VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

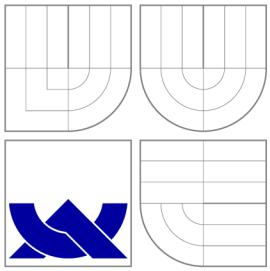
MODULY ANOTAČNÍHO SERVERU

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

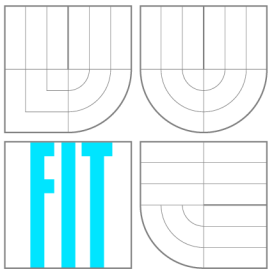
AUTOR PRÁCE
AUTHOR

JURAJ STRECHA

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

MODULY ANOTAČNÍHO SERVERU

ANNOTATION SERVER MODULES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JURAJ STRECHA
AUTHOR

VEDOUCÍ PRÁCE — Ing. JAROSLAV DYTRYCH
SUPERVISOR

BRNO 2013

Zadání bakalářské práce/15244/2012/xstrec01

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2012/2013

Zadání bakalářské práce

Řešitel: **Strecha Juraj**
Obor: Informační technologie
Téma: **Moduly anotačního serveru**
Annotation Server Modules
Kategorie: Algoritmy a datové struktury

Pokyny:

1. Seznamte se s jazykem Java a formáty XML a JSON.
2. Prostudujte dostupný server pro správu anotací vyvíjený v rámci evropského projektu Decipher a vybrané moduly tohoto serveru.
3. Navrhněte nový modul anotačního serveru pro nabízení anotací (komponent systému nazvaný Suggestion Synchronizer) a formát dat pro komunikaci s komponentem SEC Store API. Navrhněte i případné změny v dalších částech serveru potřebné pro podporu nového modulu a další vylepšení dle doporučení vedoucího.
4. Implementujte navržené řešení.
5. Zhodnoťte dosažené výsledky a srovnejte s alternativními přístupy.

Literatura:

- Dle doporučení vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- Body 1, 2 a 3.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Dytrych Jaroslav, Ing.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2012
Datum odevzdání: 15. května 2013

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2
L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Tato práce se zabývá návrhem a implementací jednoho z modulů anotačního serveru v rámci projektu Decipher. Modul má na starosti nabízení anotací k dokumentům, se kterými uživatel systému pracuje. Součástí práce je i analýza anotačního a SEC Store API serveru. Podstatná část je věnována návrhu a implementaci řešení. Samostatnou část tvoří popis testování funkčnosti modulu v systému jako celku. Práce obsahuje také návrh komunikačního protokolu mezi dvěma servery, jehož popis je uveden v dokumentu v příloze.

Abstract

The thesis deals with an Annotation server module design and its implementation in the Decipher project. The purpose of the module is to suggest annotations for document that the user works with. One will find the Annotation server and the SEC Store API server description in the beginning of the thesis. A major part of the document presents a design and an implementation of the solution. Standalone part of the thesis deals with a module functionalities testing in the system environment. The communication protocol between both servers definition and description is also included in this document.

Klíčová slova

Modul serveru, anotace, obohacování textových dat, nabídky anotací, Decipher, 4A, komunikační protokol, JSON, Java.

Keywords

Server module, annotations, textual data enrichment, annotation suggestions, Decipher, 4A, communication protocol, JSON, Java.

Citace

Juraj Strecha: Moduly anotačního serveru, bakalářská práce, Brno, FIT VUT v Brně, 2013

Moduly anotačního serveru

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing.Jaroslava Dytrycha. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Juraj Strecha
12. května 2013

Poděkování

Děkuji vedoucímu mé práce Ing. Jaroslavovi Dytrychovi za odbornou pomoc při řešení problémů, které se vyskytly při řešení této práce a za čas, který mi věnoval.

Obsah

Kapitola 1

Úvod

Úlohou mojej bakalárskej práce bolo navrhnúť, implementovať a otestovať modul anotačného serveru. Server je súčasťou výskumného projektu Decipher, ktorý sa zameriava na podporu objavovania a skúmania historického dedičstva cez vytváranie a rozprávanie príbehov [1]. Má ponúknuť riešenie pre prácu s dokumentami obohatenými o doplňujúce informácie získané pomocou špecializovaných nástrojov pre analýzu textu a znalostných databáz. Projekt je financovaný zo štrukturálnych fondov Európskej únie (FP7/2007-2013) - grant 270001 (Decipher) a rozpočtu Českej republiky (MŠMT 7E11023). Vývoj prebieha aj v rámci projektu IT4Innovations Centre of Excellence (CZ.1.05/1.1.00/02.0070). Koordinátorom projektu je univerzita Dublin Institute of Technology. Spolupracujeme tiež s The Open University.

Doplňujúce informácie k obohateniu textu vznikajú v podobe ponúk anotácií, ktoré musia byť potvrdené užívateľom. Tieto ponuky generuje SEC (Semantic Enrichment Component), ktorého riadiacou časťou je SEC Store API. Mnou vyvíjaný modul má za úlohu vytvárať komunikačný kanál medzi anotačným serverom, ktorý obsluhuje klienta s užívateľským rozhraním, a SEC Store API serverom.

Kapitola 2 čitateľovi priblíži podstatu systému vyvíjaného v rámci projektu Decipher. Časti 2.3 a 2.4 nás oboznámia s funkciami navzájom komunikujúcich serverov, aby bolo jasne vidieť, prečo je výmena správ medzi nimi dôležitá. V kapitole 3 sa dozvieme, aké technológie boli použité, a v ktorých častiach projektu pracujú. Vychádzajúc z analýzy projektu a zadaných požiadaviek som musel vytvoriť návrh riešenia modulu. Popisuje ho kapitola 4. Definuje prípady, v ktorých je potrebné použiť metódy modulu na spracovanie požiadavky od klienta. Ďalej hovorí o konkrétnych riešeniach vytvárania správy pre SEC Store API a skladaní ponuky anotácie z prijatých dát (časti 4.5 a 4.8). Čitateľ nájde detaily implementácie vybraných riešení v kapitole 5. Zistí, akým spôsobom sú serializované dáta (časť 5.1), analyzované a vytvárané nové ponuky anotácií (časť 5.3). Po naprogramovaní všetkých častí bolo potrebné toto riešenie otestovať. Metódy testovania a konkrétne prípady popisuje kapitola 6. Na záver, v kapitole 7, nájdeme zhodnotenie dosiahnutých výsledkov práce. Dočítame sa aj o možných vylepšeniach, keďže je modul súčasťou prebiehajúceho projektu vo vývoji.

Kapitola 2

Analýza

Kapitola popisuje ciele projektu a rozbor požiadaviek na modul spracovania ponúk anotácií anotačného serveru v rámci projektu Decipher. Čitateľa oboznámi so štruktúrou projektu a začlenením modulu do jeho hierarchie. Objasňuje stav, v akom sa modul nachádzal v čase, kedy som začal na zadaní pracovať.

2.1 Požiadavky projektu

Úlohou vyvíjaného systému je zjednodušiť a zefektívniť prácu s textovými dokumentami tým, že ich doplní o anotácie. Anotáciou sa rozumie doplňujúca informácia k časť textu. Ku štruktúrovaniu anotácií sa využívajú atribúty. Tými môže byť napríklad dátum, hypertextový odkaz, mesto vzťahujúce sa k danej udalosti a podobne. Príklad anotácie v editore môžeme vidieť na Obrázku 2.1. Medzi anotáciami vzniknú pomocou atribútov väzby a odkazy, čím obohatia informácie v dokumentoch o súvisiace poznatky. V projekte sa zameriavame na texty z oblasti umenia a kultúrneho dedičstva. Spolupracujeme s Írskou národnou galériou National Gallery of Ireland a Írskym múzeom moderného umenia Irish Museum of Modern Art.

Anotácie vytvárajú samotní užívatelia systému. Pre každý dokument je tiež možné požiadať o automatické vytvorenie ponuky anotácií (tzv. `annotation suggestions`). V dokumente sa zobrazia všetky dostupné ponuky, z ktorých bude možné vyrobiť anotáciu jednoduchým potvrdením pomocou ovládacieho prvku grafického rozhrania. Ak teda užívateľ ohodnotí ponuku ako prínosnú, bude ďalej zobrazovaná ako anotácia.

Ponuky generujú komponenty (moduly) SEC Store API serveru. Od anotačného serveru dostane SEC Store API identifikátor dokumentu, v ktorom sa vyznačená časť textu nachádza a pre ktorý klient požaduje vytvorenie ponuky anotácii. Dokument je spracovaný nástrojmi pre získavanie informácií z textu. Anotačný server dostane sadu ponúk. Tie zobrazí užívateľovi a ten má možnosť vybrať, ktoré sú pre neho zaujímavé a prínosné. V prípade, kedy server nie je dostupný, musí anotačný server oznámiť túto skutočnosť klientovi a vymazať všetky zobrazené ponuky, pretože s nimi ďalej nemôže pracovať. Naopak, SEC Store API musí mať v každom okamihu prehľad o zobrazených ponukách v konkrétnom sedení užívateľa. Tento stav dosiahnem výmenou správ medzi oboma servermi.

Pre výmenu informácií, posielanie požiadaviek a odpovedí je potrebné navrhnúť komunikačný protokol. Ten musí byť schopný efektívne prenášať veľké množstvo dát ponúk anotácií, ako aj krátke správy o zmene stavu u klienta, napríklad potvrdenie ponuky anotácie, vypnutie zobrazenia anotácií alebo ukončenie sedenia.

Obrázek 2.1: Anotácia so svojimi atribútmi v anotačnom editore

Cieľom mojej práce bolo vytvoriť modul anotačného serveru nazvaný Suggestion synchronizer, ktorý bude schopný spravovať ponuky anotácií pre klientskú aplikáciu a zabezpečí komunikáciu so SEC Store API serverom. Každá interakcia s návrhom anotácie a modifikácia textu dokumentu musí byť zaznamenaná SEC Store API serverom. Predpokladom pre úspešný návrh a implementáciu riešenia bolo oboznámiť sa s princípmi fungovania a štruktúrou už existujúcich serverov. Celková architektúra systému je na diagrame balíčkov v prílohe A. Detailnejší popis architektúry anotačného systému nájdeme v prílohe B. Fungovanie jednotlivých serverov je rozobraté v nasledujúcich odsekoch.

2.2 Stav na začiatku

Na začiatku mojej práce na projekte existoval v súvislosti s ponukami anotácií testovací modul – stub. Jeho možnosti boli veľmi obmedzené. Fungoval tak, že anotačný server spracoval správu s požiadavkou od klienta. Text celého dokumentu uložil do samostatného súboru. Názvy súborov generoval čítač postupným inkrementovaním číselnej hodnoty. Predchodcami SEC Store API serveru boli skripty pre spracovanie textu volané lokálne anotačným serverom. Zo súboru vyextrahovali informácie a uložili ich do nového súboru. Odtiaľ boli načítané anotačným serverom. Vytvorili sa ponuky anotácií. Ich pozície v texte stub kontroloval pomerne zložitými postupmi a vyraďoval tie, v ktorých sa fragmenty anotácií nezhodovali. Testovací stub nedokázal kontrolovať závislosti medzi ponukami, pretože neexistovali nástroje na ich vytvorenie. Potvrdené a odmietnuté ponuky prichádzali znovu aj napriek tomu, že ich prínos už užívateľ ohodnotil (potvrdil alebo odmietol). Nikde sa neudržiaval

aktuálny stav sedenia užívateľa. Identifikátory ponúk mali priradenú číselnú hodnotu a prideľoval ich inkrementujúci čítač. Testovací modul teda vykonával veľké množstvo operácií, ktoré mali v skutočnosti byť úlohou serveru pre spracovanie textu.

Toto riešenie bolo v praxi nepoužiteľné, neefektívne a pomalé kvôli častým prístupom k pevnému disku a opakovanému časovo náročnému volaniu nástrojov na spracovanie textu na rovnaký dokument. Preto vznikol koncept samostatného serveru pre spracovanie obsahu textových dokumentov – SEC Store API server.

2.3 Anotačný server

V rámci projektu Decipher slúži pre prácu s užívateľskými sedeniami, dokumentami, anotáciami a ich ponukami. Nazýva sa aj 4A server [2]. Komunikuje s klientskou časťou – Storyscope a grafickým anotačným editorom (AEd) alebo anotačným doplnkom pre internetové prehliadače (4A Extension). V databáze udržuje kópie dokumentov, anotácie každého z užívateľov, typy anotácií, zoznam užívateľov, ich užívateľských nastavení a príslušnosť ku existujúcim skupinám.

