zoznamu dokumentov. Jej návratová hodnota je list nájdených väzieb. V prípade, že žiadne väzby neboli nájdene je to null.

List<Binding> findBindings(Document doc, List<Document> docs)

Metóda, ktorá hľadá väzby medzi ktoré ma dokument doc, so všetkými ostatnými, ktoré sú dodané metóde ako parameter vo formáte zoznamu dokumentov. Jej návratová hodnota je list nájdených väzieb. V prípade, že žiadne väzby neboli nájdene je to null.

Binding findBindings(Document doc1, Document doc2)

Metóda, ktorá slúži na zistenie, či medzi dvoma dokumentmi existuje väzba. Vracia objekt typu Binding alebo null v prípade, že žiadna väzba nebola nájdená.

void run()

Metóda, ktorá spustí inštanciu algoritmu, konkrétne spustí metódu findBindings().

void stop()

Metóda zastaví vytváranie väzieb inštanciou nad ktorou je zavolaná.

void pause()

Metóda zastaví vytváranie väzieb, ale zapamätá si dokument na ktorom skončila.

void resume()

Metóda spustí vyhľadávanie väzieb, ak na zastavenie bola použitá metóda pause, začne od posledného zanalyzovaného dokumentu.

KnowledgebaseAlgorithmManager

V nasledovnej kapitole sú popísane jednotlivé metódy a premenné triedy KnowledgebaseAlgorithmManager

List<BaseAlgorithm> algInstances

Zoznam všetkých spustených algoritmov.

algClasses

Zoznam všetkých tried algoritmov.

List<XXXX> getAllClasses()

Metóda vráti všetky triedy algoritmov.

List<BaseAlgorithm> getAllInstances

Metóda vráti všetky vytvorené inštancie algoritmov.

List<BaseAlgorithm> getRunningInstances

Metóda vráti len bežiace inštancie algoritmov.

List<BaseAlgorithm> getFreeInstances

Metóda vráti zastavené alebo pozastavené inštancie algoritmov.

void startAllInstances()

Zavolá metódu run(), resume() nad všetkými zastavenými a pozastavenými algoritmami

void stopAllInstances ()

Zastaví vykonávanie všetkých vytvorených inštancií algoritmov.

void pauseAllInstances ()

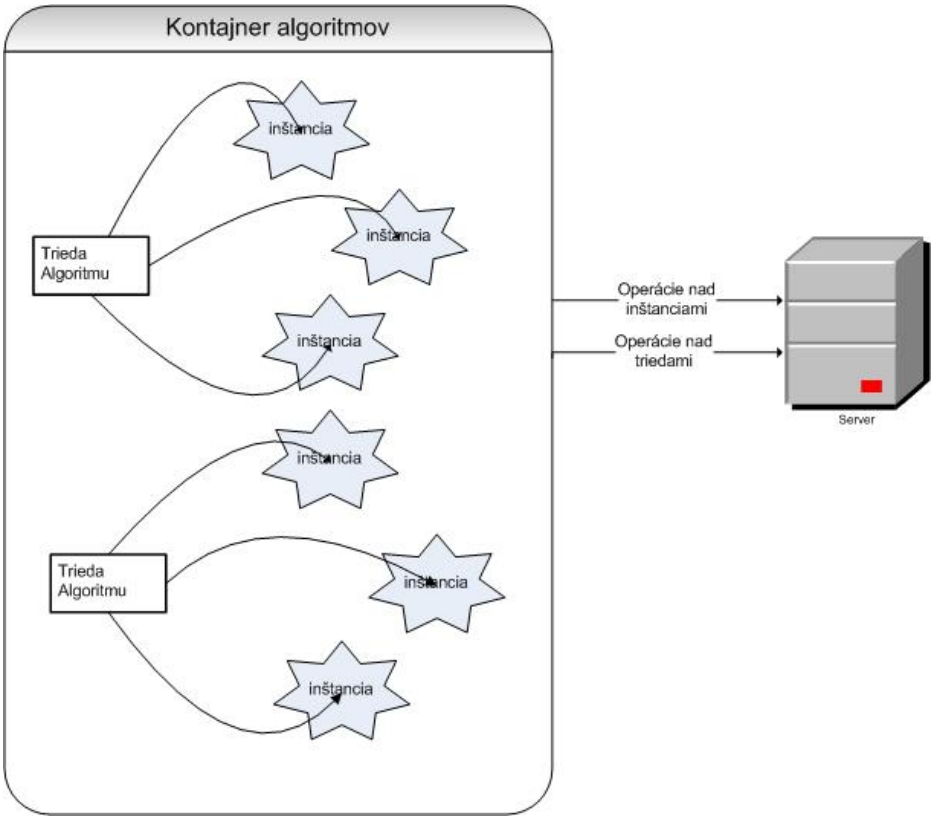
Zastaví vykonávanie všetkých vytvorených inštancií algoritmov, ale zapamätá si, kde zastavil.

void resumeAllInstances ()

Nanovo spustí všetky pozastavené inštancie.

Popis funkcionality kontajnera algoritmov

Jednotlivé inštancie tried algoritmov budú bežať v samostatných vláknach na pozadí. Pričom pomocou API ktoré poskytuje KnowledgebaseAlgoritmhManager je možné ich ovládať a riadiť. Cieľom kontajnera algoritmov a jeho rozhrania je poskytnúť používateľovi tohto API funkcionalitu typu: Chcem spustiť 10 inštancií tejto triedy s takouto konfiguráciou a 3 inštancie inej triedy s takouto konfiguráciou. Všetka ostatná logika kontajneru bude skrytá pred klientom. Túto predstavu znázorňuje náčrtok.



