Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Správce sbírek založený na technologii sémantického webu

Plzeň, 2012 Michal Madleňák

Prohlášení

Prohlašuji, že tato diplomová práce je mým původním autorským dílem, které jsem vypracoval samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používal nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

V Plzni, dne 17.5.2012

….....

Michal Madleňák

Poděkování

Chtěl poděkovat panu Ing. Petru Včelákovi za velkou trpělivost s vedení této práce a také za rady a zkušenosti, které jsem díky němu získal. Dále bych chtěl poděkovat Ing. Petru Záhorkovi za vytvoření grafického vzhledu výsledného systému.

Abstract

Collection Manager Based on Semantic Web Technology

The purpose of this thesis is to create a user-friendly tool for managing collections. This solves the management problems of a large amount of data with different formats and provides users the ability to search using metadata.

The text describes and evaluates various options for methods that can be used to create applications including an analysis of programming languages, RDF repositories and libraries for working with metadata. The resulting tool is created in the Java programming language and uses to store RDF metadata repository Jena.

A collections management tool allows you to save files to a directory structure, reading and storing metadata from files into the RDF repository. Furthermore, it enables data export and import and retrieval of data ontologies according to their parameters. Particular graphical user interface was implemtented that provides access to data and enables browsing of an extensive collection and storage of heterogeneous data.

This thesis deals with semantic web technologies and some used tools such as Jena RDF store, query language SPARQL, etc. A tool for managing collections is a graphically friendly and intuitive application that meets the principles of the Semantic Web and enables searching of large data quantities using triples in the knowledge repository.

Obsah

1 Úvod

Každý uživatel stolních počítačů, notebooků či tabletů se setkává s nepřehledností svých hudebních souborů, filmů, galerií, dokumentů a jiných důležitých záznamů bez jakékoliv možnosti jejich vyhledání dle metadat (například podle žánru, data vytvoření, obsahu, osoby zobrazené na fotce, atd.). Proto vznikl požadavek na vytvoření projektu nazvaného *Správce sbírek založený na technologii sémantického webu.*

Hlavním cílem této práce je seznámit se s sémantickým webem a vytvořit intuitivní systém, pracující na této technologii s RDF úložištěm. Systém tedy bude podporovat čtení metadat a ukládání trojic do RDF úložiště Jena, prohlížení a editaci uložených souborů a export i import dat.

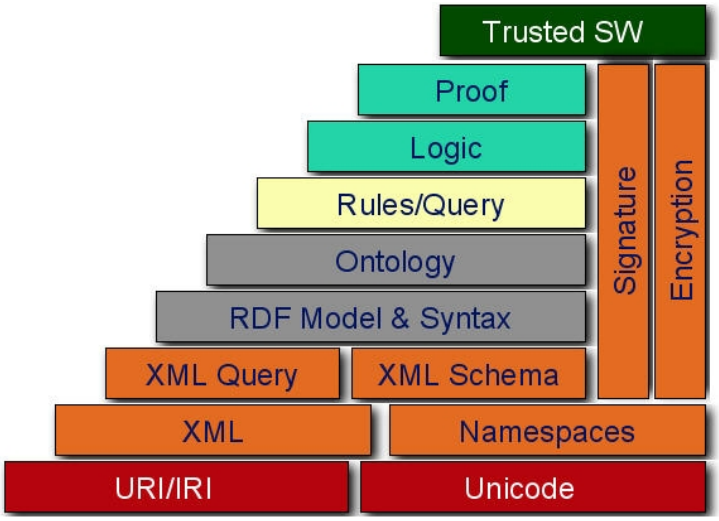
Výsledný systém umožní uživateli prohlížení uložených souborů a jejich následné vyhledávání dle metadat.

2 Sémantický web

Výchozí myšlenka sémantického webu[2], [78], [79], [80], nebo-li významového webu se poprvé zrodila v roce 2001 u vynálezce WWW[1] (World Wide Web) Timothy John Berners-Lee, který upozornil na nedostatky současného webu. Hlavní nedostatek webu spočívá v neustálém nárůstu informací a složitějším vyhledání relevantních dat.

Sémantický web je tedy rozšíření WWW o dodatečné informace, které umožňují lidem sdílet obsah za hranicemi aplikací a webových stránek. Sémantický web je založen na technologii Resource Description Framework (RDF – viz kapitola 2.2 na straně 4), která definuje rámec pro práci s metadaty (viz kapitola 2.5 na straně 9). Reprezentace sémantického webu je doposud vyvíjeným prostředkem, ve kterém jsou prezentovaná data na Internetu vyjádřena nejen v lidském jazyce, ale též srozumitelnou formou pro strojové agenty.

Jedním ze základních předpokladů k vytvoření sémantického webu je konceptualizace dostupných dat na WWW, jejíž hlavním klíčovým nástrojem jsou ontologie (viz kapitola 2.6 na straně 10). Dalším důležitým předpokladem je standardizovaný popis webových zdrojů (např. dokumenty, obrázky, hudební soubory, atd.), které jsou vybavené stejnými charakteristikami (např. autor, popis, klíčová slova, atd.), nebo-li metadaty (viz kapitola 2.5 na straně 9).



Ilustrace 1: Struktura sémantického webu podle konsorcia W3C

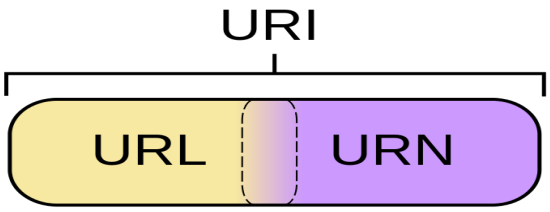
2.1 URI

Aby se zajistila jednoznačná identifikace objektů v RDF úložišti (viz kapitola 2.9 na straně 14), používá se k tomuto účelu vhodná terminologie. V tomto případě se jedná o terminologii URI [10], [45].

URI (Uniform Resource Identifier – v překladu jednotný popis zdroje) je řetězec znaků, sloužící k přesné specifikaci zdroje informací (má přesně definovanou strukturu). Tento řetězec znaků lze využít například pro popsání a identifikaci webové stránky, zjednodušení komunikace v počítačových sítích, atd.

URI obsahuje dvě podmnožiny (viz Ilustrace 2 na straně 3):

- **URL** (Uniform Resource Locator) – slouží k popisu adresy zdroje, např. http://www.e-nakupujeme.cz/, atd.
- **URN** (Uniform Resource Name) – slouží k definování zdroje (nevypovídá nám nic o místu uložení). Příkladem může být ISBN kód pro identifikaci knih, čárové kódy, atd.



Ilustrace 2: Eulerův diagram vztahu mezi URI, URN a URL[65]

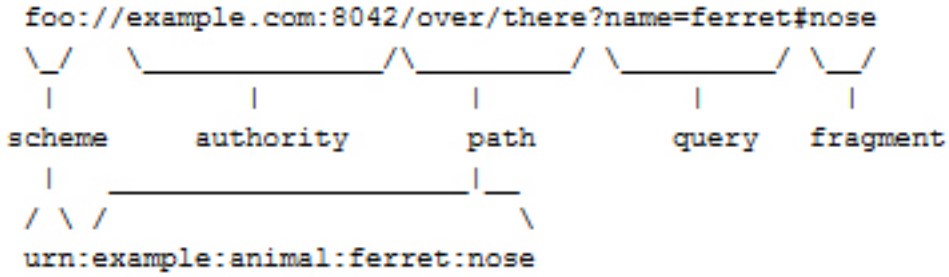
URI může popisovat zdroj jak z hlediska jeho adresy, tak i z hlediska jeho názvu (v realitě se ale URL občas vydává za obecnější URI – málokdo vidí rozdíl).

URI má následující formát:

<schéma>:<hierarchická cesta>?<dotaz>#<fragment>

- <schéma> - je složeno z libovolného počtu znaků a je zakončeno dvojtečkou. Tato část určuje o jaký druh URI se jedná (např. HTTP, HTTPS, FTP, atd.).
- <hierarchická cesta> - obsahuje identifikátor zdroje (například internetovou adresu, *www.e-nakupujeme.cz/*)
- <dotaz> - nepovinná položka, která blíže specifikuje určení požadovaného zdroje (nemá standardizovanou syntax, ale v praxi se používá tento zápis: *vysoka_skola=ZCU&katedra=KIV&rocnik=2*)
- <fragment> - nepovinná položka, která blíže specifikuje požadovaný zdroj

Detailnější znázornění URI naleznete na obrázku níže (viz Ilustrace 3, strana 3).



Ilustrace 3: Komponenty syntaxe URI

2.2 RDF

RDF (Resource Description Framework) [10], [11], [13], [14] je jazyk určený k popisu a modelování informací v různých syntaxích, který vyvinula organizace W3C. Samotný účel RDF je umožnit počítačům (strojovým agentům) zpracovávat a pochopit informace stejným způsobem, jako to dokáže člověk.

RDF funguje na principu přiřazení výrazu ve tvaru *podmět (subject) – vlastnost (predicate) - předmět (object)* k jednotlivým zdrojům. Tento tvar se v RDF nazývá trojice (viz Ilustrace 4 na straně 4).

Podmět (nebo-li subjekt) – se rozumí zdroj, který popisujeme (bývá zapisován ve formátu URI)

Vlastnost (nebo-li predikát) – vlastnost daného zdroje

Předmět (nebo-li objekt) – je konkrétní hodnota této vlastnosti (může to být buď opět nějaký zdroj, nebo obyčejný řetězec)

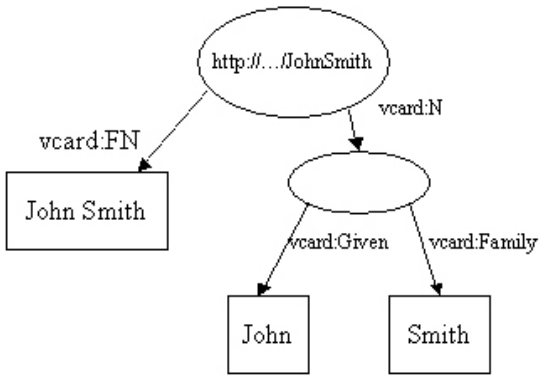
Ilustrace 4: RDF trojice[9]

2.2.1 Příklad RDF trojice

Máme jednoduché tvrzení: *„John Smith má rodné jméno Smith“*

- Subjekt je zde *John Smith*
- Predikát je *má rodné jméno*
- Objekt je *Smith*

Toto jednoduché tvrzení lze prezentovat například pomocí grafu (viz Ilustrace 5 na straně 4)



Ilustrace 5: Prezentace RDF pomocí grafu[15]

2.2.2 Kontejnery

Kontejner[38], [48], [76] (Container) je prostředek, sloužící k shlukování zdrojů nebo hodnot, neboli je to prostředek pro popis skupiny věcí. U kontejnerů je ale důležité si uvědomit, že vlastnost zdroje nemusí být vlastností všech jeho členů.

Pro pochopení předchozí věty bude citován přeložený příklad z použité literatury [76]: *„Kurník může mít tu vlastnost, že je ze dřeva. Ale neznamená to, že všechny slepice, které obsahuje, jsou vyrobeny ze dřeva. Vlastnost kontejneru není nutně vlastností všech svých členů.“*

Rozlišujeme 3 typy kontejnerů:

- **rdf:Bag –** slouží k seskupení neuspořádaných zdrojů nebo literálů
- **rdf:Seg -** slouží k seskupení uspořádaných zdrojů nebo literálů
- **rdf:Alt** – představuje alternativní popis (např. originální název knihy chceme popsat alternativními názvy v různých jazycích)

Příklad použití kontejneru rdf:Seg naleznete níže (viz Příklad 1 na straně 5).

```
<?xml version="1.0" encoding="utf-8"?>
      <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
               xmlns:ex="http://www.example.org/terms/">

      <rdf:Description
            rdf:about="http://www.exam.org/people/jones08"
            ex:fullName="Robert Jones">
            <ex:student rdf:nodeID="xyz"/>
      </rdf:Description>

      <rdf:Description rdf:nodeID="xyz"
            ex:fullName="David Smith">
            <ex:project>
                  <rdf:Seq rdf:ID="projects">
                        <rdf:_1
                        rdf:resource="http://www.exam.org/research/p1/"/>
                        <rdf:_2
                        rdf:resource="http://www.exam.org/research/p2/"/>
                        <rdf:_3
                        rdf:resource="http://www.examp.org/research/p3/"/>
                  </rdf:Seq>
            </ex:project>
      </rdf:Description>
</rdf:RDF>
```

Příklad 1: použití kontejneru rdf:Seg[48]

2.2.3 Kolekce

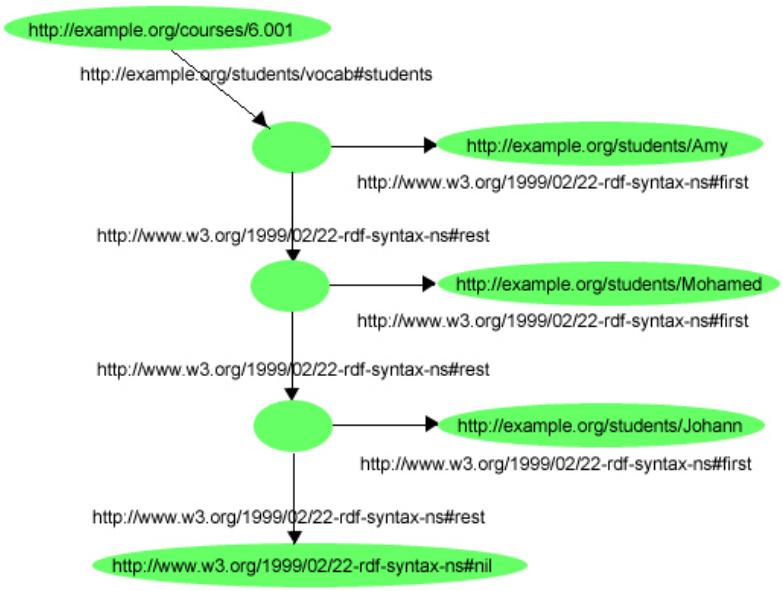
Kolekce[38], [47], [48], [76], [77] (Collections) slouží pro popis skupiny objektů, obsahující pouze určené členy, jejichž počet je konečný. K zápisu kolekcí se používá atribut *rdf:parseType=”Collection”*.

Příklad na použití kolekce naleznete níže (viz Příklad 2 na straně 6).

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:s="http://example.org/students/vocab#">

   <rdf:Description rdf:about="http://example.org/courses/6.001">
      <s:students rdf:parseType="Collection">
            <rdf:Description rdf:about="http://example.org/students/Amy"/>
            <rdf:Description
                  rdf:about="http://example.org/students/Mohamed"/>
            <rdf:Description
                  rdf:about="http://example.org/students/Johann"/>
      </s:students>
   </rdf:Description>
</rdf:RDF>
```

Příklad 2: použití kolekce[77]

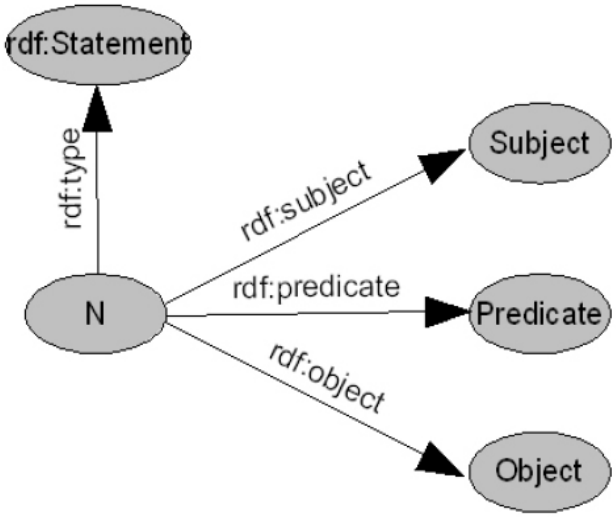


Ilustrace 6: RDF kolekce (stromová struktura) [77]

2.2.4 Reifikace

Reifikace[14], [20], [75] (rozklad) nám dovoluje formulovat tzv. *tvrzení o tvrzení*. Jinak řečeno, nám umožňuje zapsat složitější tvrzení pomocí pomocných tvrzení, která slouží k jeho identifikaci.

Na obrázku níže (viz Ilustrace 7 na straně 7) vidíte obecný rozklad vrcholu N na pomocná tvrzení.



Ilustrace 7: Graf tvrzení, zapsaného pomocí reifikace [14]

2.3 RDFS

RDF Schema[9], [11], [12], [14], [74] (nebo-li RDFS), vytvořené konsorciem W3C, je jednoduchý ontologický jazyk, který rozšiřuje RDF (viz kapitola 2.2 RDF na straně 4) slovník o mechanismy pro popis skupin souvisejících zdrojů a vztahů mezi objekty (hierarchie). RDFS tedy umožňuje bohatší integraci a interoperabilitu dat mezi skupinami.

Díky své jednoduchosti a univerzálnosti, RDFS splňuje základní požadavky pro vkládání sémantiky do obsahu webových stránek. V RDFS se definují třídy a vlastnosti, které poté slouží k popisu tříd, vlastností, vztahů a dalších zdrojů.

Přehled několika tříd:

1. **rdfs:Class –** třída tříd
2. **rdf:Property** – třída RDF vlastností
3. **rdfs:Literal –** třída písemných hodnot jako jsou textové řetězce, čísla, atd.
4. **rdfs:Resource –** zdroj třídy (může být cokoliv)

Přehled několika vlastností:

1. **rdf:type** – subjekt je instancí třídy
2. **rdfs:domain** – doména vlastnosti subjektu
3. **rdf:value –** vlastnost používaná pro strukturované hodnoty
4. **rdf:subject** – subjekt
5. **rdf:predicate** - vlastnost

Pro snadnější pochopení RDFS využijeme ukázkový příklad níže (viz Příklad 3 na straně 8).

```
<rdf:RDF
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:biblioteka="http://example.org/schemas/biblioteka#"
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

      <rdfs:Class rdf:ID="Kniha"/>

      <rdf:Property rdf:ID="pocetStran">
            <rdfs:domain rdf:resource="#Kniha"/>
            <rdfs:range rdf:resource="&xsd;integer"/>
      </rdf:Property>

      <rdfs:Datatype rdf:about="&xsd;integer"/>

      <biblioteka:Kniha rdf:ID="JavaEfektivne">
            <biblioteka:pocetStran>200</biblioteka:pocetStran>
      </biblioteka:Kniha>
</rdf:RDF>
```

Příklad 3: RDF Schema

2.4 RDFa

RDFa [10], [45], [46], [47] (Resource Description Framework in attributes) rozšiřuje klasický značkovací jazyk (X)HTML (eXtensible Hyper Text Markup Language) o sadu atributů pro popis metadat (viz kapitola 2.5 na straně 9).

Pokud například vytvoříme nějaký webový dokument a použije se v něm element pro odstavec *<p>*, můžeme tento element díky RDFa rozšířit i o atribut *property=”dc:article”*. Dalším příkladem je použití elementu pro nadpis (pod)článku (*<h1>, <h2> ... <h7>*), který lze rozšířit o RDFa atribut *property=”dc:title”*.

Účel RDFa je, aby se každý webový dokument dal snadno prezentovat nejen lidem, ale i strojovým agentům.

Příklad webového dokumentu v (X)HTML s využitím RDFa naleznete níže (viz Příklad 4 na straně 8).

```
<div xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:=cal:"http://purl.org/cal/elements/1.1/">
      <h1 property="dc:title">Title</h1>
      <p property=”dc:article”>
            Lorem ipsum dolor sit amet, consectetur adipiscing elit.
            <a property=”cal:url” href=”http://www.e-
            nakupujeme.cz/”>Phasellus</a> sed erat augue. Vivamus ac tortor
            vel tontor fementum pharetra ulticies sed lorem.
      </p>
      <h2 property="dc:subtitle">Subtitle</h2>
      <h3 property="dc:creator">Author</h3>
</div>
```

Příklad 4: RDFa v XHTML

Příklad lze převést do RDF N3 notace (viz Příklad 5 na straně 9).

```
<http://www.example.com/test>
      <http://purl.org/dc/elements/1.1/title> "Title";
      <http://purl.org/dc/elements/1.1/article> "Lorem ipsum ...";
      <http://purl.org/cal/elements/1.1/url#href> "http://www.e-
nakupujeme.cz/";
      <http://purl.org/cal/elements/1.1/url> "Phasellus";
      <http://purl.org/dc/elements/1.1/subtitle> "Subtitle";
      <http://purl.org/dc/elements/1.1/creator> "Author";
```

Příklad 5: RDF N3 notace

2.4.1 Atributy RDFa

- **about** – URI adresa, která specifikuje zdroj metadat
- **rel** a **rev –** specifikuje vztah s jinými zdroji
- **href, src** a **resource –** definuje zdroj
- **property** – vlastnost obsahu elementu
- **content –** určuje hodnotu predikátu (při jejím uvedení se přepíše hodnota elementu)
- **datatype –** definuje datový typ hodnoty
- **typeof** – atribut pro specifikaci RDF typu

2.4.2 Výhody a nevýhody RDFa

Výhody:

- u RDFa se dají využít všechny již existující ontologie
- některé vyhledávače jako Google podporují RDFa a umožňují optimalizaci SEO u webových stránek

Nevýhody:

- schází větší podpora prohlížečů

2.5 Metadata

Metadata[16], [70], [71], [72] jsou data, která nesou informace o základních datech, zkráceně bychom je mohli definovat jako data o datech. S tímto pojmem se setkáváme především v souvislosti s elektronickými zdroji, jako například u obrázků, dokumentů, hudebních souborů, atd. Metadata mohou nést například tyto informace:

- Obrázky – šířka, výška, barevná paleta, název, atd.
- Dokumenty – autor, vydavatel, počet stran, číslo vydání, atd.
- Hudební soubory – jméno skupiny, jméno zpěváka, jméno distributora, hudební žánr, atd.

2.5.1 Kde se používají metadata

Metadata v dnešní době zahrnují mnoho různých oblastí a uživatelé se s nimi setkávají, aniž by o nich věděli.

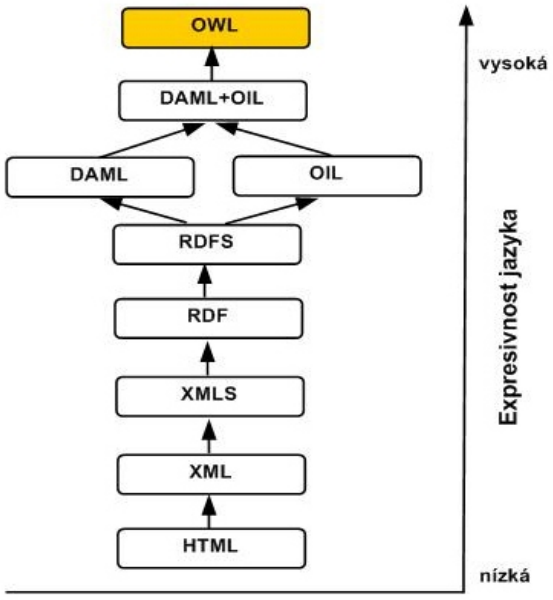
- **Metadata v SVN –** SVN (Subversion) je systém pro správu zdrojových kódů projektů, poskytující každé složce skryté soubory, které odhalují důležité informace o kódu repozitářů.
- **Metadata na internetu –** každá webová stránka umožňuje začlenění různých typů metadat (popis názvu stránky, použitá klíčová slova, atd.), která popisují zobrazovanou stránku. Metadata jsou v dnešní době jedním z klíčových prvků pro SEO optimalizaci webu.
- **Hudební soubory, obrázky, dokumenty, atd. -** každý soubor v počítači obsahuje skrytá metadata, nebo-li tagy, která nesou důležité informace o souboru. Například u obrázku je to jeho šířka, výška, název, barevná paleta, atd.
- Metadata můžeme nálezt už i ve zdravotnictví, v dopravních podnicích, chytrých telefonech, atd.

2.6 Ontologie

V informatice je ontologie[5], [6], nebo-li slovník, formalizovaný popis určité problematiky. Ontologii můžeme chápat jako datový model, který reprezentuje množinu dat a jejich vztahy. Cílem ontologie je definovat pojmy a vztahy v dané problematice.

2.6.1 OWL

OWL (Web Ontology Language)[7], [8], [9], [10], [11] je značkovací jazyk, vytvořený konsorciem W3C pro definování ontologií v prostředí sémantického webu. Je určen pro aplikace, které potřebují data prezentovat nejen lidem, ale i strojovým agentům. OWL umožňuje větší interpretovatelnost webové obsahu, než doposud známé technologie XML, RDF a RDFS, a to jen díky poskytnutí dalších slovníků spolu s formální sémantikou. O OWL můžeme říci, že je rozšíření RDF a RDFS slovníku (viz Ilustrace 8 na straně 10).



Ilustrace 8: Cesta k OWL[12]

OWL můžeme rozdělit na tyto 3 podjazyky:

1. OWL Full – jedná se o plnou variantu jazyka OWL, která umožňuje syntaktickou svobodu
2. OWL DL – umožňuje maximální výraznost a zachovává co nejmenší výpočetní náročnost
3. OWL Lite – nejjednodušší verze OWL, je určena pro uživatele, kteří potřebují klasifikovat hierarchii a jednoduchá omezení

2.7 Formáty serializace RDF

RDF je sám o sobě model, který pro prezentaci svých tvrzení využívá grafy (viz kapitola 2.2.1 na straně 4). Aby mohly s RDF grafy pracovat i strojoví agenti, je potřeba tento model uchovávat v určité podobě.

2.7.1 RDF/XML

RDF/XML[66] je navržený konsorciem W3C a jedná se o nejpoužívanější způsob prezentace RDF pomocí XML syntaxe.

Zápis příkladu z Ilustrace 5 na straně 2 by ve formátu RDF/XML vypadal následovně [15] (viz Příklad 6 na straně 11)

```
<rdf:RDF
  xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
  xmlns:vcard='http://www.w3.org/2001/vcard-rdf/3.0#'
 >
  <rdf:Description rdf:about='http://somewhere/JohnSmith'>
    <vcard:FN>John Smith</vcard:FN>
    <vcard:N rdf:nodeID="A0"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A0">
    <vcard:Given>John</vcard:Given>
    <vcard:Family>Smith</vcard:Family>
  </rdf:Description>
</rdf:RDF>
```

Příklad 6: Zápis v RDF/XML

2.7.2 N3

N3[67], nebo-li Notation3 (non-XML serialization of Resource Description Framework models), je jazyk navržený konsorciem W3C jako kompaktnější a čitelnější než XML syntax v RDF (viz 2.7.1 na straně 11). (přípona *.n3)

Zápis příkladu z Ilustrace 5 na straně 2 by ve formátu N3 vypadal následovně [15] (viz Příklad 7 na straně 12).

```
@prefix vcard: <http://xmlns.com/foat/0.1/>
@prefix rdf: <http://www.w3.org/1999/02/22-syntax-nx#>

_: http://somewhere/JohnSmith;
      vcard:FN “John Smith”;
      vcard:N rdf:nodeID=”A0”;
      rdf:nodeID=”A0”;
            vcard:Given “John”;
            vcard:Family “Smith”;
```

Příklad 7: Zápis v N3

2.7.3 Turtle

Zkratka Turtle[17] (někdy se i v literatuře objevuje Turtles) vznikla z anglické fráze *Terse RDF Triple Language* a jedná se o podmnožinu existujícího jazyka N3 (viz kapitola 2.7.2 na straně 11). Turtle je velice populární syntax mezi vývojáři sémantického webu a je používána jako čitelnější alternativa k syntaxi RDF/XML. (přípona *.ttl)

Zápis syntaxe formátu Turtle naleznete níže (viz Příklad 8 na straně 12).

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix ex: <http://example.org/stuff/1.0/> .

<http://www.w3.org/TR/rdf-syntax-grammar>
  dc:title "RDF/XML Syntax Specification (Revised)" ;
  ex:editor [
    ex:fullname "Dave Beckett";
    ex:homePage <http://purl.org/net/dajobe/>
  ] .
```

Příklad 8: Syntax formátu Turtle

2.7.4 N-Triples

N-Triples[68] je jazyk, využívající jednoduchý formát a každé jeho tvrzení je napsáno na jednom řádku textového souboru. Syntax N-Triples je podmnožinou jazyků Turtle (viz kapitola 2.7.3 na straně 12) a Notation3 (viz kapitola 2.7.2 na straně 11). (přípona souboru většinou bývá *.nt)

Zápis syntaxe formátu N-Triples naleznete níže (viz Příklad 9 na straně 12).

```
<http://www.w3.org/2001/sw/RDFCore/ntriples/>
<http://purl.org/dc/elements/1.1/creator> "Dave Beckett" .
<http://www.w3.org/2001/sw/RDFCore/ntriples/>
<http://purl.org/dc/elements/1.1/creator> "Art Barstow" .
<http://www.w3.org/2001/sw/RDFCore/ntriples/>
<http://purl.org/dc/elements/1.1/publisher> <http://www.w3.org/> .
```

Příklad 9: syntax N-Triples

2.7.5 TriX

TriX[18] je experimentální alternativa k vyjádření serializace RDF trojic v XML a byla vyvinuta společně s Jeremy Carroll (HP Labs) a Patric Stickler (Nokia). TriX si klade za cíl poskytovat vysoce normalizované a konzistentní XML pro reprezentaci grafů RDF.