Požiadavka od klienta je pomocou metód v komponente `Message Processor` spracovaná a uložená do štruktúry reprezentujúcej dáta požiadavky pre moduly. V prípade chyby generuje XML správu pre klienta. Dátová štruktúra požiadavky následne putuje do komponenty `Response Creator`. Ten postupne volá všetky moduly serveru v presne nadefinovanom poradí. `Data storing` modul sa postará o uloženie dát vytvorených užívateľom. Jeho úlohou je udržiavať správne závislosti medzi objektami databáze. Pre ich ukladanie používa MySQL [1] databázu. Pre prácu s ňou využíva Java Persistence API [2]. `Persistence manager` tvorí vrstvu pre zjednodušenie práce s Java Persistence API. `Core` module pridáva, modifikuje alebo odstraňuje anotácie. Vytvára odkazy a udržuje závislosti medzi anotáciami pomocou atribútov. Modifikuje tiež kópiu dokumentu v aktuálnom sedení. Vykonáva všetky potrebné úkony v prípade synchronizácie a resynchronizácie – porovnanie novej verzie dokumentu s uloženou a následná úprava anotácií, zaslanie nových informácií o anotáciách klientovi. Znovuzaslanie anotácií, vyhľadávanie typov anotácií v databáze a aplikovanie užívateľských nastavení má taktiež na starosti modul `Core`. `User manager` sa stará o prácu s užívateľmi, skupinami užívateľov a vyhľadávaním informácií o nich. Dictionary service pracuje s kontrolovanými slovníkmi. Po uložení dát do databázy sa znovu zavolajú moduly a nakoniec sa užívateľovi vráti výsledná odpoveď.

Pre posielanie údajov o nových, modifikovaných a odstránených anotáciách na iné systémy slúži `Annotation exporter`. Je možné nastaviť ho tak, aby posielal iba vybrané informácie. Výstupným formátom je XML. Dáta sa vždy posielajú kompletné, aby bolo vždy možné vytvoriť správne závislosti. V prípade, kedy je externý systém nedostupný, dáta sa uložia do databázy a odošlú sa po opätovnom pripojení externého systému.

Dôležitou súčasťou anotačného serveru je `Linearizer`. Pomocou neho je možné transformovať tzv. bežné fragmenty na linearizované a opačne. Proces linearizície fragmentov je dôležitý pre správne vypočítanie pozície anotácie alebo ponuky anotácie v dokumente. Nástroje SEC Store API pracujú s linearizovanou verziou dokumentu na rozdiel od anotačného serveru, ktorý v dokumente používa formátovacie značky. Pozície vypočítava pomocou hľadania zhody v XPath uzlov dokumentu a samotného fragmentu. Postupne prechádza celý dokument a ukladá textový obsah jednotlivých uzlov. Reťazec vznikajúci kumulovaním tex-

[1] http://www.mysql.com

[2] http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html

tového obsahu je linearizovaný, jeho dĺžku zaznamenáva do pomocnej premennej, ktorá slúži k určeniu absolútnej pozície textu fragmentu určeného na linearizáciu. `Linearizer` tiež vytvára linearizovanú podobu dátovej štruktúry obsahujúcej dáta o modifikácii v dokumente.

Anotačný server pre každého užívateľa udržuje zoznam ponúk anotácii, informácie o fragmente, ku ktorému sú požadované ponuky, aktuálnu kópiu dokumentu, s ktorým práve pracuje a dáta potrebné k interakcii s klientom – jazyk a pripravené správy pre grafické užívateľské prostredie.

2.4 SEC Store API server

Zabezpečuje extrakciu a spracovanie znalostí z textových dokumentov. Kolekcia objektovej databázy MongoDB [3] uchováva ponuky anotácií pre dokumenty, ku ktorým ich užívatelia požadovali počas aktuálneho sedenia. S anotačným serverom musí komunikovať navrhnutým protokolom. Čaká na HTTP správu, ktorú spracuje. Posiela odpovede na požiadavky obsahujúce identifikátor dokumentu, prípadne jeho obsah alebo modifikácie. Ak k danému dokumentu neexistuje záznam v databáze alebo bol text jeho modifikovaný, začne proces tvorby ponúk. Na tom sa podieľajú moduly SEC Store API serveru. Každý z nich musí implementovať dohodnuté aplikačné rozhranie. Moduly teda môžeme jednoducho pridávať, čím dosiahneme rozšírenie funkcionality a možností serveru.

Sada klasifikátorov dokáže v texte rozpoznať rozmery, finančné hodnoty, hmotnosť, objem a hustotu. Iné klasifikátory pracujú na úrovni celého dokumentu a rozpoznávajú žáner, štýl, pozitívny alebo negatívny sentiment a podobne.

Významným modulom je NER – Named Entity Recognizer. Pracuje so znalostnou databázou, ktorú si skompiluje do konečného automatu. Pri štarte procesu rozpoznávania je automat načítaný do operačnej pamäti, preto dokáže rýchlo a efektívne spracovávať veľké množstvo textu (30 000 slov za sekundu). Po prvom priechode dokumentom použije ďalšie nástroje, ako napríklad Relation extractor, a vráti výsledok. Jeho výstup manažér serveru spracuje a vytvorí dáta pre anotačný server.

Kolekcia ponúk anotácií dokumentu obsahuje okrem dát vytvorených modulom NER tiež anotácie od iných užívateľov. V porovnaní s automaticky vygenerovanými ponukami dostanú od modulu vyššiu hodnotu úrovne dôveryhodnosti (confidence).

Pre každé užívateľské sedenie v databáze udržiava zoznam aktuálne zobrazených ponúk anotácii. V priebehu sedenia dostáva správy o zmenách stavu na klientskej časti a databázu aktualizuje. Kolekcia potvrdených alebo odmietnutých ponúk obsahuje záznamy o tom, ktorá anotácia vznikla z ktorej ponuky a naopak, ktorá ponuka vznikla z ktorej anotácie. Tým sa zamedzí opätovnému posielaniu už zobrazených a ohodnotených ponúk. To všetko je úlohou Konwledge repository manager komponenty v rámci SEC Store API serveru.

Od anotačného serveru tiež dostáva informácie o zmenách, ktoré užívateľ vykonal v texte dokumentu. Modifikácie aplikuje na svoju verziu uloženého dokumentu a aktualizuje ponuky anotácií. Niektoré ponuky zaniknú, iné budú musieť byť za pomoci nástrojov NER vytvorené a časť už existujúcich zmení svoju pozíciu v texte. SEC Store API server udržiava väzby a závislosti medzi ponukami. Ak zanikne vnorená alebo odkazovaná ponuka, všetky atribúty odkazujúce sa na ňu musia byť odstránené.

[3] http://www.mongodb.org/

Po potvrdení ponuky dáva pozor, aby zostala zachovaná správna štruktúra odkazov. Ak atribút inej ponuky odkazoval na ponuku, ktorá bola potvrdená, aktualizuje odkaz tým, že dočasný identifikátor `tmpId` nahradí adresou uloženej anotácie, ktorá vznikla po potvrdení ponuky.

Kapitola 3

Využité technológie

Pre komunikáciu medzi klientskou aplikáciou a anotačným serverom používame v projekte technológie

- Ajax
- Comet (jednosmerná komunikácia server – klient)

Správy medzi anotačným editorom a anotačným serverom putujú vo formáte XML 1.0. Odľahčený formát pre výmenu informácií JSON [6] sa v projekte využíva k prenosu ponúk anotácií medzi anotačným serverom a extraktorom informácií z textu. Dáta sú potom zabalené do HTTP správ. Načítanie prijatého textového reťazca správy do dátového typu reprezentujúceho JSON dokument majú na starosti metódy knižnice JSON.simple. Programovacím jazykom anotačného serveru je Java Enterprise Edition (Java EE).

3.1 Ajax/Comet

Ajax je súhrnným názvom pre skupinu technológii, ktoré dokážu urobiť webovú aplikáciu viac interaktívnu a odstránia potrebu obnovenia celej stránky v prípade, kedy chceme zmeniť iba jednu jej časť. Princípy Ajax existovali ešte predtým, ako bol samotný názov definovaný v roku 2005 v článku Jesse James Garetta [7]. Namiesto toho, aby sa pri zavedení stránky načítala statická webová prezentácia, prehliadač inicializuje Ajax engine napísaný v jazyku JavaScript. Ten je zodpovedný za vykresľovanie prvkov stránky a komunikáciu so serverom [12]. Pri zasielaní správ anotačnému serveru využíva klient niektoré z Ajax prostriedkov – požiadavky vo formáte XML a asynchrónnu komunikáciu zabezpečenú pomocou jazyka JavaScript.

Comet je programovacia technika, ktorá umožňuje webovému serveru zasielať správy klientovi bez toho, aby o ne explicitne požiadal [16]. Názov navrhol v roku 2006 Alex Russel [19], kedy bolo potrebné zastrešiť množstvo technológií pracujúcich na tzv. push princípe jednosmernej komunikácie v reálnom čase. Podobnú komunikáciu používajú finančné aplikácie, aukčné servery a herné systémy. Súčasné projekty však vyžadujú obojsmernú komunikáciu, a teda vznikajú nové technológie. Tie sa, či už z marketingových dôvodov, alebo kvôli technickým odlišnostiam, nenazývajú Comet [4]. V projekte je použitá implementácia Oracle GlassFish Server 3.1 – Oracle Grizzly Comet.

3.2 XML

XML, ako skratka pre Extensible Markup Language popisuje triedu objektov nazvaných XML dokumenty a čiastočne popisuje správanie programov, ktoré ich spracovávajú [21]. V praxi sa používa pre zápis dokumentov obsahujúcich štruktúrované informácie (slová, obrázky a podobne) a akú rolu v kontexte zohrávajú [23]. Prvý štandard jazyka bol predstavený konzorciom W3C v roku 1998. Vychádza zo staršieho štandardu SGML (ISO 8879). Piata verzia existuje od roku 2006 a odporúča sa jej používanie aj naďalej. V súčasnosti existuje tiež štandard 1.1, ktorý sa od 1.0 líši hlavne definíciou povolených znakov v názvoch položiek [22].

3.3 JSON

Ako textový formát zápisu dát je ľahko čitateľný ako pre počítače, tak aj pre vývojárov aplikácie. Je odvodený od ECMAScript Programming Language Standard. Posledná špecifikácia bola uvedená v júli roku 2006 Douglasom Crockfordom [5] na stránke Introducing JSON [1].

3.4 JSON.simple

Sada nástrojov pre prácu s JSON dátami určená pre jazyk Java. Jej vývoj a testovanie má na starosti úzka komunita piatich členov. Dokáže vytvárať aj načítavať informácie do JSON formátu, ako aj escapovať špeciálne znaky. Programátorom ponúka jednoduché rozhranie. Dodržiava špecifikáciu JSON podľa RFC4627 [2]. Prvá verzia bola uvedená v roku 2006, aktuálnym ponúkaným zostavením je 1.1.1 z roku 2012. [10]

3.5 Java Enterprise Edition

Jedná sa o verziu odvodenú z Java Standard Edition (aktuálne vo verzii 7). Java SE je objektovo orientovaným triedne založeným konkurentným programovacím jazykom pre všeobecné použitie. Zdrojový kód je skompilovaný do sady inštrukcií bytecode a binárneho formátu zadefinovaného pre použitie v Java Virtual Machine. Používa Garbage Collector pre automatické uvoľnenie alokovaných zdrojov. [13] Java EE je určená pre použitie na strane serveru. Obsahuje naviac technológie pre prácu s webovými aplikáciami a databázou. Aktuálna je verzia Java EE 6 [18]. Firma Oracle pripravuje vydanie novej verzie Java EE 7 [14]. Produkčným serverom je GlassFish Server Open Source Edition v aktuálnej verzii 3.1.2.2. [3].

3.6 HTTP

HTTP je skratkou pre Hypertext Transfer Protocol. Jedná sa o textový protokol prenosu správ medzi zariadeniami pripojenými do celosvetovej siete Internetu. Prvú verziu predstavil v roku 1991 Tim Berners - Lee v rámci návrhu World Wide Web. [20] Poslednou verziou je od roku 1999 HTTP 1.1, ktorá sa používa dodnes [11]. Množstvo vyšších programovacích jazykov ponúka vo svojej štandardnej knižnici nástroje pre prácu s technológiou HTTP.

[1] http://www.json.org
[2] http://www.ietf.org/rfc/rfc4627.txt

Kapitola 4

Návrh

Kapitola sa zaoberá voľbou vhodných prostriedkov pre riešenie, prípravou riešenia a zhodnotením vhodnosti navrhnutých technologických postupov.

