

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.

23. apríla 2017

Pod'akovanie

Doplňte, ak chcete niekomu za niečo poďakovať. V opačnom prípade úplne odstráňte tento príkaz.

Prehlásenie

Prehlasujem, že som predloženú prácu vypracoval(a) samostatne a že som uviedol(uviedla) 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 23. apríla 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

V niekoľkých vetách zhrňte obsah a prínos tejto práce v slovenčine. Po prečítaní abstraktu by mal čitateľ mať dost informácií pre rozhodnutie, či Vašu prácu chce čítať.

Kľúčová slova Nahradte zoznamom kľúčových slov v slovenčine oddelených čiarkou.

Abstract

Sem doplňte ekvivalent abstraktu Vašej práce v angličtině.

Keywords Nahradte zoznamom kľúčových slov v angličtine oddelených čiarkou.

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	17
4.1 Návrh aplikácie	17
4.2 Zobrazovanie chemických dát	27
4.3 Automatický import dát	28
4.4 Kontrola stavov a prístupové práva	33
Záver	41
Literatúra	43
A Zoznam použitých skratiek	45
B Obsah priloženého CD	47

Zoznam obrázkov

3.1	Zobrazenie dát vo webovom rozhraní Fedory	15
4.1	Architektúra repozitára	17
4.2	Výpis dát v kolekcii InfraredSpectra	27
4.3	Zobrazenie detailu konkrétnej položky (Ethanolu)	28
4.4	Štruktúra databázovej tabuľky lab_journal	29
4.5	Štruktúra databázovej tabuľky project	30
4.6	Štruktúra databázovej tabuľky reaction	31
4.7	Štruktúra databázovej tabuľky reaction_chemical, 1. časť	32
4.8	Štruktúra databázovej tabuľky reaction_chemical, 2. časť	33
4.9	Diagram WebACL	35
4.10	Proces získania oprávnení vo Fedore	36

Zoznam tabuliek

2.1	MARC	6
3.1	ElasticSearch vs. Solr	16
4.1	Typy prístupu v ACL	35

Úvod

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. [1]

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. [2] Za zdroje pokladáme dáta v širokom zmysle slova (dátové súbory, textové informácie, obrazové informácie, hudbu,...). 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. 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). 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 vyvinul sa 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),... Pre systémy, ktoré kvalifikátory nepoužívajú, ale musí zostať význam zachovaný.

2.1.1.2 MARC

MARC využívajú najmä knihovníci. Bol navrhnutý pre popis bibliografických údajov v strojoivo 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,...). Takéto označenie je ľahko čitateľné pre stroje, knihovníci si pri každodennej práci s týmito číslami ich významy zapamätajú. Človek, ktorý ich vidí prvýkrát, významu nerozumie.

MARC je od DC komplikovanejší, dokáže však presnejšie popísať zdroj. Prvé číslo v číselných kódovoch 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 jednotlivých kódov sú len textové polia, pričom formát hodnoty nie je definovaný.

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 repozitára 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 prispôbiť 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/>]

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.

2. SÚČASNÉ RIEŠENIA

Používa MARC schému.

Modulárna architektúra, napísaný v jazyku Java. Pluginy 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 s inštaláciou na serveroch Oracle priamo pre koncového zákazníka alebo ako súčasť cloudových aplikácií Elements <https://elements.perkinelmer.com/> a plánovaného ChemDraw E-notebook <http://chemdrawenotebook.perkinelmer.cloud/>.

2.2.2 Open Enventory

[<https://www.chemie.uni-kl.de/goossen/open-enventory/>] Webová open source aplikácia napísaná v jazyku PHP. Využíva MySQL databázu.

