SLOVENSKÁ TECHNICKÁ
UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY
A INFORMAČNÝCH TECHNOLÓGIÍ

Tím 15

DOKUMENTÁCIA K INŽINIERSKEMU DIELU

Tímový projekt

Študijný odbor: Informačné systémy, Softvérové inžinierstvo
Miesto vypracovania: Ústav informatiky a softvérového inžinierstva, FIIT STU Bratislava
Vedúca tímu: Ing. Nadežda Andrejčíková, PhD.

December 2014

Obsah

Kapitola 1

Big picture

Úvod

Veda je nepochybne oblasťou, kde je potrebné spracovávať obrovské množstvo informácií. Navyše ak je takého spracovávanie realizované v heterogénnom prostredí je tento proces oveľa náročnejší. Nie len pre výskumníka je dôležité získať informácie a všeobecný pohľad na niektoré diela. V existujúcich citačných indexoch sa nachádzajú bibliografické dáta, ktoré slúžia ako zdroj získavania ohlasov k jednotlivým publikáciám. Ak z takýchto zdrojov získame informácie, vhodne ich analyzujeme a poprepájame, môžu s veľkou mierou uľahčiť výskumníkovi zdĺhavú prácu pri získavaní ohlasov najmä na vlastné diela.

Navrhovaný systém bude teda identifikovať a získavať zdroje z iných externých systémov – primárne z WOSu a SCOPUSu - analyzovať takto získané zdroje, ukladať získavané dáta a analyzovať ohlasy k niektorým vybraným publikáciám. Systém bude slúžiť aj ako uložisko takzvaných fulltextov.

Vybudovaním vlastnej databázy bude systém možné postupne rozširovať na základe prichádzajúcich požiadaviek od používateľov. Takto má systém potenciál byť rozšírený o ďalšie funkcionality. V snahe zabrániť multiplicite, budú duplikované dáta ignorované. Použitím vlastnej lokálnej databázy bude systém rýchlejšie pristupovať k dokumentom, ako by to bolo v prípade dopytovania sa do citačných indexov pri prijatí nových požiadaviek. Systém bude implementovaný ako webová aplikácia poskytujúca webovú službu REST, ktorá bude komunikovať s klientom pomocou JSON formátu. Poskytnutím webovej služby umožníme jednoduché rozširovanie systému o ďalších klientov, alebo jednoduché poskytovanie služieb systému ďalším systémov ako API.

Ciele na zimný semester

Získavanie a vyhľadávanie ohlasov

Najdôležitejším cieľom pre náš projekt počas zimného semestra, je umožniť používateľom systému, vyhľadávať dokumenty v citačných indexoch WOS a SCOPUS. Pre niektoré alebo všetky vyhľadané diela získať ohlasy a takýmto spôsobom prezentovať používateľovi ohlasy na diela z viacerých systémov.

Deduplikácia dát

Dôležitou a neoddeliteľnou časťou je riešenie multiplicity dát, keďže sa môže jednoducho stať, že sa minimálne v dvoch citačných databázach nachádzajú rovnaké diela a ohlasy. Ak by sme neidentifikovali a neriešili problém multiplicity, mohli by umelo zvýšiť počet ohlasov a vrátiť nesprávne výsledky.

Používateľské prostredie a API

Ďalším cieľom je poskytnúť používateľovi rozhranie, ktorým jednoducho odošle požiadavku. Z dôvodu, že používateľ nie je povinný ovládať komunikačný formát výmenu dát (JSON), je potrebné implementovať webové rozhranie s jednoduchými interaktívnymi prvkami. Pre iné systémy bude systém poskytovať API REST rozhranie, komunikujúce pomocou formátu JSON.

Systém

Funkcionálne požiadavky

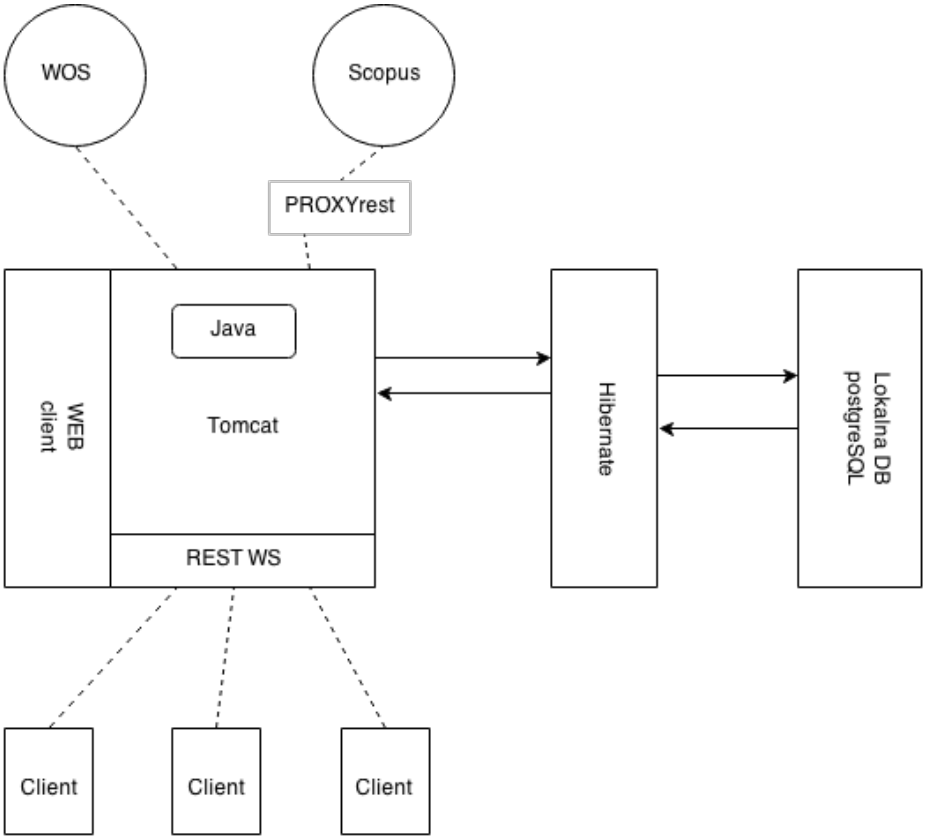
Pre systém sú zadefinované hlavné funkcionálne požiadavky, ktorými sú:

- Webová aplikácia
- Rozšíriteľnosť systému s vlastnou databázou
- Poskytovanie API pre iné systémy
- Jednoduché grafické webové prostredie

Podrobnejšie sú tieto požiadavky rozobrané spolu s opisom architektúry systému.

