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

Tímový projekt

3D grafická podpora vyhľadávania znalostí v dokumentoch

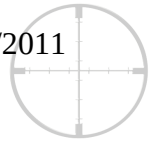
Dokumentácia k softvérovému dielu

Tím č. 9 - d9vina

Bc. Ján Chlpek
Bc. Juraj Jakabovič
Bc. Ivan Janovic
Bc. Matej Kompánek
Bc. Vladimír Polák
Bc. Marek Takáč

Vedúci pedagóg: Ing. Ivan Polášek, PhD.

d9vina@googlegroups.com
Ak. rok : 2010/2011



Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

0.1 Účel a rozsah dokumentu

Dokument predstavuje finálnu verziu dokumentácie k softvérovému dielu vytvoreného na predmete Tímový projekt pre zimný semester. Okrem prehľadu dokumentu v tejto kapitole uvádzame zadanie projektu, jeho ciele a ohraničenia, pričom na konci kapitoly sa nachádza zoznam pojmov problémovej oblasti spolu s používanými skratkami. Ostatné kapitoly predstavujú analýzu a špecifikáciu riešenia, hrubý návrh riešenia a opis prototypu.

0.2 Zadanie

Úlohou tímu bude vytvoriť modul na zobrazenie prepojenia dokumentov podľa informácií v nich v trojrozmernom grafe, ktorý by napomáhal tvorbe nových dokumentov (analytickej a technickej dokumentácie, manuálov, zdrojových kódov a pod.) pomocou zvolených dokumentov.

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.
Projekt môže nadväzovať na prácu tímu č. 12 (Dokumenty) a tímu č.20 (3DVizual) z minulého roku.

0.3 Ciele a ohraničenia

Cieľom projektu je splniť požiadavky zadania projektu a vytvoriť funkčný systém pre 3D zobrazenie bázy dokumentov podľa zadaných požiadaviek používateľa, umožniť používateľovi pohyb v priestore a editáciu grafu vytvoreného väzbami medzi dokumentmi. Navyše bude možné vytvoriť nový dokument z vybraných viacerých dokumentov podľa preferencií používateľa.

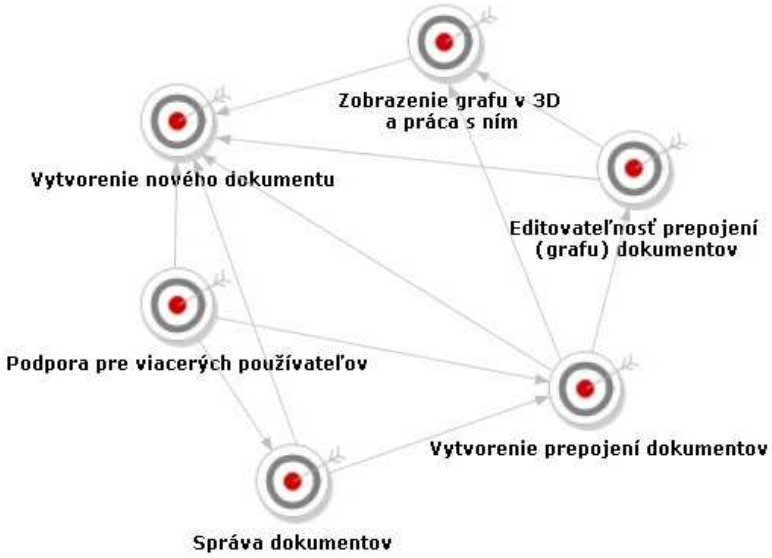
V projekte sa odporúča nadväzovať na práce tímov z minulých rokov. Po analýze týchto projektov sme dospeli k záveru, že nie je možné využiť ich v takej miere, ako sa

pôvodne predpokladalo, takže tým sa zmenili aj ciele nášho projektu, ktorý mal byť akýmsi prepojením týchto predchádzajúcich prác spolu s pridaním novej funkcionality.

Keďže sa nemôžeme opierať o rozšírenie funkčnej implementácie, v novom systéme treba zabezpečiť:

- Správu dokumentov – vkladanie, uchovávanie, automatické indexovanie, editovanie
- Vytvorenie prepojení dokumentov
- Editovateľnosť prepojení (grafu) dokumentov
- Zobrazenie grafu v 3D a práca s ním
- Vytvorenie nového dokumentu
- Podporu pre viacerých používateľov

Na obrázku 1 znázorňujeme diagram cieľov nami vytváraného systému v súvislosti s plánovanou funkcionalitou. Jednotlivé šípky znamenajú závislosti cieľov, resp. nadväznosti dosiahnutia cieľov. To znamená, že aby sme dosiahli cieľ Vytvorenie prepojení dokumentov, je potrebné, aby sme dosiahli cieľ Správa dokumentov.

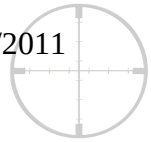


Obr. 1. Diagram cieľov produktu

0.4 Prehľad dokumentu

Úvod – Aktuálna kapitola

Kapitola 1 – Analýza problému
V tejto kapitole uvádzame analýzu tímových projektov, ktoré spracovávali problematiku prekrývajúcu sa s naším zadaním. Okrem toho analyzujeme vizualizačné technológie pre naše potreby.



Kapitola 2 - Špecifikácia riešenia
Táto kapitola obsahuje špecifikovanie kontextu, funkcionality, dát a správania systému.

Kapitola 3 – Návrh riešenia
Táto kapitola predstavuje architektonický návrh nášho riešenia.

Kapitola 4 – Prototyp
V kapitole uvádzame ciele, opis a zhodnotenie prototypu.

Kapitola 5 – Produkt
Kapitola predstavuje dokumentáciu k produktu z letného semestra, vrátane zmien oproti špecifikácii a návrhu.

Kapitola 6 – Používateľská príručka
V kapitole uvádzame používateľskú príručku k novým častiam používateľského rozhrania.

Kapitola 7 – Systémová príručka
Kapitola predstavuje systémovú príručku k dielu, konfiguráciu a inštaláciu komponentov.

0.5 Použité skratky

2D – dvojdimenzionálny

3D – trojdimenzionálny

API – Application Programming Interface

AWT – Abstract Window Toolkit

DAO – Data Access Objects

GUI – Graphical User Interface

HTTP – Hypertext Transfer Protocol

J2EE – Java 2 Enterprise Edition

JOGL – Java OpenGL

MIT – Massachusetts Institute of Technology

OSG – Open Scene Graph

RMI – Remote Method Invocation

SE – Standard Edition

SOAP – Simple Object Access Protocol

SQL – Structured Query Language

STU – Slovenská technická univerzita

UC – Use Case

XML – eXtensible Markup Language

1 Analýza problému

Problémová oblasť je vopred definovaná a rozdelili sme ju na dve hlavné domény, ktoré predstavujú práce dvoch tímov, ktoré pracovali na projektoch:

- *Grafická podpora vyhľadávania znalostí v dokumentoch*
- *Vizualizácia softvérových artefaktov v 3D priestore*

Prvý projekt bol venovaný práci s dokumentmi, spracovaním dokumentov, vytváraním prepojení a grafov z dokumentov, pričom s grafmi a zobrazeniami sa dalo pracovať. Tento projekt v širšom meradle zodpovedá tomu, čomu venuje i náš tím, s tým rozdielom, že dokumenty boli zobrazené iba v 2D rozhraní, čomu bola prispôsobená celá implementácia.
V druhom projekte sa študenti venovali vizualizácii softvéru a jednotlivých jeho artefaktov v 3D priestore pomocou grafových štruktúr, takže aj tento projekt má s naším projektom veľa spoločných prvkov.
Na nasledujúcich riadkoch predstavujeme analýzu oboch projektov a na konci kapitoly sa venujeme analýze vizualizačných technológií využiteľných v našom projekte, ich zhodnoteniu a zhodnoteniu vhodnosti použitia pre náš zámer.

1.1 Analýza projektu Grafická podpora vyhľadávania znalostí v dokumentoch

V rámci predmetu Tímový projekt I a II na našej fakulte v prechádzajúcom akademickom roku vznikol projekt s názvom *Grafická podpora vyhľadávania znalostí v dokumentoch*, ktorý bol vytvorený tímom č.12 (Šproty). Tento tím taktiež pracoval pod vedením Ing. Ivana Poláška, PhD., pričom nám bola poskytnutá možnosť nadviazať na tento projekt.

Našou hlavnou úlohou pri prípadnom nadväzovaní na tento projekt, by bolo hlavne doplniť funkcionalitu o zobrazenie vzťahov medzi dokumentmi v 3D priestore. Okrem toho by bolo vhodné vylepšiť indexáciu dokumentov spolu s vygenerovaním nového dokumentu skladajúceho sa z častí viacerých iných dokumentov, medzi ktorými boli zistené alebo definované vzťahy. Avšak ani po viactýždennej snahe sa nám nepodarilo zdrojové kódy, knižnice a ostatné konfiguračné súbory aplikácie tímu č.12 pretransformovať do takého stavu, aby bolo možné projekt spustiť bez chýb. Z toho dôvodu nebolo možné prakticky overiť správnosť funkcionality tohto projektu a preto sme pri tejto analýze vychádzali hlavne z dodanej dokumentácie a zo zdrojových kódov.

Tento projekt bol implementovaný v programovacom jazyku Java, pričom autori použili J2EE frameworky Spring a Hibernate. Okrem toho bola použitá databáza PostgreSQL a viacero Java knižníc, využitých pri vykresľovaní grafov a indexácii dokumentov. Autori podľa nášho názoru veľmi správne zvolili aplikácii architektúru klient-server. Toto rozhodnutie zdôvodňujú tým, že je potrebné oddeliť zobrazovaciu logiku od ostatných častí celého systému, tak aby bolo možné vyvíjať a aj nasadiť viacero nezávislých používateľských rozhraní. Na strane serveru bol použitý aplikačný server JBoss a strana klienta funguje ako applet. Tieto dva oddelené celky spolu komunikujú pomocou Java RMI (Remote Method Invocation).

Autori navrhli server tak, aby na jeho pozadí neustále bežali skryté vlákna, ktoré sa venujú analýze dokumentov a vyhľadávaniu väzieb medzi nimi. Taktiež na tejto strane vytvorili „kontajner algoritmov“, ktorý obsahuje viacero odlišných algoritmov, pričom každý z nich hľadá väzby medzi dokumentmi iným spôsobom (napr.: podľa autorov, kľúčových slov, vzájomných referencií atď.) To má výhodu v tom, že pri nasadzovaní nových algoritmov a novších verzií starých algoritmov by nebolo potrebné zasahovať do ostatných častí systému. Ako už bolo spomenuté klient beží ako applet a jeho úlohou je iba graficky znázorniť výsledky v podobe grafu dokumentov a vzťahov medzi nimi. Systém teda poskytuje tenkého klienta, ktorý je prístupný pomocou webového prehliadača. Klient sa skladá z ovládacieho panela, ktorý obsahuje formuláre pre vstupné dáta a časť pre samotnú vizualizáciu špecifických báz znalostí.

Tento projekt je teda zameraný na grafické zobrazenie väzieb medzi dokumentmi nachádzajúcimi sa v systéme. Aplikácia je však obmedzená iba na renderovanie v 2D priestore, na rozdiel od požiadaviek na náš projekt, ktorý má zvládať nielen vykresľovanie v 2D, ale aj v 3D. Túto funkcionalitu autori dosiahli pomocou open-source knižnice JGraph, ktorá je kompletne napísaná v jazyku Java a je vyvíjaná už niekoľko rokov, takže je pravdepodobne na vysokej úrovni. Za ďalšiu kľúčovú súčasť aplikácie, ktorá sa týka funkcionality, možno považovať indexáciu dokumentov a hľadanie vzťahov medzi nimi. Indexácia dokumentov je netriviálna úloha, ktorá je stále predmetom výskumu a preto aj existuje viacero dostupných knižníc, ktoré riešia tento problém. Autori najskôr používali Java knižnicu Lucene, no neskôr prešli na knižnicu Compass. Počas analýzy tohto projektu sme však zistili, že tento tím nemal indexáciu dokumentu dostatočne zvládnutú. To hlavne z toho dôvodu, že index bol vytvorený iba z metadát a nie z obsahu samotného dokumentu. Preto dokumenty bez definovaných metadát nie je možné použiť, čo značne znižuje kvalitu tejto aplikácie. Je síce možné okrem automatického definovania väzieb systémom pomocou skladu

algoritmov aj ručne poprepájať dokumenty, ktoré so sebou podľa používateľa súvisia. To však tento problém nerieši, avšak aspoň zlepšuje relevantnosť vzťahov medzi dokumentmi (používateľ totiž často môže poznať korelácie medzi dokumentmi, ktoré by aplikáciou nemuseli byť rozpoznané). Problémom tohto prístupu je však to, že jednou z kľúčových myšlienok projektu je automatizované vyhľadanie vzťahov medzi dokumentmi nielen z metadát, ale aj z ich obsahu.

Je evidentné, že táto aplikácia môže pre náš projekt pôsobiť ako istý stavebný kameň. To berieme na zreteľ pri definovaní architektúry našej aplikácie, ale aj pri výbere technológií a knižníc na dosiahnutie identifikovanej funkcionality našej aplikácie. V našej práci avšak nebudeme pokračovať v zdrojových kódoch tohto projektu, čo však neznamená, že tento kód nevyužijeme.

1.2 Analýza projektu Vizualizácia softvérových artefaktov v 3D priestore

V rámci predmetu Tímový projekt vznikol minulý rok pod vedením Ing. Petra Kapca zaujímavý projekt s názvom *Vizualizácia softvérových artefaktov v 3D priestore*. Tento projekt bol vytvorený tímom č. 20 (Starwalkers) a jeho cieľom bola vizualizácia softvéru a jednotlivých jeho artefaktov v 3D priestore pomocou grafových štruktúr. Táto téma sa v značnej miere prekrýva so zameraním nášho projektu a tým pádom je tento projekt vhodným kandidátom, z ktorého by sme mohli vychádzať hlavne v oblasti 3D vizualizácie grafov.

Ako bolo už vyššie spomenuté dôvodom, prečo by nám tento projekt mohol uľahčiť prácu, je najmä fakt, že tomuto tímu sa podarilo úspešne zanalyzovať oblasť zobrazovania grafov v 3D priestore a na základe zistených poznatkov implementovali takýto systém. I keď sa tento projekt okrem vizualizácie grafov venoval aj iným oblastiam, práve oblasti efektívneho vykreslenia a rozmiestnenia grafov v 3D priestore bolo venované najväčšie úsilie. Overiť, či je projekt funkčný sa doteraz podarilo len čiastočne, keďže sa ho nepodarilo oživiť úplne. Problémy sa týkajú najmä buildovania projektu a jednotlivých knižníc potrebných na vykresľovanie grafov. Nezáležiac na tom, či sa rozhodneme prevziať zdrojové kódy tohto tímu alebo si zvolíme možnosť implementovať túto časť projektu nanovo, faktom ostáva, že riešenia, ktoré tento tím priniesol sú dobrým základom pre náš ďalší postup.

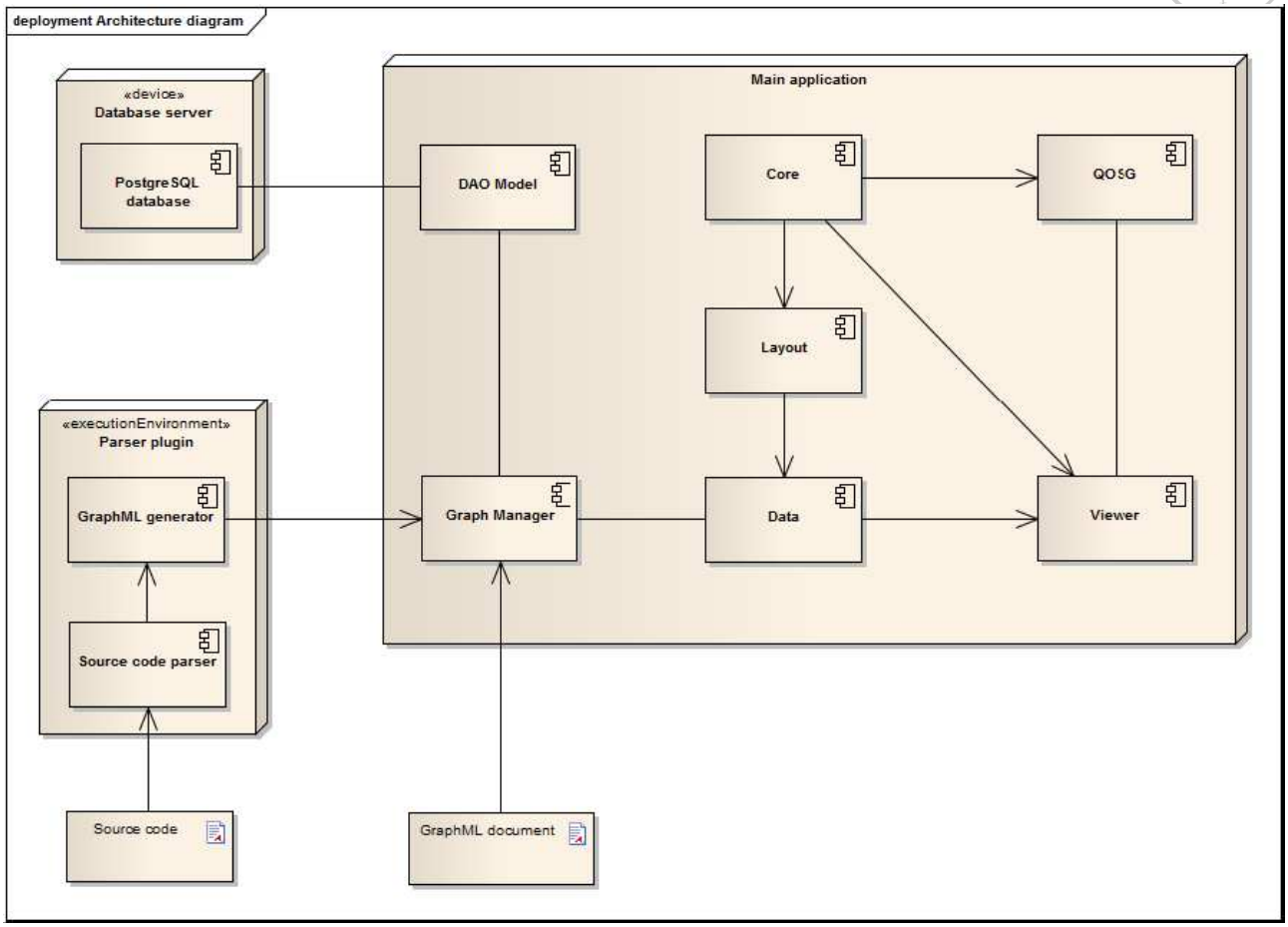
Tento systém je napísaný v jazyku C++, pričom na vytvorenie používateľského rozhrania bol využitý framework Qt. Ďalšou z dôležitých súčastí systému je knižnica Open

Scene Graph (OSG), ktorá sa používa na samotné vykresľovanie grafu, pričom poskytuje širokú funkcionalitu a výkon. Na uchovávanie dát grafu je použitá databáza PostgreSQL. Na komunikáciu medzi hlavnou časťou systému a plugin-om na spracovanie zdrojového kódu bol využitý formát GraphML. Systém sa skladá z dvoch častí a to z hlavnej aplikácie a z parser plugin-u. Keďže tento plugin nie je pre náš projekt relevantný, opíšme fungovanie len hlavnej časti systému.

Hlavná časť systému pozostáva z viacerých modulov, pričom každý má svoje špecifické funkcie. Dátová časť systému pozostáva z troch kľúčových modulov, pričom modul dátovej štruktúry grafy (Data) je najpodstatnejší, keďže je využívaný pri vizualizácii a layoutovaní. V tomto module sú reprezentované grafy prostredníctvom objektov. Zdrojom pre vytvorenie objektového modelu grafu je buď súbor, v ktorom je reprezentovaný graf vo formáte GraphML, alebo databáza PostgreSQL, s ktorou sa komunikuje prostredníctvom modulu mapovania objektov (DAO Model). Databáza teda slúži na uchovávanie jednotlivých grafov medzi jednotlivými behmi systému. Pri spustení systému modul manažér grafov (Manager) získa dostupné grafy z databázy a inicializuje z nich objektové modely.

Ďalším dôležitým modulom je modul rozmiestnenia uzlov v 3D priestore (Layout), ktorý pracuje s algoritmom Fruchterman-Reingold, ktorý upravil študentom FIIT STU Bc. Jakub Ukrop v rámci svojej bakalárskej práce. Tento algoritmus je vykonávaný v samostatnom vlákne, pričom toto vykonávanie prebieha v nekonečnej slučke. Rozhranie však umožňuje tento proces prerušiť a taktiež znova spustiť.

Nemenej dôležitým modulom je modul 3D vizualizácie (Viewer), ktorý zabezpečuje vizualizáciu grafu a taktiež interakciu používateľa s týmto grafom. Na tieto činnosti využíva knižnicu Open Scene Graph a tiež knižnicou Qt, ktorá zabezpečuje 2D prvky používateľského rozhrania. Jednotlivé moduly a ich prepojenia v rámci architektúry sú zobrazené na obrázku č.2.



Obr. 2. Návrh architektúry systému tímu Starwalkers

Znovupoužitie kódov tímu Starwalkers by nám značne uľahčilo prácu pri našom projekte, keďže ich výsledky v rámci vizualizácie boli na veľmi dobrej úrovni. Dosiahli to najmä tým, že ich grafický engine prešiel niekoľkými optimalizáciami, ktoré napomohli jeho výkonu najmä pri zobrazovaní rozsiahlejších grafov. Znovu implementovať tieto algoritmy by mohlo byť dosť náročné a to i časovo. Ďalším pozitívom je, že zdrojom pre vytvorenie nového grafu môže byť GraphML dokument. Toto vytvára priestor pre prepojenie systému pre vyhľadávanie znalostí s 3D vizualizáciou grafu. Negatíva by mohli vyplynúť najmä z preberania zdrojových kódov po iných autoroch a fakt, že doteraz sa nám tento projekt nepodarilo úplne spojazdniť, tento názor len potvrdzuje.