4.1 Rozhranie modulu

Definícia triedy modulu SuggestionsModule implementuje rozhranie modulu, ktoré je spoločné pre všetky moduly anotačného serveru. Rozlišujú sa dva prípady, kedy sa volajú jeho metódy: jedna metóda pred uložením a druhá po uložení. Ak bude modul aktívny pred uložením nových dát a zmien do databázy, skontroluje sa, či sa nezmenil dokument – došlo ku synchronizácii. V tom prípade vymaže zoznam ponúknutých anotácií, pretože nezodpovedajú aktuálnemu obsahu dokumentu. Po spracovaní správy od klienta a trvalom uložení zmien nastáva druhý prípad, kedy by sa malo spustiť nové vlákno modulu. Paralelné spracovanie je dôležité preto, aby nedochádzalo ku zastavovaniu serveru počas doby, kedy čaká na odpoveď zo SEC Store API. Metóda teda vykoná iba skonštruovanie vlákna a jeho štart.

4.2 Vlákno modulu

Pre každú požiadavku sa vytvorí jedno vlákno. To skontroluje, čo je obsahom správy od užívateľa, a vykoná zodpovedajúcu akciu. Môže nastať 9 prípadov, kedy budeme pracovať s ponukami anotácií:

- synchronizácia
- resynchronizácia
- potvrdenie ponúk
- odmietnutie ponúk
- uloženie alebo modifikácia anotácií
- odstránenie anotácií
- žiadosť o ponuky anotácií
- modifikácia textu

- odhlásenie užívateľa

  Detekciu týchto prípadov zabezpečia samostatné metódy. Každá z nich vyhodnocuje práve jeden prípad. Testovaním položiek požiadavky na neprázdny zoznam, nenulovú dĺžku reťazca alebo pravdivostnú hodnotu príznaku sa modul dozvie, či nastal jeden z prípadov, ktoré sú uvedené nižšie.

4.2.1 synchronizácia

Keď užívateľ začne pracovať s iným dokumentom, dôjde ku synchronizácii. Zobrazené ponuky už nie sú platné pre zobrazený dokument, musia sa teda odstrániť a požiadať SEC Store API server o ponuky k novému dokumentu. Ak nebol v novom texte zvolený fragment pre nové ponuky, SEC Store API si iba poznačí zmenu dokumentu.

4.2.2 resynchronizácia

Ak sa server dostane do chybového stavu, požiada sa o znovuzaslanie ponúk. Bude sa jednať o zotavenie zo situácie, kedy nastala nekonzistencia v dokumente na strane klienta a anotačného serveru. SEC Store API pošle zoznam ponúk, ktoré boli zobrazené v sedení. Tými sa nahradia ponuky na anotačnom serveri tak, aby sa zhodovali s ponukami grafického užívateľského prostredia. Táto operácia sa deje na pozadí, užívateľ ju nijako neovplyvňuje.

4.2.3 Potvrdenie ponúk

Potvrdením ponuky vzniká anotácia pre užívateľa, ktorý s ňou pracoval. Aby ju pri ďalšej požiadavke znovu nedostával, SEC Store API server dostane oznámenie o reakcii na ponuku. Kvôli zachovaniu správnych závislostí a funkčných odkazov medzi ponukami bude potrebné spolu s identifikáciou ponuky zaslať aj URI adresu novovzniknutej anotácie. Prijatie môže prebehnúť automaticky alebo s pomocou užívateľa, preto SEC Store API musí obdržať aj identifikačné číslo metódy potvrdenia. Ak užívateľ ponuku pred potvrdením modifikuje, pošle sa v správe aj jej nový obsah anotácie. Reakcia na ponuku bude identifikovaná číslom sedenia, adresou anotačného serveru a skupinou, ktorej patrí typ ponúknutej anotácie.

4.2.4 Odmietnutie ponúk

Princíp je rovnaký ako pri potvrdení ponuky s tým rozdielom, že hlavná položka obsahuje reťazec označujúci odmietnutie.

4.2.5 Uloženie a modifikácia anotácií

Nové anotácie, ktoré sa uložia do databáze anotačného serveru, musia byť preposlané aj na SEC Store API server. Pre jej znovuvytvorenie na anotačnom serveri vo forme ponuky musia byť zvolené všetky potrebné položky.

Počiatočná a koncová pozícia v texte, anotovaný text a URI adresa dokumentu budú použité pre vytvorenie fragmentu anotácie. Každá anotácia alebo ponuka je určitého typu. Typy sú identifikované pomocou cesty v strome typov.

Anotácia môže mať atribúty s doplňujúcimi údajmi. Hodnota jednoduchých typov bude číslo alebo reťazec znakov. Štruktúrované typy sú buď odkazy alebo vnorené anotácie. V tom prípade atribút obsahuje identifikáciu typu odkazovanej alebo vnorenej anotácie. Hodnotou atribútu typu odkazovaná anotácia je URI anotácie. Hodnotou atribútu typu vnorená anotácia sú dáta tejto vnorenej anotácie.

4.2.6 Odstránenie anotácií

Po vymazaní anotácie z databázy anotačného serveru užívateľom musí byť SEC Store API server oboznámený o tejto akcii. V správe dostane identifikátor anotácie a zoznam URI skupín, do ktorých užívateľ patrí.

4.2.7 Žiadosť o ponuku anotácií

Aby užívateľ mohol pracovať s ponukami anotácií, musí o ne najskôr požiadať. Môže si vybrať medzi ponukami pre celý alebo iba vyznačenú časť dokumentu, s ktorým aktuálne pracuje. Tieto prípady sa v správe budú odlišovať na základe počiatočnej a koncovej pozície vyznačeného fragmentu. Dátová štruktúra požiadavky od klienta obsahuje tento fragment. Ak je jeho dĺžka nedefinovaná, značí to celý dokument. V tom prípade SEC Store API dostane požiadavku, kde budú mať na oboch pozíciách hodnoty -1. Inak sa nastavia odsadenia od začiatku na hodnoty získané z dát fragmentu. Pri práci s ponukami môže nastať ešte jedna situácia – užívateľ vypne zobrazenie ponúk anotácií. Zo záznamu o sedení na SEC Store API sa teda musia odstrániť. Obe pozície v správe majú hodnotu 0.

Ak má užívateľ záujem o konktérny typ ponúkaných anotácií, jeho identifikácia sa odošle na SEC Store API. Ten vyfiltruje ponuky daného typu a vráti ich anotačnému serveru v odpovedi.

4.2.8 Modifikácia textu

Užívateľ môže v grafickom rozhraní meniť obsah dokumentu. Ak pridá alebo odstráni časti textu, niektoré ponuky anotácií zmiznú alebo sa zmení ich pozícia v texte. Modul ponúkania anotácií spracuje a prepošle údaje o zmene na SEC Store API. K tomu je potrebné vedieť, počiatočnú a koncovú pozíciu zmeny, nový obsah tohoto fragmentu, identifikáciu dokumentu a URI anotačného serveru, na ktorom je uložený. Počiatočná a koncová pozícia zmeny sa vypočítava z XPath [1] XML dokumentu. Tá určí odsadenie a dĺžku elementu obsahujúceho zmenu. Ďalšie dopočítavanie teda nastáva vnútri elementu. Dôležité je, aby sa pozície vypočítali v starej neupravenej verzii dokumentu. Odpoveďou bude zoznam ponúk anotácií s novými pozíciami a zoznam tých, ktoré už v texte neexistujú a musia sa odstrániť z aktuálneho sedenia.

4.2.9 Odhlásenie užívateľa

Po odhlásení užívateľa zaniká sedenie nie len v prostredí anotačného serveru, ale záznamy o ňom musí zrušiť aj SEC Store API. Ten obdrží číslo sedenia a adresa anotačného serveru, podľa ktorého vyhľadá v databáze záznamy spojené s týmto sedením.

4.3 Odpoveď

Po každom odoslaní správy na SEC Store API očakáva anotačný server odpoveď v dohodnutom formáte. Skladá sa z 2 častí:

- zoznam ponúk, ktoré sa vytvoria
- zoznam ponúk, ktoré sa odstránia

Vzniknú teda 2 metódy pre spracovanie dát odpovede.

[1] http://www.w3schools.com/xpath/xpath_intro.asp

4.4 Komunikačný protokol

Na základe návrhu z predchádzajúcich odstavcov som vytvoril definíciu komunikačného protokolu. Správy medzi servermi budú putovať v textovom formáte JSON. Ten som zvolil preto, lebo prostriedky SEC Store API veľmi dobre zvládajú prácu s ním, pri ladení je čitateľný pre človeka a vyznačuje sa nízkou réžiou pri práci. V správe bude prvou položkou názov metódy. Závisí od toho, akú operáciu užívateľ vykonal. Ďalšie údaje sa pridávajú podľa typu požiadavky, ktorú server obdržal od užívateľa.

Ak správa obsahuje element s názvom `getsugg`, znamená to, že užívateľ poslal žiadosť o sadu ponúk anotácií. Dôležité je identifikovať, pre ktorú časť dokumentu ponuky žiada. Ak sú hodnoty počiatočnej a koncovej pozície v texte -1, užívateľ požaduje všetky existujúce ponuky. Keď oba elementy nesú hodnotu 0, užívateľ si vypol zobrazovanie ponúk, o čom musí SEC Store API vedieť. Správa tiež nesie informáciu o synchronizácií, kedy užívateľ začne pracovať s iným dokumentom. Vtedy musí SEC Store API server vygenerovať sadu nových ponúk pre aktuálny dokument. Znovuzaslanie všetkých ponúk, ktoré drží v kolekcii aktuálneho sedenia užívateľa realizuje v prípade, ak je hodnota príznaku reprezentujúceho požiadavku na resynchronizáciu 1. Spolu s požiadavkou posiela anotačný server vždy aj linearizovanú verziu dokumentu, aby si ju mohol SEC Store API uložiť a vytvoriť ponuky v prípade, že tento dokument dostal prvýkrát.

Ďalej ho udržuje pomocou správ o modifikáciách. Tie sú identifikované elementom s názvom `textModification`.

Všetky anotácie vytvorené užívateľom sú zároveň odoslané na SEC Store API server v správe JSON identifikovanej elementom s názvom `store`. Ten ich pridá do zoznamu ponúk dokumentu a priradí im vyššiu dôveryhodnosť. Iní užívatelia takto môžu získať zaujímavé ponuky anotácií, ktoré nedokážu nástroje NER vygenerovať v spolupráci so znalostnou databázou.

Ak užívateľ anotáciu odstráni, SEC Store API server obdrží JSON dáta s elementom pomenovaným kľúčovým slovom `delete`.

Dôležitou správou je oznámenie o reakcii na ponuku. Užívateľ ju môže potvrdiť a prehlásiť za relevantnú, čím sa stane anotáciou dokumentu, alebo odmietnuť. Kľúčovým slovom, pomenúvajúcim element JSON, je v tomto prípade `suggFeedback`.

Ďalší z identifikátorov v správe je `logout`, ktorý serveru hovorí, aby vymazal kolekciu zobrazených ponúk, pretože užívateľ ukončil svoje aktuálne sedenie.

Na každú z požiadaviek sa anotačnému serveru vráti odpoveď obsahujúca dva zoznamy informácií o ponukách. Prvý, identifikovaný kľúčovým slovom `delete`, poskytne identifikátory ponúk, ktoré musia byť odstránené zo zoznamu zobrazených ponúk aktuálneho sedenia. Druhý, identifikovaný kľúčovým slovom `create`, nesie dáta nových alebo modifikovaných ponúk určených pre vytvorenie v prostredí anotačného serveru.

Kompletný popis komunikačného protokolu s vysvetlením všetkých položiek, syntaxou a príkladmi správ nájdeme v prílohe C.

4.5 Vytvorenie správy

Na anotačnom serveri existuje pre každú požiadavku od klienta dátová štruktúra obsahujúca jej dáta. Knižnice programovacích jazykov pracujúce s JSON formátom dokážu z Java objektov vytvoriť ich JSON reprezentáciu. Dátová štruktúra požiadavky je však rozsiahla a obsahuje referencie na iné objekty. Výhodnejšie bude spracovať významné položky do textovej podoby pomocou metód základných tried jazyka Java (toString()). Hodnotami

sú URI, čísla a textové reťazce označujúce názvy a podobne. Postupne budeme pridávať pevne nadefinované reťazce s názvami JSON položiek a hodnoty významných položiek požiadavky do objektu typu reťazec. Hodnoty položiek anotácií a atribútov získame z objektov tried, ktoré ich reprezentujú. Vhodným riešením je modifikovať už existujúce súbory definície tried a pridať do nich metódu generujúcu JSON reťazec. Kombináciou volania týchto metód, pridávania pevne nadefinovaných reťazcov názvov položiek a hodnôt z požiadavky užívateľa vznikne naformátovaná správa, ktorú dokáže spracovať metóda na strane SEC Store API serveru.

