

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.

9. 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 9. 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	8
3 Analýza a návrh riešenia	11
3.1 Analýza požiadaviek	11
3.2 Výber technológií	12
4 Implementácia	13
4.1 Návrh aplikácie	13
4.2 Automatický import dát	21
Záver	25
Literatúra	27
A Zoznam použitých skratiek	29
B Obsah priloženého CD	31

Zoznam obrázkov

4.1	Výpis dát v kolekcii InfraredSpectra	22
4.2	Zobrazenie detailu konkrétnej položky (Ethanolu)	23
4.3	Štruktúra databázovej tabuľky lab_journal	24

Zoznam tabuliek

2.1	MARC	6
-----	----------------	---

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 sa 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 ktorej využitím 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 uschovávanie 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.

Súčasn  rie enia

2.1 Repozit re

Existuj  r zne repozit re, ktor  sa od seba l šia pou itou technol giou, mo nos ou roz  irania, pou ivaj  r zne metad tov  sch my. Niektor  s  voľne dostupn  ako open source, in  ako propriet rny software alebo host van  aplik cie. V tejto  asti uv dzam prehľad dostupn ch aplik ci . Zameriavam sa najm  na vlastnosti, ktor  boli pre dal    v voj repozit ra kľ čov , a to: open source (aby bolo mo n  software ďalej upravovať), pou itie metad tovej sch my, modul rnosť softwaru (jednoduch  mo nosť roz  irania o dal    n stroje) a verziovanie (najm  kv li zdieľaniu a z lohovaniu d t).

2.1.1 Metad ta

Na popis ulo en ch dokumentov sl  ia metad ta. Metad ta s   trukturovan  d ta nes ce inform ciu o prim rnych d tach.[2] Kv li vz jomnej prepojenosti repozit rov, vyhľadávaniu d t a spr vnej interpret cii inform ci  je snaha o vyvinutie celosvetovo pou ivan ho  tandardu pre popis d t.

O to sa sna ia r zne metad tovej sch my, pomocou ktor ch je mo n  zdroje pop sať. Medzi najzn mej  ie sch my pat  Dublin Core [<http://dublincore.org/>] a MARC [<http://www.loc.gov/marc/>].

2.1.1.1 Dublin Core

Dublin Core (skr tene DC) vznikol s cieľom jednoducho a v  obecne pop sať webové zdroje. T to sch ma obsahuje 15 prvkov. To s : n zov (title), autor (creator), predmet (subject), popis (description), vydavateľ (publisher), prispievateľ (contributor), d tum (date), typ (type), form t (format), identifik tor (identifier), zdroj (source), jazyk (language), vz ťah (relation), pokrytie (coverage) a pr va (rights). Tieto prvky nie s  povinn  a m  u sa opakovať. Jednotliv  vlastnosti s  teda pomenovan . Ako sa World Wide Web menil, v sna e o vytvorenie s mantick ho webu, vyvinul sa aj  tandard Dublin Core.

2. SÚČASNÉ RIEŠENIA

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

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 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,...). 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 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.

Použitie navrhovaného repozitára by malo byť jednoduché aj pre užívateľov, ktorí s metadátami nemajú veľké skúsenosti a nepotrebuje 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 pre nás dôležitých vlastností 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.

Pretože každý software pokrýva inú kombináciu týchto vlastností, repozitáre neklasifikujem, len uvádzam 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.

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/>]

2. SÚČASNÉ RIEŠENIA

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). Pretože

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

3.1.1 Funkčné požiadavky

Repozitár musí umožniť:

- Uložiť nové dáta.
- Upraviť existujúce dáta.
- Zobraziť existujúce dáta.
- Uložiť históriu zmien dát.
- Vyhľadávať v metadátach.
- 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.

3.2 Výber technológií

3.2.1 Fedora

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. Výber vhodného repozitára bol možný z open source repozitárov, ktoré umožňujú úpravu kódu.

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 do tohto softwaru jednoduchšie pridávanie rozšírení, 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.

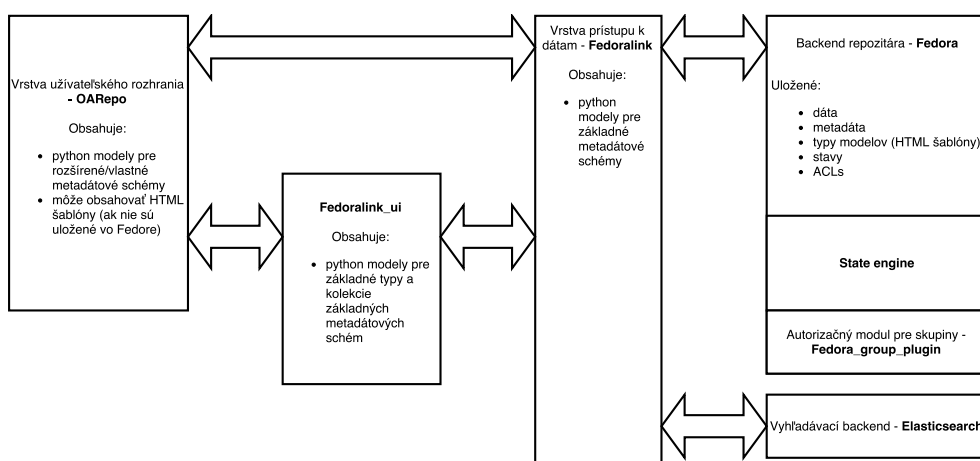
Z existujúcich možností bola zvolená Fedora ako najvhodnejší software pre možnosť ďalších potrebných úprav pre použitie v rámci služieb CESNET z.s.p.o. a splnenie požiadaviek Ústavu organickej chémie VŠCHT Praha.