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 teamu 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 pre zálohovanie, archiváciu, zdieľanie dát. [3] 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. [4] 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 priebeh meraní a pokusov. Aby bola využiteľnosť repozitára čo najlepšia a práca s ním čo najjednoduchšia, požadujú možnosť importovať dáta z aplikácie Open Eventory, 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,
- chemická rovnica bude zobrazená aj schematicky (obrázkom),
- pozorovanie priebehu pokusu,
- pri chemických prvkoch je potrebné evidovať:
 - štandardný názov prvku,
 - zápis štruktúry,
 - obrázok štruktúry,
 - chemický vzorec prvku,
 - molekulárna hmotnosť,
 - hmotnosť prvku,
 - koncentrácia.

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 objektom a poliam ale môže byť limitovaný. Vytváranie a úprava jednotlivých objektov 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 plyní 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 objektov. Taktiež tvorbu modelov pre jednotlivé typy objektov. 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 stávajúci 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. Oba repozitáre sú podobne výkonné (zvládajú milióny záznamov).

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 objekt, upraviť alebo zmazať existujúci objekt. Za kontajner sú vo Fedore považované všetky objekty okrem binárnych, teda aj tie, ktoré už žiadneho potomka nemajú. Binárneho typu sú objekty so súbormi nahranými 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 objektov vrámci kolekcie ale aj samotné dáta a metadáta objektu.

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

Existujúce rozhranie umožňuje vytvoriť nový objekt a následne k nemu pridať informácie. Tie je možné zadávať len v textovej forme vo formáte RDF (Resource Description Format) trojíc - subjekt, predikát a objekt. Je teda potrebné zjednodušiť vytváranie nových kolekcií a objektov v užívateľsky prívetivej forme.

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

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 pre 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 metadátach, ktorá je čo najrýchlejšia a zvláda veľké množstvo (milióny) záznamov.

The screenshot displays the Fedora web interface for a resource. At the top, the user 'test@cs' is logged in. The breadcrumb trail shows the path: Home / dcterms / e4 / bc / 86 / e2 / e4bc86e2-c77d-46c1-a613-15bee74782d3. The main content area is divided into two columns. The left column, titled 'Properties', lists various metadata fields such as 'dc:abstract', 'dc:alternative', 'dc:creator', 'dc:dateSubmitted', 'dc:title', 'fedora:created', 'fedora:createdBy', 'fedora:hasParent', 'fedora:hasVersions', 'fedora:lastModified', 'fedora:lastModifiedBy', 'fedora:writable', 'ns003:hasIndexingTransformation', and 'rdf:type'. The right column contains three sections: 'Create New Child Resource' with a 'Type' dropdown set to 'container' and an 'Identifier' field; 'Update Properties' with a text area for 'PREFIX' definitions; and 'Create Version Snapshot' with an 'auto-generated name' field. At the bottom right, there is a 'Delete Resource' button.

Obr. 3.1: Zobrazenie dát vo webovom rozhraní Fedory

Kedže 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 vyhľadávaci 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 [5] ide o najrozšírenejší vyhľadávaci 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

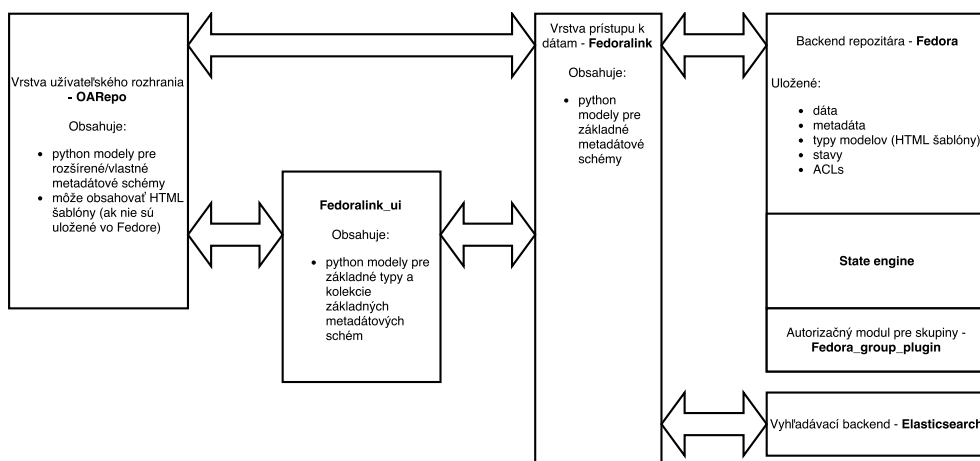
Vnorené dokumenty a možnosť vyhľadávať rodičov, ktorých deti spĺňajú špecifikovanú podmienku a opačne vyhľadávať potomkov, ktorých rodičia spĺňajú špecifikovanú podmienku viedli k tomu, že je v repozitári použitý ElasticSearch ako vyhľadávací software. Medzivrstva ale umožňuje pracovať aj s aplikáciou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