Ako sme už spomenuli v úvode, systém je navrhovaný ako webová aplikácia v podobe balíčku WAR. Na obrázku nižšie sa nachádza znázornená architektúra systému.

Základný servlet je nasadený na serveri Tomcat verzie 7. Servlet je implementovaný ako webová služba REST, ktorá komunikuje na základe protokolu HTTP, so štandardnými HTTP metódami GET a POST. Ako lokálna databáza je použitá postgresql, do ktorej sa pristupuje pomocou objektovo relačného mapovača HIBERNATE. Logika servletu je implementovaný v objektovo orientovanom jazyku JAVA, a teda pomocou JPA a HIBERNATE sa vytvárané objekty serializujú do databázy.



Obr. 1.1: Logický model

Prístup do citačných indexov WOS a Scopus je realizovaný štandardne prostredníctvom jednoduchých HttpClient-ov. WOS poskytuje svoje služby prostredníctvom api, ktorá komunikuje na základe SOAP správ v xml formáte(SOAP WS/ wsdl). Rovnako výsledky dopytov vracia v podobe xml Soap správ. Scopus poskytuje svoje api prostredníctvom REST webovej služby a výsledky dopytov vracia vo formáte JSON respektíve XML. Prístup do týchto citačných databáz je možný len z IP adries, ktoré majú k nim povolený prístup. Webová aplikácia je nasadená na serveri, ktorý má IP prístup do WOS-u, no pre Scopus je použitá proxyRest služba, ktorej IP adresa má povolený prístup.

WebClient je jednoduché grafické používateľské rozhranie, ktoré umožní vizualizovať výsledky zo systému vo formáte JSON a neskôr aj v zoznamoch. Na obrázku môžeme vidieť, že systém je jednoducho rozšíriteľný ďalšími klientmi, ktorý pristupujú cez REST WS.

Dátové modely

Mapovanie entít:

nasa databaza	WOS	Scopus
nazov_dokumentu	TI - Document title	Title
zdroj (len WOS/SCOPUS)	WOS	Scopus
zdroj_id	UT - Accession number	EID
typ dokumentu	PT - Publication Type*	Document type
autori	AU - Authors	Authors

data - ulozit cely zaznam o dokumente ako text.....pre wos aj scopus *(J=Journal; B=Book; S=Series; P=Patent)

1.1 Globálne ciele

Identifikovali sme nasledovné ciele, ktoré chceme dosiahnuť v rámci riešenia projektu počas zimného semestra:

1. **Oboznámenie sa s problematikou**
   Pred začatím práce na projekt je oblasť projektu pre väčšinu členov tímu neznáma, preto je potrebné sa s ňou oboznámiť. Cieľom je sa oboznámiť s možnosťami získavania informácií z webových citačných databáz WOS a SCOPUS zahŕňajúc spôsob prístupu do databáz a formou v akej budú dáta získané.

2. **Vytvorenie funkčného prototypu aplikácie**
   Cieľom je vytvoriť funkčnú REST webovú služby pomocou jazyka JAVA, ktorá bude poskytovať knižničným systémom sa dopytovať do citačných databáz WOS a SCOPUS. Služba bude umožňovať vyhľadávať buď diela na základe mena autora a názvu diela alebo ohlasy na vybrané dielo na základe identifikátora vybranej databázy. Taktiež chceme vytvoriť jednoduché testovacie rozhranie pre zjednodušenie testovania a možnosti testovania treťou osobou.

3. **Vybudovanie databázy**
   Za účelom obmedzenia potreby prístupov do citačných databáz a za účelom neskoršej práce nad získanými dátami sme sa rozhodli vytvárať si vlastnú, lokálnu databázu zo získavaných dát. V tejto databáze budú uložené deduplikované záznamy diel a ich ohlasy, ktoré sa získajú pri používaní nášho servletu.

4. **Získanie schopnosti pracovať s podpornými nástrojmi pre prácu v tíme**
   Podporné nástroje sú pre prácu v tíme veľmi užitočné a ich ovládanie každým členom tímu je potrebné pre zvýšenie efektivity práce. Aktívnym a správnym používaním týchto nástrojov sa centralizujú informácie ohľadne vykonávaných

činností na projekte a taktiež sa vďaka nim dá vyhnúť závažným rizikám. V našom tíme sme sa rozhodli používať nástroj Jira.

5. **Získanie schopnosti pracovať v tíme**
   Cieľom je aby sa každý člen naučil zásadám spolu práce v tíme a aktívne sa podieľal na práci v tíme. Veľmi dôležité je komunikovať s ostatnými členmi tímu podľa dohodnutých pravidiel a spolu riešiť vzniknuté problémy.

1.2 Celkový pohľad na systém

1.2.1 Architektúra

- systém je z architektonického hľadiska servlet poskytujúci dopytovacie rozhranie pre osoby a iné systémy,
- fungovanie servletu podporuje aplikačný server Apache Tomcat v.7
- metódy rozhrania, ktoré servlet poskytuje pre komunikáciu sú HTTP GET alebo POST kompatibilné
- rozhranie poskytované servletom je RESTful, prijíma aj vracia správy vo formáte JSON
- triedy servletu sú implementované v jazyku Java s využitím J2EE frameworku

1.2.2 Scenáre dopytov (Workflows)

Vyhľadanie dokumentu

- **Prijatie JSON dopytu s menom autora a názvom dokumentu**
  - dopyt vo formáte JSON:

Listing 1.1: Dopyt vo formáte JSON

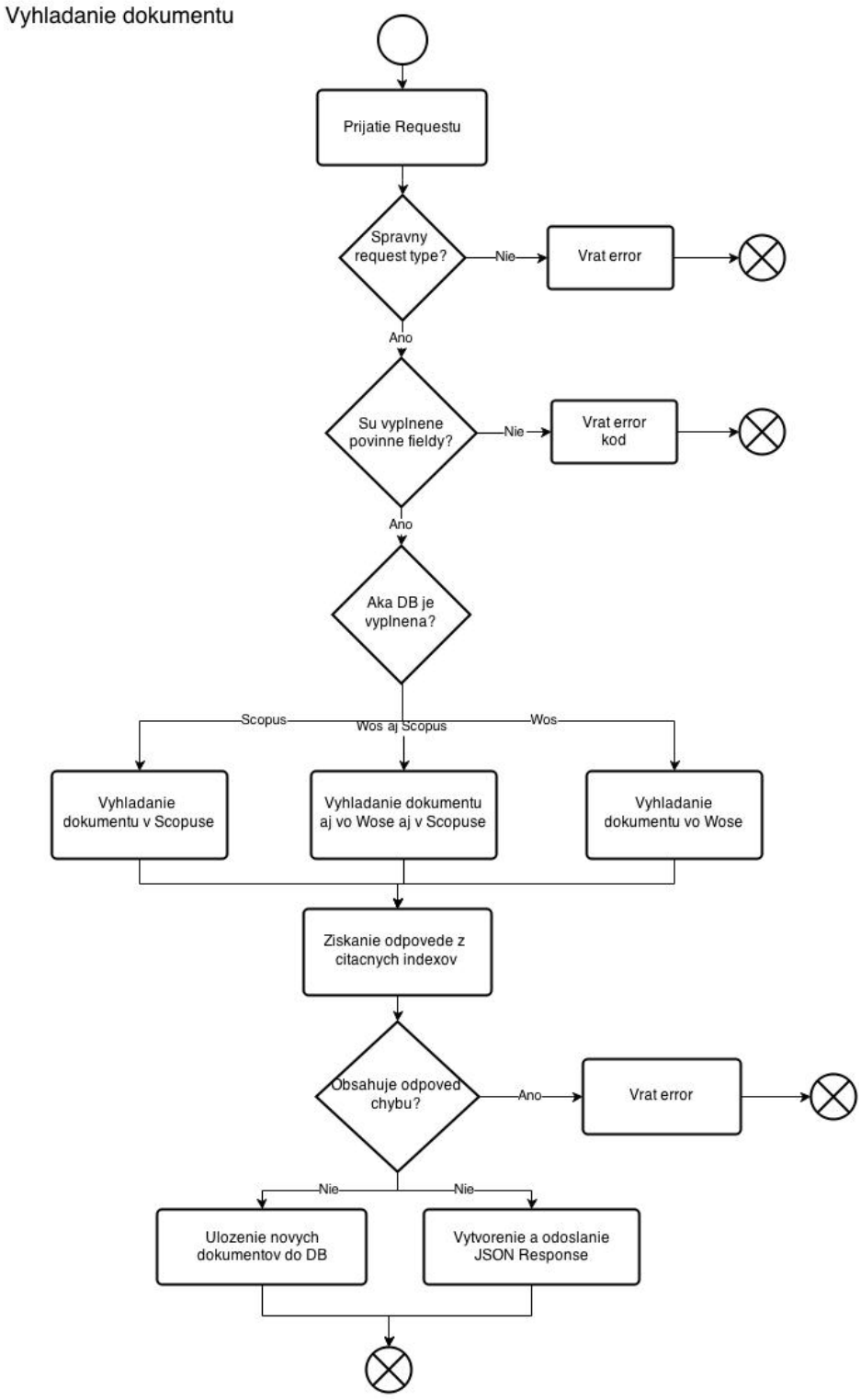
```
{
    "request_type": "search",
    "db": ["wos","scopus"],
    "title": "",
    "author": {
        "name": "",
        "surname": ""
    }
}
```

- **Príjem dopytu**
  - všeobecne pre všetky metódy

- **Správny request type?**
  - všeobecne pre všetky metódy
  - pre vyhľadanie dokumentov musí byť request_type "search"
- **Sú vyplnené povinné polia?**
  - musí byť vyplnený aspoň jednen záznam databázy, v ktorej sa má vyhľadávať
  - musí byť vyplnený autor dokumentu
  - musí byť vyplnený názov dokumentu
  - ak jeden z povinných parametrov chýba, vrátime chybovú hlášku hovoriacu o tom, ktoré parametre nie sú zadané
- **Aký záznam databázy je vyplnený?**
  - ak je vyplnený len SCOPUS, vyhľadávame len v SCOPUSe
  - ak je vyplnený len WOS, vyhľadávame len vo WOSe
  - ak sú vyplnené obe, vyhľadávame v oboch databázach
- **Vyhľadanie dokumentov v citačných indexoch**
  - vyhľadanie dokumentov aktuálne na základe mena autora a názvu dokumentu:
  - sformulovanie dopytu na API citačného indexu
  - odoslanie dopytov
- **Získanie odpovede z citačných indexov**
  - príjem odpovede z citačného indexu
- **Obsahuje odpoveď chybu?**
  - ak odpoveď z citačných indexov obsahuje chybu, vrátime chybu aj my
- **Uloženie nových dokumentov do databázy**
  - prejdeme všetky dokumenty a ak sa niektorý nenachádza v našej databáze (zistíme podľa SCOPUS/WOS ID), uložíme ho do nej
- **Vytvorenie a odoslanie JSON odpovede**
  - odpovede z WOS a SCOPUS sú vo formáte XML, no my ich potrebujeme previesť do formátu JSON a vybrať si štandard
  - odpoveď vo formáte JSON

Listing 1.2: JSON odpoveď

```
{
"searchQueryWos": "",
"searchQueryScopus": "",
"recordsWosCount": ,
"recordsWos": [
    {
        "wosID":"",
                    "author":"",
                    "title":""
    },
    {
        "wosID":"",
                    "author":"",
                    "title":""
    }
],
"recordsScopusCount": ,
"recordsScopus": [
    {
        "scopusID": "",
                    "author":"",
                    "title":""
    },
    {
        "scopusID": "",
                    "author":"",
                    "title":""
    }
]
}
```



Obr. 1.2: Workflow diagram vyhľadania dokumentu

Vyhľadanie ohlasov na dielo

- **Prijatie JSON dopytu**
  - dopyt vo formáte JSON:

Listing 1.3: Dopyt vo formáte JSON

```
{
    "request_type": "citations",
    "idWos": "", pokud je vyplnen, dohledaji se citace ve WoS
    "idScopus":"" , volitelny parameter
    "scopus_title": "", dohledaji se citace ve Scopus
    "scopus_author": "",
    "scopus_publication": "",
    "date_from": "",
            "user":"",      nazov institucie
            "userID":""    lokalne id diela v institucii
}
```