4.6 Komunikačný kanál

V začiatkoch mojej práce neexistovalo spojenie medzi anotačným serverom a SEC Store API serverom. Ten však potrebuje byť informovaný o dianí na obrazovke užívateľa. Keď žiada o ponuku anotácií, prijme alebo odmietne ponuku, synchronizuje si dokument alebo zruší zobrazenie ponúk, zašle po sieti správu oznamujúcu túto udalosť. Vhodným protokolom pre prenos textových správ je HTTP. Programovacie jazyky ponúkajú vo svojich štandardných knižniciach nástroje pre posielanie a prijímanie HTTP správ, takže nebude potrebné použiť externú knižnicu tretej strany. Jazyk Java pre tento účel implementuje triedu HTTPUrlConnection [17]. Pre preposielanie a prijímanie textových správ SEC Store API serveru som navrhol triedu `NlpConn`. Každý modul, ktorý bude chcieť vytvárať toto spojenie, bude musieť inicializovať objekt komunikačnej triedy URL adresou druhej komunikujúcej strany. Metóda pre zasielanie požiadavku dostane ako parameter vytvorenú textovú správu požiadavky a návratovou hodnotou bude odpoveď na ňu. Ak sa komunikácia nepodarí, vráti hodnotu `null`.

4.7 Spracovanie odpovede

Na každú z požiadaviek odpovie SEC Store API sadou dát, ktoré obsahujú nové ponuky anotácií a identifikátory ponúk, ktoré je potrebné odstrániť. Správa bude uložená v textovej podobe do dátovej štruktúry ako reťazec znakov. Metódy knižnice pre prácu s JSON dátami vytvoria objekt. Vzniknú 2 metódy. Jedna vyrobí nové ponuky a druhá zo serveru odstráni už nepotrebné.

4.8 Vytvorenie ponúk

SEC Store API server spracuje výstup z nástroja NER tak, aby anotačný server obdržal pripravenú sadu informácií, pomocou ktorých dokáže vyrobiť ponuku pre anotáciu. Pre prípad, že sa spojenie so serverom nepodarilo nadviazať alebo pri spracovaní požiadavky od anotačného serveru došlo ku chybe, musí byť použitá kontrola odpovede na prázdnu hodnotu (`null`). Nasledujúca metóda bude z hľadiska implementácie najzložitejšia, pretože bude vytvárať nové objekty, často naviac poprepájané odkazmi.

Objekt, reprezentujúci JSON štruktúru dát, bude postupne prechádzať a pre každý záznam spustí tvorbu anotácie. Jednotlivé položky uloží do pomocných premenných. Hodnoty budeme získavať metódou, ktorej pomocou parametra predáme názov položky. Po jej načítaní aplikácia skontroluje, či sú vyplnené tie, ktoré nemôže nadobúdať prázdnu hodnotu (identifikátor, pozície v texte, typ anotácie, názvy a hodnoty atribútov).

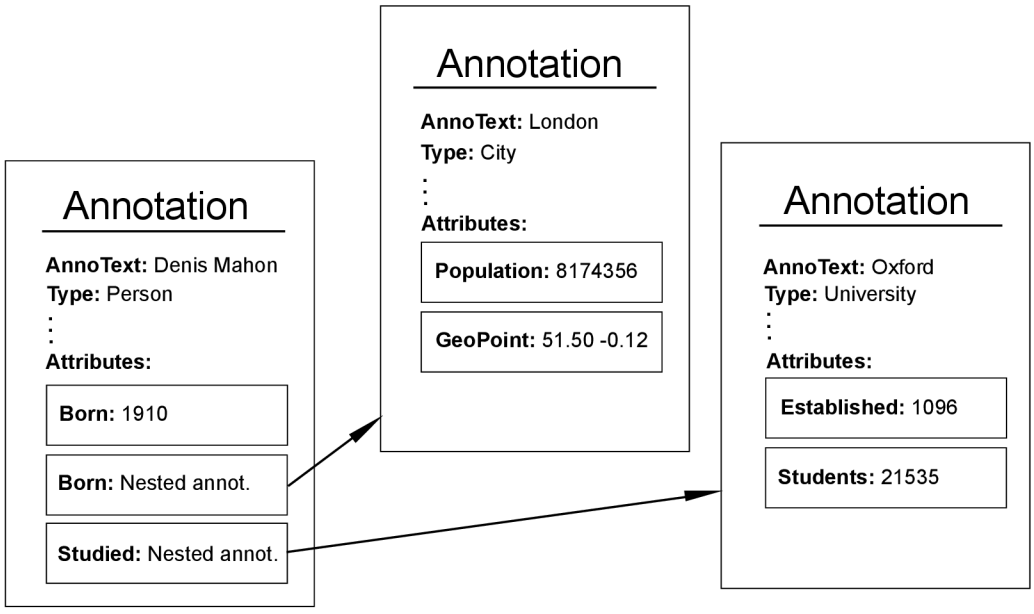
Najdôležitejšou súčasťou anotácie je fragment textu, ku ktorému patrí. Vznikne z hodnoty počiatočnej, koncovej pozície a obsahu anotovanej časti textu. Linearizovaný fragment je ale nedostatočný, neberie do úvahy formátovanie textu. Preto bude musieť byť neskôr nahradený tzv. bežným fragmentom.

Ďalším úkonom je vytvorenie atribútov. Tie, ktoré sú jednoduchých typov, použijú dáta získané spracovaním JSON objektu, nebude teda potrebné ďalšie spracovanie. Iná situácia nastáva v prípade štruktúrovaného atribútu. Referencia je v tejto chvíli vytvorená iba v prípade, že odkazovaná anotácia existuje v databáze. Atribúty typu vnorená anotácia budú zatiaľ obsahovať iba identifikátory odkazovaných ponúk, ktoré na anotačnom serveri v čase inicializácie atribútu ešte nemusia existovať.

Každej anotácii a ponuke prislúcha typ. Je dôležité si uvedomiť, že typ získaný z NER nemusí byť u každého z užívateľov známy. Počas spracovania ponuky budeme musieť najskôr zistiť, či databáza obsahuje tento typ. Ak nie, pomocou Java Persistence API ho natrvalo uloží. Kontroluje sa aj existencia nadradených rodičovských typov. V prípade, že databáza neobsahuje nadtyp, musí ho modul vytvoriť a uložiť spolu s novým typom.

Týmto spôsobom spracujeme a vytvoríme všetky ponuky. Metóda triedy `Linearizer` konvertuje linearizované fragmenty na obyčajné a nahradí ich v každej z anotácií.

Aby sme zachovali štruktúru anotácií, vyplníme položky atribútov typu vnorená anotácia referenciami na práve vytvorené objekty. Závislosti vidno na príklade z Obrázku 4.1. Ak je položka „vnorená v“ neprázdna, obsahuje identifikátor ponuky, do ktorej je zanorená. Takáto ponuka nemôžeš existovať samostatne, bude možné s ňou pracovať iba cez atribút rodičovskej ponuky. Doteraz sme vytvorili sadu anotácií v operačnej pamäti. V rámci jazyka



Obrázek 4.1: Štruktúra vnorených anotácií

Java bude inštanciou triedy Annotation. K uloženiu dochádza až po potvrdení ponuky. Tá z anotácie vznikne po tom, ako referenciu uložíme do objektu obaľovacej (wrapper) triedy ponuky anotácie (SuggestionLogEntry). Bude sa jednať o anotáciu, ktorá ešte nie je uložená natrvalo v databáze. Ponuky anotácií po vytvorení uvidí klient v grafickom užívateľskom prostredí. Modul udržiava zoznam zobrazených ponúk a všetky informácie o zmenách (nové, odstránené) posiela klientovi. Bez oznamovania zmien by klient a server obsahovali dve roz-

dielne sady ponúk.

4.8.1 Odstránenie ponúk

Spracovanie časti odpovede s ponukami pre odstránenie som vyriešil vyhľadaním zhody v identifikátoroch. Ak sa identifikačný reťazec ponuky v zozname pre vymazanie zhoduje s dočasným identifikátorom niektorej zo zobrazených, ponuka bude odstránená. Klient zároveň dostane informáciu o tom, ktoré ponuky už nemá v grafickom užívateľskom prostredí naďalej zobrazovať.

4.9 Metódy testovacieho stubu

Testovací stub v skutočnosti obsahoval aj niekoľko do budúcnosti použiteľných algoritmov. Z pôvodnej implementácie som sa v mojom riešení rozhodol ponechať metódy pre uloženie nových typov anotácií, uloženie predchodcov nového typu, ak neexistujú a vyhľadanie typu v databáze.

Kapitola 5

Implementácia

Kapitola sa zaoberá implementačnými detailmi práce. Opisuje riešenia významných problémov, s ktorými som sa stretol počas programovania navrhnutého modulu.

5.1 Vytvorenie JSON reťazca

Ako riešenie som zvolil skladanie výsledného JSON reťazca. Konštantné názvy položiek a hodnoty získané z dátovej štruktúry EditorSession (požiadavky klientskej časti systému) sú postupne pripájané ku objektu triedy StringBuilder. Nachádza sa v štandardnej knižnici jazyka Java. Ak spájame viac textových reťazcov, je výhodnejšie použiť StringBuilder namiesto spájania objektov typu String [15]. Vytváranie správy bude prehľadné a rýchle.

5.2 HTTP komunikácia

Pripravené správy v textovom JSON formáte odosiela modul pomocou metódy pomocnej triedy nlpConn. Mechanizmus v nej využíva rozhranie triedy štandardnej knižnice jazyka Java – HTTPUrlConnection. Pomocou bufferovaného vstupu dokáže odoslať aj textové reťazce veľkej dĺžky. Z bufferu následne prečíta odpoveď, ktorá môže byť ľubovolne dlhá.

5.3 Spracovanie JSON reťazca

S ponukami anotácií vo formáte JSON pracujú metódy tried JSONValue, JSONArray a JSONObject z knižnice JSON Simple [9]. Textový reťazec prevedú na dátovú štruktúru (anotáciu alebo zoznam anotácií), ktorou je možné v iteráciách prechádzať. JSON formát zapisuje informácie dvojicami názov – hodnota [5]. Extrakcia dát funguje tak, že definujeme názov položky, ktorej hodnotu chcem získať. Z hľadiska prehľadnosti zdrojového kódu najskôr získané hodnoty uložím do pomocných premenných, aby boli v čase, kedy sa vytvárajú dátové štruktúry anotačného serveru, skontrolované a pripravené.

5.4 Ukladanie hodnoty dôvery ponuky anotácie

Anotačný server obdrží v odpovedi od SEC Store API serveru aj číselnú hodnotu od 0 do 100 reprezentujúcu dôveryhodnosť ponuky. Tú je možné ku ponuke uložiť až v čase, kedy sa objekt anotácie zabalí do objektu triedy SuggestionLogEntry. Dovtedy musí byť

hodnota uložená v dočasnej dátovej štruktúre. Ako riešenie som zvolil objekt triedy HashMap zo štandardnej knižnice jazyka Java. Hodnotu kľúča nevyhľadáva postupným prechádzaním zoznamu, ale použije jej hash hodnotu [8]. Kľúčom v tabuľke je identifikátor ponuky anotácie. Ten sa na SEC Store API vygeneruje ako náhodný jednoznačný identifikátor podľa RFC 4122 [1].

5.5 Odkazy v atribútoch

K vytvorenie odkazu na anotáciu alebo ponuku cez atribút typu LinkedAnnotationAttribute dochádza hneď po volaní konštruktoru atribútu. Modul uloží identifikátor odkazovanej ponuky do dátovej štruktúry reprezentujúcej atribút a klientská časť dostane túto informáciu v XML správe. Ak sa jedná o odkaz na existujúcu anotáciu, prehľadá databázu a uloží jej referenciu.

5.6 Vnorené ponuky anotácií

Referencie vnorených anotácií môžu vzniknúť až v čase, kedy sú vytvorené všetky anotácie, ktoré budú súčasťou ponúk. Modul v cykle prechádza postupne celý zoznam nových anotácií, ktoré sa stanú ponukami, a doplní referencie na tieto objekty. Ponuky však môžu byť vytvorené iba z objektov, ktoré majú prázdnu položku `nestedIn`. Inak by dochádzalo k nechceným duplikáciám a vnorené ponuky by sa dali potvrdiť samostatne, čo je nežiadúce. Obrázok 5.1 znázorňuje správne prepojenie ponúk anotácií cez ich atribúty v grafickom užívateľskom prostredí.

5.7 Zmeny v iných častiach systému

Aby bolo možné pripojiť a správne používať modul pre prácu s anotáciami, musel som vykonať úpravy už existujúceho kódu niektorých tried.