Implementácia

4.1 Návrh aplikácie

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

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



Obr. 4.1: Architektúra repozitára

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 objektami získanými z Fedory ako s Django objektmi. Metadáta dokáže 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 objektov 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 objektu z Fedory aplikácia vytvorí objekt v Pythne, 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 vo formáte RDF (Resource Description Format), 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.

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. Rozšírenie Fedora_group_plugin umožňuje autorizáciu na základe On-Behalf-Of-Django-Groups hlavičky, o ktoré sa stará DjangoGroupPrincipalProvider. Autorom tohto pluginu je Mgr. Miroslav Šimek.

Plugin je súčasťou git repozitára federalinku. Impersonifikácia na iného užívateľa alebo skupinu je umožnená len pod FedoraAdmin užívateľom. Prístupové údaje k FedoraAdmin užívateľovi sú závislé na nastavení Java Servlet kontajnera (Tomcat, GlassFish,...), pod ktorým beží inštancia Fedory. Hodnota posiadaná 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“. Skupiny sú získané z užívateľových skupín v Django.

Vďaka tomuto rozšíreniu bude možné jednoduché rozšírenie repozitára o autentizáciu cez iného poskytovateľa identity (napr. Shibboleth).

4.1.4 Elasticsearch

Aplikácia, ktorá umožňuje rýchle vyhľadávanie v metadátach.

4.1.5 Federalink

Aplikácia pôvodne napísaná pre potreby repozitára záverečných prác VŠCHT Praha, v programovacom jazyku Python s využitím frameworku Django, stará sa o komunikáciu s Fedorou a Elasticsearch. Autorom federalinku je Mgr. Miroslav Šimek. Federalink 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 objektmi získanými z Fedory pracovať ako s objektmi v Django, boli vytvorené triedy, ktoré s týmito objektmi 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.

Objekt 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 objekt 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. Objekt 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 objektom z Fedory.

Získanie správnej triedy pre objekt 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 objektu alebo kolekcie vo Fedore.

Vzory využívajúce regulárne výrazy pre URL adresy:

- `'^$'` - index
- `r'^(?P<collection_id>[a-fA-F0-9_-]*)?search(?P<parameters>.*)$'` - vyhľadávanie v rámci kolekcie
- `'^(?P<id>.*)/addSubcollection$'` - pridanie novej subkolekcie
- `'^(?P<id>.*)/add$'` - vytvorenie nového objektu ako potomka objektu s daným ID
- `'^(?P<id>.*)/edit$'` - upravenie objektu s daným ID
- `'^(?P<id>.*)$'` - zobrazenie objektu s daným ID

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

4.1.6.2 Získanie šablóny pre zobrazenie, editáciu objektu alebo zoznam objektov v kolekcii

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

V samotnej aplikácii (vo fedoralinku alebo v koncovej aplikácii) musí byť model objektu - trieda v Pythone. Ostatné potrebné veci sú uložené priamo vo Fedore. V nej sú uložené jednotlivé typy objektov, pri kolekciách je uložený typ subkolekcií a potomkov. Tieto objekty 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 objekty. Získanie správneho typu objektu 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 objekt, ku ktorému nájde vo Fedore

uložený správny typ. Z neho následne získa šablónu, ktorú zobrazí. Ak sa správna šablóna pre objekt 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 objektu 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 objektu.