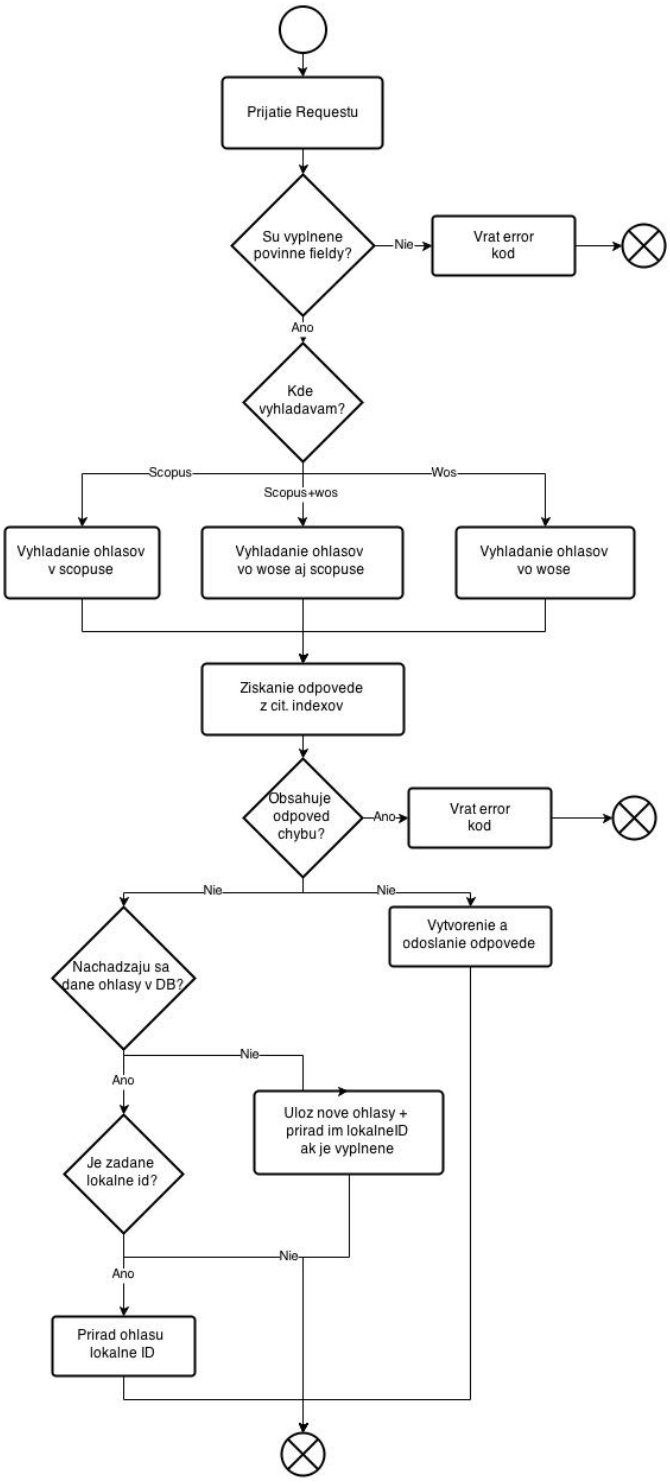
- **Príjem dopytu**
  - kontrola vstupných polí
- **Sú vyplnené povinné polia?**
  - request type pre túto metódu je "citations"
  - musí byť vyplnené buď WOS ID alebo SCOPUS author a title
- **Miesto hľadnia**
  - podľa parametrov prebehne vyhľadávanie v jednom alebo v oboch citačných indexoch
- **Vyhľadanie ohlasov**
  - vo WOS sa vyhľadá podľa WOS ID dokumentu
  - SCOPUS nepodporuje vyhľadávanie podľa ID, tak sa v ňom vyhľadá podľa iných parametrov
- **Získanie odpovede z citačných indexov**
  - príjem odpovede z citačného indexu
- **Obsahuje odpoveď chybu?**
  - ak odpoveď z citačných indexov obsahuje chybu, vrátime chybu aj my
- **Nachádzajú sa získané ohlasy v databáze?**

  - ak sa niektoré zo získasných záznamov nenachádzajú v databáze, uložia sa tam. Ak obsahoval dopyt aj userID, pridá sa k záznamu

- **Je zadané lokálne ID**
  - ak sa záznam nachádza v lokálnej databáze a nebolo zadané lokálne ID, pokračuje sa ďalej
- **Priradenie ohlasu k lokálnemu ID dokumentu**
  - ak sa záznam nachádza v lokálnej databáze a bolo zadané lokálne ID, priradí sa k danému záznamu
- **Vytvorenie a odoslanie odpovede**
  - odpoveď obsahuje získané ohlasy na dielo

Listing 1.4: JSON odpoveď

```
{
        "searchQueryWos": "",
        "searchQueryScopus": "",
        "recordsWosCount": ,,
        "recordsWos": [
            {
                "wosID":"",
                    "author":"",
                    "title":""
            },
            {
                "wosID":"",
                    "author":"",
                    "title":""
            }
        ],
        "recordsScopusCount": ,
        "recordsScopus": [
            {
                "scopusID": "",
                    "author":"",
                    "title":""
            },
            {
                "scopusID": "",
                    "author":"",
                    "title":""
            }
        ]
    }
```



Obr. 1.3: Workflow diagram vyhľadania ohlasov na dielo

Uloženie ID dopytovateľa k lokálnemu záznamu

- **Prijatie JSON dopytu**
  - dopyt vo formáte JSON:

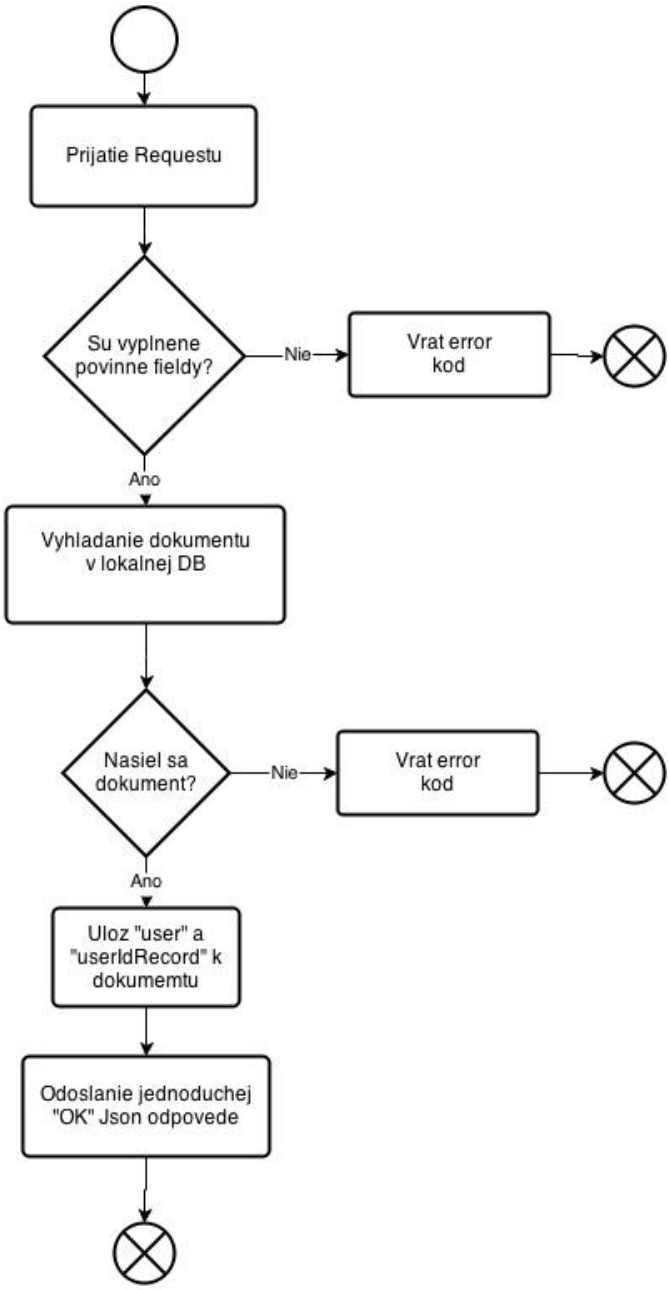
Listing 1.5: Dopyt vo formáte JSON

```
{
    "request_type": "addUserId",
    "user": "", identifikace instituce
    "userIdRecord": "", id zaznamu priradene zakaznikem
    "idWos": "",
    "idScopus": "",
}
```

