

Sem vložte zadanie Vašej práce.

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
KATEDRA SOFTWAREVÉHO INŽENÝRSTVÍ



Diplomová práce

Spracovanie a vizualizácia chemických meraní v dátovom repozitári

Bc. Lukáš Košťenský

Vedúci práce: RNDr. David Antoš, Ph.D.

1. mája 2017

Pod'akovanie

Rád by som poďakoval RNDr. Davidovi Antošovi, Ph.D. za cenné rady pri písaní práce. Taktiež by som rád poďakoval Mgr. Miroslavovi Šimkovi za pomoc a rady pri návrhu a programovaní aplikácie. Tiež by som chcel poďakovať Ing. Janovi Horníčkovovi za pomoc pri získavaní testovacích dát, ich vysvetleníu a vytvorenie testovacích a produkčných serverov pre repozitár.

Prehlásenie

Prehlasujem, že som predloženú prácu vypracoval samostatne a že som uviedol všetky informačné zdroje v súlade s Metodickým pokynom o etickej príprave vysokoškolských záverečných prác.

Beriem na vedomie, že sa na moju prácu vzťahujú práva a povinnosti vyplývajúce zo zákona č. 121/2000 Sb., autorského zákona, v znení neskorších predpisov, a skutočnosť, že České vysoké učení technické v Praze má právo na uzavrenie licenčnej zmluvy o použití tejto práce ako školského diela podľa § 60 odst. 1 autorského zákona.

V Prahe 1. mája 2017

.....

České vysoké učení technické v Praze

Fakulta informačních technologií

© 2017 Lukáš Koštenský. Všetky práva vyhrazené.

Táto práca vznikla ako školské dielo na FIT ČVUT v Prahe. Práca je chránená medzinárodnými predpismi a zmluvami o autorskom práve a právach súvisiacich s autorským právom. Na jej využitie, s výnimkou bezplatných zákonných licencií, je nutný súhlas autora.

Odkaz na túto prácu

Koštenský, Lukáš. *Spracovanie a vizualizácia chemických meraní v dátovom repozitári*. Diplomová práca. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Cielom diplomovej práce je vytvoriť software, ktorý bude slúžiť ako repozitár vedeckých dát. Zameriava sa pritom na dáta Ústavu organickej chémie VŠCHT Praha. Výsledný repozitár používa rôzne nástroje. Ako jadro repozitára je použitá Fedora, ktorá sa stará o ukladanie dát. Vyhľadávanie rieši Elasticsearch. Bolo vytvorené webové užívateľské rozhranie v programovacom jazyku Python s využitím frameworku Django. Vytvorené boli tiež nástroje pre správu a kontrolu oprávnení a nástroj na import dát z aplikácie Open Enventory.

Klíčová slova Fedora, repozitár, Elasticsearch, webová aplikácia, Python, Django, Java

Abstract

The aim of the diploma thesis is to create software, which will be used as science data repository. It is focused on data from Department of organic chemistry UCT Prague. Final repository uses different tools. Fedora is used as core of repository and takes care about data storing. Elasticsearch is used for searching in repository. Web user interface is created in Python with usage of

framework Django. I created also tools for checking and controlling permissions and tool for importing data from Open Enventory.

Keywords Fedora, repository, Elasticsearch, web application, Python, Django, Java

Obsah

Úvod	1
1 Popis problému	3
1.1 Zdieľanie dát v tíme	3
1.2 Open data/Open access	3
1.3 Zálohovanie a archivácia	4
2 Súčasné riešenia	5
2.1 Repozitáre	5
2.2 Nástroje na zber a organizáciu chemických dát	9
3 Analýza a návrh riešenia	11
3.1 Analýza požiadaviek	11
3.2 Výber technológií	13
4 Implementácia	19
4.1 Návrh aplikácie	19
4.2 Zobrazovanie chemických dát	30
4.3 Automatický import dát	37
4.4 Kontrola stavov a prístupové práva	44
Záver	53
Literatúra	55
A Zoznam použitých skratiek	57
B Obsah priloženého CD	59

Zoznam obrázkov

3.1	Zobrazenie dát vo webovom rozhraní Fedory	15
4.1	Architektúra repozitára	19
4.2	Výpis dát v kolekcii InfraredSpectra	30
4.3	Zobrazenie detailu konkrétnej položky (Ethanolu)	31
4.4	Hierarchická štruktúra kolekcí vo Fedore	34
4.5	Model osoby - vedca	35
4.6	Modely chemických prvkov a chemických prvkov v reakcii	36
4.7	Model reakcie	37
4.8	Modely analytických dát a zdrojových dát	38
4.9	Štruktúra databázovej tabuľky reaction	39
4.10	Štruktúra databázovej tabuľky analytical_data, 1. časť	41
4.11	Štruktúra databázovej tabuľky analytical_data, 2. časť	42
4.12	Štruktúra databázovej tabuľky reaction_chemical, 1. časť	43
4.13	Štruktúra databázovej tabuľky reaction_chemical, 2. časť	44
4.14	Diagram WebACL	46
4.15	Proces získania oprávnení vo Fedore	47

Zoznam tabuliek

2.1	MARC	6
3.1	ElasticSearch vs. Solr	16
4.1	Typy prístupu v ACL	45

Úvod

Nielen výskumníci, ale aj bežný užívatelia počítačov riešia problém s ukladaním dát. Tých máme stále viac, musíme ich niekam ukladať a ideálne aj zabezpečiť, že sa nestratia napríklad pri poškodení disku. Výskumníci, na rozdiel od bežných užívateľov, potrebujú závery výskumov, ale aj samotné dáta z meraní publikovať. Aby takto zverejnené množstvo dát bolo pre ostatných použiteľné, musí v nich byť možné vyhľadávať.

Ako centrálnu úložisko dát sa bežne používajú repozitáre, takúto službu poskytujú najmä knižnice, školy, ale aj iné inštitúcie. Pre ukladanie a správu ale aj zverejnenie kníh, článkov a iných textových dokumentov existuje niekoľko rôznych softwarov. Dáta uložené v nich sú popisované metadátami.

Existujú štandardizované metadátové schémy, ktoré sú určené na popis metadát ku knihám, článkom ale aj iným dátam. Nie vždy, najmä ak potrebujeme pracovať s dátami z rôznych vedeckých odvetví, je možné existujúcimi metadátovými schémami dostatočne popísať dokumenty.

V prípade vedeckých dát už taktiež nestačí v repozitári zobrazíť metadáta a umožniť stiahnuť textový dokument. Rôzne typy dát je potrebné zobrazíť rôznymi spôsobmi, v niektorých je taktiež potrebné vyhľadávať, pričom pre vyhľadávanie v nich nemusí stačiť textové zadanie hľadaného výrazu.

Repozitár, ktorý by bol vhodný pre rôzne vedecké dáta, musí vedieť uložiť veľké množstvo dát, v nich vedieť vyhľadávať a tiež ich zobrazíť. CESNET, z.s.p.o. má k dispozícii dátové úložisko [1], nad ktorým je možné vytvoriť repozitár. V rámci diplomovej práce sa teda pokusím vytvoriť software, ktorý bude slúžiť pre ukladanie vedeckých dát. Zamerám sa pritom na dáta Ústavu organickej chémie VŠCHT Praha. Pri vytváraní repozitára využijem, upravím alebo vytvorím vhodné nástroje, ktoré umožnia prácu s dátami.

Popis problému

Repozitár slúži vo všeobecnosti ako centrálné miesto, ktoré sa stará o ukladanie a správu dát. Takúto službu môžu chcieť poskytovať rôzne inštitúcie, napríklad školy, knižnice,... Pod slovom repozitár si môžeme taktiež predstaviť konkrétny software, ktorý sa stará o ukladanie, archiváciu a sprístupnenie dát. V tejto diplomovej práci budeme pod slovom repozitár rozumieť práve software.

1.1 Zdieľanie dát v tíme

Ústav organickej chémie VŠCHT Praha potrebuje vyriešiť ukladanie a sprístupnenie dát. Potrebuje ukladať, analyzovať a prezentovať infračervené vibračné, NMR a hmotnostné spektroskopické merania, chemické vzorce a reakcie.

Nad jedným datasetom môže pracovať viacero ľudí, ktorí môžu riešiť rôzne merania a pokusy alebo spoločne pracovať na jednom meraní. V oboch prípadoch počas priebehu samotného výskumu potrebujú prístup k dátam, ktoré vytvoril iný člen tímu. Taktiež musia mať možnosť dáta upravovať (napr. opakované merania a pokusy, keď je potrebné doplniť nové výsledky). Repozitár teda musí umožniť zdieľanie dát v tíme, rôzne oprávnenia pre osoby, ktoré majú mať k dátam prístup, verziovanie dát.

1.2 Open data/Open access

Pri vývoji repozitára je potrebné myslieť na možnosť zverejnenia (časti) dát pre širokú verejnosť s možnosťou ich ďalšieho využitia alebo odkazovania na ne. Takto zverejnené dáta označujeme pojmom Open data. V prípade zverejnených výskumov hovoríme o Open access (OA). Repozitár musí umožniť zverejnenie všetkých alebo časti uložených dát. Cieľom vývoja repozitára je vytvorenie platformy, s využitím ktorej bude možné publikovať nielen články

a závery výskumu, ale aj čiastkové merania a experimenty, ktoré k výsledkom viedli.

1.3 Zálohovanie a archivácia

Zálohovaním dát rozumieme vytváranie kópie práve spracúvaných alebo v relatívne nedávnej dobe uložených dát. Archiváciou rozumieme uchovávanie dokumentačných materiálov.

Zálohované dáta môžu byť poškodené degradáciou média, fyzickým poškodením média alebo v súčasnosti rozšírenými cryptovírusmi. Zálohovať dáta na jedno médium nestačí. Je dobré sa riadiť pravidlom 3-2-1. Tri kópie všetkých dôležitých dát, na dvoch rôznych médiách, pričom jedna kópia by mala byť uložená off-site, teda niekde mimo pracovného prostredia. [2]

Pri archivácii dát je potrebné myslieť na čitateľnosť dát po dlhej dobe. Preto je potrebné myslieť nielen na zabezpečenie dát, ale aj na archiváciu programu potrebného na prečítanie archivovaných dát.

Repozitár by mal byť pre užívateľov možnosťou ako dáta zálohovať. Zároveň jeho napojenie na služby CESNETu umožní ochranu dát, akú by bolo na pracovisku VŠCHT Praha ťažké dosiahnuť.

V budúcnosti bude možné repozitár rozšíriť o nástroje, ktoré by umožnili aj dlhodobú archiváciu dát.

SúčasnÉ riešenia

2.1 Repozitáre

Existujú rôzne repozitáre, ktoré sa od seba líšia použitou technológiou, možnosťou rozšírenia, používajú rôzne metadátové schémy. Niektoré sú voľne dostupné ako open source, iné ako proprietárny software alebo hosťované aplikácie. V tejto časti uvádzam prehľad dostupných aplikácií. Zameriavam sa najmä na vlastnosti, ktoré boli pre ďalší vývoj repozitára kľúčové, a to: open source (aby bolo možné software ďalej upravovať), použitie metadátovej schémy, modulárnosť softwaru (jednoduchá možnosť rozšírenia o ďalšie nástroje) a verziovanie (najmä kvôli zdieľaniu a zálohovaniu dát).

2.1.1 Metadáta

Na popis uložených dokumentov slúžia metadáta. Metadáta sú štrukturované dáta nesúce informáciu o primárnych dátach. Pojem metadát je používaný predovšetkým v súvislosti s elektronickými zdrojmi. [3] Za zdroje alebo dokumenty pokladáme knihy, články, časopisy, audio nahrávky, obrazové záznamy, binárne súbory obsahujúce dáta z meraní a ďalšie. Kvôli vzájomnej prepojenosti repozitárov, vyhľadávaniu dát a správnej interpretácii informácií je snaha o vyvinutie celosvetovo používaného štandardu pre popis dát.

O to sa snažia rôzne metadátové schémy, pomocou ktorých je možné zdroje popísať. Medzi najznámejšie schémy patrí Dublin Core [<http://dublincore.org/>] a MARC [<http://www.loc.gov/marc/>].

2.1.1.1 Dublin Core

Dublin Core (skrátene DC) vznikol s cieľom jednoducho a všeobecne popísať zdroje. Metadátová schéma je určená pre popis zdrojov umiestnených na web. Táto schéma obsahuje 15 prvkov. To sú: názov (title), autor (creator), predmet (subject), popis (description), vydavateľ (publisher), prispievateľ (contributor), dátum (date), typ (type), formát (format), identifikátor (identifier),

zdroj (source), jazyk (language), vzťah (relation), pokrytie (coverage) a práva (rights). [4, s. 37] Tieto prvky nie sú povinné a môžu sa opakovať. Jednotlivé vlastnosti sú teda pomenované. Ako sa World Wide Web menil, v snahe o vytvorenie sémantického webu sa vyvinul aj štandard Dublin Core. Od roku 2008 obsahuje formálne domény a rozsahy v definíciách vlastností. Tento aktualizovaný variant vlastností sa nazýva dcterms.

Jednotlivé prvky môžu byť ďalej rozšírené o kvalifikátor. Ten môže lepšie určiť, čo daná položka popisuje. Napríklad namiesto všeobecného autora tak môžeme upresniť, či išlo o ilustrátora (dc:creator.ilustrator), editora (dc:creator.editor),... Kvalifikátory ale nesmú meniť význam vlastnosti, dokument popísaný s využitím kvantifikátorov tak ostáva kompatibilný aj pre systémy, ktoré kvantifikátory nepoužívajú.[5, s. 295]

2.1.1.2 MARC

MARC (MACHine-Readable Cataloging) využívajú najmä knihovníci. Bol navrhnutý pre popis bibliografických údajov v strojovo čitateľnej podobe. Schéma obsahuje vlastnosti, ktoré sú očíslované. Kým názov v dcterms je označený ako title, v MARCu je označený číslom 245 (title proper statement). Na rozdiel od dcterms obsahuje niekoľko pomocných polí (ako napríklad 222 kľúčový názov, 240 unifikovaný názov,...).[5, s. 288-289] Takéto označenie je ľahko čitateľné pre stroje, knihovníci si pri každodennej práci s týmito číslami ich významy zapamätávajú. Človek, ktorý ich vidí prvýkrát, významu nerozumie.

MARC je od DC komplikovanejší, dokáže však presnejšie popísať zdroj. Prvé číslo v číselných kódoch určuje, o aký typ informácie ide, jednotlivé kódy sú popísané v tabuľke 2.1. V prípade kódov 1XX, 4XX, 6XX, 7XX a 8XX sa obsah upresňuje doplnením dvojice čísel. Zvyčajne sa dodržiavajú nasledujúce dvojice: X00 - Mená osôb, X40 - Bibliografické názvy, X10 - Názvy firiem, X50 - Tematické pojmy, X11 - Názvy stretnutí/konferencií, X51 - Názvy miest, X30 - Jednotné názvy.