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
```

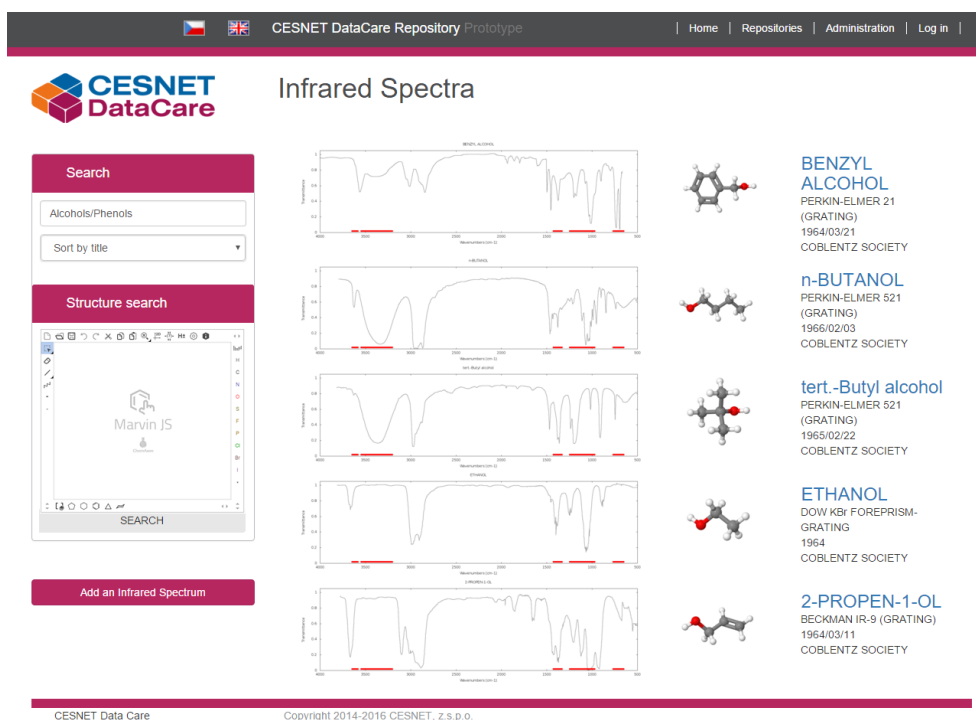
Metóda *get_resource_type* umožňuje získať správny RDF typ objektu.

```
@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 z objektu šablóny, 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*.

4.2. Zobrazovanie chemických dát



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

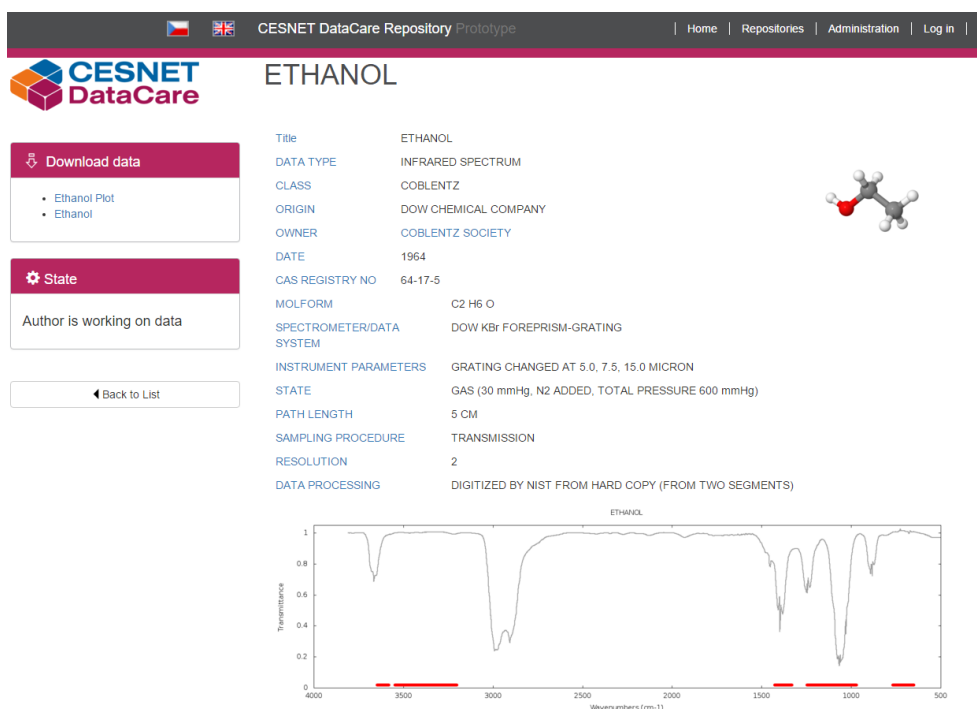
Repozitár bude nasadený ako jedna zo služieb diskového úložiska CESNET, z.s.p.o. <https://du.cesnet.cz/>, preto navrhnuté grafické rozhranie vychádza z už existujúcich služieb.