- **Sú vyplnené povinné polia?**
  - request type pre túto metódu musí byť äddUserID"
  - musí byť vyplnený dopytovateľ - skratka inštitúcie
  - musí byť vyplnené userId - lokálne ID diela v rámci dopytujúcej sa inštitúcie
  - musí byť vyplnené WOS alebo SCOPUS ID
- **Vyhľadanie dokumentu v lokálnej databáze**
  - vyhľadanie na základe WOS a SCOPUS ID z dopytu
  - ak sú vyplnené obe ID, vyhľadanie záznamu dokumentu z oboch citačných indexoch
  - ak záznam nebol nájdený, vrátime chybovú hlášku a neskôr záznam dohľadáme v citačných indexoch
- **Uloženie používateľa a userIdRecord k dokumentu**
  - používateľom je napríklad STU
  - userIdRecord je napríklad SRU ID dokumentu
- **Odoslanie odpovede**
  - jednoduchá potvrdzujúca odpoveď - OK

Listing 1.6: JSON odpoveď

```
{
    "status": "", "1-ok" alebo "2-err"
}
```

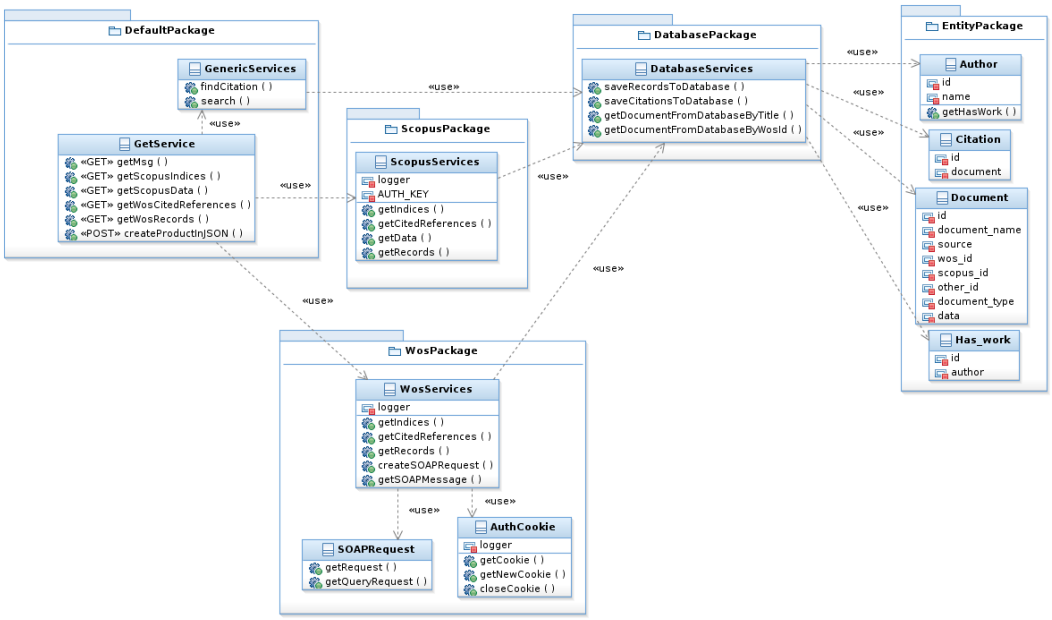


Obr. 1.4: Workflow diagram uloženia ID dopytovateľa k lokálnemu záznamu

1.2.3 Diagram tried

Diagram na nasledujúcom obrázku ilustruje rozdelenie funkcionality vytvorenej aplikácie do tried. Triedy sú rozdelené na základe funkčných aj doménových závislostí. Dospeli sme tak ku stavu, kedy:

- databázová logika sa nachádza v samostatnom balíčku. Všetok kód týkajúci sa databázy a mapovania entít je tak sústredený na jednom mieste a je tak možné jednoducho vymeniť použitú datbázu - dosiahli sme abstrakciu od použitej databázy
- metódy rozhrania, ktoré sú typu GET a POST a sú prístupné z vonka servletu sú taktiež oddelené v osobitnom balíčku. Je tak možné meniť za nimi stojacu implementáciu bez zmeny nimi poskytovaného rozhrania pre vonkajšie aplikácie.
- keďže každý z dvoch nami použitých citačných indexov komunikuje iným spôsobom (XML správy skrz SOAP vs. JSON správy skrz REST API), oddelili sme do samostatných balíčkov aj metódy špecifické pre jednotlivé domény. Triedy komunikujúce s rozhraním SCOPUS indexu tak nemusia napríklad implementovať mechanizmy pre vytváranie SOAP správ.



Obr. 1.5: Diagram tried systému

Kapitola 2

Moduly systému

2.1 1. Šprint

V tomto šprinte bolo dôležité oboznámiť sa s technológiami a citačnými indexami (SCOPUS a WOS) a ich API. Taktiež sme sa snažili analyzovať metadáta v Scopuse resp. Wose. Ďalšou časťou šprintu bolo získanie prístupu na server ARL6 a inštalácia potrebných technológií. Preto v tejto časti opíšeme len úlohy, ktoré boli technického charakteru.

2.1.1 Vytvorenie tímového webu

Analýza
Bolo potrebné dohodnúť sa a nájsť najlepší spôsob a riešenie akou formou prezentovať tím. Analýzou dostupných riešní sme dospeli k vytvoreniu jednoduchej webovej stránky pomocou nástrojov/rámcov WordPress alebo Drupal.