Pro snadnější pochopení syntaxe TriX využijeme ukázkový příklad z webu Nokia[19] (viz Příklad 10 na straně 13).

```
<?xml version="1.0"?>
<TriX xmlns="http://www.w3.org/2004/03/trix/trix-1/">
      <graph>
            <triple>
                  <uri>http://example.org/Bob</uri>
                  <uri>http://example.org/wife</uri>
                  <uri>http://example.org/Mary</uri>
            </triple>
            <triple>
                  <uri>http://example.org/Bob</uri>
                  <uri>http://example.org/name</uri>
                  <plainLiteral>Bob</plainLiteral>
            </triple>
            <triple>
                  <uri>http://example.org/Mary</uri>
                  <uri>http://example.org/age</uri>
                  <typedLiteral
      datatype="http://www.w3.org/2001/XMLSchema#integer">32</typedLiteral>
            </triple>
      </graph>
      <graph>
            <triple>
                  <uri>http://example.org/Widget</uri>
                  <uri>http://example.org/dimensions</uri>
…
```

Příklad 10: syntax TriX

2.8 Sémantický desktop

Sémantický desktop[21], [22], [23], [24] je založený na myšlence, že počítač nejen manipuluje s daty, ale snaží se i vytvářet logické struktury, ve kterých hledá vzájemnou spojitost. Mohli bychom říci, že se snaží struktuře porozumět jako běžně uvažující člověk. Tím by nabízel lepší možnosti manipulace a vyhledávání dat. Hlavním představitelem sémantického desktopu je v současnosti Nepomuk.

Existují 3 funkční implementace:

1. v grafickém prostředí KDE
2. pracující v Javě
3. poskytována v rámci SaS (komerční produkt)

2.8.1 Jaká je budoucnost sémantického desktopu?

Myšlenka sémantického desktopu je zajímavá a přináší řadu pozitiv, ale i negativ. Ač se jedná o projekt s potenciálem, který bude moci uživatelům nabídnout řadu služeb a výhod, tak obsahuje i různá bezpečnostní rizika, která mohou mít dopad na rozvoj ekonomiky či sociální život jednotlivce.

Dosavadní vývoj sémantického desktopu, i přes možná bezpečnostní rizika, nasvědčuje tomu, že tento projekt je „první vlaštovkou“ ve správě a sdílení osobních dat, atd.

2.8.2 Co je projekt Nepomuk?

Hlavním cílem projektu NEPOMUK – The Social Semantic Desktop[43] je poskytnout infrastrukturu, rozšiřující osobní desktop do prostředí podporujícího jak správu osobních informací, tak i sdílení a výměnu těchto informací v rámci sociálních a organizačních vztahů.

Tento proces probíhá na základě práce s metadaty (viz kapitola 2.5 na straně 9), která jsou získávána automaticky z metadat souborů nebo činností uživatele.

2.9 RDF úložiště

RDF úložiště[69] je datová základna, která slouží k ukládání RDF dat. Úložiště poskytuje přístup ke svým datům prostřednictvím dotazovacích jazyků (viz kapitola 2.10 na straně 16). Nejčastějším jazykem v RDF je SPARQL (viz kapitola 2.10.4 na straně 17).

2.9.1 Oracle RDF

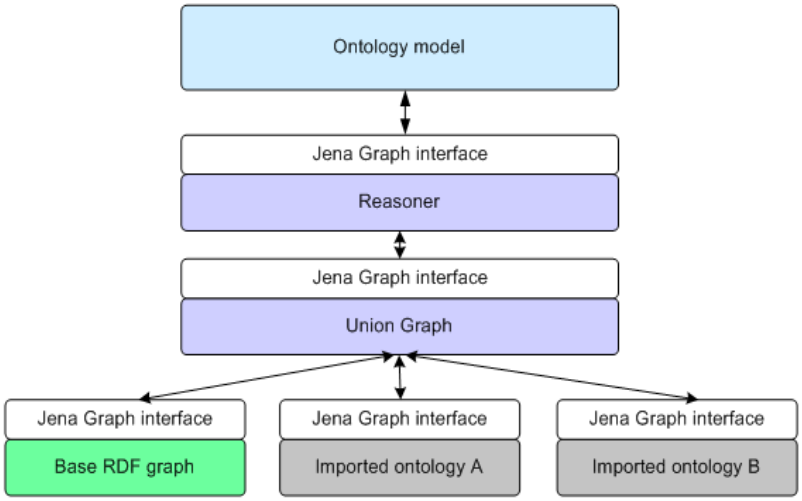
RDF úložiště[26], [27], [28], vyvíjené společností Oracle, nabízí robustní a bezpečnou platformu pro ukládání RDF a OWL (viz kapitola 2.6.1 na straně 10) dat. To umožňuje efektivní ukládání a vyhledávání dat pomocí dotazů, které jsou vylepšeny vztahy k datům pomocí ontologií. Zvláštností Oracle RDF je, že poskytuje přístup ke svým uloženým datům nejen pomocí dotazovacího jazyka SPARQL (viz kapitola 2.10.4 na straně 17), ale i prostřednictvím jazyka SQL nebo Java rozhraní.

2.9.2 Jena

Jena[10], [11], [14], [29], byla vyvinuta společností Hewlett-Packard Company, slouží jako open source framework pro tvorbu aplikací sémantického webu, vytvořeného v programovacím jazyce Java (viz kapitola 4.1.1.2 na straně 25).

Jena obsahuje API (viz Ilustrace 9 na straně 15) pro práci s RDF (viz kapitola 2.2 na straně 4) daty a poskytuje podporu pro značkovací jazyk OWL (viz kapitola 2.6.1 na straně 10). Hlavním dotazovacím jazykem úložiště je SPARQL (viz kapitola 2.10.4 na straně 17). Jena navíc umožňuje vytvářet nativní (viz kapitola 2.9.2.2 na straně 15) nebo dočasná (viz kapitola 2.9.2.1 na straně 15) úložiště.

Samotné úložiště dovoluje i importy a exporty RDF dat mezi aplikacemi a je možné použít tyto formáty: RDF/XML, N3, N-Triples, atd. (více o formátech naleznete v kapitole 2.7 na straně 11).



Ilustrace 9: Architektura Jena Ontology API (struktura pro importy)[29]

2.9.2.1 Paměťové úložiště

U paměťového úložiště jsou RDF data ukládána do operační paměti počítače.

Mezi **výhody** tohoto úložiště patří:

- nulová konfigurace
- rychlost vkládání, vyhledávání a odvozování

Nevýhody:

- kapacitní omezení
- velká spotřeba operační paměti

2.9.2.2 Nativní úložiště

U nativního úložiště[81] , nebo-li též perzistentního úložiště, jsou RDF data ukládána jako soubory v binárním formátu na pevný disk. Tento formát umožňuje kompaktní ukládání a rychlé vyhledávání dat. Nativní úložiště je určeno pro práci s velkými RDF daty.

Mezi **výhody** tohoto úložiště patří:

- velikost úložiště není omezena operační pamětí
- není nutná instalace lokálního databázového serveru
- rychlé vyhledávání

Nevýhody:

- nově vytvořené prázdné úložiště potřebuje určitý datový prostor (dle výchozího nastavení úložiště se může jednat o desítky MB či GB)

2.9.3 Sesame

Sesame[34], [35], [36], vyvíjený společností Aduna, je standardní rámec pro ukládání a dotazování RDF dat. Je postavený na programovacím jazyce Java (viz kapitola 4.1.1.2 na straně 25) a je plně konfigurovatelný a rozšiřitelný s ohledem na různé ukládací mechanismy, formáty

RDF dat nebo dotazovací jazyky. RDF úložiště plně podporuje dotazovací jazyk SPARQL (viz kapitola 2.10.4 na straně 17).

Samotné úložiště dovoluje i importy a exporty RDF dat mezi aplikacemi a je možné použít tyto formáty: Notation3, N-Triples, RDF/XML, Trig, atd. (více o formátech naleznete v kapitole 2.7 na straně 11).

2.10 Dotazovací jazyky v RDF úložišti

Dotazovací jazyky[34] jsou nástroje určené pro tvorbu příkazů, pomocí nichž lze s daty manipulovat. Existuje několik jazyků pro práci s RDF úložištěm, jako je RQL, SPARQL, atd. (viz podkapitoly níže).

2.10.1 RQL

RQL[30], [31], [32] (RDF query language) je jedním z prvních deklarativních dotazovacích jazyků pro RDF (viz kapitola 2.2 na straně 4), který vyvinul FORTH – Institute of Computer Science. Cílem instituce bylo vyvinout podobný dotazovací jazyk pro RDF jako je jazyk SQL pro relační databáze.

Charakteristikou jazyka RQL je, že se používá pro dotazování výslovně uvedených trojic RDF/S grafů, nebo-li poskytuje možnosti tvořit dotazy kombinující jak schéma, tak datovou část.

Ukázku několika příkazů[33] naleznete níže (viz Příklad 11 na straně 16).

```
SELECT      P, range(P)
FROM        Dproperty{P}
WHERE       domain(P) >= Painter



SELECT      X, (  SELECT      $C,    (SELECT     @P, Y
                                     FROM        {W; ^$C} ^@P {Y}
                                     WHERE       namespace($C) != ns1 and
                                                 namespace(@P) != ns1
                                     )
                  FROM ^$C {X})
FROM        Resource {X}
USING       NAMESPACE
ns1=&http://139.91.183.30:9090/RDF/VRP/Examples/dema/admin.rdf#
```

Příklad 11: Ukázka několika příkladu jazyka RQL

2.10.2 eRQL

eRQL[31] (easy RDF Query Language) nabízí radikální zjednodušení syntaxe od jazyka RQL (viz kapitola 2.10.1 na straně 16). Hlavním cílem jazyka je nabídnout intuitivní syntax, která by byla jednoduchá a mohla být používána uživateli bez základních znalostí ontologie (viz kapitola 2.6 na straně 10). Tento cíl byl dosažen pomocí Google-like syntaxe.

eRQL nabízí 3 typy dotazu[31]:

1. **Jednoduché dotazy** – dotazy, obsahující pouze klíčové slovo
2. **Dotazy „prozkoumávající“ okolí slova** – jedná se o jednoduché dotazy, které navíc k výsledku přidávají data související s klíčovým slovem

3. **Konjunkce (AND) a disjunkce (OR) dotazů** – předchozí typy dotazů mohou být kombinovány s operátory AND a OR

Ukázku syntaxe jazyka eRQL[33] naleznete v textu níže (viz Příklad 12 na straně 17).

```
<Query> ::= <Disjunction>

<Disjunction> ::= <Conjunction> | <Disjunction> OR <Conjunction>
<Conjunction> ::= <SubQuery> |
<Conjunction> AND <SubQuery> |
<Conjunction> <SubQuery>

<SubQuery> ::= <Literal> |
{ <Disjunction> } | ~ <Disjunction> | { <Literal> } |
< <Disjunction> > | < <Literal> > | [ <Disjunction> ] |
[ <Literal> ] | ( <Disjunction> )

<Literal> ::= " <STRING_LITERAL> " | <STRING_LITERAL> | <URI_LITERAL>
```

Příklad 12: eRQL syntax

2.10.3 SeRQL

SeRQL[31] (Sesame RDF Query Language) je dotazovací jazyk specifický pro úložiště Sesame (viz kapitola 2.9.3 na straně 15), který vychází z principů jazyka RQL, ale obsahuje nové vlastnosti. Samotná syntax jazyka se snaží napodobovat známý dotazovací jazyk SQL a umožňuje 2 typy dotazů.

- **Výběrové dotazy –** (SELECT) dotaz, vracející tabulku nalezených hodnot (viz Příklad 13 na straně 17).

```
SELECT PersonURI
FROM {PersonURI} foaf:name {PersonName}
WHERE PersonName = "Jan"
USING NAMESPACE foaf = <http://xmlns.com/foaf/0.1/>
```

Příklad 13: Výběrový dotaz

- **Konstrukční dotazy –** (CONSTRUCT) dotaz, vytvářející podgrafy na základě zdrojového RDF grafu (viz Příklad 14 na straně 17).

```
CONSTRUCT *
FROM {PersonURI} foaf:name {PersonName}
WHERE Name = "Jan"
USING NAMESPACE foaf = <http://xmlns.com/foaf/0.1/>
```

Příklad 14: Kontrukční dotaz

2.10.4 SPARQL

Jazyk SPARQL[11], [34], [37], [38], (SPARQL Protocol and RDF Query Language) je dotazovací jazyk pro RDF (viz kapitola 2.2 na straně 4), který byl vyvinut organizací W3C. Ačkoliv se jedná o „mladý“ dotazovací jazyk, tak se velice rychle stal používaným jazykem pro práci s RDF a je podporován významnými RDF úložišti dat jako je Sesame a Jena (viz kapitole 2.9 na straně 14).

Jazyk SPARQL je „read-only“, to znamená, že umožňuje pouze získávat data z RDF úložiště. Veškeré vkládání či změna dat se provádí pomocí SPARQL Update Language (někdy označován jako SPARUL).

SPARQL umožňuje 4 druhy dotazů (viz kapitoly níže).

2.10.4.1 SELECT

Dotaz *SELECT* (viz Příklad 15 na straně 18) se podobá SELECTu v dotazovacím jazyku SQL. V klauzuli SELECT jsou označeny proměnné, které se mají objevit ve výsledku. Klauzule WHERE poskytuje grafový vzor, který je vázaný na hledané proměnné. Výsledkem dotazu SELECT je tabulka.

```
PREFIX foat:     <http://xmlns.com/foat/0.1/>

SELECT      ?name
WHERE       { ?x foat:name ?name }
ORDRE BY    ?name
LIMIT       5
OFFSET      10
```

Příklad 15: SPARQL - výběrový dotaz

2.10.4.2 ASK

Dotaz *ASK* (viz Příklad 16 na straně 18) se od ostatních SPARQL dotazů liší v tom, že vrací pouze boolean hodnotu. Tímto dotazem se tedy RDF úložiště ptáme, zda existuje dotazovaný grafový vzor.

```
PREFIX foat:     <http://xmlns.com/foat/0.1/>
ASK   { ?x foat:name "Alice" }
```

Příklad 16: SPARQL - ASK

2.10.4.3 CONSTRUCT

Dotaz *CONSTRUCT* (viz Příklad 17 na straně 18) vrací RDF graf.

```
PREFIX foat       :     <http://xmlns.com/foat/0.1/>
PREFIX vcard      :     <http://www.w3.org/2001/vcard-rdf/3.0#>
CONSTRUCT   { <http://example.org/person#alice> vcard:FN ?name }
WHERE       { ?x foat: name ?name }
```

Příklad 17: SPARQL - CONSTRUCTOR

2.10.4.4 DESCRIBE

Dotaz *DESCRIBE* (viz Příklad 18 na straně 18) vrací podgraf, vyhovující grafovému vzoru.

```
PREFIX foat:     <http://xmlns.com/foat/0.1/>
DESCRIBE    ?x
WHERE       { ?x foat: name "Alice" }
```

Příklad 18: SPARQL - DESCRIBE

2.10.4.5 Modifikátory výsledků dotazů

U dotazovacího jazyka SPARQL existují i další klauzule, které ovlivňují výslednou podobu dat. [37]

LIMIT, OFFSET a ORDER BY

Klauzule LIMIT, OFFSET a ORDER BY mají stejnou sémantiku jako v SQL.

- **LIMIT** – klauzule stanovuje horní mez počtu vrácených řešení. Využívá se v případech, kdy nám stačí z celkového množství vrácených dat získat jenom část. Příklad syntaxe naleznete v textu níže (viz Příklad 19 na straně 19).

```
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>

SELECT ?name
WHERE { ?x foaf:name ?name }
LIMIT 20
```

Příklad 19: SPARQL - ukázka modifikátoru LIMIT

- **OFFSET** – klauzule stanovuje v jakém rozmezí se výsledek bude zobrazovat. Příklad syntaxe naleznete v textu níže (viz Příklad 20 na straně 19).

```
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>

SELECT  ?name
WHERE   { ?x foaf:name ?name }
ORDER BY ?name
LIMIT   5
OFFSET  10
```

Příklad 20: SPARQL - ukázka modifikátoru OFFSET

- **ORDER BY –** klauzule slouží k řazení nalezených hodnot (vzestupně či sestupně). Příklad syntaxe naleznete v textu níže (viz Příklad 21 na straně 19).

```
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>

SELECT ?name
WHERE { ?x foaf:name ?name ; :empId ?emp }
ORDER BY ?name DESC(?emp)
```

Příklad 21: SPARQL - ukázka modifikátoru ORDER BY

DISTINCT, REDUCED, FILTER, OPTIONAL

- **DISTINCT –** klauzule odstraňuje duplicitní výsledky. Příklad syntaxe naleznete v textu níže (viz Příklad 22 na straně 19).

```
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>

SELECT DISTINCT ?name WHERE { ?x foaf:name ?name }
```

Příklad 22: SPARQL - ukázka modifikátoru DISTINCT

- **REDUCED –** klauzule pracuje na podobném principu jako DISTINCT, jen je zde snížená eliminace dat. U klauzule DISTINCT se vždy odstranily všechny duplicity, ale u klauzule REDUCED mohou být eliminovány všechny, některé nebo žádné duplicity. Příklad syntaxe naleznete v textu níže (viz Příklad 23 na straně 20).

```
PREFIX foat:      <http://xmlns.com/foaf/0.1/>
SELECT REDUCED ?name WHERE { ?x foat:name ?name}
```

Příklad 23: SPARQL - ukázka modifikátoru REDUCED

- **FILTER –** díky této klauzuli lze ovlivňovat výsledek pomocí „testování“, zda určitá hodnota je v intervalu, obsahuje podřetězec, atd. Příklad naleznete v textu níže (viz Příklad 24 na straně 20).

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?name ?mbox
WHERE { ?x foaf:name  ?name ;
            foaf:mbox  ?mbox .
        FILTER regex(str(?mbox), "@work.example") }
```

Příklad 24: SPARQL - ukázka modifikátoru FILTER

- **OPTIONAL –** tato klauzule se využije v případě, kdy požadujeme získaní údajů bez ohledu na to, zda jsou všechna data zadána. Příklad naleznete v textu níže (viz Příklad 25 na straně 20).

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox ?hpage
WHERE  { ?x foaf:name  ?name .
         OPTIONAL { ?x foaf:mbox ?mbox } .
         OPTIONAL { ?x foaf:homepage ?hpage }
       }
```

Příklad 25: SPARQL - ukázka modifikátoru OPTIONAL

3 Specifikace požadavků

Opěrným bodem pro vývoj celého systému je *dokument specifikace požadavků*, který bylo nutno navrhnout/vytvořit. Během vývoje byl dokument postupně vylepšován a zdokonalován dle různých problémů či nápadů.

3.1 Zadání

Cílem práce je navrhnout a implementovat nástroj pro organizaci a procházení sbírek (fotografie, hudba, knihy, atd.) založený na technologiích sémantického webu.

1. Seznamte se s technologiemi sémantického webu, metadatového modelu Resource Description Framework (RDF) a prostudujte používané nástroje.
2. Navrhněte uživatelsky příjemný grafický nástroj pro organizaci a procházení sbírek.
3. Navržený nástroj implementujte a testujte nad reálnými daty.
4. Proveďte diskusi dosažených výsledků, možnosti dalšího rozšíření a použití.

3.2 Použité technologie

Přesné technologie pro vývoj systému nejsou zadány, tudíž je možné pracovat s jakoukoliv dostupnou technologií. Jediné, na co je kladen důraz je, aby byl splněn princip technologie sémantického webu (viz kapitola 2 na straně 2), nebo-li aby byly použity tyto technologie či principy:

1. ontologie (co jsou ontologie se dozvíte v kapitole 2.6 na straně 10)
2. RDF úložiště (viz kapitola 2.9 na straně 14)
3. knihovny pro čtení metadat (co jsou metadata se dozvíte v kapitole 2.5 na straně 9)

3.3 Podstránky systému a jejich funkčnosti

V této kapitole se čtenář seznámí s požadavky jednotlivých podstránek.

3.3.1 Úvodní stránka

Úvodní stránka slouží pouze jako rozcestník k různým funkčnostem systému, jako je vyhledávání, export / import, atd. U této stránky je velice důležitý grafický vzhled. (Citace ze života: *„V dnešní době zákazníka nezajímá funkčnost aplikace, ale hlavně vzhled.“*).

Požadavky na stránku *Úvodní stránka*:

1. intuitivní ovládání
2. vytvoření grafického vzhledu
3. dynamicky přizpůsobitelná stránka dle aktuální velikosti okna

3.3.2 Nová položka

Tato podstránka slouží k nahrávání souborů či adresářů do adresářové struktury, získání a uložení metadat do RDF úložiště.

Požadavky na podstránku *Nové položky*:

1. vícevláknové – možnost nahrávání dat s vykonáváním jiných úkonů (jako prohlížení, atd.)
2. zpracovávání ZIP archivů + zapřemýšlet nad možností ostatních archivů
3. možnost výběru adresářové struktury (ruční zadání či výběr z dosavadní struktury)
4. vytvářet duplikaci dat (kopírovat data do známé adresářové struktury – NE linkování dat)
5. získávat metadata
6. získaná metadata ukládat do RDF úložiště
7. vygenerovat název pomocí hash funkce (sha-1 či sha-2)
8. cestu k souboru ukládat do RDF úložiště

3.3.3 Vyhledávání

Tato podstránka slouží k vyhledávání souborů dle určitých kritérií, přesněji podle metadat (jako velikost, rozměr, žánr, atd.).

Požadavky na podstránku *Vyhledávání*:

1. možnost vyhledávání dle typů souborů (obrázky, hudba, videa, atd...)
2. k vyhledávání využít vložené ontologie (hledat dle vlastností)
3. vyhledávání dle všech uložených metadat
4. zobrazení výsledných dat na nové podstránce
5. při zobrazení vyhledaných dat, možnost jejich zobrazení/prezentování či prohlížení vlastností dle použité ontologie
6. není nutností dynamicky vytvářet textové inputy dle ontologie, stačí využít jeden textový input se stromem
7. vyhledané položky lze vyexportovat

3.3.4 Přehled položek

Tato podstránka slouží k přehledu všech vložených položek a k jejich editaci.

Požadavky na podstránku *Přehled položek*:

1. zobrazení celé adresářové struktury
2. editace/vytvoření vlastnosti položky
3. odstranění položky z RDF úložiště a z adresářové struktury
4. zobrazení/prezentování položky pomocí externího programu (winamp, prohlížeč obrázků atd.) - dbát na rozdíl mezi systém Windows a Linux

3.3.5 Ontologie

Tato podstránka slouží ke vkládání či editaci ontologií.

Požadavky na podstránku *Ontologie*:

1. možnost vkládat ontologie dle předem definovaných typů
2. možnost editace/vložení/odstranění vlastností ontologie
3. možnost vložit český název vlastnosti (využití hodnoty @CS)

3.3.6 Import/Export

Tato podstránka slouží k importu/exportu dat do/ze systému. Každý uživatel bude moci přesouvat data z jednoho počítače na druhý, aniž by musel znovu zjišťovat či upravovat metadata.

Požadavky na podstránku *Import/Export*:

1. zvolit k exportu konkrétní soubory či adresáře
2. výsledný export ukládat do struktury RDF/XML
3. import dat bude probíhat pouze pomocí archivovaných souborů (*.zip)
4. při importování dat nepřepisovat původní hodnoty
5. výsledný export bude archivován (*.zip)
6. vícevláknové – možnost exportu/importu dat s vykonáváním jiných úkonů (jako prohlížení, atd.)

4 Analýza problematiky

4.1 Obecná analýza požadavků

V této podkapitole se čtenář dozví, jaké jsou důležité faktory a požadavky, které je nutné splnit pro vytvoření výsledné aplikace.

4.1.1 Volba programovacího jazyka a vývojového prostředí

Zadání (viz kapitola 3.1, strana 21) je obecné a nespecifikuje konkrétní programovací jazyk a typ vývojového prostředí. Jediné důležité požadavky pro vybrání programovacího jazyka jsou tyto:

1. stolní grafická aplikace, možnost práce s aplikací i bez nutného připojení na Internet či lokální server
2. univerzálnost, práce bude fungovat minimálně pod operačním systémem Windows a Linux
3. využití RDF úložiště a dotazovacího jazyka

Se svými znalostmi a zpřesňujícími požadavky (viz výše) jsem měl na výběr pouze z těchto dvou programovacích jazyků:

1. C/C++
2. Java

4.1.1.1 C/C++

C [58], [59], [60] je programovací jazyk, který se převážně používá pro psaní systémového softwaru, ale je možno jej využít i pro aplikace. C je nízkoúrovňový, kompilovaný a relativně minimalistický programovací jazyk. Jazyk C++ je rozšíření jazyka C.

Jazyk C bychom si mohli shrnout v těchto bodech:

- univerzální programovací jazyk nízké úrovně (low level language)
- má velmi úsporné vyjadřování, je strukturovaný, má velký soubor operátorů a moderní datové struktury
- není specializovaný na jednu oblast použití
- pro mnoho úloh je efektivnější a rychlejší než jiné jazyky
- je multiplatformní

Výhody:

- existence mnoha knihoven (pro String, pro čtení metadat, atd.)
- možnost využití různých struktur či ukazatelů
- převážně rychlejší pro práci než ostatní programovací jazyky

- multiplatformní

Nevýhody:

- složitější na pochopení
- někdy nelogické připojování externích knihoven
- nutnost po sobě „čistit“ paměť (programátor sám odpovídá za uvolňování paměti)

Vývojové prostředí:

- **Eclipse** [52], [53] – vývojové prostředí. Jedná se o open source, určené především pro programování v jazyce Java. Flexibilní návrh této platformy dovoluje rozšířit např. seznam podporovaných jazyků (C, C++, PHP, atd.), návrh UML, atd. pomocí pluginů. Standardní instalace neobsahuje plugin pro tvorbu grafických aplikací. Je nutností si daný plugin nainstalovat či vše vytvářet ručně.
- **NetBeans** [54], [55] – vývojové prostředí. Jedná se o open source, určené především pro programování v jazyce Java. Toto vývojové prostředí ovšem podporuje prakticky jakýkoliv programovací jazyk (C, C++, PHP, atd.). Výhoda tohoto prostředí spočívá v tom, že je primárně určena pro vývoj GUI aplikací.
- **QT** [56] - patří mezi multiplatformní knihovny pro vytváření programů s grafickým rozhraním. QT je převážně pro programovací jazyk C++, ale existuje i pro mnoho dalších jazyků (C, C#, Python, atd.). Výhodou QT je přehledně zpracovaná dokumentace a také intuitivní vývojové programy QT Creator nebo QT Designer. Aplikace vytvořené pro grafické prostředí používají nativní vzhled operačního systému, takže vyvinuté aplikace se vždy přizpůsobí používanému prostředí.
- **Visual Studio** [57] – vývojové prostředí vyvinuté firmou Microsoft. Toto prostředí slouží jak pro vývoj konzolových aplikací, tak i aplikací s grafickým rozhraním a lze využít vestavěné jazyky jako C/C++, VB.NET, C#, atd.

