Architektúra a návrh

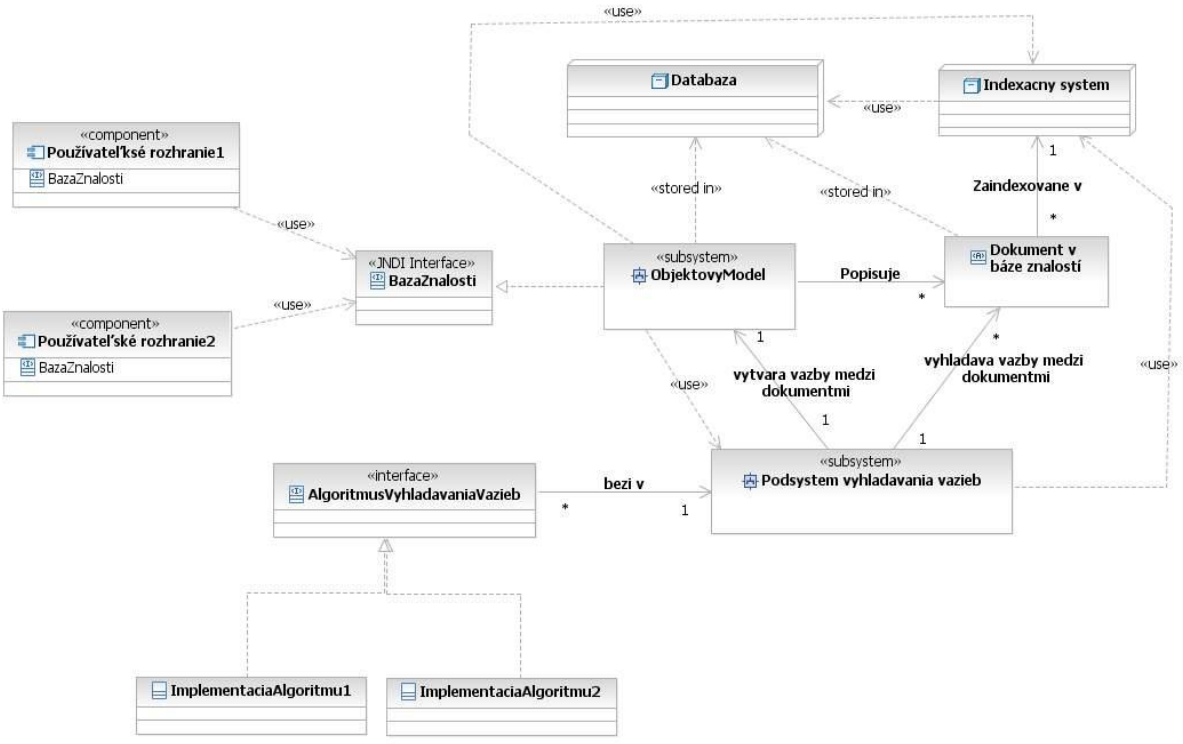
Zdôvodnenie navrhnutej architektúry

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.

Dekompozícia systému



Obr. 1 Dekompozícia systému

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.