Do súboru s definíciou triedy anotácie a základného atribútu pribudla metóda, ktorá tieto objekty serializovala do podoby JSON reťazce. Spolu s metódou XML serializácie uľahčujú prácu s anotáciami v iných triedach. Metóda pracujúca na rovnakom princípe musela byť taktiež pridaná do rozhrania atribútu. Každý atribút sa ukladá spolu s anotáciou, musí ho teda byť možné uložiť v inom formáte.

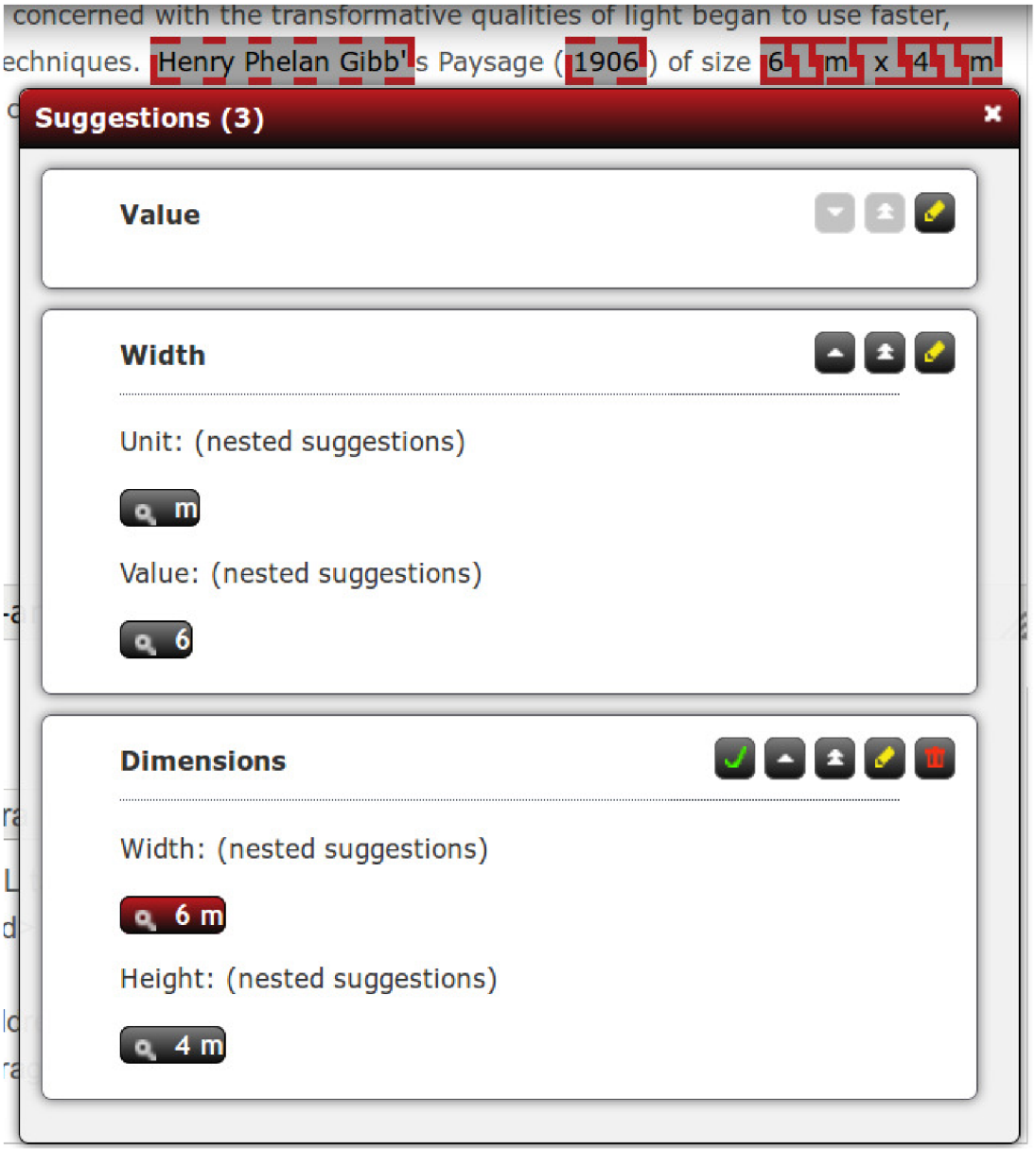
Pretože v pôvodnom testovacom stube generoval identifikátor čítač, mal numerickú hodnotu. SEC Store API však používa pre označenie ponuky náhodne generovaný reťazec znakov. Skladá sa teda aj z písmen abecedy. Dátový typ používaný doteraz už nebude vyhovovať, ale pracujú s ním viaceré moduly a metóda spracovania požiadavky. Úprava zasiahne väčšiu časť systému. Bolo dôležité správne identifikovať všetky miesta, kde mohlo dojsť ku incidentom spojených s nekompatibilitou typov. So zmenou dátového typu súvisí tiež rozdielne vzájomné porovnávanie hodnôt v prípade, kedy overujeme zhodu identifikátorov.

Po potvrdení ponuky anotácie ju metóda triedy `Persister` pretransformuje na anotáciu a pridá do zoznamu novovytvorených anotácií. Pôvodná ponuka zostáva v zozname potvrdených ponúk. V správe o potvrdení ponuky musí SEC Store API server obdržať adresu práve uloženej anotácie. Tá sa má nachádzať v položke objektu potvrdenej ponuky anotácie. Metóda triedy `Persister` však nevytvárala prepojenie medzi dvojicou potvrdená ponuka

[1] http://www.ietf.org/rfc/rfc4122.txt

a nová anotácia. Túto chybu bolo potrebné odstrániť. Po vytvorení objektu reprezentujúceho novú anotáciu som musel pridať kód pre vyhľadanie zhody dočasného identifikátoru `tmpId` anotácie a ponuky. Položka `ConfirmedVersion` teraz obsahuje referenciu na potvrdenú verziu ponuky, z ktorej je možné získať adresu uloženej ponuky.

Ďalšie opravy som musel vykonať v implementácii triedy `Linearizer`. Po zmenách iných častí systému, ktoré sa týkali práce s fragmentmi, bolo potrebné upraviť kontrolovanie pozícií jednotlivých fragmentov. V prípade anotácie celého dokumentu nie je definovaná začiatočná a koncová pozícia pozícia, s čím sa v pôvodnom návrhu nepočítalo. Pred získavanie hodnoty pozície v texte som pridal kontrolu, či položka má definovanú hodnotu. Úpravu vyžadovala aj metóda, ktorá transformovala zmeny v texte do ich linearizovanej podoby. Pre získanie textového reťazca fragmentu bez formátovacích značiek musí metóda pracovať s referenciou dokumentu, v ktorom sa fragment nachádza. Klientská aplikácia v súčasnej verzií nedokázala správne ukladať XPath fragmentu tak, aby sa zhodoval s XPath dokumentu. Dočasným riešením, ktoré funguje pre sadu dokumentov, s ktorými systém pri testovaní pracoval, bolo testovanie zhody podreťazca XPath. Skonvertovaný fragment následne vloží do zoznamu spracovaných fragmentov a poli príznakov poznačí, že bol spracovaný úspešne. Výsledkom procesu sú dva zoznamy fragmentov – úspešne a neúspešne spracovaných.



Obrázek 5.1: Štruktúra vnorených ponúk anotácií zobrazená v anotačnom editore

Kapitola 6

Testovanie

Kapitola pojednáva o metódach overovania správnosti navrhnutého a implementovaného riešenia. Popisuje postup testovania pomocou skriptov spúšťaných na lokálnej stanici, ako aj funkčnej verzií SEC Store API serveru.

Bolo potrebné zistiť, či odchádzajúce a prichádzajúce správy dodržujú požadovanú štruktúru. Poradie položiek JSON nekontroluje, žiadna z nich však nemôže chýbať. Táto chyba by sa objavila pri pokuse o načítanie textového reťazca pomocou metódy knižnice JSON.simple.

Vytvorený aj prijatý JSON reťazec musí mať správne vyplnené všetky položky, čo znamená, že musia korešpondovať s tým, čo je definované v komunikačnom protokole. Položky som kontroloval vizuálne v ladiacom móde integrovaného vývojového prostredia.

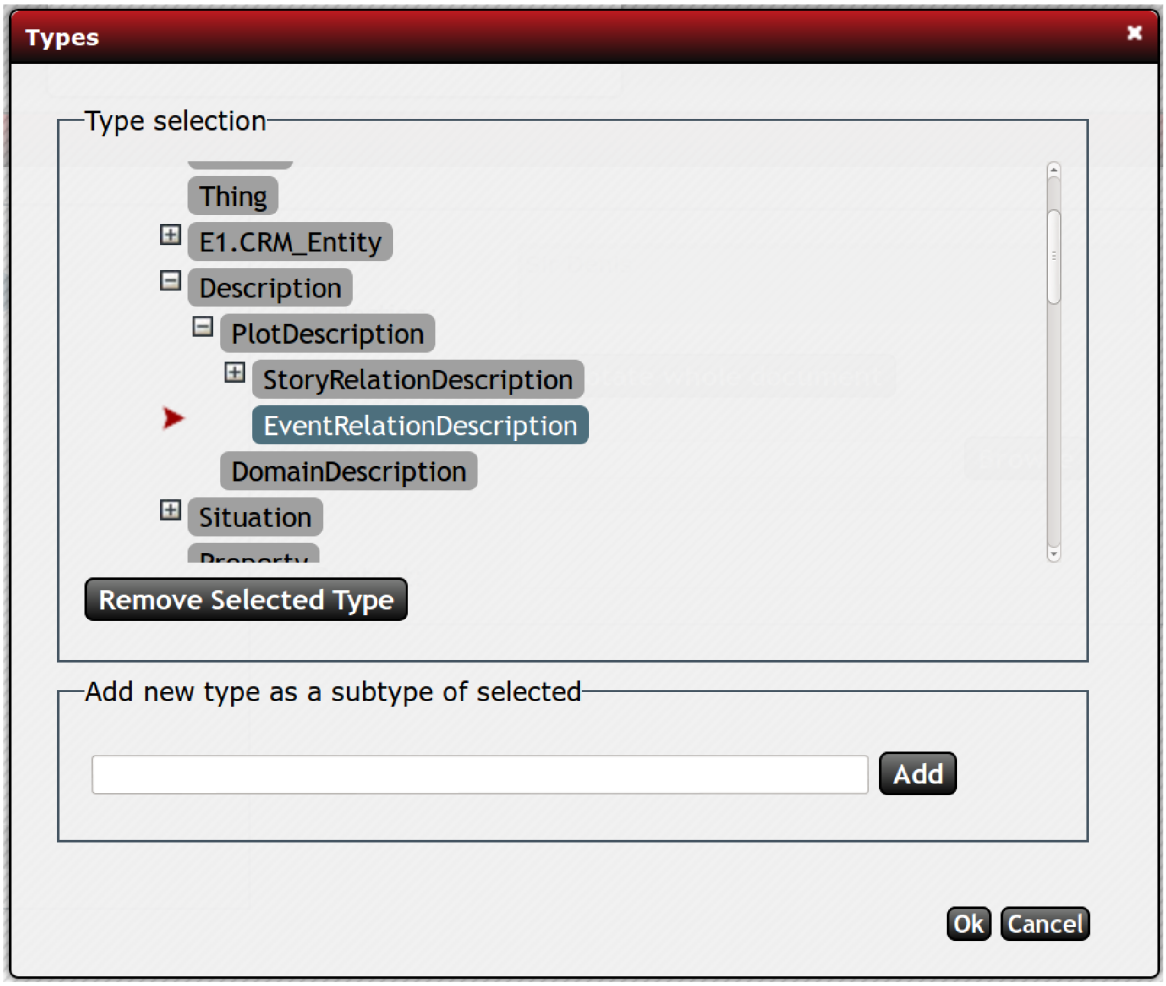
Požadovanie ponúk anotácií som overoval ako pre celý dokument, tak aj jeho časti. Preverili sa tým funkcie linearizéru textu, pretože SEC Store API pracuje s linearizovanou podobou dokumentu, ktorá neobsahuje formátovacie značky. Vyznačením fragmentu textu som musel získať nie len prislúchajúce ponuky, ale aj ich vnorené ponuky, ak existujú.

Testovanie sa týkalo aj ponúkania anotácií jedného zvoleného typu. Ak som v užívateľskom rozhraní vyplnil pole typ anotácie pri žiadosti o ponuky, nesmelo sa stať, aby prišli ponuky iných typov, ak sa nejedná o vnorené anotácie.