Obr. 8 Koncepcia kontajnera algoritmov

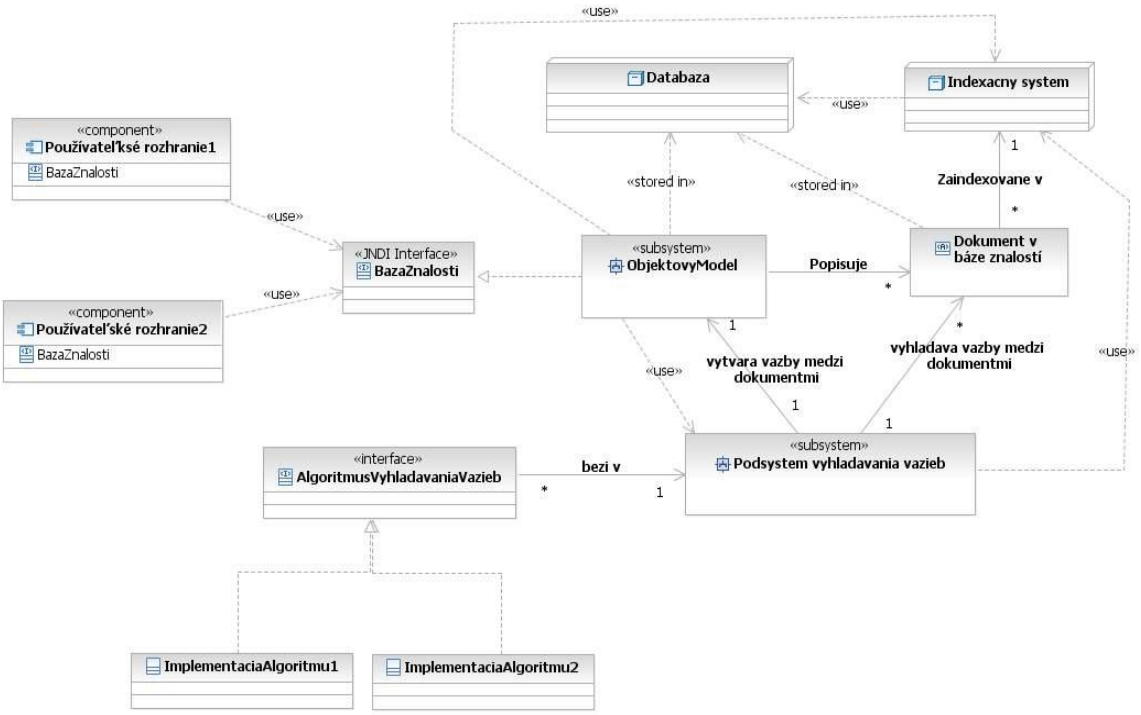
Použitie externých nástrojov

Na vyhľadávanie väzieb použijeme metódu nazývanú kategorizácia textov (z angl. text categorization), ktorá je založená na ručnom vytvorení množiny kľúčových slov určitej problémovej domény, pričom sú uvedené aj hierarchické vzťahy medzi kľúčovými slovami. Porovnaním zindexovaných dokumentov Lucene-om a tejto množiny kľúčových slov danej domény vieme vytvoriť pre každý dokument vektor relevantných kľúčových slov.

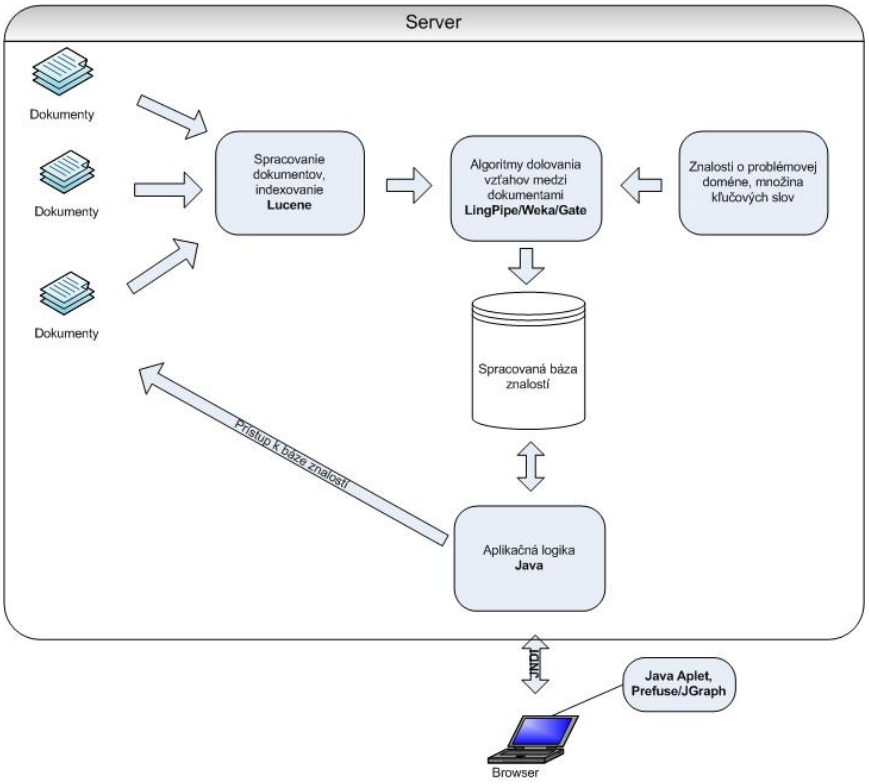
Ďalšou metódou hľadania kľúčových slov bude pravidlo, že za kľúčové slovo sa považuje slovo, ktoré sa v rámci 1 alebo niekoľko málo dokumentov nachádza často, ale celkovo v báze znalostí sa inak nevyskytuje.

Následne väzby medzi dokumentmi budeme hľadať porovnaním vektorov kľúčových slov, a keď podobnosť 2 vektorov je väčšia ako zadefinovaná hranica je medzi 2 uvažovanými dokumentmi priama väzba.

3.5 Dekompozícia systému



Obr. 9. – Dekompozícia systému 1



Obr. 10. – Dekompozícia systému 2

Opis vzťahov

Objektový **model** zachytáva doménu dokumentov a väzieb medzi nimi objektovým spôsobom. Dokumenty sú reprezentované triedou dokument a väzby triedou väzba. Objektový model tak opisuje štruktúru dát uloženú v báze znalostí ako aj obsahuje logiku uloženia, vyhľadávania a spravovania dát v báze znalostí. Tvorí jadro systému, na ktoré sú nabalené ostatné časti systému.

Objektový model je prostredníctvom objektovo-relačného frameworku Hibernate uložený v **databáze**. Objektový model popisuje **dokumenty**, presnejšie povedané súbory, ktoré boli pridané do bázy znalostí. Tieto súbory budú tiež uložené v databáze, aj keď je možné uvažovať nad iným spôsobom ich uloženia (napr. súborový systém, alebo ECM riešenie). **Dokumenty** sú zindexované v **indexačnom systéme** (napr. Lucene). Indexovanie slúži najmä na vyhľadávanie medzi dokumentmi, ale pomáha aj algoritmom na vyhľadávanie väzieb. **Indexačný systém** tiež využíva databázu na ukladanie svojich indexov.

Podsystém vyhľadávania väzieb medzi dokumentmi využíva koncept kontajnera. Tento podsystém je kontajnerom pre **algoritmy**, ktoré väzby medzi dokumentmi vyhľadávajú a vyhľadané väzby ukladajú v **objektovom modeli**. Algoritmy cez podsystém vyhľadávania väzieb pristupujú k objektovému modelu, dokumentom aj indexačnému systému. Algoritmy sú spúšťané ako samostatné vlákna, čo im umožňuje nepretržite asynchrónne pracovať na vyhľadávaní väzieb. Algoritmy spúšťané v tomto podsystéme je potrebné konfigurovať a spravovať. Predmetom konfigurácie sú nielen konfiguračné parametre algoritmov, ale aj množina samotných algoritmov (pridávanie nových implementácii a nových algoritmov). Pre jednoduchosť implementácie postačí v prvej verzii aj konfigurovanie cez xml konfiguračný súbor. Pridávanie implementácií algoritmov je dôležitým bodom rozšírenia systému a pridanie nového algoritmu by nemalo vyústiť do zmien v iných častiach systému.

