Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informačních technologií

Webová aplikace pro podporu rozhodování v oblasti bylinné medicíny

Sémantický web

Bakalářská práce

Autor: David Nekolný
Studijní obor: Aplikovaná informatika

Vedoucí práce: Ing. Martina Husáková, Ph.D.

Hradec Králové srpen 2018

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 17.8.2018 David Nekolný

Poděkování:

Děkuji vedoucímu bakalářské práce Ing. Martině Husákové, Ph.D. za metodické vedení práce.

Anotace

Tato práce se věnuje vývoji webové aplikace pro podporu rozhodování v oblasti bylinné medicíny. Aplikace je implementována pomocí webového aplikačního frameworku ASP.NET a odpovídá standardům sémantického webu a propojených dat. První část práce se věnuje popisu základních principů a pojmů technologií sémantického webu, jako jsou RDF, OWL a SPARQL. Druhá část popisuje návrh a implementaci výsledné webové aplikace.

Annotation

Title: Web application for decision making in herbal medicine

This work focuses on the development of a web application for decision support in the field of herbal medicine. The application is implemented using the ASP.NET web application framework and conforms to the semantic web and Linked data standards. The first part deals with description of basic principles and concepts of semantic web technologies such as RDF, OWL and SPARQL. The second part describes the design and implementation of the resulting web application.

Obsah

1 Úvod

Již v roce 2014 počet webových stránek překročil hranici jedné miliardy a toto číslo každým dnem narůstá. Je stále komplikovanější mezi tak velkým počtem nestrukturovaných dat vyhledat konkrétní informace. Tento problém by mohl být vyřešen pomocí konceptu sémantického webu, díky němuž by počítače a lidé dokázali lépe spolupracovat. Sémantický web je v mnoha ohledech považován za web budoucnosti a přetváří dosud nestrukturovaná internetová data do dat strukturovaných pomocí spolupráce několika zásadních technologií.

Cílem této práce je podrobně prozkoumat základní principy a pojmy sémantických technologií. Dále zjistit jaké možnosti tato technologie nabízí pro vývoj webové aplikace zabývající se problematikou bylinné medicíny. Tyto znalosti pak využít k návrhu a implementaci sémantické webové aplikace pro podporu rozhodování v oblasti bylinné medicíny.

Bakalářská práce je rozdělena na část teoretickou a praktickou. V teoretické části práce pojednává o základní myšlence, historii a architektuře sémantického webu a jeho hlavních přínosů pro budoucnost webového vyhledávání. V této části jsou dále popsány jednotlivé technologie jako například RDF, OWL, SPARQL atd. a jejich využití. V závěru teoretické části je vysvětlen princip a myšlenka propojených dat.

Praktická část práce se věnuje vývoji sémantické webové aplikace. V první části jsou nejprve definované specifikace a požadavky aplikace na základě kterých je následně vytvořen návrh ontologie. Poslední část celé práce je zaměřena na popis implementace výsledné aplikace.

2 Sémantický web a jeho technologie

2.1 Sémantický web

Klasické webové stránky jsou pouze souhrn HTML dokumentů, které jsou srozumitelné pouze pro člověka. Web byl navržen jako prostor pro sdílení informací. Problém je ale ten, že informace na webu nejsou strukturované, i když byly odvozeny z dobře strukturované databáze. Stroj vnímá informace pouze jako řetězce znaků, a ne jejich skutečný význam. Tento problém by se dal řešit pomocí stroje s umělou inteligencí, který by však musel porozumět lidské řeči a chování, aby mohl pochopit jaký mají informace na webu význam. Na druhou stranu by se nemuselo zasahovat do struktury běžných webových stránek. Tento problém se však dá řešit i jinak, a to pomocí sémantického webu. Ten se snaží pomocí tzv. metadat popsat informace na webu tak, aby měli jednoznačný význam, a i stroje s menší složitostí tak s nimi mohli efektivně pracovat. Díky strukturovaným datům bude možné lépe využívat data z více zdrojů a docílit tak nových poznatků. Této metodě se říká provázaná data neboli Linked data. (Berners-Lee, 1998a)

Sémantický web s sebou přináší značnou práci, ale ve výsledku také mnoho výhod. Jednou z nich je snazší a individuálnější přístup k informacím pomocí vyhledávacích agentů. Běžné vyhledávání probíhá pouze porovnáváním klíčových slov se slovy obsažených na webové stránce. Tento způsob vyhledávání je neefektivní a často vrací nežádoucí informace. Navíc některé uživatelem požadované informace mohou být separovány ve více dokumentech (Lassila, 1997). Sémantický web může tyto problémy vyřešit. Díky tomu, že data v sémantickém webu jsou strukturována do standardních formátů a jsou strojově srozumitelná, vyhledávač může procházením více strojově čitelných zdrojů vrátit přesně takovou informaci, na kterou se uživatel dotazuje.

2.1.1 Historie

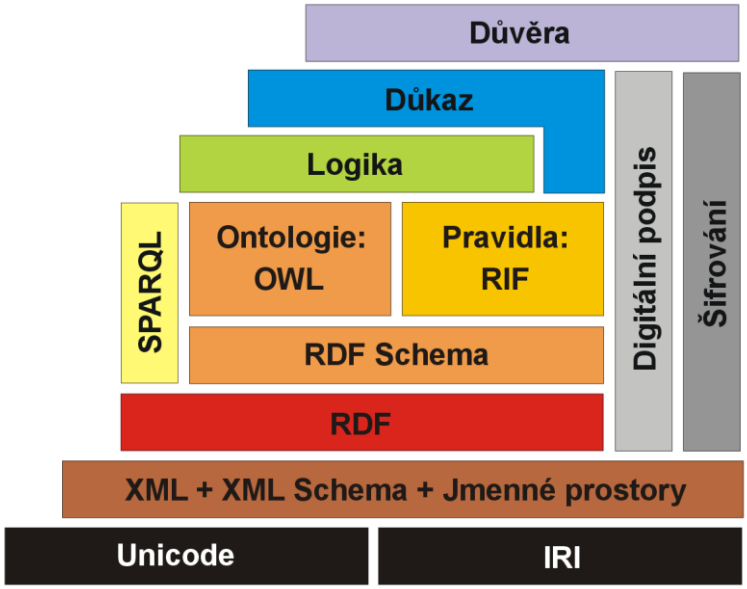
Počátkem veřejného internetu a WWW byla éra Web 1.0. V této době se web skládal ze souboru provázaných statických stránek, na kterých lidé nemohli zveřejňovat žádné informace. Tyto erbové stránky vytvářely pouze odborníci a uživatelé byli pouhými spotřebiteli informací. Web 2.0 sebou přinesl dynamické a interaktivní weby, které mohly vytvářet stránky z více zdrojů. Také byl kladen důraz na vytváření obsahu koncovým uživatelem a začaly vznikat různé blogy, sociální sítě atp. Tento web již obsahoval některé strukturované informace prostřednictvím metadat v hlavičce HTML dokumentu. Tyto metadata obsahovaly základní informace o HTML dokumentu, jako jsou například: autor, klíčová slova, popis stránky, datum poslední úpravy dokumentu a další. Web 3.0 je web budoucnosti, který nebude obsahovat nejen metadata v hlavičce, ale také bude obsahovat metadata o všech informacích v HTML dokumentu. Do Webu 3.0 spadají koncepty jako je sémantický web, linked data, internet of things atp. (Silva, Mahfujur Rahman a El Saddik, 2008, s. 10)

Koncept sémantického webu byl poprvé zveřejněn v roce 2001 v časopise Scientific American. Autorem článku jsou James Hendler, Ora Lassila a zakladatel konsorcia W3C Tim Berners-Lee, který stojí také za vznikem samotného World Wide Web (WWW). V článku autoři uvedli myšlenky a základní principy sémantického webu.

Sémantický web není konkurencí pro Web 2.0. Pouze ho obohatí o metadata, která budou vytvářet a využívat význam informací na stránce.

2.1.2 Architektura

Pro zajištění správného fungování sémantického webu je potřeba dodržet stanovené standardy a architekturu webu. Na obrázku 1 jsou zobrazeny jednotlivé vrtstvy sémantického webu.



Obrázek 1: Architektura sémantického webu, Zdroj: www.ikaros.cz

Unicode a IRI. První vrstva představuje základ všech webových stránek a tím je celosvětový kódovací systém Unicode, který obsahuje přes 120 000 znaků. Na rozdíl od jiných kódovacích systémů obsahuje znaky všech jazyků z celého světa.

Základním principem sémantického webu je to, že absolutně cokoliv na webu by mělo být identifikováno jednoznačným řetězcem znaků IRI (Internationalized Resource Identifier) (Berners-Lee, 1998b). IRI je na rozdíl od URI více zobecněný, a tím je umožněno používat více znaků z Unicode (Cyganiak, Wood a Lanthaler, 2014).

XML, XML Scheme a Jmenné prostory. Díky této vrstvě máme možnost ve vyšších vrstvách používat soubory řídící se syntaxí XML. XML umožňuje uživateli vytvořit libovolnou strukturu dokumentu podle standardu, který je dobře strojově čitelný (Meltzer, 2003, s. 2). Jmenné prostory slouží pro zlepšení čitelnosti a zkrácení XML kódu.

RDF (Resource Description Famework) slouží pro reprezentaci všech dat a vztahů mezi nimi. V RDF jsou informace formulovány do trojic

podmět-přísudek-objekt. Každému zdroji je přiřazeno IRI pro jednoznačnou identifikaci.

RDF Schema, OWL a RIF. Na této vrstvě se definují vztahy a pravidla mezi zdroji. Pomocí jazyka OWL a RDFs lze vytvářet ontologie a RIF nám dává možnost vytvářet pravidla. (Kifer a Boley, 2013)

SPARQL je dotazovací jazyk, syntaxí podobný SQL, s tím rozdílem, že je konstruován pro dotazování se nad RDF daty.

Logika, důkaz a důvěra. Logika slouží k odvozování informací ze schémat a ontologií. Vrstva důvěry obsahuje **šifrování a elektronické podpisy**, tím se stará o zabezpečení a důvěryhodnost dat (Meltzer, 2003, s. 4). Důkaz slouží pro určení pravdivosti informací.

2.2 RDF

Resource Description Framework (RDF) je souborem standardů vyvinutých organizací W3C (World Wide Web Consortium). RDF slouží pro reprezentaci informací na sémantickém webu, které budou dobře čitelné pro lidi, ale také dobře zpracovatelné stroji.

„RDF dokument vytváří tvrzení, že určité věci (lidé, webové stránky nebo cokoliv jiného) mají vlastnosti (jako například: ‚je sestra', ‚je autorem'), s jinými hodnotami (jiná osoba, jiná webová stránka). Tato struktura se ukazuje jako přirozený způsob, jak popisovat drtivou většinu dat zpracovávané stroji." [1] (Berners-Lee, Hendler a Lassila, 2001, s. 2)

Každé takto definované tvrzení se nazývá RDF trojice, která se skládá ze tří prvků, a to z podmětu, přísudku a objektu. Trojice můžeme chápat jako reprezentaci

Obrázek 2: RDF trojice, Zdroj: vlastní tvorba

[1] Původní text: „In RDF, a document makes assertions that particular things (people, Web pages or whatever) have properties (such as ‚is a sister of', ‚is the author of') with certain values (another person, another Web page). This structure turns out to be a natural way to describe the vast majority of the data processed by machines."

jednoduché věty přirozeného jazyka. Příkladem této věty může být: „Karel je kamarád Alice“. Kde „Karel“ je podmět, „je kamarád“ je přísudek a „Alice“ je objekt. Znázornění této RDF trojice můžeme vidět na obrázku 2.

Každý ze tří prvků trojice je identifikován pomocí IRI, až na dvě výjimky, a těmi jsou literály a prázdné uzly, jejichž účel a syntaxe jsou popsány níže. IRI umožňuje jednoznačně identifikovat zdroje, jako jsou například dokumenty, lidé, fyzické objekty nebo abstraktní koncepty. Například IRI pro planetu Zemi z datové sady DBpedia je http://dbpedia.org/page/Earth (Schreiber et al., 2014).

Jednou z mnoha výhod RDF je to, že systémy mohou dělat logické závěry. Pokud existuje přísudek mezi podmětem a objektem, pak danou trojici můžeme označit jako pravdivou. Tímto způsobem mohou systémy za určitých okolností vyvodit, že jiné trojice musí být logicky také pravdivé. Těmto systémům říkáme „odvozovače“ a mohou také někdy odhalit, že RDF trojice se navzájem vylučují. (Schreiber et al., 2014)

2.2.1 Literály

Literály reprezentují základní informace jako jsou například řetězce, data a čísla. Literály nemají IRI. Například na obrázku 3 můžeme vidět celkem čtyři literály a to „32“, „Alice Nováková“, „kiwi@example.org“ a „Kiwi“. Literály jsou spojeny s odpovídajícím datovým typem umožňujícím, aby tyto hodnoty byly správně analyzovány a interpretovány. Řetězcové literály mohou mít přiřazeny značku jazyka, například „en“, „fr“ nebo „cz“ (Schreiber et al., 2014). V RDF trojici se může literál objevit pouze na místě objektu.

2.2.2 Datové typy

Datové typy se používají k určení typu hodnoty literálu, jako jsou například řetězce, čísla a data. V RDF se používají datové typy z *XML Schema,* jako jsou například xsd:integer, xsd:string, xsd:boolean, xsd:date a mnoho dalších. RDF však importuje tři další datové typy, a to rdf:HTML, rdf:PlainLiteral a rdf:XMLLiteral.