4.1.1.2 Java

Java[49], [50], [51] je objektově orientovaný programovací jazyk, vyvinutý firmou Sun Mircosystems, založený na principech programovacích jazyků C a C++. Díky svým výhodám (viz níže) patří Java mezi nejpoužívanější programovací jazyky na světě.

Výhody:

- **jednoduchost** – jeho syntax je zjednodušenou podobou syntaxe jazyka C a C++
- **objektově orientovaný**
- **nezávislý na architektuře** – vytvořená aplikace běží na libovolném operačním systému nebo libovolné architektuře
- **víceúlohový** – podporuje zpracování vícevláknových aplikací

Nevýhody:

- **pomalé načítání** – neprovádí tzv. statickou kompilaci (jako C++)

- **větší paměťová náročnost**
- **psaní různých konstrukcí** – neumožňuje psaní konstrukcí známých například pod programovacím jazykem C

Vývojové prostředí

- **Eclipse –** viz kapitola 4.1.1.1, strana 24
- **NetBeans** - viz kapitola 4.1.1.1, strana 24

4.1.1.3 Vlákna

Uživatel bude předpokládat od aplikace, že bude moci vykonávat více paralelních úloh, neboli bude moci využívat grafické prostředí, aniž by z větší části pocítil zpoždění odezvy grafických prvků (grafické prvky jsou zde např. tlačítka, výběrová pole, atd.).

4.1.1.4 Volba programovacího jazyka

Vybrat výsledný programovací jazyk s ohledem na známé výhody a nevýhody bylo velice složité. Dle mého názoru je lepší vyvíjet aplikaci v programovacím jazyce Java, z důvodu lepší znalosti jazyka a přehlednosti. Ale na druhou stranu se nemůže opomenout rychlost výsledného programu a možnost využívat tvorbu různých konstrukcí v programovacím jazyce C/C++.

Po delším přemýšlení bylo rozhodnuto, že při vývoji aplikace se dá přednost programovacímu jazyku C/C++.

Po zvolení konkrétního jazyka už jenom zbývalo vybrat vývojové prostředí. Při testování vyšlo nejlépe prostředí QT Nokian, které ostatní předčilo svou rychlostí a intuitivním prostředím.

Během zpracování diplomové práce bylo zjištěno, že implementace některých důležitých částí (knihovny pro práci s metadaty) do programovací jazyka C/C++ je velice komplikovaná. Proto se diplomová práce začala vyvíjet v programovacím jazyce Java a pod vývojovým prostředím NetBeans. Toto prostředí bylo vybráno z důvodu jeho znalosti.

4.1.2 Grafický vzhled aplikace

Cílem práce je i vytvořit uživatelsky příjemný vzhled výsledné aplikace. Protože nemám dostačující zkušenosti s grafickou tvorbou, tak bude k projektu přibrán externí grafik, který navrhne úvodní stránku aplikace.

Rčení ze života: „*V dnešní době se neprodává funkčnost, ale vzhled.“*

4.1.3 RDF úložiště

RDF úložiště (nebo-li též znalostní úložiště) je datová základna, kde jsou uložena RDF data. Úložiště poskytuje přístup k uloženým datům a to prostřednictvím dotazovacího jazyka (viz kapitola 2.10 Dotazovací jazyky v RDF úložišti, strana 16), v RDF je to nejčastěji SPARQL (viz kapitola 2.10.4 SPARQL, strana 17).

4.1.3.1 Jena

Více o RDF úložišti Jena naleznete v kapitole 2.9.2 Jena, strana 14.

Výhody:

- není nutná instalace žádné databáze nebo jiných softwarových produktů 3. stran
- přehledná dokumentace
- snadné napojení knihoven
- rychlost při práci s trojicemi
- různé typy úložišť (dočasná, perzistentní, atd.)
- API pro programovací jazyky C++, C#, Java, atd.

Nevýhody:

- občas nepřehledné ukázkové příklady

4.1.3.2 Sesame 2

Více o RDF úložišti Sesame naleznete v kapitole 2.9.3 Sesame, strana 15.

Výhody:

- není nutná instalace žádné databáze nebo jiných softwarových produktů 3. stran
- různé typy úložišť (dočasná, perzistentní, atd.)
- API pro programovací jazyky C++, C#, Java, PHP, atd.

Nevýhody:

- nepřehledná dokumentace
- složitější napojení knihoven
- rychlost práce s trojicemi

4.1.3.3 Porovnání rychlostí mezi úložišti

Tabulka 1 na straně 28 ukazuje rychlost ukládání trojic do různých úložišť (tabulka citována z použité literatury [61]).

Význam sloupečků:

- **1. sloupeček –** označuje typ testovaného úložiště
- **2. - 4. sloupeček** – určuje počet testovaných trojic v miliónech (25M = 25 000 000)

Výsledné hodnoty jsou zapsány ve tvaru *[den:]hodiny:minuty:sekundy.*

Úložiště \ Počet trojic	1M	25M	100M
Sesame	00:02:59	12:17:05	3:06:27:35
Jena TDB	00:00:49	00:16:53	01:34:14
Jena SDB	00:02:09	04:04:38	1:14:53:08
Virtuoso TS	00:00:23	00:39:24	07:56:47
MySQL	00:00:06	00:02:03	00:11:45
Virtuoso SQL	00:00:34	00:17:15	01:03:53

Tabulka 1: Přehled času při nahrávání souborů o velikosti x ([den]:hh:mm:ss) [61]

Tabulka 2 na straně 28 ukazuje propustnost dotazů za sekundu u RDF úložiště Jena TDB (tabulka citována z použité literatury [61]).

Význam sloupečků:

- **1. sloupeček –** označení dotazu
- **2. - 4. sloupeček** – určuje počet testovaných trojic v miliónech (25M = 25 000 000)

Dotaz \ Počet trojic	1M	25M	100M
Query 1	494,3	164,8	34,9
Query 2	60,9	50,8	38,2
Query 3	451,3	140,8	28,4
Query 4	428,6	116,3	24,8
Query 5	1,8	0,1	0,04
Query 6	59,5	2,4	0,1
Query 7	188,8	28,1	6,3
Query 8	158,8	27	8,4

Tabulka 2: Propustnost dotazů (QpS – Queries per Second) u znalostního úložiště Jena TDB [61]

Tabulka 3 na straně 28 ukazuje možný počet dotazů za hodinu (QMpH) u různého počtu klientů v RDF úložišti Jena TDB (tabulka citována z použité literatury [61]).

Význam sloupečků:

- **1. sloupeček –** počet testovaných trojic (1M = 1 000 000)
- **2. - 6. sloupeček** – určuje počet klientů

Počet trojic \ počet klientů	1	2	4	8	64
1M	4 450	6 752	9 429	8 453	8 664
25M	353	513	694	536	555

Tabulka 3: Různé dotazy za hodinu (QMpH - Query Mixes per Hour) u různého počtu klientů (čím větší číslo, tím lepší) – Jena TDB [61]

Tabulka 4 na straně 29 ukazuje propustnost dotazů za sekundu u RDF úložiště Sesame v2.2.4 (tabulka citována z použité literatury [61]).

Význam sloupečků:

- **1. sloupeček –** označení dotazu
- **2. - 4. sloupeček** – určuje počet testovaných trojic v miliónech (25M = 25 000 000)

Dotaz \ Počet trojic	**1M**	**25M**	**100M**
Query 1	661,8	200	14,7
Query 2	251,2	168,3	32,5
Query 3	504,8	139,9	12,8
Query 4	452,5	127,9	10,2
Query 5	29,6	1,7	0,5
Query 6	14,1	0,5	0,1
Query 7	86,6	56,7	1,8
Query 8	297	90,3	4,2

Tabulka 4: Propustnost dotazů (QpS – Queries per Second) u RDF úložiště Sesame v2.2.4 [61]

Tabulka 5 na straně 29 ukazuje možný počet dotazů za hodinu (QMpH) u různého počtu klientů v RDF úložišti Sesame v2.2.4 (tabulka citována z použité literatury [61]).

Význam sloupečků:

- **1. sloupeček –** počet testovaných trojic (1M = 1 000 000)
- **2. - 6. sloupeček** – určuje počet klientů

Počet trojic \ počet klientů	**1**	**2**	**4**	**8**	**64**
1M	18 094	19 057	16 460	18 295	16 517
25M	1 343	1 485	1 204	1 300	1 271

Tabulka 5: Různé dotazy za hodinu (QMpH - Query Mixes per Hour) u různého počtu klientů (čím větší číslo, tím lepší) – Sesame v2.2.4 [61]

4.1.3.4 Důvod zvolení RDF úložiště

Při porovnání výhod, nevýhod a rychlostí obou RDF úložišť bylo snadné vybrat framework, který se použije při vývoji aplikace.

Vybraný framework Jena byl zvolen kvůli své přehledné dokumentaci, snadné implementaci do systému a rychlosti manipulace s trojicemi.

4.1.4 Dotazovací jazyk v RDF úložišti

Výběr dotazovacího jazyka byl velmi jednoduchý. Obě zmiňovaná RDF úložiště (viz kapitola výše) uváděla příklady v dokumentu pomocí dotazovacího jazyka SPARQL.

Více o dotazovacím jazyku SPARQL naleznete v kapitole 2.9.2 na straně 14.

Výhody:

- nástupce předchozích dotazovacích jazyků (eRQL, SeRQL, atd.)
- velké množství modifikátorů
- přehledná dokumentace a příklady jeho použití

- novější verze SPARQL umožňuje nejen výběrové dotazy, ale i akční (delete, insert, atd.)

Nevýhody:

- neumožňuje akční dotaz UPDATE
- na první pohled se SPARQL podobá MySQL, ale při detailnější zkoumání je jazyk více podobný logice Pythonu

4.2 Analýza podstránek

V této podkapitole se čtenář dozví, jaké podstránky budou v aplikaci vytvořeny, co se předpokládá o jejich funkčnosti a jaké jsou možnosti realizace.

4.2.1 Úvodní stránka

Úvodní stránka, nebo-li rozcestník, nebude mít žádnou speciální funkčnost. Hlavní funkce této stránky bude zobrazit ikony pro přechod na další funkční části systému (na různé podstránky).

Požadavky:

- příjemný uživatelský vzhled + snadná použitelnost (využije se znalostí externího grafika)
- tlačítka graficky zvýrazněna (nejlépe obrázkem)
- kontrola a případné vytvoření povinných adresářů

4.2.2 Vkládání dat

Tato stránka slouží k uložení dat a jejich vlastností do RDF úložiště.

Požadavky:

- práce s archivem
- ukládání cesty souboru do RDF úložiště či do jeho jména
- vygenerovat název pomocí hash funkce (sha-1 či sha-2)
- víceúlohový (možnost vkládat více dat najednou – paralelní vkládání)
- získávat metadata
- hodnoty získané z metadat ukládat do RDF úložiště

4.2.2.1 Práce s archivem

Existuje široká škála archivů jako je např. *.tar, *.tar.z, *.tar.gz, *.gz, *.rar, *.zip, atd. Pokud by výsledná aplikace měla umět využívat všechny typy archivů, tak by nezbyl čas na splnění ostatních požadavků. Proto bylo se zadavatelem dohodnuto, že aplikace bude brát v potaz pouze Zip archivy.

4.2.2.2 Duplikování či linkování souborů?

Ač se může zdát tato otázka velice jednoduchá, tak i přes to bylo nutností zanalyzovat výhody a nevýhody použití obou principů.

Duplikování dat	Linkování dat
VÝHODY	
• snadnější manipulace s daty • všechny soubory okamžitě k dispozici • menší pravděpodobnost náhodného odstranění a vytvoření nekonzistentního RDF úložiště	• nevytváří duplikované soubory • rychlejší nahrávání (nemusí kopírovat data)
NEVÝHODY	
• soubor může mít více svých kopií na různých místech disku • nárůst obsazenosti místa na disku	• velká pravděpodobnost odstranění z disku • možnost linkovat soubory i z externích disků • nelze dobře docílit konzistentnosti RDF úložiště při odstranění souboru

Tabulka 6: Výhody a nevýhody u duplikování a linkování dat

Ačkoliv oba principy mají své klady i zápory, bude nejvýhodnější využít principu duplikování dat, protože uživatel má všechny soubory „pod jednou střechou“ a při odstraňování souborů nemusí mít strach, že odstraní nějaká data, která jsou uložena v RDF úložišti. Samotná rychlost kopírování bude záležet převážně na použitém hardwaru.

4.2.2.3 Ukládané soubory

Důležitou otázkou je: „Jaké soubory se budou moci ukládat?“. Cílem práce je vytvořit *správce sbírek založený na technologii sémantického webu*, který bude ukládat obrázkové a hudební soubory, dokumenty, videa, atd. Existuje mnoho různých typů souborů, které lze použít, ale kdybychom měli pracovat se všemi, tak ve výsledku by zpracování těchto dat zabralo veškerý čas a na ostatní důležitější části systému by se nedostalo.

Proto si rozdělíme typy souborů do 3 kategorií:

- obrázkové soubory (např. jpg, png, bmp, atd.)
- hudební soubory (mp3, mp4, wma, atd.)
- dokumenty (pdf, doc, xls, csv, html, atd.)

Jak již bylo řečeno, existuje široká škála různých typů souborů a zpracování všech by trvalo spoustu času. Proto bylo dohodnuto, že se budou zpracovávat pouze níže uvedené typy a v případě dostatku času se systém rozšíří o další.

Obrázkové soubory:

- JPEG, JPG – formát obrázku využívá ztrátovou kompresi

- PNG – formát obrázku pro bezeztrátovou kompresi rastrové grafiky
- GIF – formát určený pro rastrovou grafiku
- BMP – formát určený pro rastrovou grafiku
- TIFF, TIF – souborový formát pro ukládání rastrové počítačové grafiky

Hudební soubory:

- MP3, MP4, M4A, M4P – velice oblíbený hudební formát, založený na ztrátové kompresi zvukových souborů (algoritmus MPEG, MPEG-4)
- OGG – je formát určený pro svobodné šíření a manipulaci s daty
- WMA – komprimovaný zvukový formát vyvinutý jako součást Windows Media
- WAV – běžně používaný zvukový formát, vyvinutý firmou IBM a Microsoft

Dokumenty:

- PDF – souborový formát vyvinutý firmou Adobe

4.2.2.4 Získávání metadat

Získávání metadat je komplikovaná a složitá práce. Neexistuje jednotná knihovna, která by dokázala získat metadata z jakéhokoliv souboru. Je potřeba vždy mít speciální knihovnu
pro daný obor, např. pro obrázky, hudbu, atd.

Existuje samozřejmě široká škála různých knihoven pro programovací jazyk C/C++ či Javu, ale jen některé je možné bezproblémově implementovat do aplikace.

Získávání metadat u obrázků - Metadata Extractor

Metadata-extractor[39] je knihovna určená pro čtení metadat z obrázků. Ačkoliv existuje mnoho různých knihoven v jazyce Java, tak tato knihovna byla zvolena pro tyto vlastnosti:

- snadná implementace
- přehledná dokumentace
- přehledné a snadno pochopitelné ukázkové příklady
- podporuje programovací jazyk C/C++ i Javu

Získávání metadat u hudebních souborů - JAudiotagger:

JAudiotagger[40] je knihovna určená pro čtení metadat z audio souborů. V současné době plně podporuje formáty MP3, MP4 (MP4 audio, M4A a M4P audio), OGG, FLAC, WMA, atd.

JAudiotagger samozřejmě není jediná knihovna kompatibilní s jazykem Java pro práci s metadaty, ale kvůli svým klíčovým výhodám byla upřednostněna:

- snadná implementace

- podporuje programovací jazyk C/C++ i Javu
- poskytuje obecné rozhraní pro 30 nejčastějších atributů
- podporuje čtení chráněných souborů MP4, M4A a MP4P
- podporuje ID3v1 MP3, ID3v11, ID3v2.2, v2.3 a v2.4
- umožňuje snadnou konverzi mezi ID3 tagy
- podporuje kódování UNICODE
- je aktivně vyvíjena

Získávání metadat u dokumentů:

Existuje mnoho různých knihoven pro práci s metadaty u dokumentů. Se zadavatelem nebyla vybrána žádná konkrétní a bylo dohodnuto, že se během vývoje vyzkouší většina knihoven, které se časem najdou.

V seznamu níže uvádím pár knihoven pro práci s metadaty u dokumentů (převážně u PDF souborů)

Knihovna pro získávání metadat v programovacím jazyce C/C++:

- LEADTOOLS – viz WWW: <http://www.codeproject.com/Articles/297669/How-to-Read-Write-and-Edit-PDF-Files-and-Metadata>

Knihovny pro získávání metadat v programovacím jazyce Java:

- Managing Metadata (File and File Store Attributes) – viz WWW: <http://docs.oracle.-com/javase/tutorial/essential/io/fileAttr.html>
- Read PDF Metadata – viz WWW: <http://thinktibits.blogspot.com/2011/05/java-itext-read-pdf-metadata-example.html>

4.2.3 Export/Import dat

Tato podstránka slouží k importu/exportu dat do/ze systému. Každý uživatel bude moci přesouvat data z jednoho počítače na druhý, aniž by musel znovu zjišťovat či upravovat metadata.

Požadavky:

- import dat pouze pomocí archivu
- výsledný export dat bude uložen v archivu
- u exportu bude možnost si vybrat tyto hodnoty: jaké soubory se budou exportovat, kam se budou exportovat a název souboru
- export z RDF úložiště bude ve formátu RDF

4.2.3.1 Vzhled archivu

Bude využíván Zip archiv a jeho struktura bude následující:

- *.ZIP
    - adresářová struktura – soubory obsažené v RDF exportu + kompletní stromová struktura
    - RDF3EXPORT.RDF – obsahuje RDF trojice získané z RDF úložiště

4.2.3.2 Vytváření struktury RDF

Kvůli neznalosti RDF úložiště bylo zpočátku využíváno ruční vytváření RDF struktury. Což by ve výsledném řešení nebyl špatný postup, ale při využití interní funkce v úložišti bylo docíleno rychlejší práce s daty.

Funkci pro import dat naleznete níže (viz Kód 1 na straně 34).

```
InputStream in = FileManager.get().open(sourceRDF);
Model model = TDBFactory.createModel(this.storeage);
model.read(in, "");
in.close();
```

Kód 1: Import dat

Funkci pro export dat naleznete níže (viz Kód 2 na straně 34).

```
Model resultModel = qexec.execDescribe();
out = new FileOutputStream(tmpDir + filename);
resultModel.write(out, "RDF/XML");
```

Kód 2: Export dat

4.2.4 Zobrazování položek

Tato podstránka slouží k přehledu všech vložených položek a jejich vlastností a následné editaci.

Požadavky:

1. zobrazení celé adresářové struktury (zobrazit podstromy, ale jejich obsah dodat až při rozbalení … snažit se snížit náročnost)
2. editace/vytvoření vlastnosti vybrané položky
3. odstranit položku z RDF úložiště a z adresářové struktury
4. zobrazování/prohlížení položek pomocí externích programů

4.2.4.1 Zobrazování/prezentování souborů pomocí externích programů

Zobrazování či prezentování nahraných souborů pomocí externích programů je méně prioritní požadavek, ale je též zajímavý z toho důvodu, že je důležité rozeznávat mezi operační systémy Windows a Linux.

Windows:

Spouštění externích programů pod operačním systémem Windows je jednoduché. Není nutné odlišovat mezi typem souboru a stačí do příkazové řádky zadat univerzální příkaz, který spustí požadovaný program k zadanému souboru (viz Kód 3 na straně 35).

```
String path = "rundll32 url.dll,FileProtocolHandler " + filepath;
Process pr = Runtime.getRuntime().exec(path);
```

Kód 3: Prohlížení souborů pomocí externích programů – kód do příkazového řádku

Linux:

Spouštění externích programů pod operačním systémem Linux je komplikovanější než ve Windows. Nenašel jsem informaci o univerzální příkazu, který by byl schopen spustit požadovaný program k souboru. Je proto důležité rozlišit o jaký typ souboru se jedná (obrázek, hudba, dokument, atd.) a využít konkrétního programu pro spuštění (standardně v operačním systému Linux bývá nainstalován eog – program pro prohlížení obrázků; evince – program pro otevírání souborů PDF; mplayer – program pro přehrávání hudebních souborů) (viz Kód 4 na straně 35).

```
String path = '';
path = "eog " + filepath; // spouštění obrázků
path = "evince " + filepath; // spouštění dokumentů PDF
path = "mplayer " + filepath; // spouštění hudebních souborů
Process pr = Runtime.getRuntime().exec(path);
```

Kód 4: Prohlížení souborů pomocí externích programů - kód pro terminál

4.2.5 Import ontologií

Tato podstránka vznikla až během vývoje systému a slouží ke vkládání či editaci jednotlivých ontologií dle kategorií typů souborů (seznam kategorií naleznete v kapitole 4.2.2.3 na straně 31).

Požadavky:

1. možnost vkládat ontologie dle zvolené kategorie
2. možnost editace/vložení/odstranění jednotlivých vlastností ontologie
3. většina ontologií má popisky v angličtině … připravit systém pro popisky v jiném jazyce

4.2.5.1 Popisky v českém jazyce

Většina dostupných a používaných ontologií je psána v anglickém jazyce, což by u méně jazykově schopných uživatelů mohl být problém. Proto vznikl požadavek na editaci popisků (label) do českého jazyka (viz Příklad 26 na straně 35).

```
<rdfs:label>Label EN</rdfs:label>
<rdfs:label xml:lang="CS">Popisek CZE</rdfs:label>
```

Příklad 26: XML Lang

4.2.6 Vyhledávání

Tato podstránka slouží k vyhledávání souborů dle určitých kritérií, přesněji podle metadat (jako velikost, rozměr, žánr, atd.).

Požadavky:

1. dynamické vytváření textových polí
2. rozdělit vyhledávání dle kategorie typu souboru

3. vyhledávat dle uložených metadat
4. možnost spouštění vyhledaných položek pomocí externích programů
5. možnost exportování vyhledaných položek

5 Návrh

V této části se uživatel seznámí s grafickými náčrty a s diagramy, které zobrazují logiku jednotlivých podstránek.

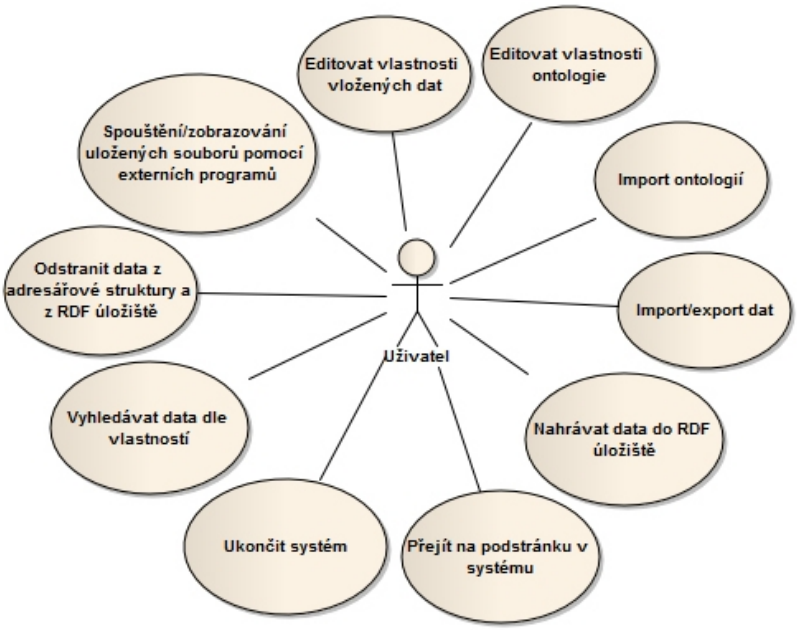
5.1 Použitá technologie

V předchozí kapitole 4 na straně 24, byly analyzovány možné technologie pro vývoj diplomové práce a bylo rozhodnuto, že se využijí tyto technologie:

- programovací jazyk Java (viz kapitola 4.1.1.2 na straně 25)
- vývojové prostředí NetBeans (viz kapitola 4.1.1.2 na straně 25)
- RDF úložiště Jena (viz kapitola 4.1.3.1 na straně 27)
- dotazovací jazyk SPARQL (viz kapitola 4.1.4 na straně 29)
- použité knihovny pro práci s metadaty (viz kapitola 4.2.2.4 na straně 32):
  - čtení metadat z obrázků: Metadata Extractor
  - čtení metadat z hudebních souborů: JAudioTagger
  - čtení metadat z PDF dokumentů: Read PDF Metadata

5.2 Diagram užití

Diagram užití aplikace naleznete níže (viz Ilustrace 10 na straně 37).



Ilustrace 10: Diagram užití programu

5.3 Úvodní stránka

Úvodní stránka, nebo-li rozcestník, nebude mít žádnou speciální funkčnost. Hlavní funkce této stránky bude zobrazit ikony pro přechod na další funkční části systému (na různé podstránky).

5.3.1 Grafický náčrt podstránky

Grafický náčrt podstránky naleznete v kapitole Příloha B: Grafický náčrt podstránky „úvodní stránky“.

5.4 Vkládání dat

Tato stránka slouží k uložení dat a jejich vlastností do RDF úložiště.

5.4.1 Grafický náčrt podstránky

Náčrt podstránky naleznete v kapitole Příloha C: Grafický náčrt podstránky „vkládání dat“).

5.4.2 Diagram aktivit: vkládání nových položek

Diagram aktivit naleznete v kapitole Příloha G: Diagram aktivit „vkládání dat“)

5.5 Export/Import dat

Tato podstránka slouží k importu/exportu dat do/ze systému. Každý uživatel bude moci přesouvat data z jednoho počítače na druhý, aniž by musel znovu zjišťovat či upravovat metadata.

5.5.1 Grafický náčrt podstránky

Náčrt podstránky naleznete v kapitole Příloha D: Grafický náčrt podstránky „export/import dat”).

5.5.2 Diagram aktivit: import dat

Diagram aktivit naleznete v kapitole Příloha H: Diagram aktivit „export/import dat“.

5.5.3 Diagram aktivit: export dat

Diagram aktivit naleznete v kapitole Příloha H: Diagram aktivit „export/import dat“.

5.6 Import ontologií

Tato podstránka slouží ke vkládání či editaci jednotlivých ontologií dle kategorií typů souborů (seznam kategorií naleznete v kapitole 4.2.2.3 na straně 31).

5.6.1 Diagram aktivit: import ontologie

Diagram aktivit naleznete v kapitole Příloha I: Diagram aktivit „import ontologií“.

5.6.2 Diagram aktivit: editace vlastností ontologie

Diagram aktivit naleznete v kapitole Příloha I: Diagram aktivit „import ontologií“).

5.7 Zobrazování položek

Tato podstránka slouží k přehledu všech vložených položek a jejich vlastností a následné editaci.

5.7.1 Grafický náčrt podstránky

Náčrt podstránky naleznete v kapitole Příloha E: Grafický náčrt podstránky „zobrazení položek”.

5.7.2 Diagram aktivit: zobrazení dat

Diagram aktivit naleznete v kapitole Příloha J: Diagram aktivit „zobrazení položek“.

Při výběru dat má uživatel možnost 3 různých akcí. Kvůli složitosti byly akce rozděleny na 3 samostatné diagramy.

- Odstranění – viz kapitola Příloha M: Diagram aktivit „odstranění položky“
- Změna dat – viz kapitola Příloha L: Diagram aktivit „obecná editace“
- Přehrání/zobrazení dat – viz kapitola Příloha N: Diagram aktivit „zobrazování položek“

5.8 Vyhledávání dat

Tato podstránka slouží k vyhledávání souborů dle určitých kritérií, přesněji podle metadat (jako velikost, rozměr, žánr, atd.)

5.8.1 Grafický náčrt podstránky

Náčrt podstránky naleznete v kapitole Příloha F: Grafický náčrt podstránky „vyhledávání dat”).

5.8.2 Diagram aktivit: vyhledání položek a následná manipulace s daty

Diagram aktivit naleznete v kapitole Příloha K: Diagram aktivit „Vyhledávání dat“.