Tabuľka 2.1: Typ informácie v kóde MARC

0XX	Kontrolná informácia, identifikačné a klasifikačné čísla,...
1XX	Hlavné údaje
2XX	Názvy a kapitoly (názov, edícia, vydanie)
3XX	Fyzický popis,...
4XX	Informácie o dieloch/sériách
5XX	Poznámky
6XX	Kontaktné informácie na subjekty
7XX	Pridané informácie (iné než o subjektoch, dieloch/sériách); linkovacie polia
8XX	Rada pridaných informácií, informácie o holdingoch
9XX	Vyhradené pre lokálnu implementáciu

Hodnoty niektorých kódov majú presne určenú dĺžku a obsah, napríklad kód 008 obsahuje číselný kód pre zdroj a kód jazyka, v akom je dokument napísaný. Iné polia nemajú obmedzenú dĺžku ani definovanú štruktúru a hodnoty v nich sú len textové polia. Tieto polia často obsahujú aj podhodnoty, ktoré sú označené písmenami a, b, c,... Napríklad pole s kódom 100 (meno autora) môže obsahovať podpolia so štandardizovaným menom, celými krstnými menami a dátumi.[5, s. 288-291]

Použitie navrhovaného repozitára by malo byť jednoduché aj pre užívateľov, ktorí s metadátami nemajú veľké skúsenosti a nepotrebujú komplikovaný popis dát. Pre skúsenejších užívateľov by však bolo dobré zachovať možnosť použitia zložitejších, prípadne vlastných metadátových schém. Repozitár by teda mal vedieť používať aj iné metadátové schémy než len Dublin Core alebo MARC.

2.1.2 Software

V tejto časti popisujem prehľad najrozšírenejších softwarov, ktoré sa používajú ako implementácie repozitárov. Uvádzam prehľad dôležitých vlastností pre ďalší vývoj repozitáray a to či ide o proprietárny alebo open source systém, kvôli možnosti ďalších úprav; programovací jazyk a modulárnosť softwaru; aké metadátové schémy používajú a či ich je možné rozširovať; a či daný software umožňuje verziovanie uložených dát.

Keďže každý software pokrýva inú kombináciu týchto vlastností, bolo by veľmi náročné pokúšať sa o nejakú klasifikáciu. Preto uvádzam len prehľad ich vlastností:

2.1.2.1 Digital Commons

[<http://digitalcommons.bepress.com/>]

Hostovaná platforma inštitucionálneho repozitára. Zameraný na školy a školské dokumenty.

Používa Dublin Core schému, v používateľskom rozhraní podporuje aj iné vlastnosti než len DC, aj keď nepodporuje iné schémy (vrátane MARC).

Autori vedia prispôsobiť repozitár požiadavkám klienta.

Nepodporuje verziovanie.

2.1.2.2 LIBSYS

[<http://www.libsys.co.in/>]

Proprietárny software. Repozitár funguje ako webová aplikácia.

Používa MARC ako schému metadát.

2.1.2.3 SimpleDL

[<http://www.simplifiedl.com/>]

2. SÚČASNÉ RIEŠENIA

Proprietárny software.

Metadáta na základe Dublin Core. Môžu byť rozšírené o iné schémy.

2.1.2.4 Greenstone

[<http://www.greenstone.org/>]

Repozitár vyvinutý na Univerzite Waikato.

Používa MARC schému.

Modulárna architektúra, napísaný v jazyku Java. Plugíny v jazyku Perl.

Nepodporuje verziovanie.

Open source

2.1.2.5 Invenio

[<http://inveniosoftware.org/>]

Software bol pôvodne vyvinutý pre CERN. Umožňuje vytvoriť digitálnu knižnicu alebo repozitár dokumentov dostupný cez web.

Používa špecifikáciu MARC pre metadáta.

Má modulárnu architektúru. Napísaný v jazyku Python.

Podporuje verziovanie uložených dát.

Open Source

2.1.2.6 EPrints

[<http://www.eprints.org/>]

Vyvinutý na Univerzite Southampton.

Používa rôzne typy metadátových polí, ktoré je možné nastavovať (upraviť zobrazovanie, indexovanie, vyhľadávanie).

Modulárny software napísaný v jazyku Perl.

Podporuje verziovanie dát.

Open Source

2.1.2.7 DSpace

[<http://www.dspace.org/>]

Software pôvodne vyvinutý MIT a Hewlett-Packard. Od vzniku má viac ako 2000 inštalácií po celom svete.

Ako východziu schému pre popis dát používa Dublin Core, je však možné použiť aj iné schémy.

Ide o súbor spolupracujúcich Java webových aplikácií. K dispozícii je RESTful webové užívateľské rozhranie.

Neumožňuje verziovanie uložených dát.

Open source

2.1.2.8 Fedora

[<http://www.fedora-commons.org/>]

Je možné použiť rôzne schémy pre popis dát.

Flexibilný, jednoducho rozširiteľný, modulárny repozitár. Napísaný v programovacom jazyku Java.

Umožňuje verziovanie uložených dát.

Open Source

2.2 Nástroje na zber a organizáciu chemických dát

Výskumníci v oblasti chémie si vedú laboratórne denníky so záznamami hypotéz, experimentov, analýz alebo interpretáciou experimentov. V súčasnosti sa denníky vedú v elektronickej forme s využitím elektronických laboratórnych denníkov (často sa pre tento software používa skratka ELN). Keďže Ústav organickej chémie VŠCHT Praha používa a naďalej chce používať len E-Notebook a Open Enventory, iné nástroje na zber a organizáciu chemických dát v prehľade neuvádzam.

Prehľad programov, ktoré používa Ústav organickej chémie VŠCHT Praha:

2.2.1 E-Notebook

[<http://www.cambridgesoft.com/Ensemble/E-notebook/>] Software od firmy Perkin Elmer. V súčasnosti je k dispozícii len ako Enterprise verzia alebo ako súčasť cloudových aplikácií Elements <https://elements.perkinelmer.com/> a plánovaného ChemDraw E-notebook <http://chemdrawnotebook.perkinelmer.cloud/>. Perkin Elmer dodáva Enterprise aplikáciu ako hotové riešenie s úpravami a nastavením pre konkrétného zákazníka. Software funguje na serveroch Oracle. Cloudové riešenie aj Enterprise verzia softwaru je uzatvorená a upravovať ich môže len dodávateľská firma.

2.2.2 Open Enventory

[<https://www.chemie.uni-kl.de/goossen/open-enventory/>] Webová open source aplikácia napísaná v jazyku PHP, ktorá má k dispozícii všetky zdrojové kódy. Neexistuje k nej ale žiadna dokumentácia. Pri prechádzaní kódu aplikácie som naviac zistil, že veľká časť komentárov a aj časť samotného kódu sú napísané v nemčine.

Využíva MySQL databázu. Databázová schéma taktiež nie je prehľadná, v jednotlivých tabuľkách nie sú označené cudzie kľúče, a teda chýbajú prepojenia na iné tabuľky. Na stĺpcoch, ktoré by mali byť cudzími kľúčmi, sú len indexy.

Analýza a návrh riešenia

3.1 Analýza požiadaviek

CESNET, z.s.p.o. bol oslovený Ústavom organickej chémie VŠCHT Praha, ktorý potrebuje vyriešiť ukladanie a sprístupnenie vlastných dát - chemické vzorce, reakcie, analýzy nameraných dát. Nad jedným datasetom môže pracovať viacero výskumníkov, niektorí dáta namerali, iní ich analyzujú alebo každý člen tímu pracujúci na jednom projekte rieši iné merania a experimenty. Výskumník teda okrem vlastných dát potrebuje mať prístup aj k dátam ostatných ľudí v tíme. Nie všetky dáta môžu byť zverejnené, na niektoré výskumy môže platiť embargo a majú byť zverejnené až neskôr, k dátam, s ktorými sa aktuálne pracuje, majú mať prístup len členovia tímu alebo dáta ich autor nechce zverejniť z iných dôvodov. Pre tieto dáta taktiež potrebujú vyriešiť zálohovanie.

Ako sa postupne zistilo, o vyriešenie ukladania a sprístupnenia dát má záujem viacero rôznych a na rôzne dáta zameraných skupín. Preto chceme vytvoriť repozitár, ktorý bude jednoduché rozšíriť pre užívateľov používajúcich iné typy dát a metadát.

CESNET v rámci služieb DataCare poskytuje dátové úložisko s celkovou hrubou kapacitou presahujúcou 21 PB. Toto úložisko poskytuje dátový priestor na zálohovanie, archiváciu, zdieľanie dát. [1] Úložisko dát umožňuje ukladať dáta tak, aby boli prístupné len pre jednu osobu alebo zdieľané pre skupinu ľudí. Služba FileSender umožňuje rýchle a jednoduché odosielanie veľkých súborov až stovkám prijímateľov, OwnCloud umožňuje sprístupniť dáta cez webové rozhranie a synchronizovať ich s inými zariadeniami alebo zdieľať s inými ľuďmi. [6] CESNET teda má vytvorené zázemie, ktoré bude možné využiť pre potreby repozitára.

3.1.1 Funkčné požiadavky

Ústav organickej chémie VŠCHT Praha v súčasnosti využíva aplikácie pre tvorbu laboratórnych denníkov, v ktorých sú uložené informácie o chemických prvkoch, reakciách, ktoré počas pokusu nastali, taktiež o priebehu meraní a pokusov. Aby bola využiteľnosť repozitára čo najlepšia a práca s ním čo najjednoduchšia, požadujú možnosť importovať dáta z aplikácie Open Enventory, v ktorej si vedú laboratórne denníky, do repozitára.

Projekty v rámci laboratórnych denníkov obsahujú:

- meno vedca alebo vedcov, ktorí na meraniach a pokusoch spolupracovali,
- priebeh meraní a pokusov,
- reaktanty,
- vzniknuté produkty,
- chemickú rovnicu, ktorá bude zobrazená aj schematicky (obrázkom),
- pozorovanie priebehu pokusu,
- NMR spektrá (spektrá nukleárnej magnetickej rezonancie) chemických prvkov,
- IR (infračervené) spektrá,
- hmotnostné spektrá,
- pri chemických prvkoch je potrebné evidovať:
 - štandardný názov prvku,
 - zápis štruktúry,
 - obrázok štruktúry,
 - chemický vzorec prvku,
 - molekulárnu hmotnosť,
 - hmotnosť prvku,
 - koncentráciu.

Repozitár musí pre tieto dáta umožniť:

- Uložiť nové dáta.
- Upraviť existujúce dáta.
- Zobraziť existujúce dáta.
- Uložiť históriu zmien dát.

- Vyhľadávanie v metadátach.
- Kontrolovať oprávnenia na prístup k dátam.
- Import dát z aplikácie Open Eventory.

Prístup k jednotlivým dokumentom a poliam ale môže byť limitovaný. Vytváranie a úprava jednotlivých dokumentov je umožnená len konkrétnym užívateľom.

Aplikácia Open Eventory musí byť upravená tak, aby umožnila export dát vo formáte vhodnom pre import do repozitára.

3.1.2 Požiadavky na vlastnosti repozitára

- Repozitár bude pre užívateľov dostupný ako webová aplikácia.
- Repozitár musí umožniť ďalšie rozšírenie pre iné typy dát.
- Repozitár bude napojený na služby CESNETu.

3.1.3 Administračné rozhranie

Aby bolo možné spravovať kolekcie, oprávnenia pre prístup, užívateľov priamo v aplikácii, je potrebné vytvoriť v repozitári administračné rozhranie. Implementácia tohto rozhrania ale nie je súčasťou tejto diplomovej práce.

Z požiadaviek na umožnenie rozšírenia pre ďalšie typy dát plynie nutnosť vytvoriť nástroje, ktoré umožnia vytvoriť a spravovať šablóny pre zobrazenie, vytvorenie a editáciu týchto dát. Môžeme predpokladať, že tieto šablóny síce bude vytvárať osoba, ktorá má aspoň nejaké skúsenosti s tvorbou HTML šablón, nedá sa predpokladať skúsenosť s programovaním v Pythone a frameworku Django.

Administračné rozhranie by teda malo poskytovať aj nástroje pre tvorbu a úpravu HTML šablón pre jednotlivé typy dokumentov. Taktiež tvorbu modelov pre jednotlivé typy dokumentov. Rozhranie pre vytváranie a editáciu oprávnení pre jednotlivé kolekcie, dáta alebo metadátové polia, správu užívateľov a skupín.

3.2 Výber technológií

3.2.1 Výber repozitára

Keďže ani jeden existujúci repozitár nespĺňa všetky požiadavky alebo nevie uložiť/zobraziť dáta pre Ústav organickej chémie VŠCHT Praha, bolo potrebné vytvoriť nový alebo upraviť existujúci software. Vytvorenie nového softwaru od základov by bolo neefektívne. Vyššie zmienené repozitáre fungujú, niektoré ich časti by teda boli programované nanovo. Zvolená bola možnosť

doplniť/upraviť funkčnosť existujúceho repozitára. Keďže zadávateľ preferuje open source riešenia, vybrali sme vhodný repozitár z open source repozitárov.

Okrem toho boli pri výbere vhodného repozitára do úvahy brané ďalšie kritériá, a to možnosť použitia viacerých metadátových schém, programovací jazyk, podpora komunity. Do finálneho výberu sa dostali DSpace a Fedora. Z testov, ktoré sme pri výbere repozitára previedli, vyplynulo, že sú oba repozitáre podobne výkonné. Vyhľadávanie v metadátach bude navyše zabezpečené samostatnou aplikáciou. Oba zvládajú milióny záznamov [7] [8, s. 213].

Vďaka návrhu Fedory je pridávanie rozšírení do tohto softwaru jednoduchšie, taktiež už má vyriešené verziovanie uložených dát. DSpace má k dispozícii webové užívateľské rozhranie, ktoré je možné upravovať a ďalej rozširovať. Fedora používa jednoduché webové rozhranie, ktoré umožňuje len základnú prácu s dátami. Vytvorenie samostatného, nového webového užívateľského rozhrania s využitím RESTapi je ale jednoduchšie než úprava jadra DSpace, aby zvládal verziovanie uložených dát.

Z existujúcich možností bola zvolená Fedora ako najvhodnejší software pre možnosť ďalších potrebných úprav pre použitie v rámci služieb CESNET, z.s.p.o. a splnenie požiadaviek Ústavu organickej chémie VŠCHT Praha.

Základné webové rozhranie, ktoré poskytuje Fedora, umožňuje zobrazit metadáta, stiahnuť binárne súbory, vytvoriť potomka typu kontajner a binárny dokument, upraviť alebo zmazať existujúci dokument. Za kontajner sú vo Fedore považované všetky dokumenty okrem binárnych, teda aj tie, ktoré už žiadneho potomka nemajú. Binárneho typu sú dokumenty obsahujúce dátové súbory nahrané do Fedory. Webové rozhranie je možné vidieť na obrázku 3.1.

Pre účely repozitára dát pre Ústav organickej chémie VŠCHT Praha je takéto zobrazenie dát nedostatočné. Bolo potrebné vytvoriť užívateľské rozhranie, ktoré vhodným spôsobom zobrazí zoznam dokumentov v rámci kolekcie ale aj samotné dáta a metadáta dokumentu.

Vytvorené webové užívateľské rozhranie tiež musí umožniť vyhľadávanie v uložených dátach.

Existujúce rozhranie umožňuje vytvoriť nový dokument a následne k nemu pridať informácie. Tie je možné zadávať len v textovej forme s využitím jazyka RDF (Resource Description Framework), teda trojíc - subjekt, predikát a objekt. RDF je jazyk používaný pre reprezentáciu informácií na webe. [9] Preto je potrebné zjednodušiť vytváranie nových kolekcií a dokumentov v užívateľsky prívetivej forme.