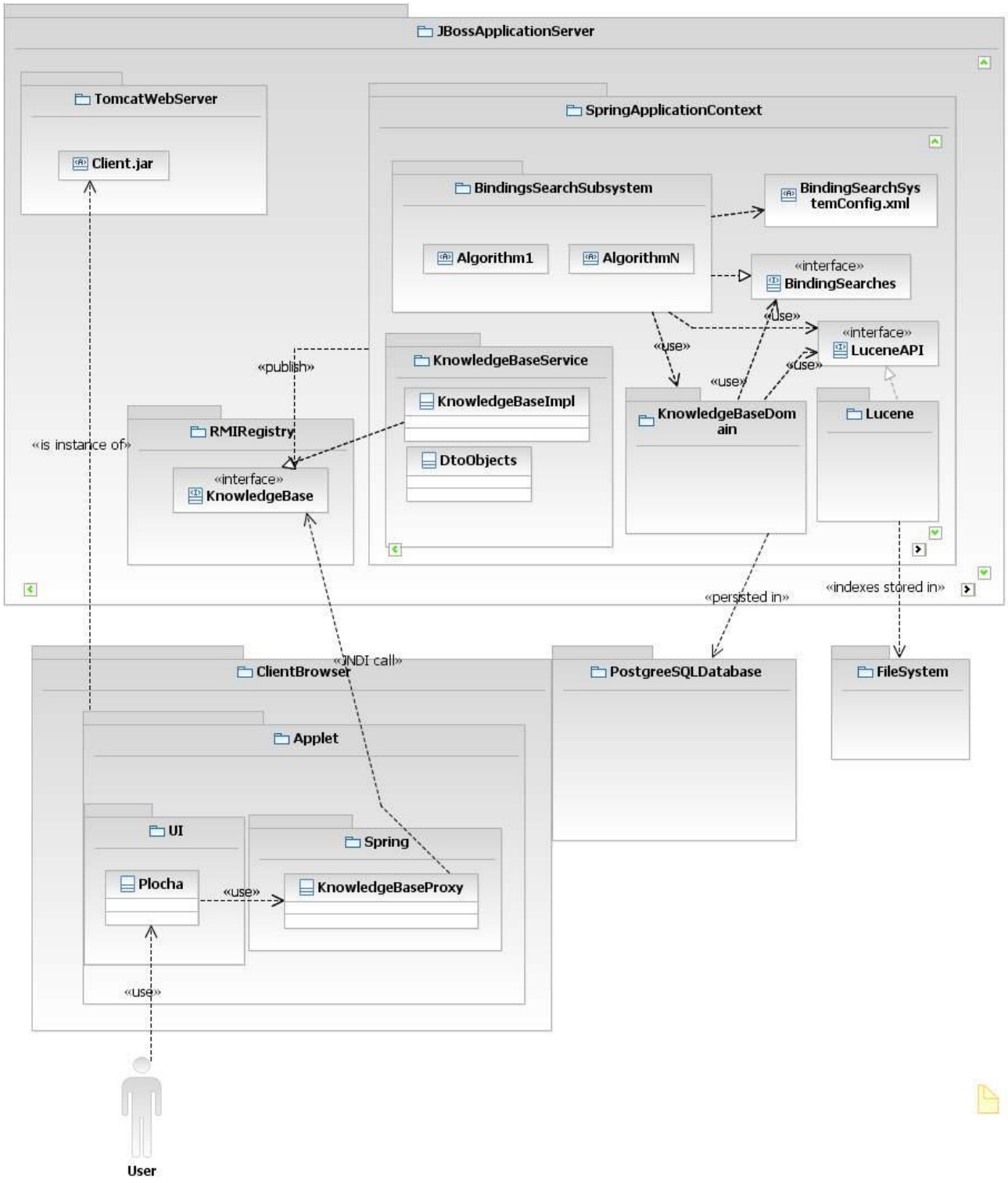
Objektový model potrebuje v niektorých situáciách využiť služby **podsystému na vyhľadávanie väzieb**. Napríklad pri pridaní nového dokumentu do bázy znalostí je potrebné okamžite vyhľadať väzby s už uloženými dokumentmi, aby tieto mohol používateľ hneď zvalidovať.

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. 2 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.

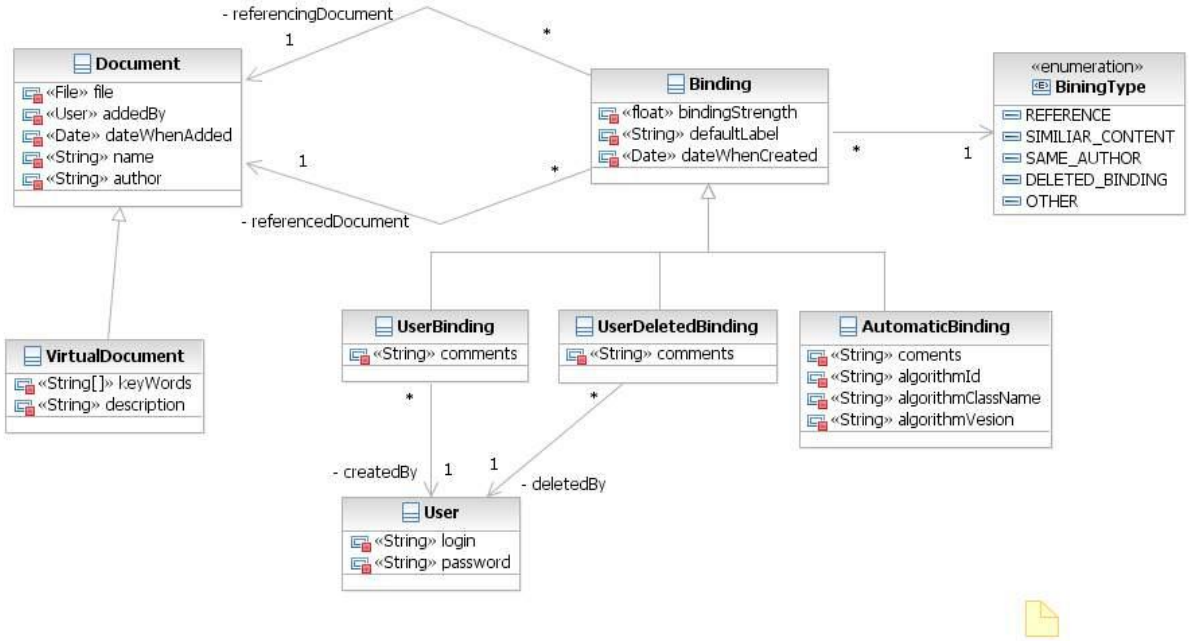
KnowledgeBaseDomain je vrstva doménových objektov a business logiky okolo nich. Perzistencia je vyriešená pomocou odskúšaného frameworku Hibernate v spojení s PostgreeSQL relačnou databázou.