Při výběru dat má uživatel možnost 3 různých akcí. Kvůli složitosti byly akce rozděleny na 3 samostatné diagramy.

- Export dat – je velice podobný klasickému exportu, viz kapitolaPříloha H: Diagram aktivit „export/import dat“
- Změna dat – viz kapitola Příloha L: Diagram aktivit „obecná editace“

- Přehrání/zobrazení dat – viz kapitola Příloha N: Diagram aktivit „zobrazování položek“

6 Implementace a realizace

V této kapitole se uživatel seznámí s programem z hlediska jeho omezení, použitých knihoven a ontologií, testování, vnitřní logiky, atd.

6.1 Použité knihovny a ontologie

Tato podkapitola představuje jednotlivé knihovny a ontologie včetně verzí, které byly použity při vývoji systému.

6.1.1 Metadata Extractor

Metadata-extractor je knihovna určená pro čtení metadat z obrázkových souborů (více informací naleznete v kapitole 4.2.2.4 na straně 32).

Název: metadata-extractor

Verze: 2.5.0-RC2

URL pro popis jednotlivých verzí: http://code.google.com/p/metadata-extractor/wiki/ChangeLog

URL pro stažení knihovny: http://code.google.com/p/metadata-extractor/downloads/list

API dokumentace: http://metadata-extractor.googlecode.com/svn/trunk/Javadoc/index.html

6.1.2 JAudiotagger

JAudiotagger je knihovna určená pro čtení metadat z audio souborů. V současné době plně podporuje formáty MP3, MP4 (MP4 audio, M4A a M4P audio), OGG, FLAC, WMA, atd. (více informací naleznete v kapitole 4.2.2.4 na straně 32).

Název: jaudiotagger

Verze: 2.0.4

URL pro stažení knihovny: http://www.jthink.net/jaudiotagger/maven/source-repository.html

API dokumentace: http://www.jthink.net/jaudiotagger/maven/apidocs/index.html

6.1.3 PDFReader

PDFReader je standardní knihovna, implementovaná v programovacím jazyce Java, která slouží ke čtení metadat z PDF souborů.

Název: PdfReader

Verze: 5.1.3

API dokumentace: http://api.itextpdf.com/itext/com/itextpdf/text/pdf/PdfReader.html

6.1.4 Zip

Balíček java.util.zip je standardní knihovnou programovacího jazyka Java a slouží pro práci s archivem.

Název: java.util.zip

Verze: -

API dokumentace:
http://developer.apple.com/library/mac/documentation/java/reference/javase6_api/api/java/util/zip/package-summary.html

6.1.5 Jena

Jena slouží jako open source framework pro tvorbu aplikací sémantického webu, vytvořeného v programovacím jazyce Java. Byla vyvinuta společností Hewlett-Packard Company v rámci programu pro rozvoj sémantického webu. (viz kapitola 4.1.3.1 na straně 27)

Název: Jena

Verze: 2.6.4

URL pro stažení knihovny: http://sourceforge.net/projects/jena/files/Jena/Jena-2.6.4/

API dokumentace: http://jena.sourceforge.net/ontology/

6.1.6 Ontologie NFO

NFO (NEPOMUK File Ontology) má v úmyslu poskytnout slovní zásobu k vyjádření informací získaných z různých zdrojů. Mezi zdroje patří soubory, kousky softwaru či vzdálené počítače.

Název: NEPOMUK File Ontology

Verze: revize 8 postavená na ontologii NIE (22.03.2007)

URL pro stažení ontologie:
http://www.semanticdesktop.org/ontologies/2007/03/22/nfo/nfo_data.rdfs

Dokumentace: http://www.semanticdesktop.org/ontologies/2007/03/22/nfo/

6.1.7 Ontologie NID3

ID3 je společný standard pro hudební metadata.

Název: NEPOMUK ID3

Verze: revize 8 postavená na ontologii NIE (10.05.2007)

URL pro stažení ontologie:
http://www.semanticdesktop.org/ontologies/2007/05/10/nid3/nid3_data.rdfs

Dokumentace: http://www.semanticdesktop.org/ontologies/2007/05/10/nid3/

6.1.8 Ontologie NEXIF

NEXIF – EXIF je běžným standardem základního obrazu metadat používaných v digitálních fotoaparátech a softwarem pro správu snímků.

Název: NEPOMUK EXIF ontology

Verze: revize 8 postavená na ontologii NIE (10.05.2007)

URL pro stažení ontologie:
http://www.semanticdesktop.org/ontologies/2007/05/10/nexif/nexif_data.rdfs

Dokumentace: http://www.semanticdesktop.org/ontologies/2007/05/10/nexif/

6.2 Implementační omezení

V této kapitole se čtenář dozví možná omezení výsledného systému, jako je operační systém, GUI omezení, atd.

6.2.1 Java Platform (JRE)

Pro spuštění aplikace je nutností mít nainstalované:

- verzi JDK – 1.6.0_23
- verzi JRE – 1.6.0_23-b05

6.2.2 Operační systémy

Systém byl vyvíjen a testován pod operačními systémy Windows 7 Home Premium (64 bi-tový) a Ubuntu v11 (verze kernelu – 3.0.0-14-generic).

6.2.3 GUI omezení

Z důvodu neznalosti vývojového prostředí NetBeans a nedostatečných znalostí GUI aplikací se mohou vyskytnout v aplikaci následující problémy:

1. špatné zobrazení informačních oken
2. grafické rozhození prvků při zvětšování / zmenšování okna (byl kladen důraz na dynamičnost okna, ale může se stát, že prvky budou rozhozené)
3. počáteční nepřehlednost prvků (ačkoliv bylo dbáno i na samotnou orientaci a přehlednost, může se stát, že uživatel nebude zcela spokojen se vzhledem a jeho orientace bude díky tomu snížena)
4. autor nenese zodpovědnost za vzhled externích oken pro výběr souborů či adresářů (tento vzhled je zcela závislý na samotném operačním systému)
5. nepřehlednost aktuálního stavu nahrávání

6.2.4 Programové omezení

Z důvodu složité a časově náročné práce je ve výsledné aplikaci několik programových omezení:

1. při vkládání dat je možné využít pouze archivu ZIP (ostatní archivy nejsou podporovány)
2. při vkládání dat se ukládají předem dohodnutá metadata do předem zvolených ontologických vzorů (viz kapitoly jednotlivých ontologií: 6.1.6 na straně 42; 6.1.7 na straně 42; 6.1.8 na straně 43)
3. pokud uživatel bude ukládat stejný soubor (musí se shodovat s cestou ve struktuře aplikace a samotném názvu), tak se využije tzv. pštrosího algoritmu, nebo-li soubor se nahradí, ale jeho původní trojice zůstanou
4. export a import dat využívá pouze ZIP archivu
5. při odstranění či nahrazení ontologie se nikde neuchovává její záloha (pokud uživatel upravil původní ontologii, tak při odstranění či nahrazení přijde o všechny změněné položky)
6. z důvodu množství různých formátů souborů, dokáže aplikace zpracovávat pouze tyto formáty: JPG, JPEG, PNG, GIF, BMP, TIFF, TIF, MP3, MP4, M4A, M4P, OGG, WMA, WAV, PDF, AVI, MOV
7. při importu ontologie je doporučený formát *.rdfs (XML/RDF) (při využití jiného typu formátu nelze zaručit správný běh aplikace)
8. při využívání aplikace pod operačním systémem Linux je sám uživatel odpovědný za správné nastavení programů pro zobrazování/prezentování vložených dat (např. pro zobrazování obrázků se používá aplikace eog)
9. vyhledávání dat probíhá pouze u konkrétního typu ontologie (nelze vyhledávat data z více ontologií)

6.3 Struktura projektu a popis jednotlivých souborů

V této kapitole se čtenář dozví programové informace, např. adresářovou strukturu, popis jednotlivých tříd, atd.

6.3.1 Adresářová struktura spustitelného programu a její popis

- .store_direction/ - v tomto adresáři se ukládají soubory pomocí aplikace (při prvním spuštění aplikace se adresář automaticky vytvoří)
- .storeage/ - tento adresář slouží pro uložení RDF úložiště (při prvním spuštění aplikace se adresář automaticky vytvoří)
- lib/ - adresář, obsahující externí knihovny, které jsou nutné pro běh aplikacemi
- tmp/ - pomocný adresář, který se používá k rozbalování archivů či zabalování složek do archivu (při prvním spuštění aplikace se adresář automaticky vytvoří)
- DiplomovaPrace.jar – spustitelná aplikace
- errors.txt – soubor, do kterého se ukládá převážná část chybových stavů
- README.txt – automaticky vygenerovaný soubor s informacemi o možnostech spuštění aplikace

6.3.2 Popis tříd a adresářová struktura zdrojového programu

6.3.2.1 Adresářová struktura zdrojového programu

- insertItems/ - adresář s vláknem pro vkládání dat
    - InsertItems.java
- plugins/ - adresář, obsahující třídy pro práci se systémem
    - exportImport/ - adresář, obsahující třídy pro export/import dat
        - Export.java
        - Import.java
    - listItems/ - adresář, obsahující třídu pro prohlížení vložených dat
        - ListItems.java
    - metadataPlugins/ - adresář, obsahující třídy pro získávání metadat z jednotlivých typů souborů
        - MetadataAudios.java
        - MetadataCross.java
        - MetadataDocuments.java
        - MetadataPictures.java
        - MetadataVideos.java
    - ontologies/ - adresář, obsahující třídu pro práci s ontologiemi
        - Ontologies.java
    - searchItems/ - adresář, obsahující třídu pro vyhledávání dat dle určitých kritérií
        - SearchItems.java
    - storeage/ - adresář, obsahující třídu pro práci s RDF úložištěm
        - Storeage.java
    - Errors.java
    - Files.java
    - GlobalsProperties.java
    - InformationMessages.java
    - LinuxWindowsModulator.java
    - StructNameFilesInExport.java
    - StructOwlReader.java
    - StructTriples.java
    - Validation.java
    - Zip.java

- resources/ - adresář, obsahující obrázky k aplikaci a výchozí nastavení systému
- DiplomovaPraceAboutBox.form – nastavení formuláře, nebo-li nastavení elementů a jejich jednotlivých vlastností (např. velikost a barva písma, pozadí, atd.)
- DiplomovaPracceAboutBox.java – zdrojový kód formuláře
- DiplomovaPraceApp.java – hlavní část programu, která spouští a zobrazuje aplikaci
- DiplomovaPraceView.form - nastavení formuláře, nebo-li nastavení elementů a jejich jednotlivých vlastností (např. velikost a barva písma, pozadí, atd.)
- DiplomovaPraceView.java – zdrojový kód formuláře

6.3.2.2 Export.java

Jak už samotný název napovídá, tak tato třída slouží k exportu dat z RDF úložiště a z adresářové struktury do archivu ZIP.

Funkčnost této třídy můžete nalézt v UML diagramu v kapitole Příloha H: Diagram aktivit „export/import dat“.

Funkce třídy:

- **public Export** - konstruktor třídy
- **public void setMessageValues –** funkce pro nastavení výstupního dialogového okna
- **public void startExport** – spuštění vlákna pro export dat
- **private List<StructNameFilesInExport> exportCopyFilesToTmp** – funkce na kopírování dat do pomocného adresáře
- **public void run –** v této funkci běží export pomocí vlákna

6.3.2.3 Import.java

Jak už samotný název napovídá, tak tato třída slouží k importu dat z archivu ZIP do RDF úložiště a do adresářové struktury.

Funkčnost této třídy můžete nalézt v UML diagramu v kapitole Příloha H: Diagram aktivit „export/import dat“.

Funkce třídy:

- **public Import –** konstruktor třídy
- **public void setMessageValues –** funkce pro nastavení výstupního dialogového okna
- **public void startImport –** spuštění vlákna pro import dat
- **private List<StructNameFilesInExport> importCopyFilesFromTmp –** kopírování dat do pomocného adresáře
- **public void run() -** v této funkci běží import dat pomocí vlákna

6.3.2.4 ListItems.java

Tato třída slouží k zobrazování informací o vložených datech v RDF úložišti. Mezi její funkčnost patří např.:

- pří vybrání souboru se uloží všechny jeho predikáty do elementu combobox
- při vybrání souboru a jeho predikátu se uloží jeho vlastnost do textového boxu
- editace vlastností vybraného souboru
- odstranění
- atd.

Funkce třídy:

- **public ListItems –** konstruktor třídy
- **public void setMessage –** funkce pro nastavení výstupního dialogového okna
- **public void setPredicateToComboBoxByNameFile –** funkce nastaví do výběrového pole všechny predikáty dle názvu souboru
- **public void setLabelAndTextAreaBySubject –** pří vybrání souboru a predikátu se nastaví textová pole dle subjektu
- **public void setLabelAndTextFieldBySubject –** nastaví textová pole dle vybraného subjektu
- **public int getOntologieByNameFile –** získá číslo vložené ontologie dle vybraného souboru
- **public void updateProperty -** při stisku tlačítka tato funkce aktualizuje vybranou vlastnost
- **public void deleteProperty –** funkce odstraní u vybraného souboru jeho vlastnost
- **public void deleteItems –** hromadné odstranění souborů z adresáře a jejich trojic z RDF úložiště

6.3.2.5 MetadataCross.java

Tato třída slouží jako křižovatka při získávání metadat ze souborů. Protože získávání metadat je komplikované a při jeho získávání je důležité rozeznávání typů souborů.

Funkce třídy:

- **public MetadataCross –** konstruktor třídy
- **public void getMetadata –** rozcestník, který volá patřičné třídy pro práci metadat dle typu souboru
- **public String getImageHeight –** funkce, která získá výšku obrázku
- **public String getImageWidth –** funkce, která získá šířku obrázku
- **public String getImageAuthor –** funkce, která získá autora obrázku
- **public String getImageDescription –** funkce, která získá popis obrázku

- **public String getImageName –** funkce, která získá jméno obrázku
- **public String getMusicAlbum –** funkce, která získá název alba hudebního souboru
- **public String getMusicArtist –** funkce, která získá umělce hudebního souboru
- **public String getMusicComment –** funkce, která získá komentář hudebního souboru
- **public String getMusicDescription –** funkce, která získá popis hudebního souboru
- **public String getMusicGenre –** funkce, která získá žánr hudebního souboru
- **public String getMusicYear –** funkce, která získá rok vydání hudebního souboru
- **public String getDocCreator –** funkce, která získá jméno tvůrce vytvořeného dokumentu
- **public String getDocAuthor –** funkce, která získá jméno autora dokumentu
- **public String getDocCreationDate –** funkce, která získá datum vytvoření dokumentu
- **public String getDocProducer –** funkce, která získá jméno vydavatele dokumentu
- **public String getDocTitle –** funkce, která získá jméno dokumentu

6.3.2.6 MetadataAudios.java

Tato třída získává metadata z hudebních souborů. V následujícím přehledu si ukážeme, z jakých hudebních souborů a jakým způsobem se daná metadata získávají.

- **WAV –** způsob získávání metadat z tohoto formátu naleznete v textu níže (viz Kód 5 na straně 48)

```
File wavFile = new File(this.filename);
AudioFile f = AudioFileIO.read(wavFile);
WavTag wavtag = (WavTag)f.getTag();

/* ziskani metadat */
this.setMusicAlbum(wavtag.getFirst(FieldKey.ALBUM));
this.setMusicArtist(wavtag.getFirst(FieldKey.ARTIST));
this.setMusicComment(wavtag.getFirst(FieldKey.COMMENT));
this.setMusicDescription(wavtag.getFirst(FieldKey.TITLE));
this.setMusicGenre(wavtag.getFirst(FieldKey.GENRE));
this.setMusicYear(wavtag.getFirst(FieldKey.YEAR));
/* konec */
```

Kód 5: Získávání metadat z formátu WAV

- **WMA –** způsob získávání metadat z tohoto formátu naleznete v textu níže (viz Kód 6 na straně 48)

```
File wmaFile = new File(this.filename);
AudioFile f = AudioFileIO.read(wmaFile);
Tag wmatag = f.getTag();

/* ziskani metadat */
this.setMusicAlbum(wmatag.getFirst(FieldKey.ALBUM));
this.setMusicArtist(wmatag.getFirst(FieldKey.ARTIST));
this.setMusicComment(wmatag.getFirst(FieldKey.COMMENT));
this.setMusicDescription(wmatag.getFirst(FieldKey.TITLE));
this.setMusicGenre(wmatag.getFirst(FieldKey.GENRE));
this.setMusicYear(wmatag
```

Kód 6: Získávání metadat z formátu WMA

- **MP4, M4A, M4P –** způsob získávání metadat z těchto formátů naleznete v textu níže (viz Kód 7 na straně 49)

```
File mp4File = new File(this.filename);
AudioFile f = AudioFileIO.read(mp4File);
Mp4Tag mp4tag = (Mp4Tag)f.getTag();

/* ziskani metadat */
this.setMusicAlbum(mp4tag.getFirst(Mp4FieldKey.ALBUM));
this.setMusicArtist(mp4tag.getFirst(Mp4FieldKey.ARTIST));
this.setMusicComment(mp4tag.getFirst(Mp4FieldKey.COMMENT));
this.setMusicDescription(mp4tag.getFirst(Mp4FieldKey.DESCRIPTION));
this.setMusicGenre(mp4tag.getFirst(Mp4FieldKey.GENRE));
this.setMusicYear(mp4tag.getFirst(Mp4FieldKey.MM_ORIGINAL_YEAR));
```

Kód 7: Získávání metadat z formátů MP4, M4A, M4P

- **OGG –** způsob získávání metadat z tohoto formátu naleznete v textu níže (viz Kód 8 na straně 49)

```
File oggFile = new File(this.filename);
AudioFile f = AudioFileIO.read(oggFile);
VorbisCommentTag oggTag = (VorbisCommentTag)f.getTag();

/* ziskani metadat */
this.setMusicAlbum(oggTag.getFirst(VorbisCommentFieldKey.ALBUM));
this.setMusicArtist(oggTag.getFirst(VorbisCommentFieldKey.ARTIST));
this.setMusicComment(oggTag.getFirst(VorbisCommentFieldKey.COMMENT));
this.setMusicDescription(oggTag.getFirst(VorbisCommentFieldKey.
     DESCRIPTION));
this.setMusicGenre(oggTag.getFirst(VorbisCommentFieldKey.GENRE));
this.setMusicYear(oggTag.getFirst(VorbisCommentFieldKey.
     ORIGINAL_YEAR));
```

Kód 8: Získávání metadat z formátu OGG

- **MP3 –** způsob získávání metadat z tohoto formátu naleznete v textu níže (viz Kód 9 na straně 49)

```
MP3File mp3 = (MP3File) AudioFileIO.read(mp3File);
Tag tag = mp3.getTag();
ID3v1Tag v1Tag = mp3.getID3v1Tag();
ID3v24Tag v24Tag = mp3.getID3v2TagAsv24();

// Jestlize se jedna o MP3 v2
if(mp3.hasID3v2Tag()) {
     ….
} else {
     ….
}
```

Kód 9: Získávání metadat z formátu MP3

Funkce třídy:

- **public MetadataAudios –** konstruktor třídy
- **public void musicWAV –** funkce pro získávání metadat z hudebního formátu WAV

- **public void musicWMA –** funkce pro získávání metadat z hudebního formátu WMA
- **public void musicMP4 –** funkce pro získávání metadat z hudebního formátu MP4
- **public void musicOGG –** funkce pro získávání metadat z hudebního formátu OGG
- **public void musicMP3 –** funkce pro získávání metadat z hudebního formátu MP3
- **private void setMusicAlbum –** nastaví do privátní proměnné jméno alba
- **private void setMusicDescription -** nastaví do privátní proměnné popis
- **private void setMusicArtist -** nastaví do privátní proměnné umělce
- **private void setMusicYear -** nastaví do privátní proměnné rok vydání
- **private void setMusicGenre -** nastaví do privátní proměnné žánr
- **private void setMusicComment -** nastaví do privátní proměnné komentář
- **public String getMusicAlbum –** získá z privátní proměnné jméno alba
- **public String getMusicArtist –** získá z privátní proměnné umělce
- **public String getMusicDescription –** získá z privátní proměnné popis
- **public String getMusicYear –** získá z privátní proměnné rok vydání
- **public String getMusicGenre –** získá z privátní proměnné žánr
- **public String getMusicComment –** získá z privátní proměnné komentář

6.3.2.7 MetadataDocuments.java

Tato třída získává metadata z dokumentů. V následujícím přehledu si ukážeme, z jakých typů souborů a jakým způsobem se daná metadata získávají.

- **PDF -** způsob získávání metadat z tohoto formátu naleznete v textu níže (viz Kód 10 na straně 51)

```
PdfReader reader = new PdfReader(this.filename);
Map info = reader.getInfo();
String key = "";

Terminals activeTerminal;

for (Iterator i = info.keySet().iterator(); i.hasNext();) {
     key = (String) i.next();
            activeTerminal = Terminals.valueOf(key.toUpperCase());

            String value = (String) info.get(key);
            System.out.println(key + ": " + value);

            switch(activeTerminal) {
                 case CREATOR:
                              this.creator = (String) info.get(key);
                               break;
                 case AUTHOR:
                              this.author = (String) info.get(key);
                               break;
                 case CREATIONDATE:
                              this.creationDate = (String) info.
                                    get(key);
                              break;
                       case PRODUCER:
                              this.producer = (String) info.get(key);
                              break;
                       case TITLE:
                              this.title = (String) info.get(key);
                              break;
     }
}
```

Kód 10: Získávání metadat z formátu PDF

Funkce třídy:

- **public MetadataDocuments –** konstruktor třídy
- **public void documentPDF –** funkce pro získání metadat z dokumentu PDF
- **public String getCreator -** získá z privátní proměnné jméno tvůrce
- **public String getAuthor -** získá z privátní proměnné jméno autora
- **public String getCreationDate -** získá z privátní proměnné datum vytvoření
- **public String getProducer -** získá z privátní proměnné vydavatele
- **public String getTitle -** získá z privátní proměnné název díla

6.3.2.8 MetadataPictures.java

Tato třída získává metadata z obrázkových dokumentů. V následujícím přehledu si ukážeme, z jakých typů souborů a jakým způsobem se daná metadata získávají.

- **JPG, JPEG, TIFF, TIF -** způsob získávání metadat z těchto formátů naleznete v textu níže (viz Kód 11 na straně 52)

```
Metadata metadata = ImageMetadataReader.readMetadata(new
File(this.filename));
// projedu vsechny tagy a podle hexa cisla ukladam metadata
for (Directory directory : metadata.getDirectories()) {
     for (Tag tag : directory.getTags()) {
           if("0x0001".equals(tag.getTagTypeHex()))
                          this.setImageHeight(tag.getDescription());
                     else if("0x0003".equals(tag.getTagTypeHex()))
                          this.setImageWidth(tag.getDescription());
                     else if("0x010e".equals(tag.getTagTypeHex()))

     this.setImageDescription(tag.getDescription());
                     else if("0x9c9d".equals(tag.getTagTypeHex()))
                          this.setImageAuthor(tag.getDescription());
                     else if("0x9c9b".equals(tag.getTagTypeHex()))
                          this.setImageName(tag.getDescription());
     }
}
```

Kód 11: Získávání metadat z formátů JPG, JPEG, TIFF, TIF

- **BMP, GIF, PNG -** způsob získávání metadat z těchto formátů naleznete v textu níže (viz Kód 12 na straně 52)

```
ImageInputStream iis = ImageIO.createImageInputStream(file);
Iterator<ImageReader> readers = ImageIO.getImageReaders(iis);
if (readers.hasNext()) {

     // pick the first available ImageReader
     ImageReader reader = readers.next();

     // attach source to the reader
     reader.setInput(iis, true);

     this.setImageWidth(Integer.toString(reader.getWidth(0)));
     this.setImageWidth(Integer.toString(reader.getHeight(0)));
}
```

Kód 12: Získávání metadat z formátů BMP, GIF, PNG

Funkce třídy:

- **public MetadataPictures –** konstruktor třídy
- **public void pictureJPG –** získání metadat z obrázku JPG, JPEG
- **public void pictureBMP –** získání metadat z obrázku BMP
- **public void pictureGIF –** získání metadat z obrázku GIF
- **public void pictureTIFF –** získání metadat z obrázku TIFF, TIF
- **public void picturePNG –** získání metadat z obrázku JPG, JPEG
- **private void getMetadataFromPngGif –** rekurzivní funkce, která hledá metadata ve stromu
- **private void setImageHeight –** nastaví do privátní proměnné výšku obrázku
- **private void setImageWidth –** nastaví do privátní proměnné šířku obrázku

- **private void setImageDescription –** nastaví do privátní proměnné popis
- **private void setImageAuthor –** nastaví do privátní proměnné jméno autora
- **private void setImageName –** nastaví do privátní proměnné jméno obrázku
- **public String getImageHeight –** získá z privátní proměnné výšku obrázku
- **public String getImageWidth –** získá z privátní proměnné šířku obrázku
- **public String getImageDescription –** získá z privátní proměnné popis obrázku
- **public String getImageAuthor –** získá z privátní proměnné jméno autora
- **public String getImageName –** získá z privátní proměnné jméno obrázku

6.3.2.9 MetadataVideos.java

Bohužel, tato třída není funkční a získávání metadat z video souborů nelze.

6.3.2.10 Ontologies.java

Třída slouží pro práci s ontologiemi. Mezi její funkčnost patří:

- vkládání ontologií
- odstraňování ontologií
- editace jednotlivých vlastností vybrané ontologie
- vkládání nových vlastností vybrané ontologie
- vložení vlastností dané ontologie do elementu combobox
- atd.

Funkce třídy:

- **public Ontologies –** konstruktor třídy
- **public void setComboBoxTypesOntologies –** nastaví výběrové pole typu ontologie (obrázky, hudba, dokumenty)
- **public void setMessage –** funkce pro nastavení výstupního dialogového okna
- **public void setTreeItemsByIdOntology –** při vybrání typu ontologie se všechny jeho vlastnosti vypíší do stromu
- **public void deleteTriplesByValueInSelectedRow –** odstraní všechny trojice z ontologie dle vybraného subjektu
- **public void getValuesAndShowItToTextBoxes –** po vybrání ontologie a její vlastnosti se vypíší její detaily do textových boxů
- **public void insertNewTriple –** vloží novou vlastnost k vybrané ontologii
- **public void updateSelectedSubject –** aktualizuje vybranou vlastnost
- **public void insertNewOntology –** vloží novou ontologii

6.3.2.11 SearchItems.java

Tato třída slouží k vyhledávání vložených dat dle nastavených kritérií (princip vyhledávání naleznete v kapitole Příloha K: Diagram aktivit „Vyhledávání dat“).

Hlavní funkčnost třídy:

- vytvoření podmínky pro dotazovací jazyk
- nalezení hodnot odpovídajících podmínce

Funkce třídy:

- **public SearchItems –** konstruktor třídy
- **public void setMessage –** funkce pro nastavení výstupního dialogového okna
- **public List<StructTriples> search –** nalezení všech trojic odpovídajících vyhledávací podmínce

6.3.2.12 Storeage.java

Mohli bychom říci, že tato třída je nejdůležitější částí systému, protože nám slouží pro práci s RDF úložištěm.

Hlavní funkčnost třídy:

- získání trojic z RDF úložiště dle různých podmínek
- získání subjektu, predikátu nebo objektu z RDF úložiště dle různých podmínek
- odstranění trojic dle podmínky
- export trojic do souboru
- vložení nových trojic do RDF úložiště
- import RDF dat
- atd.