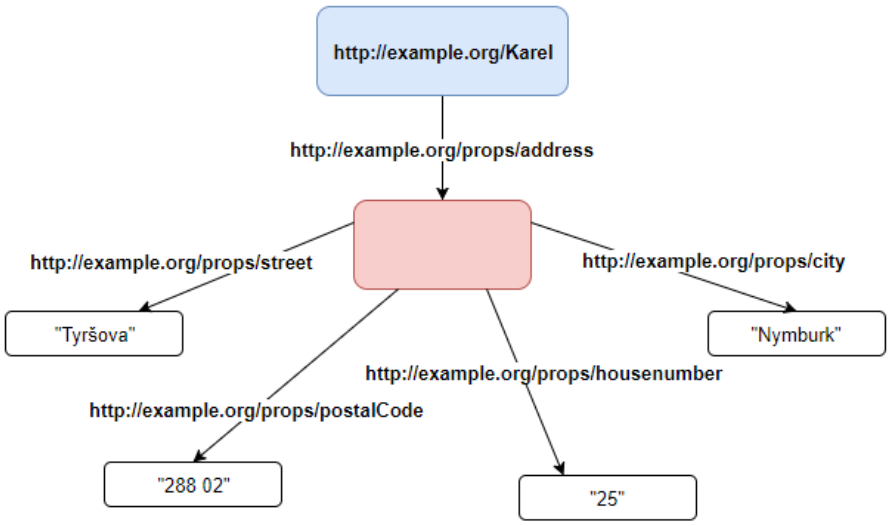
Datový typ se skládá z lexikálního prostoru, hodnotového prostoru a mapováním mezi těmito dvěma prostory. Mapování obsahuje sadu dvojic, jejichž první prvek je z lexikálního prostoru a druhý z hodnotového. Tímto způsobem musí být ke každému prvku lexikálního prostoru přidělen jeden prvek z hodnotového prostoru. Například datový typ xsd:boolean, kde každá hodnota má dva lexikální prvky, by vypadal takto:

- **Lexikální prostor**: {"*true*", "*false*", "1", "0"}
- **Hodnotový prostor**: {**true**, **false**}
- **Mapování:** { <"*true*", **true**>, <"*false*", **false**>, <"1", **true**>, <"0", **false**>, } (Cyganiak, Wood a Lanthaler, 2014)

2.2.3 Prázdné uzly

„Na rozdíl od IRI a literálů, prázdné uzly neidentifikují konkrétní zdroje. Tvrzení obsahující prázdné uzly říkají, že něco s danými vztahy existuje, aniž by to bylo výslovně pojmenováno“ [2] (Cyganiak, Wood a Lanthaler, 2014). Tím, že jsou prázdné uzly nepojmenované, je myšleno, že nejsou identifikovány pomocí IRI, ale v rámci RDF dokumentu mají přiřazeno nějaké lokální jméno. Prázdné uzly se mohou v RDF trojici objevit na místě podmětu nebo objektu.

[2] Původní text: „Unlike IRIs and literals, blank nodes do not identify specific resources. Statements involving blank nodes say that something with the given relationships exists, without explicitly naming it.“



Obrázek 3: Prázdný uzel, Zdroj: vlastní tvorba

Prázdný uzel lze použít například jako adresu, jak můžeme vidět na obrázku 4, na kterou jsou navázány další vlastnosti. Prázdný uzel zde zastupuje jak roli podmětu, tak i objektu a není potřeba aby byl identifikován pomocí IRI. Takto lze také vyjádřit vztah nebo vlastnosti nějaké věci, i když nevíme o jakou věc jde, nebo neznáme její IRI.

2.2.4 Datová sada

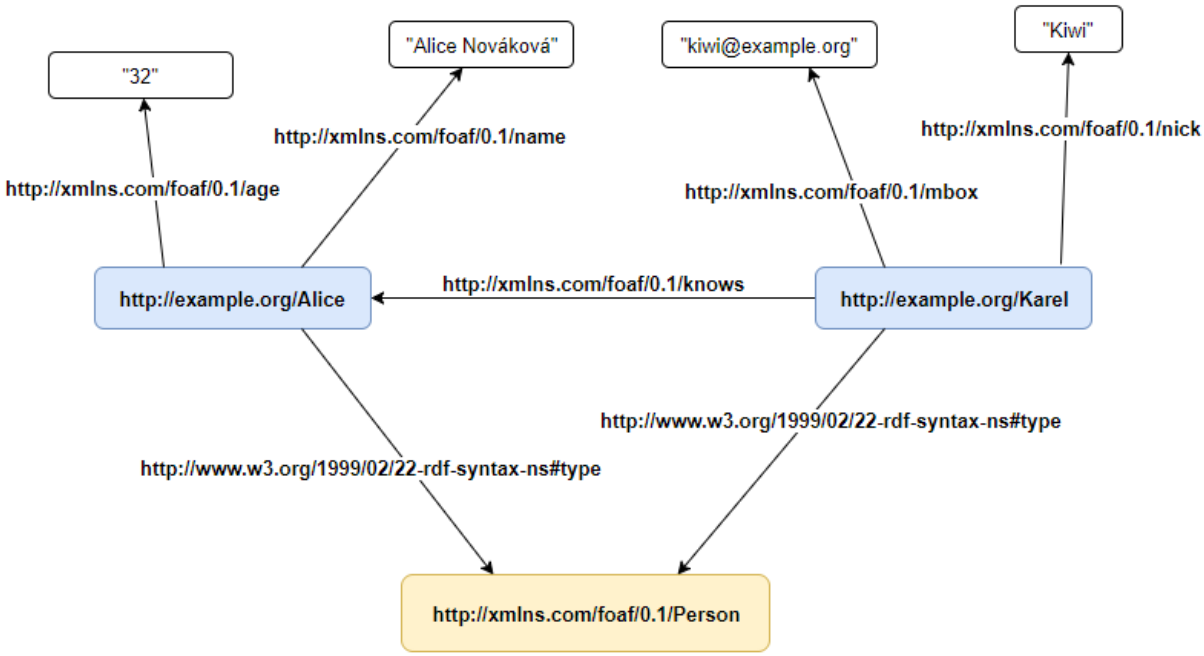
Množinu RDF trojic lze vyobrazit jako orientovaný graf, kterému se říká RDF Graf. Na obrázku 3 je znázorněn jednoduchý RDF Graf, který popisuje základní informace o Karlovi a Alici pomocí známého RDF slovníku FOAF[3] (Friend Of A Friend). Protože RDF grafy jsou pouze sady RDF trojic, pak RDF umožňuje snadnou práci s více grafy, což podporuje propojovat informace z více zdrojů. (Cyganiak, Wood a Lanthaler, 2014)

RDF datová sada je souborem takovýchto grafů. Skládá se přesně z jednoho výchozího grafu, který nemá žádný název a může být prázdný. RDF datová sada může také obsahovat libovolný počet pojmenovaných grafů, které jsou pojmenovány pomocí IRI nebo prázdného uzlu.

[3] http://xmlns.com/foaf/spec/

Velké množství RDF dat je k dispozici jako propojená data. Datové sady jsou publikovány a propojovány na webu pomocí RDF. Mnohé z nich nabízejí dotazovací zařízení pomocí dotazovacího jazyka SPARQL. Příkladem takovýchto veřejných datových sad jsou: (Schreiber et al., 2014)

- DBPedia[4] obsahuje data, která jsou extrahována z tzv. infoboxů na wikipedii.
- VIAF[5] poskytuje údaje o zeměpisných místech, lidech a prací z mnoha národních knihoven a dalších agentur.
- WikiData[6] je bezplatná databáze založená na spolupráci a mnohojazyčnosti.
- Europeana[7] zveřejňuje údaje o kulturních objektech z velkého počtu evropských institucí.
- WordNet[8] je lexikální databáze anglických termínů s řadou sémantických vztahů.



Obrázek 4: RGF graf, Zdroj: vlastní tvorba

[4] https://wiki.dbpedia.org/
[5] https://viaf.org/
[6] https://www.wikidata.org/wiki/Wikidata:Main_Page
[7] https://www.europeana.eu/portal/cs
[8] https://wordnet.princeton.edu/

2.2.5 Syntaxe

Jádrem RDF je model uzlů, atributů a jejich hodnot. K uložení instancí tohoto modelu do souborů, nebo k předání instancí z jedné aplikace do jiné, potřebujeme syntaxi pro serializaci grafu. (Lassila, 1997)

Existuje několik způsobů pro serializaci RDF grafů. Nicméně různé způsoby zápisu stejného grafů vedou přesně na stejnou sadu RDF trojic a jsou tedy logicky ekvivalentní. Mezi tyto formáty patří:

- Turtle jazyky (**N-Triples**, **Turtle**, **TriG** a **N-Quads**)
- **JSON-LD** (syntaxe RDF založena na JSON)
- **RDFa** (pro vkládání RDF do HTML a XML)
- **RDF/XML** (XML syntaxe pro RDF)

V některých z těchto syntaxí je běžné zkrátit IRI, která začínají jmenným prostorem, prefixem. Tímto způsobem můžeme například zkrátit zápis datového typu http://www.w3.org/1999/02/22-rdf-syntax-ns#PlainLiteral na rdf:PlainLiteral. Prefixy slouží pouze pro usnadnění čitelnosti a nemají žádný vliv na sadu RDF trojic definovaných v dokumentu. Jedná se pouze o tzv. syntaktický cukr. V tabulce 1 můžeme vidět čtyři nejběžněji používané prefixy. Tyto zkratky však nejsou platné IRI a nesmějí být použity jako hodnoty vlastností, kde jsou očekávány IRI. (Cyganiak, Wood a Lanthaler, 2014)

Tabulka 1: RDF prefix, Zdroj: přeloženo a upraveno z https://www.w3.org/TR/rdf11-concepts/

Prefix	Jmenný prostor	RDF slovník
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#	RDF slovník
rdfs	http://www.w3.org/2000/01/rdf-schema#	RDF Schema
xsd	http://www.w3.org/2001/XMLSchema#	XML Schema
owl	http://www.w3.org/2002/07/owl#	OWL slovník

2.2.5.1 Turtle

Do rodiny jazyků Turtle se řadí celkem čtyři jazyky a jsou spolu úzce spjaty. N-Triples poskytuje základní syntaxi pro zápis RDF tříd. Turtle obohacuje základní

syntaxi N-Triples o tzv. syntaktický cukr pro zlepšení čitelnosti kódu. TriG a N-Quads přináší možnost pracovat v kódu s více grafy.

N-Triples představuje základní textový způsob pro zakódování RDF dat do dokumentu. Každá řádka se skládá z podmětu, přísudku a objektu a představuje tak jednu RDF trojici. IRI jsou uzavřeny ve špičatých závorkách (<>) a datový typ je připojen k literálům pomocí dvou znaků stříšky (^^). Protože většina literálů má datový typ xsd:string, N-Triples umožňuje tento datový typ vynechat a psát pouze literál. Jazyková zkratka se zobrazí přímo za řetězcem, oddělená symbolem zavináč (@). Tento jazyk se využívá zejména pro výměnu velkých datových sad. V následujícím kódu můžeme vidět jednoduchou ukázku syntaxe N-Triples: (Schreiber et al., 2014)

```
1. <http://example.org/Alice> <http://www.w3.org/1999/02/22-rdf-
   syntax-ns#type> <http://xmlns.com/foaf/0.1/Person>
2. <http://example.org/Alice> <http://xmlns.com/foaf/0.1/name> "Alice
   Nováková"@cs.
3. <http://example.org/Alice> <http://xmlns.com/foaf/0.1/age>
   "32"^^<http://www.w3.org/2001/XMLSchema#int> .
4. <http://example.org/Alice> <http://xmlns.com/foaf/0.1/knows>
   <http://example.org/Karel> .
```

Zdrojový kód 1: Ukázka syntaxe N-Triples, Zdroj: vlastní tvorba

Turtle obohacuje N-Triples o řadu syntaktických zkratek, jako je například podpora prefixů jmenných prostorů, seznamů, prázdných uzlů a zkratek pro datové typy literálů. Dále Turtle umožňuje zkrácený zápis RDF tripletů se stejným podmětem a také zapisování komentářů pomocí křížku (#) na začátku řádky. Tento jazyk poskytuje syntaxi, která je jak dobře čitelná pro stroj, tak i pro člověka. (Schreiber et al., 2014)

V ukázce zdrojového kódu 2 můžeme vidět informace z předchozí ukázky zapsaná pomocí syntaxe Turtle. Na řádce 4 je zobrazen způsob zkrácení kódu provázáním BASE IRI se zkratkou „#me". zkrácené IRI „Alice#me" odpovídá „http://example.org/Alice". Dalším přínosem pro čitelnost kódu je skutečnost, že lze použít písmeno „a", namísto predikátu *rdf:type*, který slouží pro určení typu daného podmětu.

```
1. BASE <http://example.org/>
2. PREFIX foaf: <http://xmlns.com/foaf/0.1/>
3. PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
```

```
<Alice#me> a foaf:Person ;
                    foaf:name "Alice Nováková"@cs ;
                    foaf:age "32"^^xsd:int ;
                    foaf:knows <Karel#me> .
```