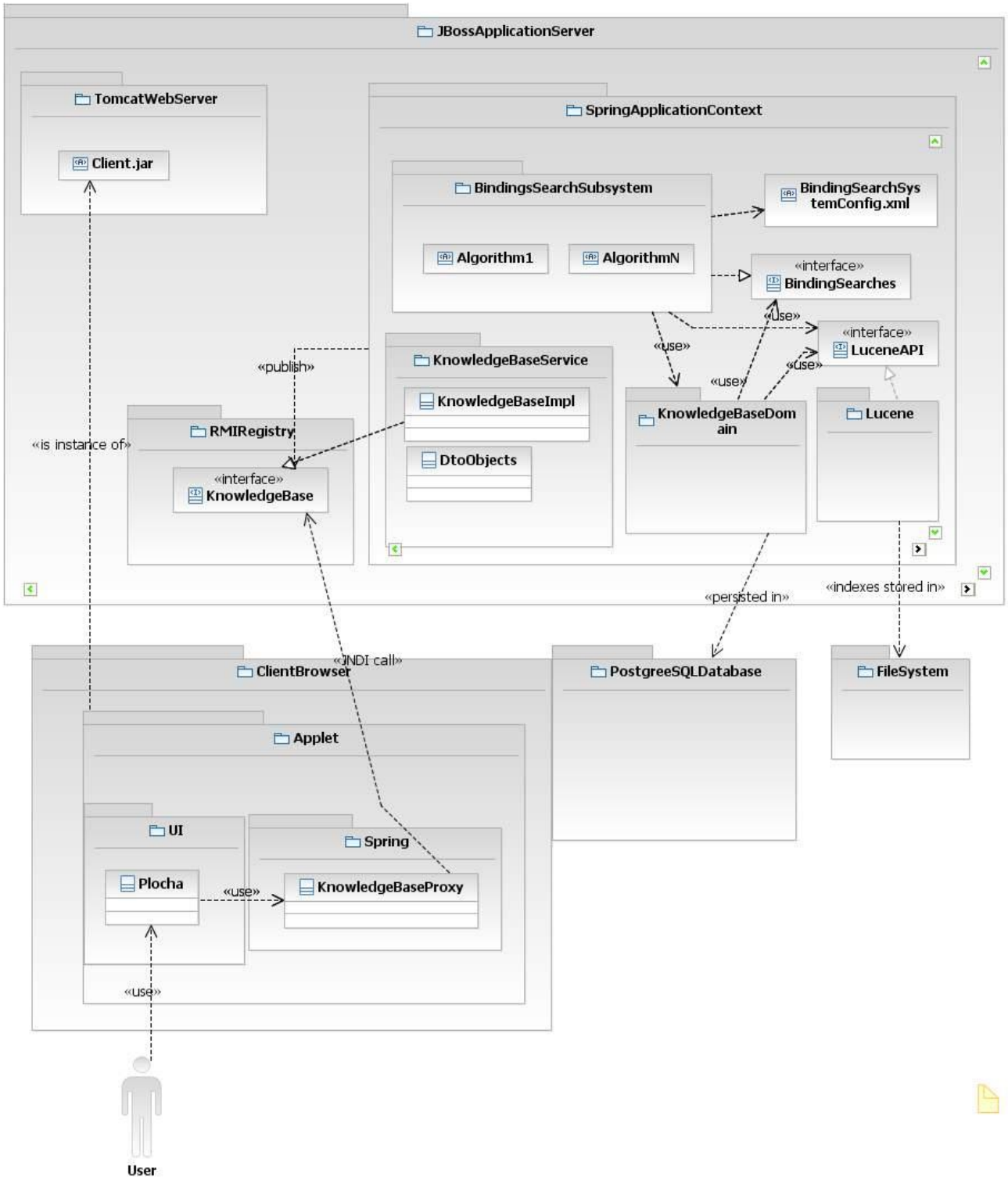
Objektový model potrebuje v niektorých situáciách využiť služby **podsystému na vyhľadávanie väzieb**. Napríklad pri pridaní nového dokumentu do bázy znalostí je potrebné okamžite vyhľadať väzby s už uloženými dokumentmi, aby tieto mohol používateľ hneď zvalidovať.

Objektový model vystavuje rozhranie **BazaZnalosti**. Toto rozhranie musí byť veľmi dobre nadefinované a zdokumentované, lebo bude prístupovým bodom do systému pre používateľské rozhranie aj iné systémy. Vďaka takémuto oddeleniu bude možné vyvinúť viacero nezávislých používateľských rozhraní a vytvorenie nového používateľského rozhrania by nemalo vyústiť do zmien v iných častiach systému.

Používateľské rozhranie sleduje podnety od používateľa a na ich základe cez rozhranie BazaZnalosti vyhľadáva dokumenty a väzby medzi nimi a prezentuje používateľovi graf dokumentov uložených v báze znalostí. Používateľské rozhranie komunikuje so serverom pomocou vzdialených volaní.

Architektúra systému

Vzhľadom k tomu, že nasledujúci text je už viac orientovaný na implementačné technológie, menia sa názvy zo slovenských na anglické.



Obr. 11 Architektúra systému

Navrhnutá je klient-server architektúra. Ako hlavný kontajner je na server strane použitý JBoss aplikačný server. Na klient strane nami vytvorená implementácia bude fungovať ako applet.

V JBoss aplikačnom serveri bude systém bežať ako Spring. To znamená, že necháme technológiu Spring, aby nakonfigurovala väzby v rámci nášho systému. BindingSearchSubsystem je konfigurovaný cez samostatné xml, kde sú definované jeho algoritmy. Tento xml súbor bude tiež vo formáte frameworku Spring. Po jeho zavedení do aplikačného kontextu sa vytvoria démon vlákna, reprezentujúce jednotlivé algoritmy vyhľadávania väzieb. Pri zavádzaní nového algoritmu je potrebné naprogramovať triedu tohto algoritmu, nakonfigurovať ju v xml, prekompilovať celý systém a opätovne deploynúť aplikáciu. Výhodou oddelenia je, že pri pridávaní nového algoritmu nebudú zasiahnuté iné časti systému. Vyhľadávacím algoritmom je k dispozícii Lucene API, ako aj KnowledgeBaseDomain API.

Lucene používa vyhradený adresár v súborovom systéme pre ukladanie svojich indexov.