Dôležitou vecou je spracovanie typu prijatej ponuky anotácie. Testovanie tohto prípadu overí funkčnosť metód pre ukladanie typov do databázy, ako aj prípadné rozloženie typov na podtypy. Ak nadradené typy ešte u klienta neexistujú, uložia sa spolu s podradeným typom do databázy. K vytvoreniu štruktúry typu som použil možnosť pridávať anotácie iných užívateľov do zoznamu ponúknutých anotácií pre aktuálneho užívateľa metódou „store“. Anotácia vytvorená u jedného užívateľa v dokumente je pre iného ponúknutá s vyššou dôveryhodnosťou. Druhý užívateľ teda prijal ponuku anotácie, ktorej typ v jeho sedení ešte neexistuje. Po spracovaní rodičovských typov a typov ich potomka musí byť v užívateľskom rozhraní možné vytvoriť anotácie všetkých týchto typov. Štruktúru typov anotácií vidíme na obrázku 6.1.

Modul Suggestion synchronizer anotačného serveru musí dokázať spracovať všetky typy atribútov. Väčšina definovaných typov pracuje s textovým reťazcom alebo číselnou hodnotou. Atribút typu `GeoPoint` však pozostáva z dvoch hodnôt. Musí dojsť ku správnemu spracovaniu hodnoty prijatej zo SEC Store API, čo najjednoduchšie zistíme pomocou nahliadnutia na ponuku v editore anotácií.

Overoval som aj správne previazanie ponúk anotácií pomocou štruktúrovaných atribútov typu vnorená a odkazovaná anotácia. Pre testovanie som pripravili scenár, kedy vytvorím previazanie anotácií pomocou odkazov u iného užívateľa. SEC Store API ich zaradí



Obrázek 6.1: Štruktúrované typy anotácií v anotačnom editore

do zoznamu ponúk ku dokumentu. Prijaté ponuky nesmeli obsahovať prázdne atribúty bez odkazov. V grafickom anotačnom editore musí byť možné cez atribúty získať informácie o odkazovaných ponukách a zobraziť ich.

Pre zistenie správnej funkčnosti modifikácie textu som zvolil scenár, kedy som v texte urobil tri zmeny, každú jednu po odhlásení a vrátení zmien vykonaných pri predchádzajúcom pokuse: vymazanie slova pred ponúknutou anotáciou, vymazanie časti slova, ktoré je označené ako ponuka anotácie a vymazanie slova za ponukou. Sledoval som správanie jednej zvolenej ponuky. Výsledkom mal byť stav, kedy dôjde k odstráneniu ponuky, ktorej sa zmenila pozícia v texte a vytvorenie ponuky s novou platnou pozíciou. Odstránením časti textu, ktorý zasahuje do ponuky, musí dojsť ku vymazaniu tejto ponuky zo zoznamu zobrazených.

Po odhlásení užívateľa zo systému posiela anotačný server identifikátor sedenia na SEC Store API. V spolupráci s vývojárom SEC Store API Martinom Hořákom sme overovali, či je sedenia správne identifikované a jeho obsah sa korektne zmaže, aby v systéme nezostávali nedokončené sedenia.

6.1 Metodika testovania pomocou skriptov

Pre tento účel som vytvoril sadu testovacích skriptov. Nahrádzajú prítomnosť SEC Store API serveru a simulujú jeho funkcionalitu. Každý skript funguje samostatne a reprezentuje jeden z možných scenárov. Testovacie skripty som spúšťal na lokálnom stroji. Program prijímal požiadavky od anotačného serveru. Do terminálu, v ktorom je spustený, vypisuje prijaté dáta vo formáte JSON, kvôli možnosti kontroly, či server správne sformoval požiadavku. Validitu formátu overuje Python metóda `loads()`. Ak pri spracovaní reťazca vypadne výnimka, nejedná sa o správne formátovanú požiadavku. Aby som jednoduchšie našiel miesto, kde sa chyba nachádza, použil som online nástroj JSONLint [1]. Keď je formát správy v poriadku, odošle vopred pripravenú odpoveď anotačnému serveru. Dáta som získaval z ladiacich výpisov, kedy prichádzali odpovede od SEC Store API. Taktiež odchytením textového reťazca správy „store“, kedy bola v užívateľskom prostredí vytvorená nová anotácia, som mohol získať cenné dáta a nechať si ich v iteráciách lokálne posielať. Nebolo potrebné využívať výpočtové prostriedky SEC Store API server a správy som mohol modifikovať podľa potreby. Ďalšou výhodou tohoto typu testovania bola nezávislosť na ostatných kolegoch, ktorí by v prípade potreby museli upravovať kód funkcií na SEC Store API. Požiadavky z lokálneho sieťového rozhrania prichádzali podstatne rýchlejšie a znížili sa riziko straty spojenia medzi komunikujúcimi stranami.

Po správnom nadefinovaní testovacích prípadov a overení funkčnosti modulu na týchto dátach som mohol pristúpiť k overovaniu na funkčnej verzii SEC Store API serveru v spolupráci s jeho vývojármi.

6.2 Testovanie na funkčnej verzii

Testovacími skriptami nebolo možné nasimulovať kompletnú funkcionalitu a správanie SEC Store API serveru, preto bolo potrebné implementáciu otestovať v prostredí, ktoré pripomína reálnu prevádzku. Veľká časť testovania anotačného serveru prebiehala na serveroch výskumnej skupiny KNOT FIT VUT v Brně. Na nich bola spustená aktuálna funkčná verzia SEC Store API serveru s MongoDB databázou. Prichádzajúce a odchádzajúce dáta, chyby a hlásenia stavy boli spolu s časovými razítkami zapisované do logovacieho súboru, aby ich mohli kontrolovať všetci členovia, ktorí sa zúčastňovali testovania. Úpravy a ladenie SEC Store API serveru mal na starosti kolega Martin Hořák. Vedel o testovacích scenároch, ich požiadavkách a cieľoch. Sledoval prebiehajúce pokusy a kontroloval dáta uložené v databáze. Musel tiež vykonávať údržbu databázy, keď sa do nej dostali chybné dáta, ktoré znemožňovali správne testovanie. S jeho spoluprácou došlo ku korekciám komunikačného protokolu a odstráneniu chýb v module.

[1] http://www.jsonlint.com

Kapitola 7

Záver

V rámci mojej bakalárskej práce vznikol funkčný modul pre ponúkanie a správu anotácií projektu Deciher. Boli splnené všetky požiadavky vedúcich projektu.

V zimnom semestri som analyzoval existujúce časti projektu a pripravil návrh modulu spolu s technológiami. Vytvoril som komunikačný protokol, ktorý definuje, aké informácie si budú medzi sebou vymieňať anotačný a SEC Store API server. Bol navrhnutý na základe požiadaviek vedúcich projektu Decipher. Implementoval som aj prototyp modulu.

V letnom semestri zanikol testovací stub. Spracovanie textových dokumentov už naďalej neprebieha na anotačnom serveri. Prebrali ho nástroje SEC Store API serveru. Ten si s anotačným serverom vymieňa správy vo formáte popísanom v navrhnutom komunikačnom protokole. Pridaním podpory štruktúrovaných atribútov je možné vytvoriť prepojenie jednotlivých ponúk anotácií do ľubovolnej úrovne zanorenia. Potvrdenie a odmietnutie ponúk sleduje SEC Store API, pre každé sedenie udržuje zoznam ponúknutých anotácií. Zoznam modifikuje na základe správ, ktoré dostáva od anotačného serveru vždy, keď užívateľ nejakým spôsobom interaguje s ponukami. Ak sa spojenie nepodarí vytvoriť, užívateľ je informovaný a zobrazené ponuky sa mu už naďalej v grafickom prostredí nezobrazujú. Došlo aj ku modifikácií existujúcich súborov definícií tried, kde pribudla serializácia ich dát do formátu JSON. Pre uľahčenie ladenia ostatných modulov, ako aj nástrojov SEC Store API sa všetky chybové hlásenia a varovania týkajúce sa ponúk vypisujú do záznamu GlassFish serveru. Modul, ako súčasť 4A serveru, sa používa v testovacej prevádzke v jednom z dublinských múzeí. Do budúcnosti sú naplánované vylepšenia podpory nových typov atribútov.

Literatura

[1] *decipher-research [online]*. Decipher Project, 2007, [Online; navštíveno 28.4.2013].
URL http://decipher-research.eu/

[2] *4A Framework (Annotations Anywhere, Annotations Anytime) created in NLP@FIT [online]*. Faculty of Information Technology BUT, 2011, [Online; navštíveno 20.6.2012].
URL http://knot.fit.vutbr.cz/annotations/

[3] *GlassFish Product Documentation [online]*. GlassFish Comunity, 2012, [Online; navštíveno 3.5.2013].
URL http://glassfish.java.net/docs/index.html

[4] Alinone, A.: *10 Years of Push Technology, Comet, and WebSockets [online]*. 2011, [Online; navštíveno 23.4.2013].
URL http://cometdaily.com/2011/07/06/push-technology-comet-and-websockets-10-years-of-history-from\-lightstreamers-perspective/

[5] Crockford, D.: *The application/json Media Type for JavaScript Object Notation (JSON) [online]*. 2006, [Online; navštíveno 25.4.2013].
URL http://www.ietf.org/rfc/rfc4627.txt

[6] Crockford, D.: *JavaScript: The Good Parts*. O'Reilly, 2008, ISBN 978-0-596-51774-8.

[7] Dykes, L.; Ullman, C.: *Beginning Ajax*. Wrox, 2007, ISBN 978-0-470-10675-4.

[8] Eckel, B.: *Thinking in Java*. Prentice Hall, 2006, ISBN 0-13-187248-6.

[9] Fang, Y.: *DecodingExamples [online]*. 2010, [Online; navštíveno 1.7.2012].
URL http://code.google.com/p/json-simple/wiki/DecodingExamples

[10] Fang, Y.: *JSON.simple - A simple Java toolkit for JSON [online]*. 2010, [Online; navštíveno 1.7.2012].
URL http://code.google.com/p/json-simple/

[11] Fielding, R.; et al.: *Hypertext Transfer Protocol – HTTP/1.1 [online]*. 1999, [Online; navštíveno 21.6.2012].
URL http://tools.ietf.org/html/rfc2616

[12] Garrett, J. J.: *Ajax: A New Approach to Web Applications [online]*. 2005, [Online; navštíveno 23.4.2013].
URL
http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications

[13] Gosling, J.; et al.: *The Java Language Specification Third Edition*. ADDISON-WESLEY, 2005, ISBN 0-321-24678-0.

[14] Heiss, J. J.: *Arun Gupta on Higher Productivity from Embracing HTML5 with Java EE 7 [online]*. 2013, [Online; navštíveno 25.4.2013].
URL http://www.oracle.com/technetwork/articles/java/gupta-1911481.html

[15] Oracle: *The StringBuilder Class [online]*. 1995, [Online; navštíveno 1.7.2012].
URL http://docs.oracle.com/javase/tutorial/java/data/buffers.html

[16] Oracle: *Oracle GlassFish Server 3.1 Application Development Guide [online]*. 2010, 2011, [Online; navštíveno 23.4.2013].
URL http://docs.oracle.com/cd/E18930_01/html/821-2418/ggrgt.html#ggrgy

[17] Oracle: *HttpUrlConnection (Java Platform SE 6) [online]*. 2011, [Online; navštíveno 22.6.2012].
URL http://docs.oracle.com/javase/6/docs/api/java/net/HttpURLConnection.html

[18] Oracle: *Java EE at a Glance [online]*. 2012, [Online; navštíveno 3.5.2013].
URL http://www.oracle.com/us/technologies/java/enterprise-edition/overview/index.html

[19] Russell, A.: *Comet: Low Latency Data for the Browser [online]*. 2006, [Online; navštíveno 18.4.2013].
URL http://infrequently.org/2006/03/comet-low-latency-data-for-the-browser/

[20] Shiflett, C.: *HTTP Developer's Handbook*. Sams Publishing, 2003, ISBN 0-672-32454-7.