Inštalácia
Rozhodli sme sa pre jednoduché riešenie pomocou redakčného systému WordPress, s voľne dostupným template-om. Jedná sa o štandardnú kombináciu HTML a PHP s vlastným rozhraním na správu obsahu, ku ktorému môže pristupovať viacero používateľov.

Testovanie
Po inštalácií WordPress boli nahrané prvé zápisnice zo stretnutia a bol vytvorený prvý príspevok cez redakčný systém. Tímový web je vytvorený a dostupný na URL: http://labss2.fiit.stuba.sk/TeamProject/2014/team15is-si/

2.1.2 Inštalácia Tomcat

Analýza
Na základe analýzy sme vybrali najvhodnejšiu verziu aplikačného servera Apache Tomcat. Najnovšie verzie boli vo viacerých zdrojoch hodnotené ako nestabilné. Preto

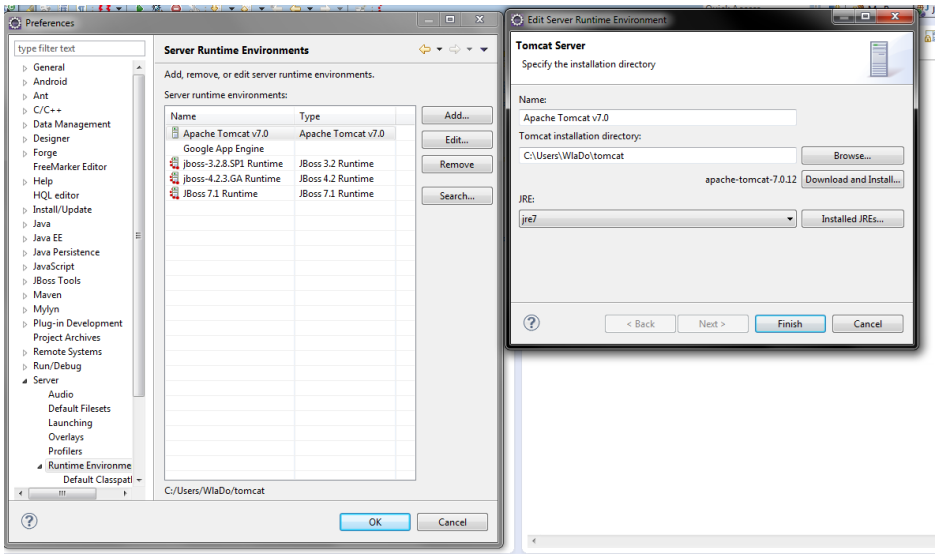
sme sa rozhodli pre verziu 7. Tomcat je spustený a dostupný na serveri ARL6 a porte 8080, z toho dôvodu sme museli vypnúť lokálnu inštanciu systému JiRa, ktorá na spomenutom serveri používala tento port.

Inštalácia

Z dôvodu lokálnych problémov s verziou Tomcatu z oficiálnej stránky poskytovateľa u členov tímu, sme sa rozhodli použiť Tomcat server s modifikovaným conf\server.xml. Inštalácia a integrácia v IDE Eclipse je zobrazená na obrázku nižšie.

Testovanie

Pre testovanie spustenia a stavu serveru je potrebné jednoducho zadať do prehliadača adresu: http://localhost:8080 prípadne remote http://arl6.library.sk:8080 Ak server beží zobrazí sa hláška „I am alive“



Obr. 2.1: Server Tomcat

2.1.3 Oboznámenie sa s technológiami

Analýza

REST webová služba musí byť implementovaná na servlete, komunikuje cez HTTP protocol využívajúci najmä metódy GET a POST. Pred každú metódu, ktorá bude zadefinovaná v rámci REST-u, je potrebné uviesť anotáciu. Pre GET @GET a pre POST@POST. GET vráti response pre nejakú požiadavku a POST spracováva údaje a navyše vráti Response. Pre každý Response sa definuje o aký typ Response ide.

Implementácia

Servlet je inicializovaný vo web.xml pomocou xml súboru nasledovne:

Listing 2.1: web.xml

```
<servlet>
  <servlet-name>bigdata-serlvet</servlet-name>
  <servlet-class>
    com.sun.jersey.spi.container.servlet.ServletContainer
  </servlet-class>
  <init-param>
    <param-name>com.sun.jersey.config.property.packages</param-name>
        <param-value>sk.fiit.team15</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
  <servlet-name>bigdata-serlvet</servlet-name>
  <url-pattern>/rest/*</url-pattern>
</servlet-mapping>
```

Cesta k REST WS je teda cez /rest/

Listing 2.2: REST WS

```
@GET
@Path("sayHi/{name}")
public Response getMsg(@PathParam("name") String name) {
  String returnName = "What is going on " + name.toUpperCase();
  return Response.status(200).entity(returnName).build();
}
```

Testovanie

Za týmto účelom bola vytvorená trieda (GetService.java), a jednoduchá metóda anotovaná pomocou @GET. Táto testovacia metóda je schopná zadaním mena ako parametra v ceste pozdraviť používateľa. Pre sayHi/BigData api vráti Response v tvare „What is going on BigData“ ako Plain text.

Ďalšie úlohy, ktoré sa riešili počas šprintu. Tieto úlohy majú predovšetkým charakter štúdia a preto nie sú podrobnejšie opisované.

- Naštudovanie dokumentácií
- Manuálne vyhľadávanie dát v Scopus-e
- Analýza dátového modelu
- Vytvorenie JiRa a jej naplnenie taskami
- Návrh dátového modelu schémy pre uchovávanie ohlasov
- Návrh spôsobu ukladania FULLTEXTov

2.2 2. Šprint

2.2.1 Dopyt pre WOS

Úloha

Vytvoriť jednoduchý servlet pre získavanie záznamov z citačnej databázy WOS na základe základných parametrov.

Analýza

Pre riešenie danej úlohy boli analyzované metódy pre získavanie záznamov bežiace na servlete Thomson, ktorý sa vie dopytovať do citačnej databázy WOS.

Implementácia

Pre implementovanie požadovanej funkcionality bola využitá get metóda servletu Thomson http://search.webofknowledge.com/esti/wokmws/ws/WokSearch. Výsledkom bola novovytvorená get metóda nášho servletu vracajúca xml dokument obsahujúci pre všetky nájdené záznamy názov diela, meno autora a ide diela vo WOSe.

Listing 2.3: Get metóda servletu

```
public static Response getRecords(String author, String title) {...}
```

Pre kazdy dopyt na servlet Thomson je potrebné Cookie, ktoré sa získava poslaním požiadavky na službu autentifikácie http://search.webofknowledge.com/esti/wokmws/ws/WOKMWSAuthenticate. Každý dopyt je teda možné využívať len istý čas (približne 5 minút), potom platnosť Cookie vyprší. Pre získavanie Cookie bola implementovaná samostatná metóda.