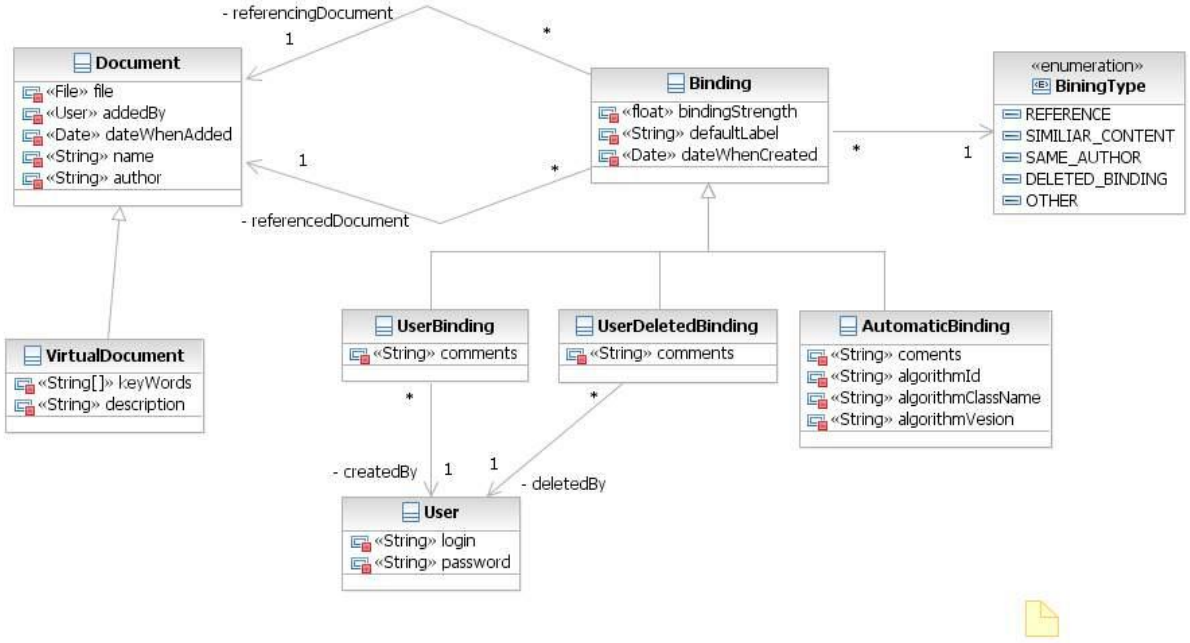
KnowledgeBaseDomain je vrstva doménových objektov a business logiky okolo nich. Perzistencia je vyriešená pomocou odskúšaného frameworku Hibernate v spojení s PostgreeSQL relačnou databázou.

KnowledgeBaseService je vrstva, ktorá vystavuje rozhranie KnowledgeBase. Toto je pre klientov vystavené v RMI registri aplikačného serveru JBoss, aby bolo možné klientom na ňom volať vzdialené volania. Toto rozhranie používa koncept data transfer object (DTO). Všetky objekty sú preklápané do týchto objektov určených na prenos dát. Rozhranie je tým pádom lepšie definované a lepšie oddelené od implementácie KnowledgeBaseDomain (prenášajú sa len POJO objekty).

Keď chce klient používať systém zadá do prehliadača príslušnú URL. Prehliadač stiahne súbor Client.jar a spustí vo svojom JRE applet. Tento applet je používateľským rozhraním a so serverom komunikuje pomocou technológie RMI. V klient vrstve je tiež použitý framework Spring pre pohodlné nainicializovanie všetkých vzťahov a súčastí.

3.6 Návrh systému

Dátový model KnowledgeBaseDomain



Obr. 12 Dátový model bázy znalostí

Dátový model bázy znalostí je relatívne jednoduchý

Document – reprezentuje dokument v báze znalostí. Eviduje podrobnosti kto a kedy ho vložil, ako aj niektoré metadáta, ktoré môžu pomôcť pri vyhľadávaní väzieb (autor, atď.). File je súbor, ktorý je týmto dokumentom reprezentovaný.

Povinné atribúty: name

VirtualDocument – ide o taký dokument, ktorý neobsahuje samotné znalosti, ale slúži na združenie dokumentov obsahujúcich znalosti o určitej doméne. Kľúčové slová slúžia na to, aby aj takýto dokument bol zaindexovaný a dal sa vyhľadať pri fulltextovom vyhľadávaní. Taktiež slúžia kľúčové slová a popis ako informácia pre jeho čitateľov.

Binding- reprezentuje väzbu medzi dokumentmi. Sila väzby je číslo medzi 0-1, ktoré určuje, aká je silná previazanosť dokumentov. Čím je číslo väčšie, tým je väzba silnejšia. Typ väzby určuje, o aký druh previazanosti sa jedná.

Povinné atribúty: bindingStrength, bindingType, referencingDocument, referencedDocument

BindingType – určuje aký reálny význam má väzba. Reálnym významom myslíme spôsob, ako sú sémanticky dokumenty spojené.

- REFERENCE – jeden dokument sa priamo odkazuje druhý referenciou, alebo referencovaný dokument detailnejšie popisuje niektoré koncepty, aspekty, fakty z odkazujúceho sa dokumentu.
- SIMILIAR_CONTENT – dokumenty majú podobný obsah, venujú sa podobnej téme.
- SAME_AUTHOR – dokumenty majú toho istého autora.
- DELETED_BINDING – väzba, ktorá bola používateľom manuálne zmazaná. Väzba takéhoto typu je tu len pre algoritmy vyhľadávajúce väzby, aby medzi týmito dokumentmi ďalšie väzby nevytvárali.
- OTHER – väzba iného nešpecifikovaného významu.

Väzby sú ďalej dedičnosťou rozčlenené na nasledovné triedy (BindingClass):

UserBinding – väzba vytvorená niektorým používateľom. Eviduje sa okrem iného aj používateľ, ktorý väzbu vytvoril a jeho komentár k väzbe. Nesmie byť typ väzby DELETED_BINDING

UserDeletedBinding – väzba, ktorá bola používateľom zmazaná. Ide o informáciu pre algoritmy vyhľadávajúce väzby, že medzi týmito dokumentmi väzby tvoriť nemajú. Eviduje sa okrem iného aj používateľ, ktorý väzbu vytvoril a jeho komentár k väzbe. Väzba musí mať typ väzby DELETED_BINDING.

AutomaticBinding – všetky väzby, ktoré boli vytvorené automaticky algoritmom hľadajúcim väzby medzi dokumentmi. Eviduje sa ktorý algoritmus väzbu vytvoril (id, kvalifikované meno triedy, verzia). Atribút comments eviduje poznámky algoritmu, z ktorých má byť jasné na akom základ väzbu vytvoril. Typ väzby nesmie byť DELETED_BINDING.