1.3 Analýza vizualizačných technológií

Pri implementácii vizualizácie grafov existuje viacero možností, čo sa týka vizualizačných knižníc a technológii, ako aj nástrojov pre tvorbu používateľského rozhrania. V ďalšej časti sú bližšie popísané a zhodnotené ich prínosy a nevýhody.

1.3.1 Processing

Processing je open source programovací jazyk a vývojárske prostredie na implementáciu problémov vizuálneho dizajnu. Tento projekt bol iniciovaný pred 10 rokmi a doteraz je vedený univerzitou MIT.

Jadro tejto knižnice je implementované v jazyku Java, vďaka čomu sú projekty vytvorené v tomto prostredí multiplatformové. Takisto každý kód napísaný v jazyku Processing je validný Java program.

Knižnica umožňuje vytváranie 2D aj 3D grafiky, pričom pracuje na úrovni vysokej abstrakcie, vďaka čomu je vývoj týchto aplikácii niekoľkonásobne rýchlejší oproti klasickým 3D nástrojom.

Komponenty vytvorené v tejto knižnici je jednoduché preniesť do klasických Java projektov, pretože každý takto vytvorený vizuálny komponent je použiteľný s GUI nástrojmi Java SE ako je Swing alebo AWT. Na realizáciu tohto prepojenia je potrebné využitie základných knižníc Processing. Zároveň teda treba dodať, že táto technológia nie je viazaná na svoje vývojárske prostredie.

Processing poskytuje dva spôsoby vykresľovania grafiky. Prvou z nich je možnosť natívneho Java vykresľovania. Výhodou tejto metódy je fakt, že takto napísaný program bude bežať na každej implementácii virtuálneho stroja Javy bez nutnosti inštalovania dodatočných knižníc. Takisto nevyžaduje prítomnosť grafickej karty. Nevýhodou je značné spomalenie procesu vykresľovania, čo sa týka zložitosti vykresľovanej scény alebo počtu objektov. Druhou možnosťou je vykresľovanie 3D scény prostredníctvom technológie JOGL (Java OpenGL), ktorá prináša značné urýchlenie vykresľovania. Nevýhodou je nutnosť dodatočných knižníc (JOGL + OpenGL). Oba spôsoby poskytujú štandartné schopnosti grafických rozhraní ako je napríklad vyhladzovanie alebo nasvietenie scény.

Hlavnou výhodou tejto technológie je jednoduchosť nasadenia, keďže implementované vizualizácie je možné priamo použiť v klasických Java rozhraniach alebo zobrazovať v internetových prehliadačoch prostredníctvom technológie Java Appletov bez nutnosti inštalácie.

Je nutné dodať, že knižnica nedokáže konkurovať natívnym implementáciám v nižších jazykoch (C++), takže nie je vhodná pre zobrazovanie a prácu s veľmi komplikovanými scénami, ktoré vyžadujú rýchlu odozvu pre pohodlnú prácu s rozhraním.

1.3.2 Java 3D / JoGL

Tieto dve knižnice predstavujú možnosti platformy Java na vykresľovanie 3D scény. Obe pracujú na základe natívnych volaní a využitia hardvérovej podpory vykresľovania, vďaka čomu sú efektívnejšie oproti predchádzajúcej knižnici.

V súvislosti s technológiou Java 3D je potrebné spomenúť bakalársku prácu študenta FIIT Bc. Jakuba Ukropa, ktorá venuje problematike vykresľovania rozsiahlych 3D grafov. Riešenie bolo implementované na platforme Java s využitím technológie Java 3D. Implementácia bola dostatočne efektívna na zobrazovanie grafov s počtom uzlov rádovo v tisícoch.

Výhodou knižnice Java 3D je poskytnutý objektový model, ktorý môže byť výhodou oproti klasickým procedurálnym rozhraniam grafických knižníc.

1.3.3 C++ / OpenGL / Direct3D

Knižnice OpenGL a Direct3D predstavujú rozhranie pre programovanie aplikácii, ktoré vytvárajú 2D a 3D zobrazenie s využitím hardvérovej podpory grafických kariet. Obe rozhrania pracujú v jazyku C / C++, pričom existujú nadstavby pre vyššie programovacie jazyky (ako spomínaný JOGL).

OpenGL predstavuje multiplatformovú špecifikáciu, ktorá zahŕňa aj open-source implementácie, zatiaľ čo Direct3D je súčasťou proprietárnych knižníc DirectX spoločnosti Microsoft, ktoré sú určené pre operačný systém Windows.

Obe tieto knižnice predstavujú najefektívnejší spôsob zobrazovania 3D scény. Napriek tomu zložitosťou implementácie presahujú možnosti tohto projektu.

1.3.4 OSG

Open Scene Graph (OSG) je open-source, multiplatformové programové rozhranie na vývoj grafických aplikácií. Je určené na implementáciu problémov vizuálnej simulácie, virtuálnej reality, modelovania a pod. Pracuje na základe rozhrania OpenGL a ponúka objektový model,

ktorý uľahčuje prácu s 3D zobrazením. Tento nástroj je podporovaný väčšinou operačných systémov.

Výhodou použitia knižnice OSG je už spomínaný objektový model, ktorý pracuje ako nadstavba nad OpenGL. Okrem toho pracuje s dátovou štruktúrou grafu scény, ktorý reprezentuje vizualizované objekty. Vďaka tomu odbremeňuje programátora od nutnosti práce priamo s primitívnymi útvarmi 3D zobrazenia, následkom čoho je uľahčenie a zrýchlenie vývoja. Okrem toho podporuje optimalizáciu 3D zobrazenia na nízkej úrovni, čo sa týka vzdialených objektov, úrovne detailu alebo prekrývania jednotlivých objektov.

Táto knižnica je najbežnejšie používaným nástrojom pri vizualizácii rozsiahlych grafov. V rámci projektov FIIT môžeme spomenúť projekty tímu Starwalkers alebo bakalársku prácu Bc. Petra Kajana.

1.3.5 Qt

Qt je open-source knižnica na tvorbu okien a GUI pracujúca v jazyku C / C++. Je vyvíjaná spoločnosťou Nokia. Je vysoko oceňovaná hlavne odborníkmi na problematiku HCI (human computer interaction), keďže ponúka množstvo prvkov pre tvorbu grafického rozhrania. Okrem toho poskytuje programové rozhranie aj pre tvorbu klasických konzolových programov.

Hlavnou výhodou tejto knižnice sú jej bohaté možnosti pri tvorbe okien, efektívna objektová štruktúra a jej multiplatformovosť (okrem klasických operačných systémov existujú aj implementácie pre mobilné platformy ako Symbian alebo Windows Mobile).

Qt vo svojej distribúcii okrem vývojárskej sady obsahuje aj samostatné prostredie, ktoré umožňuje jednoduchý vývoj okien. Okrem toho poskytuje možnosť integrácie s prostredím Visual Studio 2008.

V oblasti tvorby GUI má táto knižnica minimálnu konkurenciu. Spomenúť môžeme knižnicu MFC, ktorá je však výrazne zastaraná s menšou možnosťou prispôsobenia a vysokou náročnosťou implementácie. Aj vďaka tomu je Qt je ideálnou možnosťou tohto projektu pre tvorbu štandardných GUI prvkov.

2 Špecifikácia riešenia

V tejto kapitole uvádzame špecifikáciu riešenia, zahŕňajúcu kontext nášho systému a jeho súvislosť s problematikou, funkcie systému a prípady použitia, údaje v systéme s dátovým modelom a správanie systému vyjadrené niekoľkými diagramami.

2.1 Kontext systému

Po analýze práce tímov č. 12 (Grafická podpora vyhľadávania znalostí v dokumentoch) a tímu č.20 (Vizualizácia softvérových artefaktov v 3D priestore) sme sa rozhodli pokračovať v ich práci a využiť maximum z toho, čo sa im podarilo dosiahnuť. Aj napriek nášmu rozhodnutiu začať implementáciu od začiatku, algoritmy, ktoré tieto dva tímy implementovali vieme využiť a prípadne migrovať do nami zvolených programovacích jazykov. Rovnako pri návrhu systému a špecifikovaní požiadaviek sme využili analýzy týchto dvoch tímov, čím sme ušetrili čas, ktorý sme mohli venovať oživovaniu ich projektov.

Nami navrhovaný systém bude umožňovať 3D zobrazenie vzťahov medzi dokumentmi. Vzťahy medzi dokumentmi budú určené na základe metadát, ale aj obsahu dokumentov. Používateľ si bude môcť samostatne, nezávisle na vygenerovaných väzbách vytvoriť vlastné väzby. Každá väzba bude mať isté vlastnosti, ktoré budú vizuálne odlišné (farba, hrúbka, a i.). Informácie o dokumentoch sa budú ukladať v báze znalostí na serverovej časti systému. Na strane klienta, bude prebiehať spracovanie informácií aj samotné zobrazovanie. Zobrazovanie a samotné jadro systému navrhneme čo najviac oddelené, pre prípadné zmeny v zobrazení, napríklad v prípade nutnosti optimalizácie rýchlosti zobrazovania.

2.1.1 Nefunkcionálne požiadavky na systém

Od systému sa očakáva, že po určitej dobe a vylepšeniach (napríklad ďalšími tímami) bude vhodný na využívanie v praxi. Mal by byť teda konkurencieschopný a cieľom je, že by sa v budúcnosti mohol stať súčasťou produktu Gratex Knowledge Office a napomáhal tvorbe nových dokumentov (analytickej a technickej dokumentácie, manuálov, zdrojových kódov a pod.) pomocou získaných znalostí. Toto predpokladá fakt, že síce v rámci projektu budeme pracovať rádovo s desiatkami dokumentov, ale v budúcnosti je možné, že systém bude

pracovať s tisíckami dokumentov, ktorých 3D zobrazenia a práca s nimi môže byť veľmi náročná a preto treba zvážiť technológie, ktoré budeme používať aj smerom do budúcnosti.

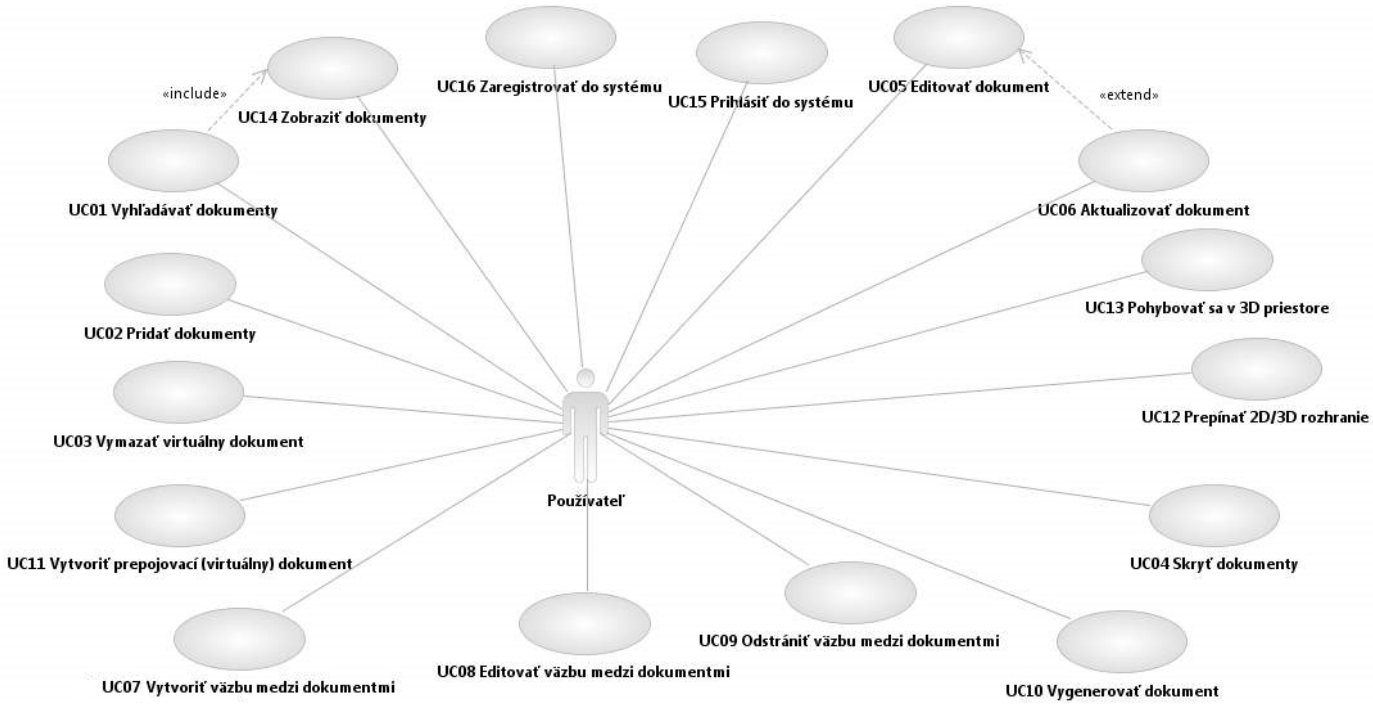
2.2 Špecifikácia funkcií a správania systému

Systém pre naše potreby vytvárame pre jeden typ používateľov, ktorí budú mať rovnaké práva. Je dôležité, aby bola vytvorená podpora pre viacerých používateľov, aby sme mohli riešiť záležitosti s históriou prezerania dokumentov. Týmto rozumieme odporúčanie podobných dokumentov na základe toho, aké dokumenty prezeral nejaký konkrétny používateľ v minulosti.

V tejto kapitole uvedieme prehľad prípadov použitia s popisom a následne pre každý prípad použitia uvedieme tabuľku s opisom a scenárom. Pre zaujímavé prípady použitia uvádzame aj diagramy aktivít. Okrem toho pre kľúčové entity uvádzame stavové diagramy.

2.2.1 Prehľad prípadov použitia

Na obrázku č. 3 sú v diagrame prípadov použitia znázornené identifikované prípady použitia pre systém. Pre každý prípad použitia v jeho podrobnom opise uvádzame aj prioritu, teda dôležitosť pre splnenie tých hlavných vytýčených cieľov pre úspešnosť nášho projektu.



Obr. 3. Diagram prípadov použitia.

Prípady použitia pre navrhovaný systém sú teda nasledovné:

UC01 – Vyhľadávať dokumenty

Systém poskytne používateľovi v grafickom rozhraní vyplniť formuláre a podľa zadaných informácií vyhľadá príslušné dokumenty. V tabuľke č. 1 je prípad použitia UC01 podrobne opísaný. Obrázok č. 4 znázorňuje diagram aktivít pre daný prípad použitia.

UC02 – Pridať dokumenty

Systém poskytne používateľovi možnosť pridať jeden alebo viac dokumentov. V tabuľke č. 2 je prípad použitia UC02 podrobne opísaný.

UC03 – Vymazať virtuálny dokument

Systém umožní používateľovi vymazať existujúci virtuálny dokument zo systému. V tabuľke č. 3 je prípad použitia UC03 podrobne opísaný.

UC04 – Skryť dokumenty

Systém umožní používateľovi v grafe dokumentov skryť jeden alebo viacero uzlov (t.j. dokumentov). V tabuľke č. 4 je prípad použitia UC04 podrobne opísaný.

UC05 – Editovať dokument

Používateľ pomocou systému môže upraviť metadáta dokumentu, ako napríklad názov dokumentu, meno autora, kľúčové slová atď., alebo priamo jeho obsah. V tabuľke č. 5 je prípad použitia UC05 podrobne opísaný. Obrázok č. 6 znázorňuje diagram aktivít pre daný prípad použitia.

UC06 – Aktualizovať dokument

Používateľ môže nahrať do systému novú verziu dokumentu. Systém novú verziu indexuje, starú verziu odstráni a uchová novú verziu namiesto pôvodného dokumentu. V tabuľke č. 6 je prípad použitia UC06 podrobne opísaný.

UC07 – Vytvoriť väzby medzi dokumentmi

Pomocou systému používateľ vytvorí prepojenie medzi dokumentmi, ktoré spolu súvisia, ale v systéme zatiaľ prepojené nie sú. V tabuľke č. 7 je prípad použitia UC07 podrobne opísaný. Obrázok č. 6 znázorňuje diagram aktivít pre daný prípad použitia.

UC08 – Editovať väzbu medzi dokumentmi

Používateľ môže pomocou systému upraviť atribúty niektorej väzby na základe nových zistení, prípadne iných okolností. V tabuľke č. 8 je prípad použitia UC08 podrobne opísaný.

UC09 – Odstrániť väzbu medzi dokumentmi

Ak používateľ zistí, že niektorá väzba nevyjadruje skutočnú súvislosť medzi dokumentmi, pomocou systému ju môže odstrániť. Takáto situácia môže nastať napríklad preto, že systém niektoré väzby medzi dokumentmi vytvoril automaticky, ale tieto neboli vytvorené správne. V tabuľke č. 9 je prípad použitia UC09 podrobne opísaný.

UC10 – Vygenerovať dokument

Systém vygeneruje nový dokument na základe zvolených dokumentov. V tabuľke č. 10 je prípad použitia UC10 podrobne opísaný.

UC11 – Vytvoriť prepojovací (virtuálny) dokument

Systém vytvorí dokument, ktorý slúži na umelé prepojenie jednotlivých dokumentov. Tento dokument neobsahuje samotné znalosti, ale slúži na združenie dokumentov obsahujúcich znalosti o určitej doméne. V tabuľke č. 11 je prípad použitia UC11 podrobne opísaný.

UC12 – Prepínať 2D/3D rozhranie

Systém používateľovi umožní prepínanie medzi zobrazením 2D a 3D. V tabuľke č. 12 je prípad použitia UC12 podrobne opísaný.

UC13 – Pohybovať sa v 3D priestore

Systém používateľovi umožní pohybovanie grafom v 3D priestore pozostávajúce z približovania a otáčania. V tabuľke č. 13 je prípad použitia UC13 podrobne opísaný.

UC14 – Zobraziť dokumenty

Systém umožní používateľovi zvoliť dokument, ktorý chce zobraziť v grafe spolu s jeho vzťahmi k ostatným dokumentom a zobrazí ho. V tabuľke č. 14 je prípad použitia UC14 podrobne opísaný. Obrázok č. 7 znázorňuje diagram aktivít pre daný prípad použitia.

UC15 – Prihlásiť do systému

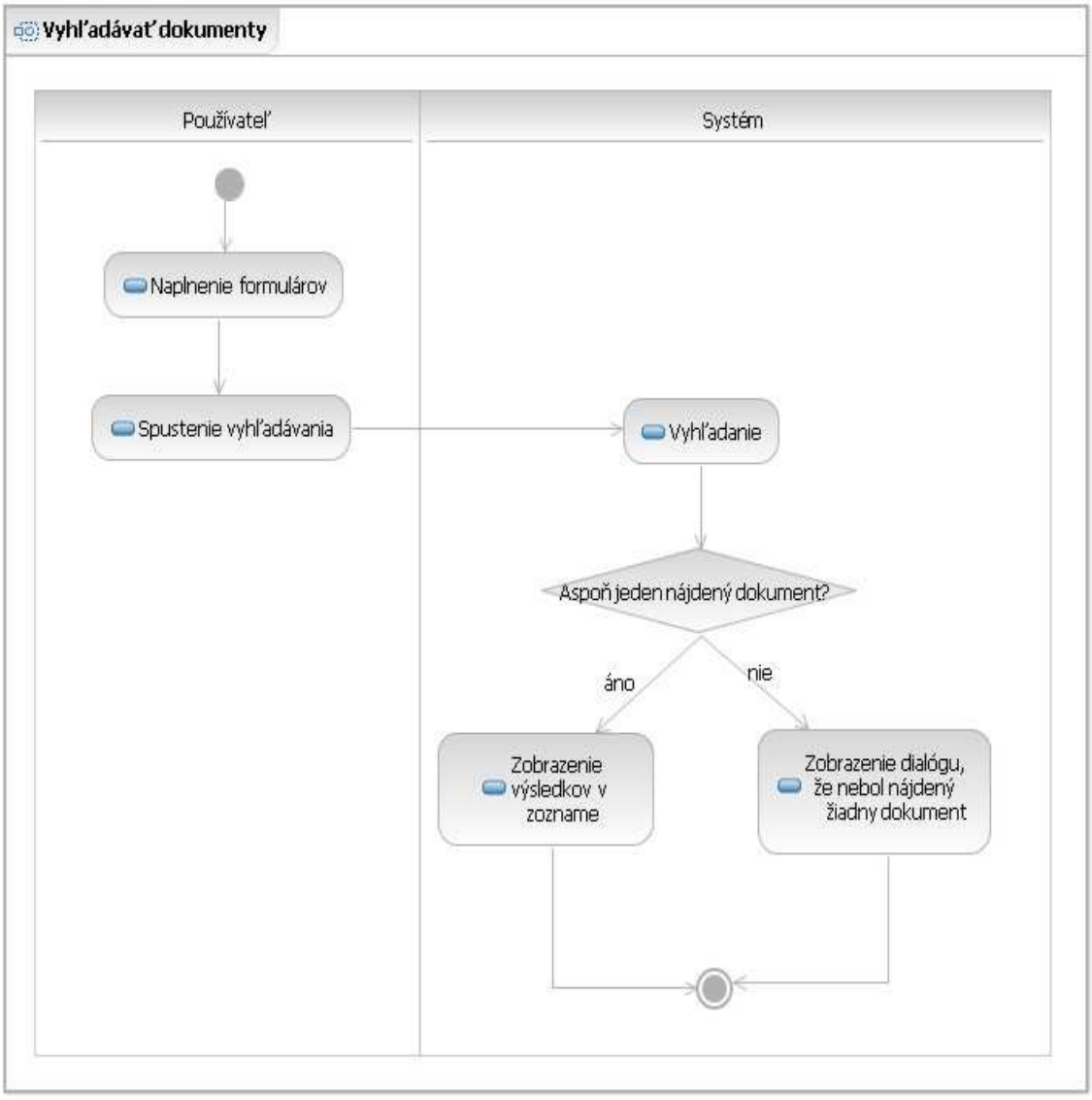
Používateľ, aby mohol plnohodnotne využívať funkcionalitu programu, sa pomocou systému môže prihlásiť do systému. V tabuľke č. 15 je prípad použitia UC15 podrobne opísaný.