Zdrojový kód 2: Ukázka syntaxe Turtle, Zdroj: vlastní tvorba

TriG je dalším rozšířením Turtle rodiny. Tato syntaxe přidala možnost specifikovat více grafů v dokumentu. Protože se jedná o rozšíření jazyka Turtle, tak každý Turtle dokument je zároveň platným TriG dokument. Dnes už se jedná o jeden jazyk, kterému se říká spíše Turtle. (Schreiber et al., 2014)

V ukázce zdrojového kódu 3 můžeme vidět, jak lze definovat jednotlivé grafy. Pokud je nějaká RDF trojice zapsána mimo definici grafu, stává se součástí tzv. nepojmenovaného grafu.

```
BASE <http://example.org/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

GRAPH <http://example.org/graph>
   {
   <Alice#me> a foaf:Person ;
                       foaf:name "Alice Nováková"@cs ;
                       foaf:age "32"^^xsd:int ;
                       foaf:knows <Karel#me> .
   }
```

Zdrojový kód 3: Ukázka syntaxe TriG, Zdroj: vlastní tvorba

N-Quads je značně podobný syntaxi N-Triples s tím rozdílem, že na každé řádce nejsou pouze tři prvky, ale čtyři. Poslední prvek řádku určuje jméno grafu, do kterého daná RDF trojice patří. V následující ukázce lze vidět stejná data jako u předešlé ukázky zapsány pomocí syntaxe N-Quads.

```
<http://example.org/Alice> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://xmlns.com/foaf/0.1/Person> <http://example.org/graph> .
<http://example.org/Alice> <http://xmlns.com/foaf/0.1/name> "Alice Nováková"@cs. <http://example.org/graph> .
<http://example.org/Alice> <http://xmlns.com/foaf/0.1/age> "32"^^<http://www.w3.org/2001/XMLSchema#int> <http://example.org/graph> .
<http://example.org/Alice> <http://xmlns.com/foaf/0.1/knows> <http://example.org/Karel> <http://example.org/graph> .
```

Zdrojový kód 4: Ukázka syntaxe N-Quads, Zdroj: vlastní tvorba

2.2.5.2 RDF/XML

Jak už napovídá název tohoto jazyka, jedná se o vyjádření RDF pojmů v XML syntaxi. K tomu se hojně využívá tzv. XML QNames, který poskytuje definici jmenných prostorů za pomoci prefixů (Beckett a McBride, 2004). Tato syntaxe nám umožňuje serializovat RDF graf jakékoliv složitosti. V následující ukázce jsou znázorněna data z předešlých ukázek za použití základní syntaxe jazyka RDF/XML. V ukázce je vidět přiřazení jazyka pomocí xml:lang nebo datového typu pomocí rdf:datatype.

```
1. <?xml version"1.0"?>
2. <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3.          xmlns:foaf="http://xmlns.com/foaf/0.1/">

4.   <foaf:Person rdf:about="http://example.org/Alice">
5.     <foaf:name xml:lang="cs">Alice Nováková</foaf:name>
6.     <foaf:age rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
7.       32
8.     </foaf:age>
9.     <foaf:knows rdf:resource="http://example.org/Karel" />
10.  </foaf:Person>
11.</rdf:RDF>
```

Zdrojový kód 5: Ukázka syntaxe RDF/XML, Zdroj: vlastní tvorba

Jedním z mnoha rozdílů RDF/XML od ostatních serializačních jazyků je ten, že RDF/XML můžeme přímo vložit do HTML nebo XHTML (dále jen HTML) dokumentu. To však způsobí že dokument neodpovídá definici HTML dokumentu a stává se tak nevalidním. Jediný způsob, jak lze tento problém vyřešit, je vložení odkazu na RDF/XML soubor do hlavičky HTML souboru. Tento způsob lze použít u jakékoliv syntaxe. (Beckett a McBride, 2004)

2.2.5.3 JSON-LD

JSON-LD je jednoduchá a užitečná syntaxe pro serializaci RDF v JSON. Je navržena tak, aby se mohla snadno integrovat do systémů, které již používají JSON a aby existující JSON dokumenty mohly být hladce přetransformovány na JSON-LD s minimálními změnami. JSON-LD poskytuje možnost vyjádření více grafů, definovat datové typy a jazyk řetězců, identifikovat objekty pomocí IRI a mechanismus, díky

kterému může hodnota v objektu JSON odkazovat na objekt z jiného zdroje. (Sporny, et al., 2014)

Základním stavebním kamenem JSON-LD je tzv. kontext, který slouží pro mapování pojmů na IRI. Pojem může být definovaný jako jednoduché IRI, nebo jako JSON objekt. V případě objektu lze specifikovat například typ nebo jazyk řetězce přidružené k vlastnosti. (Sporny et al., 2018)

Kontext může být buď vložený přímo v dokumentu, nebo na něj lze odkázat pomocí odkazu. Díky tomu není potřeba vždy vytvářet nový kontext a lze použít stávající kontext ve více souborech. Odkaz na JSON-LD kontext také lze sdílet mezi různými aplikacemi.

JSON-LD také podporuje tzv. aliasing, díky němu lze libovolně pojmenovat všechna klíčová slova JSON-LD jinými. Například pokud nechceme používat klíčové slovo „@id“, které určuje IRI daného objektu, a místo toho chceme používat slovo „iri“, můžeme jednoduše použít *„"iri": "@id"“*.

```
{
  "@context": {
    "foaf": "http://xmlns.com/foaf/0.1/",
    "xsd": "http://www.w3.org/2001/XMLSchema#",
    "ex": "http://example.org/",

    "name_cs": {
      "@id": "foaf:name",
      "@language": "cs"
    },
    "age": {
      "@id": "foaf:age",
      "@type": "xsd:int"
    },
    "knows": {
      "@id": "foaf:knows",
      "@type": "@id"
    }
  }

  "@id": "ex:Alice",
  "@type": "foaf:Person",
  "name_cs": "Alice Nováková",
  "age": "32",
  "knows": "ex:Karel"
}
```

Zdrojový kód 6: Ukázka syntaxe JSON-LD, Zdroj: vlastní tvorba

V předchozí ukázce zdrojového kódu můžeme vidět základní syntaxi jazyka JSON-LD. V kontextu jsou definovány tři prefixy, které v dokumentu zastupují jmenné prostory. Zvýrazněné pojmy představují určitou vlastnost, kterou lze použít v dokumentu. IRI každé vlastnosti určuje prvek @id a datový typ lze definovat pomocí prvku @type. Dalším důležitým prvkem je @language, který slouží k určení jazyka hodnoty dané vlastnosti.

2.2.5.4 RDFa

Stávající metadata v HTML stránce vám dovolují mluvit o samotném dokumentu, například datum vytvoření, autora nebo popis. Co ale nelze je použití metadat v těle stránky a přidat tak informaci k jiné věci, něž je samotný HTML dokument. RDFa tento problém řeší a umožňuje použití metadat v těle stránky. (Birbeck, 2009a)

RDFa je jedním z několika způsobů, jak obohatit o sémantiku a strukturu přímo data v HTML dokumentu. Tato technologie pouze poskytuje sadu značkovacích atributů, pomocí kterých lze zvýraznit stávající obsah webových stránek, který je čitelný pro lidi a obohatit ho o metadata, aby byl čitelný i pro stroje. Pomocí RDFa můžeme na webové stránce vyjádřit tak jednoduchá data jako je jméno člověka, nebo tak komplexní jako například úplná sociální síť. (Herman et al., 2015)

Google a Yahoo! vyhledávače používají RDFa k obohacení výsledku vyhledávání o některá data jako například hodnocení a počet hodnocení vyhledávané věci.

```
<html xmlns:foaf="http://xmlns.com/foaf/0.1/"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema#">
 <head>
   ...
 </head>
 <body>
   <div typeof="foaf:Person">
     <span property="foaf:name" content="Alice Nováková"
     xml:lang="cs">
       Alice
     </span>
     <span property="foaf:age" datatype="xsd:int">
```

```
13.       32
14.     </span>
15.     <a rel="foaf:knows" href="http://example.org/Karel">
16.       Karel
17.     </a>
18.   </div>
19. </body>
20.</html>
```

Zdrojový kód 7: Ukázka syntaxe RDFa, Zdroj: vlastní tvorba

Ve zdrojovém kódu 7 uvedeném výše lze vidět jednoduchou ukázku RDFa metadat v HTML dokumentu. Data v této ukázce by měla větší informační přínos pro stroj než pro člověka. Člověk by například nevěděl, co znamená číslo 32, ale stroj by pomocí metadat mohl zjistit, že se jedná o věk. Základními atributy, které RDFa používá jsou například *typeof*, který určuje typ daného objektu, *property*, slouží pro definici vlastnosti a *content* určuje hodnotu. Pokud element neobsahuje atribut *content*, pak je hodnota vlastnosti automaticky nastavena na vnitřní text elementu (Birbeck, 2009a). V případě odkazu se hodnota nastavuje pomocí *href*. Pokud je potřeba přiřadit informace o jiném zdroji, pak je nutné použít atribut *about* k nastavení IRI daného zdroje (Birbeck, 2009b). Jazyk literálu se nastavuje pomocí elementu *language*. Občas můžeme také přiřadit k vlastnosti datový typ a k tomu slouží právě atribut *datatype* (Pemberton, 2009).

K RDFa existují dvě alternativy, které jsou založené na stejném principu, a to že vkládají strukturovaná data přímo do HTML dokumentu. Tyto značkovací jazyky se jmenují Mikrodata a Mikroformáty.

Mikrodata jsou založena na třech základních atributech a to *itemscope*, *itemprop* a *itemtype*. Kde *itemscope* znázorňuje novou položku, *itemprop* určuje vlastnost položky a *itemtype* definuje typ položky (Microdata, 2018). Mikrodata mohou využívat libovolné slovníky pro přidání sémantiky do HTML dokumentu.

Na rozdíl od Mikrodat, Mikroformáty nemohou využívat libovolné slovníky a jsou omezeny pouze na definované přímo pro tuto technologii. Zato nabízejí velké množství těchto slovníků a bohatě stačí pro většinu běžně používaných informací na webu. Najdete zde například:

- hCalendar – události,
- hCard – lidé, organizace a kontakty,

- hMedia – mediální informace o obrázcích, videích a zvuků,
- hProduct – produkty,
- hRecipe – recepty na vaření,
- hReview – recenze a hodnocení, a mnoho dalších (Welcome to the microformats wiki!, 2017)

2.3 Ontologie

Slovo ontologie vychází z řečtiny a je složeno ze slov „jsoucno“ a „slovo“. Ve filozofii se jedná o disciplínu, která se zabývá základními otázkami života a podstaty existence. V problematice sémantického webu a umělé inteligence je ontologie, někdy také nazývána RDF slovník, dohodnutý, společný a formální popis pojmů, které jsou v dané problematické doméně důležité.

Ontologie obsahuje taxonomii a sadu odvozovacích pravidel. Taxonomie definuje třídy objektů a vztahů mezi nimi. Tím lze definovat například podtřídy, které mohou dědit vlastnosti předka. Pomocí ontologie můžeme vyjádřit různá pravidla na základě kterých lze odvozovat užitečné závěry. (Berners-Lee, Hendler a Lassila, 2001, s. 2)

Díky ontologiím nemusíme vyvíjet stroje s inteligencí blížící se k té lidské, stačí pouze popsat důležité pojmy dané domény a tím zajistíme, že budou stroje rozumět co který pojem představuje (Smrž a Pitner, 2004, s. 1). Pak lze například docílit toho, že dva stejné výrobky nabízené různými e-shopy, které jsou popsány pomocí stejných tříd jedné ontologie, lze snadno automaticky porovnávat a doporučovat (Svátek, Zamazal a Vacura, 2017, s. 1). Tím ale vzniká problém, jak docílit toho, aby byla použita ta samá ontologie, protože výběr ontologie není jednoduchý proces a teprve v posledních letech vznikají různé metody pro vyhledání ontologií, např. katalog ontologií LOV[9].

V problematice ontologií také vznikla otázka, zda je pro budoucnost přínosnější vytvořit jednu ontologii, která bude obsahovat specifikaci celého vědomí

[9] https://lov.linkeddata.es/dataset/lov/

lidstva, nebo je lepší vytvářet ontologie, které se budou zabývat jen určitou znalostní doménou.

Pro vytváření ontologii existuje velké množství ontologických jazyků. Ontologické jazyky, které se používají pro sémantický web jsou především RDF Schema, OWL (Web Ontology Language) a OWL2.