Funkce třídy:

- **public Storeage –** kontruktor třídy
- **public void setStoreage –** nastaví RDF úložiště
- **public void getAllTriples –** testovací funkce, která získá všechny trojice z RDF úložiště
- **public void saveTriplesToRDF –** export trojic do souboru
- **public List<StructTriples> getTriplesByList** – funkce získá trojice dle určité podmínky a uloží je do struktury
- **public String getObjectByUrlAndPredicatePrefixAndLang –** získá objekt dle URL, prefixu predikátu a jazyka
- **public String getObjectByUrlAndPredicatePrefix –** získá objekt dle URL a prefixu predikátu

- **public String getObjectByUrlAndPredicate –** získá objekt dle URL a celého predikátu
- **public boolean existSubject –** testuje, zda existuje subjekt
- **public void deleteTriplesBySubject –** odstraní trojice dle subjektu
- **public void deleteTriplesBySubjectAndPredicate –** odstraní trojice dle subjektu a predikátu
- **public String getPredicateeByPrefixAndUrlAndLang –** získá predikát dle svého prefixu, URL a jazyka
- **public String getPredicateByPrefixAndPrefixUrlAndLang –** získá predikát dle svého prefixu, prefixu URL a jazyka
- **public boolean insertTriples –** hromadné vložení trojic do RDF úložiště
- **public String getSubjectByPredicateAndObject –** získá subjekt dle svého predikátu a objektu
- **public int getCountOntologies –** funkce zjistí počet ontologii
- **public void deleteOntology –** odstraní ontologii
- **public List<StructTriples> getTriplesBySubject –** získá všechny trojice z RDF úložiště dle subjektu
- **public List<StructTriples> getTriplesByFilter –** získá všechny trojice z RDF úložiště dle vstupní podmínky
- **public List<StructTriples> getAllSubjectsBySubstring –** nalezení všech subjektů dle svého podřetězce
- **public void insertOntology –** vloží ontologii
- **public void setImport –** import dat do RDF úložiště
- **public void replaceFilePath –** nahradí staré nekonzistentní cesty k souborům za nové
- **public void saveToStoreage()** - rozcestník, který nám slouží k uložení metadat ze souboru do RDF úložiště
- **private void saveToStoreagePictures –** uloží metadata z obrázku
- **private void saveToStoreageAudios –** uloží metadata z hudebního souboru
- **private void saveToStoreageDocuments –** uloží metadata z dokumentu

6.3.2.13 Errors.java

Třída slouží k zápisu chyb do textového souboru. Pokud se podíváte na adresářovou strukturu v kapitole 6.3.1 na straně 44, tak uvidíte soubor s názvem errors.txt, který slouží k uchovávání těchto chybových stavů.

Funkce třídy:

- **public Errors –** konstruktor třídy
- **public void setErrorToFile –** zápis chyby do výstupního souboru

6.3.2.14 Files.java

Třída slouží pro práci se soubory a adresáři.

Funkčnost třídy:

- kopírování souborů
- vytváření adresářů
- kontrola, zda se jedná o soubor či adresář
- získávání adresářů, které se následně vypíší do výběrového pole
- řazení adresářové struktury (abecedně, zda se jedná o adresář či soubor, atd.)
- při kliknutí na rozbalení stromu se získává aktivní podstrom
- odstraňování souborů či adresářů
- atd.

Funkce třídy:

- **public Files –** konstruktor třídy
- **public int existDirectory –** testování existence adresáře
- **public int existFile –** testování existence souboru
- **public void run –** hlavní metoda vlákna
- **public void existErrorFile –** kontroluje, zda existuje chybový soubor, pokud ne, tak ho vytvoří
- **public void existStoreDirection –** kontroluje existenci adresáře, v případě že neexistuje, tak ho vytvoří
- **public void setAllDirectoriesToComboBox –** funkce vypíše všechny adresáře do výběrového pole
- **public void setAllDirectoriesToTree –** nahraje adresářovou strukturu do stromu
- **public void setExpandTree –** při rozbalení větve se načte podstrom
- **private DefaultMutableTreeNode getActiveSubTree –** zjistí, zda existuje podstrom či ne
- **private String[] orderDirectionsAndFiles –** řazení souborů a adresářů dle podmínek
- **public int insertNewItems –** funkce slouží při hromadném vkládání nových dat
- **private int copyDirectoriesAndFiles –** slouží ke kopírování adresářů a souborů do předem vytvořené adresářové struktury
- **public int unrarZip –** rozbalení archivu
- **public String setNewDirectories** - tato funkce slouží k vytvoření nové struktury
- **public int createDirectory –** vytvoření adresáře
- **private String deleteSeparatorsFromDir –** odstranění nepovolených znaků při vytváření adresáře

- **public void deleteDirectoriesAndFiles –** odstraní adresář i jeho podvětve
- **public void deleteFile –** odstraní soubor
- **public List<StructNameFilesInExport> copyDirWithNewNames –** kopírování dat mezi adresáři a přejmenování souborů dle hash funkce
- **public String getHash –** vytvoření nového jména souboru dle hash funkce

6.3.2.15 GlobalsProperties.java

Třída slouží k nastavení globálních proměnných (viz Kód 13 na straně 57), které se velice často využívají ve zdrojových kódech aplikace.

```
public GlobalsProperties() {
        this.typeOfOntologyModel = "http://localhost/ontologyModel#type";
        this.typeOfRdfSyntax = "http://localhost/rdf-syntax#type";
        this.urlPrefix = "http://localhost/";
        this.path = "file#path";
}
```

Kód 13: Globální proměnné

typeOfOntologyModel – URI adresa, která určuje typ ontologie

typeOfRdfSyntax – URI adresa, která určuje RDF syntax

urlPrefix - URL prefix (počáteční část adresy)

path – název predikátu pro určení cesty k souboru

Funkce třídy:

- **public GlobalsProperties** – konstruktor třídy
- **public String getTypeOfOntologyModel –** získá URL pro označení ontologie
- **public String getUrlFilePath –** získá URL pro označení vlastnosti cesty k souboru
- **public String getTypeOfRdfSyntax –** získá URL pro práci s vlastností
- **public String getUrlPrefix –** získá URL prefix pro práci v RDF úložišti

6.3.2.16 InformationMessages.java

Tato třída slouží pouze k vypsání informační hlášky pomocí nově vytvořeného okna.

Funkce třídy:

- **public void printMessageToDialog –** zobrazí informační hlášku

6.3.2.17 LinuxWindowsModulator.java

Třída, sloužící k rozeznávání operačního systému, ke změně lomítek v cestě k souboru a k vytvoření příkazu pro spuštění souboru pomocí příkazové řádky (viz Kód 14 na straně 57).

```
if(this.isWindows())
     path = "rundll32 url.dll,FileProtocolHandler " + path.replace("/",
     this.backslashes);
else
     path = this.getRunApplicationForLinuxInCommand(path) + " " +
            path.replace("\\", this.backslashes);
```

Kód 14: Vytvoření příkazu pro příkazovou řádku ve Windows

Funkce třídy:

- **LinuxWindowsModulator –** konstruktor třídy
- **public void setResourceMap –** nastaví zdrojovou mapu
- **public boolean isWindows –** testuje, zda se jedná o operační systém Windows
- **public boolean isLinux –** testuje, zda se jedná o operační systém Linux
- **public String createRunCommand –** vytvoří příkaz pro prohlížení souborů z příkazové řádky
- **private String getRunApplicationForLinuxInCommand –** pokud se jedná o operační systém Linux, je důležité vytvořit nový konkrétně spustitelný příkaz pro každý typ souboru

6.3.2.18 StructNameFilesInExport

Třída slouží k uchovávání nových a starých cest k souborům při exportu dat. Struktura se využívá z důvodu, že při exportu dat je původní cesta uložena do RDF úložiště a tudíž není konzistentní. Proto je důležité upravit všechny exportované cesty k souborům na nové.

Vzhled proměnných naleznete v textu níže (viz Kód 15 na straně 58).

```
private String pathOld; // Stary nazev
private String pathNew; // Novy nazev
```

Kód 15: Proměnné použité ve struktuře

Funkce třídy:

- **public StructNameFilesInExport –** konstruktor třídy
- **public String getPathOld –** získá z privátní proměnné starou cestu k souboru
- **public String getPathNew –** získá z privátní proměnné novou cestu k souboru

6.3.2.19 StructOwlReader.java

Třída slouží k uchovávání detailních informací o objektu v předem definované struktuře.

Vzhled proměnných naleznete v textu níže (viz Kód 16 na straně 58).

```
private String typeOwl; // typ OWL ... jako NEXIF, NID3 atd...
private String typeMetadata; // typ metadata ... jako imageWidth atd...
private String label; // originální popis metadata
private String cs_label; // popis metadat v češtině
```

Kód 16: Proměnné použité ve struktuře

Funkce třídy:

- **public StructOwlReader –** konstruktor třídy
- **public String getTypeOwl –** získá z privátní proměnné typ ontologie
- **public String getTypeMetadata –** získá z privátní proměnné typ metadat
- **public String getLabel –** získá z privátní proměnné popis metadat
- **public String getCSLabel –** získá z privátní proměnné český popis metadat

6.3.2.20 StructTriples.java

Třída slouží k uchovávání trojic z RDF úložiště v předem definované struktuře. Struktura se využívá pří výběrových a akčních dotazech u RDF úložiště.

Vzhled proměnných naleznete v textu níže (viz Kód 17 na straně 59).

```
private String s; // subject
private String o; // object
private String p; // predicate
private String lang; // jazyk
```

Kód 17: Proměnné použité ve struktuře

Funkce třídy:

- **public StructTriples –** konstruktor třídy
- **public String getS –** získá subjekt z privátní proměnné
- **public void setS –** nastaví do privátní proměnné subjekt
- **public String getP –** získá predikát z privátní proměnné
- **public void setP –** nastaví do privátní proměnné predikát
- **public String getO –** získá objekt z privátní proměnné
- **public void setO –** nastaví do privátní proměnné objekt
- **public String getLang –** získá jazyk z privátní proměnné

6.3.2.21 Validation.java

Tato třída nám slouží pouze k validaci hodnot. Při vývoji systému bylo nutností vytvořit pouze jednu funkci, která kontroluje validitu URI adresy.

```
public boolean isUrl(String url) {
      String urlPattern = "^http(s{0,1})://[a-zA-Z0-9_/\\-\\.]+\\.([A-Za-
            z/]{2,5})[a-zA-Z0-9_/\\&\\#\\?\\=\\-\\.\\~\\%]*";
      return url.matches(urlPattern);
}
```

Kód 18: Kontrola URL

Funkce třídy:

- **public Validation –** konstruktor třídy
- **public boolean isUrl –** funkce pro testování, zda se jedná o URL (viz Kód 18 na straně 59)

6.3.2.22 Zip.java

Jak už samotný název napovídá, tak tato třída slouží pro práci s archivem.

Hlavní funkčnost třídy:

- extrakce souboru – zdrojový kód naleznete níže v textu (viz Kód 19 na straně 60)

```
zipFile = new ZipFile(source);
entriesEnum = zipFile.entries();
File directory = new File(destination);

while (entriesEnum.hasMoreElements()) {
     try {
           ZipEntry entry = (ZipEntry) entriesEnum.nextElement();
           if (entry.isDirectory()) {
                 ret = c_file.createDirectory(destination +
entry.getName());
           } else {
                 int index = 0;
                 String name = entry.getName();
                 index = entry.getName().lastIndexOf("/");
                 if (index > 0 && index != name.length())
                       name = entry.getName().substring(index + 1);

                       writeFile(zipFile.getInputStream(entry),
                 new BufferedOutputStream(new
                       FileOutputStream(destination +
entry.getName())));
           }
     } catch (Exception e) {}
}
zipFile.close();
```

Kód 19: Extrakce souboru

- archivace souboru – zdrojový kód naleznete v textu níže (viz Kód 20 na straně 60)

```
FileInputStream in = null;
try {

     in = new FileInputStream(files[i]);
     ZipEntry entry = new ZipEntry(files[i].getPath().substring(
           base.getPath().length() + 1));
     zos.putNextEntry(entry);
     while (-1 != (read = in.read(buffer))) {
           zos.write(buffer, 0, read);
     }
     in.close();

} catch (IOException ex) {
     Logger.getLogger(Zip.class.getName()).log(Level.SEVERE, null,
ex);
} finally {
     try {
           in.close();
     } catch (IOException ex) {
           Logger.getLogger(Zip.class.getName()).
                 log(Level.SEVERE, null, ex);
     }
}
```

Kód 20: Archivace souboru

Funkce třídy:

- **public Zip –** konstruktor třídy

- **public int unzip –** pomocná funkce, která vrací stav rozbalení
- **private void doUnzip –** funkce pro rozbalení archivu
- **public void doZip –** funkce pro vytvoření archivu
- **private static void writeFile –** funkce se používá pro zápis z bufferu (využívá se při rozbalování archivu)

6.4 Testování

Tato kapitola seznamuje čtenáře s dosaženými rychlostmi vytvořené aplikace při různých vstupních datech.

Veškeré testované soubory naleznete na přiloženém CD.

6.4.1 Vkládání dat

Tabulka 7 na straně 28 vznikla při tvorbě oborového projektu a ukazuje rychlosti čtení metadat s ukládáním i bez ukládání dat do RDF úložiště.

Význam jednotlivých sloupečků:

- Formát dat – označuje typy souborů (v našem případě obrázkový formát JPG a hudební formát MP3), které se nachází v testovacím archivu ZIP
- Velikost [MB] – jedná se o popis velikosti archivu v MB
- Počet položek – počet položek uložených v archivu
- Typ běhu – označení o jaký typ běhu se jedná (popis jednotlivých typů naleznete pod tabulkou)
- Celková doba běhu [hh:mm:ss] – určuje čas vykonání úkonu v hodinách:minutách:sekundách

Formát dat	Velikost [MB]	Počet položek	Typ běhu	Celková doba běhu [hh:mm:ss]
JPG	126.45	2069	1	00:01:08
JPG	126.45	2069	2	01:02:04
MP3	1024,95	193	1	00:00:30
MP3	1024,95	193	2	00:01:07

Tabulka 7: Doba běhu vkládání dat (u oborového projektu)

1 – program běžel bez ukládání do RDF úložiště

2 – program běžel s ukládáním do RDF úložiště

6.4.2 Vkládání dat

Tabulka 8 na straně 62 ukazuje rychlost čtení metadat s ukládáním do RDF úložiště.

Význam jednotlivých sloupečků:

- Formát dat – označuje typy souborů (v našem případě obrázkový formát JPG a hudební formát MP3), které se nachází v testovacím archivu ZIP

- Velikost [MB] – jedná se o popis velikosti archivu v MB
- Počet položek – počet položek uložených v archivu
- Celková doba běhu [hh:mm:ss.set] – určuje čas vykonání úkonu v hodinách:minutách:sekundách.setinách

Formát dat	Velikost [MB]	Počet položek	Celková doba běhu [hh:mm:ss.set]
JPG (1JPG.zip)	3,15	48	00:00:11.2
JPG (2JPG.zip)	6,33	140	00:00:27.2
JPG (3JPG.zip)	10,5	243	00:00:49.8
JPG (4JPG.zip)	28,9	540	00:01:42.3
JPG (5JPG.zip)	53,7	1035	00:03:10.3
MP3 (1MP3.mp3)	73,4	1	00:00:01.4
MP3 (2MP3.zip)	186	21	00:00:13.5

Tabulka 8: Doba běhu vkládání dat

6.4.3 Vkládání ontologií

Tabulka 9 na straně 62 ukazuje rychlost ukládání různých ontologií do RDF úložiště.

Význam jednotlivých sloupečků:

- Formát dat (typ ontologie) – informace o vkládané ontologii a jejím formátu
- Velikost [kB] – jedná se o popis velikosti archivu v kB
- Počet položek – počet nahrávaných ontologií
- Celková doba běhu [hh:mm:ss.set] – určuje čas vykonání úkonu v hodinách:minutách:sekundách.setinách

Formát dat (typ ontologie)	Velikost [kB]	Počet položek	Celková doba běhu [hh:mm:ss.set]
RDFS (NEXIF)	113	1	00:00:01.8
RDFS (NFO)	49,4	1	00:00:00.9
RDFS (NID3)	50,1	1	00:00:01.8

Tabulka 9: Doba běhu vkládání ontologií

6.4.4 Vyhledávání dat

Tabulka 10 na straně 63 ukazuje rychlost vyhledávání dat dle vybrané ontologie a specifického dotazu.

Význam jednotlivých sloupečků:

- Ontologie – v jakém typu ontologie se vyhledává
- Specifický dotaz – jedná se o specifičtější parametry vyhledávání
- Počet uložených dat v RDF úložišti / nalezených – ukazuje počet uložených dat v RDF úložišti a počet všech nalezených dat dle zadané podmínky a ontologie

- Celková doba běhu [hh:mm:ss.set] – určuje čas vykonání úkonu v hodinách:minutách:sekundách.setinách

Ontologie	Specifický dotaz	Počet uložených dat v RDF úložišti / nalezených	Celková doba běhu [hh:mm:ss.set]
NEXIF	- bez dotazu -	2028/2006	00:00:00.6
NEXIF	- width = 600 -	2028/286	00:00:00.4
NID3	- bez dotazu -	2006//22	00:00:00.3

Tabulka 10: Doba běhu u vyhledávání dat

6.5 Zhodnocení programu

Správce sbírek založený na technologii sémantického webu je GUI aplikace, u které bylo dbáno na vytvoření příjemného a intuitivního uživatelského prostředí. Z tohoto důvodu byl k diplomové práci přibrán externí grafik, který navrhl základní vzhled programu (viz Ilustrace 11 na straně 75).

Při implementaci grafiky nebylo docíleno totožného vzhledu, protože některé prvky nebylo možné ovlivnit (např. horní menu či rámeček okna aplikace). Program byl testován u reálných uživatelů. Při testování vyplynuly některé grafické nedostatky:

- vzhled informačních hlášek
- nezobrazování aktuálního stavu při nahrávání dat (progress bar)
- drobné estetické vady u vzhledu podstránky „vyhledávání“ a s „výsledkem vyhledávání“

I přes tyto nedostatky mohu říci, že grafické prostředí programu je pro uživatele příjemné a intuitivní.

Program dokáže zpracovávat metadata různých datových formátů jako je např.: JPG, JPEG, MP3, MP4, atd. (viz kapitola 4.2.2.3 na straně 31). Z počátku bylo dohodnuto zpracování video souborů jako je MOV a AVI, ale kvůli nedostatku času a rozšíření aplikace o dynamickou manipulaci s ontologiemi bylo rozhodnuto, že požadavek je méně prioritní a není nutné ho vytvořit v rámci diplomové práce. Stejný problém vznikl i u souborů, u kterých bylo domluveno, že se budou zpracovávat alespoň soubory ve formátu PDF.

Pří nahrávání nových souborů do aplikace se ukládají (překopírovávají) data do adresářové struktury a metadata do RDF úložiště. Při prohlížení uložených souborů se jejich názvy zobrazují v hash tvaru (např. 04Lk4339slk4001doiq93045450ddj021os.jpg), který není příjemný pro uživatele při práci s výsledným nástrojem. Veškeré zobrazování/prezentování souborů probíhá pomocí externích programů (neexistuje možnost zobrazovat/prezentovat soubory uvnitř aplikace).

Nástroj pro správu sbírek byl rozšířen o nadstandardní požadavek „Import ontologií“. Uživatel má možnost vkládat a editovat ontologie 3 různých kategorií (rozdělení kategorií a seznam jednotlivých typů souborů naleznete popsané v kapitole 4.2.2.3 na straně 31). Nevýhodou této části systému je, že neexistuje export upravené ontologie z RDF úložiště (pokud tedy uživatel upraví ontologii, přidáním českých popisků či nových vlastností, tak nelze ontologii získat z RDF úložiště a dále ji využívat mimo program).

V systému existuje stránka *„Export/Import dat”*, která umožňuje uživateli přesouvat data z jednoho počítače na druhý aniž by musel znovu zjišťovat či upravovat metadata každého

souboru. Import dat je možný pouze se ZIP archivem, který byl vytvořen touto aplikací (importovaný ZIP archiv musí mít přesný název RDF souboru a správnou adresářovou strukturu, viz kapitola 4.2.3.1 na straně 33).

Nástroj pro správu sbírek také umožňuje vyhledávat data dle typu kategorie (viz kapitola 4.2.2.3 na straně 31) a její vlastností. Uživatel může tedy vyhledat soubory pomocí více zadaných hodnot, například u obrázků si zvolí šířku, výšku, název, atd. a podle všech těchto hodnot se soubory vyhledají. Po nalezení dat se otevře „nová podstránka s výsledkem vyhledávání“, ve které lze vyhledaná data exportovat do ZIP archivu a zobrazovat/prezentovat je pomocí externích programů. Hlavním nedostatkem stránky „vyhledávání“ je, že se „nezvýrazní“ vlastnosti (nebo-li řádky), které jsou použity v podmínce pro vyhledání dat.

Při vytvoření oborového projektu byly naměřeny časové hodnoty při získávání metadat ze souborů a jejich následné ukládání do RDF úložiště (viz Tabulka 7 na straně 61). Po naměření těchto hodnot bylo nutno v rámci diplomové práce změnit logiku vkládání dat tak, aby se čas běhu programu razantně snížil. Proto bylo důležité upravit celý kód a odstranit zbytečné cykly, podmínkové proměnné a vylepšit práci s vlákny. Úpravami bylo docíleno zrychlení systému, cca 6x až 7x (viz Tabulka 8 na straně 62).

Výsledná diplomová práce umožňuje ukládání souborů do adresářové struktury, čtení a ukládání metadat ze souborů do RDF úložiště a jejich následnou editaci. Nástroj dále umožňuje export i import dat, vkládání ontologií a jejich editaci, vyhledávání dat dle parametrů a zobrazování uložených dat pomocí externích programů. Jsem přesvědčen, že všechny zadané požadavky v rámci diplomové práce jsem splnil.

6.5.1 Další rozšíření

Během vývoje a testování aplikace vzniklo několik nápadů na rozšíření:

1) rozšířit čtení metadat pro více typů souborů (např. DOC, ODT, XLS, CVS, TXT, RFF, MOV, AVI, atd.)
2) možnost práce s různými archivy (GZIP, RAR, atd.)
3) práce se soubory dle data jejich vložení
4) možnost práce více uživatelů pod různými přístupovými údaji
5) historie jednotlivých uživatelů
6) možnost zobrazovat/prezentovat data pomocí aplikace (nevyužívat externí programy)
7) možnost exportu ontologie
8) kontrola validity souborů a trojic v RDF úložišti (pokud uživatel nechtěně odstraní soubor, tak trojice jsou v systému už zbytečné)
9) zlepšení stránky s vyhledáváním
10) zobrazovat skutečné názvy souborů získané z metadat (momentálně se názvy souborů zobrazují v hash tvaru)
11) možnost „třídění“ výsledku vyhledávání dle metadat (budu chtít soubory rozdělit například pomocí šířky obrázku atd.)
12) při vyhledávání dat pomocí vlastností aktivovat řádky, které uživatel vyplnil
13) vytvořit webovou aplikaci, která by fungovala na principu diplomové práce (možnost prezentovat určité soubory na Internetu)

14) rozšířit export dat o zvolení výsledného RDF souboru (N-Triples, Turtles atd...)
15) import dat rozšířit o dynamické zpracování různých typů RDF souborů
16) přizpůsobit aplikaci pro dotykový display
17) změnit tlačítka na ikony (například tlačítko s nápisem *úvodní stránka* změnit za ikonu domečku, atd.)

7 Závěr

Výsledný nástroj pro správu sbírek, který jsem zpracoval v rámci diplomové práce, umožňuje ukládání souborů do adresářové struktury, čtení a ukládání metadat ze souborů do RDF úložiště a jejich následnou editaci. Nástroj dále umožňuje export i import dat, vkládání ontologií a jejich editaci, vyhledávání dat dle parametrů a zobrazování uložených dat pomocí externích programů.

Během vývoje diplomové práce jsem se seznámil s technologiemi sémantického webu, jako je například Resource Description Framework (RDF), Web Ontology Language (OWL) či sémantický desktop. Dále jsem prostudoval některé používané nástroje pro práci s technologií sémantického webu, jako jsou například RDF úložiště Jena či Sesame a dotazovací jazyk SPARQL.

Dále bylo implementováno vlastní grafické uživatelské rozhraní, které zpřístupňuje data a umožňuje procházet rozsáhlé sbírky a úložiště s heterogenními daty.

Diplomová práce umožňuje pracovat s metadaty různých formátů dat, jako jsou hudební soubory (MP3, MP4, OGG, atd.), obrázky (JPG, JPEG, PNG, GIF, atd.) či dokumenty (PDF). V původním návrhu bylo požadováno čtení metadat z širší škály různých formátů, jako jsou dokumenty (XLS, TXT, ODT, DOC, atd.) a video soubory (AVI a MOV). Od těchto požadavků se odstoupilo z důvodu, že nástroj pro správu sbírek byl rozšířen o možnost dynamické práce s ontologiemi.

Aplikace byla rozšířena o již zmiňovanou dynamickou práci s ontologiemi, která umožňuje uživateli vkládat a editovat ontologie do 3 různých kategorií (obrázky, hudba, dokumenty).

Diplomová práce umožňuje vyhledávat data dle typu kategorie (obrázky, hudba a dokumenty) a její vlastností (vlastnosti jsou dynamicky definované dle přiřazené ontologie). Uživatel může vyhledávat soubory pomocí více zadaných hodnot (např. u obrázků bude uživatel vyhledávat data podle jeho šířky, výšky, názvu, atd.). Po nalezení souborů pomocí zadaných metadat se otevře „nová podstránka s výsledkem vyhledávání“, ve které lze data exportovat do ZIP archivu či je zobrazovat/prezentovat pomocí externích programů.

Jsem přesvědčen, že všechny požadavky zadané v rámci diplomové práce jsem splnil. Výsledná aplikace byla testována jak na reálných datech, tak i u reálných uživatelů, kteří i přes některé grafické nedostatky (zobrazování názvu souborů v hash tvaru, vzhled informačních hlášek, nezobrazování aktuálního stavu při nahrávání dat, atd.) hodnotili práci kladně.

Slovníček pojmů a přehled zkratek

Archiv	- soubor záznamů uložených v zabaleném souboru pomocí archivní metody (například ZIP)
C	- je programovací jazyk, který se využívá především při psaní systémového softwaru, ale i pro aplikace.
Inference	- usuzování, odvozování určitých výroků z jiných
Interoperabilita	- schopnost systémů vzájemně si poskytovat služby a efektivně spolupracovat - v dopravě: vícesystémová mezinárodní provozuschopnost (železnic, vozidel)
JAVA	- je objektově orientovaný programovací jazyk, vycházející z programovacího jazyka C++, ke kterému se také syntakticky nejblíže podobá
JENA	- slouží jako open source framework pro tvorbu aplikací sémantického webu vytvořeného v programovacím jazyce Java. Byl vyvinut společností Hewlett-Packard Company v rámci programu pro rozvoj sémantického webu.
Open source	- označuje program, který má volně k dispozici svoje zdrojové kódy
RDF	- (*Resource Description Framework)* zkráceně by jsme mohli říci, že se jedná o datový model reality, který se popisuje trojicemi ve tvaru subject – predikát - objekt
Relační databáze	- je databáze založená na relačním modelu. Často se tímto pojmem označuje nejen databáze samotná, ale i její konkrétní softwarové řešení.
SPARQL	- je dotazovací jazyk určený pro práci v RDF úložišti *(SPARQL Protocol and RDF Query Language)*
SQL	- je standardizovaný dotazovací jazyk používaný pro práci s daty v relačních databázích
UML	- je nástroj, který se používá při vývoji systému. Jedná se o souhrn grafických prvků pro vyjádření návrhových a analytických modelů. (the Unified Modeling Language)
URI	- Uniform Resource Identifier, nebo-li jedinečná identifikace zdroje. (například ftp://svn.enakupujeme.cz/diplomovaPrace/branches/)
URL	- je způsob, jak jednoznačně zapsat umístění souboru na Inernetu nebo na intranetu. URL je synonymem pro internetové adresy. (*Unique Resource Locator – jednoznačné určené zdroje)*
W3C	World Wide Web Consortium je mezinárodní konsorcium, jehož členové společně s veřejností vyvíjejí webové standardy pro WWW
World Wide Web	- WWW nebo také zkráceně web, ve volném překladu „celosvětová pavučina“, je označení pro aplikace internetového

	protokolu http. Je tím myšlena soustava propojených hypertextových dokumentů. [1]
XML	- je značkovací jazyk, který byl vyvinut a standardizován konsorciem W3C
ZIP	- je všeobecně rozšířený souborový formát pro kompresi a archivaci dat

Literatura

[1] Bradley Mitchell. WWW- World Wide Web. About.com [online]. [cit. 2012-05-02]. Dostupné z WWW: <http://compnetworking.about.com/cs/worldwideweb/g/bldef_www.htm>.