Listing 2.4: Metóda na získanie Cookie autentifikácie

```
static String getNewCookie() {...}
```

Na sledovanie expirácie cookie bol vytvorený časovač podľa návrhového vzoru Singleton, ktorý sa vytvorí len raz počas pri spustení nášho servletu.

Listing 2.5: Metóda na získanie aktuálneho času trvania session

```
public static long getActTime() {
        return actTime;
}
```

Vždy po prekročšní limitu trvanlivosti Cookie sa obnoví identifikácia žiadosťou o nové Cookie.

Listing 2.6: Kontrola dĺžky session

```
if (actTime - beforeTime >= 300000) {
        closeCookie(Timer.getInstance().getCookie());
        cookie = AuthCookie.getNewCookie();
        timer.setCookie(cookie);
```

```
        timer.setActTime(actTime);
}
```

Takisto bola potrebná implementácia metódy pre uzavretie aktuálnej session.

Listing 2.7: Metóda uzatvárajúca aktuálnu session

```
private static boolean closeCookie(String cookie) {...}
```

Testovanie

Metóda bola testovaná zatiaľ iba ručne, prostredníctvom náhodných vstupov. Pre overenie správnosti výsledkov, bol vytvorený JUnit test, ktorý kontroluje, či počet vrátených záznamov sedí s reálnym počtom výsledkov pri vyhľadávaní priamo cez web WOSu.

2.2.2 Dopyt pre Scopus

Úloha

Vytvoriť jednoduchý servlet pre získavanie záznamov z citačnej databázy Scopus na základe základných parametrov.

Analýza

Pre riešenie danej úlohy boli analyzované metódy pre získavanie záznamov bežiace na servlete Elsevier, ktorý sa vie dopytovať do citačnej databázy Scopus.

Implementácia

Pre implementovanie požadovanej funkcionality bola využitá get metóda servletu Elsevier https://api.elsevier.com&path=content/search/index:SCOPUS. Z dôvodu potreby testovania služby aj mimo akademickej pôdy bolo nutné použiť rest proxy, pretože služby servera je možné používať iba z akademických sietí. Výsledkom bola novovytvorená get metóda nášho servletu vracajúca xml dokument obsahujúci pre všetky nájdené záznamy názov diela a meno autora. Vstupnými parametrami metódy sú JSON objecty, ktoré používame pri požiadavke a sú vracané aj ako odpoveď. Túto funcktionalitu máme však spravenú zatiaľ len pre citačnú databázu Scopus.

Listing 2.8: Get metóda servletu

```
public List<JSONObject> getRecords(JSONObject jObj, JSONObject Obj)
```

Testovanie

Metóda bola testovaná zatiaľ iba ručne, prostredníctvom vlastnoručne zostrojených JSON vstupov. Pre overenie správnosti výsledkov, bol vytvorený JUnit test, ktorý kontroluje, či počet vrátených záznamov sedí s reálnym počtom výsledkov pri vyhľadávaní priamo cez web Scopusu.

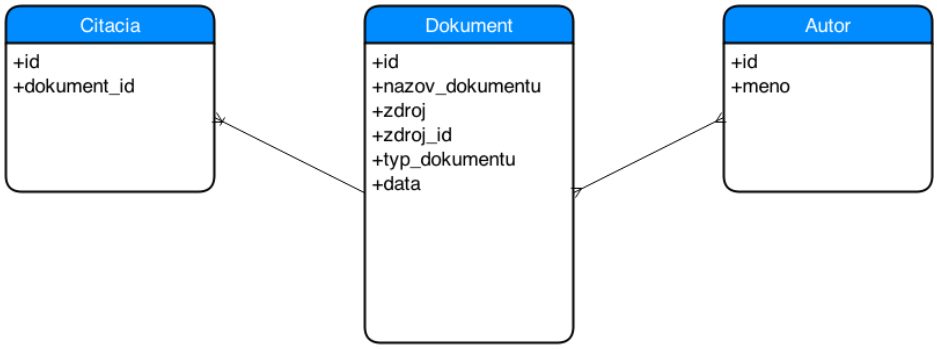
2.2.3 Vlastná databáza

Úloha

Vybudovať vlastnú databázu záznamov z WOSu, Scopusu spolu s databázou pre ukladanie fulltextov.

Analýza

Ako prvý krok bolo porovnanie dostupných databáz. Konečné rozhodnutie bolo v prospech databázového serveru postgresql, s ktorým majú viacerí členovia skúsenosti a pre ukladanie atribútov citačných databáz je výhodné použitie objektovo-relačnej databázy. Pre uchovávanie fulltextov sme sa rozhodli pre nosql databázu MongoDB, ktorá nám prišla pre tento problém najužitočnejšia. S fulltextami sme ale doposiaľ nepracovali. Po inštalácií sme sa zaoberali analýzou vhodného modelu databázy pre počiatočné ukladanie dát, ktoré zahŕňa ukladanie názvu diela, mena autora, id diela danej citačnej databázy, id dopytujúceho sa klienta a zvyšný text. Na základe vymedzených požiadaviek pre ukladanie nájdených záznamov bol vytvorený logický model databázy. Pri tvorbe modelu sa rátalo už aj s implementáciou funkcionality dopytovania sa na ohlasy daného diela.