Pre vytváranie nových dokumentov, úpravu existujúcich a zobrazenie dát bude potrebné vytvárať HTML šablóny. Repozitár má byť čo najľahšie rozširiteľný pre rôzne dáta a metadáta. Preto bolo potrebné vytvoriť nástroj, ktorý by umožnil používanie šablón pre nové typy dát s čo najmenším zásahom do kódu aplikácie. Tento nástroj vie pracovať so šablónami uloženými priamo vo Fedore.

test@cs

Home / dcterms / e4 / bc / 86 / e2 / e4bc86e2-c77d-46c1-a613-15bee74782d3

Created at
2016-12-12T11:27:25.815Z by guest

Last Modified at
2016-12-12T11:27:25.815Z by guest

Children 0

Properties

dc: **abstract**
Bla bla bla
En en en

dc: **alternative**
testovací

dc: **creator**
Lukas

dc: **dateSubmitted**
2016-12-12T00:00:00+01:00

dc: **title**
test
test

fedora: **created**
2016-12-12T11:27:25.815Z

fedora: **createdBy**
guest

fedora: **hasParent**
<http://127.0.0.1:8080/fcrepo/rest/dcterms>

fedora: **hasVersions**
<http://127.0.0.1:8080/fcrepo/rest/dcterms/e4/bc/86/e2/e4bc86e2-c77d-46c1-a613-15bee74782d3/fcr:versions>

fedora: **lastModified**
2016-12-12T11:27:25.815Z

fedora: **lastModifiedBy**
guest

fedora: **writable**
true

ns003: **hasIndexingTransformation**
dc_object

rdft: **type**
<http://fedora.info/definitions/v4/indexing#indexable>
<http://fedora.info/definitions/v4/repository#Container>
<http://fedora.info/definitions/v4/repository#Resource>
<http://purl.org/dc/elements/1.1/Object>
<http://www.w3.org/ns/ldp#Container>
<http://www.w3.org/ns/ldp#RDFSource>

Create New Child Resource

Type
container

Identifier
(auto-generated identifier)

Add

Update Properties

PREFIX premis: <http://www.loc.gov/premis /rdfriv1#>
PREFIX nt: <http://www.jcp.org /jcr/nt/1.0>
PREFIX rdfs: <http://www.w3.org /2000/01/rdf-schema#>
PREFIX ns004: <http://cesnet.cz /ns/repository/type#>
PREFIX ns003: <http://fedora.info /definitions/v4/indexing#>

Update

Create Version Snapshot

(auto-generated name)

Create Version

Delete Resource

Delete

Obr. 3.1: Zobrazenie dát vo webovom rozhraní Fedory

Prístupové práva je taktiež možné nastaviť pomocou RDF priamo v existujúcom webovom rozhraní. Musím však prácu s nastavovaním a kontrolou oprávnení zjednodušiť. Na to bolo potrebné upraviť aj kód Fedory.

3.2.2 Výber aplikácie na vyhľadávanie

Na vyhľadávanie v repozitári bude použitá samostatná aplikácia. Synchronizáciu dát s Fedorou zabezpečuje medzivrstva, ktorá komunikuje s oboma aplikáciami. Potrebujeme teda aplikáciu na vyhľadávanie v textových meta-dátach, ktorá je čo najrýchlejšia a zvláda veľké množstvo (milióny) záznamov.

Keďže zadávateľ preferuje open source, pri výbere sme sa rozhodovali medzi týmito najrozšírenejšími vyhľadávacími aplikáciami:

3.2.2.1 Elasticsearch

<https://www.elastic.co/products/elasticsearch>

ElasticSearch je distribuovaný, RESTful vyhľadávací a analytický software. Umožňuje veľmi rýchle vyhľadávanie v indexovaných dátach. Dopyty je možné posilať s využitím RESTful api a JSONu, knižnice sú dostupné pre rôzne programovacie jazyky vrátane Pythnu a Javy. Podľa [10] ide o najrozšírenejší vyhľadávací engine.

3.2.2.2 Solr <http://lucene.apache.org/solr/>

Solr je taktiež RESTful vyhľadávací software s podporou pre dáta vo formáte JSON, XML, CSV alebo binárnych dát cez HTTP.

Obe vyhľadávacie aplikácie vychádzajú z jadra Apache Lucene, čo je vysokovýkonná, plnohodnotná, v texte vyhľadávacia knižnica napísaná v Jave. Umožňuje full-textové vyhľadávanie v dokumentoch. <http://lucene.apache.org/core/>

3.2.2.3 Porovnanie vyhľadávacích aplikácií

V tabuľke 3.1 je porovnanie vlastností, ktoré rozhodovali pri výbere vyhľadávacieho enginu.

Tabuľka 3.1: Porovnanie vyhľadávacích enginov

Vlastnosť	Solr	ElasticSearch
Formát vstupných dát	XML, CSV, JSON	JSON
HTTP REST API	Áno	Áno
Knižnice pre Javu	Áno	Áno
Knižnice pre Python	Áno, vytvorená komunitou	Áno
Integrácia vo frameworku Django	Áno	Áno
Vnorené dokumenty	Nie	Áno
Vzťah rodič-potomok	Nie	Áno

Vnorené dokumenty a možnosť vyhľadávať rodičov, ktorých deti spĺňajú špecifikovanú podmienku a opačne vyhľadávať potomkov, ktorých rodičia spĺňajú špecifikovanú podmienku, viedli k tomu, že je v repozitári použitý ElasticSearch ako vyhľadávací software. Medzivrstva ale umožňuje pracovať aj s aplikáciou Solr.

3.2.3 Výber programovacieho jazyka a frameworku pre vytvorenie užívateľského rozhrania

Na vývoji repozitára pracuje viacero ľudí, každý ovláda a preferuje iné jazyky. Keďže sme sa rozhodli využiť Fedoru ako repozitár dát a implementovať vlastné webové rozhranie, bolo potrebné určiť programovací jazyk, v akom má byť toto rozhranie naprogramované.

Fedora je vytvorená s využitím programovacieho jazyka Java, ja som mal okrem Javy skúsenosti s PHP, Pythonom a ďalšími programovacími jazykmi, ktoré sa ale pre tvorbu webových aplikácií zvyčajne nepoužívajú alebo som s nimi mal len minimálne skúsenosti. Mgr. Miroslav Šimek, ktorý vytváral repozitár záverečných prác na VŠCHT Praha má najväčšie skúsenosti s vývojom aplikácií v Pythone s využitím frameworku Django. Väčšina webových služieb, ktoré ma na VŠCHT na starosť oddelenie CIS (Centrum informačných služieb) a práve Mgr. Miroslav Šimek, využíva Python a framework Django.

Zhodnotil som svoje možnosti naučiť sa Django a nakoniec sme sa rozhodli pre využitie Pythonu <https://www.python.org/> ako programovacieho jazyka pre vytvorenie webového rozhrania a frameworku Django <https://www.djangoproject.com/>, ktorý by mal prácu uľahčiť.

3.2.3.1 Python

Python je interpretovaný programovací jazyk vyššej úrovne. Obsahuje efektívne dátové štruktúry a umožňuje jednoduchý, ale efektívny prístup k objektovo-orientovanému programovaniu. [11]

3.2.3.2 Django

Django je webový framework pre Python. Na prvý pohľad na fungovanie Djanga sa zdá, že využíva návrhový vzor MVC (Model-view-controller), avšak kontrolérom nazveme v prípade Djanga „view“ a view podľa vzoru MVC „template“. V Django interpretácii vzoru MVC ale „view“ popisuje dáta, ktoré sú zobrazené užívateľovi. Pritom nie je dôležité ako sa dáta zobrazia, podstatné je, ktoré dáta budú zobrazené.

Obsah dát je oddelený od prezentácie dát. K prezentácii sú využívané šablóny („template“), ktoré definujú ako sú dáta zobrazené.

Kontrolér podľa definície MVC je v prípade Djanga skôr framework samotný. Kód, ktorý obslúži dopyt a požiadavok pošle na správne view podľa konfigurácie Django URL. V konfigurácii Django URL sú regulárne výrazy. Ak sa splní niektorý z výrazov, zavolá sa funkcia, ktorá má na starosť obsluhuje požiadavkov z danou URL adresou.

Preto sa používa skôr označenie MTV (Model-template-view), ktoré presnejšie popisuje to ako funguje Django. [12]

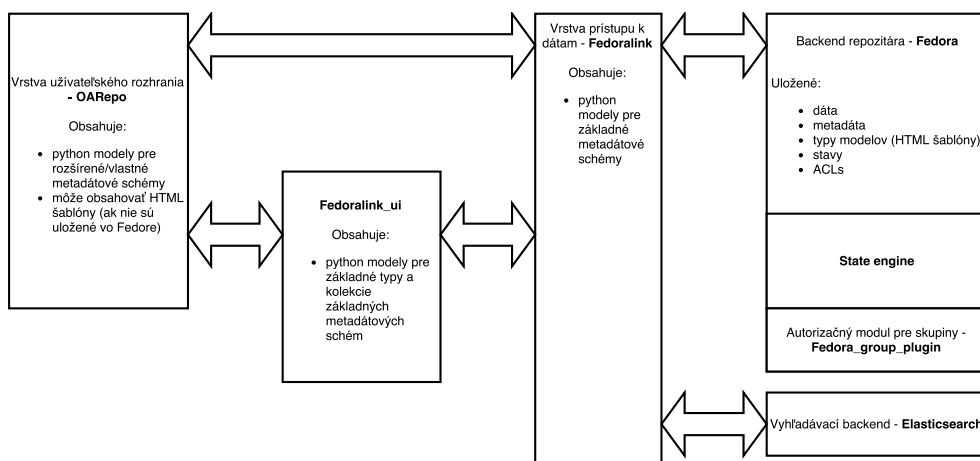
Model je klasická dátová vrstva, ktorá obsahuje logiku potrebnú pre prácu s dátami.

Implementácia

4.1 Návrh aplikácie

Ako backend repozitára je použitá Fedora. Pre zjednodušenie práce s oprávneniami budú použité stavy. Tento stav je dokument uložený vo Fedore, ktorý môže napríklad vyjadrovať stav publikovania dát (novovytvorený dokument, v procese schvaľovania, schválený dokument). Stav obsahuje informáciu o oprávneniach (kto má právo dokument s týmto prideleným stavom vidieť, kto ho môže upravovať,...) a povolené zmeny stavov. Takto je možné dokumentom meniť stavy, v akých sa nachádzajú a meniť tak oprávnenia. Preto nebude potrebné pre každý dokument definovať nové oprávnenia.

Fedora síce umožňuje definíciu prístupových práv (ACLs, z angličtiny Access Control Lists teda zoznamy prístupových práv), ale stavy nepoužíva. Je



Obr. 4.1: Architektúra repozitára

ju preto potrebné upraviť tak, aby s týmito stavmi vedela pracovať (kontrolovať prechody medzi stavmi). To zabezpečí modul State engine, ktorý upraví kód Fedory.

Oprávnenia chceme nastavovať nielen pre konkrétnych užívateľov, ale aj pre rôzne skupiny, do ktorých užívatelia patria. Zároveň sme pri návrhu repozitára mysleli na možnosť prihlásenia cez rôzne autorizačné služby ako je Shibboleth (<https://shibboleth.net/>) alebo Perun (<https://perun.cesnet.cz>). Využijeme štandardnú On-Behalf-Of HTML hlavičku, s ktorou vie pracovať Fedora a vlastnú On-Behalf-Of-Django-Groups hlavičku. Pre spracovanie tejto hlavičky je nutné do Fedory doplniť autorizačný modul (Fedora_group_plugin), ktorý bude súčasťou Fedory.

Užívateľské rozhranie chceme vytvoriť s využitím programovacieho jazyka Python a frameworku Django. Pre možnosť vytvorenia rôznych užívateľských rozhraní pre rôzne používateľské skupiny (ktoré potrebujú pracovať s inými dátami a metadátami) boli vytvorené medzivrstvy Fedoralink a Fedoralink_ui. Fedoralink má na starosť komunikáciu s Fedorou a Elasticsearchom s využitím REST api. Umožňuje pracovať s dokumentmi získanými z Fedory ako s Django objektmi. Django objekt je inštancia nejakej triedy, s ktorou je možné pracovať podľa definovaných štandardov frameworku Django. Fedoralink dokáže metadáta podľa ich typu získavať z Elasticsearch alebo z Fedory. S využitím Elasticsearch aplikácia taktiež môže v repozitári vyhľadávať.

Fedoralink_ui je aplikácia napísaná v jazyku Python s využitím frameworku Django, ktorá obsahuje logiku pre prácu so šablónami. Tie chceme ukladať priamo v repozitári, bolo teda potrebné vytvoriť nástroj, ktorý dokáže z repozitára získať správne šablóny, cachovať ich a po doplnení dát z dokumentov do HTML šablóny zobrazí výslednú webovú stránku.

Samotná aplikácia OARepo obsahuje modely a prípadne šablóny, ak nie sú uložené vo Fedore, pre jednotlivé typy dát. Tieto modely definujú vlastné alebo rozširujú existujúce metadátové schémy. Model je trieda v Pythone, po získaní dokumentu z Fedory aplikácia vytvorí objekt v Pythone, ktorý je inštanciou takejto triedy (modelu).

Navrhnutú architektúru repozitára je možné vidieť na obrázku 4.1. Nižšie sú detailnejšie rozpísané funkcie jednotlivých modulov.

4.1.1 Fedora

Ako už bolo zmienené v predchádzajúcej kapitole, ako backend pre repozitár bola zvolená Fedora. V nej sú uložené dáta, metadáta, typy modelov spolu s HTML šablónami, stavy a oprávnenia (ACL).

Metadáta vo Fedore sú uložené s využitím RDF (Resource Description Framework), teda ako trojice - subjekt, predikát a objekt.

4.1.2 State engine

Pre možnosť využívania stavov bude nutné rozšíriť Fedoru o tento modul. Modul rieši prechody medzi stavmi, zmenu stavov, zmenu kontroléru stavov, konkrétnu operáciu povolí len oprávneným osobám. Oprávnené osoby sú určené pomocou ACL. Keďže návrh dátového formátu pre popis stavov pracujúcich s prístupovými právami nadväzujúcimi na štandard W3C WebAccessControl a implementácia tohto modulu boli uvedené ako časť zadania, podrobnejšie sa tomuto modulu venujem v samostatnej sekcii 4.4.

4.1.3 Fedora_group_plugin

Doplňujúci modul do Fedory, ktorý umožňuje overenie oprávnení aj na základe členstva v django skupinách. Framework Django má vyriešenú prácu s užívateľmi, obsahuje kód, ktorý umožňuje ich registráciu, prihlásenie, v administráčnom rozhraní správu užívateľov a tiež užívateľských skupín. Pre skupiny je možné nastaviť rôzne oprávnenia. Priamo v kóde aplikácie, využívajúcej framework Django, je tak možné získať objekt, ktorý obsahuje informácie o prihlásenom užívateľovi. Takto sa vieme dostať k jeho používateľskému menu, ale aj ku skupinám, ktorých je členom.

Samotná Fedora s webac umožňuje overenie autorizácie na základe štandardnej On-Behalf-Of hlavičky, o ktoré sa stará DelegateHeaderPrincipalProvider. On-Behalf-Of HTTP hlavička umožňuje impersonifikáciu na iného užívateľa. Často sa používa, aby administrátor mohol testovať systém ako niektorý z užívateľov alebo simulovať chybu, ktorá niektorému užívateľovi nastala.

Fedora musí bežať pod Java Servlet kontajnerom (Tomcat, GlassFish, Jetty...), autentizáciu rieši s využitím týchto kontajnerov. Užívateľské údaje sú definované v nastavení kontajneru. Pri pridávaní nového užívateľa by teda bolo potrebné zmeniť nastavenie Java Servlet kontajnera. Zmena týchto nastavení sa väčšinou prejaví až po reštarte kontajnera. Užívateľov by sme museli mať definovaných v aplikácii, ktorá sa stará o webové rozhranie, aby sa užívateľ mohol prihlásiť a pridávať nové dáta, v nastavení Java Servlet kontajnera a taktiež v definícii prístupových práv priamo vo Fedore, aby bolo možné riešiť autorizáciu.