2.3.1 RDF Schema

RDF Schema (RDFs) je jednoduchý ontologický jazyk patřící do rodiny RDF jazyků, který se používá především pro definování základní struktury ontologie a jednoduchých vztahů mezi zdroji. Ke klasifikaci zdroje RDFs používá pojem „Class" a vztah mezi instancí a jeho třídou je určen vlastností „type". Pomocí RDFs lze vytvořit hierarchii tříd a podtříd a také vlastností a sub-vlastností. Lze také definovat omezení, díky kterým můžeme například určit kterou instanci třídy lze použít v RDF trojici nebo jaký má mít rozsah. (Brickley a Guha, 2014)

Hlavní syntaktické pojmy jazyka RDFs jsou shrnuty v následující tabulce:

Tabulka 2: RDFs syntaktické pojmy, Zdroj: přeloženo a upraveno z https://www.w3.org/TR/rdf11-primer/

Pojem	Syntaxe	Popis
Class (třída)	**C** *rdf:type rdfs:Class*	**C** (zdroj) je RDF třída
Property (třída)	**P** *rdf:type rdf:Property*	**P** (zdroj) je RDF vlastnost
type (vlastnost)	**I** *rdf:type* **C**	**I** (zdroj) je instance **C** (třídy)
subClassOf (vlastnost)	**C1** *rdfs:subClassOf* **C2**	**C1** (třída) je podtřída **C2** (třídy)
subPropertyOf (vlastnost)	**P1** *rdfs:subPropertyOf* **P2**	**P1** (vlastnost) je sub-vlastnost **P2** (vlastnost)
domain (vlastnost)	**P** *rdfs:domain* **C**	Každý zdroj, který má vlastnost **P**, musí být instancí třídy **C**
range (vlastnost)	**P** *rdfs:range* **C**	Hodnota vlastnosti **P**, musí být instancí třídy **C**

V druhém sloupci je syntaxe vyjádřena v prefixové notaci, kde se jako prefix vyskytuje buď *rdf,* nebo *rdfs*. Skutečnost, že tyto základní běžně používané pojmy

mají dva různé prefixy, je poněkud otravný historický artefakt, který se zachovává pouze pro zpětnou kompatibilitu. (Cyganiak, Wood a Lanthaler, 2014)

Příklad použití těchto základních pojmů RDFs se syntaxí Turtle je zobrazen v následujícím kódu. 1-3 řádka definují jmenné prostory, které zajišťují lepší přehlednost kódu. Přísudek *„a“* ukazuje jeden příklad ze syntaktického cukru jazyka Turtle, a zkracuje zápis vlastnosti *rdf:type*. Řádek číslo 4 by se tedy dal zapsat také jako *„ex:Clovek rdf:type rdfs:Class“.* Řádky 5-7 definují vlastnost *ex:potomek* a jeho omezení použití, například že podmět přísudku ex:*potomek* musí být instancí třídy ex:*Clovek* (řádka 6) a objekt musí být také instancí třídy ex:*Clovek* (řádka 7). Další dva řádky deklarují třídy *ex:Muz* a *ex:Zena*, které jsou podtřídami třídy *ex:Clovek*. *ex:Muz* a *ex:Zena* automaticky dědí vlastnosti *ex:Cloveka* a lze tedy použít instance těchto tříd v trojici s přísudkem *ex:potomek*, jak lze vidět na řádce 12.

```
1.  @prefix rdf: < http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2.  @prefix rdfs: < http://www.w3.org/2000/01/rdf-schema#>
3.  @prefix ex: < http://www.example.org/people#>

4.  ex:Clovek a rdfs:Class.
5.       ex:potomek a rdfs:Property;
6.       rdfs:domain ex:Clovek;
7.       rdfs:range ex:Clovek.
8.  ex:Muz rdfs:subClassOf ex:Clovek.
9.  ex:Zena rdfs:subClassOf ex:Clovek.
10. :karel a ex:Muz.
11. :anna a ex:Zena;
12.      ex:potomek :karel.
```

Zdrojový kód 8: Ukázka RDF Schema, Zdroj: vlastní tvorba

2.3.2 OWL

Web Ontology Language 2 (OWL2) je sémantický deklarativní jazyk, který je navržen tak, aby reprezentoval bohaté a komplexní znalosti o věcech, skupinách věcí a vztazích mezi nimi. Na rozdíl od RDFs nabízí mnoho funkcí pro deklaraci složitějších ontologických pravidel a vztahů. Dále umožňuje snadno vyjádřit vztahy mezi různými ontologiemi pomocí standardního rámce anotací. OWL2 je rozšířením staršího jazyka OWL1 a rozšiřuje ho o novou funkcionalitu. Některé nové prvky jsou pouze tzv. syntaktický cukr, např. disjunktivní spojení tříd, zatímco představují novou funkcionalitu, včetně: klíčů, bohatších datových typů, rozsahů dat atd.

Všechny ontologie OWL1 jsou validními ontologie OWL2. (W3C OWL Working Group, 2012)

OWL2 lze zapsat pomocí různých syntaxí, které slouží různým účelům. Primární a jedinou vyžadovanou syntaxí je RDF/XML, která zajišťuje vzájemnou spolupráci mezi všemi validními softwary OWL. Další hojně využívanou syntaxí je OWL2 Manchester, která nabízí lepší čitelnost a používá se především v nástrojích pro editaci ontologií. Další alternativní syntaxe jsou Turtle a OWL/XML. Poslední syntaxe s názvem „OWL2 Structural Specification" lze použít také pro serializaci, ale především slouží ke specifikaci struktury jazyka. (Hitzler et al., 2012)

Příklad OWL2 se syntaxí Turtle je zobrazen ve zdrojovém kódu 9, který znázorňuje deklaraci třídy *f:Child*. Můžeme si povšimnout, že v kódu jsou použity jak rdf tagy, tak i rdfs, owl a owl2. Pro zkrácení kódu na obrázku je vynechána deklarace třídy *f:Person*. Celý kód představuje popis třídy *f:Child*, která podle stanovených pravidel musí být potomek *f:Person* a obsahovat vlastnost *f:hasAge* s číselnou hodnotou menší než 21.

```
1.  @prefix rdf: <http://www.w3.org/1999/02/22/rdf/syntax-ns#> .
2.  @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
3.  @prefix owl: <http://www.w3.org/2002/07/owl#> .
4.  @prefix owl2: <http://www.w3.org/2006/12/owl2#> .
5.  @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
6.  @prefix f: <http://example.org/owl/families#> .

7.  f:Child rdf:type owl:Class .
8.  _:x8 rdf:type owl2:DataRange ;
9.         owl2:onDataRange xsd:integer ;
10.        owl2:minInclusive "21"^^xsd:int .
11. _:x9 rdf:type owl:Restriction ;
12.        owl:onProperty f:hasAge ;
13.        owl:someValuesFrom _:x8 .
14. _:x10 rdf:type owl:Class ;
15.        owl:complementOf _:x9 .
16. _:x11 rdf:type owl:Class ;
17.        owl:intersectionOf (f:Person _:x10) .
18. f:Child owl:equivalentClass _:x11 .
```

Zdrojový kód 9: OWL2 Turtle, Zdroj: upraveno z https://www.w3.org/2007/OWL/wiki/PrimerExampleTurtle

2.3.3 Existující ontologie

Pro správné fungování konceptu sémantického webu je nutné na všech webových aplikacích, které se zabývají stejnou problémovou doménou, používat totožnou ontologii. To nás přivádí k problému, jak vyhledat správnou ontologii, která bude zároveň vyhovovat potřebám aplikace a také bude používána jinými. Dnes pro to existuje značné množství ontologických knihoven, kde každá z nich je zaměřena na určitou problémovou oblast. Mezi ty nejznámější a nejpoužívanější patří:

- BioPortal[10] – obsahuje téměř 700 biomedicínských ontologií. Nejznámější z nich je CPT (Current Procedural Terminology), neboli aktuální procedurální terminologie.
- The OBO Foundary[11] – knihovna se zaměřením na biologické a biomedicínské ontologie.
- oeGov[12] – neboli „Ontologies for e-Government" poskytuje ontologie pro vládní dokumenty.
- OLS[13] – další z řad knihoven, kde lze vyhledat biomedicínské ontologie.
- Swoogle[14] – je sémantický vyhledávač, který lze využít k vyhledávání ontologií.
- LOV[15] – poskytuje přes 600 ontologií různého druhu, mezi ty nejznámější patří například FOAF nebo DC (Dublin Core Ontology).
- LOV4IoT[16] – je jedinečnou knihovnou, která publikuje ontologie zabývající se otázkou „Internet Of Thinks", neboli internet věcí.

[10] https://bioportal.bioontology.org/
[11] http://www.obofoundry.org/
[12] http://www.oegov.us/
[13] https://www.ebi.ac.uk/ols/index
[14] http://swoogle.umbc.edu/2006/
[15] https://lov.linkeddata.es/dataset/lov/
[16] http://sensormeasurement.appspot.com/?p=ontologies

2.4 SPARQL

SPARQL je dotazovací jazyk, který slouží pro dotazování se nad RFD daty. SPARQL je svou syntaxí podobný jazyku SQL a přebírá z něj několik klíčových slov jako například: SELECT, FROM, WHERE, UNION, GROUP BY, HAVING a několik funkcí (SPARQL vs SQL, 2018). Na začátku dotazu, stejně jako u SQL, se používá výraz SELECT, za kterým se píší proměnné, které by se měly objevit ve výsledku. Pak následuje výraz WHERE, který poskytuje základní grafový vzor tvořený vzorovými RDF trojicemi, kde každý podmět, přísudek nebo objekt může být proměnná. Všechny informace, které jsou definovány pomocí příkazu SELECT a zároveň odpovídají grafovému vzoru definovanému výrazem WHERE, jsou navráceny jako výsledky dotazu. Součástí jednoho dotazu může být více grafových vzorů, které se mohou následně spojovat pomocí příkazu UNION (SPARQL Nuts & Bolts, 2018).

Na zdroje můžeme klást určitá omezení pomocí výrazu FILTER, na základě zadané podmínky vyhodnocuje, zda hodnota odpovídá podmínce nebo ne. V podmínce lze použít logické, matematické i srovnávací znaky. Dále SPARQL nabízí několik testovacích funkcí jako *isUri*, *isBlank* a *isLiteral*, které slouží pro zjištění, zda daná hodnota je URI, prázdný uzel anebo literál. Také lze filtrovat hodnoty podle jazyka pomocí *lang* a datového typu pomocí *datatype*. (Feigenbaum, 2009)

Pomocí SPARQL se můžeme například dotazovat i na literály, prázdné uzly nebo IRI. Také lze pomocí výrazu CONSTRUCT vytvořit z výsledku dotazu nový RDF graf. SPARQL umožňuje například: provádět dotazy nad více databázemi, filtrovat literály pomocí regulárních výrazů nebo podle rozsahu hodnot. (Clark, Feigenbaum a Torres, 2008)

Další předností jazyka SPARQL je, že dokáže vytvářet dotazy nad více grafy pomocí klauzule FROM NAMED. Klíčové slovo GRAPH umožňuje, aby některé části dotazu odpovídaly jednomu pojmenovanému grafu a jiné části odpovídali jinému grafu. Cokoliv mimo tuto klauzuli odpovídá výchozímu grafu. (Feigenbaum, 2009)

SPARQL celkem nabízí čtyři typy dotazovacích formulářů, a jsou jimi: SELECT, ASK, DESCRIBE a CONTRUST. Formulář SELECT je nejběžněji používaný a vrací hodnoty zadaných proměnných. Pokud bychom však chtěli zjistit, zda existuje určitý grafový vzor v databázi, pak použijeme formulář ASK (SPARQL Nuts

& Bolts, 2018). Dotaz začínající klíčovým slovem ASK vrací pouze hodnotu „true“ v případě, že hledaný záznam existuje, nebo „false“, v případě že neexistuje. Klauzule DESCRIBE vrací RDF graf, který obsahuje informace o zdroji. Tato data nejsou předepsána jazykem SPARQL, ale jsou určena samotnou databází (Harris a Seaborne, 2013). Namísto toho klauzule CONTRUST vrací RDF graf, který je vytvořen pomocí zadané šablony v dotazu (SPARQL Nuts & Bolts, 2018).

```
1.  PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2.  PREFIX foaf: <http://xmlns.com/foaf/0.1/>
3.  PREFIX ex: <http?//example.org/>

4.  SELECT ?mbox
5.  WHERE {
6.       ?person a foaf:Person;
7.            foaf:mbox ?mbox;
8.            foaf:knows ex:Karel;
9.            foaf:age ?age.
10.      FILTER (?age > 18 && ?age < 40).
11. }
12. ORDER BY ?mbox
13. LIMIT 10 OFFSET 0
```

Zdrojový kód 10: SPARQL ukázka, Zdroj: vlastní tvorba

Ve zdrojovém kódu 10 můžeme vidět základní syntaxi jazyka SPARQL. V první části kódu definujeme jmenné prostory, které se v kódu používají. Dále následuje prvek SELECT, který určuje, jaké hodnoty budeme požadovat ve výsledku. V sekci WHERE lze určit jaké podmínky musí výsledky splňovat. Například prvek FILTER určuje rozsah hodnoty proměnné *?age*. ORDER BY ovlivňuje pořadí výsledku a LIMIT definují maximální počet záznamů ve výsledku. Posledním výrazem je OFFSET, který se používá ve spojení s GRUP BY a LIMIT a určuje kolikátá sada záznamů bude ve výsledku (tzv. paging). (Prud'hommeaux, 2008)

Dotaz ze zdrojového kódu 10 vrací množinu emailů, které patří člověku, co zná Karla a jeho věk se pohybuje od 18 do 40 let. Ve výsledku bude maximálně 10 emailů seřazených podle abecedy.