UC16 – Zaregistrovať do systému

Používateľ sa pomocou systému môže zaregistrovať do systému. V tabuľke č. 16 je prípad použitia UC16 podrobne opísaný.

Tab. 1. Opis UC01 – Vyhľadávať dokumenty

Identifikátor	UC01	
Názov	Vyhľadávať dokumenty	
Opis	Používateľ v grafickom rozhraní vyplní formuláre a nechá systém podľa týchto informácií vyhľadať príslušné dokumenty.	
Priorita	Vysoká	
Vstup. podm.	Používateľ je prihlásený do systému.	
Výstup. podm.	Používateľ môže vidieť zoznam dokumentov, ktoré spĺňajú kritéria, ktoré navolil.	
Aktér	Používateľ	
Základná postupnosť	**Krok**	**Činnosť**
	1	Používateľ vyplní formuláre pre vyhľadanie dokumentov (autor, názov, kľúčové slová atď.).
	2	Používateľ dá vyhľadať dokumenty.
	3	Systém zobrazí zoznam dokumentov, ktoré vyhovujú vyhľadávacím informáciám.
	4	Prípad použitia končí.
Alternatívna postupnosť	**Krok**	**Činnosť**
	3.a	Systém nenájde žiadne vyhovujúce dokumenty a zobrazí upozornenie pre používateľa.
	4.a	Prípad použitia končí.
Rozširujúce body		



Obr. 4. Diagram aktivít pre UC01 – Vyhľadávať dokumenty

Tab. 2. Opis UC02 – Pridať dokumenty

Identifikátor	UC02	
Názov	Pridať dokumenty	
Opis	Používateľ zvolí dokument/y, ktoré chce pridať do systému.	
Priorita	Vysoká	
Vstup. podm.	Používateľ je prihlásený do systému.	
Výstup. podm.	Zvolené dokumenty sú pridané v systéme.	
Aktér	Používateľ	
Základná postupnosť	**Krok**	**Činnosť**
	1	Používateľ zvolí že chce pridať nové dokumenty do systému.
	2	Používateľ zvolí dokumenty a potvrdí ich vloženie do systému.
	3	Systém vloží dokumenty do databázy.
	4	Prípad použitia končí.
Alternatívna postupnosť	**Krok**	**Činnosť**
	3.a	Systém zistil, že aspoň jeden súbor nie je v podporovanom formáte a zobrazí správu pre používateľa .
	4.a	Pokračuje sa krokom 1 primárneho behu.
Rozširujúce body		

Tab. 3. Opis UC03 – Vymazať virtuálny dokument

Identifikátor	UC03	
Názov	Vymazať virtuálny dokument	
Opis	Používateľ chce vymazať existujúci virtuálny dokument zo systému.	
Priorita	Stredná	
Vstup. podm.	Používateľ je prihlásený do systému. Dokument, ktorý má byť vymazaný je v systéme a je zobrazený.	
Výstup. podm.	Systém vymaže zvolený virtuálny dokument.	
Aktér	Používateľ	
Základná postupnosť	**Krok**	**Činnosť**
	1	Používateľ zvolí zobrazený virtuálny dokument, ktorý chce vymazať.
	2	Používateľ zvolí možnosť vymazania tohto dokumentu.
	3	Systém vymaže tento uzol spolu so všetkými väzbami, ktoré do tohto uzla smerujú alebo vychádzajú.
	4	Systém vymaže tento virtuálny dokument.
	5	Prípad použitia končí.
Alternatívna postupnosť	**Krok**	**Činnosť**
	1.a	Používateľ zvolí viacero virtuálnych dokumentov, ktoré chce vymazať.
	2.a	Používateľ zvolí možnosť vymazania všetkých označených dokumentov.
	3.a	Systém vymaže všetky tieto uzly spolu so všetkými väzbami, ktoré do týchto uzlov smerujú alebo z nich vychádzajú.

	4.a	Systém vymaže tento virtuálny dokument.
	5.a	Prípad použitia končí.
Rozširujúce body		

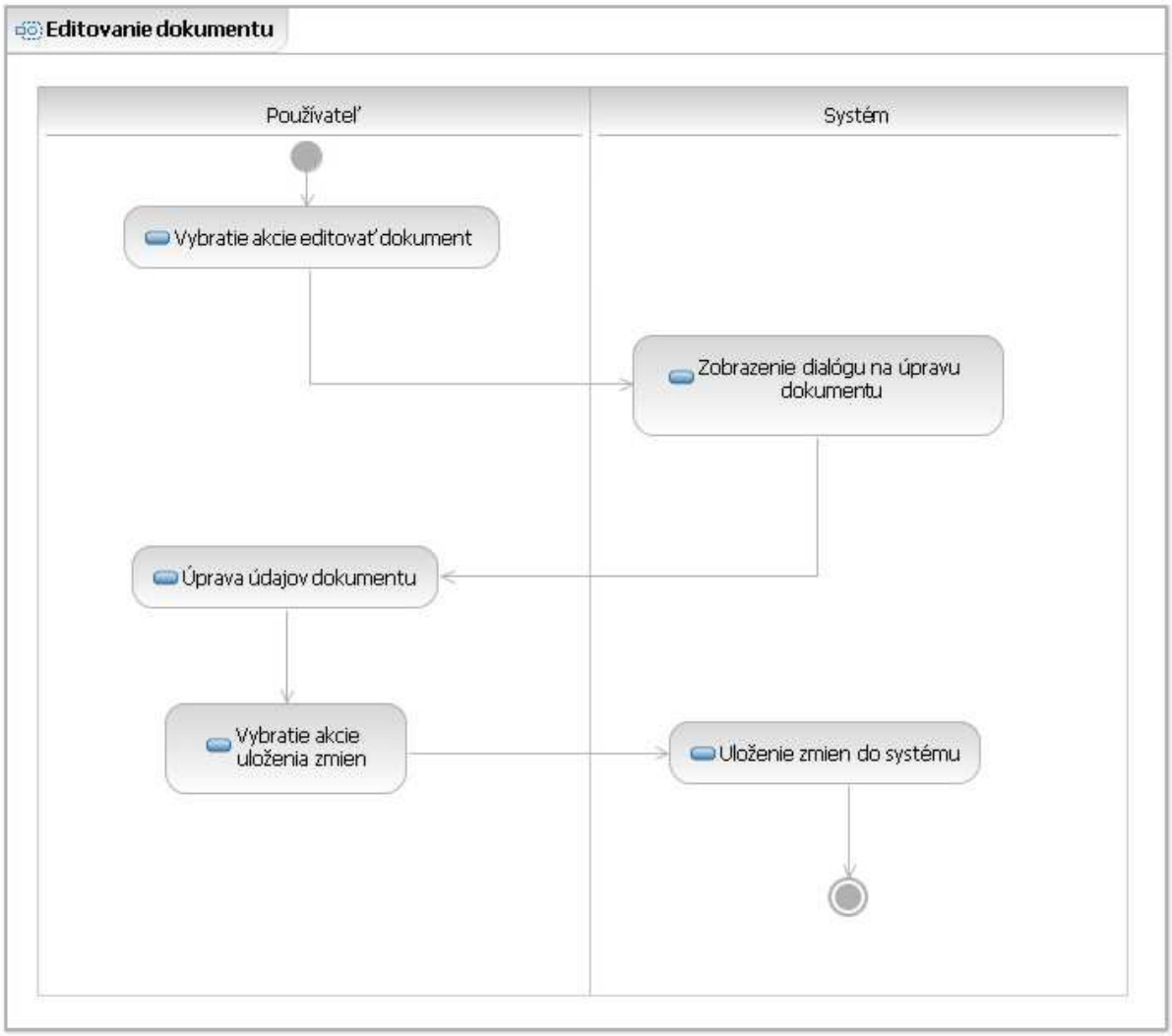
Tab. 4. Opis UC04 – Skryť dokumenty

Identifikátor	UC04	
Názov	Skryť dokumenty	
Opis	Používateľ chce v grafe dokumentov skryť jeden alebo viacero uzlov (t.j. dokumentov).	
Priorita	Vysoká	
Vstup. podm.	Používateľ je prihlásený do systému. Dokument, ktorý má byť skrytý je v systéme a je zobrazený.	
Výstup. podm.	Označené dokumenty spolu s väzbami, ktoré z týchto uzlov vychádzali alebo vchádzali nie sú zobrazené. Ak ostali dokumenty, ktoré nemajú žiadne väzby, tiež sú skryté.	
Aktér	Používateľ	
Základná postupnosť	**Krok**	**Činnosť**
	1	Používateľ zvolí zobrazený dokument, ktorý chce skryť.
	2	Používateľ zvolí možnosť skrytia tohto dokumentu.
	3	Systém nezobrazí tento uzol spolu so všetkými väzbami, ktoré do tohto uzla smerujú alebo vychádzajú. Ak ostali dokumenty, ktoré nemajú žiadne väzby, tiež ich systém nezobrazí.
	4	Prípad použitia končí.
Alternatívna postupnosť	**Krok**	**Činnosť**
	1.a	Používateľ zvolí viacero dokumentov, ktoré chce skryť .
	2.a	Používateľ zvolí možnosť skrytia všetkých označených dokumentov.
	3.a	Systém nezobrazí tieto uzly spolu so všetkými väzbami, ktoré do týchto uzlov smerujú alebo z nich vychádzajú. Ak ostali dokumenty, ktoré nemajú žiadne väzby, tiež ich systém nezobrazí.
	4.a	Prípad použitia končí.
Rozširujúce body		

Tab. 5. Opis UC05 – Editovať dokument

Identifikátor	UC05
Názov	Editovať dokument
Opis	Používateľ upraví metadáta dokumentu, ako napríklad názov dokumentu, meno autora, kľúčové slová atď., alebo priamo jeho obsah.
Priorita	Stredná
Vstup. podm.	Používateľ je prihlásený v systéme. Dokument, ktorý chce používateľ upraviť je viditeľný v aktuálnom zobrazení grafu.
Výstup. podm.	Dokument je upravený a uložený v systéme.

Aktér	Používateľ	
Základná postupnosť	**Krok**	**Činnosť**
	1	Používateľ zvolí akciu editovať dokument.
	2	Systém zobrazí dialóg na úpravu dokumentu.
	3	Používateľ upraví údaje popisujúce dokument (jeho názov, meno autora, atď.), prípadne obsah dokumentu.
	4	Používateľ v dialógu zvolí akciu uložiť.
	5	Systém uloží zmeny, ktoré zadal používateľ.
	6	Prípad použitia končí.
Alternatívna postupnosť	**Krok**	**Činnosť**
Rozširujúce body	3.	Možné rozšírenie o nahratie novšej verzie dokumentu.



Obr. 5. Diagram aktivít pre UC05 – Editovať dokumenty

Tab. 6. Opis UC06 – Aktualizovať dokument

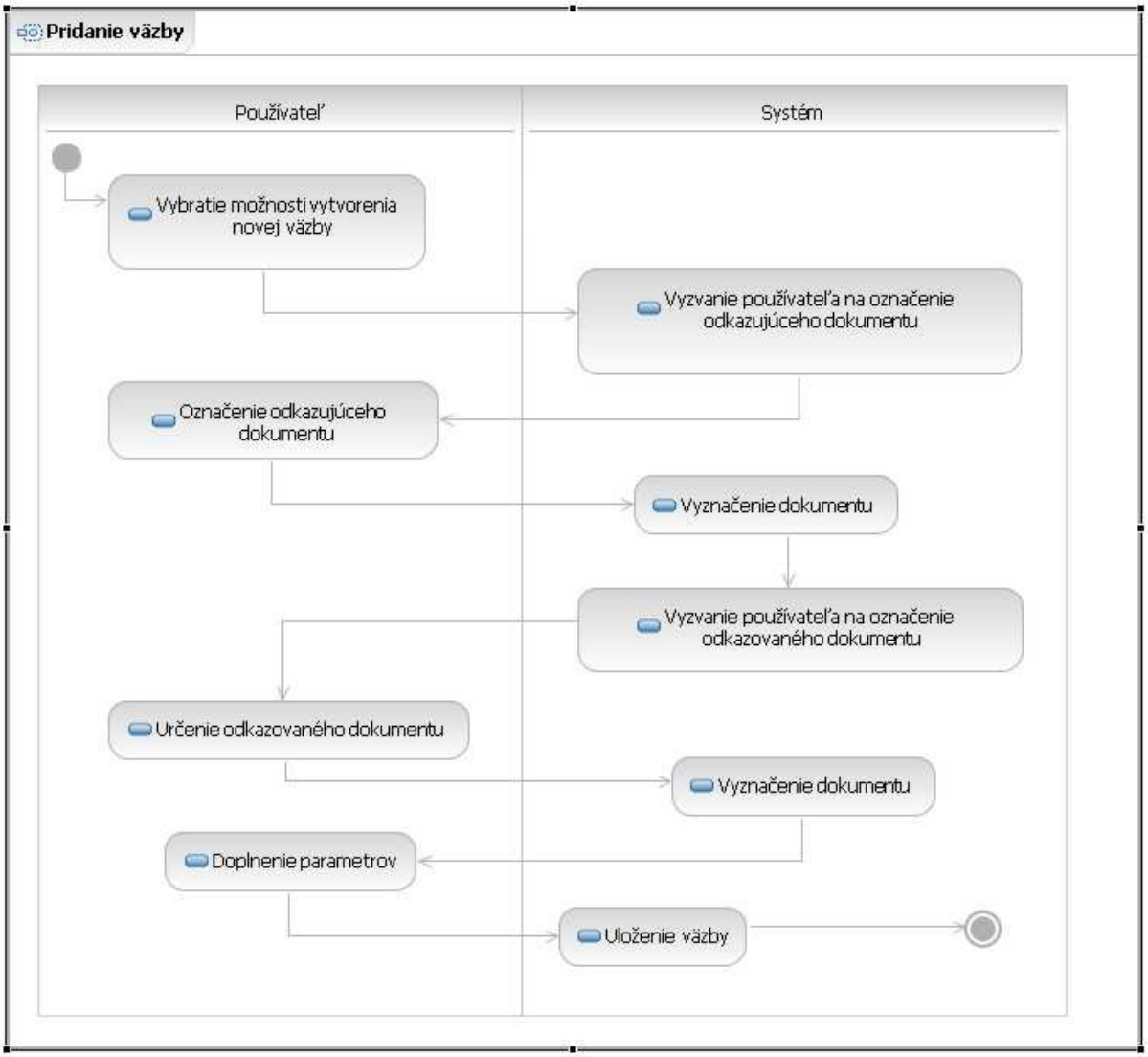
Identifikátor	UC06
Názov	Aktualizovať dokument

Opis	Používateľ nahrá do systému novú verziu dokumentu. Systém novú verziu zaindexuje namiesto pôvodného dokumentu a starú verziu odstráni.	
Priorita	Stredná	
Vstup. podm.	Tento prípad použitia rozširuje UC05 Editovať dokument v bode 3.	
Výstup. podm.	Súbor dokumentu je nahradený novšou verziou a je korektne zindexovaný v systéme.	
Aktér	Používateľ	
Základná postupnosť	**Krok**	**Činnosť**
	1	Používateľ zvolí akciu pre nahratie novšej verzie dokumentu.
	2	Systém používateľovi ponúkne dialóg na výber súboru.
	3	Používateľ vyberie a označí súbor určený na nahratie do systému.
	4	Systém zaindexuje nový súbor a starý odstráni. Novým súborom taktiež prepíše jeho neaktuálnu verziu v databáze.
	5	Systém zaktualizuje väzby pre novú verziu dokumentu.
	6	Prípad použitia končí.
Alternatívna postupnosť	**Krok**	**Činnosť**
Rozširujúce body		

Tab. 7. Opis UC07 – Vytvoriť väzbu medzi dokumentmi

Identifikátor	UC07	
Názov	Vytvoriť väzbu medzi dokumentmi	
Opis	Používateľ vytvorí prepojenie medzi dokumentmi, ktoré spolu súvisia, ale v systéme zatiaľ prepojené nie sú.	
Priorita	Stredná až vysoká	
Vstup. podm.	Používateľ je prihlásený v systéme. Obidva dokumenty, medzi ktorými má byť vytvorená väzba sú zobrazené.	
Výstup. podm.	Je vytvorená a uložená nová väzba medzi dokumentmi.	
Aktér	Používateľ	
Základná postupnosť	**Krok**	**Činnosť**
	1	Používateľ zvolí možnosť vytvoriť novú väzbu medzi dokumentmi.
	2	Systém používateľa vyzve, aby vyznačil odkazujúci dokument.
	3	Používateľ vyberie odkazujúci dokument spomedzi zobrazených dokumentov.
	4	Systém na ploche vyznačí odkazujúci dokument a vyzve používateľa na označenie odkazovaného dokumentu.
	5	Používateľ vyberie odkazovaný dokument spomedzi zobrazených dokumentov.
	6	Systém na ploche vyznačí odkazovaný dokument a vyzve používateľa na doplnenie parametrov väzby (typ, sila, obojsmernosť, atď.).
	7	Používateľ zadá ostatné parametre väzby.
	8	Systém novú väzbu vytvorí a uloží.
	9	Prípad použitia končí.
Alternatívna	**Krok**	**Činnosť**

postupnosť		
Rozširujúce body		



Obr. 6. Diagram aktivít pre UC07 – Vytvoriť väzbu medzi dokumentmi

Tab. 8. Opis UC08 – Editovať väzbu medzi dokumentmi

Identifikátor	UC08	
Názov	Editovať väzbu medzi dokumentmi	
Opis	Používateľ upraví atribúty niektorej väzby na základe nových zistení, prípadne iných okolností.	
Priorita	Stredná až vysoká	
Vstup. podm.	Používateľ je prihlásený do systému. Väzba, ktorú chce používateľ upraviť je zobrazená.	
Výstup. podm.	Atribúty väzby sú zmenené a tieto zmeny sú uložené v systéme.	
Aktér	Používateľ	
Základná postupnosť	**Krok**	**Činnosť**
	1	Používateľ označí väzbu, ktorej atribúty si želá upraviť.