[2] W3C SEMANTIC WEB ACTIVITY. W3C [online]. 2011-07-11 [cit. 2012-05-02]. Dostupné z WWW: <http://www.w3.org/2001/sw/>.

[3] Resource Description Framework (RDF). W3C [online]. 2004-02-10 [cit. 2012-05-02]. Dostupné z WWW: <http://www.w3.org/RDF/>.

[4] Matulík, Petr; Pitner, Tomáš. Sémantický web a jeho technologie. Zpravodaj ÚVT MU. [online]. 2004, roč. XIV, č. 3, s. 15-17 [cit. 2011-12-18]. Dostupné z WWW: <http://www.ics.muni.cz/bulletin/articles/296.html>. ISSN 1212-0901.

[5] Doc. Ing. Vojtěch Svátek, Dr. Ontologie, OWL a deskripční logika. vse.cz [online]. 2012 [cit. 2012-05-02]. Dostupné z <http://nb.vse.cz/~svatek/rzzw/owl-dl.pdf>.

[6] Ontologie (informatika). Wikipedie Infostar [online]. [cit. 2011-12-18]. Dostupné z WWW: <http://wikipedia.infostar.cz/o/on/ontology__computer_science_.html>.

[7] OWL Web Ontology Language : Overview. W3C [online]. 2004, 2009-11-12 [cit. 2011-12-18]. Dostupné z WWW: <http://www.w3.org/TR/owl-features/>.

[8] Hanyáš, Petr. Jazyk OWL : Tutoriál. Hanyas [online]. 2007 [cit. 2011-12-18]. Dostupné z WWW: <http://www.hanyas.net/seweb/tutorial.php?article=105>.

[9] Motejlková, Anna. Technologie sémantického webu. Ikaros [online]. 2011, roč. 15, č. 10 [cit. 2011-12-18]. Dostupné z WWW: <http://www.ikaros.cz/technologie-semanticke-ho-webu>. URN-NBN: cz-ik7162. ISSN 1212-5075.

[10] Zezula, Ondřej. Znalostní úložiště. [online]. 2010 [cit 2011-12-18]. Dostupné z WWW: <http://is.muni.cz/th/172856/fi_b/?jazyk=en;info>.

[11] Holec, Matěj. Agent pro extrakci informací z genomických databází na webu. [online]. 2007 [cit. 2011-12-18]. Dostupné z WWW: <https://dip.felk.cvut.cz/browse/pdfcache/holecm1_2007dipl.pdf>.

[12] Představení jazyka OWL. Znalostní technologie I [online]. [cit 2011-12-18]. Dostupné z WWW: <http://lide.uhk.cz/fim/ucitel/fshusam2/lekarnicky/zt1/zt1_kap04.html>.

[13] RDF: Wikiverzita [online]. 2011-08-21 [cit. 2011-12-18]. Dostupné z WWW: <http://cs.wikiversity.org/wiki/RDF>

[14] Čížek, Pavel: Úložiště znalostí. [online]. 2007 [cit. 2011-12-18]. Dostupné z WWW: <http://is.muni.cz/th/60897/fi_m/>.

[15] McBride, Brian. An Introduction to RDF and the Jena RDF API. Jena [online]. 2010-12-17 [cit. 2011-12-18]. Dostupné z WWW: <http://jena.sourceforge.net/tutorial/RDF_API/>.

[16] Ing. Sklenák, Csc, Vilém. Metadata, sémantika a sémantický web. [online]. 2004 [cit. 2011-12-18]. Dostupné z WWW: <http://www.inforum.cz/pdf/2004/Sklenak_Vilem1.pdf>.

[17] David Beckett, Tim Berners-Lee. Turtle – Terse RDF Triple Language. W3C [online]. 2011-03-28 [cit. 2012-05-02]. Dostupné z WWW: <http://www.w3.org/TeamSubmission/turtle/>.

[18] Stickler, Patrick. TriX - RDF Triples in XML. Nokia [online]. [cit. 2011-12-18]. Dostupné z WWW: <http://swdev.nokia.com/trix/>.

[19] Stickler, Patrick. TriX – RDF Triples in XML (examples). Nokia [online]. [cit. 2011-12-18]. Dostupné z WWW: <http://swdev.nokia.com/trix/examples.xml>.

[20] Hanyáš, Petr. Základní informace : Tutoriál. Hanyas [online]. 2007 [cit. 2011-12-18]. Dostupné z WWW: <http://www.hanyas.net/seweb/tutorial.php?article=115>.

[21] ParLannero. NEPOMUK – The Social Semantic Desktop. NEPOMUK [online]. Ver. 1.22, 2007-04-13 [cit. 2011-12-18]. Dostupné z WWW: <http://nepomuk.semantic-desktop.org/xwiki/bin/view/Main1/>.

[22] Černý, Michal. Sémantický desktop: praktické aplikace. Myšlenkové mapy [online]. 2011 [cit. 2011-12-18]. Dostupné z WWW: <http://www.myslenkove-mapy.cz/osobni-rozvoj/aktuality/semanticky-desktop-prakticke-aplikace/>.

[23] Černý, Michal. Sémantický desktop: aplikace myšlenkových map v počítačích?. Myšlenkové mapy [online]. 2011 [cit. 2011-12-18]. Dostupné z WWW: <http://www.myslenkove-mapy.cz/osobni-rozvoj/aktuality/semanticky-desktop-aplikace-myslenkovych-map-v-pocitacich/>.

[24] Černý, Michal. Sémantický... web i desktop. Root [online]. 2011 [cit. 2011-12-18]. Dostupné z WWW: <http://www.root.cz/clanky/semanticky-desktop-chytrejsi-prace-s-daty-v-souvislostech/>.

[25] Zemský, Igor. Vizualizace RDF dat. [online]. 2009 [cit. 2011-12-18]. Dostupné z WWW: <http://is.muni.cz/th/60726/fi_m/thesis-print.pdf>.

[26] Oracle Database Semantic Technologies. Oracle [online]. [cit. 2011-12-18]. Dostupné z WWW: <http://www.oracle.com/technetwork/database/options/semantic-tech/index.html>.

[27] Oracle database semantic technologies. Oracle [online]. [cit. 2011-12-18]. Dostupné z WWW: <http://www.oracle.com/technetwork/database/options/semantic-tech/semtech11gr2-featover-131765.pdf>.

[28] Network Data Model Overview. Oracle [online]. [cit. 2011-12-18]. Dostupné z WWW: <http://docs.oracle.com/cd/B28359_01/appdev.111/b28399/sdo_net_concepts.htm>.

[29] Dickinson, Ian. Jena Ontology API. Jena [online]. 2009-02-24 [cit. 2011-12-18]. Dostupné z WWW: <http://jena.sourceforge.net/ontology/>.

[30] RDF query language. Wikipedia [online]. 2010-06-25 [cit. 2011-12-18]. Dostupné z WWW: <http://en.wikipedia.org/wiki/RDF_query_language>.

[31] Sohlich, Radomír: Dotazování a pravidla v Sémantickém webu. [online]. 2010 [cit. 2011-12-18]. Dostupné z WWW: <https://dip.felk.cvut.cz/browse/pdfcache/sohlirad_2010bach.pdf>.

[32] Karvounarakis, Greg; Christophides, Vassilis. The RDF Query Language (RQL). FORTH [online]. 2008-10-09 [cit. 2011-12-18]. Dostupné z WWW: <http://139.91.183.30:9090/RDF/RQL/>.

[33] RQL v2.1 User Manual. FORTH [online]. 2003-07-17 [cit. 2011-12-18]. Dostupné z WWW: <http://139.91.183.30:9090/RDF/RQL/Manual.html>.

[34] Warchil, Miroslav. Vizualizace dat pomocí stylového jazyka W3C Fresnel [online]. 2010 [cit. 2011-12-18]. Dostupné z WWW: <http://is.muni.cz/th/143256/fi_m/>.

[35] openRDF.org: Home [online]. c1997 [cit. 2011-12-18]. Dostupné z WWW: <http://www.openrdf.org/>.

[36] Sesame. W3C – Semantic Web [online]. 2011-03-14 [cit. 2011-12-18]. Dostupné z WWW: <http://www.w3.org/2001/sw/wiki/Sesame>.

[37] Prud'hommeaux, Eric; Seaborne, Andy. SPARQL Query Language for RDF. W3C [online]. 2006, 2008-01-25 [cit. 2011-12-18]. Dostupné z WWW: <http://www.w3.org/TR/rdf-sparql-query/>.

[38] Raška, Jiří. RDF úložiště v PHP. [online]. 2011 [cit. 2011-12-18]. Dostupné z WWW: <http://www.fit.vutbr.cz/study/DP/rpfile.php?id=11612>.

[39] Metadata Extraction in Java. [online]. [cit. 2011-12-18]. Dostupné z WWW: <http://www.drewnoakes.com/code/exif/>.

[40] JThink. Jaudiotagger Library [online]. c2004 [cit. 2011-12-18]. Dostupné z WWW: <http://www.jthink.net/jaudiotagger/>.

[41] Mylka, Antoni; Sauermann, Leo; Sintek, Michael; Elst, Ludger van. NEPOMUK EXIF Ontology. NEPOMUK [online]. c2007 [cit. 2011-12-18]. Dostupné z WWW: <http://www.semanticdesktop.org/ontologies/nexif/>.

[42] Mylka, Antoni; Sauermann, Leo; Sintek, Michael; Elst, Ludger van. NEPOMUK ID3 Ontology. NEPOMUK [online]. c2007 [cit. 2011-12-18]. Dostupné z WWW: <http://www.semanticdesktop.org/ontologies/nid3/>.

[43] Vadinský, Ondřej. Nepomuk-KDE : Sémantický desktop pro Linux. ABC Linux [online]. 2010 [cit. 2011-12-18]. Dostupný z WWW: <http://www.abclinuxu.cz/clanky/nepomuk-kde-semantickydesktopprolinux#projekt-nepomuk>.

[44] OSCA Foundation | Open Semantic Collaboration Architecture Foundation [online]. [cit. 2011-12-18]. Dostupné z WWW: <http://www.oscaf.org/>.

[45] Štencek, Jiří. Užití sémantických technologií ve značkovacích jazycích [online]. 2009-12 [cit. 2012-03-28]. Dostupné z WWW: <http://vse.stencek.com/semanticky-web/>.

[46] Adida, Ben; Birbeck, Mark. RDFa Primer: Bridging the Human and Data Webs [online]. 2008 [cit. 2012-03-28]. Dostupné z WWW: <http://www.w3.org/TR/xhtml-rdfa-primer/>.

[47] Adida, Ben; Birbeck, Mark; McCarron, Shane; Pemberton, Steven. RDFa in XHTML: Syntax and Processing [online]. 2007 [cit. 2012-03-28]. Dostupné z WWW: <http://www.w3.org/TR/rdfa-syntax/>.

[48] Malinský, Marek. Značkovací jazyky sémantického webu [online]. 2010 [cit. 2012-03-28]. Dostupné z WWW: <http://is.muni.cz/th/172622/fi_b/bc_znackovaci_jazyky_semantickeho_webu.txt>

[49] Programovací jazyk Java: Úvodní informace [online]. [cit. 2012-03-30]. Dostupné z WWW: <http://v1.dione.zcu.cz/java/uvod.html>

[50] Herout, Pavel. Učebnice jazyka Java [kniha: ISBN 80-7232-115-3]. 2005 [cit. 2012-03-30]

[51] Hatina, Petr. Programování v jazyku Java [online]. 2004-07-09 [cit. 2012-03-30]. Dostupné z WWW: <http://www.linuxsoft.cz/article.php?id_article=244>

[52] Vývojová prostředí a překladače. Sallyx [online]. [cit. 2012-03-30]. Dostupné z WWW: <http://www.sallyx.org/sally/c/vyvojova-prostredi.php>

[53] Tišnovský, Pavel. Eclipse – integrované vývojové prostředí pro Javu i další programovací jazyky [online]. 2012-01-18 [cit. 2012-03-30]. Dostupné z WWW: <http://fedora.cz/eclipse-integrovane-vyvojove-prostredi-pro-javu-i-dalsi-programovaci-jazyky/>

[54] Netbeans. Oracle [online]. 2012 [cit. 2012-03-30]. Dostupné z WWW: <http://netbeans.org/index_cs.html>

[55] Pitka, Lukáš. Vývojové prostředí NetBeans [online]. 2007-06-18 [cit. 2012-03-30]. Dostupné z WWW: <http://theses.cz/id/zjkh9a/>

[56] QT knihovna. Wikipedie [online]. 2008-10-06 [cit. 2012-03-30]. Dostupné z WWW: <http://qt-aplikace.blogspot.com/2008/10/qt-knihovna.html>

[57] Visual Studio. Sallyx [online]. [cit. 2012-03-30]. Dostupné z WWW: <http://www.sallyx.org/sally/c/visual-studio.php>

[58] Vogel, Jiří. Programovací jazyk C++ [online]. 1998-07-11 [cit. 2012-03-30]. Dostupné z WWW: <http://www.fsid.cvut.cz/cz/U201/skrcpp.html#tth_chAp1>

[59] Programování v jazyku C/C++. Sallyx [online]. [cit. 2012-03-30]. Dostupné z WWW: <http://www.sallyx.org/sally/c/>

[60] Herout, Pavel. Učebnice jazyka C [kniha: ISBN 80-7232-220-6]. 2005 [cit. 2012-03-30]

[61] Bizer, Chris; Schultz, Andreas. Berlin SPARQL Benchmark Results [online]. 2009-03-23 [cit. 2012-03-31]. Dostupné z WWW: <http://www4.wiwiss.fu-berlin.de/bizer/BerlinSPARQLBenchmark/results/index.html>

[62] Pavus. UML – úvod a historie [online]. [cit. 2012-04-08]. Dostupné z WWW: <http://mpavus.wz.cz/uml/uml-uvod-1.php>

[63] Šimerda. Projektování softwarových systémů – UML [online]. 2006-10-09 [cit. 2012-04-04]. Dostupné z WWW: <http://pss.tym.cz/>

[64] Tolle, Karsten. eRQL Documentation. DBIS Informatik Frankfurt [online]. [cit. 2012-04-05]. Dostupné z WWW: <http://www.dbis.informatik.uni-frankfurt.de/~tolle/RDF/eRQL/eRQLDocumentation.html>

[65] Torres, David. File: URI Euler Diagram no lone URIs.svg. WikiMedia Commons [online]. 2012-03-25 [cit. 2012-05-05]. Dostupné z WWW: <http://commons.wikimedia.org/wiki/File:URI_Euler_Diagram_no_lone_URIs.svg>

[66] Beckett, Dave; McBride, Brian. RDF/XML Syntax Specification (Revised). W3C [online]. 2004-02-10 [cit. 2012-05-05]. Dostupné z WWW: <http://www.w3.org/TR/REC-rdf-syntax/>

[67] Berners-Lee, Tim; Connolly, Dan. Notation 3 (N3): A readable RDF syntax. W3C [online]. 2011-03-28 [cit. 2012-05-05]. Dostupné z WWW: <http://www.w3.org/TeamSubmission/n3/>

[68] Beckett, David. N-Triples. W3C [online]. 2001-09-06 [cit. 2012-05-05]. Dostupné z WWW: <http://www.w3.org/2001/sw/RDFCore/ntriples/>

[69] Hertel, Alice; Broekstra, Jeen; Stuckenschmidt, Heiner. RDF Storage and Retrieval Systems. Universität Mannheim [online]. [cit. 2012-05-05]. Dostupné z WWW: <http://ki.informatik.uni-mannheim.de/fileadmin/publication/Hertel08RDFStorage.pdf>

[70] Bratková, Eva. Metadata jako nový nástroj pro komunikaci webových informačních zdrojů. Národní knihovna [online]. 1999 [cit. 2012-05-05]. Dostupné z WWW: <http://full.nkp.cz/nkkr/Nkkr9904/9904178.html>

[71] Understanding Metadata. NISO Press [online]. 2004 [cit. 2012-05-05]. Dostupné z WWW: <http://www.niso.org/publications/press/UnderstandingMetadata.pdf>

[72] Málek, Vilém. Metadata a hlavička pro XHTML dokument. Interval.cz [online]. Dostupné z WWW: <http://interval.cz/clanky/metadata-a-hlavicka-pro-xhtml-dokument/>

[73] RDF Vocabulary Description Language 1.0: RDF Schema (RDFS). W3C – Semantic Web [online]. 2004-02-10 [cit. 2012-05-06]. Dostupné z WWW: <http://www.w3.org/2001/sw/wiki/RDFS>

[74] Obítko, Marek. RDF Schema RDFS. Ontologies and Semantic Web [online]. 2007 [cit. 2012-05-06]. Dostupné z WWW: <http://www.obitko.com/tutorials/ontologies-semantic-web/rdf-schema-rdfs.html>

[75] Ing. Matoušek, Kamil, Ph.D.; Ing. Kléma, Jiří, Ph.D.; Ing. Kubalík, Jiří, Ph.D.; Ing. Křemen, Petr. Přehled znalostních systémů – Systémy s umělou inteligencí. FELK ČVUT [online]. [cit. 2012-05-06]. Dostupné z WWW: <http://cw.felk.cvut.cz/lib/exe/fetch.php/courses/a7b33sui/znalostni_systemy.pdf>

[76] Brickley, Dan; Guha, R.V.; McBride, Brian. RDF Vocabulary Description Language 1.0: RDF Schema. W3C [online]. 2004-02-10 [cit. 2012-05-06]. Dostupné z WWW: <http://www.w3.org/TR/rdf-schema>

[77] Manola, Frank; Miller, Eric. RDF Primer. W3C [online]. 2004-02-10 [cit. 2012-05-06]. Dostupné z WWW: <http://www.w3.org/TR/rdf-primer>

[78] Soubor: W3c semantic web stack.jpg. Wikipedia [online]. 2006-04-06 [cit. 2012-05-06]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Soubor:W3c_semantic_web_stack.jpg>

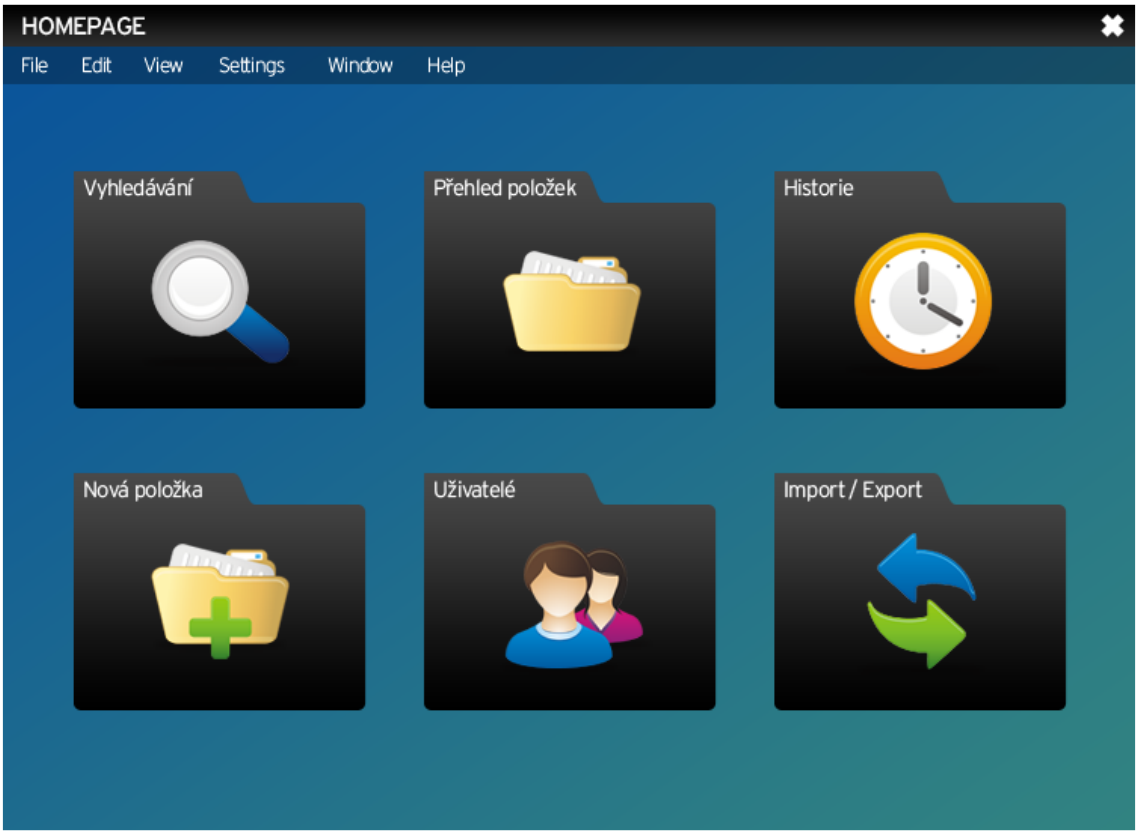
[79] Semantic Web – Main Page. Semantic Web [online]. 2012-05-01 [cit. 2012-05-06]. Dostupné z WWW: <http://semanticweb.org/wiki/Main_Page>

[80] Hawke, Sandro; Herman, Ivan; Prud'hommeaux, Eric. W3C SEMANTIC WEB ACTIVITY. W3C [online]. 2011-11-07 [cit. 2012-05-06]. Dostupné z WWW: <http://www.w3.org/2001/sw/>

[81] Chapter 8. The Repository API. OpenRDF.org [online]. [cit. 2012-05-11]. Dostupné z WWW: <http://www.openrdf.org/doc/sesame2/users/ch08.html#d0e766>

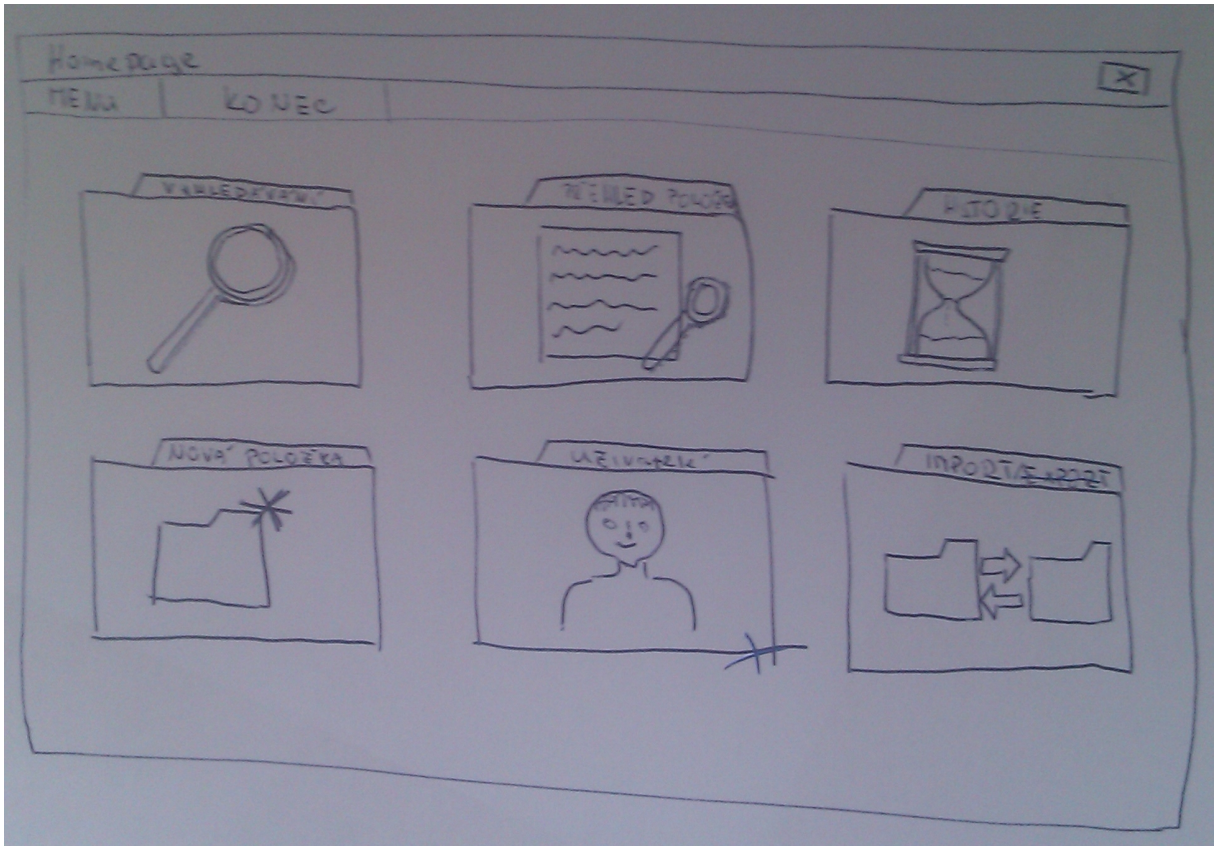
Seznam příloh

Příloha A: Grafický návrh úvodní stránky



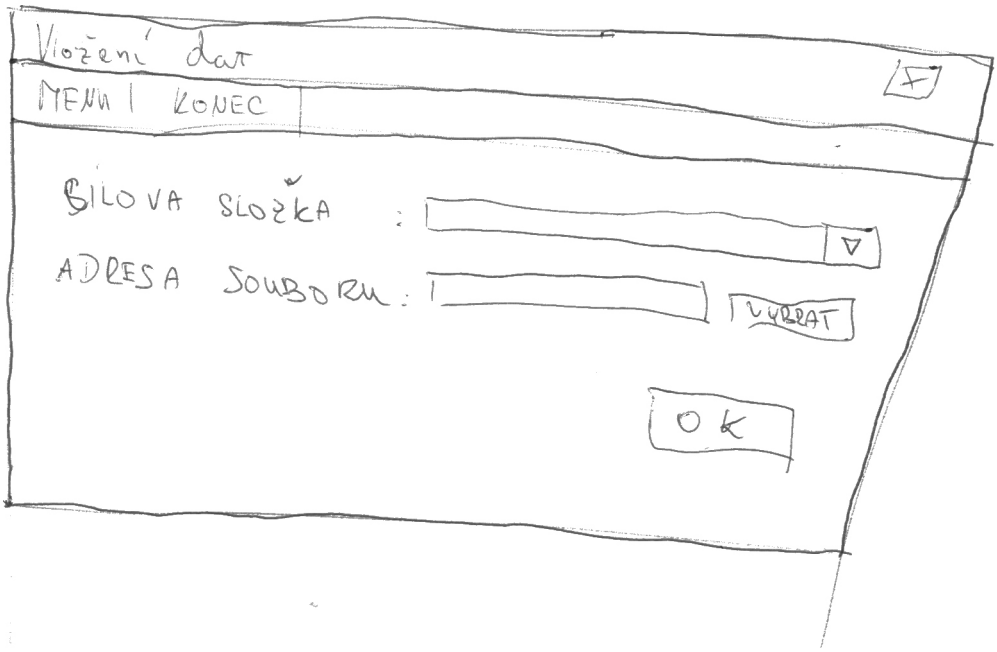
Ilustrace 11: Grafický návrh úvodní stránky externího grafika