S využitím On-Behalf-Of hlavičky je možné požiadavky na Java Servlet kontajner prevádzať stále pod jedným užívateľom (FedoraAdmin) ale Fedora už užívateľa autorizuje na základe užívateľského mena uvedeného v On-Behalf-Of hlavičke.

Podobne ako užívatelia sú aj skupiny v Java Servlet kontajneroch definované v ich nastaveniach. Fedora však nemá implementáciu pre využitie žiadnej HTTP hlavičky, ktorá by umožňovala impersonifikáciu na nejakú skupinu. Rozšírenie Fedora_group_plugin umožňuje autorizáciu na základe On-Behalf-Of-Django-Groups hlavičky, o ktoré sa stará DjangoGroupPrincipalProvider.

Skupinu teda stačí mať uloženú v databáze webovej aplikácie, kde ich ukladá Django.

Hodnota posielaná v týchto hlavičkách je vo forme urn. Ak teda máme na vstupe užívateľské meno vo forme emailu (napr. „user@vscht.cz“), výsledná hodnota bude „urn:vscht.cz:user“, inak je v tvare „urn:user“.

Vďaka tomuto rozšíreniu bude možné aj jednoduché rozšírenie repozitára o autentizáciu cez iného poskytovateľa identity (napr. Shibboleth), keď sa zároveň vynechá nutnosť registrácie užívateľa do webovej aplikácie.

Plugin je súčasťou git repozitára federalinku. Autorom tohto pluginu je Mgr. Miroslav Šimek.

4.1.4 Elasticsearch

Aplikácia, ktorá umožňuje rýchle vyhľadávanie v metadátach.

4.1.5 Fedoralink

Ako je spomenuté vyššie, framework Django obsahuje logiku pre prístup k dátam. Je možné vytvoriť modely - teda triedy, ktoré využívajú techniku ORM (object-relational mapping) a umožňujú tak pracovať s dátami získanými z relačných databáz. Relačné databázy sú navrhnuté tak, aby umožňovali ukladanie dát ale taktiež selekciu a získavanie dát (teda vyhľadávanie v dátach).

Repozitáre sú však navrhnuté pre ukladanie dát. Prístup k jednotlivým dokumentom vo Fedore je možný priamo, ak vieme jeho id (nazývaný „slug“). Tento slug môžeme pri vytváraní nového dokumentu navrhnúť, avšak Fedora môže uložiť dokument s iným slugom a ten vrátiť. Ak id dokumentu nepoznáme, musíme prechádzať dokumenty od koreňa repozitára. Preto využívame pre indexovanie dát a vyhľadávanie samostatnú aplikáciu Elasticsearch.

Django nie je možné priamo napojiť na obe aplikácie tak, aby stačilo vytvoriť v dátovej vrstve modely. Keďže sme chceli pre programátorov zvyknutých na Django zachovať prácu s modelmi, na akú sú zvyknutý, vznikla medzivrstva Fedoralink. Aplikácia je rozhranie, ktoré umožňuje prácu s Fedorou a Elasticsearchom. Modely umožňujú ukladanie dát do Fedory, získanie dokumentov priamo cez id, ale aj vyhľadávanie s využitím Elasticsearch. Navonok tieto modely vyzerajú ako klasické Django modely.

Aplikácia pôvodne vznikla pre potreby repozitára záverečných prác VŠCHT Praha, v programovacom jazyku Python s využitím frameworku Django. Autorom federalinku je Mgr. Miroslav Šimek. Fedoralink bol počas vývoja repozitára ďalej upravovaný v rámci diplomovej práce. Aktuálnu verziu je možné nájsť na <https://github.com/CESNET/federalink>

4.1.5.1 FedoraObject

Aby bolo možné s dokumentmi získanými z Fedory pracovať ako s objektmi v Django, boli vytvorené triedy, ktoré s týmito dokumentmi pracujú. Trieda

FedoraObject je základnou triedou pre tieto objekty. Umožňuje pracovať s dátami získanými vo formáte RDF z Fedory, aj keď vopred nepoznáme štruktúru týchto dát.

Ak potrebujeme získať alebo upraviť niektorú metadátovú položku, pracujeme s objektom nasledovne (v tomto prípade chceme získať alebo upraviť názov - title zo schémy Dublin Core):

```
# [RDF:Name]
obj [DC.title]
```

Z objektu môžeme získať jeho ID, URL identifikátor (slug), jeho potomkov, objekt uložiť späť do Fedory alebo ho zmazať.

Dokument vo Fedore môže byť typu container alebo bitstream. V druhom prípade môže mať k sebe priradený súbor. Preto aj táto trieda FedoraObject umožňuje prácu so získaným bitstreamom, a to pomocou funkcie get_bitstream.

4.1.5.2 IndexableFedoraObject

Rozširuje triedu FedoraObject. Ak využívame túto triedu, o schéme metadát musíme vedieť ďalšie informácie. Vieme, akého typu sú jednotlivé metadátové polia (napr. či ide o reťazec, pole, ktoré môže byť vo viacerých jazykoch alebo o dátum...).

Príklad využitia IndexableFedoraObject:

```
#
# DCObject is indexable and provides .title and .creator
#   property, that get mapped to
# DC.* predicates in RDF by simple_namespace_mapper
#
class DCObject(IndexableFedoraObject):

    title = IndexedLanguageField(DC.title,
                                required=True,
                                verbose_name=_(
                                    'Title'))

    alternative = IndexedTextField(DC.alternative,
                                  verbose_name=_(
                                      'Alternative_□
                                      title'))

    abstract = IndexedLanguageField(DC.abstract,
                                    verbose_name=_(
                                        'Abstract'),
```

```
        attrs={
            'presentation': 'textarea'
        })

    creator = IndexedTextField(DC.creator,
                               verbose_name=_('Creator'))

    contributor = IndexedTextField(DC.contributor,
                                   verbose_name=_('Contributor'))

    dateSubmitted = IndexedDateTimeField(DC.dateSubmitted,
                                          verbose_name=_('Date submitted'))

    dateAvailable = IndexedDateTimeField(DC.dateAvailable,
                                          verbose_name=_('Date available'))

    class Meta:
        rdf_types = (DC.Object,)
```

Trieda `DCObject` bude využitá pri dátach získaných z Fedory, ktoré obsahujú metadáta podľa schémy DublinCore. Samotná trieda dedí z triedy `IndexableFedoraObject`.

`IndexedLanguageField` - metadátové pole, ktoré môže byť vo viacerých jazykoch. Za samotnou hodnotou je vložený parameter „@lang“, ktorý podľa skratky jazyka („en“, „cs“) určuje, v akom jazyku je daná hodnota. `IndexedTextField` - metadátové pole, ktoré obsahuje textový reťazec. `IndexedDateTimeField` - metadátové pole obsahujúce dátum a čas.

RDF typ je taktiež uložený vo Fedore a umožňuje mapovať získaný dokument na správnu triedu.

4.1.5.3 FedoraTypeManager

Trieda (singleton) zodpovedná za vytvorenie inštancie `FedoraObject` (alebo jej podtriedy) podľa RDF metadát získaných z Fedory počas behu aplikácie. Dokument získaný z Fedory môže mať viacero RDF typov, a teda môže spadať do

viacero tried. Z nich sa vyberie najvhodnejšia alebo sa pomocou viacnásobnej dedičnosti vytvorí nová trieda kombinujúca vlastnosti viacerých existujúcich tried, ktorá sa následne použije pre prácu s takto získaným dokumentom z Fedory.

Získanie správnej triedy pre dokument má na starosti funkcia `get_object_class`:

```
@staticmethod
def get_object_class(metadata, model_class=None):
    """
    Returns the best python class for the given metadata

    :param metadata:    the metadata
    :return:            python class which fits the
                        metadata
    """

    from .models import FedoraObject

    types = metadata[RDF.type]

    possible_classes = {FedoraObject: 0}
    if model_class:
        possible_classes[model_class] = 1

    # look at classes registered on rdf types and if the
    # class match, add it to the dict of possible
    # classes
    for clz, rdf_and_priority in FedoraTypeManager.
        on_rdf_types.items():
        if _type_matches(types, rdf_and_priority[0]):
            possible_classes[clz] = max(possible_classes
                .get(clz, 0), rdf_and_priority[1])

    # look at classes registered on rdf predicates and
    # if the class match, add it to the dict of
    # possible classes
    for clz, rdf_and_priority in FedoraTypeManager.
        on_rdf_predicates.items():
        if _has_predicates(metadata, rdf_and_priority
            [0]):
            possible_classes[clz] = max(
                possible_classes.get(clz, 0),
                rdf_and_priority[1])
```

```
# call class method handles_metadata and if it
returns a priority, add the class as well
for clz in FedoraTypeManager.models:
    priority = getattr(clz, 'handles_metadata')(
        metadata)
    if priority is not None and priority >= 0:
        possible_classes[clz] = max(
            possible_classes.get(clz, 0), priority)

# convert to a list, add priorities from
superclasses as well
# (i.e. 2 * current_priority + sum of priorities of
superclasses)
propagated_possible_classes = []

for clazz, priority in possible_classes.items():

    for clz in inspect.getmro(clazz):
        if clz in possible_classes:
            priority += possible_classes[clz]

    propagated_possible_classes.append((clazz,
        priority))

# sort by priority
propagated_possible_classes.sort(key=lambda x: -x
    [1])

# remove classes that are in mro of other classes
classes = []
seen_classes = set()
for clazz, priority in propagated_possible_classes:
    if clazz in seen_classes:
        continue

    classes.append(clazz)

    for clz in inspect.getmro(clazz):
        seen_classes.add(clz)

# got a list of classes, create a new type (or use a
cached one ...)
return FedoraTypeManager.generate_class(classes)
```

4.1.6 Fedoralink__ui

Je súčasťou git repozitára fedoralinku. Modul sa stará o užívateľské rozhranie aplikácie. Pre komunikáciu s Fedorou využíva fedoralink.

4.1.6.1 Mapovanie generických URL adries

Funkcie v rámci súboru *generic_urls.py* mapujú URL adresy v aplikácii na správne časti kódu pre zobrazenie, editáciu alebo vyhľadávanie. Z URL adresy zistíme ID dokumentu alebo kolekcie vo Fedore.

Vzory využívajúce regulárne výrazy pre URL adresy:

- `'^$',` - index
- `r'^(?P<collection_id>[a-zA-Z0-9_-]*)?search(?P<parameters>.*)$'` - vyhľadávanie v rámci kolekcie
- `^(?P<id>.*)/addSubcollection$'` - pridanie novej subkolekcie
- `^(?P<id>.*)/add$'` - vytvorenie nového dokumentu ako potomka dokumentu s daným ID
- `^(?P<id>.*)/edit$'` - upravenie dokumentu s daným ID
- `^(?P<id>.*)$'` - zobrazenie dokumentu s daným ID

Tieto generické URL adresy môžu byť ďalej rozšírené v niektorej časti aplikácie.

4.1.6.2 Získanie šablóny pre zobrazenie, editáciu dokumentu alebo zoznam dokumentov v kolekcii

O zobrazenie správnych údajov v správnej šablóne, prípadne o vytvorenie nového dokumentu/kolekcie so správnymi údajmi sa ďalej stará kód v súbore *views.py*.

V samotnej aplikácii (vo fedoralinku alebo v koncovej aplikácii) musí byť model dokumentu - trieda v Pythone. Ostatné potrebné veci sú uložené priamo vo Fedore. V nej sú uložené jednotlivé typy dokumentov, pri kolekciiach je uložený typ subkolekcií a potomkov. Tieto dokumenty môžu mať navyše uložené šablóny pre zobrazenie, úpravu a vytvorenie potomkov. Taktiež je možné do Fedory uložiť typy jednotlivých polí a k nim šablóny pre ich zobrazenie/editáciu.

Trieda *ResourceType*, ktorá dedí z *IndexableFedoraObject*, umožňuje uložiť šablóny pre rôzne typy dokumentov. Získanie správneho typu dokumentu je možné vďaka párovaniu cez RDF typ.

```
class ResourceType(IndexableFedoraObject):
    label = IndexedTextField(CESNET_TYPE.label,
        verbose_name=_('Label'), level=IndexedField.
        MANDATORY)

    template_view = IndexedLinkedField(CESNET_TYPE.
        template_view, Template, verbose_name=_('Template
        _for_view'))

    template_edit = IndexedLinkedField(CESNET_TYPE.
        template_edit, Template, verbose_name=_('Template
        _for_edit'))

    template_list_item = IndexedLinkedField(CESNET_TYPE.
        template_list_item, Template,
        verbose_name
        =_('
        Template
        _for_item
        _list_view
        '))

    controller = IndexedTextField(CESNET_TYPE.controller
        , verbose_name=_('Controller_class'),
        level=IndexedField.
        MANDATORY)

    rdf_types = IndexedTextField(CESNET_TYPE.rdf_types,
        verbose_name=_('RDF_types'), level=IndexedField.
        MANDATORY,
        multi_valued=True)

    federalink_model = IndexedTextField(CESNET_TYPE.
        federalink_model, verbose_name=_('Federalink_
        model_class_name'),
        level=
        IndexedField.
        MANDATORY)

class Meta:
    rdf_types = (CESNET_TYPE.ResourceType,)
```

Kód vo views.py teda z ID získa dokument, ku ktorému nájde vo Fedore

uložený správny typ. Z neho následne získa šablónu, ktorú zobrazí. Ak sa správna šablóna pre dokument alebo pole nenachádza vo Fedore, skúsi ju nájsť v aplikácii alebo použije všeobecné šablóny uložené vo `federalink_ui`, ktoré umožňujú aspoň základné zobrazenie informácií.

4.1.6.3 Cachovanie šablón

`Federalink_ui` taktiež obsahuje kód potrebný pre cachovanie výsledných šablón zložených zo šablón typu dokumentu a jednotlivých polí, keďže získanie týchto údajov z Fedory je časovo náročné. Pre získanie výslednej šablóny je potrebné množstvo dopytov na Elasticsearch a následne na Fedoru, počet dopytov závisí hlavne na komplikovanosti modelu dokumentu.

O cachovanie sa stará trieda *FedoraTemplateCache*.