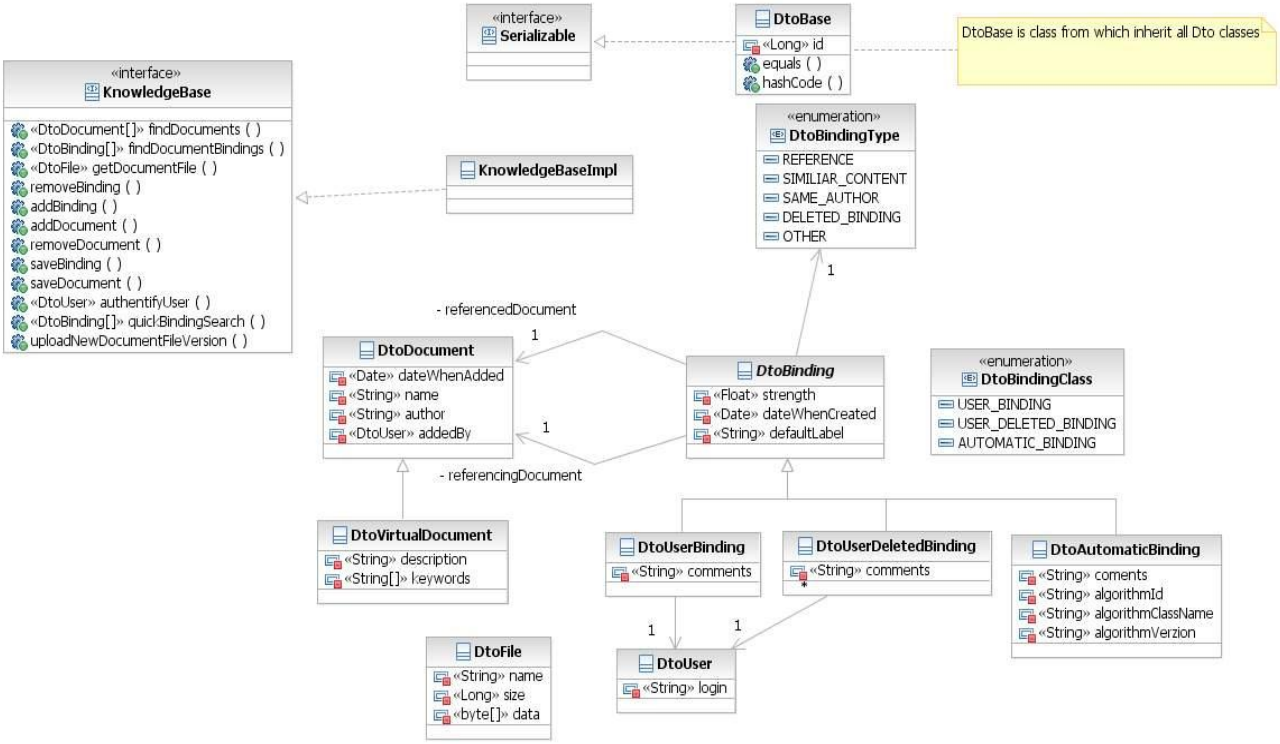
Povinné atribúty: algorithmId, algorithmClassName, algorithmVerzion

User – reprezentuje používateľa systému.

Povinné atribúty: login, password

Rozhranie KnowledgeBase

KnowledgeBase je diaľkovo prístupné rozhranie, cez ktoré vzdialené používateľské rozhranie komunikuje so serverom, kde sa nachádza model a dáta bázy znalostí. Rozhranie používa koncept objektov na prenos dát (data transfer objects), aby bolo toto rozhranie možné volať čisto vzdialenými volaniami. DTO objekty spravidla kopírujú KnowledgeBaseDomian, aj keď sú medzi nimi drobné rozdiely.



Obr. 13 – rozhranie KnowledgeBase

Metódy rozhrania KnowledgeBase

DtoDocument[] findDocuments (String phrase, int offset, int limit, DtoUser user) – vráti zoznam dokumentov, ktoré najviac zodpovedajú vyhľadávanej fráze. Offset určuje, koľko prvých dokumentov sa preskočí (nebudú vo výsledkoch volania). Limit limituje počet vrátených dokumentov. Offset a limit slúžia na stránkovanie. Ak vyhľadávacej fráze nezodpovedá dostatočné množstvo dokumentov, vráti sa prázdny, alebo nekompletný zoznam. Vrátené dokumenty nemajú atribút súbor (potenciálne veľa dát), preto si v prípade potreby (otvorenie súboru) je tento potrebné vyžiadať cez metódu **getDocumentFile().** User je používateľ prihlásený na klientovi.

DtoBinding[] findDocumentBindings(DtoDocument document, float minimalStrength, List<DtoBindingClass> excludedBindingClasses, List<DtoBindingType> excludedBindingTypes, User user) – vráti zoznam väzieb priamo vychádzajúcich z nejakého dokumentu. Vráti len tie väzby, ktorých sila je väčšia ako minimalStrength. Pomocou parametrov excludedBindingClasses a excludedBindingTypes je možné vylúčiť z vráteného zoznamu niektoré triedy väzieb, alebo typy väzieb (napríklad UserDeletedBinding). User je používateľ prihlásený na klientovi. Vrátená väzba má nainicializované dokumenty, ale nemá nainicializované ich súbory.

DtoFile getDocumentFile(DtoDocument document, DtoUser user) – vráti súbor, ktorý je reprezentovaný dokumentom. Vrátený objekt DtoFile obsahuje dáta súboru a je tak možné jeho zobrazenie, alebo uloženie. User je používateľ prihlásený na klientovi.

void removeBinding(DtoBinding binding, boolean isUserDeletion, DtoUser user) – predložená väzba je odstránená zo systému. User je používateľ, ktorý maže väzbu. IsUserDeletion určuje, či sa má toto vymazanie trvalo zapamätať vytvorením väzby triedy UserDeletedBinding.

void addBinding(DtoBinding binding, DtoUser user) – vytvorí v systéme novú väzbu v mene daného používateľa. Binding je objekt, ktorý nebol pred tým perzistovaný a vznikol na strane klienta. Väzba musí mať vyplnené povinné atribúty, inak je vyvolaná BusinessException.

void addDocument(DtoDocument document, DtoFile file, DtoUser user) – vytvorí v systéme nový dokument v mene daného používateľa. Dokument nesmel byť pred tým perzistovaný a vznikol na strane klienta. Dokument musí mať korektne vyplnené všetky povinné atribúty. Systém uloží dokument a zindexuje súbor, ktorý je týmto dokumentom reprezentovaný.

void removeDocument(DtoDocument document, DtoUser user) – odstráni zo systému v mene daného používateľa daný dokument. Súbor dokumentu bude okamžite vyradený z indexov. Takisto budú vymazané všetky väzby s ním súvisiace. Vymazaný bude aj súbor, ktorý tento dokument reprezentuje.

void saveDocument(DtoDocument document, DtoUser user) – systém uloží zmeny atribútov predloženého dokumentu. Uložia sa všetky jeho atribúty okrem súboru, ktorý je

týmto dokumentom reprezentovaný. Ten je potrebné meniť len pomocou metódy uploadNewDocumentFileVersion().

void saveBinding(DtoBinding binding, DtoUser user) – systém uloží zmeny atribútov predloženej väzby. Neuložia sa však zmeny atribútov odkazovaných dokumentov.

DtoUser authetificateUser(String login, String password) – systém overí existenciu používateľa sa uvedeným prihlasovacím menom a heslom. Ak existuje, vráti DtoUser reprezentujúci tohto používateľa. Ak neexistuje, vráti null.