Příloha B: Grafický náčrt podstránky „úvodní stránky“



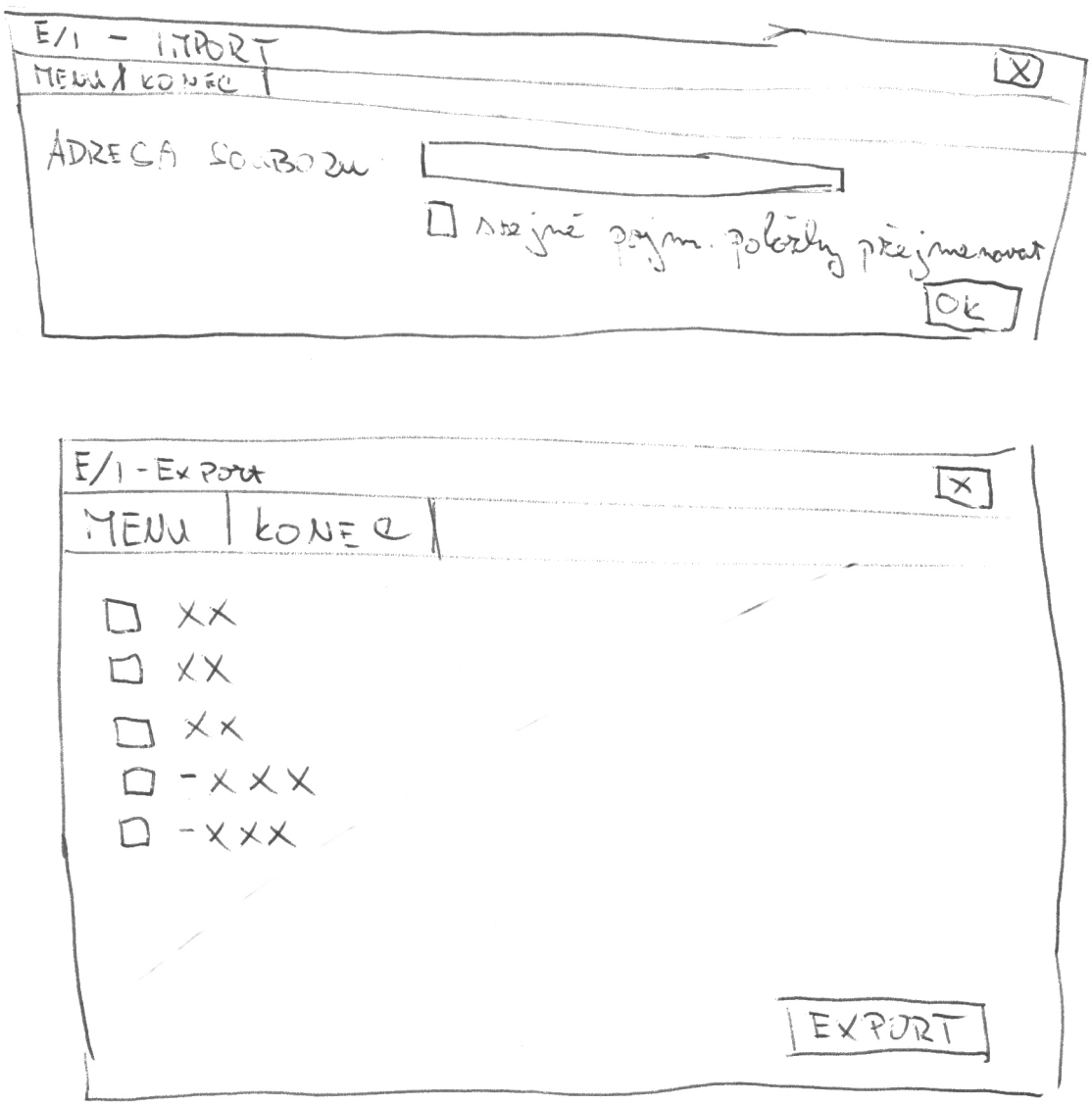
Ilustrace 12: Grafický náčrt podstránky „úvodní stránky“

Příloha C: Grafický náčrt podstránky „vkládání dat“



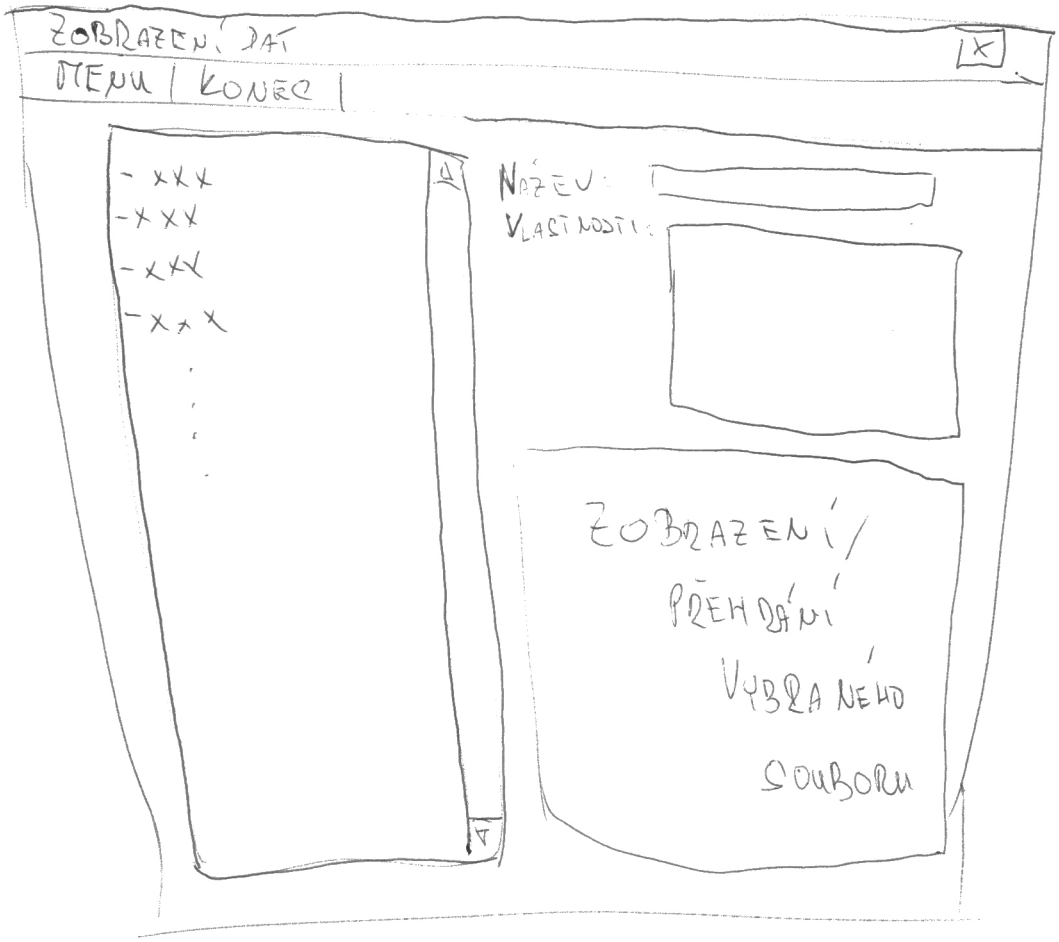
Ilustrace 13: Grafický náčrt podstránky "vkládání dat"

Příloha D: Grafický náčrt podstránky „export/import dat”



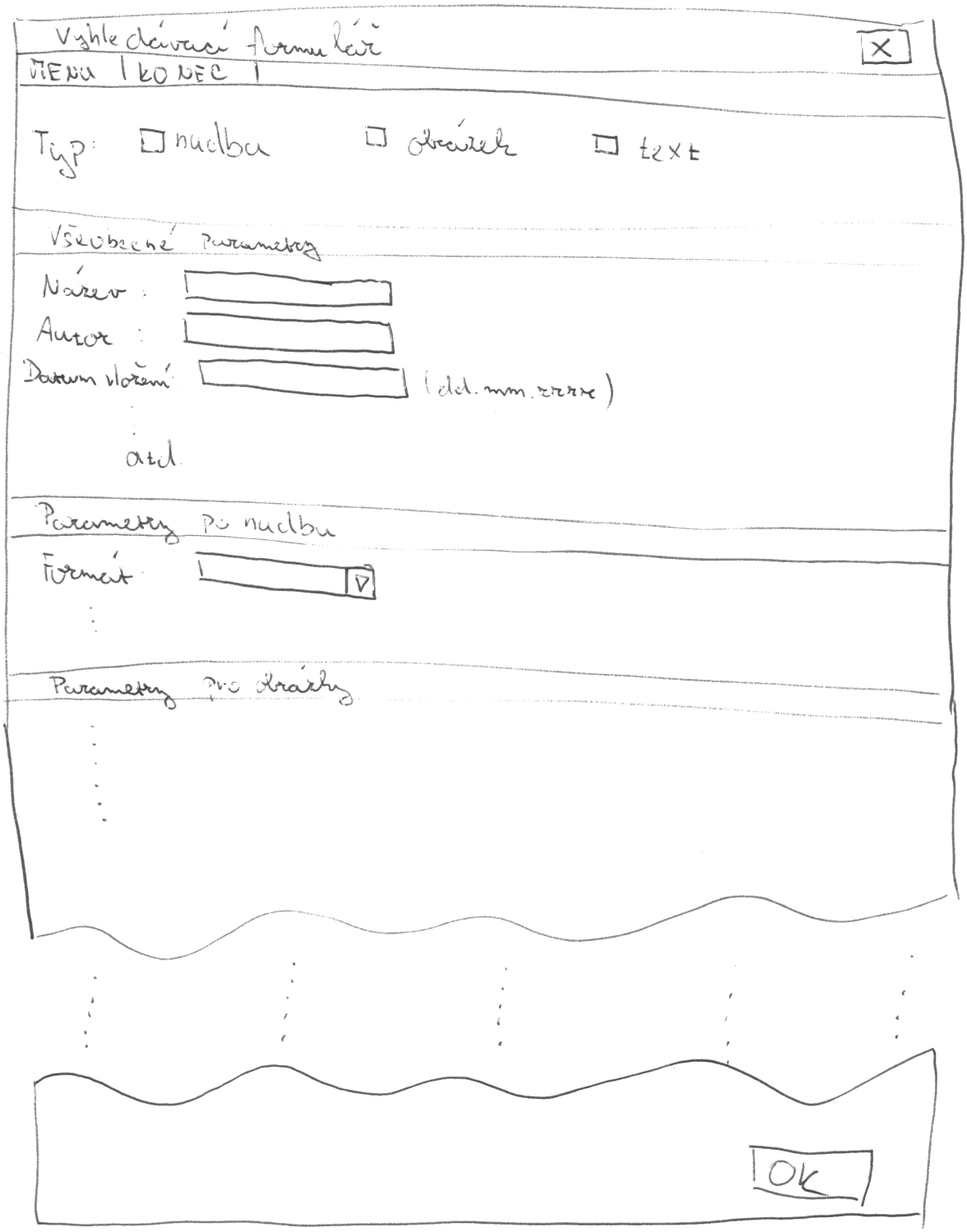
Ilustrace 14: Grafický náčrt podstránky "export/import dat"

Příloha E: Grafický náčrt podstránky „zobrazení položek”

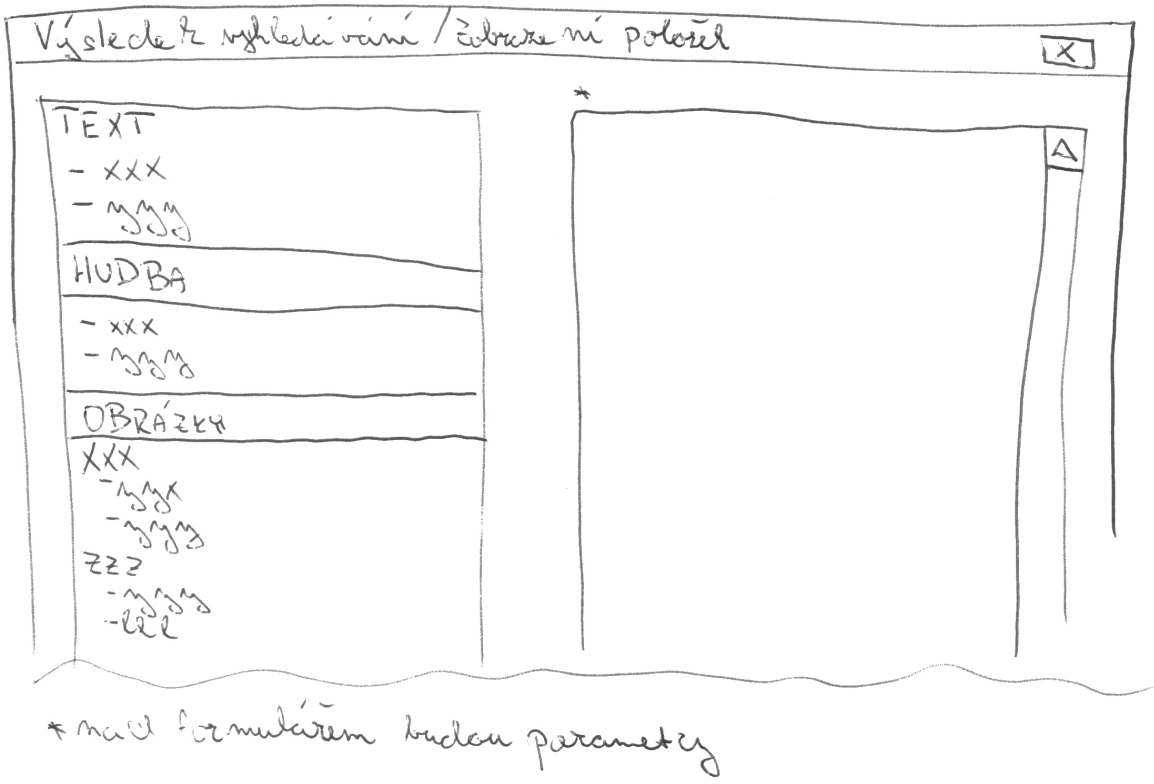


Ilustrace 15: Grafický náčrt podstránky "zobrazení položek"

Příloha F: Grafický náčrt podstránky „vyhledávání dat”

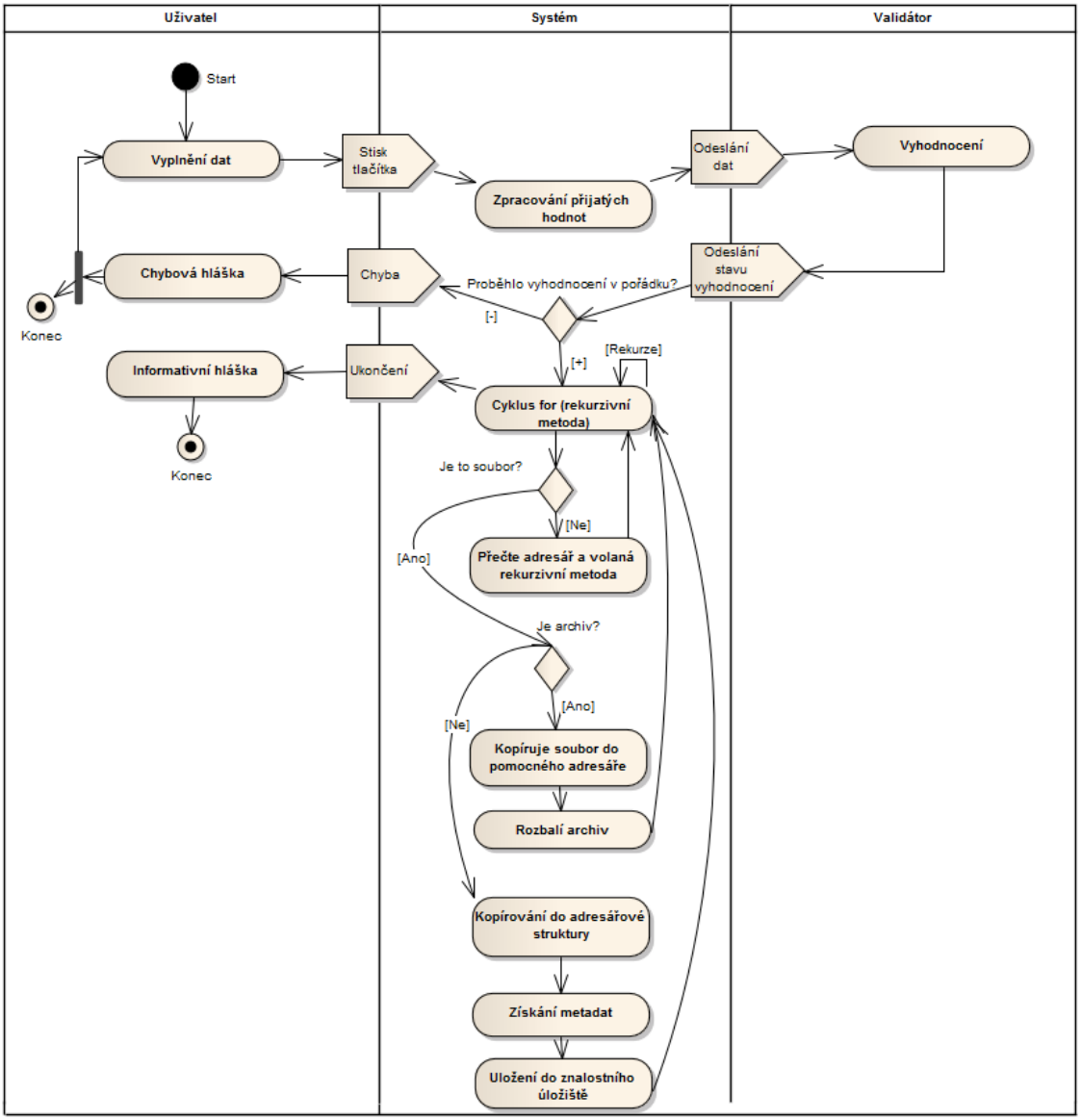


Ilustrace 16: Grafický náčrt podstránky "vyhledávání dat" - vyhledávací formulář



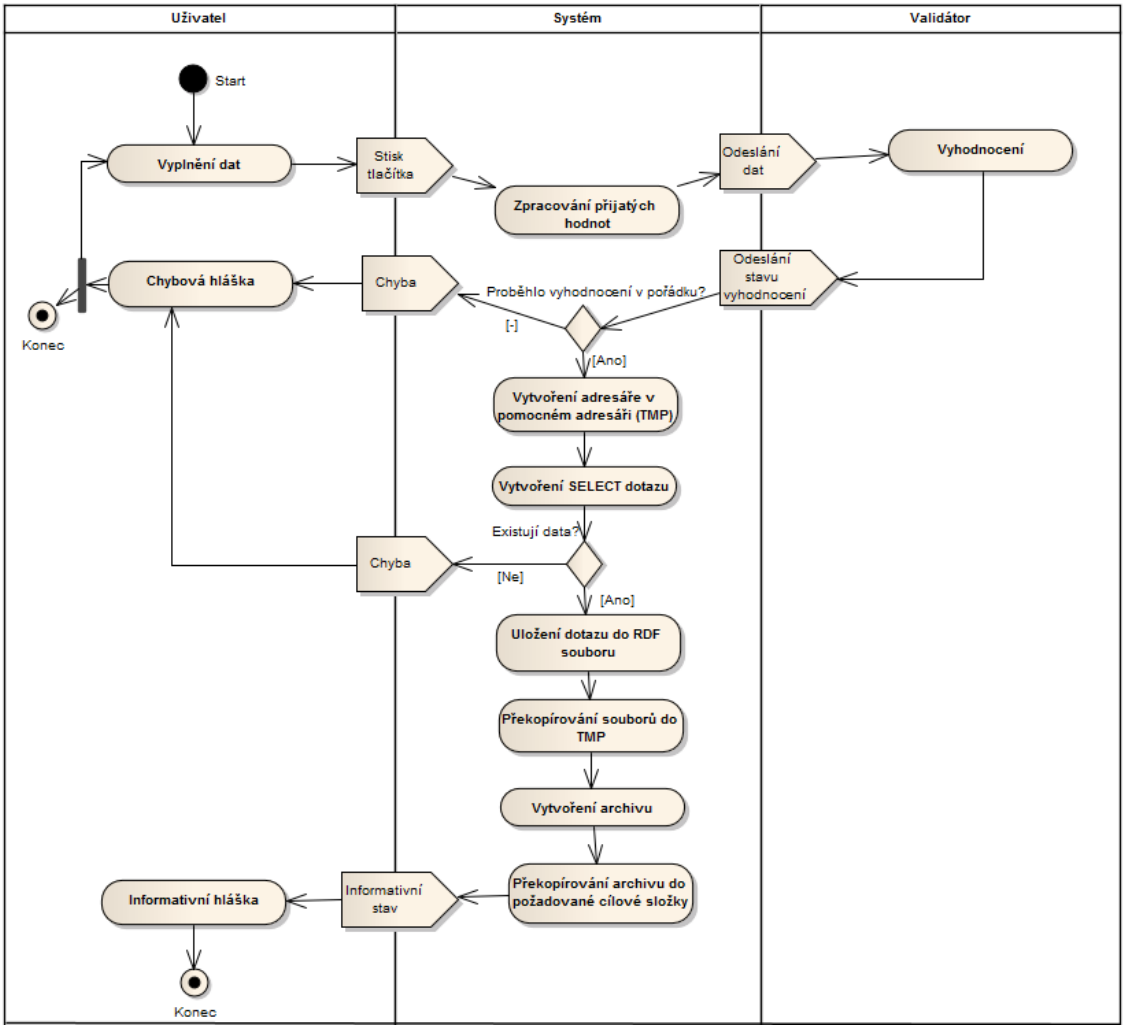
Ilustrace 17: Grafický náčrt podstránky "vyhledávání dat" - výsledek vyhledávání

Příloha G: Diagram aktivit „vkládání dat“

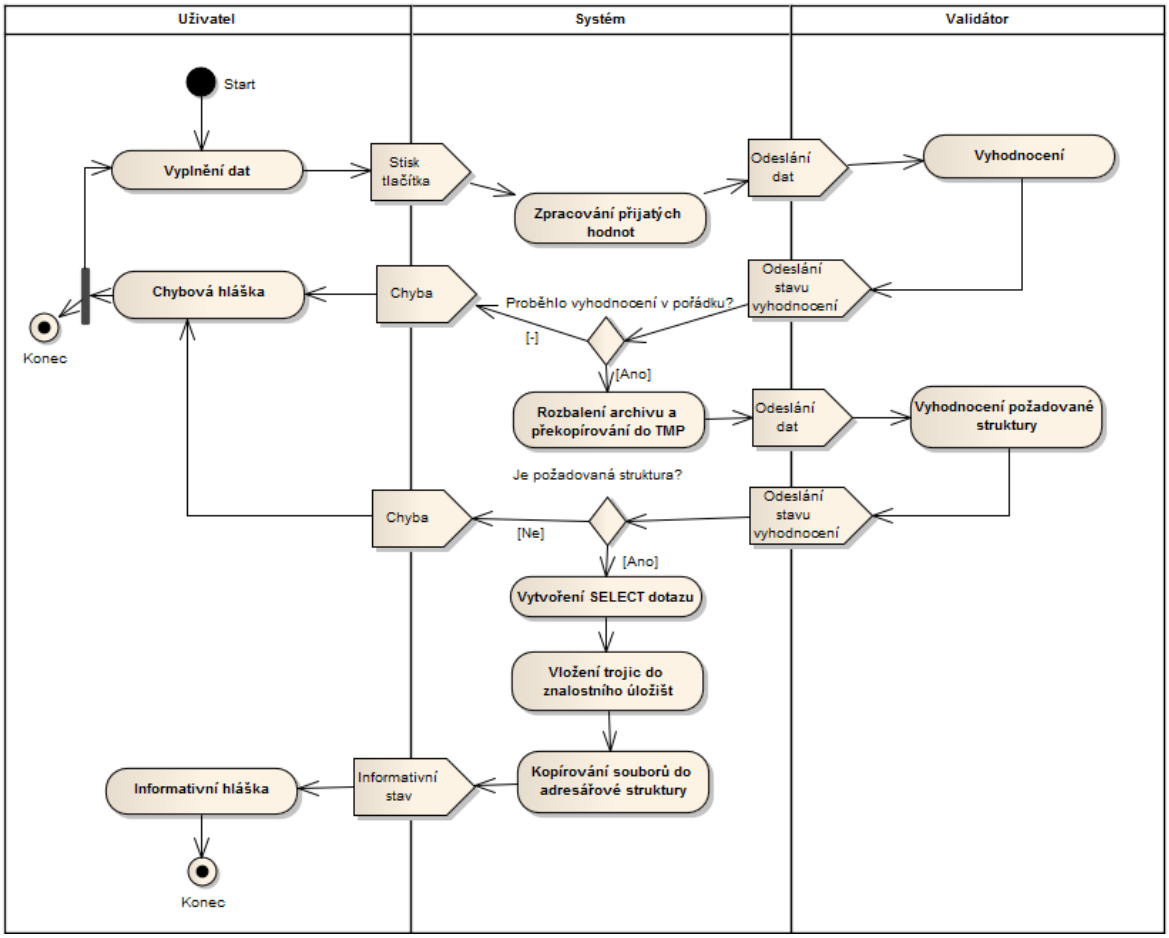


Ilustrace 18: Diagram aktivit podstránky "vkládání dat"