Jazyk SPARQL však neumožňuje pouze dotazování se nad databází, ale obsahuje také příkazy k přidávání, úpravě či mazání informací v databázi. Základní akce, které slouží k aktualizaci grafů v databázi jsou INSERT a DELETE. Tyto akce se skládají ze skupiny RDF trojic, které mají být buď přidány nebo odstraněny. Dále je

možné pomocí příkazu LOAD, na základě zadaného IRI, které odkazuje na libovolný RDF graf, vložit RDF trojice z odkazovaného grafu do zadaného grafu v úložišti. Odstranění všech dat z jednoho grafu se provádí pomocí akce CLEAR. (Gearon, Passant a Polleres, 2013)

Pro přívětivější přístup k datům se používají tzv. SPARQL koncové body, který umožňuje uživatelům dotazovat se na databázi prostřednictvím jazyka SPARQL. Koncový bod je koncipován jako strojově přívětivé rozhraní k přístupu ke znalostní databázi. Jak formulace dotazů, tak i čitelná prezentace výsledků by měla být zpravidla prováděna pomocí stroje, nikoliv manuálně lidskými uživateli. (SPARQL endpoint, 2011)

SPARQL je však možné použít i k dotazování se nad klasickou relační databází. K tomu slouží mapovací jazyk R2RML, který umožní zobrazit existující relační data v datovém modelu RDF. Vstup do mapování je relační databáze a výstupem je množina RDF dat. Mapovací soubory R2RML jsou samy osobě RDF grafy, které jsou zapsány v syntaxi Turtle. Každé mapování R2RML je uzpůsobeno konkrétní struktuře databáze a cílovému slovníku. R2RML také poskytuje vytvoření virtuálního koncového bodu díky kterému se může uživatel pomocí jazyka SPARQL přímo dotazovat na relační databázi. (Das, Sundara a Cyganiak, 2012)

2.5 Propojená data

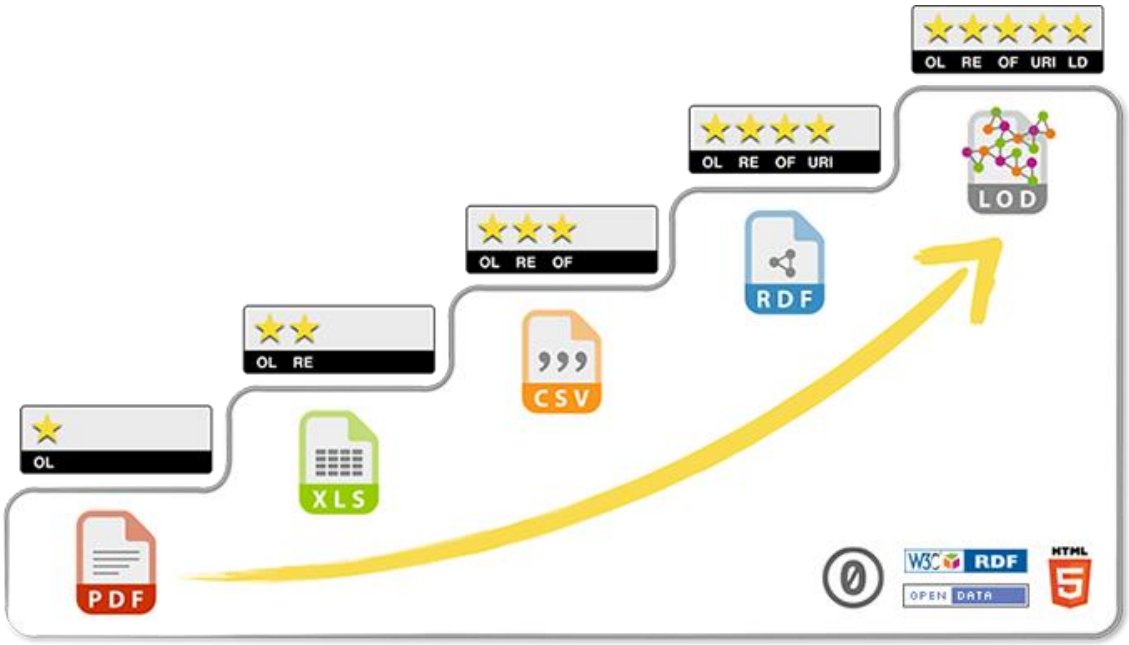
RDF dokumenty sami o sobě nemají takovou sílu a někdy nastává otázka, zda má vůbec cenu se obtěžovat mapovat aplikaci v RDF. Ačkoli tyto data jsou v rámci jedné aplikace omezené a jednoduché, očekává se, že budou kombinovány s daty z jiných aplikací. Tomuto konceptu se říká „Propojená data“ (Linked data) (Berners-Lee, 1998a)

Aby tato myšlenka mohla fungovat, musí sémantické weby mezi sebou vytvářet vazby. Díky tomu osoba nebo stroj může prozkoumat web dat. Pokud webová stránka splňuje pravidla propojených dat, pak může uživatel snadno zjistit další související data pomocí odkazů na ně. (Berners-Lee, 2006)

Propojená data však nejsou vždy přístupná komukoliv. Některá data jsou uzavřena v interních systémech a mají k nim přístup pouze autorizovaní uživatelé.

Ve většině případů se však jedná o data veřejně přístupná s licencí „Open data“. Těmto datům se říká „Propojená otevřená data“ (Linked Open Data).

Tim Bernesr-Lee, iniciátor projektu Linked Data, navrhl způsob ohodnocování dat na webu pomocí tzv. „5-hvězdičkové schéma nasazení“. Tento hvězdičkový systém je kumulativní, což znamená, že každá další hvězdička předpokládá, že data splňují kritéria předchozích kroků. (Hyland et al., 2013)



Obrázek 5: 5-hvězdičkové schéma LOD, Zdroj: https://5stardata.info/en/

Na obrázku výše můžeme vidět způsob ohodnocení dat na webu. Jednou hvězdičkou jsou označena data, která mají libovolný formát a jsou zveřejněna na webu pod otevřenou licencí. Data označená dvěma hvězdičkami jsou strukturována v jednoduché tabulce. Další hvězdička určuje data, která jsou zpřístupněna v neproprietárním otevřeném formátu, například CSV. Data se čtyřmi hvězdičkami musí k identifikaci věcí používat jednoznačné identifikátory URI. A na závěr pětihvězdičková data musí odkazovat na jiná data a stávají se tak oficiálně propojenými otevřenými daty. (Berners-Lee, 2006)

3 Vývoj webové aplikace

Tato kapitola popisuje jednotlivé fáze vývoje webové aplikace pro podporu rozhodování v oblasti bylinné medicíny.

Cílem této bakalářské práce není vytvořit plnohodnotný sémantický web s obrovskou databází léčivých bylinek, nýbrž nastínit možnosti, jak toho lze pomocí sémantických technologií dosáhnout.

3.1 Analýza a návrh

3.1.1 Specifikace a požadavky

Před implementací každé aplikace je přínosné specifikovat základní požadavky, které by měla aplikace splňovat.

Pro běžné uživatele bude tato aplikace sloužit jako běžná informační stránka o bylinné medicíně. Uživatelé budou mít možnost zjistit základní informace o rostlině, její taxonomické zařazení a pozitivní či negativní účinky. Ke každému účinku se může vázat nemoc nebo také příznak na který rostlina působí. Uživatel bude mít přístup k popisu jednotlivých nemocí a příznaků. Součástí webu budou také recepty na domácí rostlinné produkty. Pro přehlednější a pohodlnější pohybování se mezi značným množstvím dat by měla být aplikace obohacena o stránkování a filtraci nad množinou dat. Dále by měl web také obsahovat fulltextové vyhledávání nad všemi daty.

Pro správu informací na webu bude vytvořeno jednoduché CMS, ke kterému bude mít přístup pouze administrátor prostřednictvím přihlašovacího jména a hesla. Administrátor by měl mít možnost přihlašovací jméno a heslo změnit.

Hlavním požadavkem na webovou aplikaci je to, že všechna data o zdrojích musí být strojově čitelná a musí splňovat pravidla propojených otevřených dat. To znamená, že web musí obsahovat otevřenou licenci, veškeré věci by měli být identifikovány pomocí URI a data musí být přístupná ve validním strukturovaném dokumentu RDF.

3.1.1.1 Rozdělení aplikace

Web bude rozdělen na uživatelskou část a administrativní část. Do uživatelské části bude mít přístup jakýkoliv neautorizovaný uživatel a bude sloužit k prezentaci informací. Do administrativní části bude mít přístup pouze autorizovaný uživatel. Tato část bude sloužit k přidávání, mazání a upravování informací prezentovaných na webu. Uživatelská část bude obsahovat celkem čtyři sekce:

- Rostliny – tato část bude obsahovat grafický výpis všech léčivých rostlin. Bude umožňovat filtraci rostlin na základě jména rostliny, binomického jména a pozitivních či negativních účinků. Výběrem jedné rostliny by se měla zobrazit stránka s detailními informacemi o rostlině jako například: popis, výskyt, pěstování, pozitivní a negativní účinky, recepty, taxonomické kategorie a odkazy na stejné zdroje.
- Recepty – sekce by měla prezentovat grafický výpis receptů s možností filtrace podle názvu, účinku a rostliny obsažené v receptu. V detailu receptu bude detailní popis postupu, seznam ingrediencí a výpis účinků obsažených rostlin.
- Nemoci – v této části by měl být jednoduchý výpis nemocí s možností filtrace podle názvu, odborného názvu a příznaků. V detailu nemoci by neměl chybět popis nemoci, výpis příznaků a výpis rostlin, které tuto nemoc léčí.
- Příznaky – Obdobně jako u nemocí, bude tato část obsahovat jednoduchý výpis příznaků. Bude možné je filtrovat podle názvu příznaku, odborného názvu a nemoci. Při otevření detailu příznaku by se měl uživateli zobrazit popis příznaku, odborná jména, rostliny, které tento příznak tlumí a nemoci, které se daným příznakem projevují.

Administrativní část aplikace bude obsahovat CMS, které bude strukturováno podobně jako uživatelská část s tím rozdílem, že informace o věcech nebude pouze prezentovat, nýbrž je bude moci upravovat, přidávat a mazat. Dále bude navíc obsahovat sekci k editaci taxonomických kategorií rostlin. U těchto kategorií se bude zaznamenávat název, taxonomický typ, popis a zařazení do

taxonomické struktury. Administrativní část by měla být graficky rozlišitelná od uživatelské, aby měl uživatel přehled kde se nachází.

3.1.2 Návrh ontologie

Návrh ontologie by měl vycházet ze specifikací a požadavků aplikaci, které jsou popsány víše. Tato aplikace protíná celkem tři problémové domény a to: rostliny a jejich taxonomii, recepty a lidské nemoci.

První problémová oblast by se dala popsat pomocí mezidoménové ontologie DbPedia, která byla ručně vytvořena na základě používaných infoboxů na wikipedii. V současné době tato ontologie zahrnuje 685 tříd a 2795 vlastností. (Ontology, 2018)

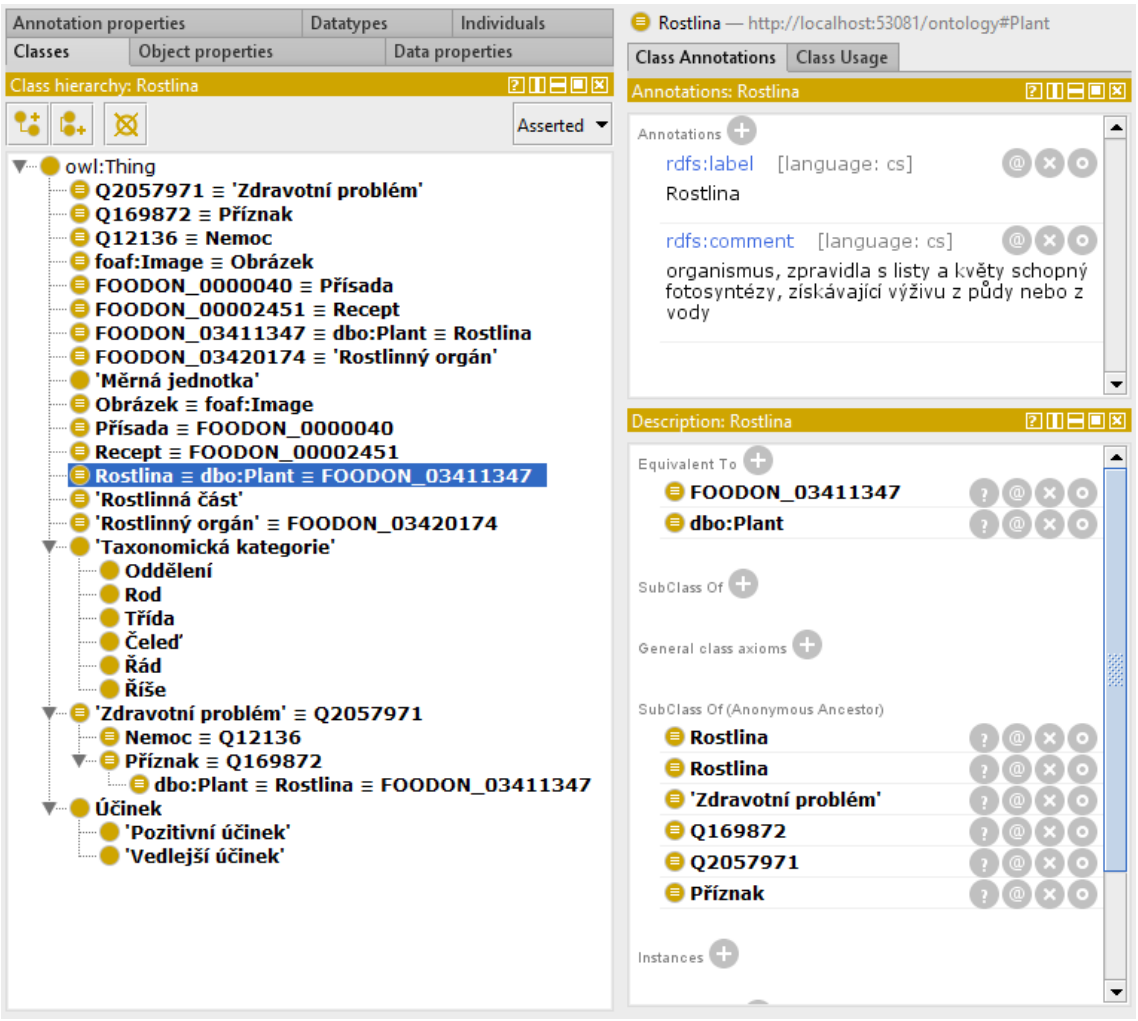
Dále obsahuje taxonomické vlastnosti *dbo:genus*, *dbo:family*, *dbo:order*, *dbo:class*, *dbo:division* a *dbo:kingdom*, které nám dobře poslouží pro zařazení rostlin do taxonomických kategorií. Tato ontologie také nabízí vlastnost *dbo:binomial* pro určení binomického jména rostliny a třídu dbo:Plant, která nám dobře poslouží jako typ rostliny. Velkou výhodou této ontologie je, že je na webu hojně používaná a tím pádem je velmi pravděpodobné, že jí budou používat i internetové vyhledávače. Problémem však je, že DbPedia ontologie nenabízí žádný popis receptu ani nemoci či příznaku.