	2	Používateľ vyvolá akciu upraviť väzbu.
	3	Systém zobrazí dialóg na úpravu väzby.
	4	Používateľ upraví atribúty väzby a zvolí uloženie vykonaných zmien.
	5	Systém zmenenú väzbu uloží.
	6	Prípad použitia končí.
Alternatívna postupnosť	**Krok**	**Činnosť**
Rozširujúce body		

Tab. 9. Opis UC09 – Odstrániť väzbu medzi dokumentmi

Identifikátor	UC09	
Názov	Odstrániť väzbu medzi dokumentmi	
Opis	Používateľ zistí, že niektorá väzba nevyjadruje skutočnú súvislosť medzi dokumentmi a tak ju odstráni. Takáto situácia môže nastať napríklad preto, že systém niektoré väzby medzi dokumentmi vytvoril automaticky, ale tieto neboli vytvorené správne.	
Priorita	Stredná až vysoká	
Vstup. podm.	Používateľ je prihlásený do systému. Väzba, ktorú chce používateľ vymazať je zobrazená.	
Výstup. podm.	Väzba je odstránená zo systému a nie je zobrazená.	
Aktér	Používateľ	
Základná postupnosť	**Krok**	**Činnosť**
	1	Používateľ označí väzbu, určenú na vymazanie.
	2	Používateľ vyvolá akciu odstrániť väzbu.
	3	Systém vyzve používateľa na potvrdenie svojej voľby.
	4	Používateľ voľbu potvrdí.
	5	Systém vyznačenú väzbu odstráni.
	6	Prípad použitia končí.
Alternatívna postupnosť	**Krok**	**Činnosť**
	4.a	Používateľ voľbu odmietne.
	5.a	Prípad použitia končí.
Rozširujúce body		

Tab. 10. Opis UC10 – Vygenerovať dokument

Identifikátor	UC 10
Názov	Vygenerovať dokument
Opis	Vygenerovanie nového dokumentu na základe zvolených dokumentov.
Priorita	Vysoká
Vstup. podm.	Používateľ je prihlásený do systému. Musí byť zvolený aspoň jeden dokument.
Výstup. podm.	Je vytvorený dokument na základe zvolených dokumentov.
Aktér	Používateľ

Základná postupnosť	**Krok**	**Činnosť**
	1	Používateľ zvolí vytvorenie nového dokumentu.
	2	Systém ponúkne časti jednotlivých dokumentov pre ďalšiu selekciu.
	3	Používateľ vyberie časti dokumentov.
	4	Systém vygeneruje nový dokument.
	5	Systém zobrazí novo vygenerovaný dokument.
	6	Prípad použitia končí.
Alternatívna postupnosť	**Krok**	**Činnosť**
Rozširujúce body		

Tab. 11. Opis UC11 – Vytvoriť prepojovací (virtuálny) dokument

Identifikátor	UC 11	
Názov	Vytvoriť prepojovací (virtuálny) dokument	
Opis	Vytvorenie dokumentu, ktorý slúži na umelé prepojenie jednotlivých dokumentov.	
Priorita	Vysoká	
Vstup. podm.	Používateľ je prihlásený do systému. Musia byť načítané a zobrazené aspoň dva dokumenty.	
Výstup. podm.	Je vytvorený virtuálny dokument a je prístupný používateľovi spolu s ostatnými dokumentmi.	
Aktér	Používateľ	
Základná postupnosť	**Krok**	**Činnosť**
	1	Používateľ zvolí vytvorenie virtuálneho dokumentu.
	2	Používateľ zvolí z existujúcich dokumentov časti, ktoré sa využijú.
	3	Systém vytvorí virtuálny dokument a pridá ho k existujúcim.
	4	Systém zobrazí virtuálny dokument.
	5	Prípad použitia končí.
Alternatívna postupnosť	**Krok**	**Činnosť**
	2.a	Používateľ vytvorí väzby od zobrazených dokumentov k novému virtuálnemu dokumentu
	3.a	Pokračuje sa bodom 3.
Rozširujúce body		

Tab. 12. Opis UC12 – Prepínať 2D/3D rozhranie

Identifikátor	UC 12
Názov	Prepínať 2D/3D rozhranie
Opis	Prepínanie medzi zobrazením 2D a 3D.
Priorita	Vysoká
Vstup. podm.	Používateľ je prihlásený do systému. Musí existovať zobrazovaný graf.
Výstup. podm.	Zobrazenie je prepnuté do zvoleného režimu.
Aktér	Používateľ

Základná postupnosť	**Krok**	**Činnosť**
	1	Používateľ zvolí prepnutie do 3D.
	2	Systém zobrazí graf v 3D.
	3	Prípad použitia končí.
Alternatívna postupnosť	**Krok**	**Činnosť**
	1.a	Používateľ zvolí prepnutie do 2D.
	2.a	Systém zobrazí graf v 2D.
	3.a	Prípad použitia končí.
Rozširujúce body		

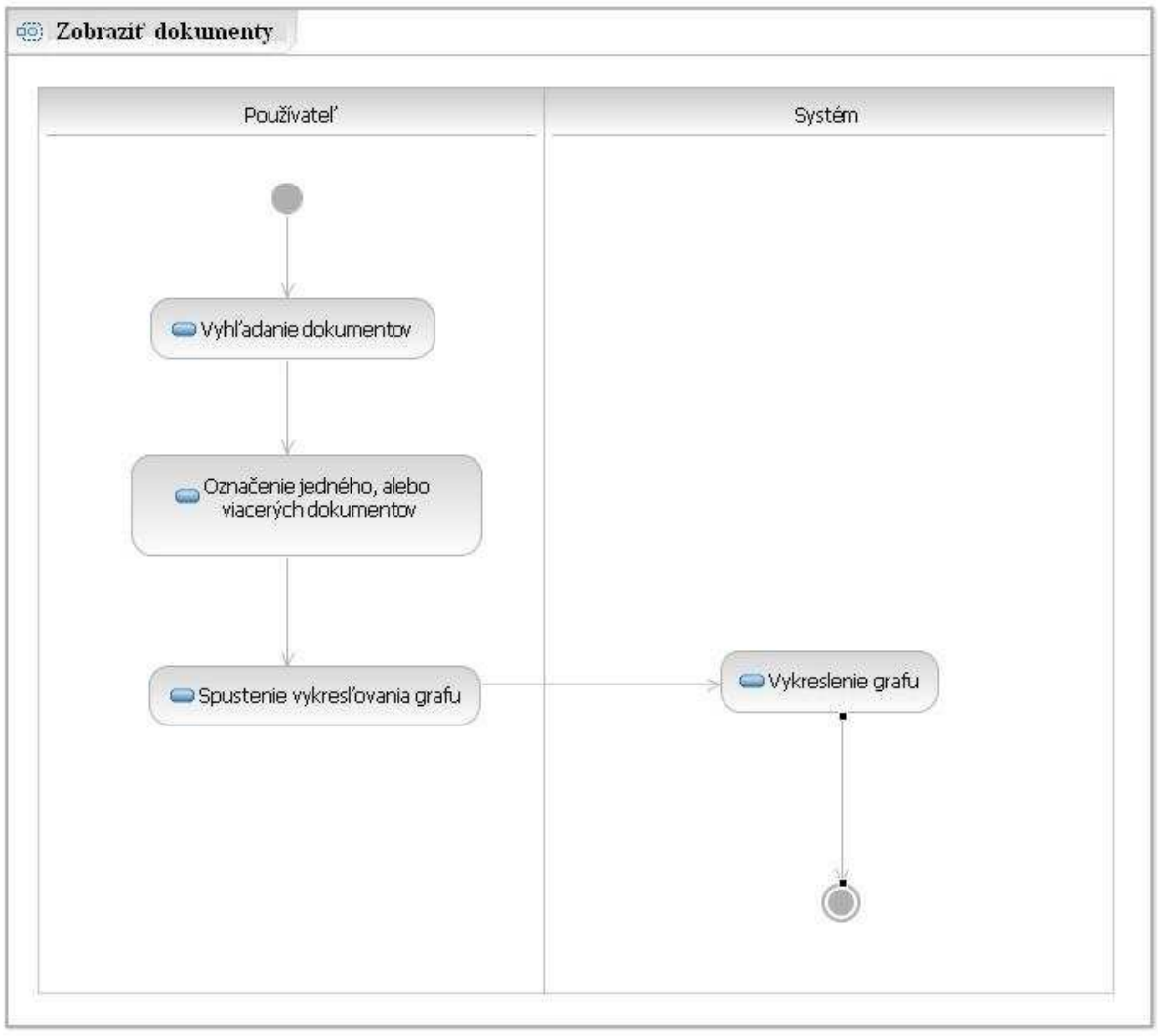
Tab. 13. Opis UC13 – Pohybovať sa v 3D priestore

Identifikátor	UC 13	
Názov	Pohybovať sa v 3D priestore	
Opis	Pohybovanie grafom v 3D priestore pozostávajúce z približovania a otáčania.	
Priorita	Vysoká	
Vstup. podm.	Používateľ je prihlásený do systému. Zobrazenie musí byť prepnuté do 3D zobrazenia. Musí existovať vytvorený graf.	
Výstup. podm.	Používateľovi je dovolené sa pohybovať v 3D priestore.	
Aktér	Používateľ	
Základná postupnosť	**Krok**	**Činnosť**
	1	Používateľ použije na otáčanie grafu myš alebo klávesy.
	2	Systém zobrazí požadované otočenie.
	3	Prípad použitia končí.
Alternatívna postupnosť	**Krok**	**Činnosť**
	1.a	Používateľ použije na priblíženie alebo oddialenie grafu koliesko na myši alebo klávesy.
	2.a	Systém zobrazí požadované priblíženie.
	3.a	Prípad použitia končí.
Rozširujúce body		

Tab. 14. Opis UC14 – Zobraziť dokumenty

Identifikátor	UC14	
Názov	Zobraziť dokumenty	
Opis	Používateľ zvolí dokument, ktorý chce zobraziť v grafe spolu s jeho vzťahmi k ostatným dokumentom.	
Priorita	Vysoká	
Vstup. podm.	Používateľ je prihlásený do systému. Na zozname vyhľadaných dokumentov sa nachádza aspoň jedna položka.	
Výstup. podm.	Systém zobrazí vybrané dokumenty aj s príslušnými vzťahmi.	
Aktér	Používateľ	
Základná postupnosť	**Krok**	**Činnosť**
	1	Používateľ vyhľadá dokumenty pomocou primárneho toku UC01.

	2	Používateľ si zvolí jeden dokument ako centrálny a dá ho zobraziť.
	3	Systém zobrazí v grafe centrálny dokument spolu so všetkými dokumentmi, ktoré sú priamo spojené väzbou s centrálnym dokumentom.
	4	Prípad použitia končí.
Alternatívna postupnosť	**Krok**	**Činnosť**
	2.a	Používateľ si zvolí možnosť zobraziť viaceré (prípadne všetky vyhľadané) dokumenty.
	3.a	Systém zobrazí plochu, na ktorej budú všetky zvolené dokumenty považované za centrálne a následne sa postupuje tak isto ako v kroku 3 primárnej postupnosti.
Rozširujúce body		



Obr. 7. Diagram aktivít pre UC14 – Zobraziť dokumenty

Tab. 15. Opis UC15 – Prihlásiť do systému

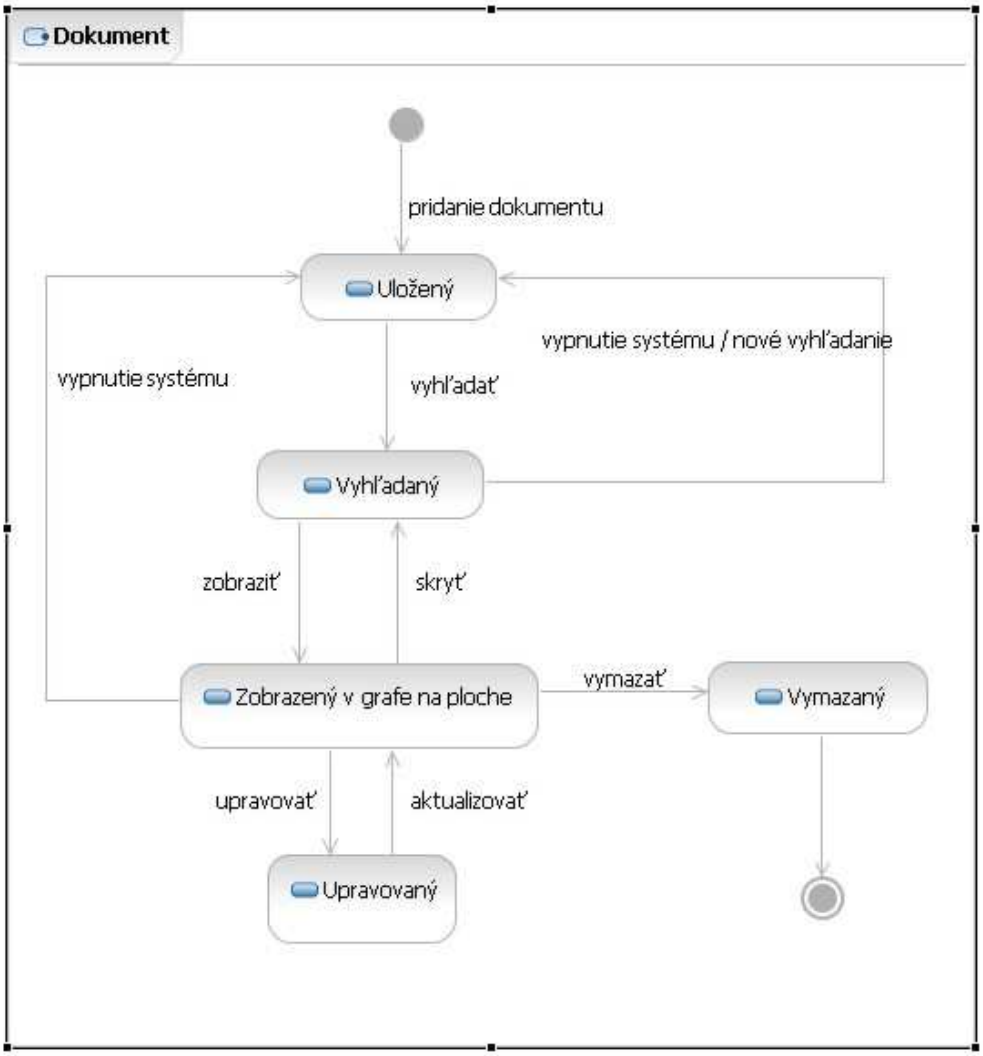
Identifikátor	UC15
Názov	Prihlásiť do systému
Opis	Používateľ sa chce prihlásiť do systému, aby mohol plnohodnotne využívať

	funkcionalitu programu.	
Priorita	Vysoká	
Vstup. podm.	Používateľ je zaregistrovaný v systéme.	
Výstup. podm.	Používateľ je prihlásený v systéme	
Aktér	Používateľ	
Základná postupnosť	**Krok**	**Činnosť**
	1	Používateľ zvolí možnosť, prihlásiť do systému.
	2	Používateľ napíše meno a heslo a potvrdí prihlásenie.
	3	Systém overí, že zadané údaje sú správne.
	4	Systém prihlási používateľa do systému
	5	Prípad použitia končí.
Alternatívna postupnosť	**Krok**	**Činnosť**
	3.a	Systém zistil, že údaje nie sú správne a upovedomí o tom používateľa.
	4.a	Pokračuje sa krokom 2 primárneho toku.
Rozširujúce body		

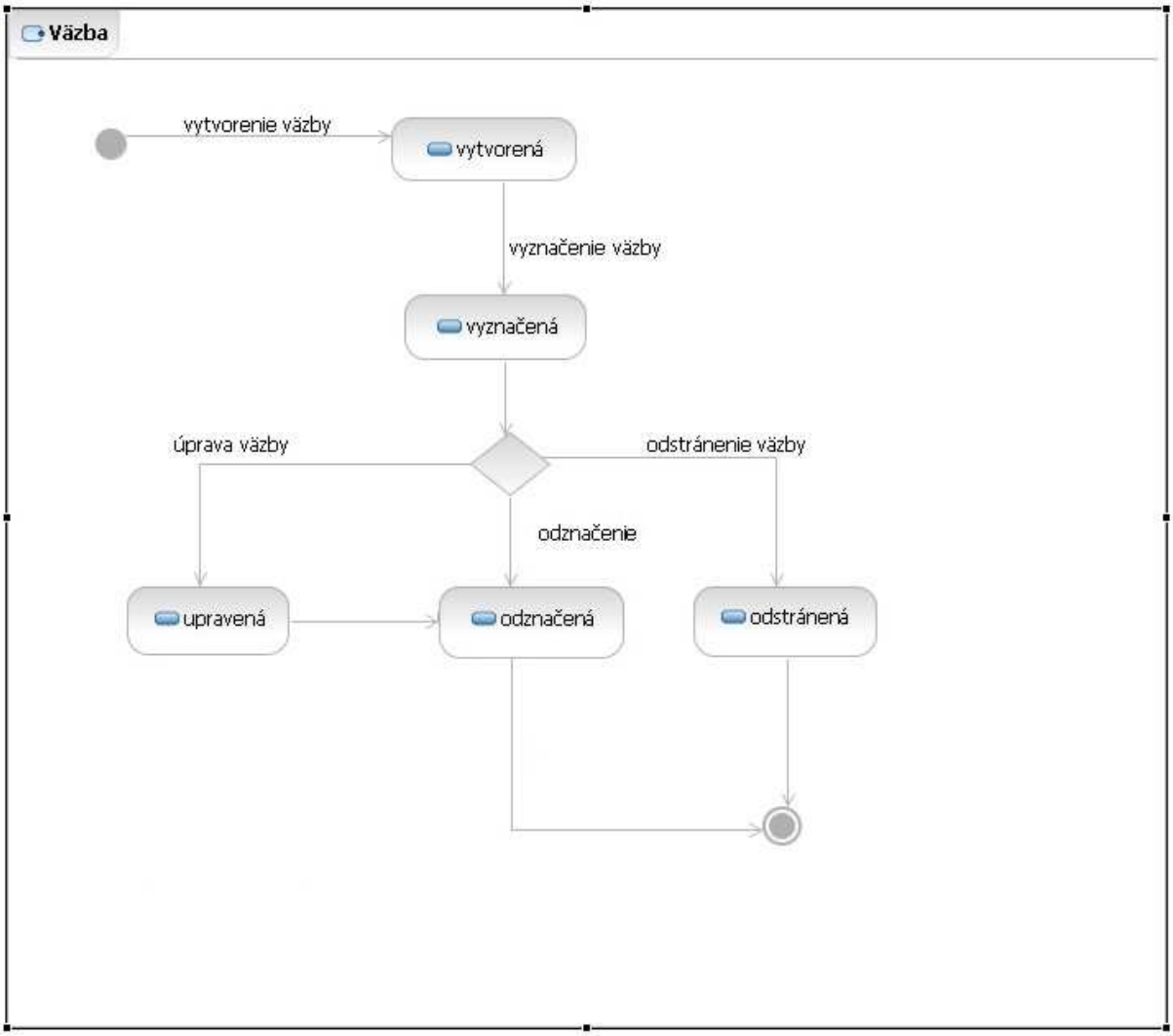
Tab. 16. Opis UC16 – Zaregistrovať do systému

Identifikátor	UC16	
Názov	Zaregistrovať do systému	
Opis	Používateľ sa chce zaregistrovať do systému.	
Priorita	Vysoká	
Vstup. podm.	Používateľ nie je zaregistrovaný v systéme.	
Výstup. podm.	Používateľ je zaregistrovaný v systéme.	
Aktér	Používateľ	
Základná postupnosť	**Krok**	**Činnosť**
	1	Používateľ zvolí možnosť zaregistrovať do systému.
	2	Systém zobrazí registračný formulár.
	3	Používateľ vyplní zobrazené formuláre.
	4	Systém overí, či sú tieto údaje správne.
	5	Systém odošle kontrolný e-mail na adresu, ktorú zadal používateľ a upovedomí ho o tom.
	6	Používateľ navštívi odkaz, ktorý dostal v tele e-mailu a tým sa stane platne zaregistrovaným.
	7	Prípad použitia končí.
Alternatívna postupnosť	**Krok**	**Činnosť**
	4.a	Systém zistil, že vyplnené údaje nie sú správne (nesedí štruktúra e-mailu, už existuje používateľ s daným menom atď.) a upovedomí o tom používateľa.
	5.a	Pokračuje sa krokom 2 primárneho toku.
Rozširujúce body		

Pre entity Dokument a Väzba sme vytvorili aj stavové diagramy, aby sme znázornili ďalšie kľúčové záležitosti týkajúce sa špecifikácie správania systému. Diagramy predstavujú obrázky č. 8 a 9. Entity a dátový model sú popísané v nasledujúcej kapitole.



Obr. 8. Stavový diagram pre dokument

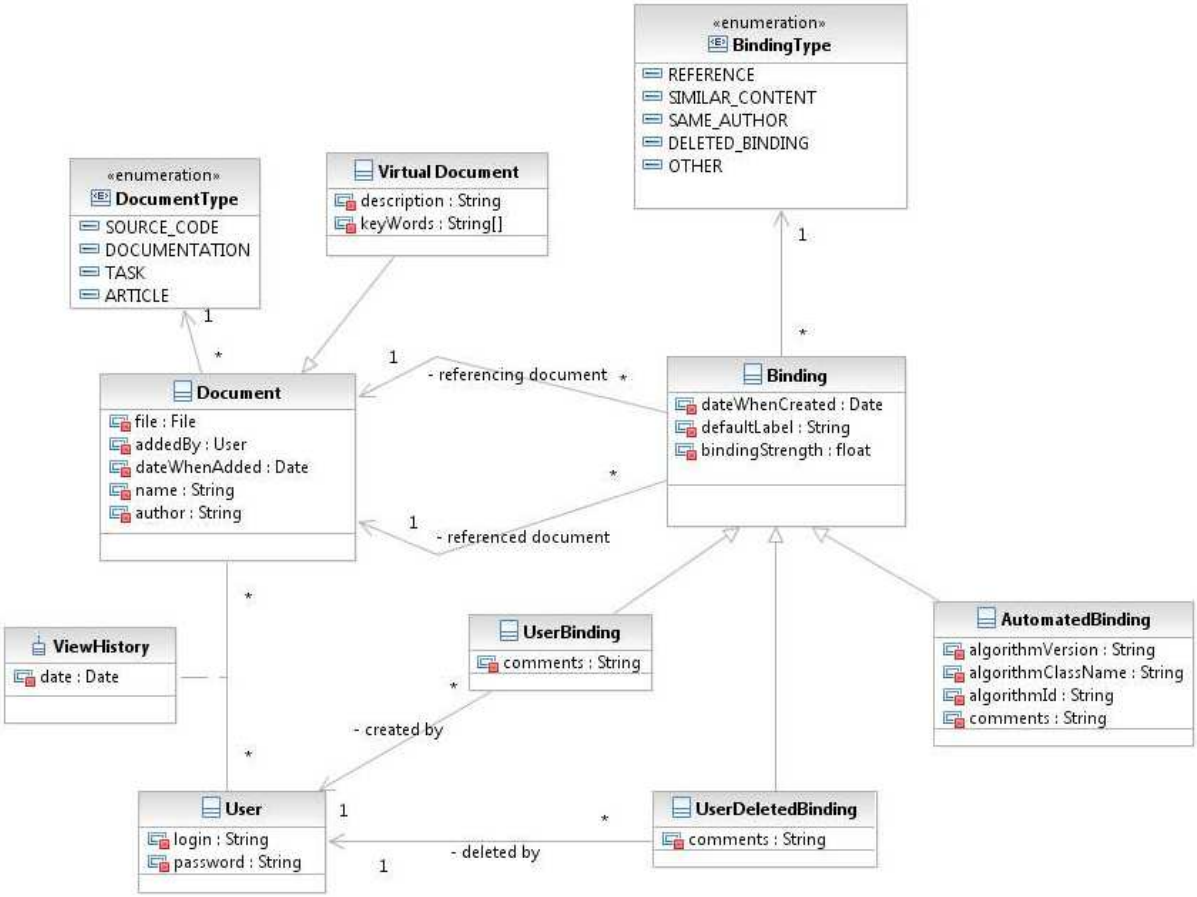


Obr. 9. Stavový diagram pre väzbu

2.3 Špecifikácia údajov v systéme

V tejto kapitole sa nachádza dátový model systému spolu s opisom jednotlivých tried, ich atribútov, ohraničení a iných podmienok. Na konci kapitoly uvádzame odlišnosti špecifikácie údajov nášho systému a systémov, z ktorých vychádzame.

2.3.1 Dátový model systému



Obr. 10. Dátový model

Dátový model zobrazený na obrázku č. 10 bol vytvorený na základe návrhu minuloročného tímu č. 20. Nasleduje popis jednotlivých entít prevzatý z dokumentácie.