DtoBinding[] quickNewBindingSearch(DtoDocument document, DtoUser user) – využije dostupné rýchle indexačné algoritmy a rýchle algoritmy na vyhľadávanie väzieb, aby našiel niektoré väzby tohto dokumentu. Nájdené nové väzby potom vráti.

void uploadNewDocumentFileVersion(DtoDocument document, DtoFile file, DtoUser user) – nahradí pôvodnú verziu súboru novou verziou (novým súborom). Ten starý je vymazaný z indexov a nový nanovo zindexovaný. File musí mať vyplnené všetky povinné atribúty. Dokument musí byť dokumentom, ktorý už je uložený v báze znalostí.

Pravidlá používania rozhrania KnowledgeBase

Toto rozhranie je určené pre vzdialené volania. Dôležité je uvedomiť si fakt, že vždy, keď je toto rozhranie volané, sú vytvárané vracané nové objekty Dto. Preto ak klient využíva cache na prechovávanie Dto objektov, nevráti sa mu objekt z tejto cache, ani sa mu tento objekt automaticky neaktualizuje, ale z volania metódy rozhrania sa mu vráti úplne nový objekt. Preto v prípade použitia cache, alebo akéhokoľvek uchovávania referencií je vhodné preklopiť obsah objektu získaného volaním metódy rozhrania do objektu, ktorý sa už nachádza na strane klienta, aby v jeden čas existoval len jeden objekt s rovnakým id.

Rozhranie LuceneIndexSystem

Pre prácu s nástrojom Lucene sa používajú nasledujúce metódy, ktoré pracujú s vlastným typom dokumentu tohto nástroja (ďalej už len LuceneDocument). Podstatou LuceneDocument-u sú tzv. polia, ktoré sú zadefinované ako úložiská dát a metadát dokumentu typu DtoDocument. V poli path sa nachádza cesta k súboru resp. celý názov súboru, pole id uchováva id DtoDocument-u, v poli author celé meno autora dokumentu, v title sa nachádza názov dokumentu, v poli keywords kľúčové slová oddelené čiarkami, pole

references je vyhradené pre odkazy na iné dokumenty a pole contents obsahuje zindexovaný obsah dokumentu.

Metódy rozhrania LuceneIndexSystem

long indexFile(DtoDocument document) – odparsuje obsah a metadáta DtoDocument-u, aktualizuje atribúty tohto dokumentu a zaindexuje ho ako LuceneDocument. Metóda následne vráti id zaindexovaného dokumentu, ktoré je zhodné s id DtoDocument-u.

LuceneDocument getDoc(long docID) – metóda na základe zadaného id nájde a vráti zaindexovaný LuceneDocument. Tento dokument pozostáva z viacerých polí.

List<LuceneDocument> getAllDocs() – vráti list všetkých dokumentov zaindexovaných v LuceneIndexSystem.

int getNumberOfDocs() – táto metóda sa používa na zistenie počtu zaindexovaných dokumentov.

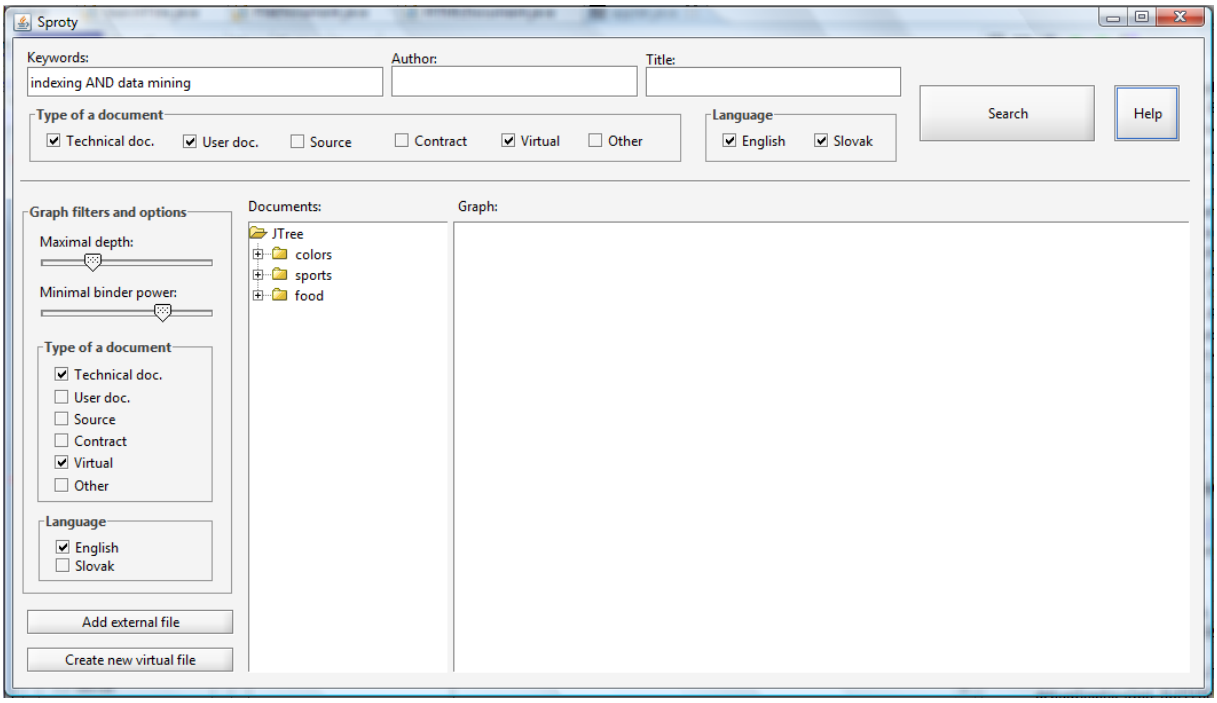
void removeDoc(long docID) – odstráni z indexu dokument, ktorého id sa zhoduje s id zadaným ako parameter.

void deleteAllDocs() – odstráni z indexu všetky dokumenty.

void printAllDocs() – vypíše na stdout základné informácie o všetkých dokumentoch uložených v indexe (teda obsah polí id, path, title a author).

List<LuceneDocument> searchDoc(String[] queryLines, int maxDocs) – metóda zameraná na vyhľadávanie dokumentov. Na vstupe je pole Stringov, pričom nultá položka obsahuje používateľom zadané kľučové slová, druhá autora a posledná tretia názov vyhľadaného dokumentu. Druhým parametrom je maximálny počet dokumentov, ktorý sa má vrátiť. Vyhľadávanie je prevádzané ako operácia AND nad týmito dopytmi. Metóda vyhľadané dokumenty vracia ako list Lucene dokumentov, pričom sú zoradené podľa ich relevantnosti.

3.7 Návrh používateľského rozhrania



Obr. 14 – predbežný návrh používateľského rozhrania

Používateľské rozhranie budeme implementovať ako Java applet - program, ktorý je možné vložiť do html stránky podobne ako napríklad obrázok. Takto bude náš systém ako tenký klient prístupný prostredníctvom ľubovoľného webového prehliadača. Bude sa skladať z dvoch základných častí:

1. Ovládací panel - formuláre pre vstupné dáta a filtre
2. Vizualizácia špecifikovanej časti bázy znalostí

V ovládacom paneli bude používateľ zadávať kľúčové slová, autorov alebo názvy dokumentov, o ktoré sa zaujíma. Ďalej bude môcť pomocou filtrov zúžiť finálny počet a vlastnosti vizualizovaných dát.