[21] W3C: *Extensible Markup Language (XML) 1.0 (Fifth Edition) [online]*. 2008, [Online; navštíveno 24.4.2013].
URL http://www.w3.org/TR/2008/REC-xml-20081126/

[22] W3C: *Extensible Markup Language (XML) 1.1 (Second Edition) [online]*. 2008, [Online; navštíveno 24.4.2013].
URL http://www.w3.org/TR/xml11/

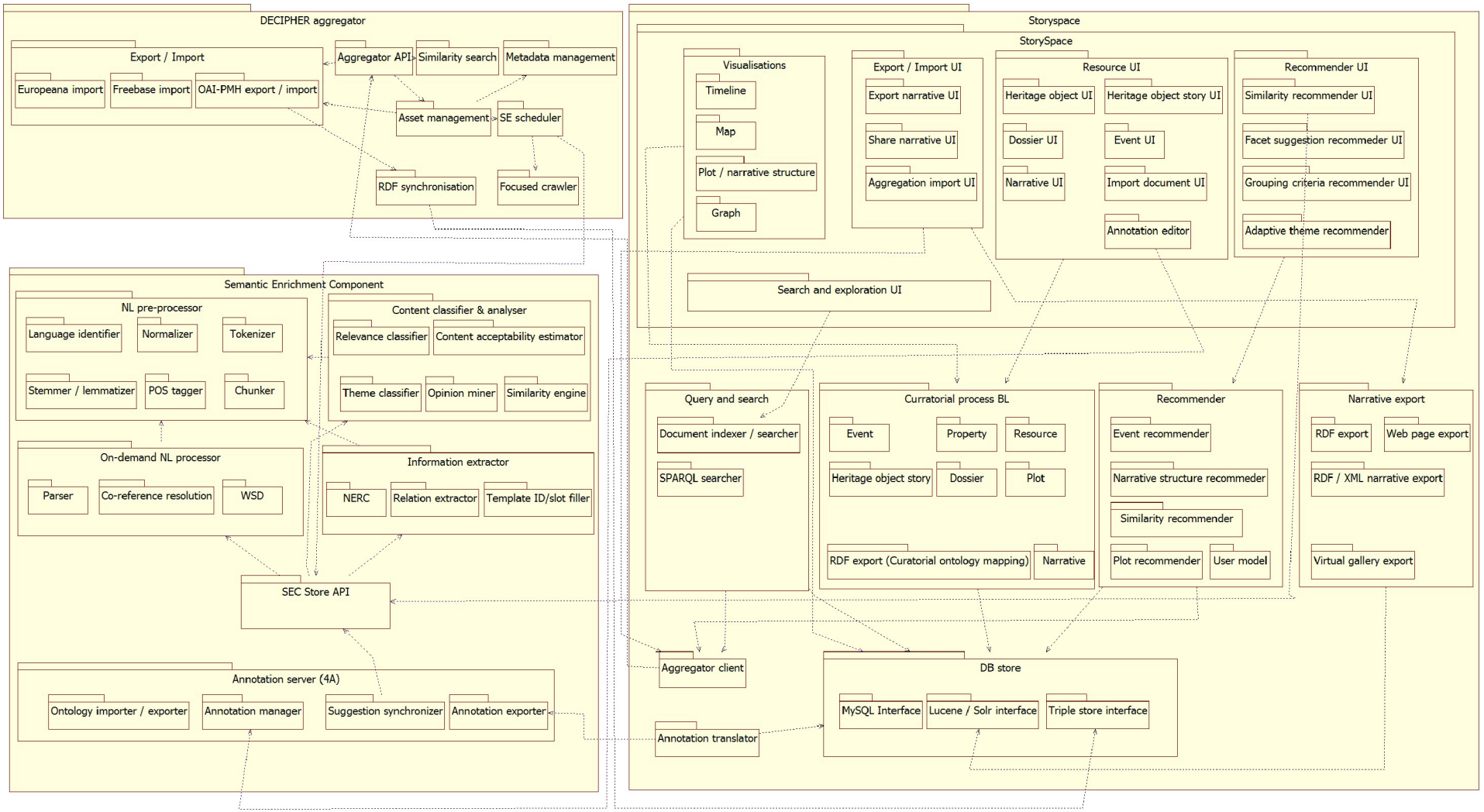
[23] Walsh, N.: *A Technical Introduction to XML [online]*. 1998, [Online; navštíveno 24.4.2013].
URL http://www.xml.com/pub/a/98/10/guide0.html?page=2#AEN58

Přílohy

Seznam příloh

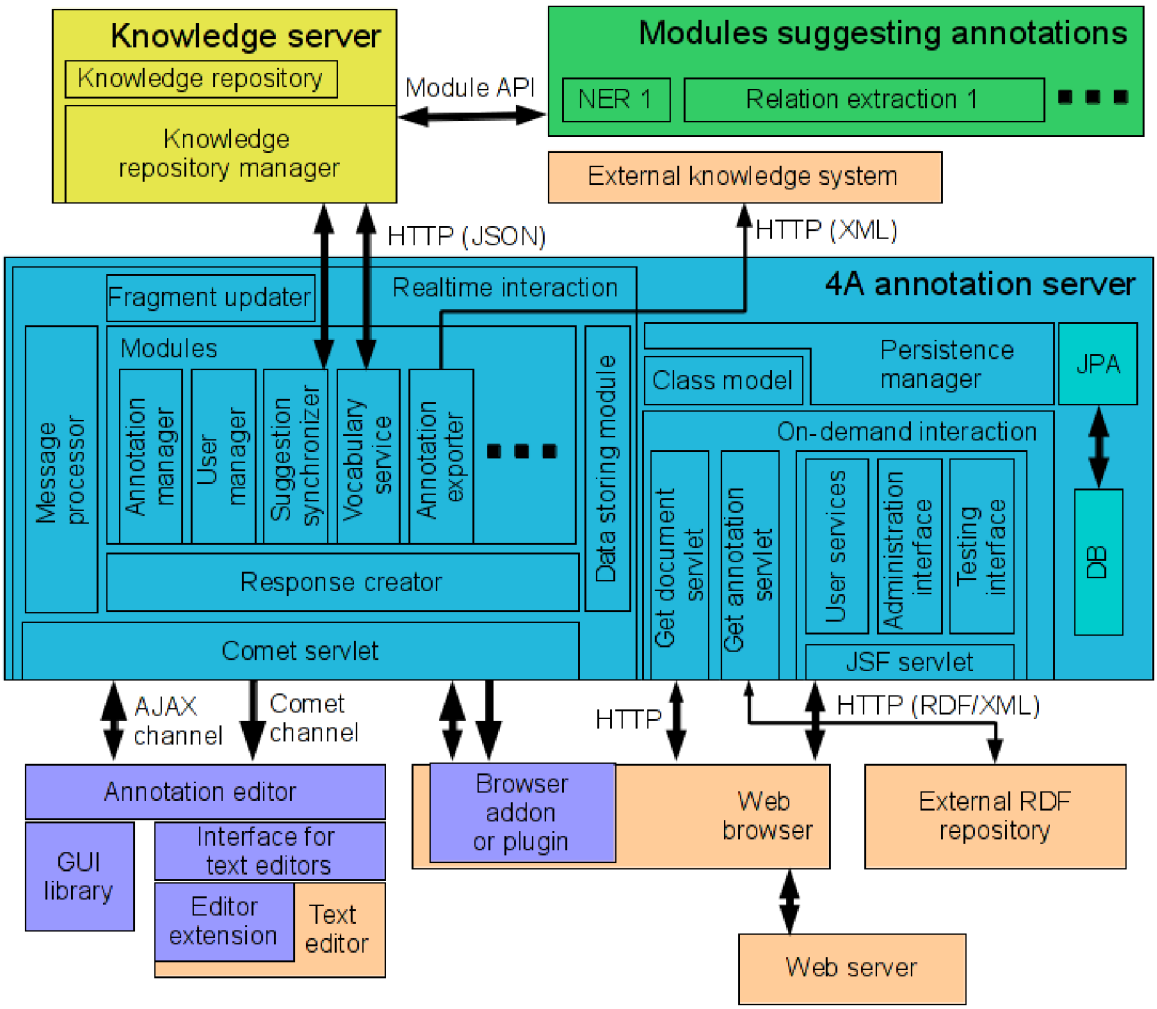
Příloha A

Architektúra projektu DECIPHER


DECIPHER aggregator
Export / Import
Europeana import
Freebase import
OAI-PMH export / import
Aggregator API
Similarity search
Metadata management
Asset management
SE scheduler
RDF synchronisation
Focused crawler
Storyspace
StorySpace
Visualisations
Timeline
Map
Plot / narrative structure
Graph
Export / Import UI
Export narrative UI
Share narrative UI
Aggregation import UI
Resource UI
Heritage object UI
Heritage object story UI
Dossier UI
Event UI
Narrative UI
Import document UI
Annotation editor
Recommender UI
Similarity recommender UI
Facet suggestion recommeder UI
Grouping criteria recommender UI
Adaptive theme recommender
Search and exploration UI
Semantic Enrichment Component
NL pre-processor
Language identifier
Normalizer
Tokenizer
Stemmer / lemmatizer
POS tagger
Chunker
Content classifier & analyser
Relevance classifier
Content acceptability estimator
Theme classifier
Opinion miner
Similarity engine
On-demand NL processor
Parser
Co-reference resolution
WSD
Information extractor
NERC
Relation extractor
Template ID/slot filler
SEC Store API
Annotation server (4A)
Ontology importer / exporter
Annotation manager
Suggestion synchronizer
Annotation exporter
Query and search
Document indexer / searcher
SPARQL searcher
Curratorial process BL
Event
Property
Resource
Heritage object story
Dossier
Plot
RDF export (Curatorial ontology mapping)
Narrative
Recommender
Event recommender
Narrative structure recommeder
Similarity recommender
Plot recommender
User model
Narrative export
RDF export
Web page export
RDF / XML narrative export
Virtual gallery export
Aggregator client
Annotation translator
DB store
MySQL Interface
Lucene / Solr interface
Triple store interface

Príloha B

Architektúra anotačného systému v projekte DECIPHER



Příloha C

Komunikačný protokol

4A server <---> SEC Store API server communication protocol

Table of content

1 Introduction

The document describes an application communication protocol between the Annotation server and the SEC Store API server in the Decipher Project. Data are carried in the JSON format and are encapsulated in HTTP protocol messages. The SEC Store API server expects identification of the client server. In case of the Annotation server, the header field, named `Client`, contains an identification string `4A`. The messages contain a request or an answer to the processed JSON data request.

Next two chapters define and explain the meaning of JSON data messages and their respective fields. There is always an example of included.

2 Requests

2.1 Get Suggestions

When a user requests annotation suggestions, the SEC Store API has to send all suggestions that belong to the document and are not yet displayed in the user's session.

The method is identified by the keyword '*getsugg*' as a name of the first level element of the JSON. For a suggestion request for the whole document, values of the `startOffset` and the `endOffset` are set to `-1`. When the user turns the suggestions in his session off, both values are `0`.

- `synchronization` – flag represents document change on the client side, values 0 and 1
- `resynchronization` – when the value is 1, the user needs to get all the suggestions from his session again
- `startOffset` – start position of the fragment for suggestions in the document
- `endOffset` – end position of the fragment for suggestions in the document
- `sessId` – identification number of user's current session
- `authorGroups` – identification of groups user belongs to (`URI` and the `name` of the group)
- `annotTypeGroup` – URI of the group to which the requested type belongs
- `annotTypePath` – path of the type in the tree of types, includes names of parent types delimited by the slash sign
- `typeUriInOntology` – URI of the type in the ontology

The following fields provide information about a document that the user works with:

- `annotSrvUri` – URI of the annotation server where the document is stored
- `docContent` – field `change` tells the SEC Store API that new content of the document identified by the `sourceUri` field is present in the element named `linText`

Syntax

```
{
     "getsugg": {
        "synchronization":,
        "resynchronization":,
        "startOffset":,
        "endOffset":,
        "sessId":,
        "authorGroups": [
          {
            "uri": "",
            "name":""
          },
          {
            "uri": "",
            "name":""
          }
        ],
        "annotTypeGroup": "",
        "annotTypePath": "",
        "typeUriInOntology": ""
    },
    "annotSrvUri":"",
    "docContent":{
        "change":,
        "sourceUri": "",
        "linText":""
    }
}
```

Example

```
{
    "getsugg": {
        "synchronization":0,
        "resynchronization":0,
        "startOffset": 1,
        "endOffset": 12,
        "sessId": 1,
        "authorGroups": [
          {
            "uri": "http://knot09.fit.vutbr.cz:8080/Annotations/groups/1",
            "name":"administrators"
          },
          {
            "uri": "http://knot09.fit.vutbr.cz:8080/Annotations/groups/2",
            "name":"testers"
          }
        ],
        "annotTypeGroup": "http://knot09.fit.vutbr.cz:8080/Annotations/groups/2",
        "annotTypePath": "E1.CRM_Entity/E77.Persistent_Item",
        "typeUriInOntology":
"http://www.cidoc-crm.org/rdfs/cidoc_crm_v5.0.2_english_label.rdfs#E77.Persistent_Item"
    },
    "annotSrvUri":"http://knot09.fit.vutbr.cz:8080",
    "docContent":{
        "change":0,
        "sourceUri":
"http://localhost/storyscope7/casecid/5b27c09e-8041-11e2-8753-002590937d1c",
        "linText":"Denis Mahon lived in …"
    }
}
```

2.2 Store Annotations

When the user creates new annotation in the document, its data are sent to the SEC Store API. Another user who asks for annotation suggestions for the same document gets this annotation with higher confidence level. All of the fields mentioned are required to reconstruct the annotation on the Annotation server. The message is identified by the name `store` of the top level element of JSON.