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):

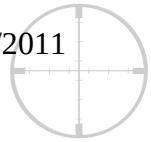
UserBinding – väzba vytvorená niektorým používateľom. Eviduje sa okrem iného aj používateľ, ktorý väzbu vytvoril a jeho komentár k väzbe. Nesmie byť typ väzby DELETED_BINDING.

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, algorithmVersion

User – reprezentuje používateľa systému.



Povinné atribúty: login, password

Nad existujúcim dátovým modelom navrhujeme nasledovné zmeny:

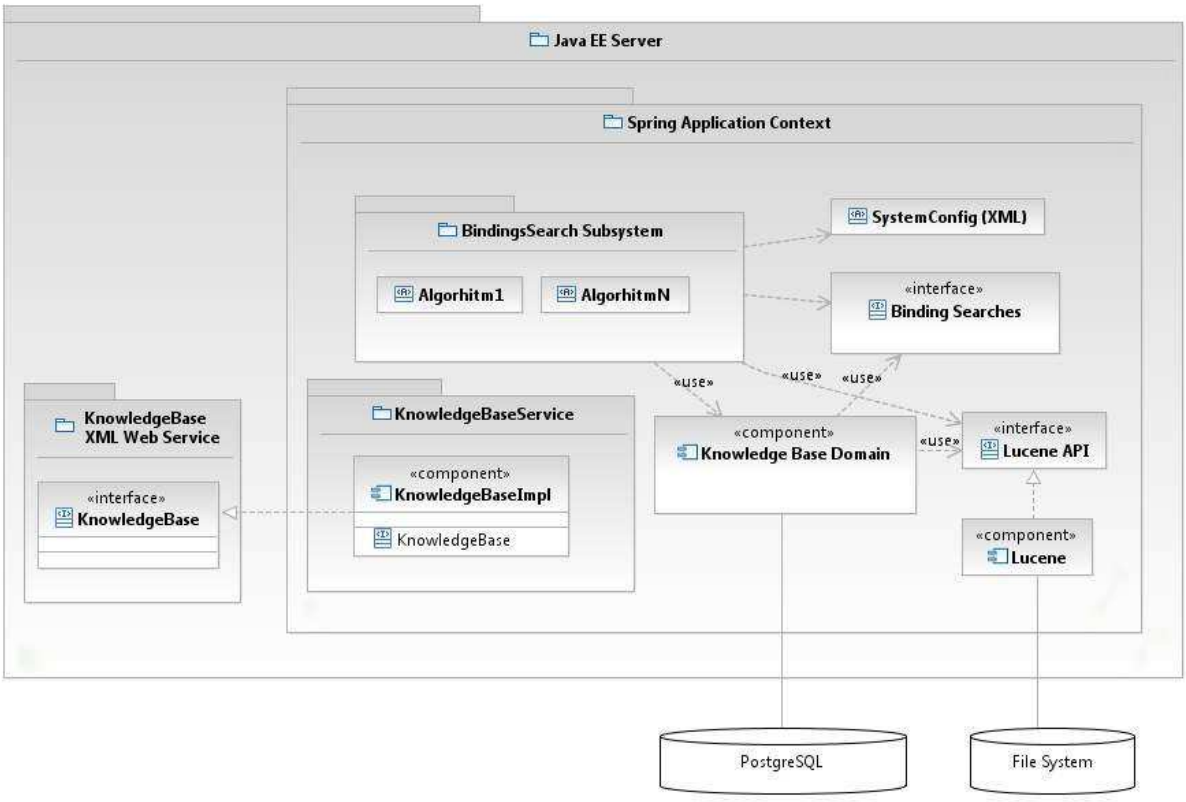
- Vytvorenie entity **DocumentType**, ktorá by reprezentovala typ vloženého dokumentu. Potreba definovania typu dokumentu vznikla na základe úlohy generovania nových dokumentov. Vďaka typu dokumentu je možné definovať rôzne spôsoby spájania týchto dokumentov. Takisto na základe typu dokumentu bude možné ich rôzne zobrazenie v grafe.
- Vytvorenie asociačnej triedy **ViewHistory**, ktorá reprezentuje históriu prehliadania jednotlivých dokumentov používateľmi.

3 Návrh riešenia

3.1 Návrh architektúry

Základným rozdelením architektúry vytváraného systému je rozdelenie na klientskú a serverovú časť. Obe tieto časti nadväzujú na prácu minuloročných tímov a ich návrhov. V závere kapitoly znázorňujeme komunikáciu klientskej a serverovej časti.

3.1.1 Server



Obr. 11. Server

Serverová časť aplikácie znázornená na obrázku č. 11 z veľkej časti preberá štruktúru z minuloročného projektu. Ako hlavný kontajner je na server strane použitý Java aplikačný server. V aplikačnom serveri bude systém bežať ako Spring aplikácia. 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 vlákna, reprezentujúce jednotlivé algoritmy vyhľadávania väzieb. Pri zavádzaní nového algoritmu je potrebné naprogramovať triedu tohto algoritmu,

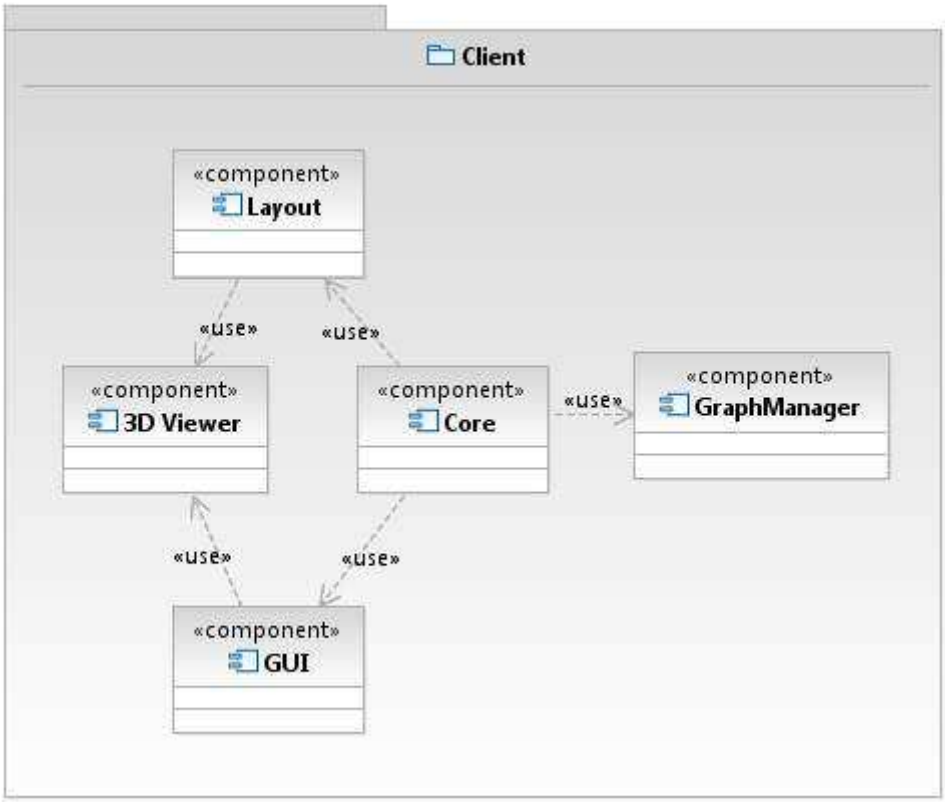
nakonfigurovať ju v XML, prekompilovať celý systém a opätovne nasadiť aplikáciu. Výhodou oddelenia je, že pri pridávaní nového algoritmu nebudú zasiahnuté iné časti systému. Vyhľadávacím algoritmom je k dispozícii Lucene API, ako aj KnowledgeBaseDomain API. Lucene používa vyhradený adresár v súborovom systéme pre ukladanie svojich indexov. KnowledgeBaseDomain je vrstva doménových objektov a business logiky okolo nich. Perzistencia je vyriešená pomocou frameworku Hibernate v spojení s PostgreSQL relačnou databázou. KnowledgeBaseService je vrstva, ktorá vystavuje rozhranie KnowledgeBase.

Nami navrhovaná zmena predstavuje zmenu technológie vzdialeného volania metód z použitého Java Remote Method Invocation na multiplatformový štandard XML Web Services, ktorý nám umožní volanie metód bázy znalostí na ľubovoľnej implementačnej platforme klienta, keďže komunikácia prebieha na báze SOAP (XML) nad protokolom HTTP.

3.1.2 Klient

Vzhľadom na rozdelenie serverovej a klientskej časti a možnosti použitia viacero platforiem / programovacích jazykov bude možné jednotlivých klientov jednoducho nahradzovať, prípadne používať viacero klientov naraz.

Vzhľadom na to, že nie je jednoznačne definovaná implementačná platforma, navrhovaná architektúra je na vyššej úrovni abstrakcie. Podobne ako serverová časť, vychádza z návrhu minuloročného tímu.



Obr. 12. Klient

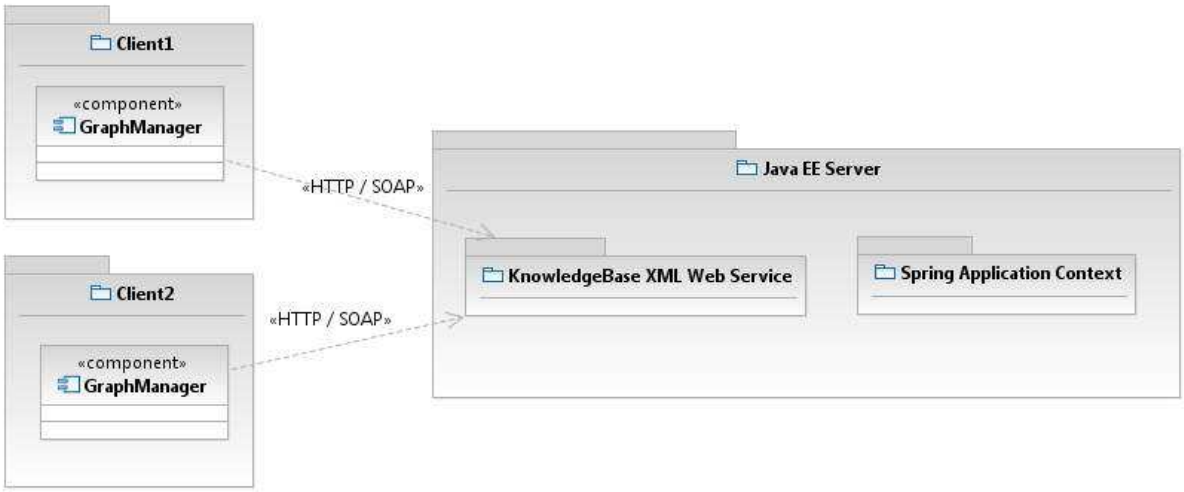
Popis jednotlivých komponentov klienta zobrazeného na obrázku č. 12:

- Core – zabezpečuje samotný beh aplikácie, riadenie ostatných komponentov, ich synchronizáciu a pod.
- Layout – zabezpečuje rozmiestňovanie uzlov v trojdimenzionálnom priestore
- 3D Viewer – predstavuje komponent, ktorý bude realizovať vizualizáciu grafu
- GUI – komponent, ktorý bude obsahovať interakciu s používateľom nad zobrazovaním grafom
- GraphManager – komponent, ktorého hlavnou zodpovednosťou bude komunikácia so serverovou časťou

Implementácia bude prioritne prebiehať na platforme C++. Hlavným cieľom však je, aby bolo možné klientov jednoducho prepínať a v prípade potreby implementovať klienta aj inom prostredí. Vďaka tomuto prístupu bude umožnená aj súbežná činnosť už existujúceho 2D klienta a nami implementovaného 3D klienta.

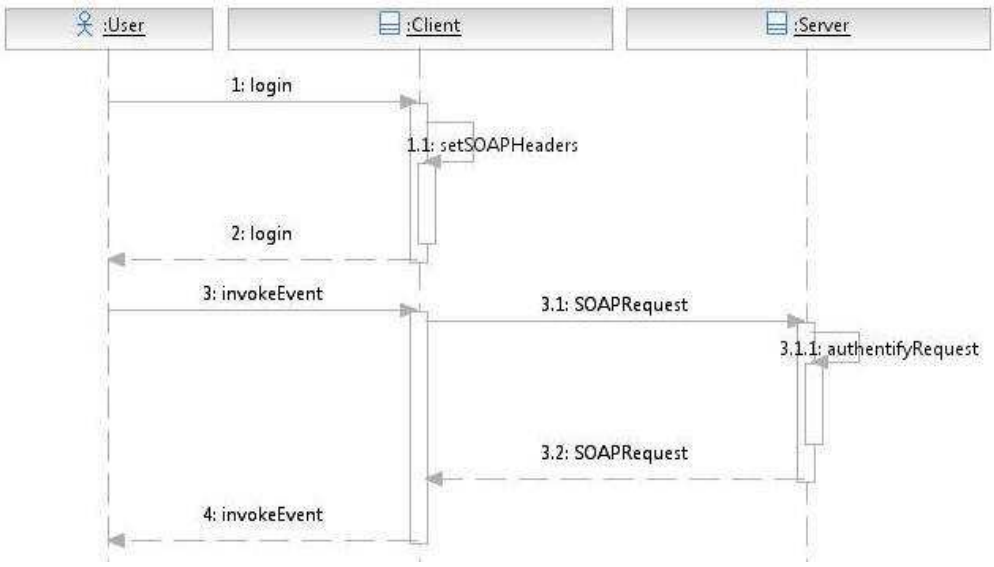
3.1.3 Komunikácia server – klient

Komunikácia medzi serverovou a klientskou aplikáciou, ako už bolo vyššie spomenuté, bude prebiehať prostredníctvom protokolu SOAP (Simple Object Access Protocol) nad protokolom HTTP. Komunikácia bude prebiehať medzi komponentom webovej služby na strane servera a komponentom graf manažéra na strane klienta. Znázorňujeme ju na obrázku č. 13.



Obr. 13. Komunikácia klient-server

Keďže používanie serverovej aplikácie bázy znalostí si vyžaduje autentifikáciu používateľa, budú autentifikačné údaje súčasťou každej SOAP správy. Údaje, ktoré slúžia na identifikáciu používateľa budú zapísané v časti SOAP Header. Tieto údaje zadá používateľ pri spúšťaní klienta. Pri každom volaní nejakej vzdialenej metódy serverová časť overí používateľa a vráti odpoveď v závislosti od výsledku autentifikácie. Komunikácia je znázornená na obrázku č. 14.



Obr. 14. Sekvenčný diagram komunikácie klient-server

4 Prototyp

4.1 Ciele prototypovania

Naším cieľom v tejto fáze tvorby projektu bolo vytvoriť prototyp, ktorý bude ponúkať základnú funkcionalitu výsledného systému. Použijeme metódu evolučného prototypovania, teda hotový prototyp budeme postupne ďalej upravovať a vylepšovať až do stavu, ktorý bude zodpovedať špecifikácii systému - teda výsledný produkt. Vytvorenie prototypu je dôležité hlavne pre zákazníka (resp. nášho pedagogického vedúceho tímu), aby videl priebežné výsledky našej práce. Pri prezentácii prototypu je tiež možné odhaliť prípadné nezrovnalosti v požiadavkách zákazníka na hotový systém oproti predstavám vývojárskeho tímu.

Témou nášho projektu je prepojenie systému zobrazujúceho dokumenty a ich vzájomné väzby v 2D grafe a programu zobrazujúceho 3D graf. Prototyp by teda mal ukázať práve toto prepojenie existujúcich systémov do jedného celku. Na splnenie tohto cieľa je potrebné vytvoriť server, s ktorým bude klient komunikovať. Na tomto serveri bude bežať databáza, v ktorej bude uložená tzv. báza znalostí. Sú to dokumenty, v ktorých bude používateľ môcť vyhľadávať a zobrazovať ich v grafe. Ďalej bude potrebné upraviť existujúcich klientov tak, aby medzi sebou vedeli komunikovať. To je nevyhnutné na to, aby mohol používateľ premietnuť vytvorený 2D graf do 3D prostredia.
Ďalším cieľom prototypu je implementovať algoritmy na extrakciu informácií z dokumentu, presnejšie kľúčových slov priamo z textu dokumentu.

Prototyp nebude zahŕňať prihlásenie používateľa do systému, ani viacpoužívateľské prostredie. Taktiež nebude ešte implementované hľadanie väzieb medzi dokumentmi a pridávanie nových dokumentov do systému.

4.2 2D klient

Táto kapitola obsahuje zmeny vykonané v 2D klientovi oproti výsledku minuloročného tímu.

4.2.1 Odstránenie nepotrebných častí

- Odstránenie závislostí aplikácie na knižnici Spring
    - Odstránenie triedy ApplicationContextHolder
    - Prepísanie závislých častí kódu

4.2.2 Konfigurácia

- Vytvorená nová trieda Config
  - statická inicializácia
  - dáta vo formáte XML
  - prístup k hodnotám v štýle kľúč-hodnota

Ukážka zápisu v XML

```
<add key="service" value="http://147.175.159.184:8080/D9vinaServer/docs" />
```

Prístup v kóde:

```
Config.get(“service”)
```

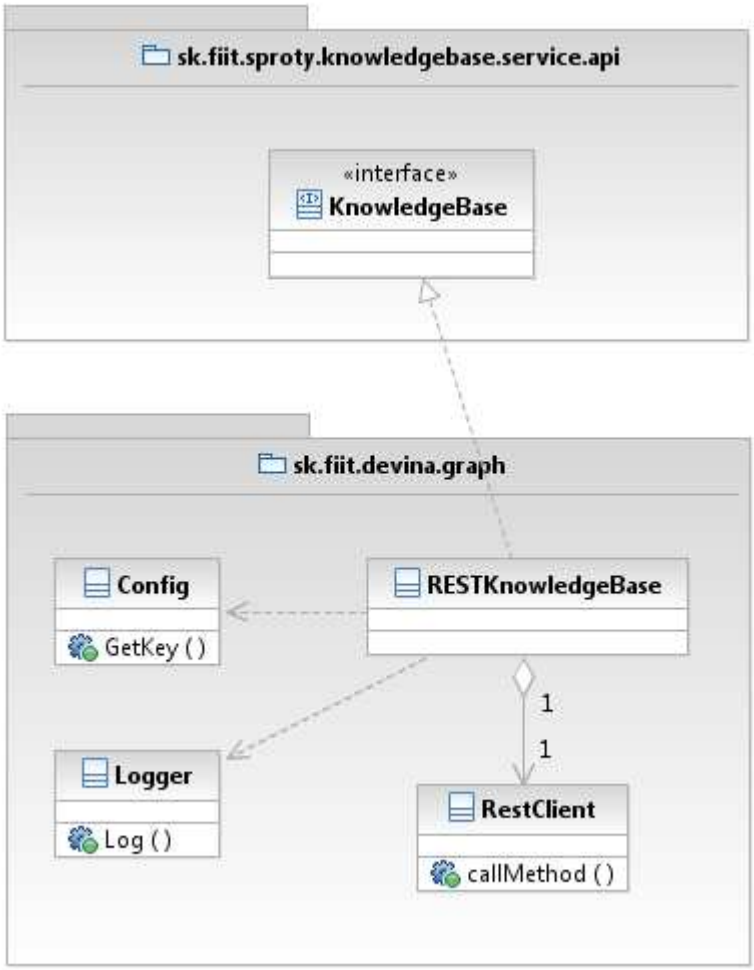
4.2.3 Logovanie

- Vytvorenie triedy Logger
  - statická inicializácia
  - nahradenie výpisov do konzoly
  - konfigurovateľná cez config.xml

4.2.4 Nová implementácia rozhrania KnowledgeBase

- Nová trieda RESTKnowledgeBase
  - implementuje pôvodné rozhranie KnowledgeBase
  - odstránenie technológie RMI
  - komunikácia prebieha na základe HTTP a architektúre REST

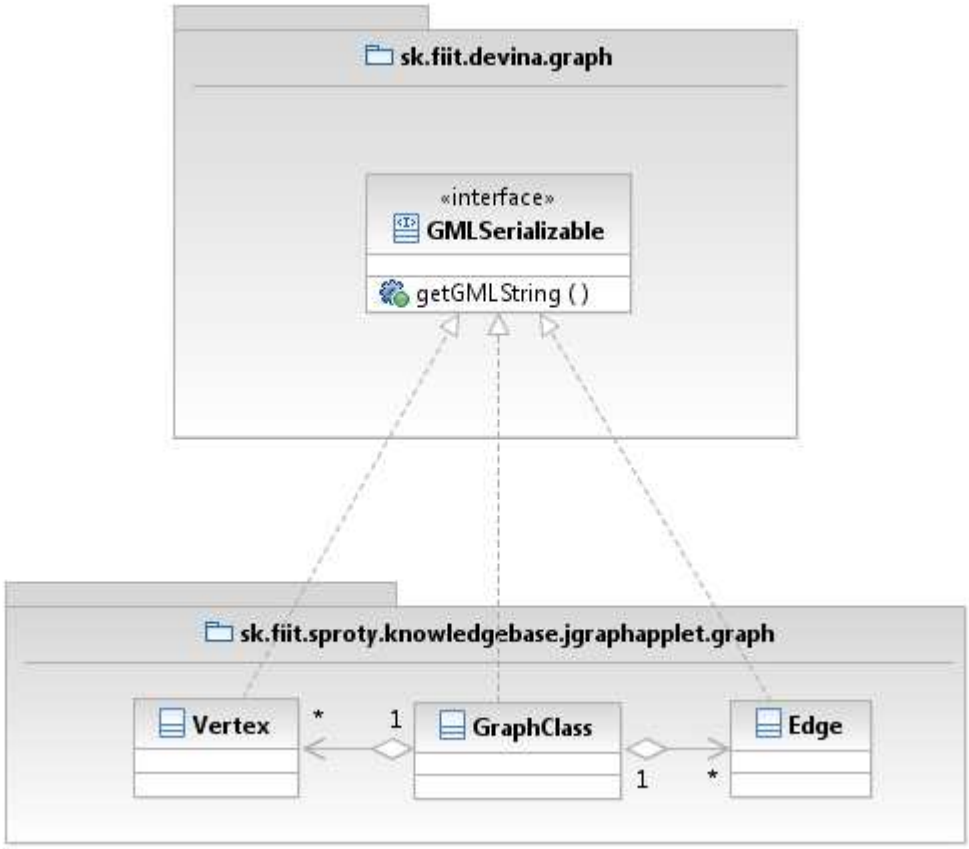
- Nová trieda RestClient
  - sprehľadňuje HTTP komunikáciu
  - volanie metód prebieha s využítim protokolu HTTP a XML s využitím serializačných mechanizmov Java platformy



Obr. 15. Nová implementácia rozhrania KnowledgeBase

4.2.5 GraphML serializácia grafov

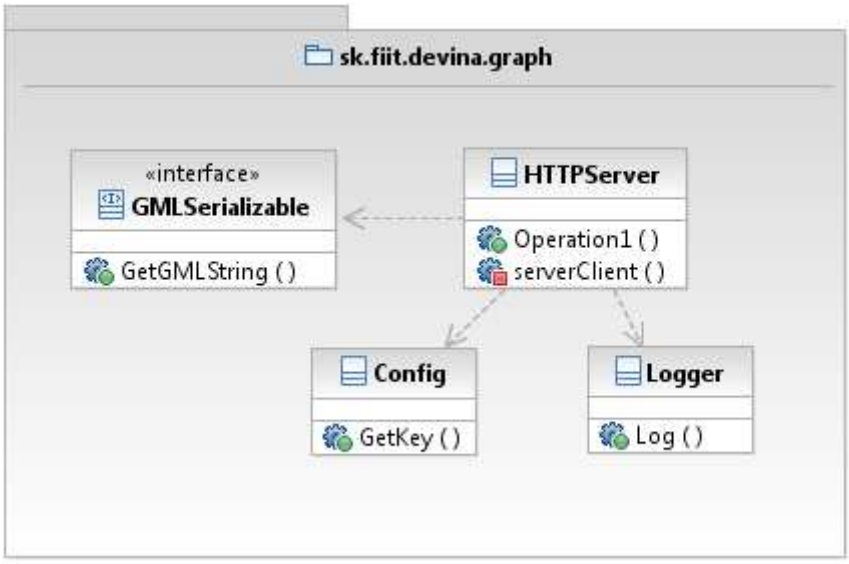
- Vytvorené nové rozhranie GMLSerializable:
  - doplnená implementácia pôvodných tried pre reprezentáciu grafu, vrcholov a hrán
- Serializáciu grafu do GraphML je možné vyvolať metódou rozhrania getGMLString() nad konkrétnou inštanciou grafu.