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

4.2 Zobrazovanie chemických dát

Jednotlivé časti repozitára sú navrhnuté tak, aby bolo pridanie nových modelov a šablón do systému čo najjednoduchšie.

4. IMPLEMENTÁCIA



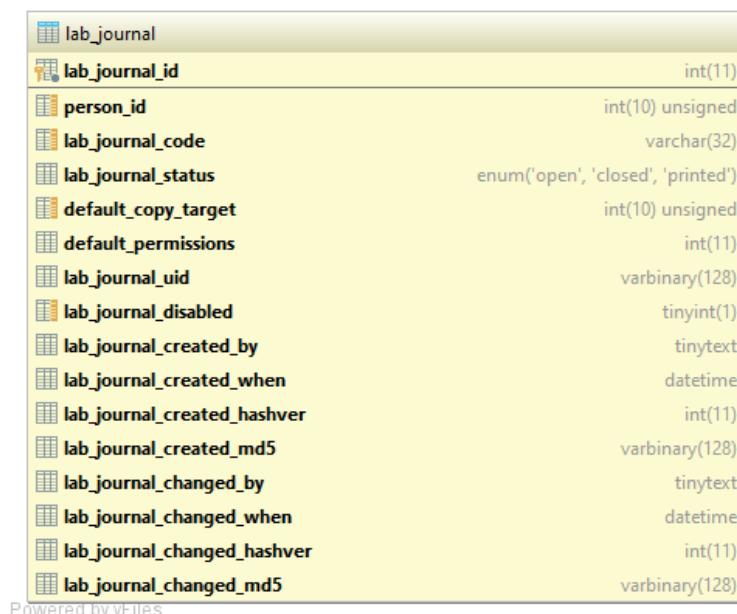
Obr. 4.3: Zobrazenie detailu konkrétnej položky (Ethanolu)

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.

K importu dát má dôjsť po kliknutí na tlačidlo umiestnené priamo v tejto webovej aplikácii. Aby bolo možné aplikáciu upraviť, je potrebné vedieť ako funguje, aké sú vzťahy medzi tabuľkami v databáze a nájsť vhodné umiestnenie pre tlačidlo. Open Enventory je open source aplikácia, ktorá má k dispozícii všetky zdrojové kódy. Neexistuje k nej ale žiadna dokumentácia.