Vizualizácia bude tvorená grafom, ktorého uzly predstavujú dokumenty z bázy znalostí a hrany slúžia na vyjadrenie väzieb medzi jednotlivými dokumentmi.

Uzly budú obsahovať názov dokumentu a bude možné dvojklikom otvoriť daný dokument na používateľovej strane. Podľa typu dokumentu (dokumentácia, zdrojový kód, zmluva a pod.) budú uzly farebne odlíšené.

Hrany predstavujúce väzby musia byť vizuálne rozlíšiteľné pre rýchle a jasné pochopenie vizualizovaných dát. Rozlišovať budeme rôznou farbou, hrúbkou, vzorkou čiary a textovým označením hrán.

Formát textového označenia hrany:
[význam väzby]:[sila väzby]
Ak je medzi dvoma dokumentmi viacero väzieb, v označení oddelíme popisy čiarkami:
KW:4,Auth:2, Ref:1

Textové označenie hrán a ich farebné rozlišovanie:

- KW – keywords - modrá
- Ref – referencie - červená
- Auth – autori - zelená
- User – používateľské (pridané) väzby - čierna
- Zmazaná väzba – šedá

Dokumenty (uzly) a väzby medzi nimi (hrany grafu) bude používateľ môcť zo zobrazenia odstraňovať. Táto akcia bude mať dva módy:

1. Odstránenie dokumentu z aktuálneho zobrazenia – slabé vymazanie
2. Odstránenie dokumentu z bázy znalostí – silné vymazanie

Dokumenty nebudú po odstránení reálne vymazané z databázy, iba sa v databáze označia ako nevalidné.

4. Prototyp

Táto kapitola sa zaoberá opisom prototypovania v zimnom semestri. Uvádzame v nej ciele prototypovania a opis vytvoreného prototypu.

4.1 Ciele prototypovania

V našom projekte sme použili evolučné prototypovanie, ktorého hlavnou charakteristikou je, že vytvorený prototyp sa bude postupne vylaďovať a upravovať jeho funkcionalita až do stavu, keď sa z neho stane finálna verzia systému, ktorá bude spĺňať špecifikáciu a bude mať požadovanú funkcionalitu.

Cieľom vytvárania prototypu bolo ukázať pracujúci systém zákazníkovi (pedagógovi). Na základe požiadaviek sa vytvorila prvá verzia prototypu, ktorá bola následne prezentovaná zákazníkovi. Ten mohol pripomienkovať súčasný stav prototypu a vyjadriť sa k jeho funkcionalite. Týmto spôsobom sa ľahšie odhaľovali nedorozumenia, alebo zlé pochopené požiadavky. Zákazník mal väčšiu kontrolu nad tým, aby bol systém vytvorený podľa jeho požiadaviek. Na druhej strane sa taktiež vývojári mohli pri vyvíjaní prototypu stretnúť s problémami a tieto mohli pri prezentácii prototypu konzultovať so zákazníkom. Predišlo sa tak implementácii zle pochopených požiadaviek, či nepotrebných častí systému.

Ďalším, nemenej dôležitým cieľom prototypovania bolo oboznámenie sa s technológiami používanými v systéme. Vývojári získali potrebné vedomosti, skúsenosti a prehľad o poskytovaných funkcionalitách jednotlivých technológií v systéme. Tieto vedomosti budú môcť neskôr zúročiť pri ďalšej implementácii finálneho systému.

4.2 Prototyp

Pri vytváraní prototypu sme sa zamerali na implementáciu požiadaviek, ktoré boli dobre pochopené. Dbali sme pritom na to, aby sme do konca semestra vytvorili prototyp, ktorý bude predstavovať základnú kostru systému a bude poskytovať základnú funkcionalitu. Za základnú funkcionalitu nášho systému považujeme grafické zobrazovanie dokumentov a prepojení medzi nimi. Na to, aby sme mohli dokumenty a prepojenia zobrazovať, musíme byť schopný dokumenty do bázy znalostí vložiť. Systém musí dané dokumenty zindexovať

a vyhľadať väzby medzi nimi. Základná funkcionalita systému by sa dala zhrnúť do nasledovných bodov:

- Vloženie dokumentu do bázy znalostí (systému)
- Zindexovanie vložených dokumentov
- Vyhľadanie väzieb medzi vloženými dokumentmi
- Zobrazenie dokumentov a prepojení medzi nimi vo forme prehľadných grafov

Okrem tejto základnej funkcionality má systém implementované aj ďalšie funkcie, ako napr. mazanie jednotlivých dokumentov a väzieb z bázy znalostí, či zobrazovanie detailných informácií o dokumentoch a väzbách.

Náš prototyp systému sa skladá z dvoch hlavných častí – server a klient. Server má za úlohu spravovať bázu znalostí, indexovať dokumenty, vytvárať medzi nimi väzby, atď. Úlohou klienta je zobrazovať jednotlivé dokumenty a väzby v báze znalostí a umožniť používateľovi ich vyhľadávanie a následnú manipuláciu s nimi. Pri vývoji prototypu sme implementovali dvoch klientov. Jeden je založený na vizualizačnej knižnici JGraph a druhý na knižnici Prefuse. Klienti sú v prototype implementovaní ako standalone aplikácie pripájajúce sa na server.

4.3 Zhodnotenie výsledkov prototypovania

Prototyp spĺňa všetky náležitosti, ktoré sme chceli v prototype obsiahnuť. Pričom prototypovanie nám umožnilo lepšie pochopenie požiadaviek na výsledný systém.