Obr. 16. GraphML serializácia grafov

4.2.6 Server

- Vytvorená nová trieda HTTPServer, ktorá sprístupňuje graf vo formáte GraphML prostredníctvom HTTP protokolu.
- Pri každej požiadavke zo strany HTTP klienta je GraphML vygenerované nanovo na základe aktuálneho grafu.



Obr. 17. Server

4.3 Server

V kapitole uvádzame zmeny serveru oproti serveru minuloročného tímu spolu s vlastnosťami a konfiguráciou serveru.

4.3.1 Zmeny oproti serveru minuloročného tímu

- Nami vytvorený server po tíme z minulého roka takmer nič nepreberá.
- Využité sú iba zdrojové kódy tried, ktoré reprezentujú tzv. DTO objekty (Data Transfer Object)
- Tieto triedy boli prepísané na komponenty Java Beans, z dôvodu, aby mohli byť plnohodnotne využívané triedami XMLEncoder a XMLDecoder, ktoré patria do balíčka java.beans. Tie sú využité na zakódovanie a dekódovanie DTO objektov do XML formy, ktoré sú použité pri komunikácii medzi klientom a serverom.

4.3.2 Vlastnosti serveru

- Server beží ako Java HTTP Servlet.
- Server beží na aplikačnom serveri Jetty, ktorý je celý napísaný v programovacom jazyku Java.

4.3.3 Konfigurácia serveru

- Konfigurácia je sústredená do súboru web.xml:
  - obsahuje názov triedy, ktorý predstavuje hlavný HTTP servlet (v našom prípade je to trieda MainServlet)
  - taktiež obsahuje vzor pre mapovanie daného servletu na konkrétne triedy URL (v našom prípade existujú dve triedy: /docs/* - trieda zaoberajúca sa prácou s dokumentmi, väzbami atď. a /user/* - trieda zaoberajúcou sa prácou s používateľom)
  - neobsahuje mapovanie konkrétnych URL na vybranú funkcionalitu podľa architektúry REST (to je súčasťou triedy MainServlet)

Štruktúra súboru web.xml:

```
<servlet>
    <servlet-name>názov servletu</servlet-name>
    <servlet-class>názov triedy, ktorá reprezentuje
servlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>názov servletu</servlet-name>
    <url-pattern>URL, na ktorú je namapovaný daný
servlet</url-pattern>
</servlet-mapping>
```

4.3.4 Komunikácia serveru s 2D klientom

- Komunikácia prebieha pomocou HTTP protokolu, výhradne pomocou metód POST
- Podľa architektúry REST sú konkrétne URL namapované na dané zdroje (napr. URL http://147.175.159.184:8080/D9vinaServer/docs/findDoc predstavuje funkcionalitu pre vyhľadanie konkrétnych dokumentov na serveri).
- Jednotlivé Java objekty prenášané pomocou metódy POST protokolu HTTP sú vo forme XML, ktoré sú vytvorené triedou XMLEncoder. Tieto objekty možno jednoducho po ich prijatí deserializovať pomocou triedy XMLDecoder.

Príklad štruktúry XML vytvoreného triedou XMLEncoder pre objekt typu DtoDocument:

```
<object
class="sk.fiit.sproty.knowledgebase.service.api.dto.DtoDocumen
t">
    <void property="addedBy">
      <long>4</long>
    </void>
    <void property="author">
      <string>Klensin</string>
    </void>
    <void property="dateWhenAdded">
      <object class="java.util.Date">
        <long>1290294000000</long>
     </object>
    </void>
  ...
```

4.3.5 Komunikácia serveru s databázou Postgre SQL

- Servlet využíva konfiguračný súbor XML na napojenie sa na databázu Postgre SQL.
- Servlet taktiež využíva mapovacie súbory XML, vytvorené pre konkrétne DTO objekty.
- Servlet okrem toho využíva metódy napísané v programovacom jazyku Java, ktoré spolupracujú so spomínanými XML súbormi a vracajú konkrétne DTO objekty. Tieto objekty sú následne posielané klientskej časti aplikácie.

4.4 3D klient

V klientovi (C++) boli vykonané nasledujúce zmeny:

- pridanie funkcionality na získanie grafu uloženého vo formáte GraphML zo vzdialeného servera prostredníctvom http protokolu
- upravenie vytvárania návestí pri uzloch a hranách zobrazovaného grafu

4.4.1 Získanie grafu zo vzdialeného servera

Implementovanie tejto funkcionality bolo nevyhnuté v dôsledku komunikácie klienta so serverom. Momentálne prostredníctvom nej komunikuje 3D klient s 2D klientom, avšak v budúcnosti sa plánuje, že 3D klient bude prostredníctvom tejto funkcionality komunikovať priamo so serverom.

Táto časť je implementovaná preťažením metódy **parseGraph()**, ktorá ako parameter obdrží URL adresu vzdialeného servera. Táto adresa je uložená v konfiguračnom súbore pod

položkou *GraphMLParser.serverUrl*. Následne je vytvorené spojenie zo serverom a z odpovede, ktorá je odoslaná zo servera je vytvorený graf.

4.4.2 Upravenie vytvárania návestí vrcholov a hrán

Návestia zobrazované pri jednotlivých vrcholoch a hranách sa získavajú zo zdrojového GraphML súboru. Štruktúra toho súboru je nasledovná.

```
<graphml>
     <graph id="G" edgedefault="directed">
          <node id="n0">
               <data key=”type”>TYP_DOKUMENTU</data>
               <data>Nazov dokumentu</data>
          </node>
          <node id="n1">
               <data key=”type”>TYP_DOKUMENTU</data>
               <data>Nazov dokumentu</data>
          </node>
          <edge source="n0" target="n1">
               <data key=”relation”>TYP_VAZBY</data>
               <data key="KW">4</data>
               <data key="Auth">2</data>
          </edge>
     </graph>
</graphml>
```

Návestie pri vrchole je potom tvorené názvom dokumentu a návestie pri hranách vytvorené nasledovne: **KW : 4 | Auth : 2**

4.5 Algoritmy na extrakciu kľúčových slov a získavanie metadát z dokumentov

Táto kapitola obsahuje popis metódy extrakcie kľúčových slov z dokumentov a informácií z metadát.

4.5.1 Pridaná funkcionalita

Minuloročný tím, ktorý pracoval na získavaní znalostí z dokumentov, neimplementoval extrakciu kľúčových slov priamo z dokumentu. Tím navrhol získavanie kľúčových slov a iných potrebných dát iba z metadát.

- Pridali sme reálnu extrakciu kľúčových slov z dokumentov.
- Implementovali sme získavanie dát (napr. autor) z metadát dokumentov.

- Podpora MS Word Document, PDF, plain text.
- Podpora slovenského jazyka – stop slová, lematizácia.

4.5.2 Extrakcia kľúčových slov z dokumentov

- Spracovanie DOC, PDF, TXT súborov
- Konverzia DOC, PDF do TXT
    - V rámci spracovania sa volá externá aplikácia ConvertDoc spušťaná s parametrami (vstup, výstup, možnosti).
- Problémy s kódovaním PDF, čiastočne vyriešené.
- Odstránenie stop slov a čísel
    - vlastný zoznam slovenských stop slov, možnosť použiť anglické stop slová.
    - odstraňovanie stop slov - dva krát.
        - 1. pred lematizáciou, aby sme zrýchlili proces lematizácie (aby neboli vyhľadávané základné tvary stop slov)
        - 2. po lematizácii (odstránenie stop slov prevedených do základného tvaru)
    - odstránenie čísel, pretože ponechanie čísel - čísla sú nevhodné (číslované zoznamy...)
- Lematizácia
    - zmena slov do základného (slovníkového tvaru) - potreba slovníka - súbor form2lemma.cdb (pochádza z JÚĽŠ SAV). práca s týmto databázovým súborom – využitie manuálu k CDB a konkrétne použitie knižnice strangegizmo.cdb.
- Početnosti slov – TF (term frequency)
- Výber kľúčových slov
    - vybraté slová, ktorých početnosť v dokumente nie je menšia ako konštanta (0,5) * početnosť najviac vyskytujúceho sa slova. Nastaviteľná metóda.

4.5.3 Získavanie informácií z metadát

- Využitie knižnice ICEpdf
    - extrakcia metadát z dokumentu pre následné použitie
    - potrebné informácie – autor, prípadne keywords

4.6 Databáza Postgre SQL

Kapitola opisuje špecifiká vytvorenia, naplnenia a práce s Postgre SQL databázou.

4.6.1 Vytvorenie novej databázy

- Vytvorenie databázy na základe analýzy
- Vzťahy m:n nahradené väzobnými tabuľkami
- Dedenie a špecializácia využité pri vytváraní tabuliek, ktoré sa týkajú väzieb

4.6.2 Naplnenie databázy

Väzby medzi dokumentmi pre naplnenie databázy prototypu zobrazené v Prílohe A zachytáva tabuľka č. 17.

Tab. 17. Väzby medzi dokumentmi

ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14
14			x					x			x			
13		x			x					x				
12			x	x		x	x		x					
11				x		x		x						
10	x		x		x		x							
9		x				x								
8		x		x	x									
7	x		x	x		x								
6	x		x		x									
5		x		x										
4	x		x											
3		x												
2	x													
1														

4.6.3 iBatis

Nahradenie zbytočne robustného nástroja Hibernate iBatisom.

4.6.4 Konfigurácia

Prostredníctvom konfiguračného XML nastavujeme umiestnenie databázy, prihlasovacie údaje a mapovacie XML, s ktorými ďalej pracujeme.
Štruktúra xml je nasledovná:

```
<sqlMapConfig>
      <transactionManager>
          <!--Obsahuje údaje o pripojení (meno, heslo, adresa,
meno databázy)-->
      </transactionManager>
          <!--Použité mapovacie xml dokumenty-->
     <sqlMap resource="Contact.xml"/>
     <sqlMap resource="DtoDocument.xml"/>
     <sqlMap resource="DtoBinding.xml"/>
</sqlMapConfig>
```

4.6.5 Mapovacie súbory

Údaje z databázy získavame prostredníctvom funkcií definovaných v mapovacích súboroch. Pre zvýšenie prehľadnosti sú tieto funkcie rozdelené do súborov podľa objektov, ku ktorým sa viažu.
Používané objekty:

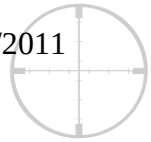
```
–DtoBinding
–DtoAutomaticBinding
-DtoDocument
```

Pri mapovaní údajov z databázy na objekty triedy DtoBinding sme vytvorili triedu DbDtoBinding, ktorá je presným obrazom dát v databáze a umožňuje jednoduchšie mapovanie.

4.6.6 Príklad funkcie v DtoDocument.xml

```
<select id="getByKeyWords" resultClass="DtoDocument"
parameterClass="list">
```

- `Id` názov ktorým je funkcia select volaná z Java prostredia



- `resultClass` meno triedy objektu, na ktorý sa majú dáta namapovať
- `parameterClass` druh vstupných parametrov (map, list, object)

Pri použití vstupného parametru list môžeme nad ním iterovať a vytvoriť požadovaný select

```
<iterate open="(" close=")" conjunction="OR">
   keyword LIKE '%$[]$%'
 </iterate>
```

- `open="(" close=")"` určuje ohraničenie príkazu vutvoreného iteráciou
- `conjunction="OR"` určuje spojenie medzi častami príkazu
- `keyword LIKE '%$[]$%'` určuje, že porovnávané pole zo vstupného listu bude keyword, na porovnanie sa použije funkcia LIKE a daný reťazec sa bude hľadať kdekoľvek v slove

4.7 Zhodnotenie výsledkov prototypovania

Cieľom prototypovania bolo vytvorenie prototypov všetkých súčastí systému. Podarilo sa nám implementovať analýzu dokumentov, vizualizáciu v 2D a 3D prostredí, editáciu a manipuláciu s grafmi.

Tiež sme v rámci implementácie zmenili predchádzajúce technológie a to:

- Hibernate na iBatis
  - Nahradenie zbytočne robustného nástroja so zachovaním potrebnej funkcionality
- RMI na REST
  - Umožnenie prepojenia 3D klienta (C++) na server
  - Univerzálnosť architektúry REST
- JBoss na Jetty
  - Využitie plnej funkcionality Jetty namiesto čiastočnej funkcionality serveru JBoss.

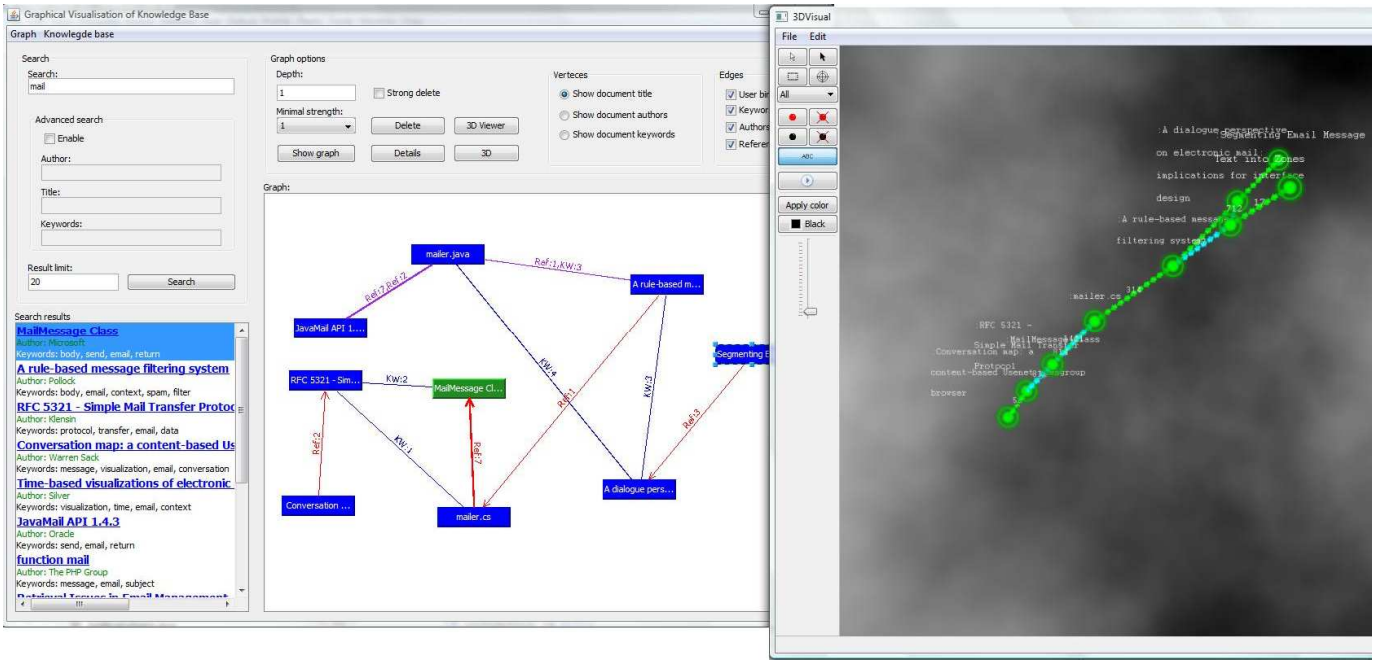
Čo sa týka klientov, vytvorili sme prepojenie 2D a 3D klienta prostredníctvom http, zobrazenie editovaného grafu z 2D do 3D prostredníctvom GML, jednoduchú editácia grafu v 2D, možnosti grafu - collapse, expand, delete, zmenu rozloženia uzlov.

Server predstavuje Java HTTP servlet, beží na aplikačnom serveri Jetty, komunikuje s 2D klientom na báze architektúry REST, s databázou komunikuje prostredníctvom frameworku iBatis.

PostgreSQL databáza využíva dedenie pri vytváraní tabuliek. Urobili sme zmenu v dátovom modeli - väzobnú tabuľka pre kľúčové slová. iBatis nám umožňuje mapovanie dát na objekty - XML mapovacie súbory, konfiguráciu pripojenia prostredníctvom XML.

Tiež sme implementovali reálne získavanie kľúčových slov z dokumentov a informácií z metadát dokumentov.

Na obrázku č. 18 vidíme ukážku používateľských rozhraní oboch klientov.



Obr. 18. Screenshot prototypov 2D a 3D klienta

5 Produkt

V tejto kapitole uvádzame dokumentáciu k finálnemu produktu vrátane zmien voči hrubému návrhu.

V prvej časti sa nachádza prehľad zapracovaných nedostatkov špecifikácie a hrubého návrhu, kde sa vyjadrujeme aj k hodnoteniu a pripomienkam k práci za prvý semester. V ďalšej časti sa nachádza zmenený dátový model, ohraničenia, zmeny špecifikácie a naše priority riešenia. Posledná, najrozsiahlejšia časť, obsahuje popis jednotlivých častí systému a ich implementácie, spolu s uvedením zmien voči prototypu.

5.1 Zapracovanie nedostatkov špecifikácie a hrubého návrhu

- Zmenili sme pôvodne používaný Hibernate na menej robustné riešenie. Rozhodli sme sa pre iBatis, ktorý nám prakticky umožňuje mapovanie dát z databázy na objekty. Vďaka jeho schopnosti mapovať dáta z databázy podľa príznaku v tabuľke na rôzne typy objektov sa nám podarilo zoptimalizovať databázovú štruktúru.
- REST rozhranie bolo v tomto semestri zmenené, hlavne na základe pripomienky o nepochopení REST rozhrania. Pri zmene REST rozhrania sme sa inšpirovali kapitolou REST and ROA Best Practices v knižke RESTful Web Services od L. Richardsona a S. Rubyho.
- Kód na serverovej časti bol kompletne prerobený, pričom bola zachovaná iba logika jednotlivých service-ov. Na serveri sme taktiež začali používať framework Spring, čo nám značne uľahčilo prácu a pomohlo vytvoriť viacej čitateľný kód.
- Chyby pri využívaní repozitára vznikli z dôvodu chýbajúcich pravidiel pre prácu s týmto úložiskom.
- Čo sa týka vyčítaného neupratáného kódu – kód 2D klienta, ktorý bol doplnený bol a je v samostatných packagoch, jasne oddelených od kódov predošlého tímu, použitý objektový model.
- Komunikácia prostredníctvom DTO objektov je bežne používaný prístup pre vzdialené volanie metód, tento spôsob zvolil minuloročný tím.
- Architektúra peer-to-peer bola zvolená vzhľadom na požiadavku vedúceho, aby bol prechod medzi 2D a 3D klientom čo najjednoduchší.