Prehľad vybraných metód v triede:

```
@staticmethod
def get_resource_type(rdf_meta):
    for rdf_type in rdf_meta:
        retrieved_type = list(ResourceType.objects.
                               filter(rdf_types__exact=rdf_type))
        if retrieved_type:
            return retrieved_type[0]
    return None
```

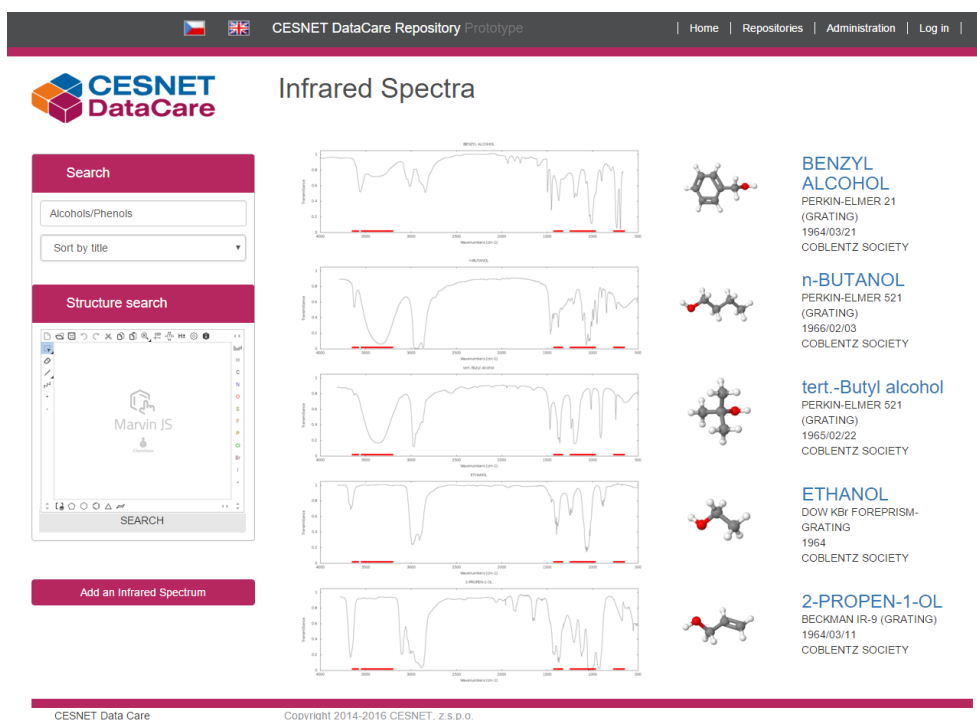
Metóda *get_resource_type* umožňuje získať správny RDF typ dokumentu.

```
@staticmethod
@simple_cache
def _get_template_string_internal(rdf_types, view_type):
    return FedoraTemplateCache._load_template(
        FedoraTemplateCache.get_template_object(rdf_types,
        view_type))

@staticmethod
def _load_template(template_object):
    if template_object is not None and template_object.
    get_bitstream() is not None:
        return template_object.get_bitstream().stream.
        read().decode("utf-8")
    return None
```

Metóda *_load_template* získa bitstream šablóny z dokumentu, ktorý máme z Fedory. Bitstream následne dekoduje a uloží ako reťazec, dáta do Fedory ukládame v kódovaní utf-8. O cachovanie tejto šablóny sa stará metóda *_get_template_string_internal*.

4. IMPLEMENTÁCIA



Obr. 4.2: Výpis dát v kolekci Infrared Spectra

Repozitár záverečných prác VŠCHT Praha pôvodne využíval šablóny uložené priamo v kóde aplikácie. Po vzniku federalink_ui ale aj tento repozitár začal využívať federalink_ui.

4.1.7 Návrh grafického rozhrania

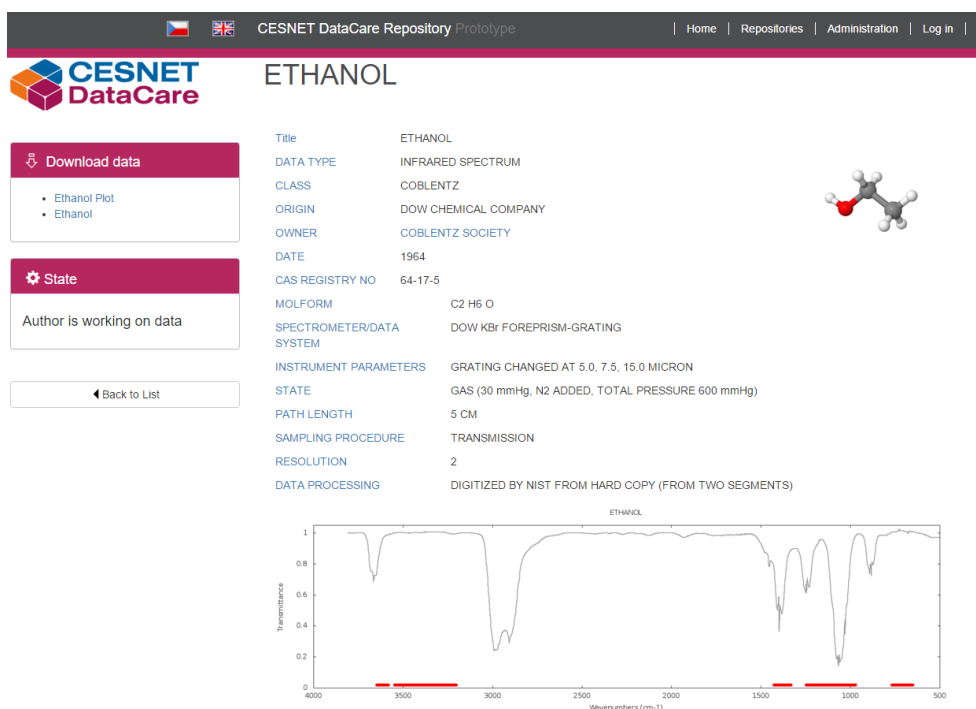
Repozitár bude nasadený ako jedna zo služieb diskového úložiska CESNET, z.s.p.o. <https://du.cesnet.cz/>, preto navrhnuté grafické rozhranie vychádza z už existujúcich služieb.

Návrh zobrazenia pre dáta organickej chémie je na obrázkoch 4.2 a 4.3

4.2 Zobrazovanie chemických dát

Technológie, ktoré máme vybrané, Fedora a Elasticsearch, umožňujú ukladanie dát a vyhľadávanie v dátach. Fedora tiež poskytuje základné webové užívateľské rozhranie, ktoré umožňuje vytváranie nových dokumentov, zobrazenie a úpravu existujúcich. Toto webové rozhranie ale nie je užívateľsky prívetivé, taktiež je potrebné zobrazovať aj dáta v inej než textovej podobe (napríklad

4.2. Zobrazovanie chemických dát



Obr. 4.3: Zobrazenie detailu konkrétnej položky (Ethanolu)

grafy, obrázky, štruktúru chemických látok,...), je preto potrebné vytvoriť nové webové rozhranie, ktoré umožní prácu s dokumentmi v repozitári.

Prácu s dokumentmi zjednoduší Fedoralink, ktorý tvorí medzivrstvu medzi Fedorou, Elasticsearchom a samotným webovým rozhraním napísaným v programovacom jazyku Python s využitím frameworku Django. Prácu so šablónami zase zjednodušuje fedoralink_ui, ktorý umožňuje získavať šablóny pre jednotlivé typy dokumentov priamo z Fedora alebo z kódu aplikácie. Prečo využívam tieto technológie a ako fungujú je popísané v predchádzajúcich častiach diplomovej práce.

Pre prácu s jednotlivými dokumentmi je potrebné mať v aplikácii vytvorený model. Modely pre základné typy dokumentov sú už vytvorené vo Fedoralinku. Napríklad trieda DCOBJECT - model pre dokumenty, ktoré obsahujú len metadáta zo schémy Dublin Core a majú priradený RDF typ DC.Object (<http://purl.org/dc/elements/1.1/Object>). V aplikácii fedoralink_ui sú zase vytvorené modely potrebné pre prácu so šablónami. Napríklad trieda ResourceType, ktorá je určená pre dokumenty s RDF typom CESNET_TYPE.ResourceType. Dokumenty s týmto RDF typom obsahujú šablóny pre zobrazenie v zozname potomkov kolekcie, zobrazenie dokumentu, editáciu dokumentu, názov a cestu k modelu v aplikácii.

Jednotlivé časti repozitára sú navrhnuté tak, aby bolo pridanie nových modelov a šablón do systému čo najjednoduchšie. Aby bolo v repozitári možné pracovať s novými typmi dokumentov (napr. s dátami z Ústavu organickej chémie VŠCHT Praha) je potrebné vytvoriť modely pre tieto dokumenty a šablóny.

4.2.1 Kolekcie vo Fedore

Dokumenty vo Fedore sú uložené buď ako kolekcie alebo binárne dáta. Dokument, ktorý je typu kolekcia môže mať pod sebou uložené ďalšie dokumenty. Vzniká teda hierarchická štruktúra, ktorá začína koreňom repozitára.

Ako prvú vec pri ukladaní nových dokumentov je potrebné vyriešiť ich hierarchické usporiadanie do rôznych kolekcí. V tejto časti diplomovej práce popisujem hierarchickú štruktúru rôznych typov dát Ústavu organickej chémie VŠCHT Praha, ktoré je potrebné ukladať, v repozitári.

4.2.1.1 Inštitúcie

Do repozitára budú môcť dáta ukladať rôzne inštitúcie. Jednotlivé dokumenty je potrebné priradiť k inštitúcii, ktorej patria. Niektoré dáta ale môžu byť zdieľané medzi viacerými inštitúciami. Preto bude vytvorená kolekcia „Institutions“, v ktorej sú uložené jednotlivé inštitúcie. Ak pri jednej inštitúcii potrebujeme evidovať aj jej súčasti, tie môžu byť uložené ako potomci inštitúcie.

4.2.1.2 Projekty

Dáta, s ktorými Ústav organickej chémie VŠCHT Praha a iné inštitúcie pracujú, sú priradené k jednotlivým projektom. Na projekte môže spolupracovať niekoľko inštitúcii a vedcov. Preto je pre projekty vytvorená kolekcia „Projects“, pod ňou sú uložené jednotlivé projekty.

4.2.1.3 Vedci

Vedec je osoba, ktorá sa podieľa na niektorom z projektov. Môže ale pracovať na viacerých a taktiež pracovať pre viacero inštitúcii. Preto je vytvorená samostatná kolekcia „Scientists“, pod ktorou sú uložené jednotlivé osoby.

4.2.1.4 Laboratórne denníky

Laboratórne denníky Ústavu organickej chémie VŠCHT Praha, prípadne aj iných inštitúcii sú uložené pod kolekciou „Laboratory journals“.

4.2.1.5 Reakcie

Reakcie, ktoré boli pozorované v rámci niektorého laboratórneho denníka sú uložené pod daným denníkom. Detaily (ako použité množstvo, koncentrácia,...) o reaktantoch a produktoch, teda chemických látkach, ktoré boli pri reakcii použité alebo pri nej vznikli, sú uložené pod touto reakciou.

4.2.1.6 Chemické látky

Samotné dáta (názov chemikálie, jej štruktúra, vlastnosti,...) o chemikáliach, ktoré môžu byť používané v rôznych reakciách, sú uložené v kolekcii „Chemicals“.

4.2.1.7 Analytické dáta

NMR, IR, hmotnostné spektrá jednotlivých chemických látok a iné analytické dáta sú uložené v samostatnej kolekcii „Analytical data“. Všetky zdrojové binárne dáta (výstupy zo zariadení, serializované dáta z Open Enventory,...) sú uložené v kolekcii „Source data“.

Hierarchickú štruktúru kolekcii je možné vidieť na obrázku 4.4

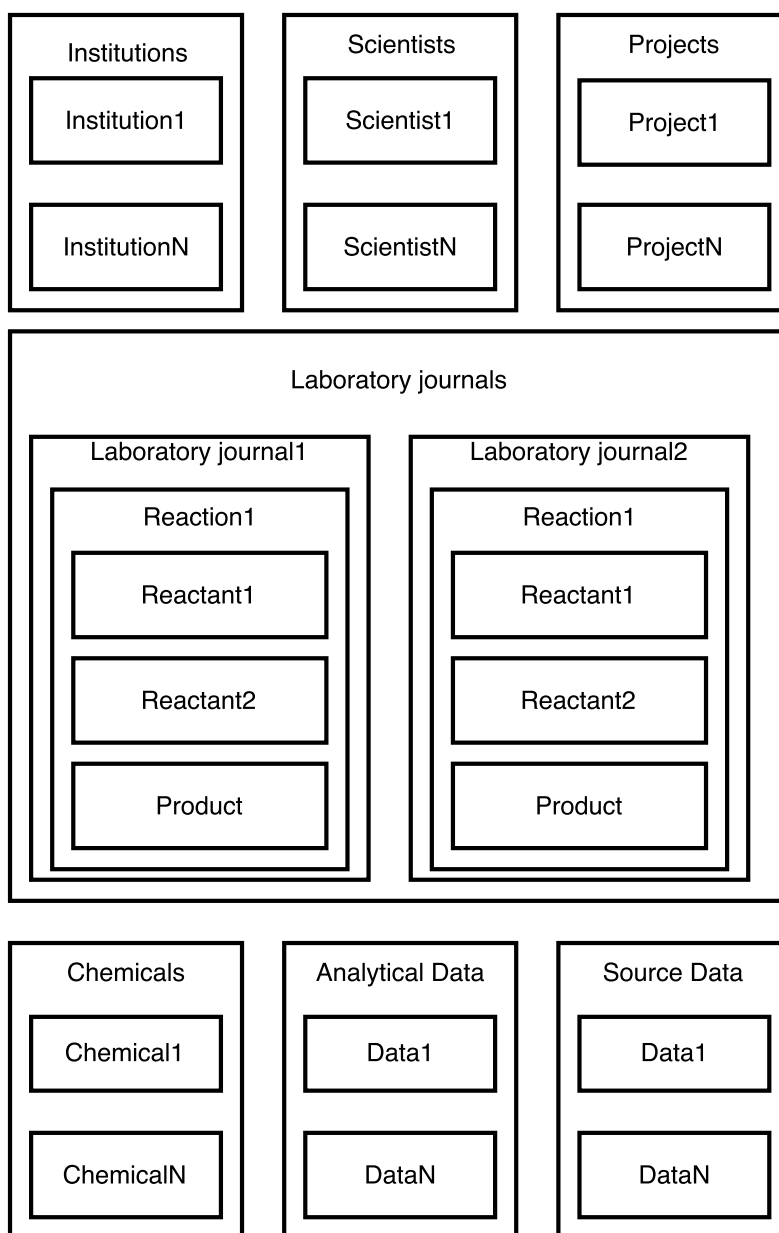
4.2.2 Modely

Modely, teda triedy pre jednotlivé dokumenty sú navrhnuté tak, aby boli čo najvšeobecnejšie, ale zároveň pokrývali všetky potrebné dáta, ktoré chcú výskumníci z Ústavu organickej chémie VŠCHT Praha v repozitári ukladať. Zároveň bolo pri ich návrhu myslené na potrebu importu dát z aplikácie Open Enventory.

V repozitári potrebujeme ukladať informácie o vedcoch, ktorí na jednotlivých projektoch pracujú. Preto bola vytvorená trieda ScientistPerson (diagram triedy je možné vidieť na obrázku 4.5).