Pri prechádzaní kódu aplikácie som navyše zistil, že veľká časť komentárov a aj časť samotného kódu sú napísané v nemčine. Databázové 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.



lab_journal	
lab_journal_id	int(11)
person_id	int(10) unsigned
lab_journal_code	varchar(32)
lab_journal_status	enum('open', 'closed', 'printed')
default_copy_target	int(10) unsigned
default_permissions	int(11)
lab_journal_uid	varbinary(128)
lab_journal_disabled	tinyint(1)
lab_journal_created_by	tinytext
lab_journal_created_when	datetime
lab_journal_created_hashver	int(11)
lab_journal_created_md5	varbinary(128)
lab_journal_changed_by	tinytext
lab_journal_changed_when	datetime
lab_journal_changed_hashver	int(11)
lab_journal_changed_md5	varbinary(128)

Powered by yFiles

Obr. 4.4: Štruktúra databázovej tabuľky lab_journal

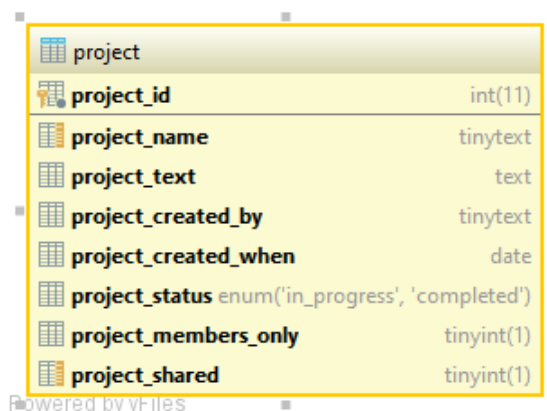
4.3.1 Schéma databázy Open Enventory

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, ktorej štruktúra je na obrázku 4.4, 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. Štruktúru databázovej tabuľky s uloženými projektmi je možné vidieť na obrázku 4.5.

Samotné dáta meraní, popis a priebeh reakcií sú uložené v databázovej tabuľke reaction, jej schému si je možné pozrieť na obrázku 4.6. 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 rxnfile_blob, rxn_gif_file - chemická rovnica reakcie vo formáte GIF, rxn_svg_file - obrázok vo formáte SVG. RXN



project	
project_id	int(11)
project_name	tinytext
project_text	text
project_created_by	tinytext
project_created_when	date
project_status	enum('in_progress', 'completed')
project_members_only	tinyint(1)
project_shared	tinyint(1)

Obr. 4.5: Štruktúra databázovej tabuľky project

je formát pre popis reakcie, ktorý pozostáva z bloku reaktantov, produktov a prípadne (nie veľmi zvyčajne) aj bloku agentov.