Obr. 2.2: Logický model

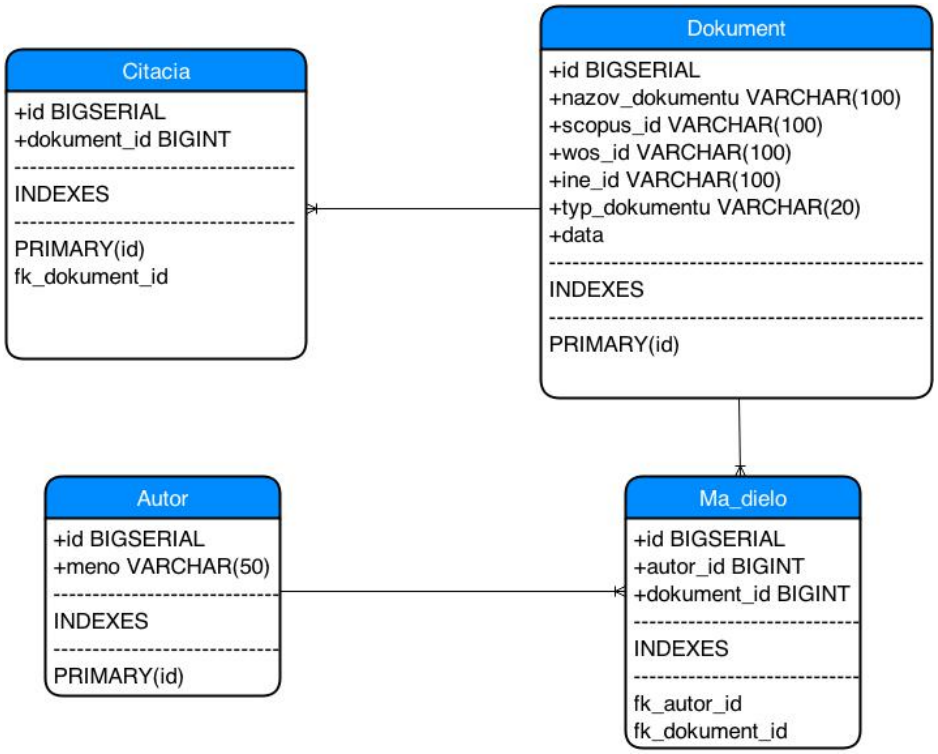
Na základe logického modelu bol vytvorený model fyzický. Po menšej úvahe sa dospelo k niekoľkým zmenám a tými boli pridanie samostatných atribútov pre WOS a Scopus id diela a pridanie atribútu ine_id označujúceho id klienta.

Implementácia

Na základe fyzického modelu bola vytvorené dva databázy, jedna spustená pre master vetvu a druhá pre developerskú. Následne boli v projekte implementované triedy-entity namapované na tabuľky databázy.

Testovanie

Testovanie funkčnosti databázy bolo vykonané rovnakými testovacímu vstupmi ako testovania oboch dopytovacích metód pre WOS a Scopus. Pri testoch sme pozerali, či databáza správne ukladá diela vrátené službami servletu.



Obr. 2.3: Logický model

2.3 3. Šprint

Bola vytvorená prvá verzia servletov. V priebehu šprintu bolo potrebné ich funkcionalitu zjednotiť a navzájom poprepájať. Jadro funkcionality aplikácie bolo testovateľné a z väčšej časti pokryté testami. Celý proces získavania ohlasov na diela bol funkčne spracovaný spolu s ukladaním dokumentov a ohlasov na ne. Celá komunikácia s aplikáciou aj v rámci nej prebieha štandardnou formou, vymieňaním správ vo formáte JSON.

2.3.1 Implementácia metódy pre vyhľadávanie citácií v indexoch WOS a SCOPUS

Analýza

Bolo treba vytvoriť metódu ktorá na základe mena autora alebo názvu diela vyhľadá citácie v databázach WoS a SCOPUS a uloží ich do databázy.

Implementácia

Vytvorili sme metódu ktorá spracováva vstupné parametre na vyhľadanie ohlasov v databáze. Neboli však ošetrené ešte vstupné dáta. Všetko sa bude meniť podľa zmeneného workflow diagramu.

Testovanie

Na testovanie sme použili ako vstupné parametre názov diela alebo meno autora. Pomocou týchto parametrov sme vyhľadali ohlasy na daného autora alebo dielo. Výstupné dáta sme ukladali do databázy.

2.3.2 Refaktoring kódu

Implementácia

Keďže kód bol pomerne rozsiahly, musel, pre zabezpečenie jeho čitateľnosti a udržiavateľnosti, prebehnúť jeho refaktoring. Na tvorbe sa podieľalo viac programátorov a preto ani jeho štruktúra nebola dokonalá. Bola vytvorená iba jedna trieda ktorá sa rozdelila na viac trieda podľa metód. Takisto prebehla ja úprava formátu kódu aby vyzeral esteticky dobre. Zmenili sa aj názvy premenných lebo niekto používal slovenské názvy iný zase anglické.

Testovanie

Po týchto úpravách bol program najskôr spustený kompilátorom aby sa otestovalo či sa dá spustiť. Následne sa otestovali už vytvorené metódy testami rovnakými ako predtým.

Príklad testovacieho JSONu na vyhľadanie diela použitý v tomto testovaní:

Listing 2.9: Testovací JSON

```
{
  "request_type": "search",
  "db": ["scopus"],
  "title": ["Beehive based machine to give snapshot",
  " of the ongoing stories on the web"],
  "author": {
     "name": "",
     "surname": "Navrat"
  }
}
```

2.3.3 Vytvorená POST metóda na rozoznávanie request typu

Analýza

Keďže máme vyhľadať ohlasy na dielo a takisto aj dielo v databázach WoS a SCOPUS, je potrebné vytvoriť metódu ktorá dokáže rozpoznať či ide o vyhľadanie ohlasu alebo či ide o vyhľadanie diela.

Implementácia

Vytvorili sme post metódu ktorá rozlíšila request type. V prípade že používateľ zadal do vstupného JSONu request_type „search“ išlo o JSON, ktorým chce používateľ v databázach vyhľadať dielo/diela na základe ďalších vstupných parametrov. Ak išlo o

request_type „citations“ tak išlo vyhľadanie ohlasov na dané dielo a autora z databáz WoS a SCOPUS.

Testovanie

Testovanie prebehlo na vstupných JSONoch kde boli zadané nejaké parametre. Boli vytvorené testovacie JSONy a implementované do testov v Jave. Pritom bola vytvorená trieda Tests, ktorá dostávala vstupné testovacie JSONy.

Príklad vstupného JSONu na request_type „search“ na vyhľadanie diela v SCOPUSe kde bolo zadané meno autora a dielo:

Listing 2.10: Hľadanie v Scopuse - vstupný JSON

```
{
  "request_type": "search",
  "db": ["scopus"],
  "title": ["Beehive based machine to give snapshot",
  " of the ongoing stories on the web"],
  "author": {
     "name": "",
     "surname": "Navrat"
  }
}
```

Príklad vstupného JSONu na request_type „citations“ na vyhľadanie diela v SCOPUSe kde bolo zadané meno autora a dielo:

Listing 2.11: Ohlasy v Scopuse - vstupný JSON

```
{
  "request_type": "citations",
  "idWos": "",
  "scopus_title": ["Beehive based machine to give snapshot",
  " of the ongoing stories on the web"],
  "scopus_author": "Navrat",
  "scopus_publication": "",
  "date_from": ""
}
```

Ďalšie úlohy, ktoré sa riešili počas šprintu. Tieto úlohy nemali charakter implementačný alebo inštalačný, preto nie sú podrobnejšie opisované.

Príprava testovacích dát a testov pre niektoré scenáre. Spísané a dostupné konfiguračné hodnoty programov a podporných prostriedkov vo Wiki.