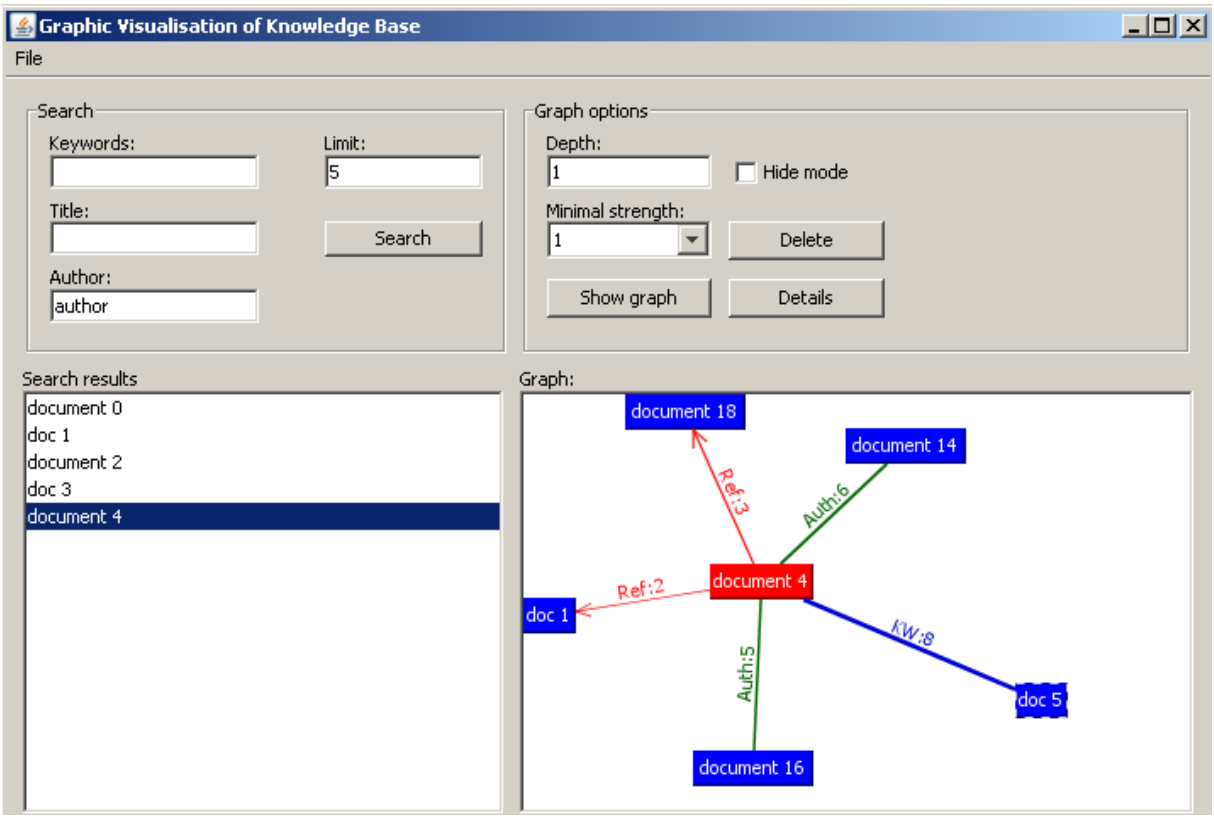
5. Používateľská príručka k prototypu

5.1 Server

Server sa spúšťa spusteným súboru run.bat, ktorý je uložený v dist priečinku modulu servera knowledge-base. Po spustení sa server nainicializuje a počúva na porte 1099. So serverom klient komunikuje prostredníctvom rozhrania, ktoré je definované v tomto dokumente v časti návrhu. Na použitie servera a jeho funkcionality využite klient založený na knižnici Jgraph, ktorý poskytuje používateľské rozhranie, k funkcionalite servera.

5.2 Klient založený na knižnici JGraph

Grafické rozhranie tohto klienta je zobrazené na obrázku 1.



Obr. 15 – rozhranie Jgraph klienta

Klient umožňuje používateľovi vyhľadávať v báze znalostí, zobrazuje mu výsledky vyhľadávania. Po dvojkliku na vybraný dokument zobrazí graf dokumentov so zadanými parametrami. S grafom je možné manipulovať, čo vlastne predstavuje manipuláciu s dokumentmi v báze znalostí. Klient umožňuje pridať dokument do bázy znalostí. Po jeho pridaní server zindexuje tento dokument a vyhľadá možné väzby medzi daným dokumentom a inými dokumentmi v báze znalostí. Následne je možné tento dokument vyhľadať a zobraziť v grafe. Práca s týmto klientom je detailnejšie opísaná v nasledujúcej kapitole.

Používateľská príručka ku klientovi založenému na knižnici JGraph

Po spustení aplikácie sa otvorí okno s používateľským rozhraním klienta. V časti *Search* je možné zadať vyhľadávacie kritériá. Vyhľadávať je možné na základe kľúčových slov, autorov a názvu dokumentu. Je možné aj zadať maximálny počet výsledkov hľadania v poli *Limit*. Po stlačení tlačidla *Search* sa v časti *Search results* zobrazia výsledky hľadania, teda jednotlivé dokumenty.

Následne je možné zobraziť graf dvojklikom na vybraný dokument. Vybraný dokument bude v grafe predstavovať centrálny dokument, z ktorého sa bude v grafe vychádzať. Graf sa zobrazí v časti *Graph*, pričom centrálny dokument je označený červenou farbou. V grafe sú dokumenty reprezentované vrcholmi a väzby medzi dokumentmi sú reprezentované hranami. Rôzne typy väzieb majú rôzne farby hrán. Na každej hrane je zobrazený popis, ktorý označuje, akú väzbu daná hrana predstavuje (KW – spoločné kľúčové slová, Auth – spoločný autor, Ref - referencia). Za typom väzby je číslo naznačujúce silu väzby, čo môže byť číslo od 1 (najslabšia väzba) po 9 (najsilnejšia väzba). S grafom je možné manipulovať – presúvať jednotlivé dokumenty, mazať ich, alebo mazať väzby. Po dvojkliku na vybraný dokument v grafe sa vytvorí nový graf, ktorý bude mať ako centrálny dokument vybraný dokument.

V časti *Graph options* je možné nastavovať jednotlivé parametre grafu:

- Depth – v tomto parametri je možné zadať maximálnu hĺbku, do ktorej sa graf zobrazí. Maximálna hĺbka je číslo naznačujúce v akej maximálnej vzdialenosti od centrálneho dokumentu sa môže zobraziť dokument s ním súvisiaci.
- Minimal strength – číslo od 1 po 9, naznačuje minimálnu silu väzieb, ktoré sa majú v grafe zobraziť

Tlačidlom *Delete* je možné mazať jednotlivé dokumenty, alebo väzby z bázy znalostí. Prvok, ktorý chceme zmazať je nutné označiť v grafe kliknutím. V súčasnosti je možné využívať dva typy mazania – iba grafické mazanie prvkov z grafu a mazanie dokumentov a väzieb z bázy znalostí. To, ktorý typ mazania sa použije, indikuje položka *Hide mode*. Ak je zaškrtnutá, stlačením tlačidla *Delete* sa vykoná iba grafické zmazanie prvku z grafu, pričom táto zmena sa neprejaví v báze znalostí. Ak zaškrtnutá nie je, prvok sa zmaže z grafu a taktiež z bázy znalostí.

Tlačidlom *Show graph* sa zobrazí graf. Je to alternatíva ku dvojkliku na vybraný dokument z časti *Search results*.

Stlačením tlačidla *Details* sa zobrazia detaily vybraného dokumentu alebo väzby.

Do bázy znalostí je možné vkladať dokumenty, a to vybraním možnosti *Add file* z menu *File*. Zobrazí sa formulár na vkladanie dokumentu. Následne je nutné vyplniť autora dokumentu a jeho názov, vybrať súbor, ktorý chceme do bázy znalostí vložiť a vybrať typ dokumentu. Po potvrdení sa daný dokument vloží do bázy znalostí. Systém ho zindexuje a vytvorí väzby.