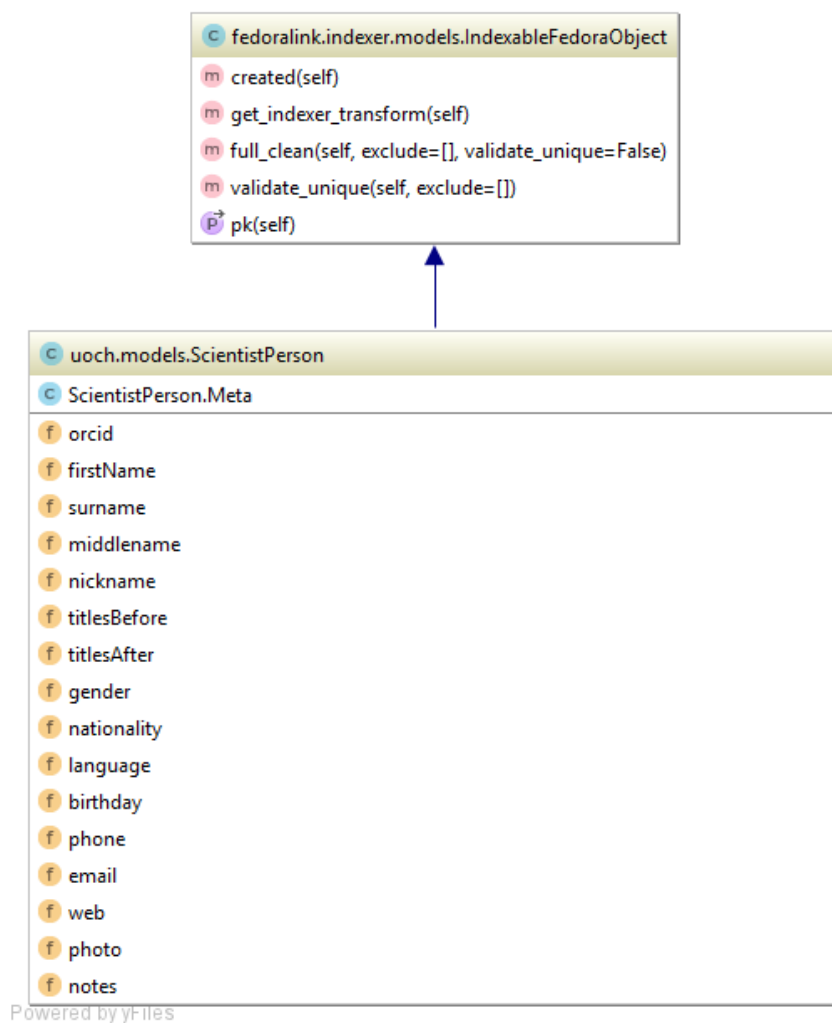
Pre projekty je vytvorená trieda Project, ktorá okrem názvu projektu má aj odkaz na členov projektu (z modelu ScientistPerson) a tiež na inštitúcie, ktoré sa na projekte podieľajú. Trieda Institution umožňuje prácu so základnými informáciami o inštitúcii.

V laboratórnych denníkoch popisované reakcie má na starosť trieda Reaction (diagram triedy je možné nájsť na obrázku 4.7). Medzi uložené údaje patrí popisi reakcie, pozorovanie, čas začiatku a trvanie reakcie. Reaktanty a produkty zapísané vo formáte SMILES (zjednodušený jednoriadkový zápis štruktúry). Samotné chemické látky v reakcii ale obsluhuje model ChemicalInReaction (diagram je možné vidieť na obrázku 4.6), kde sú uvedené hodnoty ako váha, objem koncentrácia použitej látky v reakcii, či je v reakcii reaktant alebo produkt. Samotné chemické látky má na starosť trieda Chemical (4.6)). Pri chemických látkach ukladáme hodnoty ako je štandardný názov prvku, CAS number, štruktúru chemického prvku.



Obr. 4.4: Hierarchická štruktúra kolekcí vo Fedore

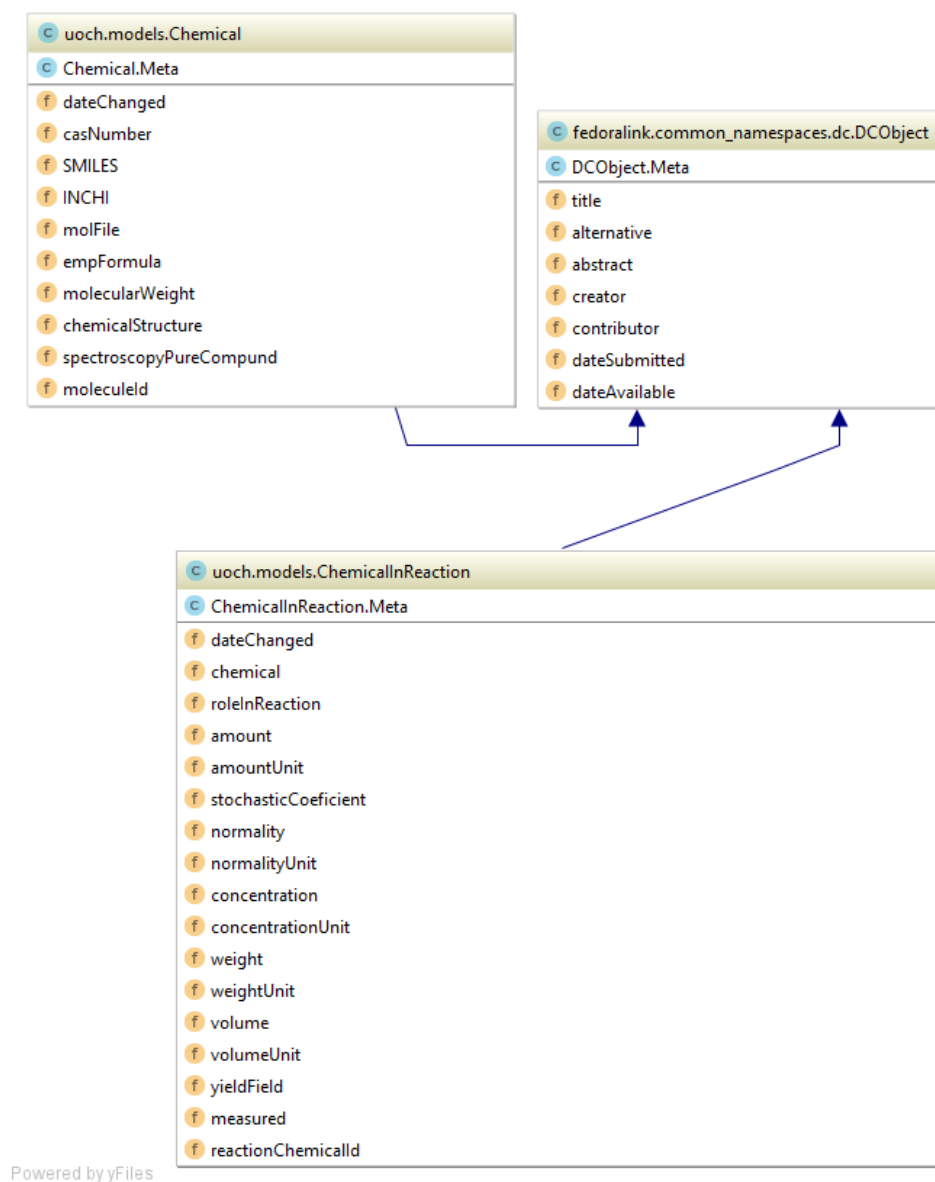
Analytické dáta k reakciám a chemickým látkam obsluhujú modely *AnalyticalMethod*, *SpectroscopyMethod* (napríklad NMR spektrá, IR spektrá, hmotnostné spektrá) a binárne dáta obsluhuje model *SourceData* a *OpenInventorySourceData* (diagramy sú zobrazené na obrázku 4.8). Za zdrojové dáta, ktoré ma na starosť trieda *SourceData*, považujeme napríklad dátové súbory s výsledkami analýz. *OpenInventorySourceData* obsluhuje všetky serializované



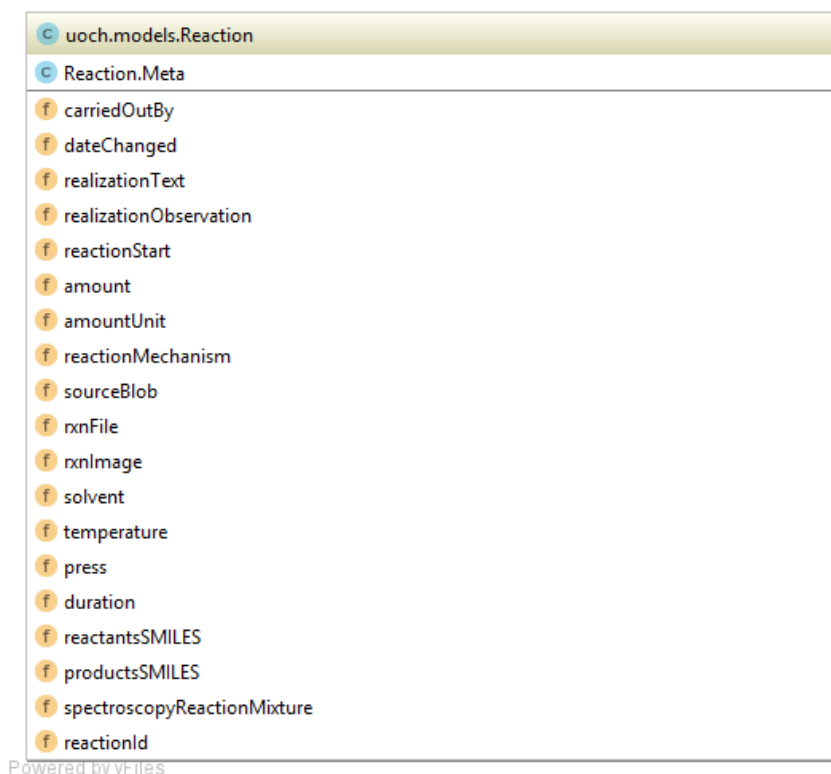
Powered by yFiles

Obr. 4.5: Model osoby - vedca

4. IMPLEMENTÁCIA



Obr. 4.6: Modely chemických prvkov a chemických prvkov v reakcii



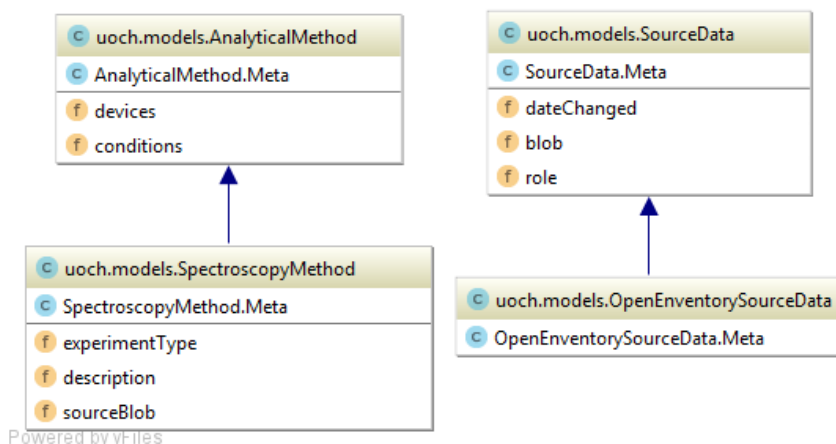
Obr. 4.7: Model reakcie

dáta k reakcii, získané importom z OpenEnventory.

4.3 Automatický import dát

Aby sme čo najviac zjednodušili vkladanie nových dát do repozitára, je potrebné upraviť niektorý z nástrojov na zber a organizáciu chemických dát, ktorý používajú výskumníci z Ústavu organickej chémie VŠCHT Praha. Pre účely diplomovej práce bol po dohode s výskumníkmi ako zdroj dát pre import vybraný nástroj Open Enventory.

Výskumníci na Ústave organickej chémie VŠCHT Praha chcú mať možnosť importovať do repozitára vybrané reakcie. K importu nemá dochádzať plne automaticky, ale až po potvrdení užívateľom. Pre tento účel je najlepšie implementovať tlačidlo umiestnené priamo vo webovej aplikácii Open Enventory. Aby bolo možné aplikáciu upraviť, je potrebné vedieť ako funguje, aké sú vzťahy medzi tabuľkami v databáze a nájsť vhodné umiestnenie pre tlačidlo.



Obr. 4.8: Modely analytických dát a zdrojových dát

4.3.1 Schéma databázy Open Enventory

Celá schéma databázy je na priloženom CD vo formáte UML aj SVG. V tejto časti sú popísané a okomentované databázové tabuľky, v ktorých sú uložené dáta z aplikácie Open Enventory, ktoré chceme importovať do repozitára alebo ich k tomuto importu potrebujeme.

V databázovej tabuľke `lab_journal`, sú uložené základné údaje o laboratórnom denníku. Pre naše účely budú ďalej dôležité stĺpce `person_id` a primárny kľúč `lab_journal_id`.

V rámci laboratórnych denníkov sú vytvorené jednotlivé projekty užívateľov.

Samotné dáta meraní, popis a priebeh reakcií sú uložené v databázovej tabuľke `reaction`, jej schému si je možné pozrieť na obrázku 4.9. Tá je cez stĺpec `project_id` prepojená s tabuľkou `project` a cez stĺpec `lab_journal_id` s tabuľkou `lab_journal`. Zároveň má každá reakcia jedinečné poradové číslo v denníku uložené v stĺpci `nr_in_lab_journal`.

Popis reakcie je uložený v stĺpci `realization_text_fulltext`, pozorovanie v stĺpci `realization_observation_fulltext`, množstvo výslednej látky v stĺpci `ref_amount` a jednotka, v akej je toto množstvo uvedené, v stĺpci `ref_amount_unit`. Ďalšie namerané hodnoty sú ako binárne hodnoty uložené v stĺpcoch `rxn_file_blob`, `rxn_gif_file` - chemická rovnica reakcie vo formáte GIF, `rxn_svg_file` - obrázok vo formáte SVG. RXN je formát pre popis reakcie, ktorý pozostáva z bloku reaktantov, produktov a prípadne (nie veľmi zvyčajne) aj bloku agentov.

Ďalšie dáta ako doba trvania reakcie, použité rozpúšťadlo a jeho koncentrácia alebo teplota, pri ktorej reakcia prebiehala, sú uložené v tabuľke `reaction_property`. V stĺpci `reaction_property_name` je uložený názov vlastnosti (napríklad „solvent“ teda rozpúšťadlo, „temperature“ teda teplota, ...) a v

reaction	
reaction_id	int(11)
project_id	int(10) unsigned
reaction_type_id	int(10) unsigned
reaction_prototype	int(10) unsigned
reaction_prototype_db_id	int(11)
lab_journal_id	int(10) unsigned
nr_in_lab_journal	smallint(6)
realization_text	mediumtext
realization_text_fulltext	mediumtext
realization_observation	mediumtext
realization_observation_fulltext	mediumtext
rxn_smiles	text
rxnfile_blob	mediumblob
rxn_gif_file	mediumblob
rxn_svg_file	mediumblob
reaction_carried_out_by	tinytext
reaction_title	tinytext
reaction_quality	double
reaction_started_when	datetime
status	enum('planned', 'started', 'performed', 'completed', 'approved', 'printed')
ref_amount	double
ref_amount_unit	tinytext
reaction_uid	varbinary(128)
reaction_ext_archive_id	tinytext
reaction_shared	tinyint(1)
reaction_disabled	tinyint(1)
reaction_created_by	tinytext
reaction_created_when	datetime
reaction_created_hashver	int(11)
reaction_created_md5	varbinary(128)
reaction_changed_by	tinytext
reaction_changed_when	datetime
reaction_changed_hashver	int(11)
reaction_changed_md5	varbinary(128)

Obr. 4.9: Štruktúra databázovej tabuľky reaction

stĺpci `reaction_property_value` je uložená samotná hodnota.

V tabuľke `analytical_data` (schéma je na obrázkoch 4.10 a 4.11) sú uložené rôzne analýzy reakcie a chemických prvkov. Analytické dáta môžu byť H NMR spektrá, C NMR spektrá, teda spektroskopia nukleárnej magnetickej rezonancie s využitím vodíka alebo uhlíka; infračervené spektrá, hmotnostné spektrá. Do Open Enventory mohli byť tieto analýzy uložené v binárnom formáte (väčšinou ako dokument vo formáte PDF alebo obrázkov vo formáte TIF, RAW), v databázovej tabuľke sa okrem binárnych hodnôt nachádzajú grafy (obrázky) k jednotlivým analytickým dátam. Tieto grafy/obrázky sú uložené v stĺpci `analytical_data_graphics_blob`, textová interpretácia dát (ak je možná) v stĺpci `analytical_data_interpretation`, komentár k dátam v stĺpci `analytical_data_comment`. Pôvodné dáta su uložené v stĺpci `analytical_data_raw_blob`, kde sú súbory uložené do archívu TAR a komprimované metódou gzip. Prepojenie na tabuľku `reaction_chemical` je cez stĺpec `reaction_chemical_id`.