- Vlastný algoritmus extrakcie kľúčových slov pomocou TF sme implementovali preto, lebo počas odkazovania na iné riešenie sme ho mali už navrhnutý a začatý, jeho implementácia nebola náročná a takýto spôsob extrakcie kľúčových slov je nami jednoducho modifikovateľný, čo sme neskôr aj využívali.
- Odstraňovanie stop slov na dva krát sa vykonáva z nasledovného dôvodu. Prvé odstránenie stop slov odstráni priemerne 30% všetkých slov dokumentu. V prípade slovenského jazyka dokumentu následne prebehne lematizácia. Tá spôsobí, že niektoré slová (napr. slovo „ktorému“) sa prevedú do základného tvaru (na slovo „ktorý“). V zozname stop slov sa slová väčšinou nachádzajú v základnom tvare. Preto treba vykonať ďalšie odstránenie stop slov, ktoré odstráni zostávajúce stop slová dokumentu. Experimentálne bolo overené, že prvé odstránenie stop slov je potrebné, pretože keby sa nevykonávalo, lematizácia celého textu slovníkovou metódou by trvala dlhšie, ako keď vykonáme odstránenie stop slov a zostávajúci text lematizujeme.

5.1.1 Známe nedostatky prototypu

- V databáze existujú tabuľky, ktoré sú využité len čiastočne. Ide o tabuľku týkajúcu sa histórie zobrazovania dokumentov. Zatiaľ ju len napĺňame.
- Neúplná funkcionalita na strane servera a z toho vyplývajúca neúplnosť 2D klienta.
- Pri pridávaní nového dokumentu spolu s upladom fyzického súboru na server beží táto operácia až príliš dlho (pri niektorých dokumentoch aj 20-30 sekúnd). Preto by bolo vhodné zoptimalizovať túto službu.

5.1.2 Iné zapracované nedostatky prototypu

- Na komunikáciu medzi Java serverom a databázou sme použili framework iBatis. Obnášalo to vytvorenie konfiguračných a mapovacích *.xml súborov a vytvorenie funkcií pre potreby servera v Jave, ktoré tieto mapovacie súbory používajú.
- Na pripomienky externého hodnotiteľa sme zapracovali na serveri vystavenie služieb cez REST rozhranie, použitie web servera Jetty a použitie frameworku Spring na vývoj.
- Prepracovanie REST rozhrania, dokončenie a pretestovanie funkcionality.

5.2 Zmeny v návrhu systému

Zmeny v návrhu systému sa udiali hlavne v databázovej časti, preto uvádzame aktuálny logický model, kde boli vykonané zmeny. K nemu prikladáme príslušný fyzický model údajov.

5.2.1 Zmeny v architektúre

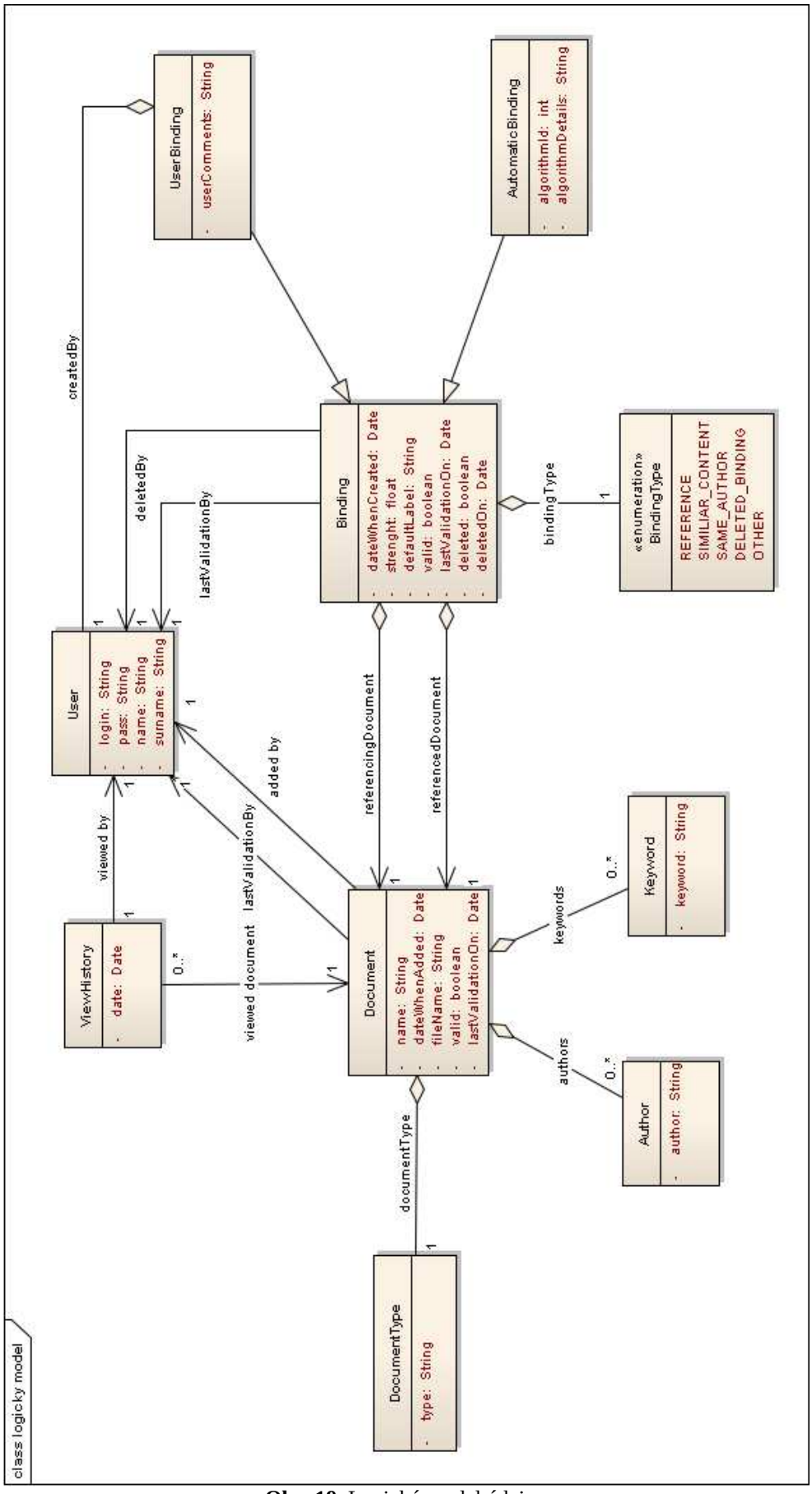
Od špecifikovania riešenia a od prototypovania neboli vykonané žiadne zmeny v architektúre systému. Návrh architektúry a príslušné modely sú uvedené v kapitole Špecifikácia riešenia.

5.2.2 Logický model údajov

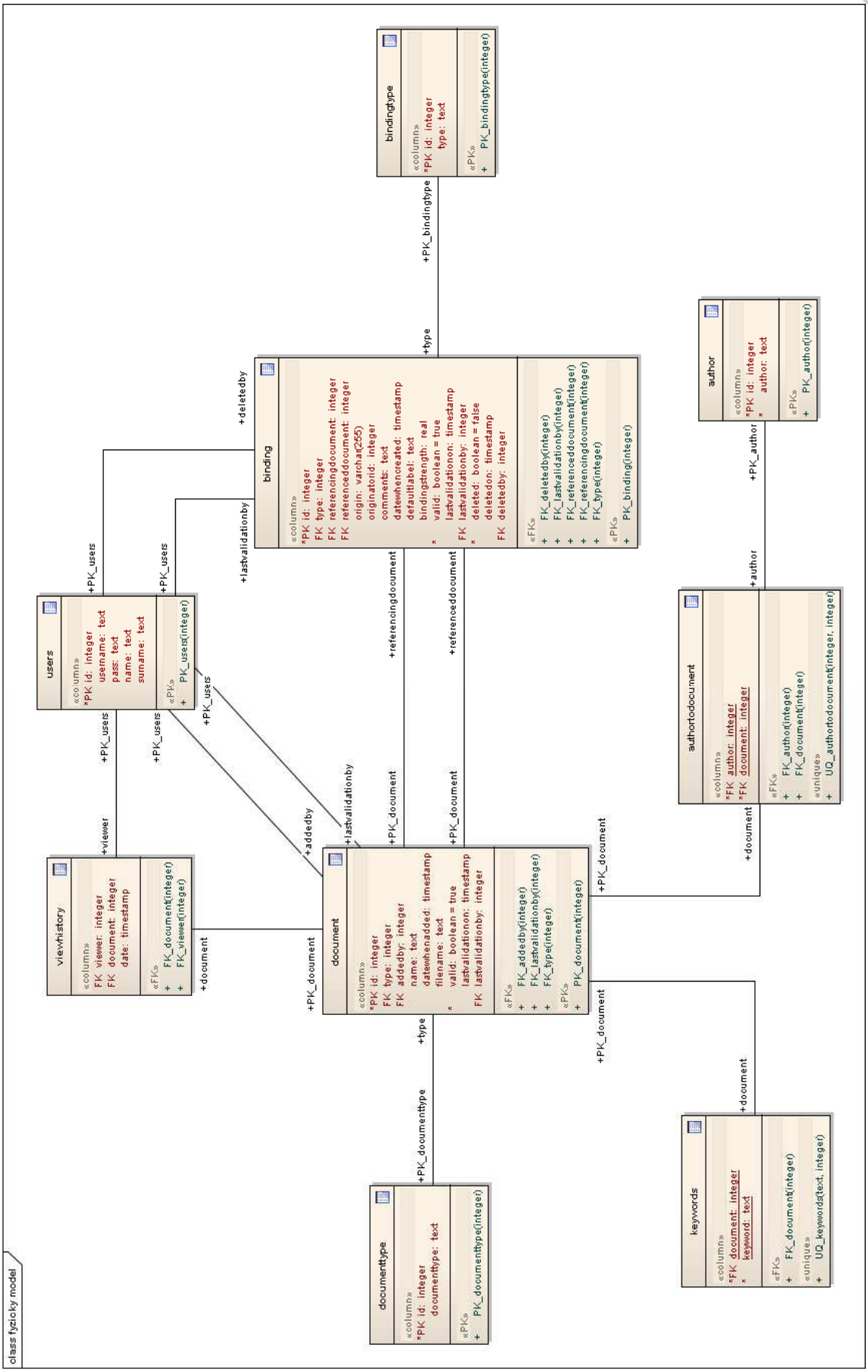
Na obrázku č. 19 je zobrazený pozmenený logický model údajov systému. Triedy sme už opísali v kapitole Špecifikácia údajov v systéme.

5.2.3 Fyzický model údajov

Na obrázku č. 20 sa nachádza fyzický model údajov systému.



Obr. 19. Logický model údajov



Obr. 20. Fyzický model údajov

5.3 Ohraničenia, zmeny špecifikácie, priority riešenia

V súvislosti s prototypom neboli vykonané žiadne zmeny v implementačnom jazyku alebo prostredí. Implementačné jazyky a prostredia sme si stanovili už pri analýze problematiky.

Jedno z ohraničení nášho systému je, že umožňuje pracovať len registrovaným používateľom. Nový používateľ sa zatiaľ nevie sám zaregistrovať, resp. prihlásiť bez registrácie.

Rýchlosť nahrávania dokumentov je obmedzená vzhľadom na rýchlosť bežného pripojenia a rýchlosti spracovania serverom. Ohraničením v spracovávaní dokumentov je podpora zatiaľ len 3 formátov dokumentov a podpora 3 jazykov dokumentov.

Hlavným ohraničením 3D klienta je jeho závislosť na knižnici OpenSceneGraph, ktorá sprostredkuje práve vizualizáciu grafu v 3D priestore. Spolu s touto knižnicou aplikácia využíva aj ďalšie dve pomocné (zlib a libpng) a taktiež Qt knižnice, ktoré zastrešujú nielen používateľské rozhranie, ale ponúkajú aj mnohé pokročilejšie štruktúry a algoritmy, ktoré v základných knižniciach jazyka chýbajú.

V databázovej vrstve je potrebné používať knižnicu, ktorá poskytuje Java API pre použitie frameworku iBatis.

5.3.1 Zmeny v prioritách riešenia

V rámci 3D klienta sa vypustila implementácia priamej komunikácie 3D klienta so serverom. Taktiež v dôsledku nedostatku času sa vypustila implementácie pokročilej funkcionality 2D klienta do 3D klienta, medzi ktorú patria napríklad aj funkcie collapse a expand.
Upustilo sa aj od funkcie, ktorá by na základe používateľom zadaných kritérií vytvorila nový dokument pozostávajúci z častí existujúcich dokumentov v databáze. Implementácia tejto funkcie však bola vždy skôr víziou do budúcnosti, akým smerom by sa mal projekt uberať, než reálnym cieľom počas nášho pôsobenia na tomto projekte.
Okrem týchto záležitostí sa upustilo od plánovanej zmeny architektúry. Tiež sme neriešili prácu s používateľmi, čiže ich ukladanie do databázy a riešenie bezpečnosti.

Na druhej strane, pridali sme autentifikáciu používateľov. Uvažovalo sa, že systém bude fungovať bez autentifikácie používateľov, ak by bolo náročné túto autentifikáciu zabezpečiť. Nakoniec bola táto funkcionalita do systému zapracovaná, teda prihlasovanie a odhlasovanie používateľov bolo implementované aj na serveri.

Neplánovane pridaná bola vyššia interaktivita s používateľom pri nahrávaní a indexovaní dokumentov, aby bol umožnený vklad používateľa do tohto procesu.
Taktiež sme neuvažovali aj o možnosti spracovania rôznych jazykov dokumentov, čo sa pri implementácii stali problémom, vzhľadom na spracovávanie textu, preto sme implementovali spracovanie nami vybraných 3 jazykov. V spracovaní dokumentov sme spočiatku neuvažovali viacerých autorov dokumentu, čo sme však prehodnotili a implementovali rozpoznanie možných viacerých autorov z metadát dokumentu.

Celkovou prioritou pri databázovej časti bolo ucelenie modelu, aby sa prípadné zmeny neprenášali nižšie na server alebo klientov. Veľkú prioritu v databázovej časti malo vyhľadávanie dokumentov podľa rôznych kritérií. Implementácii prihlasovania používateľa sme dali nižšiu prioritu.
Hlavnou prioritou 3D klienta bolo implementovať online komunikáciu so vzdialeným serverom a následne korektne zobraziť graf v 3D priestore. I keď momentálne 3D klient komunikuje len s 2D klientom, spôsob komunikácie bol navrhnutý a implementovaný tak, aby bolo možné v ďalšom vývoji 2D klienta úplne vypustiť a komunikovať so serverom priamo (využitie HTTP protokolu a REST rozhrania). Korektné zobrazovanie grafu predstavuje správne zobrazenie vrcholov a hrán a ich prislúchajúcich popisov (labelov). Taktiež farebné odlíšenie typu dokumentov a typu väzieb a zobrazenie sily väzieb prostredníctvom hrúbky hrany.
Najväčšiu prioritu v časti 2D klient bolo zabezpečenie základnej funkčnosti a pri indexácii dokumentov správny výber kľúčových slov dokumentu, správna extrakcia metadát dokumentov a správne nájdenie referencií medzi dokumentami.
Pri serverovej časti sme sa zamerali hlavne na implementáciu metód v presnom poradí podľa ich dôležitosti pre klienta. Taktiež sme sa snažili otestovať všetku novú funkcionalitu ešte pred nasadením na server.

5.4 Opis implementácie jednotlivých modulov, optimalizácia, doplnenia oproti návrhu

Táto kapitola obsahuje popis jednotlivých častí produktu, konkrétne doplnenia oproti návrhu a prototypu. Systém a jeho implementáciu sme rozdelili do nasledovných častí: 2D klient, server, 3D klient, indexácia, databáza. V prípade optimalizácie je uvedený opis optimalizačných zmien.

5.4.1 2D klient

Do 2D klienta bola pridaná funkcionalita prihlasovania používateľov a vytvorené nové okná – prihlasovanie, upload dokumentu.

Prihlasovanie používateľov

- Aplikácia pri spustení vyžiada prihlasovacie údaje.
- V konfiguračnom súbore je možné nastaviť vopred vyplnené hodnoty.
- Trieda LoginManager predstavuje jednoduchú implementáciu uchovávania prihlasovacích údajov, ktoré sa odosielané s každou požiadavkou na server.

Nové okná

- Bolo vytvorené nové okno pre vkladanie prihlasovacích údajov.
- Bolo vytvorené nové okno pre upload dokumentu, aby mal používateľ možnosť ručne editovať metadáta dokumentu.

Všetky triedy, ktoré boli pridané do zdrojových kódov sa nachádzajú v balíku sk.fiit.devina.graph a sk.fiit.devina.graph.forms.

5.4.2 Server

Serverová časť aplikácie, ako sme už uviedli v dokumentácii k prototypu, nepreberá od minuloročného tímu nič okrem zdrojových kódov tried, ktoré reprezentujú tzv. DTO (Data Transfer Object).

Server je založený ako Java HTTP Servlet, čo predstavuje komponent, ktorý umožňuje zachytiť klientske požiadavky prichádzajúce prostredníctvom HTTP protokolu, následne ich

spracovať a vrátiť adekvátnu odpoveď. Servlety musia bežať vo webovom serveri, ktorý mapuje jednotlivé klientske requesty na odpovedajúce triedy servletov. My sme ako webový server (a zároveň servlet kontajner) použili server Jetty, ktorý je celý napísaný v programovacom jazyku Java, pretože vieme efektívne využiť jeho potenciál, na rozdiel od iných, príliš robustných serverov ako je napr. JBoss.

Serverovú časť aplikácie sme založili na populárnom Java frameworku Spring, ktorý výrazne uľahčuje prácu programátora pri tvorbe Java enterprise aplikácií. Potenciál tohto frameworku sme využili hlavne na vystavenie zdrojov servera cez REST (Representational State Transfer) rozhranie, efektívne spracovanie URL daného zdroju a získanie parametrov requestu. Spring taktiež umožnuje ľahkú kooperáciu s objektovo relačnými mapovačmi, prístupom k databáze, ale aj zjednodušuje prácu programátorom pri riešení úloh týkajúcich sa napríklad transakcií alebo bezpečnosti. Z toho dôvodu môže tento framework výrazne prispieť na vývoji toho čo už bolo nad rámec nášho riešenia projektu a bude pravdepodobne riešené v budúcnosti.

Ako už bolo spomenuté, server vystavuje jednotlivé služby prostredníctvom REST rozhrania, pričom jednotlivé parametre a návratové hodnoty posielané medzi klientom a serverom sú vo formáte XML. Štruktúru týchto XML súborov sme si špecifikovali vzhľadom na DTO objekty tohto systému.

V rámci serveru taktiež bežia algoritmy na indexáciu dokumentov. Pri pridávaní alebo modifikácii dokumentu s fyzickým súborom (t.j. nie virtuálneho dokumentu), sú spustené algoritmy na extrakciu informácií z dokumentu, nielen z jeho fyzickej reprezentácie v podobe súboru, ale aj z jeho metadát, ak sa jedná o súbory typu DOC alebo PDF.

Pri pridávaní alebo modifikácii virtuálneho alebo obyčajného dokumentu sú taktiež spustené algoritmy na hľadanie väzieb medzi týmto dokumentom a ostatnými dokumentmi v databáze. Tieto algoritmy bežia v samostatných vláknach, tak aby nepredstavovali blokujúce operácie v rámci volaní príslušných služieb. Vyhľadávanie väzieb sa teda vykonáva na pozadí, pričom sú poskytnuté algoritmy na vyhľadávanie podľa kľúčových slov, rovnakých autorov, referencií a podobného obsahu dokumentu.

Server taktiež analyzuje headre jednotlivých requestov, podľa ktorých vykonáva príslušné operácie (napr. ak klient pošle header Accept encoding: gzip, tak server odošle response v takto skomprimovanej podobe). Server taktiež rieši posielanie prislúchajúcich response kódov podľa toho či sa operácia skončila správne (kódy typu 2XX) alebo s chybou (kódy skupiny 4XX a 5XX).

Vyvolané výnimky sa samozrejme na serveri logujú do súboru a taktiež sa dočasne v rámci prototypu posielajú aj klientom obalené v XML, z toho dôvodu, aby boli chyby serveru alebo klienta pre nás ľahšie opraviteľné.

Optimalizácia pozostávala z implementácie GZIP kompresie pri uploade a downloade súboru. Okrem toho sa hľadanie väzieb pri pridaní nového dokumentu deje v samostatnom vlákne, tak aby sa nejednalo o blokujúcu operáciu.

5.4.3 3D klient

Aplikácia bola prevzatá od minuloročného tímu Starwalkerov a aj napriek veľkému úsiliu sa zdrojové kódy podarilo spojazdniť až v druhej polovici prvého semestra. Príčinou tohto zdržania bol najmä fakt, že s týmto prostredím (OpenSceneGraph a Qt) sme doteraz nemali žiadne skúsenosti a taktiež to, že knižnice, ktoré aplikácia vyžaduje, bolo treba zvlášť skompilovať a buildnúť.

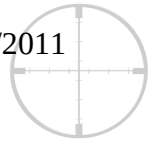
Aplikácia predstavovala dobrý základ pre splnenie dlhodobých cieľov tohto projektu, medzi ktorými figuruje zobrazenie a prezeranie grafu v 3D priestore. Pre jej integráciu do nášho projektu však boli nevyhnutné nasledujúce zmeny:

- implementovať komunikáciu so vzdialeným serverom
- prispôsobiť zobrazovanie grafu oblasti vizualizácie znalostí
- odstrániť využívanie databázy

Komunikácia so vzdialeným serverom

Prevzatá aplikácia bola navrhnutá ako stand-alone aplikácia, ktorá ako vstup využíva súbory vo formáte GraphML resp. grafy ukladá a načítava z databázy, ktorej štruktúra bola navrhnutá práve pre tento účel.

Implementovanie funkcionality, ktorá zabezpečí komunikáciu so vzdialeným serverom, bolo preto nevyhnuté v dôsledku aktuálnej klient-server architektúry projektu. Po dôslednom zvážení všetkých kritérií sa rozhodlo, aby bola komunikácia realizovaná prostredníctvom HTTP protokolu a vo formáte GraphML. Následne bola implementovaná a momentálne prostredníctvom nej komunikuje 3D klient s 2D klientom, avšak v budúcnosti sa uvažuje o úplnom vypustení 2D klienta a teda 3D klient bude prostredníctvom tejto funkcionality komunikovať priamo so serverom.