Ontologie s názvem FoodOn se zabývá právě problémovou doménou kolem jídla a receptů. Tato ontologie nabízí třídu určenou pro recept, část rostliny a rostlinu, jako zdroj jídla. Dále také obsahuje vlastnosti k přiřazení taxonomie nebo ingredience.

Posledním problémem je nalézt ontologii zabývající se zdravotními problémy. Takovýchto ontologií je celá řada. Všechny se zabývají problémem příliš do hloubky, a to pro tuto aplikaci není důležité. Proto pro popis tohoto problému použijeme všeobecnou ontologii s názvem WikiData, která obsahuje třídy jako „health problem“, „symptom“ a „disease“.

Protože žádná z výše uvedených ontologií neposkytuje přesně ty vlastnosti a třídy, které by byly vhodné pro vyvíjenou aplikaci, je potřebné vytvořit ontologii novou. Výhoda ontologií je ta, že lze propojovat již existující ontologie a případně

doplnit potřebné chybějící vlastnosti či třídy. K prohlížení a vytváření ontologií existuje mnoho softwarů jako například: NeOn Toolkit, SWOOP, Neologism, Vitro,



Obrázek 6: Ontologie v Protégé, Zdroj: vlastní tvorba

Knoodl a mnoho dalších. Jedním z nich je také Protégé, který je k dispozici jako open-source software. Tento editor posloužil k vytvoření potřebné ontologie.

Na obrázku 6 je vidět navržená ontologie v programu Protégé. V levé části editoru se nachází hierarchie tříd, ve které na prvním místě můžeme vidět třídu *owl:Thing,* ze které dědí všechny ostatní třídy. Tato třída reprezentuje jakoukoliv věc. Pro přehlednost a intuitivní práci s ontologií jsme znovu vytvořili i třídy, které již existují v jiných ontologiích. Následně jsme naimportovali i již existující třídy a provázali je pomocí vlastnosti *owl:equivalentClass* s odpovídajícími třídami. Obdobný proces jsme provedli s vlastnostmi s tím rozdílem, že nově definované vlastnosti jsme provázali s existujícími pomocí vlastnosti *owl:equivalentProperty*.

Nevýhoda nových tříd a vlastností je ta, že v moment, kdy je ontologie publikována na web nejsou tyto vlastnosti a třídy využívány žádnou webovou stránkou. Stávají se tak pro vyhledávače nezajímavé a nemají takřka žádný smysl. Až postupem času, kdy je lidé začnou používat, se tyto třídy a vlastnosti stávají užitečnými.

3.2 Implementace Aplikace

Tato kapitola se věnuje popisu stěžejních metod, které byly použity při vývoji aplikace. V první části nalezneme popis vybraných technologií, které byly pro vývoj použity. V další části pak nalezneme charakteristiku zdrojového kódu.

3.2.1 Použité technologie

Tato část slouží k popisu několika specifických technologií, které byly použity k vývoji. Vyhneme se popisu všeobecně známých technologií pro vývoj webových aplikací jako jsou HTML, CSS, JavaScript nebo JQuery.

3.2.1.1 ASP.NET MVC

Tento framework nabízí výkonný způsob vytváření dynamických webových stránek. K vývoji v ASP.NET existuje kvalitní nástroj Microsoft Visual Stuido. Framework je založený na architektuře *Model-View-Controler* (MVC), která umožňuje lepší oddělení vnitřní logické vrstvy od prezenční. MVC rozděluje aplikaci na modely (model), pohledy (view) a řadiče (controller). Model se stará o manipulaci a přístup k informacím. Pohled prezentuje uživateli data poskytnutá modelem. Řadič reaguje na uživatelské žádosti, získá potřebné informace z modelu a pomocí pohledu vrátí výsledné informace uživateli.

ASP.NET MVC je založen na .NET knihovně, která poskytuje mnoho přínosných funkcí nejen pro vývoj webových aplikací. Od verze 3.5 je například součástí .NET integrovaný jazyk LINQ, který umožňuje dotazování nad různými typy dat. Nejčastěji se využívá pro dotazování se na seznam objektů, XML nebo

k vytváření SQL dotazů. Od verze 3.5 také lze využívat technologie AJAX, která je popsána níže.

3.2.1.2 Virtuoso

Virtuoso je hybridní databázový server, který podporuje správu dat reprezentovaných jako relační tabulky nebo grafy vlastností. Umožňuje spravovat grafovou databázi, nativní XML databázi, relační databázi a RDF úložiště, které slouží k ukládání RDF trojic. Velkou výhodou systému Virtuoso je, že je kompatibilní s většinou běžně používaných operačních systému jako jsou Windows nebo Linux. Také podporuje širokou škálu programovacích jazyků. Přínosné pro naší aplikaci je skutečnost, že Virtuoso implementuje dotazovací jazyk SPARQL.

3.2.1.3 Semiodesk Trinity

Trinity je platforma pro vývoj aplikací v .NET a Mono, vyvinutá společností Semiodesk. Umožňuje snadnější vývoj sémantických webových aplikací, splňující kritéria propojených dat. Základním kamenem této platformy je knihovna *dotNetRDF*. API také umožňuje přímý přístup pomocí SPARQL koncových bodů, k úložištím propojených otevřených dat jako jsou DBPedia, Freebase, Geonames a mnoho dalších. (Bitbucket, 2018)

Velkou předností této platformy je jednoduché mapování ontologických tříd do C# tříd, přiřazením anotace s identifikátorem ontologické třídy. V následujícím zdrojovém kódu můžeme vidět příklad anotace třídy *Recipe*. Zkratka HRB odkazuje na vytvořenou ontologii, kde se nachází IRI třídy *Recipe*. Klíčové slovo *RdfClass* určuje, že se jedná o RDF třídu, která je typu *HRB.Recipe*. V případě mapování vlastnosti se používá klíčové slovo *RdfProperty*.

```
1. [RdfClass(HRB.Recipe)]
2. public class Recipe : Resource
3. {}
```

Zdrojový kód 11: RdfClass anotace třídy, Zdroj: vlastní tvorba

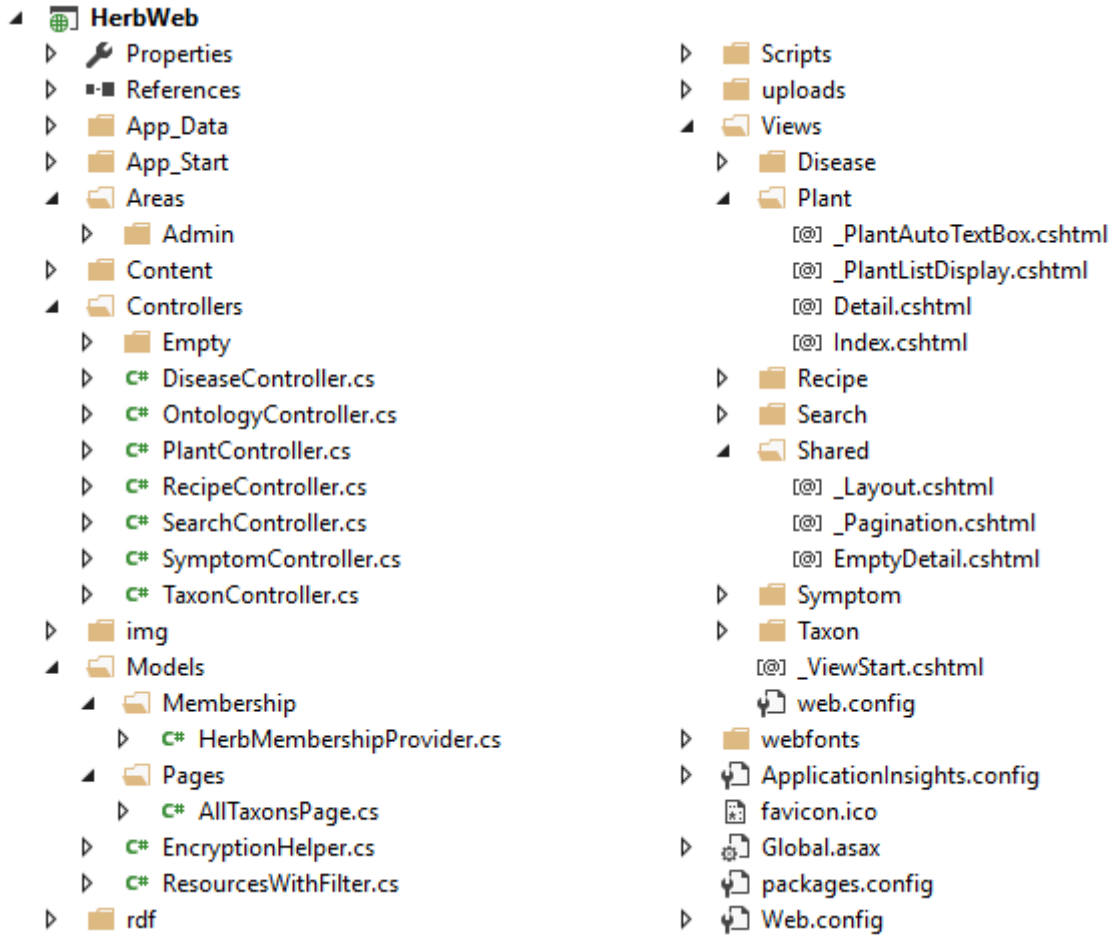
Semiodesk Trinity také obsahuje nástroj pro vytváření SPARQL dotazů bez nutnosti znalosti syntaxe. Tento nástroj sice nelze využít pro vytvoření dotazu libovolné složitosti, ale pro běžné dotazy je dostačující.

3.2.2 Struktura zdrojového kódu

Zdrojový kód je rozdělen na dva .NET projekty: DataAccess a HerbWeb. Knihovna DataAccess je na druhém projektu zcela nezávislá. Zprostředkovává komunikaci s databází a reprezentuje data prostřednictvím tříd. Druhý projekt se stará o kooperaci s uživatelem.

Hlavním úkolem knihovny DataAccess je dotazování se pomocí SPARQL na Virtuoso databázi. Tyto dotazy jsou buď napsány ručně nebo vygenerovány pomocí Trinity. Po odeslání dotazu server vrátí data, která se transformují na základě mapovacích anotací, na vytvořené BLL třídy a pak je lze využívat dále v objektové podobě.

Než však můžeme provést takovýto dotaz, musíme všechny třídy a vlastnosti těchto tříd namapovat na ontologii. Toho lze docílit jednoduše přidáním anotace ke každé vlastnosti a třídě, jak je popsáno v předchozí kapitole.



Obrázek 7: ASP.NET MVC struktura složek, Zdroj: vlastní tvorba

Knihovna DataAcces je záměrně oddělena od hlavního dění, a to především kvůli znuvupoužitelnosti. Pokud by v budoucnu bylo potřeba vyvíjet software, který by pracoval se stejnou databází dat, pak lze tuto knihovnu jednoduše použít znovu.

Na obrázku výše je zobrazeno rozložení souborů a složek v ASP.NET MVC architektuře. Ve složce *Controllers* jsou uloženy řadiče aplikace, které musí mít podle konvence na konci svého názvu slovo „Controller“. Složka *Models* obsahuje třídy, které provádí práci s daty. Tato složka obsahuje pouze zlomek tříd, které jsou součástí modelové vrstvy MVC architektury. Tato složka běžně obsahuje i BLL třídy, které jsme však od tohoto projektu záměrně oddělily. Modely a řadiče by se však neobešly bez pohledů, které jsou definovány ve složce *View*. Každý pohled obsahuje celé, nebo pouze některé části HTML stránek. Tato složka se dělí na podsložky, kde každá z těchto složek je přidělena jednomu z řadičů, které tyto pohledy využívají. Radiče však nejsou omezeny na použití pouze pohledů z jejich složky.

MVC architektura nám umožňuje skládat výslednou HTML stránku z více částí. Těmto částem se říká *„Partial View“*. Tyto části se používají především kvůli zamezení redundanci kódu. Základní část, která slouží jako šablona stránky, se nazýva *„_Layout.cshtml“*. Tento dokument utváří základní vzhled celé stránky. Obsahuje také hlavičku s metadaty. Součástí těchto metadat jsou také odkazy na RDF soubory různých syntaxí, které obsahují strojově čitelné informace o dané rostlině či jiné entitě. Hlavička poskytuje RDF data celkem v šesti syntaxích: RDF/XML, Turtle, N3, CSV, NTriples a Json. O vygenerování těchto souborů se stará řadič *OntologyController*.