3.2.2 Elasticsearch

Na vyhľadávanie v repozitári bude použitá samostatná aplikácia Elasticsearch <https://www.elastic.co/products/elasticsearch>. Aplikácia umožňuje veľmi rýchle vyhľadávanie v indexovaných dátach. Dopyty je možné posielat s využitím RESTful api a JSONu.

Implementácia

4.1 Návrh aplikácie



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. 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. 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 vyhľadavanie v metadátach.

4.1.5 Fedoralink

Aplikácia 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. 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 z objektami získanými z Fedory pracovať ako s objektami v Django, boli vytvorené triedy, ktoré s týmito objektami pracujú. Trieda `FedoraObject` je základnou triedou pre tieto objekty. S dátami získanými vo formáte RDF z Fedory umožňuje pracovať aj keď vopred nepoznáme štruktúru týchto dát.

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

<pre># [RDF:Name] obj[DC.title]</pre>

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 spadať do viacerých 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 starosť 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 federalinku 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 Resource Type, 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 dekóduje a uloží ako reťazec, dáta do Fedory ukladáme v kódovaní utf-8. O cachovanie tejto šablóny sa stará metóda *_get_template_string_internal*.

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

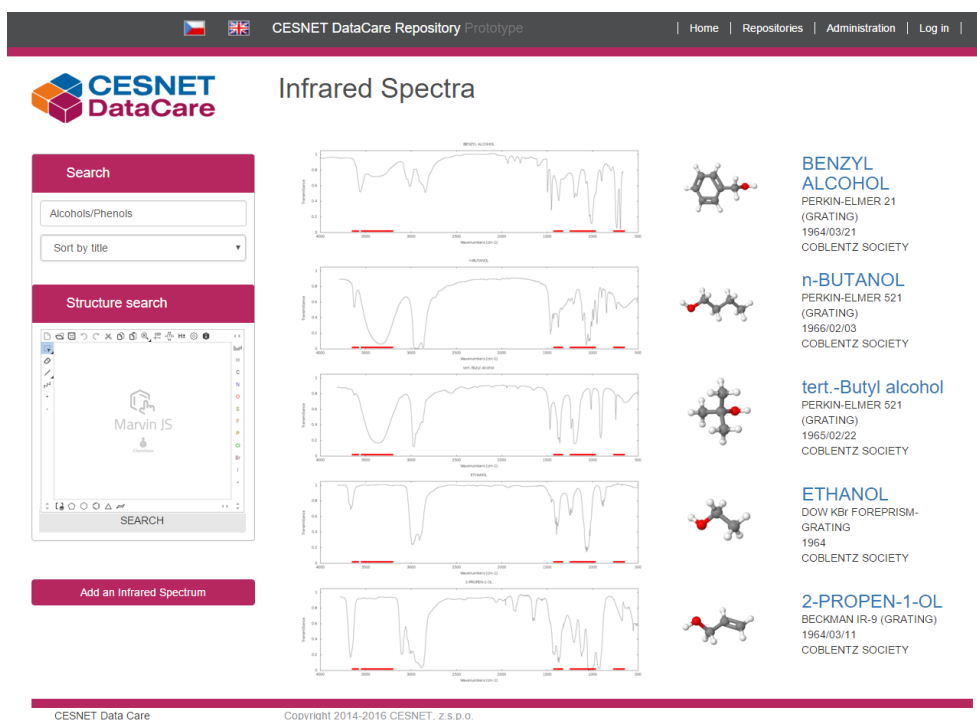
Repozitár bude nasadený ako jedna zo služieb diskového úložiska CESNET, z.s.p.o. <https://du.cesnet.cz/>, preto navrhnuté grafické rozhranie vychádza z už existujúcich služieb.

Návrh zobrazenia pre dáta organickej chémie:

4.2 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,

4. IMPLEMENTÁCIA



Obr. 4.1: Výpis dát v kolekci Infrared Spectra

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.