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.

Návrh systému

Dátový model KnowledgeBaseDomain



Obr. 3 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.

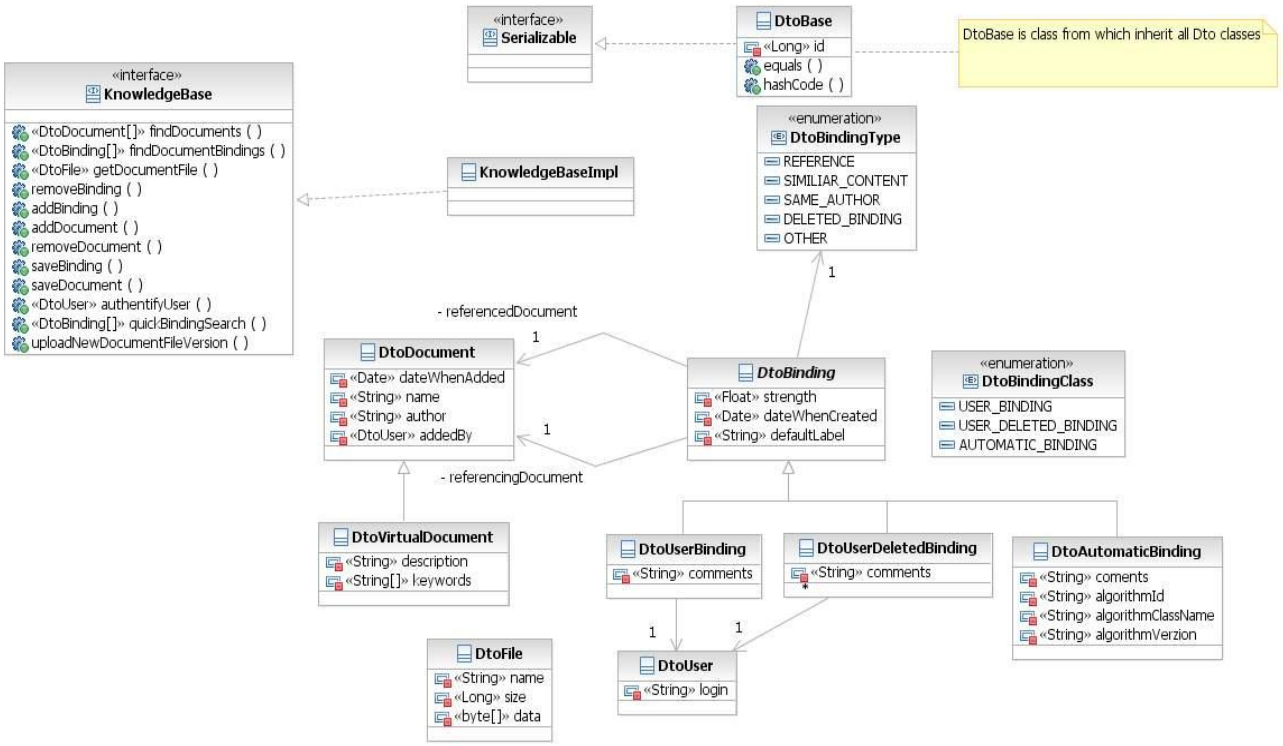
Povinné atribúty: algorithmId, algorithmClassName, algorithmVerzion

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. 4 – 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 zaindexuje 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 zaindexovaný. 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.