Další nedílnou součástí HTML stránek jsou JS (JavaScript) a CSS soubory, které se starají o vzhled a funkčnost stránek na straně klienta. CSS soubory se nachází ve složce *Content* a JS soubory ve složce *Scripts*.

Tehnologie ASP.NET MVC nám umožňuje separovat web na více částí pomocí tzv. oblastí. V našem případě je aplikace rozdělena na část pro běžné uživatele a administrátorskou část. Část pro administrátory se nachází ve složce *Area/Admin* a má do ní přístup pouze autorizovaný uživatel. V této složce se nachází modely, pohledy a řadiče, které tvoří CMS aplikace. O autorizaci se stará řadič *Login-Controller*, který využívá *HerbMembershipProvider* pro kontrolu správnosti přihlašovacích údajů.

```
public class PlantController : Controller
{
    public ActionResult Detail(string name)
    {
        Plant plant = VirtuosoAccess.GetByUri<Plant>(ConstUri.PLANT_URI +
                                                     ResourceBase.ModifyNameToUriName(name));

        return View(plant);
    }
```

Obrázek 8: Ukázka metody Detail, Zdroj: vlastní tvorba

Pokud například bude chtít uživatel zobrazit informace o libovolné rostlině, odešle se na řadič požadavek, který spustí metodu *Detail* zobrazenou na obrázku výše. Jako vstupní parametr metody je řetězec *name*, jehož hodnota je název rostliny. Na základě tohoto názvu se pomocí třídy *VirtuosoAccess*, která slouží pro přístup k datům, vyhledá v databázi rostlina. Ta se pak příkazem „return View(plant)“ vrátí do pohledu, který má stejný název jako je název metody, tedy *Detail*. Tento pohled pak prezentuje data uživateli pomocí značkovacího jazyka Razor, který umožňuje použití C# kódu přímo v pohledu. Příklad jazyka Razor můžeme vidět na obrázku 9, kde je znázorněno, jakým způsobem lze vložit do HTML stránky obrázek, název a binomické jméno rostliny. Razor syntaxe je oddělena od HTML kódu znakem zavináč. Na první řádce pomocí klíčového slova *model*, určujeme, jakého typu budou vstupní data. V tomto případě se jedná o rostlinu, ke které lze přistoupit pomocí vlastnosti *Model*. ASP.NET obsahuje pomocné třídy *Url* a *Html*, které slouží k vygenerování určitých HTML dat. Na obrázku je použita metoda *Url.Content()*, která se stará o přetváření relativní Url adresy na absolutní. V našem případě tato metoda vygeneruje absolutní cestu

k obrázku dané rostliny. Třída *Html* slouží především pro vytvoření nějakého HTML elementu jako jsou: *form, input, a, textarea, label,* a mnoho dalších.

```
@model Plant

@using DataAccess.HerbWeb
@using Semiodesk.Trinity

@if (Model.Image != null)
{
    <img class="w-resp float-sm-right" src="@Url.Content("~/uploads/" + Model.Image)" />
}

<h2 class="mt-2 mt-md-0">@Html.DisplayFor(model => model.Name)</h2>
<span>(@Model.Binomial)</span>
```

Obrázek 9: Ukázka pohledu Detail

3.2.2.1 Dotazování se na databázi

Nedílnou součástí fungování webové aplikace je možnost uchování velkého množství dat. K tomu všeobecně slouží databáze, v našem případě grafová databáze OpenLink Virtuoso, která je popsána výše. Pro pohodlnou práci s těmito daty slouží dotazovací jazyk SPARQL.

Skoro každý požadavek od uživatele vyžaduje nějaké informace z databáze. To způsobuje velmi časté dotazovaní se nad databází, a proto je velmi důležité, aby dotazy byli efektivně napsané. Toho lze docílit použitím třídy *ResourceQuery* z knihovny *Trinity*. Tato třída nám umožňuje pomocí C# metod vygenerovat jednoduché SPARQL dotazy. Bohužel touto třídou nelze generovat komplexnější dotazy, a proto je někdy nutné tvořit SPARQL dotazy ručně. Jednoduchá ukázka generování dotazů pomocí třídy *ResourceQuery* je zobrazena níže.

```
1. ResourceQuery query = new ResourceQuery(hrb.Plant);
2. query.Where(rdfs.label).Contains(EncodeToUTF8("obecná"));
```

Zdrojový kód 12: Příklad použití ResourceQuery, Zdroj: vlastní tvorba

Následně zavoláním metody *ExecuteQuery()* vznikne plnohodnotný SPARQL dotaz, který můžeme vidět ve zdrojovém kódu 13.

```
1. SELECT ?s0 ?p0 ?o0
2. WHERE { ?s0 ?p0 ?o0 .
3. { SELECT DISTINCT ?s0 WHERE
4.   {
5.     ?s0 rdf:type hrb:Plant .
6.     ?s0 rdfs:label ?o00 .
```

```
7.      FILTER ISLITERAL(?o00) .
8.      FILTER REGEX(STR(?o00), "obecnÃ¡", "i") .
9.  }}}
```

Zdrojový kód 13: Ukázka vygenerovaného SPARQL dotazu, Zdroj: vlastní tvorba

Tento dotaz používá klíčové slovo DISTINCT, které slouží k zamezení duplicity výsledných hodnot. Dotaz vrátí všechny vlastnosti a hodnoty všech zdrojů, které jsou typu *hrb:Plant* a jejich název obsahuje slovo „obecná“. Každý řetězec znaků, který slouží k porovnání hodnot v dotazu, musí být převeden do kódování UTF8. V případě, že se hodnota nepřevede a zároveň obsahuje diakritiku, dotaz nevrátí ve výsledku žádná data.

V několika případech bylo však nutné vytvářet SPARQL dotazy ručně, a to především tehdy, když bylo potřeba použít v dotazu klauzuli UNION. V následujícím zdrojovém kódu je zobrazen ručně vytvořený dotaz. Ten slouží k navrácení všech zdrojů, které mají vlastnost *rdf:type,* jednoho z taxonomických typů. Na řádce 12 jsou použity dva příkazy, které se starají o stránkování. Příkaz OFFSET určuje, kolik výsledků se má vynechat a příkaz LIMIT určuje, kolik výsledků bude celkem navráceno. Klauzule ORDER BY seřadí výsledky abecedně podle názvu.

```
1.  SELECT ?a WHERE
2.  {
3.    { ?a rdf:type hrb:Kingdom } UNION
4.    { ?a rdf:type hrb:Division } UNION
5.    { ?a rdf:type hrb:Class } UNION
6.    { ?a rdf:type hrb:Order } UNION
7.    { ?a rdf:type hrb:Family } UNION
8.    { ?a rdf:type hrb:Genus }
9.    ?a rdfs:label ?name.
10. }
11. ORDER BY ?name
12. OFFSET 0 LIMIT 20
```

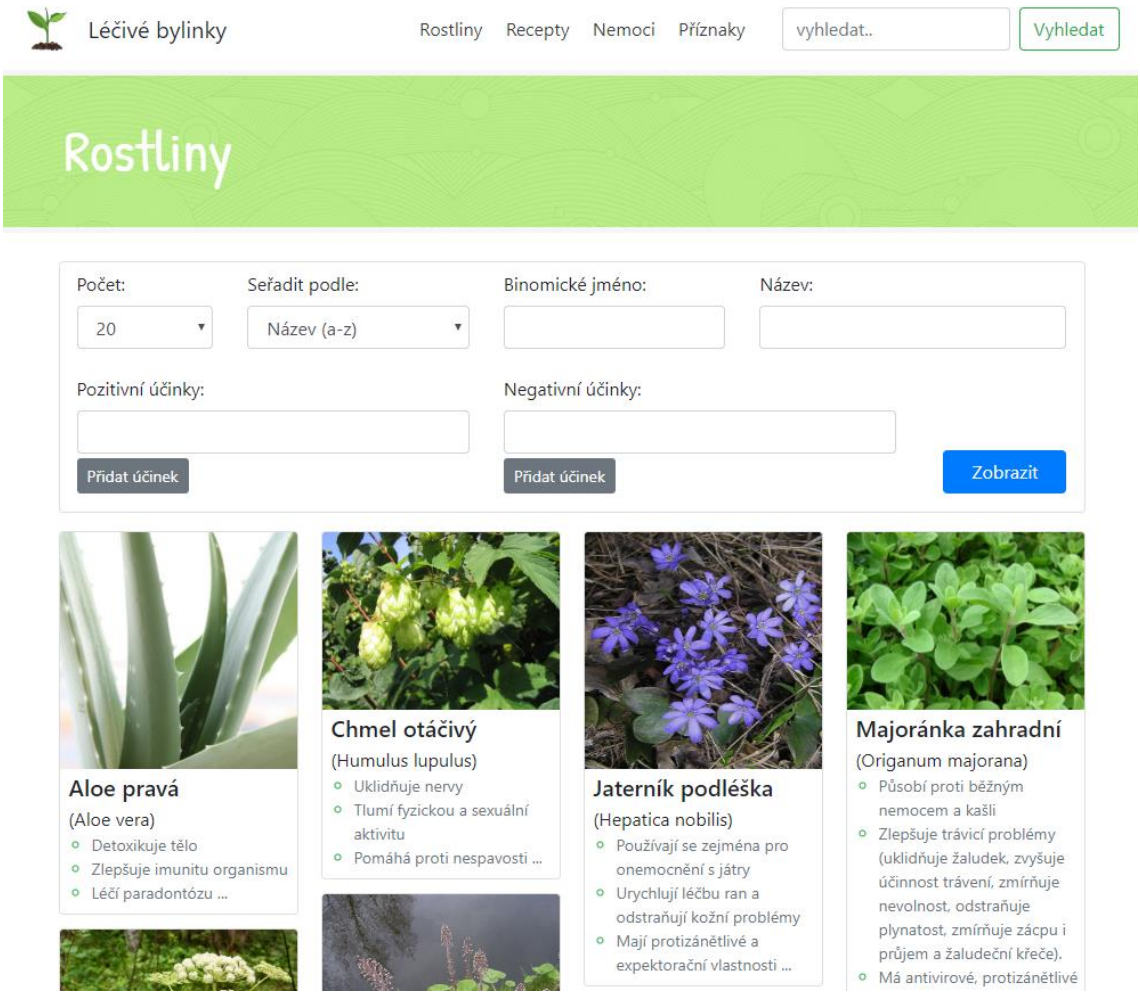
Zdrojový kód 14: Ukázka SPARQL dotazu, Zdroj: vlastní tvorba

3.2.3 Uživatelské rozhraní

Uživatelské rozhraní (UI) slouží především pro kooperaci uživatele s aplikací a pro prezentaci informací. Jak již bylo uvedeno výše, tato aplikace se skládá z uživatelské a administrativní části. Každá tato část je záměrně vizuálně odlišná, aby měl uživatel přehled, v jaké části se nachází. Kvůli usnadnění stylizace stránek

jsme použili knihovnu Bootstrap. Tato knihovna nabízí velké množství responzivních komponentů postavených na HTML, CSS a JS.

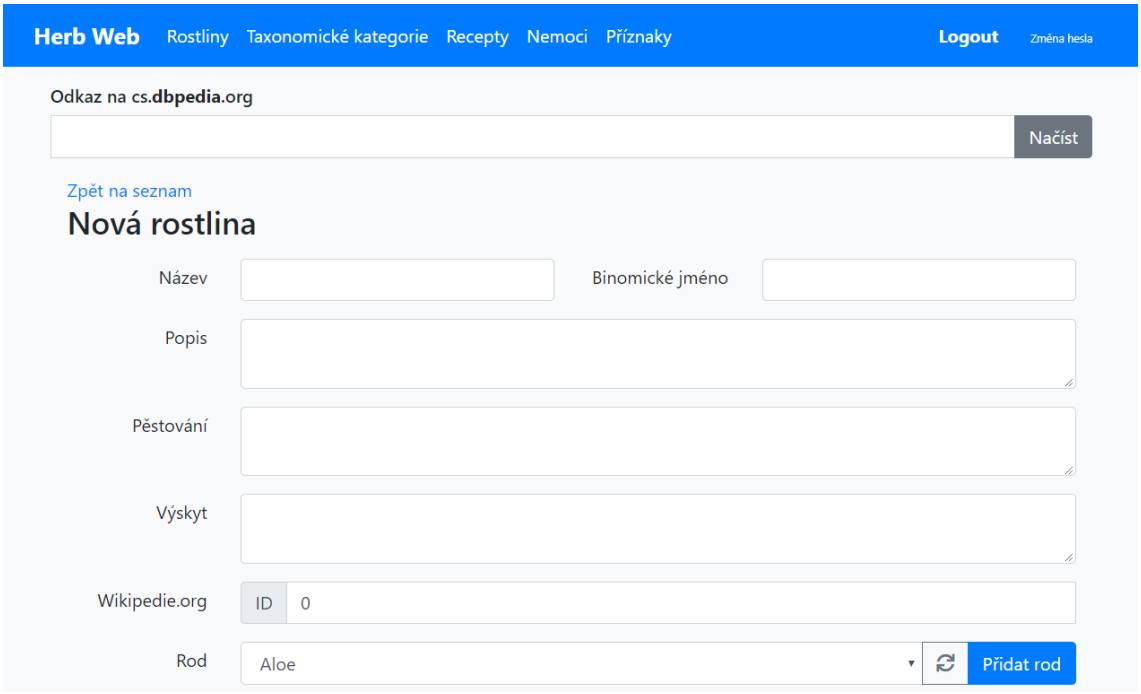
Příklad uživatelského UI je zobrazeno na obrázku 10, který slouží pro výpis a vyhledávání rostlin. K více specifickému vyhledávání slouží filtrace, která se nachází těsně nad výpisem seznamu rostlin. Díky této filtraci lze vyhledávat rostliny podle názvu, binomického jména, pozitivního či negativního účinku. V horní části stránky se nachází navigační menu, díky kterému lze přepínat mezi stránkami. Vedle navigačního menu se nachází jednoduchý vyhledávací formulář, který provádí fulltextové vyhledávání nad všemi daty.