V tabuľke `reaction_chemical` (schéma je na obrázkoch 4.12 a 4.13) sú uložené informácie o reaktantoch a produktoch. Môže tu byť uvedený štandardný názov prvku v stĺpci `standard_name`, tzv. SMILES teda zjednodušený jednoriadkový zápis štruktúry (uložené v stĺpci `smiles`). Binárne hodnoty popisujúce prvok v stĺpcoch `molfile_blob` (MDL Molfile je súborový formát, ktorý umožňuje ukladať informácie o atónoch, spojeniach, prepojeniach a polohe molekúl), `molecule_serialized`, `gif_file` - obsahuje obrázok štruktúry prvku vo formáte GIF, `svg_file` - štruktúra prvku na obrázku vo formáte SVG.

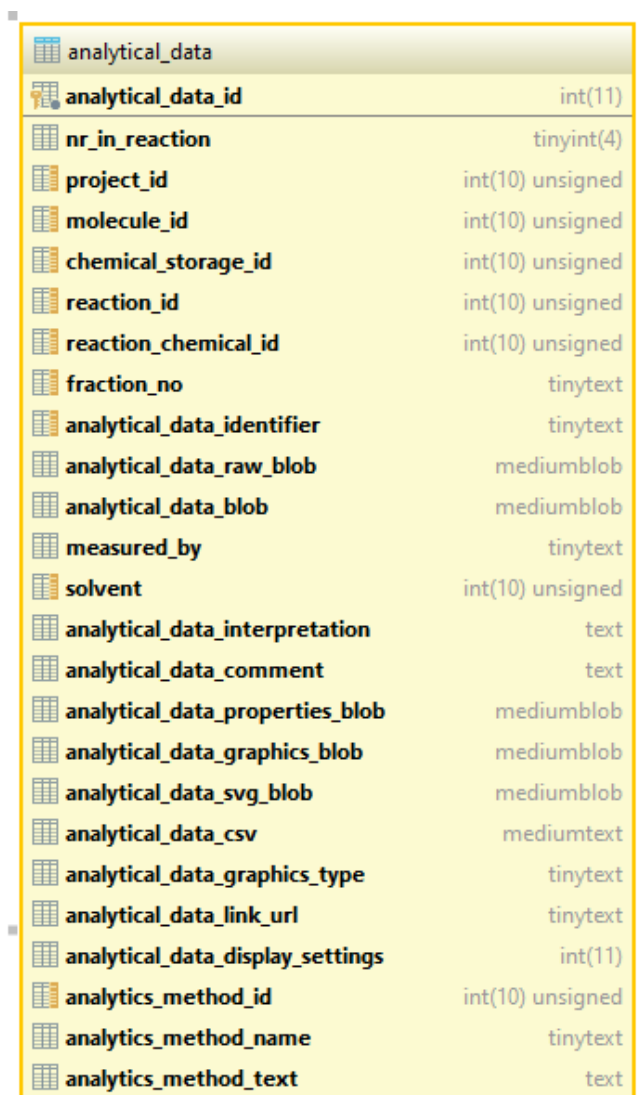
Chemický vzorec prvku je uložený v stĺpci `emp_formula`, molekulárna hmotnosť v stĺpci `mw`, poradie reaktantov a produktov je v stĺpci `nr_in_reaction`. Či ide o reaktant alebo produkt, sa dozvieme zo stĺpca `role`. Stochastický koeficient v stĺpci `stoch_coeff`, hmotnosť v stĺpci `m_brutto`, jednotka, v akej je hmotnosť uvedená, je v stĺpci `mass_unit`, objem v stĺpci `volume` a jednotka objemu v stĺpci `volume_unit`. Koncentrácia v stĺpci `rc_amount` a jednotka, v akej je koncentrácia uvedená, je v stĺpci `rc_amount_unit`. V prípade produktov je v stĺpci `yield` uložené reálne získané množstvo v % a z `rc_amount` sa dá zistiť teoreticky získateľná koncentrácia.

4.3.2 Implementácia automatického importu

Štruktúru databázových tabuliek a prepojenia medzi nimi už poznáme, ďalej sa bolo potrebné rozhodnúť na spôsobe implementácie automatického importu dát z Open Enventory do repozitára.



















Do úvahy prichádzajú dve možnosti:

1. Implementácia v programovacom jazyku PHP, ktorá by bola priamo súčasťou Open Enventory. Autor Open Enventory ale na získavanie dát využíva volania z javascriptu, kde jednotlivé hodnoty získava postupne. Nemá tu teda v PHP pripravené objekty ani funkcie, ktoré by získanie



analytical_data	
analytical_data_id	int(11)
nr_in_reaction	tinyint(4)
project_id	int(10) unsigned
molecule_id	int(10) unsigned
chemical_storage_id	int(10) unsigned
reaction_id	int(10) unsigned
reaction_chemical_id	int(10) unsigned
fraction_no	tinytext
analytical_data_identifier	tinytext
analytical_data_raw_blob	mediumblob
analytical_data_blob	mediumblob
measured_by	tinytext
solvent	int(10) unsigned
analytical_data_interpretation	text
analytical_data_comment	text
analytical_data_properties_blob	mediumblob
analytical_data_graphics_blob	mediumblob
analytical_data_svg_blob	mediumblob
analytical_data_csv	mediumtext
analytical_data_graphics_type	tinytext
analytical_data_link_url	tinytext
analytical_data_display_settings	int(11)
analytics_method_id	int(10) unsigned
analytics_method_name	tinytext
analytics_method_text	text

Obr. 4.10: Štruktúra databázovej tabuľky analytical_data, 1. časť

	analytics_type_id	int(10) unsigned
	analytics_type_name	tinytext
	analytics_type_code	tinytext
	analytics_type_text	text
	analytics_device_id	int(10) unsigned
	analytics_device_name	tinytext
	analytics_device_driver	tinytext
	analytical_data_uid	varbinary(128)
	analytical_data_ext_archive_id	tinytext
	analytical_data_shared	tinyint(1)
	analytical_data_created_by	tinytext
	analytical_data_created_when	datetime
	analytical_data_created_hashver	int(11)
	analytical_data_created_md5	varbinary(128)
	analytical_data_changed_by	tinytext
	analytical_data_changed_when	datetime
	analytical_data_changed_hashver	int(11)
	analytical_data_changed_md5	varbinary(128)

Powered by yFiles

Obr. 4.11: Štruktúra databázovej tabuľky analytical_data, 2. časť

laboratórnych denníkov uľahčili. Kód aplikácie je navyše prehľadný s komentármi a aj časťou kódu v nemeckom jazyku.

- Implementácia samostatného skriptu s napojením priamo na databázu, ktorý by sa staral o automatický import dát do repozitára. V samotnom Open Enventory je v tomto prípade potrebné umiestniť tlačidlo, ktoré spustí tento skript a odovzdá mu parametre potrebné pre import (teda id reakcie alebo laboratórneho denníka, ktorý chce užívateľ importovať do repozitára).

Rozhodol som sa pre druhú možnosť, pričom využívam programovací jazyk Python a framework Django. Django, po nastavení pripojenia k databáze, umožňuje automatické vytvorenie modelov pre tabuľky v databáze. Vzhľadom na nezadefinované cudzie kľúče a ďalšie zistené chyby, bolo pre ľahšiu prácu s modelmi potrebné upraviť vygenerované triedy.

4.3. Automatický import dát

reaction_chemical	
reaction_chemical_id	int(11)
project_id	int(10) unsigned
reaction_id	int(10) unsigned
from_reaction_id	int(10) unsigned
from_reaction_chemical_id	int(10) unsigned
other_db_id	int(11)
molecule_id	int(10) unsigned
chemical_storage_id	int(10) unsigned
chemical_storage_barcode	varbinary(20)
mixture_with	int(10) unsigned
standard_name	tinytext
package_name	tinytext
cas_nr	tinytext
smiles	text
smiles_stereo	text
inchi	text
molfile_blob	mediumblob
molecule_serialized	mediumblob
gif_file	mediumblob
svg_file	mediumblob
emp_formula	tinytext
mw	double unsigned
density_20	double
rc_conc	double
rc_conc_unit	tinytext
safety_r	tinytext
safety_h	tinytext
safety_s	tinytext
safety_p	tinytext
safety_sym	tinytext
safety_sym_ghs	tinytext
nr_in_reaction	tinyint(4)
addition_delay	time
addition_duration	time
role	enum('reactant', 'reagent', 'solvent', 'catalyst', 'intermediate', 'product', 'other')
stoch_coeff	double
rc_purity	double
m_brutto	double
m_tara	double
mass_unit	tinytext
volume	double
volume_unit	tinytext
rc_amount	double
rc_amount_unit	tinytext
gc_yield	double
yield	double
measured	enum('mass', 'volume', 'amount')
colour	tinytext
consistency	tinytext
description	text

Obr. 4.12: Štruktúra databázovej tabuľky reaction_chemical, 1. časť

4. IMPLEMENTÁCIA



fingerprint1	int(11)
fingerprint2	int(11)
fingerprint3	int(11)
fingerprint4	int(11)
fingerprint5	int(11)
fingerprint6	int(11)
fingerprint7	int(11)
fingerprint8	int(11)
fingerprint9	int(11)
fingerprint10	int(11)
fingerprint11	int(11)
fingerprint12	int(11)
fingerprint13	int(11)
fingerprint14	int(11)
fingerprint15	int(11)
fingerprint16	int(11)
reaction_chemical_shared	tinyint(1)
reaction_chemical_created_by	tinytext
reaction_chemical_created_when	datetime
reaction_chemical_created_hashver	int(11)
reaction_chemical_created_md5	varbinary(128)
reaction_chemical_changed_by	tinytext
reaction_chemical_changed_when	datetime
reaction_chemical_changed_hashver	int(11)
reaction_chemical_changed_md5	varbinary(128)

Obr. 4.13: Štruktúra databázovej tabuľky reaction_chemical, 2. časť

4.4 Kontrola stavov a prístupové práva

Fedora umožňuje nastaviť práva pre jednotlivé dokumenty pomocou ACL. Takýmto spôsobom vieme nastaviť, kto môže daný dokument zobraziť, kto ho môže upraviť... Keďže má byť repozitár pre užívateľov čo najjednoduchší, nie je takéto nastavovanie prístupových práv ideálne. Tieto práva by bolo potrebné nastavovať a hlavne meniť pre každý dokument individuálne. Tak by mohlo dochádzať k častým chybám s nesprávne nastavenými oprávneniami. Preto bol navrhnutý spôsob kontroly prístupových práv cez stavy. Tieto určujú, v akom stave alebo stavoch sa konkrétny dokument nachádza.

Napríklad môže ísť o schvaľovací proces. Vložené dáta nesmú byť zverejnené hneď. Je potrebné, aby prešli procesom schvaľovania cez viacero ľudí. Prístup k tomuto dokumentu má na začiatku osoba, ktorá dokument vytvorila, tá môže požiadať o jeho zverejnenie. Zmení sa stav a nové nastavenie prístupových práv umožní, aby dáta videli aj osoby, ktoré majú právo schváliť zverejnenie. Po schválení všetkými potrebnými osobami dôjde opäť ku zmene stavu a prístupu k týmto dátam. Keď sa dokument nachádza v stave „zverejnený“, môžu si ho prezerať všetci. Právo na editáciu ale ostane len

pôvodnému autorovi prípadne editorovi. Po úprave bude ale opäť potrebné požiadať o schválenie.

Možnosti Fedory, nutné úpravy a detailnejšie informácie o stavoch sú popísané nižšie.

4.4.1 Prístupové práva vo Fedore

Fedora 4 obsahuje modul WebAC Authorization Delegate, ktorý je implementáciou W3C návrhu decentralizovaného autorizačného mechanizmu na základe RDF. Tento mechanizmus sa nazýva WebAccessControl <https://www.w3.org/wiki/WebAccessControl>.

4.4.1.1 WebAccessControl

Je decentralizovaný systém umožňujúci rôznym užívateľom a skupinám rozdielne prístupy k dokumentom na základe identifikácie užívateľov a skupín cez HTTP URI.

Takýmto spôsobom je možné nastaviť prístup k dokumentu uloženom na jednom mieste užívateľom a skupinám hostovaným na inom mieste. Užívateľ tak nemusí mať vytvorený profil na mieste, kde je dokument uložený.

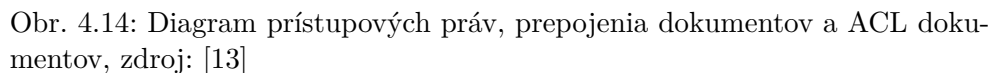
Každý dopyt na webový zdroj vráti HTTP dokument obsahujúci hlavičku s odkazom na ACL dokument, ktorý popisuje prístupové práva pre daný dokument (prípadne pre iné dokumenty). [13]

Nastavenie prístupových práv, prepojenie dokumentov a ACL dokumentov je možné vidieť na diagrame znázornenom na obrázku 4.14. Podľa diagramu je umožnený prístup na čítanie všetkým, kto pristupujú k dokumentu /2013/card, ale k dokumentu /2013/protected majú prístup len užívatelia, ktorí patria do skupiny conf.example (teda účastníci konferencie).

ACL je zoznam oprávnení priradených k nejakému dokumentu. Pomocou ACL môžeme určiť, ktoré osoby (parameter *agent*) alebo skupiny (parameter *agentClass*) majú mať prístup k danému dokumentu. Tento dokument, prípadne dokumenty sú určené parametrom *accessTo* alebo *accessToClass*. Ako trieda pre všetkých užívateľov sa štandardne používa foaf:Agent a znamená, že dokument je verejne dostupný. Typ prístupu je určený parametrom *mode*. Typy prístupu sú popísané v tabuľke 4.1.

Tabuľka 4.1: Typy prístupu v ACL

Read	Oprávnenie na čítanie obsahu
Write	Oprávnenie na prepísanie obsahu
Append	Oprávnenie na pridanie informácie (na koniec už existujúcej, nedáva právo na zmazanie)
Control	Oprávnenie na nastavenie prístupu na seba (prístup k danému ACL)

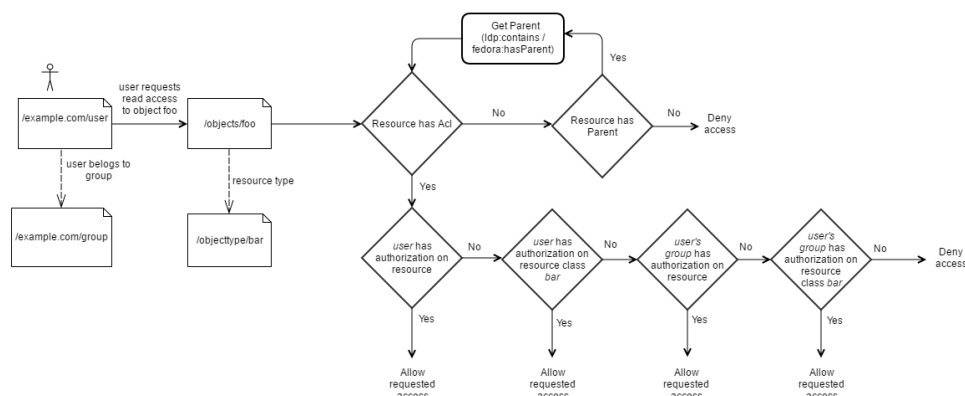


```
@prefix acl: <http://www.w3.org/ns/auth/acl#> .

◇ a acl:Authorization;
  acl:accessTo <$FCREPOREL/administration/states>;
  acl:agentClass <$FCREPOREL/groups/groupAll>;
  acl:mode acl:Read.
```

4.4.1.2 Proces získania oprávnení pre dokument vo Fedore

¹Turtle je definícia textovej syntaxe, ktorá umožňuje zápis celého RDF grafu v textovej podobe. [9]



Obr. 4.15: Proces získania oprávnení vo Fedore, zdroj: [14]

Ak nie, prejde všetkých rodičov až po koreň. Pokiaľ ACL nenájde, prístup k danému dokumentu zakáže. V opačnom prípade zistí, či má užívateľ prístup na daný dokument, na daný typ dokumentov alebo či má oprávnenie užívateľova skupina. Ak nenájde žiadne oprávnenie, prístup zakáže, inak vráti dopytovaný dokument.