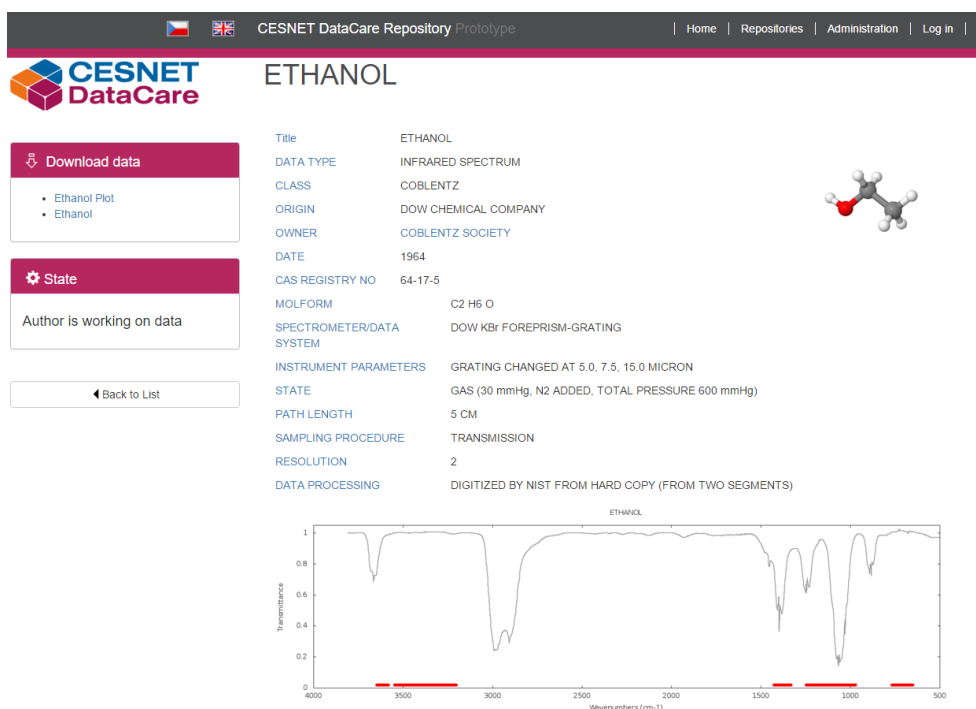
K importu dát má dôjsť po kliknutí na tlačítko umiestnené priamo v tejto webovej aplikácii. Aby bolo možné aplikáciu upraviť, je potrebné vedieť ako funguje, aké sú vzťahy medzi tabuľkami v databázi a nájsť vhodné umiestnenie pre tlačítko. Open Enventory je open source aplikácia, ktorá má k dispozícii 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 kľúčami sú len indexy.

4.2.1 Schéma databáze

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

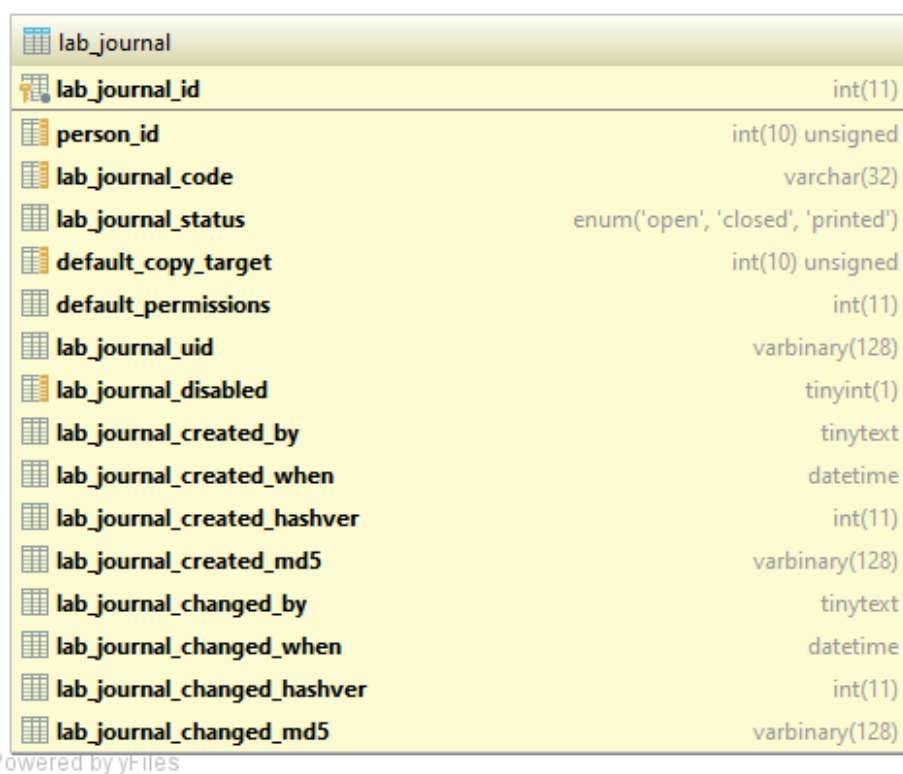
4.2. Automatický import dát



Obr. 4.2: Zobrazenie detailu konkrétnej položky (Ethanolu)

V databázovej tabuľke `lab_journal`, ktorej štruktúra je na obrázku 4.3 sú uložené základné údaje o laboratórnom denníku. Pre naše účely budú ďalej dôležité stĺpce `person_id` a primárny kľúč `lab_journal_id`.

4. IMPLEMENTÁCIA



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. 4.3: Štruktúra databázovej tabuľky lab_journal

Záver

Literatúra

- [1] Strnad, M.: Svěřte svá data vhodnému médiu – díl 1. V: *LinuxEXPRES [online]*, október 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.

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

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