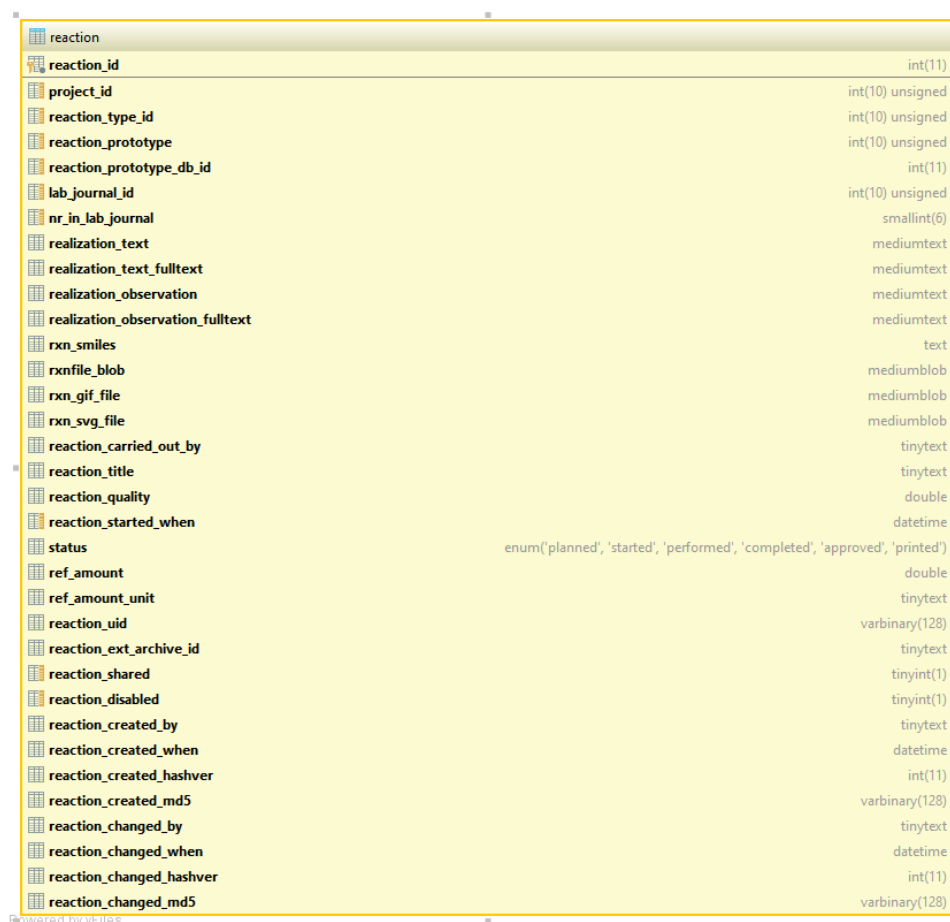
V tabuľke reaction_chemical (schéma je na obrázkoch 4.7 a 4.8) sú uložené informácie o reaktantoch a produktoch. Môže tu byť uvedený štandardný názov prvku v stĺpci standard_name, tzv. SMILES teda zjednodušený jednoriadkový zápis štruktúry (uložené v stĺpci smiles). Binárne hodnoty popisujúce prvok v stĺpcoch molfile_blob (MDL Molfile je súborový formát, ktorý umožňuje ukladať informácie o atómoch, 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



Field Name	Data Type
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. 4.6: Štruktúra databázovej tabuľky reaction

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 laboratórnych denníkov uľahčili. Kód aplikácie je navyše neprehľadný s komentármi a aj časť 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 predá 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

4. IMPLEMENTÁCIA

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. 4.7: Štruktúra databázovej tabuľky reaction_chemical, 1. časť



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. 4.8: Štruktúra databázovej tabuľky reaction_chemical, 2. časť

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 vygenerované triedy upraviť.

4.4 Kontrola stavov a prístupové práva

Fedora umožňuje nastaviť práva pre jednotlivé objekty pomocou ACL. Takýmto spôsobom vieme nastaviť, kto môže daný objekt 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 objekt individuálne. Tak by mohlo dochádzať k častým chybám s nesprávne nastavenými oprávneniami. Preto bol navrhnutý spôsob kontroli prístupových práv cez stavy. Tieto určujú v akom stave alebo stavoch sa konkrétny objekt nachádza.

Napríklad môže ísť o schvalovací proces. Vložené dáta nesmú byť zverejnené hneď. Je potrebné, aby prešli procesom schvalovania cez viacero ľudí. Prístup k tomuto objektu má na začiatku osoba, ktorá objekt 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 objekt nachádza v stave „zverejnený“, prezerať si ho môžu 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 objektom 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 objekt vráti HTTP dokument obsahujúci hlavičku s odkazom na ACL objekt, ktorý popisuje prístupové práva pre daný dokument (prípadne pre iné dokumenty). [6]

Nastavenie prístupových práv, prepojenie dokumentov a ACL objektov je možné vidieť na diagrame znázornenom na obrázku 4.9. 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).