Táto časť je implementovaná preťažením metódy parseGraph(), ktorá ako parameter obdrží URL adresu vzdialeného servera. Táto adresa je uložená v konfiguračnom súbore pod položkou *GraphMLParser.serverUrl*. Následne je vytvorené spojenie zo serverom a z odpovede, ktorá je odoslaná zo servera je vytvorený graf.

Prispôsobenie zobrazovania grafu oblasti vizualizácie znalostí

Oblasť zobrazovania grafu vyhovovala našim požiadavkám až pár detailov, ktoré bolo treba zapracovať. Konkrétne sa jednalo o nasledovné veci:

- zobrazenie sily väzieb medzi dokumentmi prostredníctvom hrúbky hrán
- upravenie zobrazovania návestí jednotlivých vrcholov a hrán

Pre zobrazenie sily väzieb prostredníctvom bol do zdrojového GraphML súboru pridaný atribút *strength*, ktorého hodnota sa pohybuje v intervale <0;10>. Návestia zobrazované pri jednotlivých vrcholoch a hranách sa získavajú z elementov *data*. Ak je týchto elementov viac, návestie sa vytvorí zreťazením jednotlivých hodnôt týchto elementov s použitím oddeľovača „|". Štruktúra GraphML súboru je uvedená nižšie pričom návestie pri vrchole je potom tvorené názvom dokumentu a návestie pri hranách vytvorené nasledovne:
KW : 4 | Auth : 2

```
<graphml>
     <graph id="G" edgedefault="directed">
          <node id="n0">
               <data key=”type”>TYP_DOKUMENTU</data>
               <data>Nazov dokumentu</data>
          </node>
          <node id="n1">
               <data key=”type”>TYP_DOKUMENTU</data>
               <data>Nazov dokumentu</data>
          </node>
          <edge source="n0" target="n1" strength=”5”>
               <data key=”relation”>TYP_VAZBY</data>
               <data key="KW">4</data>
               <data key="Auth">2</data>
          </edge>
     </graph>
</graphml>
```

Odstránenie použitia databázy

Prevzatá aplikácia využíva pre prácu s grafmi vlastnú databázu. Toto je celkom logické, keďže sa jedná o stand-alone aplikáciu. V dôsledku zmeny architektúry na klient-server

využívanie tejto databázy už nie potrebné a jej ponechanie by zbytočne spomaľovalo aplikáciu. Odstránenie jej používania zo zdrojových kódov by však bolo zbytočne namáhavé a taktiež by sa mohlo stať, že v budúcnosti sa databáza na strane klienta predsa len využije. Implementované riešenie teda zahŕňa jednoduchú aktiváciu a deaktiváciu používania databázy prostredníctvom konfiguračného súboru a položky *Model.DB.active*, pričom sa jedná o klasickú bool hodnotu (true – false).

Priestor pre pokračovanie

Keďže v rámci nášho projektu nemal 3D klient hlavnú prioritu a jeho spojazdnenie trvalo viac ako sme očakávali, bola implementovaná len základná funkcionalita. Priestor pre pokračovanie teda ostáva otvorený, pričom ďalšie kroky sa logicky odvíjajú od zámeru vypustiť 2D klienta z architektúry. To znamená implementovať komunikáciu 3D klienta priamo so serverom a taktiež doplniť pokročilé funkcie, ktorými 2D klient disponuje ako napr. collapse, exapand, virtuálne dokumenty atď.

5.4.4 Indexácia

Systém podporuje spracovanie 3 typov fyzických dokumentov: DOC, PDF, TXT. Ak je to možné, z metadát dokumentov sa extrahujú autori, názov dokumentu, dátum poslednej zmeny, kľúčové slová. Okrem toho sa z textu dokumentu extrahujú kľúčové slová na základe spracovania dokumentu opísaného v dokumentácii k prototypu. V pridávanom dokumente sa vyhľadávajú aj referencie na iné dokumenty v systéme.

V doméne indexácia boli od prototypovania vykonané nasledovné zmeny a pridaná ďalšia funkcionalita:

- Vypustili sme používanie aplikácie ConvertDoc, pretože sa nám podarilo získať iba 30-dňovú trial verziu a plnohodnotne sme ju nahradili knižnicami IcePDF a Apache Tika.
- Pridali sme získavanie metadát aj z dokumentov MS Word (kompletné pre: TXT, DOC, PDF)
- Pridali sme získavanie dátumu a názvu dokumentu (Title) z metadát.
- Pridali sme rozpoznanie rôznych jazykov (slovenský, anglický, nemecký) dokumentu a následné spracovanie dokumentu v tomto jazyku. Toto je potrebné kvôli odstraňovaniu stop slov a kvôli lematizácii v prípade slovenského jazyka.

- Pre spomenuté 2 nové jazyky (anglický, nemecký) sme doplnili odstraňovanie príslušných stop slov.
- Pridali sme zjednotenie získaných kľúčových slov z tela dokumentu a z metadát (väčšiu váhu majú kľúčové slová z metadát, lebo boli pravdepodobne ručne zadávané).
- Pridali sme rozpoznanie viacerých autorov z poľa Author v metadátach dokumentu - kvôli možnosti viacerých autorov jedného dokumentu.
- Pridali sme funkciu na hľadanie referencií medzi dokumentmi (z odkazov a zdrojov) a následné vytváranie väzieb
- Vytvorili sme obmedzenie pre kľúčové slová - min. 3 znaky a max. 10 kľúčových slov pre dokument. Ak je 10 kľúčových slov prekročených, všetky nájdené kľúčové slová s váhou rovnakou ako 10. slovo sa medzi kľúčové slová dokumentu nezapočítajú.
- Optimalizovali sme spracovanie dokumentov – spojili sme funkcie removeStopWords a removeNumbers - kvôli pôvodnému dvojnásobnému prehľadávaniu dokumentu.

5.4.5 Databáza

Zbytočne robustný nástroj Hibernate sme nahradili iBatisom, pretože nám poskytuje možnosti ako mapovať údaje na rôzne triedy na základe príznaku umiestneného v tabuľke.

Oproti pôvodnému modelu sme vypustili generalizáciu v databáze, čím sme optimalizovali a znížili počet údajov v databáze. Zmeny v modeli sú zachytené v kapitole Logický model a Fyzický model. Fyzický model odráža štruktúru databázy. Zmeny vykonané v DTO triedach sa týkali hlavne implementácie funkcií *get*, *set* a konštruktorov. Tieto funkcie boli použité, aby sa dosiahlo správneho mapovania údajov z databázy do novovytvorených objektov.

Prototyp obsahuje v databáze 13 dokumentov, ktoré sú tam vložené umelo. Ostatné dokumenty, ktoré sa v databáze nachádzajú sú výsledkom pridávania a testovania funkčnosti klienta a serverovej časti, kde sa hľadali kľúčové slová a väzby. Jednotlivé neúspešné pokusy, ktoré sa pri testovaní objavili sme následne z databázy odstránili.

Okrem toho sme využili mapovanie údajov na rôzne objekty na základe parametru. Toto nám umožnili result mapy.

6 Používateľská príručka

V tejto kapitole uvádzame používateľskú príručku k nami implementovaným častiam systému. Používateľskú príručku k používateľskému rozhraniu ostatných častí systému vytvorili predchádzajúce tímy vo svojej dokumentácii (Šproty, Starwalkers).

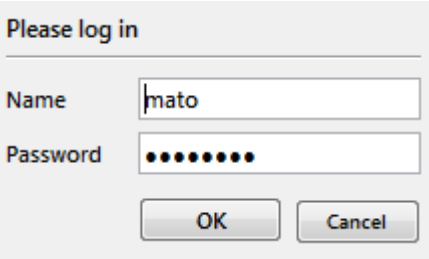
6.1 Zmeny v používateľskom prostredí 2D klienta

V 2D klientovi sme pridali prihlasovanie používateľov do systému a rozšírili sme možnosti pri uploadovaní dokumentu.

6.1.1 Prihlasovanie

Na obrázku č. 21 je znázornené prihlasovacie okno 2D klienta.

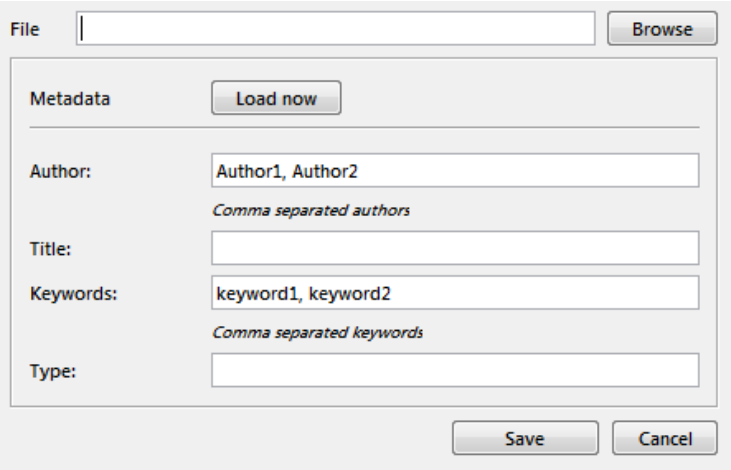
- Pri spustení systém vyzve na zadanie používateľských údajov
- Potvrdenie – OK
- Zrušenie – Cancel



Obr. 21. Prihlasovacie okno

6.1.2 Upload dokumentu

Na obrázku č. 22 je zobrazené okno pre upload dokumentu. Pri uploadovaní dokumentu pribudlo nové tlačidlo “Load now”, ktoré odošle súbor na server a predvyplní jednotlivé polia so získanými metadátami



Obr. 22. Okno pre upload dokumentu

6.1.3 3D Viewer

- Nové tlačidlo v hlavnom okne programu, ktoré spustí externý program na prehliadanie grafu v 3D.

7 Systémová príručka

Táto kapitola slúži ako systémová príručka k častiam systému – server, 3D klient, databáza.

7.1 Server

Ako už bolo spomenuté v predošlých kapitolách, serverová časť aplikácie beží na webovom serveri Jetty. Preto je nutné si tento web server stiahnuť[1] a nainštalovať na cieľovom zariadení, kde bude bežať server. Na samotné nasadenie servletov a ostatného kódu je nutné vytvoriť tzv. war súbor, ktorý zapuzdruje konfiguračné súbory, skompilované zdrojové kódy a ostatné súbory webovej aplikácie do samostatného archívu. Tento súbor je nutné nakopírovať na miesto určené pre nasadzovanie web aplikácií na serveri Jetty a následne ho spustiť alebo reštartovať.

Samotný projekt možno konfigurovať prostredníctvom viacerých XML súborov. Na manažment projektu týkajúci sa buildovania sme použili nástroj Apache Maven. V konfiguračnom súbore *pom.xml*, sú zadefinované všetky potrebné externé knižnice či už pre správny chod Spring frameworku, ale aj pre ostatné časti, ktoré si server vyžadoval (napr. na indexáciu dokumentov). Pri builde projektu sú automaticky stiahnuté zadefinované *jar* súbory s ich závislosťami z Maven repozitára. Konfiguračný súbor web.xml predstavuje deskriptor nasadenia, ktorý popisuje ako má byť web aplikácia nasadená v servlet kontajneri. Ku konfigurácii Spring projektu taktiež súvisia viaceré XML súbory (napr. servlet-context.xml, controllers.xml atď.), pričom každý súbor má určenú individuálnu úlohu v projekte. Ako IDE je odporúčaná aplikácia SpringSource Tool Suite, ktorá predstavuje populárne IDE Eclipse mierne modifikované pre Spring projekty.

7.2 3D klient

7.2.1 Požiadavky

Beh aplikácie vyžaduje mať nainštalované prostredia:

- Qt framework (otestované na verzii 4.7.1)
- OpenSceneGraph (otestované na verzii 2.8.3)
- knižnice libpng a zlib

[1] Webový server Jetty je možné stiahnuť napr. na: http://docs.codehaus.org/display/JETTY/Downloading+Jetty

7.2.2 Postup inštalácie pre platformu Windows

Inštalácia Qt frameworku

Inštalačný balík sa nachádza na stránke výrobcu (http://qt.nokia.com/downloads). K správnemu behu aplikácie nie je potrebné kompletné vývojové prostredie (Qt SDK), ale iba knižnice Qt frameworku.

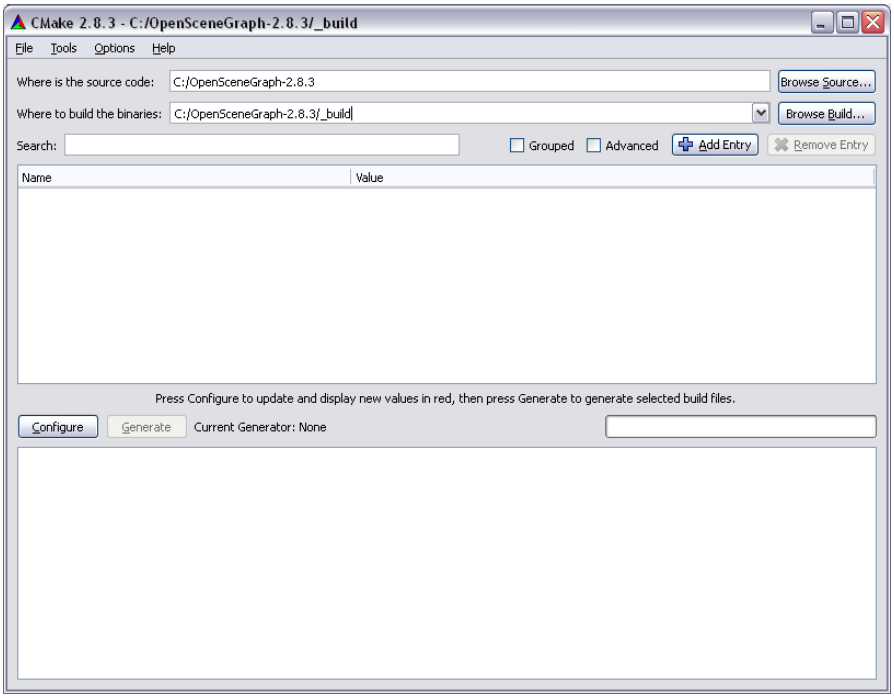
Po úspešnej inštalácii sa odporúča skontrolovať, či sa v systémovej premennej *Path* nachádza cesta k binárkam Qt frameworku. V prípade, že sa tam nenachádza, je potrebné ju tam zadať.

Inštalácia prostredia OpenSceneGraph

Inštalačný balík pre toto prostredie je pripravovaný pre verziu 3.0, momentálne existuje len jeho oklieštená verzia. Z tohto dôvodu je nutné si toto prostredie skompilovať ručne pomocou programu Cmake (dostupný na http://www.cmake.org/cmake/resources/software.html).

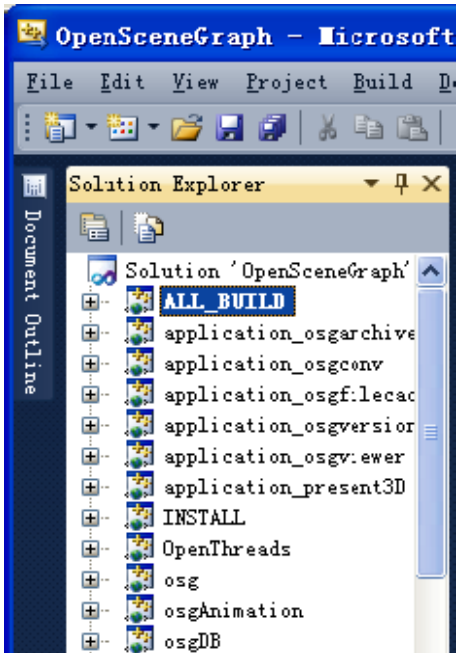
Postup inštalácie je potom nasledovný:

- Stiahneme si zdrojové kódy OpenSceneGraph (http://www.openscenegraph.org/projects/osg/wiki/Downloads) spolu s príslušnými dependencies (http://www.openscenegraph.org/projects/osg/wiki/Downloads/Dependencies)
- Spustíme používateľské rozhranie aplikácie Cmake pomocou súboru *cmake-gui.exe*, viď. obrázok č. 23.



Obr. 23. Cmake - kompilovanie OpenSceneGraph

- Nastavíme cestu k zdrojovým súborom OpenSceneGraphu a taktiež nastavíme cestu, kde sa majú skompilovať binárky (môže sa jednať aj o ten istý adresár tzv. **in-source build,** vhodné je však umiestniť tieto súbory do samostatného adresára napr. do podadresára *_build* – **out-of-source build**)
- Spustíme konfiguráciu pomocou tlačidla *Configure* a špecifikujeme prostredie (verziu Visual Studia, ktorú máme nainštalovanú)
- Prostredníctvom zobrazených premenných je možné ovplyvniť proces kompilácie, dôležité sú však 2 premenné:
    - OSG_3RDPARTY_DIR – priečinok kde sa nachádzajú dependencies
    - CMAKE_INSTALL_PREFIX – priečinok, kde budú nainštalované binárky
- Po nastavení požadovaných premenných opäť použijeme tlačidlo *Configure* a pokiaľ už nemáme žiadnu položku vysvietenú červenou, vygenerujeme pomocou tlačidla *Generate* príslušné súbory (ak sú položky vysvietené červenou farbou použijeme opäť tlačidlo *Configure*)
- V ďalších krokoch pracujem s vygenerovaným solution súborom a príslušnou verziou Microsoft Visual Studia
- Po otvorení solution súboru si zvolíme typ buildu (Debug, Release atď.) a dáme buildnúť projekt ALL_BUILD, viď. obrázok č. 24



Obr. 24. OpenSceneGraph solution v Microsoft Visual Studiu

- Ak všetko prebehlo bez problémov, tak dáme buildnúť projekt INSTALL, ktorý nám vygeneruje skompilované knižnice do adresára, ktorý sme špecifikovali pomocou premennej CMAKE_INSTALL_PREFIX

Týmto je proces kompilácie ukončený, ale pre správne fungovanie je potrebné ešte nastaviť systémové premenné nasledovne:

- **OSG_ROOT** - označuje adresár, v ktorom sa nachádzajú skompilované súbory (tento adresár bol definovaný prostredníctvom premennej CMAKE_INSTALL_PREFIX)
- **OSG_BIN_PATH** = %OSG_ROOT%\bin
- **OSG_INCLUDE_PATH** = %OSG_ROOT%\include
- **OSG_LIB_PATH** = %OSG_ROOT%\lib
- **OSG_SAMPLES_PATH** = %OSG_ROOT%\share\OpenSceneGraph\bin

Po zadefinovaní týchto premenných by mal príkaz *osgversion* v príkazovom riadku vypísať verziu OpenSceneGraph prostredia.

Inštalácia knižníc zlib a libpng

Keďže obe tieto knižnice nemajú inštalačný balík, treba ich taktiež ručne skompilovať. Tento proces je rovnaký ako pri kompilovaní OpenSceneGraph prostredia. Knižnice však treba skompilovať v nasledovnom poradí, keďže sú jedna od druhej závislé:

1. **zlib** (http://zlib.net/)
2. **libpng** (http://www.libpng.org/pub/png/libpng.html) – pred konfiguráciu kompilovania pomocou aplikácie Cmake treba pridať nasledovné premenné ZLIB_INCLUDE_DIR a ZLIB_LIBRARY, aby proces prebehol správne

V prípade potreby odporúčame preštudovať si dokumentáciu tímu Starwalkers, kde je možné dozvedieť sa ďalšie informácie.

7.3 Databáza

Prototyp používa databázu Postgres. Pre správny chod treba vytvoriť štruktúru pre údaje podľa fyzického modelu, alebo použiť dump databázy, ktorý obsahuje aj základné testovacie dáta.

Pre správny chod je potrebné do projektu pridať knižnice frameworku iBatis. Nasleduje správne nastavenie cesty v konfiguračnom súbore k jednotlivým mapovacím súborom. V konfiguračných súboroch sú definované funkcie, ktorými sa mapujú údaje na novovytvorené objekty.

Príloha A

Naplnenie databázy prototypu (tabuľka)

Documents

id	1	2	3	4	5	6	7	8	9	10	11	12	13	14
type	4	4	4	4	4	4	4	2	2	2	2	1	1	1
addedBy	1	2	3	4	1	2	3	4	1	2	3	4	1	2
fileName	D09-1096.pdf	dredze_ceas05.pdf	p232-pollock.pdf	nssilver-025.pdf	warsac-213.pdf	ajovo-613.pdf	kseadp-916.pdf	jklein-326.pdf	jma143.htm	fmphg.htm	mmmsc.htm	mailer.java	mailer.php	mailer.cs
name	Segmenting Email Message Text into Zones	Reply Expectation Prediction for Email Management	A rule-based message filtering system	Time-based visualizations of electronic mail	Conversation map: a content-based Usenet newsgroup browser	Retrieval Issues in Email Management	A dialogue perspective on electronic mail: implications for interface design	RFC 5321 - Simple Mail Transfer Protocol	JavaMail API 1.4.3	function mail	MailMessage Class	mailer.java	mailer.php	mailer.cs
author	Andrew Lampert	Mark Dredze	Pollock	Silver	Warren Sack	Jovicic	Severinson-Eklundh	Klensin	Oracle	The PHP Group	Microsoft	Marek	Juraj	Matej
keywords	email, segmentation, email zones	email, reply, context, email management	email, filter, spam, context, body	email, visualization, time, context	message, conversation map, browser, visualization	email, issue, segmentation	vsualization, email, message, design	email, protocol, trasnfer, data	email, send, return	email, subject, message	email, send, return, body	send, return, throw, try, catch	mail, message	send, try, catch

Doc Type

id	1	2	3	4
type	Source code	Documentation	Task	Article