4.4.2 Stavy

Aby bola práca užívateľov s repozitárom čo najviac zjednodušená, každý dokument v repozitári môže mať definovaný stav. Ten následne určuje oprávnenia pre prístup k danému dokumentu.

4.4.2.1 Stav uloženého dokumentu v repozitári

Stav je k dokumentu priradený pomocou parametru state:State. V jednom okamžiku môže mať dokument priradených aj viacero stavov (parameter state:State je v takom prípade použitý viackrát).

Takýto stav môže napríklad vyjadrovať stav schvaľovania publikovania dát. Nový dokument, ktorý chce autor zverejniť, je potrebné schváliť vedúcim pracovníkom. Dokument sa nachádza v stave, kedy čaká na zverejnenie. Prístup k nemu v danej chvíli má jeho autor a osoba, ktorá má zverejnenie potvrdiť.

Stavy, v ktorých sa môže nejaký dokument nachádzať, sú uložené v kolekcii stavov. Tá je k danému dokumentu priradená parametrom state:stateControl.

Príklad dokumentu s priradenými stavmi:

```

@prefix acl: <http://www.w3.org/ns/auth/acl#>.
@prefix state: <http://cesnet.cz/ns/repository/state#>.
@prefix dc: <http://purl.org/dc/elements/1.1/>.
  
```

```
<> acl:accessControl </fcrepo/rest/acl>;
    state:stateControl </fcrepo/rest/states>;
    state:State </fcrepo/rest/states/wg1_approving>,
                </fcrepo/rest/states/wg2_approving>;
    dc:title "Hello ,□World!" .
```

4.4.2.2 Stav

Stav je dokument typu `state:State`, ktorý je uložený v kolekcii stavov v repozitári. Pre stav sú definované prístupové práva (ACL). Aby bolo možné zmeniť stav, v akom sa nejaký dokument nachádza, je potrebné taktiež definovať povolené zmeny stavov.

- `state:defaultAccessControl` definuje ACL pre dokumenty s týmto stavom.
- `state:allowedStateTransitions` definuje povolené zmeny stavov. Ako hodnota tohto parametra je dokument typu `state:Transition`.

4.4.2.3 Prechod medzi stavmi

Dokument typu `state:Transition` definuje prechod medzi dvoma stavmi a práva, kto môže danú zmenu realizovať.

`State:targetState` určuje cieľový stav prechodu. ACL priradené na dokument typu `state:Transition` určuje, kto môže zmenu realizovať.

4.4.3 Stavy vo Fedore

Vo Fedore žiadne stavy ani nástroje na kontrolu stavov a prechodov medzi nimi neexistujú. Aplikáciu je teda potrebné upraviť tak, aby takúto kontrolu umožňovala.

Pri každom dopyte do Fedory potrebujeme skontrolovať, či dochádza k zmene stavu dokumentu. V prípade, ak áno, je potrebné skontrolovať oprávnenia a zmenu povoliť alebo zamietnuť. K ukladaniu upravených dokumentov dochádza vo funkcii `patchResourcewithSparql` v triede `ContentExposingResource`. K volaniu tejto funkcie ale dochádza z objektu, ktorý je typu `FedoraLdp`. Trieda `FedoraLdp` dedí z triedy `ContentExposingResource`. Obe sa nachádzajú v module `fcrepo-http-api`.

Fedora sa stále rýchlo vyvíja. Aktuálne vývojári pracujú aj na API-X, ktoré umožní rozširovanie kódu Fedory. Momentálny stav vývoja API-X však neumožňuje jeho využitie pre úpravu potrebného kódu. Upravovať priamo kód modulu `fcrepo-http-api` by spôsobilo komplikácie pri aktualizácii Fedory, ale taktiež pri šírení takto upravenej verzie. Vytvoril by som v podstate novú vetvu vývoja Fedory, pričom by bolo pri vydaní novej verzie Fedory robiť

zlúčenie medzi vetvami. Čo by mohlo byť vzhľadom na možné zmeny kódu aj v časti, ktorú potrebujem upraviť, komplikované.

Menej často sa menia rozhrania metód a ich parametre, to je možné využiť, ak existuje nástroj, ktorý umožní rozšírenie existujúceho kódu mimo kódu Fedory. Riešením je vytvorenie Java aplikácie využívajúcej rozšírenie AspectJ. Pri správnom nastavení Mavenu vytvorí nový modul fcrepo-http-api obsahujúci upravený kód. V skompilovanej Fedore následne stačí vymeniť súbor fcrepo-http-api.jar za novovytvorený fcrepo-http-api-with-states.jar.

4.4.3.1 AspectJ

V programovaní sa často využíva proces oddelenia zodpovedností (v angličtine Separation of concerns - SoC), ide o snahu rozdeliť program do samostatných častí, z ktorých každá má inú funkciu. Funkcie a kódy jednotlivých častí by sa mali čo najmenej prekrývať. O takéto oddelenie zodpovedností sa snaží aj aspektovo orientované programovanie (AOP).

Vysvetlenie používaných termínov:

- Extension methods - umožňujú pridať metódy, premenné a rozhrania do existujúcej triedy v rámci aspektu.
- Pointcut - umožňuje definovať tzv. join points, teda presne definované miesta v bežiacom programe (ako sú volania metód, inicializácia objektov alebo prístup k premenným).
- Advices - kód, ktorý sa spustí po splnení podmienky špecifikovanej v pointcute. Tento kód môže byť spustený pred, po alebo okolo definovaného miesta v bežiacom programe.
- Weaving - proces vkladania aspektov do aplikácie.
- Aspekt - je kombináciou advice a pointcutu, umožňuje teda spustiť kód na presnom mieste v bežiacей aplikácii.

Vďaka aspektom môžeme upraviť chovanie programu na presne definovanom mieste. Takéto aspekty sa často využívajú k logovaniu, umožňujú spúšťať jeden kód na viacerých miestach programu, pričom nie je potrebné meniť samotný kód programu. V prípade potrebných zmien stačí kód pre logovanie upraviť na jednom mieste. [15] V mojom prípade bude vďaka aspektom možné upraviť kód pre ukladanie upravených objektov do Fedory tak, aby kontroloval zmenu stavov a oprávnenia pre ich zmenu.

AspectJ <http://www.eclipse.org/aspectj/> je rozšírenie využívajúce aspektovo orientované programovanie pre jazyk Java. Toto rozšírenie umožňuje využívať aspekty.

Konfigurácia AspectJ je uložená v súbore aop.xml

```
<!DOCTYPE aspectj PUBLIC "-//AspectJ//DTD//EN" "http://
www.eclipse.org/aspectj/dtd/aspectj.dtd">

<aspectj>

    <weaver options="-verbose -showWeaveInfo">
        <include within="org.fcrepo.http.api.FedoraLdp"/>
        <include within="org.fcrepo.http.api.
            ContentExposingResource"/>
        <include within="org.fcrepo.http.api.
            FedoraBaseResource"/>
    </weaver>

    <aspects>
        <aspect name="cz.cesnet.fcrepo.interceptor.
            PatchWrapAspect"/>
    </aspects>

</aspectj>
```

V časti weaver je nastavený spôsob weavingu. Potrebujeme mať prístup k triedam FedoraLdp, ContentExposingResource a FedoraBaseResource, ktoré od seba dedia.

Kód aspektu, ktorý sa spustí pri každom volaní metódy patchResource-withSparql:

```
@SuppressWarnings("PackageAccessibility")
public privileged aspect PatchWrapAspect {
    @Inject
    protected Session session;

    public pointcut wrapPatch(FedoraResource resource,
        String requestBody, RdfStream resourceTriples) :
        execution(* patchResourcewithSparql(FedoraResource,
            java.lang.String, RdfStream)) && this(org.fcrepo.
                http.api.FedoraLdp) && args(resource, requestBody
                    , resourceTriples);

    Object around(FedoraResource resource, String
        requestBody, RdfStream resourceTriples) :
        wrapPatch(resource, requestBody, resourceTriples)
    {
        try {
```

```

        ContentExposingResource currentResource = (
            ContentExposingResource) thisJoinPoint.
            getThis();
        HttpSession session = currentResource.
            session;
        ContainerRequestContext request = (
            ContainerRequestContext) currentResource.
            request;
        IdentifierConverter<Resource, FedoraResource
        > translator = currentResource.translator
            ();

        final UpdateRequest update_request =
            UpdateFactory.create(requestBody,
                translator.reverse().convert(
                    resource).toString());

        StateAuthorizationDelegate sad = new
            StateAuthorizationDelegate(this.session,
                translator,
                session.getFedoraSession().getUserId
                    ());

        requestBody+= sad.checkState(resource,
            update_request);

        return proceed(resource, requestBody,
            resourceTriples);
    }
    catch (RuntimeException e){
        e.printStackTrace();
        System.out.println(e.getMessage());
        throw e;
    }
}
}

```

4.4.4 Kontrola stavov

S využitím aspektu po nahradení pôvodného kódu potrebujeme zistiť, či sa niekto pokúša zmeniť StateControl upravovaného dokumentu. V prípade ak to robí užívateľ, ktorý nemá oprávnenie na zmenu, aplikácia vráti chybový HTTP kód. Ak StateControl nie je k dokumentu priradený, ale užívateľ sa

pokúša priradiť k dokumentu stavy, aplikácia taktiež vráti chybový HTTP kód.

Ak je StateControl upraveného dokumentu a pôvodného rovnaký, porovnáme stavy. Ak došlo k zmene v stavoch, je potrebné získať všetky povolené prechody z pôvodných stavov. Keďže operáciu, pri ktorej získavame dokumenty typu StateTransition, robí aplikácia pod užívateľom, ktorý poslal dopyt, získa len prechody, ktoré má daný užívateľ povolené.

Po získaní možných cieľových stavov z týchto prechodov aplikácia zistí, či sa medzi nimi nachádza každý nový cieľový stav. Ak niektorý z nových stavov nie je medzi povolenými cieľovými stavmi pre daného užívateľa, aplikácia vráti chybový kód.

Ak sú povolené všetky nové stavy, aplikácia upraví pôvodný dopyt tak, aby sa všetky pôvodné stavy zmazali.

Záver

Literatúra

- [1] CESNET, z.s.p.o.: *Oddělení datových úložišť CESNET*. 2016, [cit. 2017-04-22]. Dostupné z: <https://du.cesnet.cz/cs/start>
- [2] Strnad, M.: Svěřte svá data vhodnému médiu – díl 1. V: *LinuxEXPRES [online]*, november 2013, [cit. 2016-11-07]. Dostupné z: <https://www.linuxexpres.cz/praxe/sverte-sva-data-vhodnemu-mediu-dil-1>
- [3] Český normalizační institut: *ČSN ISO 8459-5 (01 0175) Informace a dokumentace - sborník bibliografických datových prvků. Část 5, Datové prvky pro výměnu katalogizačních dat a metadata*. 2004.
- [4] Zhang, A.; Gourley, D.: *Creating Digital Collections: A Practical Guide*. Chandos Information Professional Series, Elsevier Science, 2014, ISBN 9781780631387. Dostupné z: <https://books.google.cz/books?id=qlmpAgAAQBAJ>
- [5] Witten, I.; Bainbridge, D.; Nichols, D.: *How to Build a Digital Library*. Morgan Kaufmann series in multimedia information and systems, Elsevier Science, 2009, ISBN 9780080890395. Dostupné z: <https://books.google.cz/books?id=HiJNbEy5f70C>
- [6] CESNET, z.s.p.o.: *Jak začít využívat služby datových úložišť*. 2016, [cit. 2017-04-22]. Dostupné z: https://du.cesnet.cz/cs/navody/jsme_tady_poprve
- [7] Duraspace: *Fedora Features*. [cit. 2017-04-27]. Dostupné z: <http://fedorarepository.org/features>
- [8] Borbinha, J. L.; Kapidakis, S.; Papatheodorou, C.; aj.: *Research and Advanced Technology for Digital Libraries: 13th European Conference. ECDL 2009, Corfu, Greece, September 27 - October 2, 2009, Proceedings*. Springer Science & Business Media, 2009, ISBN 978-3-642-04345-1.

- [9] Beckett, D.; Berners-Lee, T.; Prud'hommeaux, E.; aj.: *RDF 1.1 Turtle*. W3C, 2014, [cit. 2017-04-27]. Dostupné z: <https://www.w3.org/TR/turtle/>
- [10] DB-Engines: *DB-Engines Ranking of Search Engines*. 2017, [cit. 2017-04-23]. Dostupné z: <https://db-engines.com/en/ranking/search+engine>
- [11] Python Software Foundation: *The Python Tutorial*. 2017, [cit. 2017-04-28]. Dostupné z: <https://docs.python.org/3/tutorial/index.html>
- [12] Django Software Foundation: *FAQ: General*. 2017, [cit. 2017-04-28]. Dostupné z: <https://docs.djangoproject.com/en/1.11/faq/general/>
- [13] W3C: *WebAccessControl - W3C Wiki*. 2016, [cit. 2017-04-17]. Dostupné z: <https://www.w3.org/wiki/WebAccessControl>
- [14] Eichman, P.; Coburn, A.: *Determining the Effective Authorization Using WebAC*. Fedora Commons, 2016, [cit. 2017-04-17]. Dostupné z: <https://wiki.duraspace.org/display/FEDORA4x/Determining+the+Effective+Authorization+Using+WebAC>
- [15] Churý, L.: Aspektově orientované programování v Javě. V: *programujte.com [online]*, december 2006, [cit. 2017-04-17]. Dostupné z: <http://programujte.com/clanek/2006120416-aspektove-orientovane-programovani-v-jave/>

Zoznam použitých skratiek

VŠCHT	Vysoká škola chemicko-technologická
CIS	Centrum informačných služieb
ELN	Electronic lab notebook
GUI	Graphical user interface
XML	Extensible markup language
RDF	Resource Description Framework
ACL	Access control list
UML	Unified Modeling Language
SVG	Scalable Vector Graphics
GIF	Graphics Interchange Format
SMILES	simplified molecular-input line-entry system
HTTP	Hypertext Transfer Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Location
AOP	Aspect-oriented programming
TAR	Tape archiver
GZIP	GNU zip

Obsah priloženého CD

	readme.txt.....	stručný popis obsahu CD
	exe.....	adresár so spustiteľnou formou implementácie
	src	
	impl	zdrojové kódy implementácie
	thesis.....	zdrojová forma práce vo formáte L ^A T _E X
	text	text práce
	thesis.pdf	text práce vo formáte PDF
	thesis.ps	text práce vo formáte PS