- `startOffset/endOffset` – start/end position of the annotation in the linearized document
- `uri` – URI of the annotation stored on the Annotation server
- `annotTypePath` – path of the type in the tree of types, includes names of parent types
- `typeUriInOntology` – URI of the type in the ontology
- `authorGroups` – group(s) that the author belongs to
- `annotTypeGroup` – URI of the group which the type of the stored annotation belongs to
- `sourceUri` – URI of the document in which the annotation was created, contains `casecid`
- `content` – additional information provided by the user
- `annoText` – textual content of the selected part of the document
- `attributes` – list of an annotation's attributes, each attribute consists of following fields:
  - `attName` – name of the attributes
  - `simpleType` – type of the attribute identified by the keyword, determines the attribute's properties
  - `nestedUri/linkedUri` – URI of the nested/linked annotation
  - `nestedId/linkedId` – ID of the nested/linked suggestion
  - `structuredType` – type path of the nested/linked annotation/suggestion
- `nestedIn` – URI of the parent annotation if the annotation is nested

Syntax:

```
{
  "store": [
    {
      "annotation": {
        "startOffset":,
        "endOffset":,
        "uri": "",
        "annotTypePath": "",
        "typeUriInOntology": "",
        "authorGroups": [
          {
            "uri": "",
            "name":""
          },

        ],
        "annotTypeGroup": "",
        "sourceUri": "",
        "content": "",
        "annoText": "",
        "attributes": [
          {
            "attName":"",
            "simpleType":"",
            "attValue":"",
            "nestedURI":"",
            "nestedID":"",
            "linkedURI":"",
            "linkedID":"",
            "structuredType":
          }
        ],
        "nestedIn": ""
      }
    }
  ]
}
```

Example:


```
{
  "store": [
    {
      "annotation": {
        "startOffset":1,
        "endOffset":10,
        "uri": "http://knot09.fit.vutbr.cz:8080/Annotations/annotations/12",
        "annotTypePath": "E1.CRM_Entity/E77.Persistent_Item",
        "typeUriInOntology":
"http://www.cidoc-crm.org/rdfs/cidoc_crm_v5.0.2_english_label.rdfs#E77.Persistent_Item",
        "authorGroups": [
          {
            "uri": "http://knot09.fit.vutbr.cz:8080/Annotations/groups/1",
            "name":"administrators"
          },
          {
            "uri": "http://knot09.fit.vutbr.cz:8080/Annotations/groups/2",
            "name":"testers"
          }
        ],
        "annotTypeGroup": "http://knot09.fit.vutbr.cz:8080/Annotations/groups/2",
        "sourceUri":
"http://localhost/storyscope7/casecid/5b27c09e-8041-11e2-8753-002590937d1c",
        "content": "",
        "annoText": "Denis Mahon",
        "attributes": [
          {
            "attName":"freebase",
            "simpleType":"URI",
            "attValue":"m/0gt0sg",
            "nestedURI":"null",
            "nestedID":null,
            "linkedURI":"null",
            "linkedID":null,
            "structuredType":null
          },
          {
            "attName":"InfluencedBy",
            "simpleType":"NestedAnnotation",
            "attValue":"",
            "nestedURI": "http://knot09.fit.vutbr.cz:8080/Annotations/annotations/333",
            "nestedID":null,
            "linkedURI":"null",
            "linkedID":null,
            "structuredType":null
          }
        ],
        "nestedIn": "http://knot09.fit.vutbr.cz:8080/Annotations/annotations/5"
      }
    }
  ]
}
```

2.3 Delete Annotations

When the user removes one of the annotations, a message with information about this event must be sent to the SEC Store API server. The message is identified by the `delete` name of the element on the top level in JSON.

- `uri` – URI of the annotation on the Annotation server
- `authorGroups` – list of groups the user belongs to
  - `uri` – URI of the group from the Annotation server
  - `name` – name of the group
- `annotTypeGroup` – URI of the group to which the type of deleted annotation belongs

Syntax

```
{
  "delete": [
    {
      "uri": "",
      "authorGroups": [
          {
            "uri": "",
            "name":""
          }
      ],
      "annotTypeGroup": ""
    }
  ]
}
```

Example

```
{
  "delete": [
    {
      "uri": "http://knot09.fit.vutbr.cz:8080/Annotations/annotations/12",
      "authorGroups": [
          {
            "uri": "http://knot09.fit.vutbr.cz:8080/Annotations/groups/1",
            "name":"administrators"
          },
          {
            "uri": "http://knot09.fit.vutbr.cz:8080/Annotations/groups/2",
            "name":"testers"
          }
      ],
      "annotTypeGroup": "http://knot09.fit.vutbr.cz:8080/Annotations/groups/2"
    }
  ]
}
```

2.4 Confirm / Refuse Suggestions

After the user responds to any of the suggested annotations, the SEC Store API server must be informed about this act. There has to be an identification of the suggestion sent along with the response method and information about the user. The message is identified by the name `suggFeedback` of the top level element of JSON.

- `value` – type of response, `conf` for confirmation an `ref` for refuse
- `id` – identification of the suggestion
- `annotSrvUri` – URI of the annotation server the feedback comes from
- `uri` – URI of the confirmed version of the suggestion – newly created annotation
- `method` – information about the response method, whether the user reacted (value 0 for both manual confirm and manual refuse), suggestion was modified before the user confirmed it or it was refused automatically (value 1), the suggestion was confirmed automatically (value 2) or automatically with editation (value 3)

- `session` – identification of the user's session in which the reaction took place
- `groupUri` – URI of the group which contains a type of the suggestion
- `annotation` – data of the modified suggestion in case the suggestion was confirmed by the Modify and Confirm method
  (for annotation data format see 2.2 Store Annotations)

Syntax:

```
{
  "suggFeedback": [
    {
      "value": "",
      "id":"",
      "annotSrvUri": "",
      "uri": "",
      "method":,
      "session":,
      "groupUri": "",
      "annotation": ""
    }
  ]
}
```

Example:


```
{
  "suggFeedback": [
    {
      "value": "conf/ref",
      "id": 15,
      "annotSrvUri": "http://knot09.fit.vutbr.cz:8080",
      "uri": "http://knot09.fit.vutbr.cz:8080/Annotations/annotations/123",
      "method": 0,
      "session": 234,
      "groupUri": "http://knot09.fit.vutbr.cz:8080/Annotations/groups/1",
      "annotation": "null"
    },
    {
      "value": "conf/ref",
      "id": 15,
      "annotSrvUri": "http://knot09.fit.vutbr.cz:8080",
      "uri": "http://knot09.fit.vutbr.cz:8080/Annotations/annotations/123",
      "method": 1,
      "session": 234,
      "groupUri": "http://knot09.fit.vutbr.cz:8080/Annotations/groups/2",
      "annotation": {
        "startOffset":1,
        "endOffset":10,
        "uri": "http://knot09.fit.vutbr.cz:8080/Annotations/annotations/12",
        "annotTypePath": "E1.CRM_Entity/E77.Persistent_Item",
        "typeUriInOntology":
"http://www.cidoc-crm.org/rdfs/cidoc_crm_v5.0.2_english_label.rdfs#E77.Persistent_Item",
        "authorGroups": [
          { "uri": "http://knot09.fit.vutbr.cz:8080/Annotations/groups/1",
            "name":"administrators"
          },
          { "uri": "http://knot09.fit.vutbr.cz:8080/Annotations/groups/2",
            "name":"testers"
          }
        ],
        "annotTypeGroup": "http://knot09.fit.vutbr.cz:8080/Annotations/groups/2",
        "sourceUri":
"http://localhost/storyscope7/casecid/5b27c09e-8041-11e2-8753-002590937d1c",
        "content": "",
        "annoText": "Denis Mahon",
        "attributes": [
          {
            "attName":"freebase",
            "simpleType":"URI",
            "attValue":"m/0gt0sg",
            "nestedURI":"null",
            "nestedID":null,
            "linkedURI":"null",
            "linkedID":null,
            "structuredType":null
          },
          {
            "attName":"InfluencedBy",
            "simpleType":"NestedAnnotation",
            "attValue":"",
            "nestedURI": "http://knot09.fit.vutbr.cz:8080/Annotations/annotations/333",
            "nestedID":null,
            "linkedURI":"null",
            "linkedID":null,
            "structuredType":null
          }
        ],
        "nestedIn": "http://knot09.fit.vutbr.cz:8080/Annotations/annotations/5"
      }
    }
  ]
}
```

2.5 Text Modifications

The message informs the SEC Store API server about the document's text modifications. Every change in the document must be applied on the document stored on the SEC Store API and the list of annotation suggestions must be processed so that offsets of the suggestions match a new document content.

The message is identified by the name `textModif` of the top level element of JSON.

- `annotSrvUri` - URI of the annotation server where the modified document is stored
- `sourceUri` - URI of the modified document
- `startOffset` – position of the beginning of the changed fragment
- `modifications` – list of modifications data
- `startOffset/endOffset` –start/end position of the change in the document
- `newContent` – textual data of the change

Syntax

```
{
  "textModif":{
    "annotSrvUri":"",
    "sourceUri": "",
    "modifications":[
      {
        "startOffset":,
        "endOffset":,
        "newContent":""
      }
    ]
  }
}
```

Example

```
{
  "textModif":{
    "annotSrvUri":"http://knot09.fit.vutbr.cz:8080",
    "sourceUri":
"http://localhost/storyscope7/casecid/5b27c09e-8041-11e2-8753-002590937d1c",
    "modifications":[
      {
        "startOffset":0,
        "endOffset":436,
        "newContent":"Robert Ballagh was born in Dublin in 1943. He studied architecture
the Dublin Institute of Technology and began painting in the 1960's without any formal
training. He was influenced by Pop Art and his work developed towards photo-realism in
the late 1970's. A large body of his work comprises portraits of well known twentieth
century figures. A versatile artist, his work also encompasses graphic design, stage
design and illustration."
      }
    ]
  }
}
```

2.6 Logout

After the user logs out of the system, his suggestions session has to be cleared from the SEC Store API server. The message is identified by the name `logout` of the top level element of JSON.

- `annotSrvUri` – the Annotation server where the user comes from
- `sessId` – ID of the ended session

Syntax

```
{
  "logout":{
    "annotSrvUri":"",
    "sessId":
  }
}
```

Example

```
{
  "logout":{
    "annotSrvUri":"http://knot09.fit.vutbr.cz:8080",
    "sessId":1
  }
}
```

3 Response

For every request sent from the Annotation server, the SEC Store API generates a response message. This message consists of two lists:

- `delete` – list of identificators of suggestions to be removed from the Annotation server
- `create` – list of suggestions to be created on the Annotation server
  - the suggestion data, compared to the annotation data, uses ID for identification, not URI

Syntax:

```
{
  "delete":[
    {
      "id":""
    }
  ],
  "create":[
    {
      "annotation": {
        suggestion data
        "confidence":
      }
    }
  ]
}
```

Example:


```
{
  "delete":[
    {
      "id": 1
    },
    {
      "id": 2
    }
  ],
  "create":[
    {
      "annotation": {
        "content": "",
        "sourceUri":
"http://localhost/storyscope7/casecid/bddb44ce-849a-11e2-8753-002590937d1c",
        "nestedIn": "null",
        "annoText": "Robert Ballagh",
        "annotTypePath": "/visual_art/visual_artist",
        "authorGroups": [
          {
            "uri": "http://knot09.fit.vutbr.cz:8080/Annotations/groups/1",
            "name":"administrators"
          }
        ],
        "attributes": [
          {
            "attValue": "freebase",
            "nestedURI":"null",
            "nestedID":null,
            "linkedURI":"null",
            "linkedID":null,
            "simpleType": "String",
            "structuredType": "null",
            "attName": "Source"
          },
          {
            "attValue": "/en/robert_ballagh",
            "nestedURI":"null",
            "nestedID":null,
            "linkedURI":"null",
            "linkedID":null,
            "simpleType": "URI",
            "structuredType": "null",
            "attName": "freebase"
          }
        ],
        "id": "c864e832-a79a-11e2-a0b8-002590937d1c",
        "startOffset": 0,
        "endOffset": 14,
        "confidence": 95,
      }
    }
  ]
}
```