Příloha H: Diagram aktivit „export/import dat“

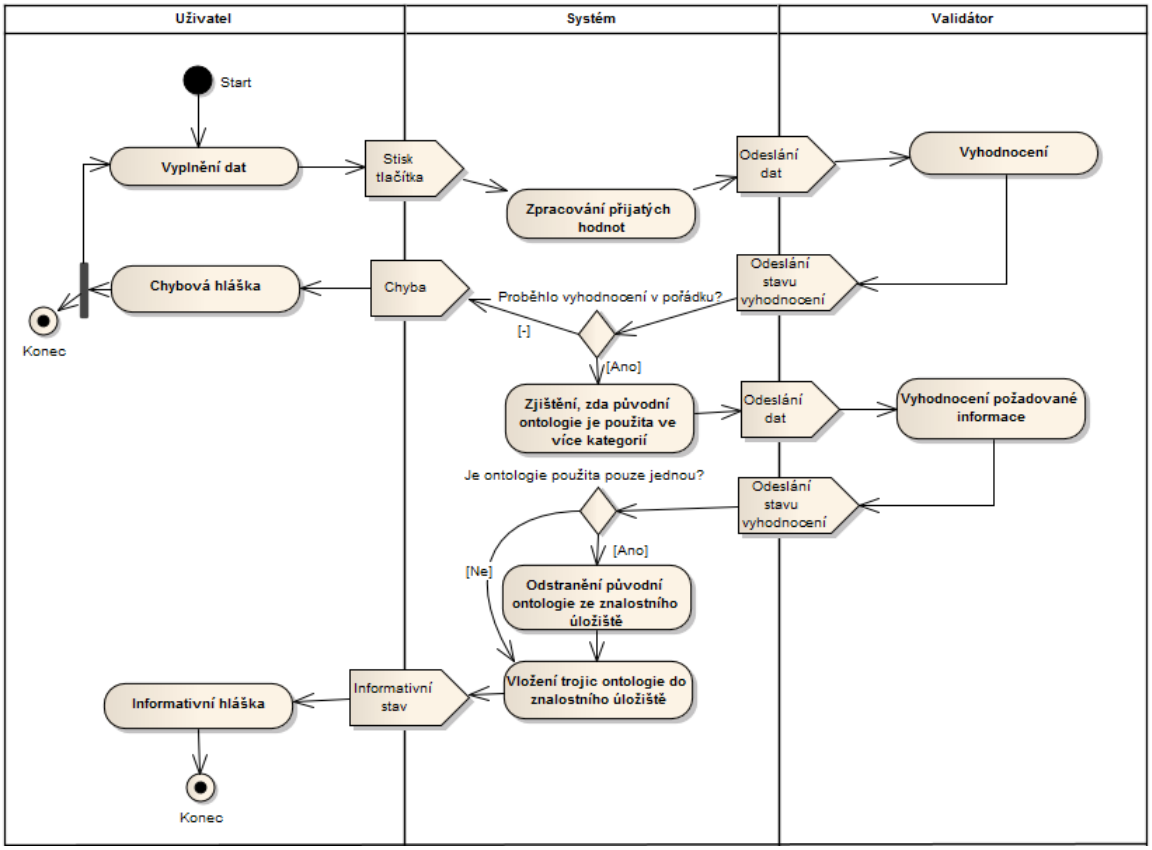


Ilustrace 19: Diagram aktivit podstránky "export/import dat" - export

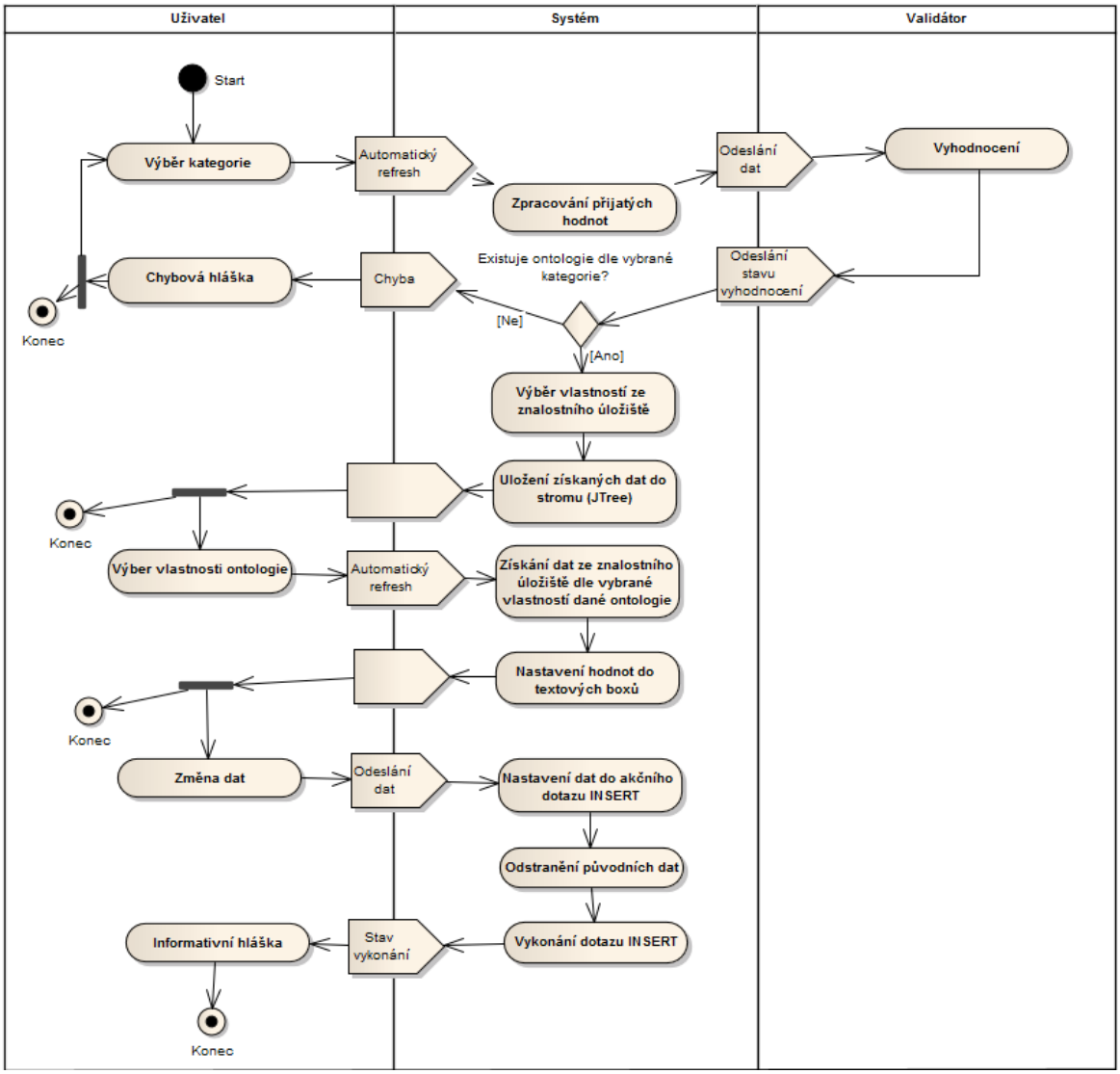


Ilustrace 20: Diagram aktivit podstránky "export/import dat" - import

Příloha I: Diagram aktivit „import ontologií“

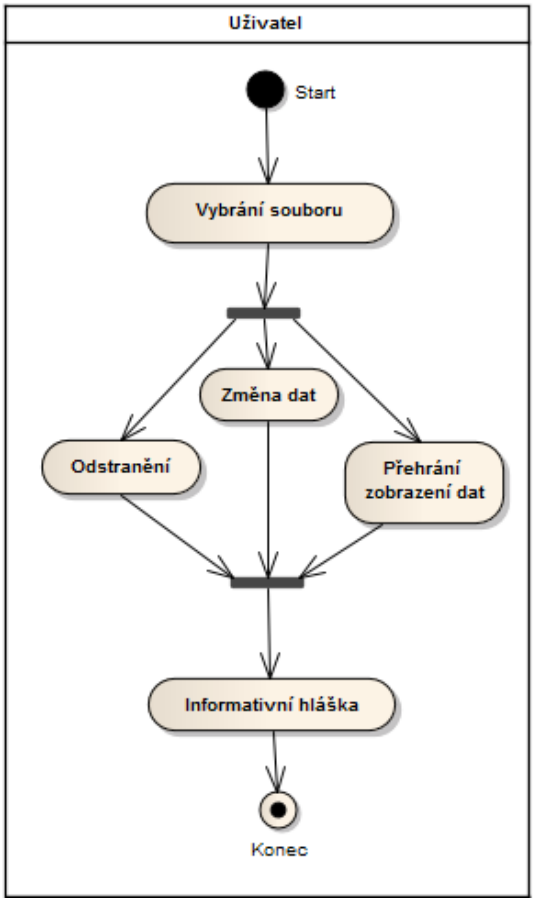


Ilustrace 21: Diagram aktivit podstránky "export ontologií" - import



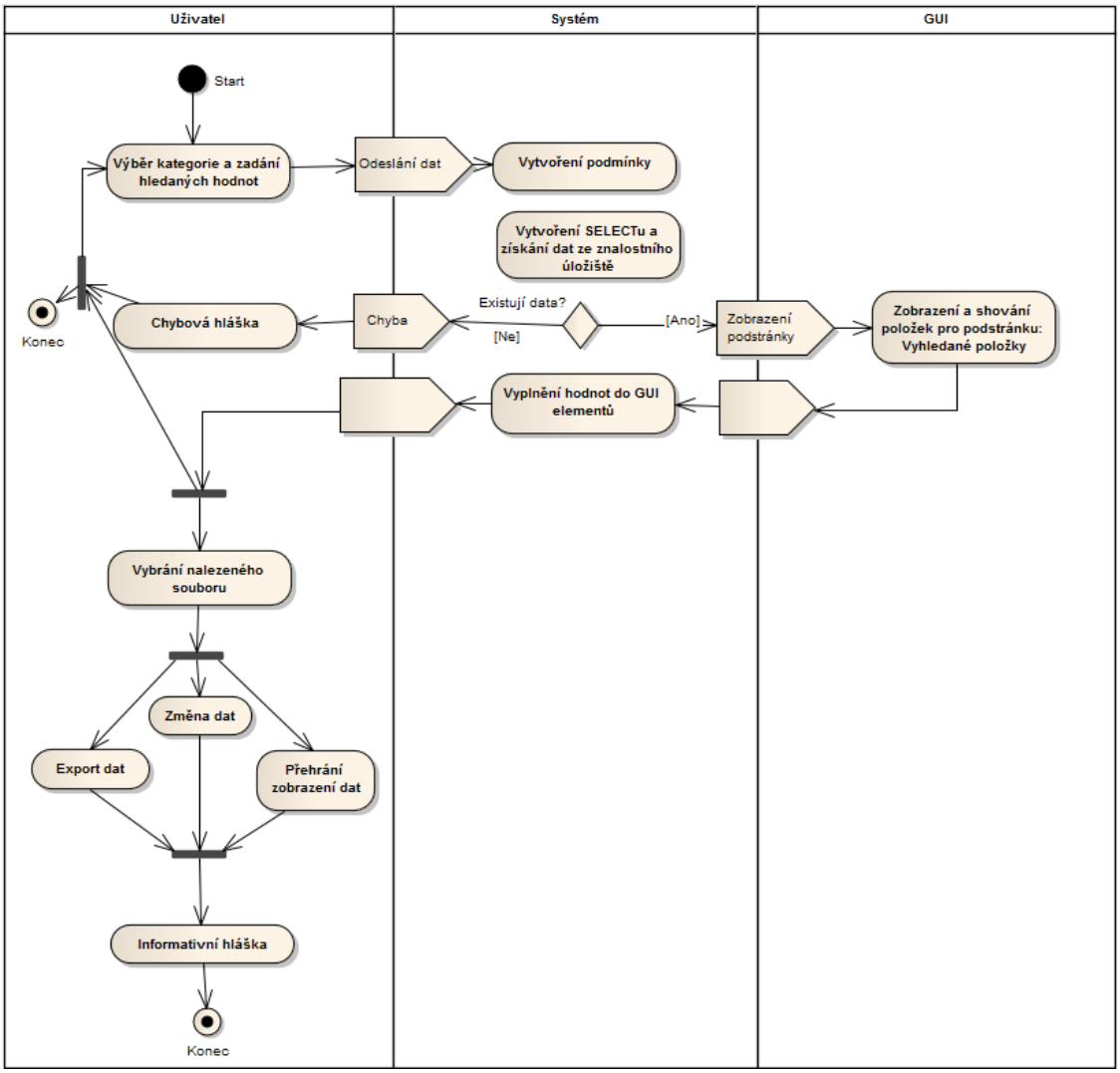
Ilustrace 22: Diagram aktivit podstránky "export ontologií" - editace vlastností

Příloha J: Diagram aktivit „zobrazení položek“



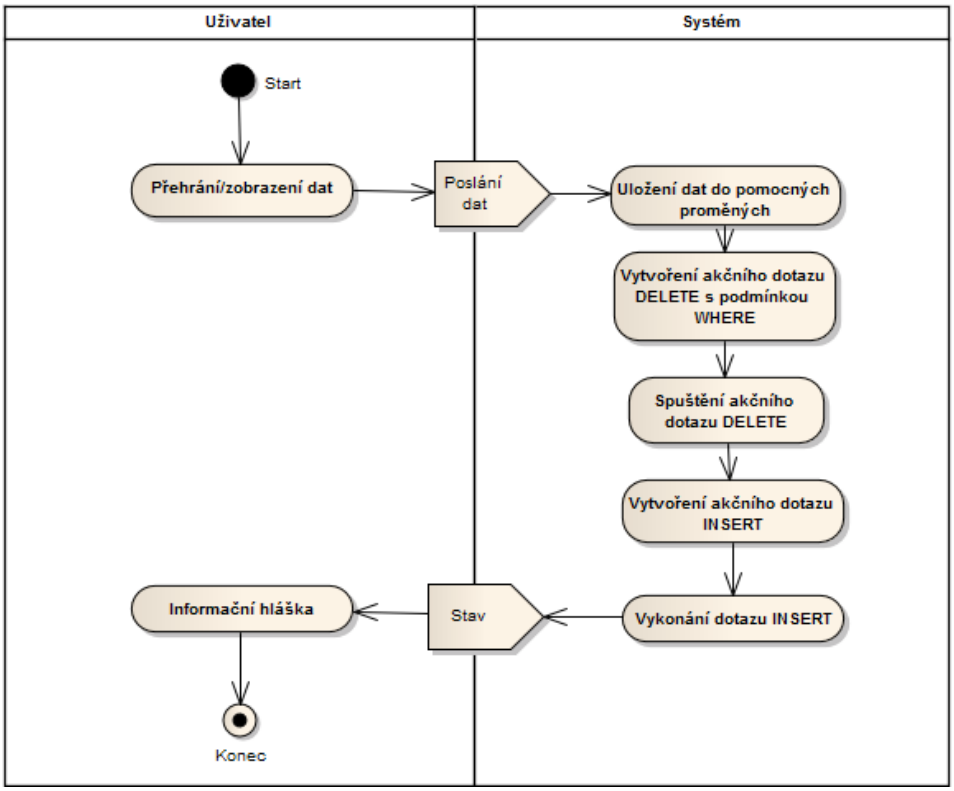
Ilustrace 23: Diagram aktivit podstránky "zobrazení položek"

Příloha K: Diagram aktivit „Vyhledávání dat“



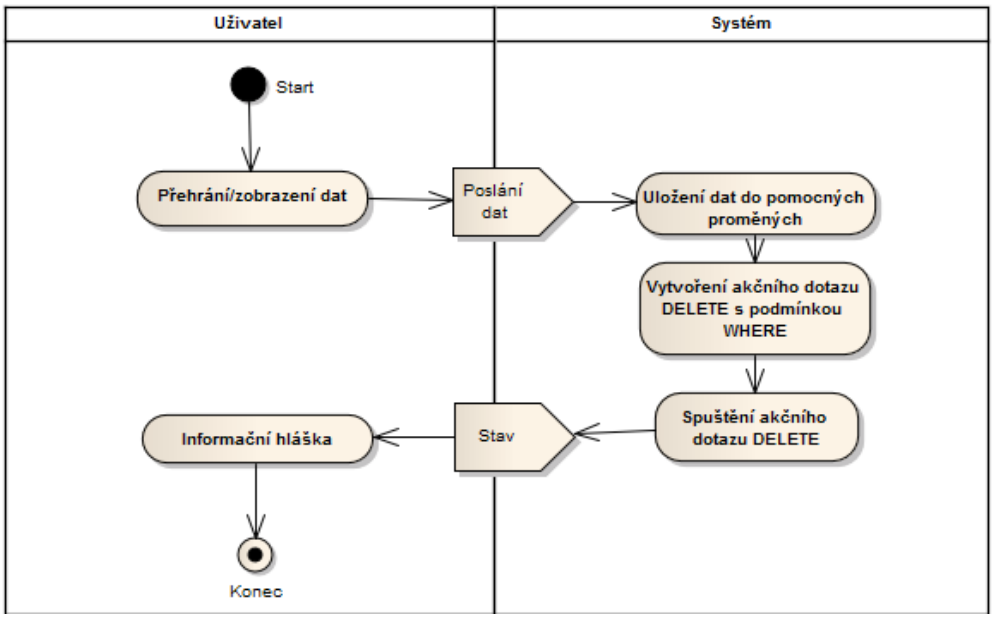
Ilustrace 24: Diagram aktivit podstránky "Vyhledávání dat"

Příloha L: Diagram aktivit „obecná editace“



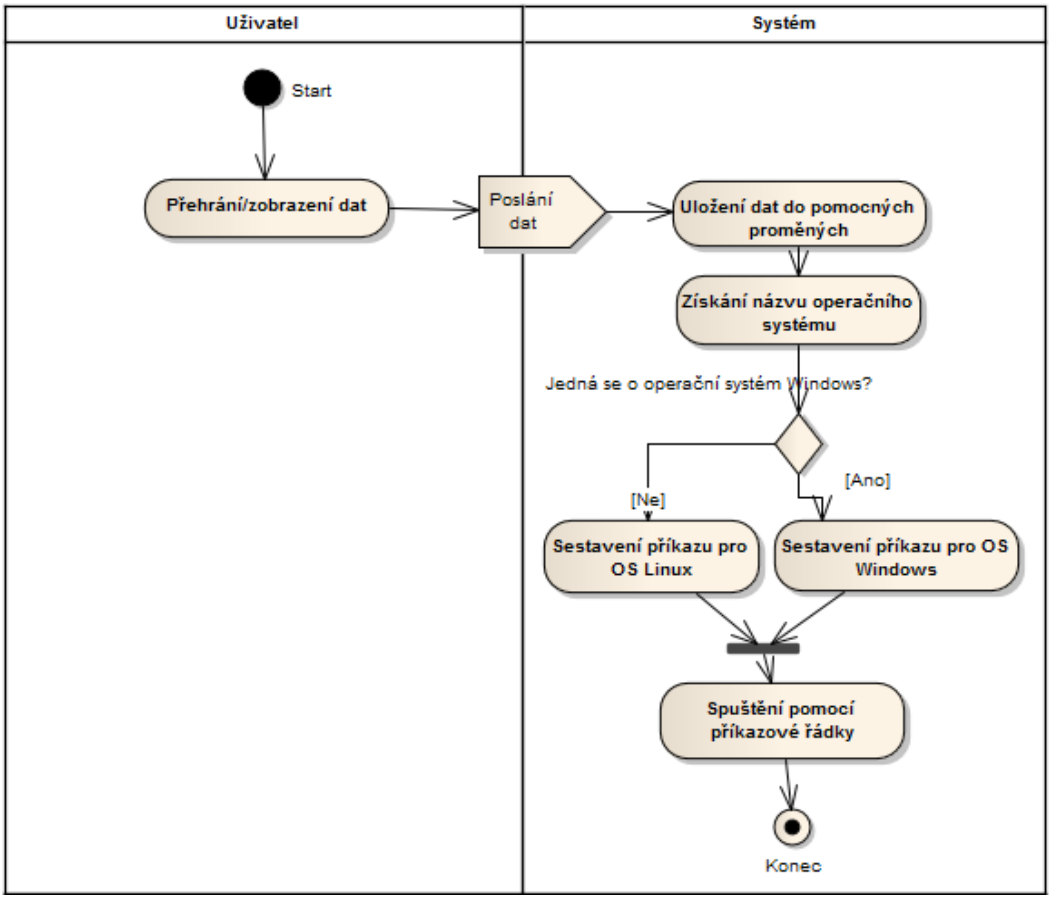
Ilustrace 25: Diagram aktivit podstránky "obecná editace"

Příloha M: Diagram aktivit „odstranění položky“



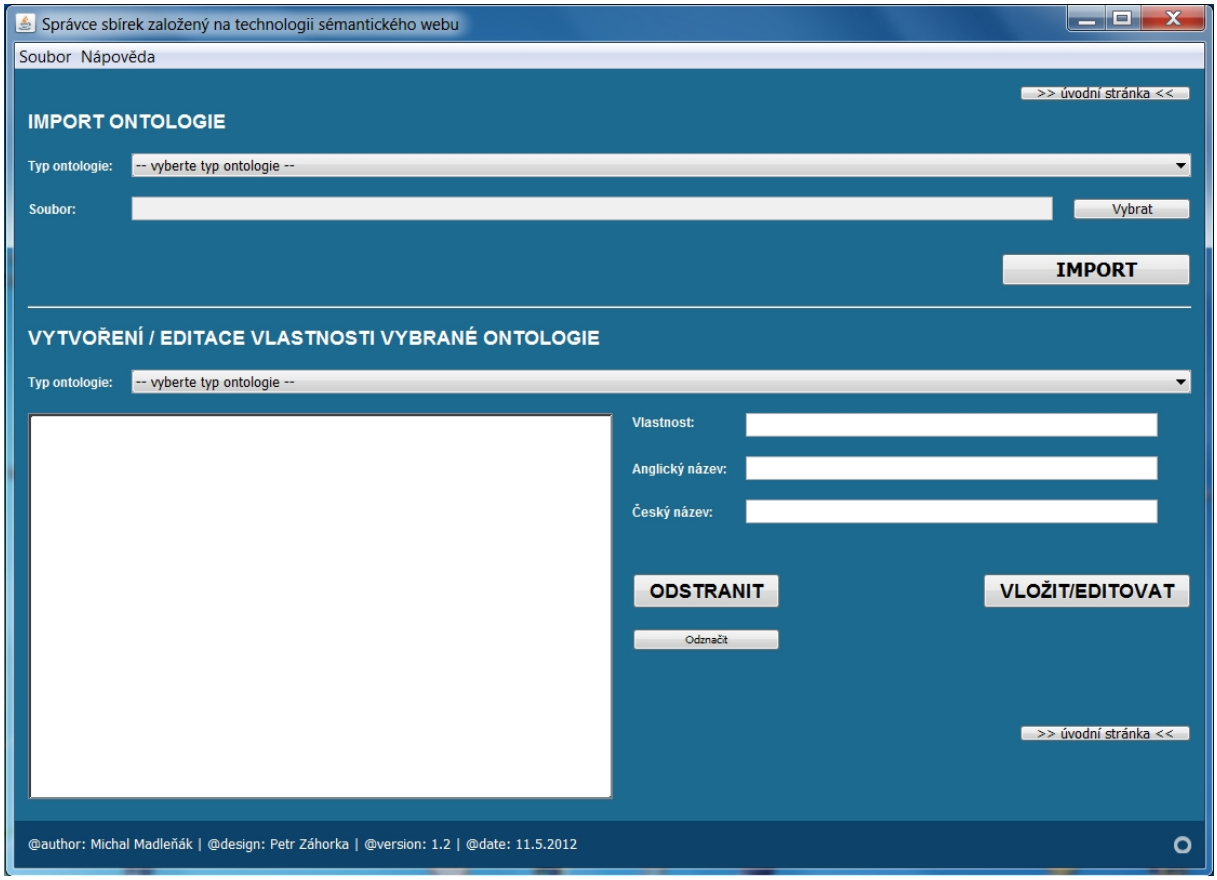
Ilustrace 26: Diagram aktivit podstránky "odstranění položky"

Příloha N: Diagram aktivit „zobrazování položek“

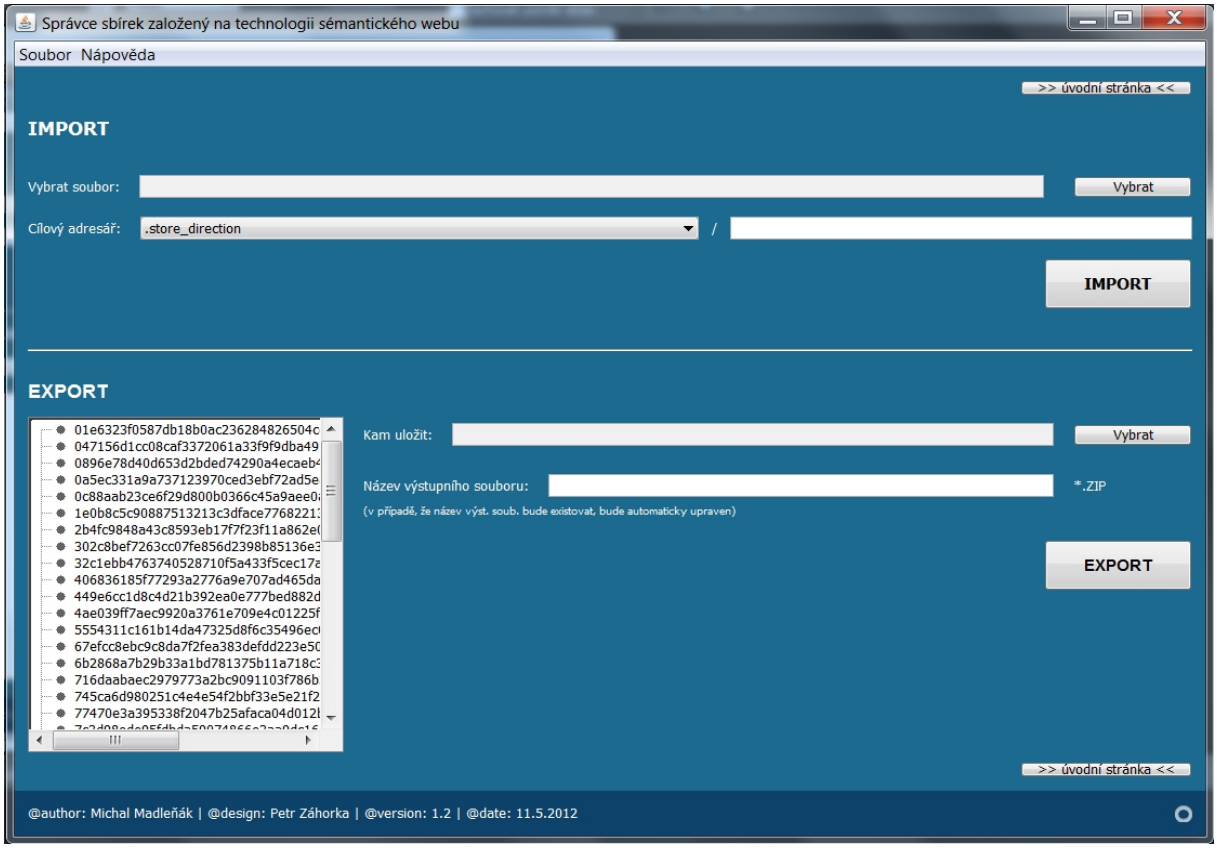


Ilustrace 27: Diagram aktivit podstránky "přehrávání/zobrazování položek"

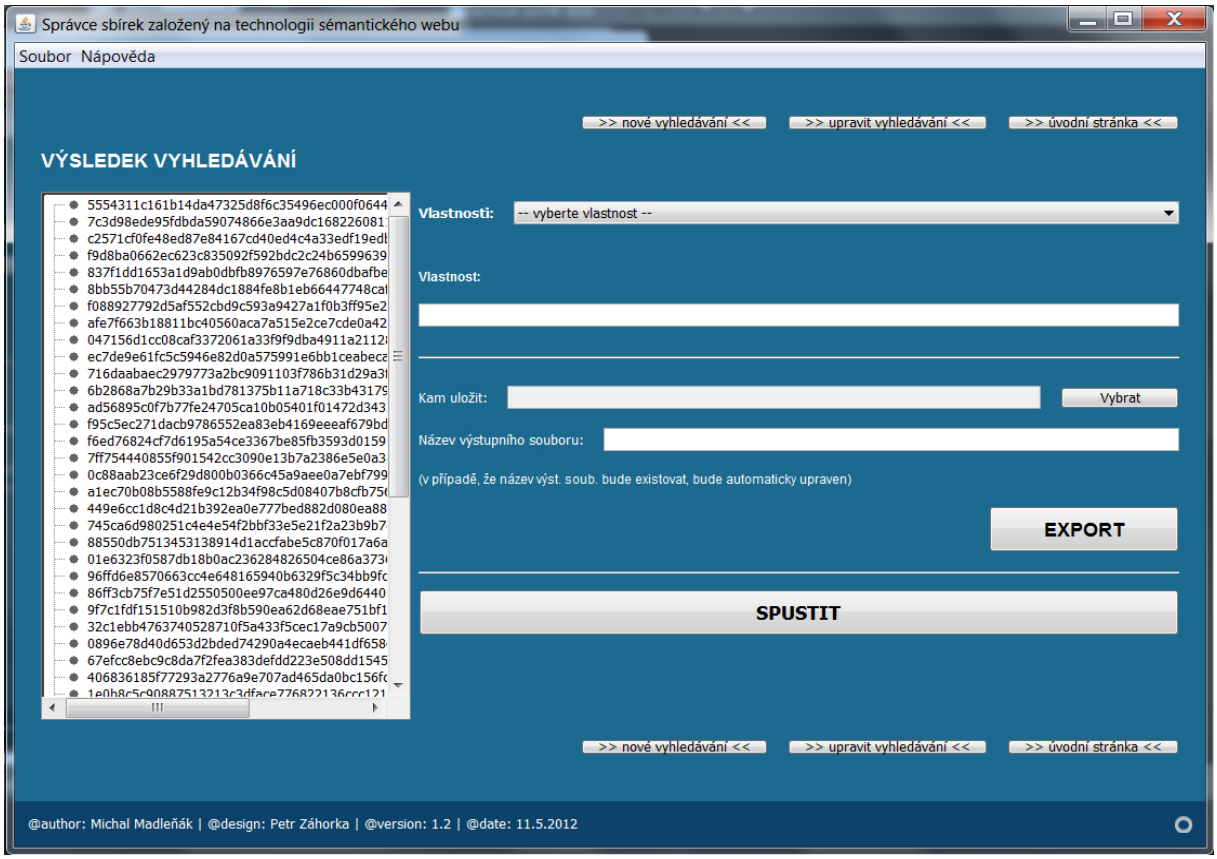
Příloha O: Screenshot několika podstránek výsledné aplikace



Ilustrace 28: Screenshot podstránky "Import ontologií"



Ilustrace 30: Screenshot podstránky "Import/Export dat"



Ilustrace 31: Screenshot podstránky "s výsledkem vyhledávání"

Příloha P: Obsah CD

K diplomové práci je přiloženo CD, na kterém jsou uloženy zdrojové kódy, spustitelný program, testovací data, uživatelská příručka a dokumentace.

CD obsahuje následující adresářovou strukturu:

- **Dokumentace a uživatelská příručka**
  - adresář obsahuje dokumentaci a uživatelskou příručku v elektronické podobě
- **Spustitelný program**
  - adresář obsahuje spustitelný program s knihovnami
- **Testovací soubory**
  - adresář obsahuje testované soubory a ontologie
- **Zdrojové kódy programu**
  - adresář obsahuje zdrojové kódy
- **README.TXT**
  - informační soubor