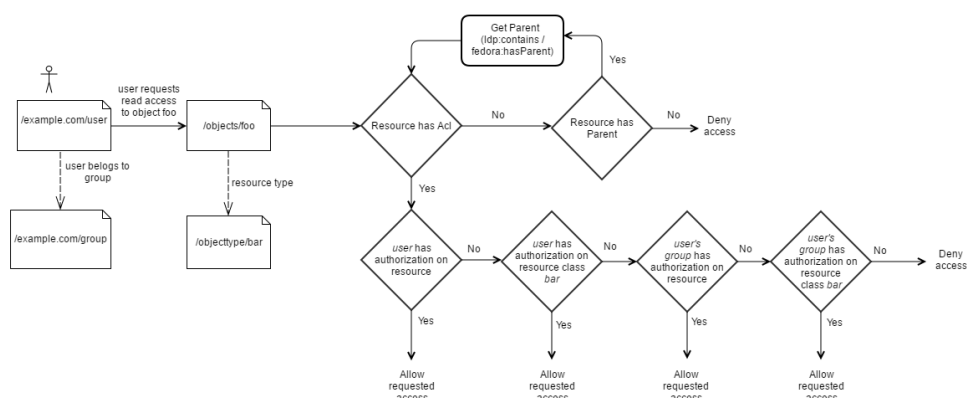
ACL je zoznam oprávnení priradených k nejakému objektu. Pomocou ACL môžeme určiť, ktoré osoby (parameter *agent*) alebo skupiny (parameter *agentClass*) majú mať prístup k danému objektu. Tento objekt, prípadne objekty sú určené parametrom *accessTo* alebo *accessToClass*. Ako trieda pre všetkých užívateľov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

Príklad nastavenia prístupu (vo formáte turtle):

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

<> a acl:Authorization;
    acl:accessTo <$FCREPOREL/administration/states>;
```


4. IMPLEMENTÁCIA



Obr. 4.10: Proces získania oprávnení vo Fedore, zdroj: [7]

existuje ACL (objekt má nastavený parameter `acl:AccessControl`). Ak nie, prejde všetkých rodičov až po koreň. Pokiaľ ACL nenájde, prístup k danému objektu zakáže. V opačnom prípade zistí, či má užívateľ prístup na daný objekt, na daný typ objektov alebo či má oprávnenie užívateľova skupina. Ak nenájde žiadne oprávnenie, prístup zakáže, inak vráti dopytovaný objekt.

4.4.2 Stavý

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

4.4.2.1 Stav uloženého objektu v repozitári

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

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

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

Príklad objektu 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 objekt 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 objekt nachádza, je potrebné taktiež definovať povolené zmeny stavov.

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

4.4.2.3 Prechod medzi stavmi

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

`State:targetState` určuje cieľový stav prechodu. ACL priradené na objekt 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 objektu. V prípade, ak áno, je potrebné skontrolovať oprávnenia a zmenu povoliť alebo zamietnuť. K ukladaniu upravených objektov 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`.

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.

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

AspectJ <http://www.eclipse.org/aspectj/> je rozšírenie využívajúce aspektovo orientované programovanie (AOP) pre jazyk Java. Toto rozšírenie umožňuje využívať špeciálne konštrukty nazývané aspekty (aspect).

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 týmto 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. [8] 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.

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 objektu. V prípade ak to robí užívateľ, ktorý nemá oprávnenie na zmenu, aplikácia vráti chybový HTTP kód. Ak StateControl nie je k objektu priradený, ale užívateľ sa pokúša priradiť k objektu stavy, aplikácia taktiež vráti chybový HTTP kód.

Ak je StateControl upraveného objektu 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 objekty 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] 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>
- [2] 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.
- [3] CESNET, z.s.p.o.: *Oddělení datových úložišť CESNET*. 2016, [cit. 2017-04-22]. Dostupné z: <https://du.cesnet.cz/cs/start>
- [4] 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
- [5] DB-Engines: *DB-Engines Ranking of Search Engines*. 2017, [cit. 2017-04-23]. Dostupné z: <https://db-engines.com/en/ranking/search+engine>
- [6] W3C: *WebAccessControl - W3C Wiki*. 2016, [cit. 2017-04-17]. Dostupné z: <https://www.w3.org/wiki/WebAccessControl>
- [7] 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>
- [8] 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

ELN	Electronic lab notebook
GUI	Graphical user interface
XML	Extensible markup language
RDF	Resource Description Format
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
AOP	Aspect-oriented programming

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