Obrázek 10: UI přehled rostlin, Zdroj: vlastní tvorba

Na obrázku 11 je ukázka administrativního UI. Na první pohled si můžeme všimnout, že toto UI má jednodušší design a v navigačním menu je o jednu položku více. Ukázka na obrázku obsahuje formulář k vytvoření nové rostliny. Za zmínku zde

stojí první formulářové pole s označením „Odkaz na cs.dbpedia.org“. Toto pole očekává IRI rostliny nacházející se na doméně české DbPedie. Díky tomuto IRI aplikace vyhledá rostlinu na DbPedii a může tak vyplnit některá textová pole jako například: název, binomické jméno, popis, wikiID a rod. Aplikace však na základě tohoto IRI také v pozadí generuje taxonomické kategorie této rostliny.



Obrázek 11: UI vytvoření rostliny, Zdroj: vlastní tvorba

4 Shrnutí výsledků

Cílem bakalářské práce bylo prozkoumat technologie sémantického webu a zjistit jaké možnosti nabízí pro podporu rozhodování v oblasti bylinné medicíny. Na základě těchto informací dále vytvořit webovou aplikaci, která nastíní možnosti použití sémantických technologií v praxi.

Po prozkoumání těchto technologií lze říci, že sémantický web není omezený jen na některé problémové domény. Naopak je otevřený všem novým znalostem spadajících do libovolné oblasti, a to prostřednictvím vytváření nových ontologií. Nové ontologie se však potýkají s problémem nedostatku důvěry, bez které postrádají tyto ontologie smysl. Pro zvětšení důvěry ontologie je potřeba aby tuto ontologii využívalo více aplikací. Při vývoji aplikací se však spíše přiklání k použití důvěrných ontologií a tím se tento problém stává cyklickým.

Tento problém by bylo v našem případě možné obejít, kdyby se nám podařilo vyhledat ontologii, která by odpovídala potřebám aplikace. Bohužel se nám tuto ontologii nepodařilo nalézt. Následně však sloučením vlastností a tříd z více existujících důvěrných ontologií mohla vzniknout výsledná ontologie.

Výsledná aplikace prokazuje že vývoj sémantické webové aplikace není o tolik komplexnější než vývoj klasické webové aplikace. Je potřeba porozumět několika zásadním technologiím a případně se naučit pracovat s novými nástroji a knihovnami. Výsledné stránky jsou laickými uživateli nerozeznatelné od klasických dynamických webových stránek, ale přináší s sebou řadu výhod.

5 Závěry a doporučení

Sémantický web se dá považovat za průkopníka ve vývoji moderních webových stránek, a to především proto, že umožňuje rozšířit stávající dokumenty na webu o strojově čitelné informace. Usnadňuje tak například práci vyhledávačům, které do nedávna nedokázaly porozumět významu informací na webu.

Stěžejní technologií sémantického webu je RDF, který představuje standardizovaný způsob zapisování strojově čitelných dat. Jednou z výhod RDF je to, že systémy procházející tyto soubory mohou odvozovat nové logické závěry. Další důležitou technologií je dotazovací jazyk SPARQL. Ten umožňuje efektivní práci s RDF databázemi.

Sémantický web je podle dosažených poznatků web, který stojí za to dále vyvíjet a propagovat. Na jednu stranu je značně technologicky a implementačně náročný, na druhou stranu však přináší do světa internetu mnoho podstatných výhod a pokud je správně navržen, může napomáhat vývoji lidského poznání jako celku.

Z počátku měl sémantický web řadu nevyřešených problémů. Postupem času se spousta nedostatků vyřešila, přesto ale některé stále zůstávají. Jeden z mnoha příkladů těchto nedostatků může být otázka, zda lidé dokáží pomocí ontologií popsat všechny problémové domény, nebo je potřeba vymyslet a vyvinout nějakou obecnou ontologii, která bude popisovat všechny lidské znalosti.

Cílem práce bylo vytvořit sémantický web, který napomáhá při rozhodování v oblasti bylinné medicíny. Dle zadaného cíle byla vytvořena aplikace, která obsahuje strojově čitelná data v dané oblasti. Aplikace je příkladem toho, jakým způsobem lze vytvářet sémantické webové stránky.

Dle zadaného byla navrhnuta a implementována aplikace, která obsahuje strojově čitelná data v dané oblasti. Nejprve bylo nutné určit specifikace a požadavky na aplikaci na základě kterých bylo možné vytvořit ontologii. Poté autor práce implementoval aplikaci pomocí moderního frameworku ASP.NET MVC.

Přínosem pro budoucí chod aplikace by bylo vhodné tuto aplikaci obohatit o více dat. Popřípadě rozšířit použitou ontologii více do hloubky.

6 Seznam použité literatury

[1] BECKETT, Dave; McBRIDE, Brian. RDF/XML Syntax Specification (Revised). World Wide Web Consortium (W3C) [online]. 2004 [cit. 16.08.2018]. Dostupné z: http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/

[2] BERNERS-LEE, Tim; HENDLER, James; LASSILA, Ora. The semantic web. Scientific american, 2001, 284.5: 28-37.

[3] BERNERS-LEE, Tim. Linked Data. World Wide Web Consortium (W3C) [online]. 2006 [cit. 16.08.2018]. Dostupné z: https://www.w3.org/DesignIssues/LinkedData.html

[4] BERNERS-LEE, Tim. Semantic Web Road map. World Wide Web Consortium (W3C) [online]. 1998a [cit. 16.08.2018]. Dostupné z: https://www.w3.org/DesignIssues/Semantic.html

[5] BERNERS-LEE, Tim. Web Architecture from 50,000 feet. World Wide Web Consortium (W3C) [online]. 1998b [cit. 16.08.2018]. Dostupné z: https://www.w3.org/DesignIssues/Architecture.html

[6] BIRBECK, Mark. Introduction to RDFa. A List Apart [online]. 2009a [cit. 16.08.2018]. Dostupné z: https://alistapart.com/article/introduction-to-rdfa

[7] BIRBECK, Mark. Introduction to RDFa II. A List Apart [online]. 2009b [cit. 16.08.2018]. Dostupné z: https://alistapart.com/article/introduction-to-rdfa-ii

[8] BRICKLEY, Dan; GUHA, V. Ramanatha. RDF Schema 1.1. World Wide Web Consortium (W3C) [online]. 2014 [cit. 18.01.2018]. Dostupné z: https://www.w3.org/TR/rdf-schema/

[9] CLARK, Grant Kendall; FEIGENBAUM, Lee; TORRES, Elias. SPARQL Protocol for RDF. World Wide Web Consortium (W3C) [online]. 2008 [cit. 25.01.2018]. Dostupné z: https://www.w3.org/TR/rdf-sparql-protocol/

[10] CYGANIAK, Richard; WOOD, David; LANTHALER, Markus. RDF 1.1 Concepts and Abstract Syntax. World Wide Web Consortium (W3C) [online]. 2014 [cit. 20.01.2018]. Dostupné z: http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/

[11] DAS, Souripriya; SUNDARA Seema; CYGANIAK Richard. R2RML: RDB to RDF Mapping Language. World Wide Web Consortium (W3C) [online]. 2012 [cit. 16.08.2018]. Dostupné z: http://www.w3.org/TR/2012/REC-r2rml-20120927/

[12] FEIGENBAUM, Lee. SPARQL By Example. World Wide Web Consortium (W3C) [online]. 2009 [cit. 16.08.2018]. Dostupné z: https://www.w3.org/2009/Talks/0615-qbe/

[13] GEARON, Paula; PASSANT, Alexandre; POLLERES Axel. SPARQL 1.1 Update. World Wide Web Consortium (W3C) [online]. 2013 [cit. 16.08.2018]. Dostupné z: http://www.w3.org/TR/2013/REC-sparql11-update-20130321/

[14] HARRIS, Steve; SEABORNE, Andy. SPARQL 1.1 Query Language. World Wide Web Consortium (W3C) [online]. 2013 [cit. 16.08.2018]. Dostupné z: http://www.w3.org/TR/2013/REC-sparql11-query-20130321/

[15] HERMAN, Ivan et al. RDFa 1.1 Primer - Third Edition: Rich Structured Data Markup for Web Documents. World Wide Web Consortium (W3C) [online]. 2015 [cit. 16.08.2018]. Dostupné z: http://www.w3.org/TR/2015/NOTE-rdfa-primer-20150317/

[16] HITZLER, Pascal et al. OWL 2 Web Ontology Language Primer (Second Edition). World Wide Web Consortium (W3C) [online]. 2012 [cit. 24.01.2018]. Dostupné z: https://www.w3.org/TR/owl2-primer/

[17] HYLAND, Bernadette et al. Linked Data Glossary. World Wide Web Consortium (W3C) [online]. 2013 [cit. 16.08.2018]. Dostupné z: http://www.w3.org/TR/2013/NOTE-ld-glossary-20130627/

[18] KIFER, Michael; BOLEY Horald. RIF Overview (Second Edition). World Wide Web Consortium (W3C) [online]. 2013 [cit. 17.01.2018]. Dostupné z: https://www.w3.org/TR/2013/NOTE-rif-overview-20130205/

[19] LASSILA, Ora. Introduction to RDF Metadata. World Wide Web Consortium (W3C) [online]. 1997 [cit. 16.08.2018]. Dostupné z: https://www.w3.org/TR/NOTE-rdf-simple-intro

[20] MELTZER, Cindy. Semantic Web: the devil is in the details. SA Journal of Information Management, 2003, 5.2.

[21] PEMBERTON, Steven. RDFa for HTML Authors,. Worl Wide Web Consortium (W3C) [online]. 2009 [cit. 16.08.2018]. Dostupné z: https://www.w3.org/MarkUp/2009/rdfa-for-html-authors#Using

[22] PRUD'HOMMEAUX, Erik et al. SPARQL Query Language for RDF. World Wide Web Consortium (W3C) [online]. 2008 [cit. 25.01.2018]. Dostupné z: https://www.w3.org/TR/rdf-sparql-query/

[23] SCHREIBER, Guus et al. RDF 1.1 Primer. World Wide Web Consortium (W3C) [online]. 2014 [cit. 19.01.2018]. Dostupné z: https://www.w3.org/TR/rdf11-primer/

[24] SILVA, Juan M.; MAHFUJUR RAHMAN, Abu Saleh Md; EL SADDIK, Abdulmotaleb. Web 3.0: a vision for bridging the gap between real and virtual. In: Proceedings of the 1st ACM international workshop on Communicability design and evaluation in cultural and ecological multimedia system. ACM, 2008. p. 9-14.

[25] SMRŽ, Pavel; PITNER, Tomáš. Sémantický web a jeho technologie (3). Zpravodaj ÚVT MU, 2004, 14-16.

[26] SPORNY, Manu et al. A JSON-based Serialization for Linked Data. World Wide Web Consortium (W3C) [online]. 2014 [cit. 16.08.2018]. Dostupné z: http://www.w3.org/TR/2014/REC-json-ld-20140116/

[27] SPORNY, Manu et al. A JSON-based Serialization for Linked Data. World Wide Web Consortium (W3C) [online]. 2018 [cit. 16.08.2018]. Dostupné z: https://json-ld.org/spec/latest/json-ld/

[28] SVÁTEK, Vojtěch; ZAMAZAL, Ondřej; VACURA, Miroslav. Fokusovaná kategorizační síla webových ontologií. 2017.

[29] Bitbucket. The Git solution for professional teams [online]. 2018 [cit. 16.08.2018]. Dostupné z: https://bitbucket.org/semiodesk/trinity

[30] Microdata. Web Hypertext Application Technology Working Group (WHATWG) [online]. 2018 [cit. 16.08.2018]. Dostupné z: https://html.spec.whatwg.org/multipage/microdata.html#items

[31] Ontology. DBpedia [online]. 2018 [cit. 16.08.2018]. Dostupné z: https://wiki.dbpedia.org/services-resources/ontology

[32] SPARQL Nuts & Bolts. The Smart Data Company® [online]. 2018 [cit. 16.08.2018]. Dostupné z: https://www.cambridgesemantics.com/blog/semantic-university/learn-sparql/sparql-nuts-bolts/

[33] SPARQL endpoint. Semanticweb.org.edu. [online]. 2011 [cit. 16.08.2018]. Dostupné z: http://semanticweb.org/wiki/SPARQL_endpoint.html

[34] SPARQL vs SQL. The Smart Data Company® [online]. 2018 [cit. 16.08.2018]. Dostupné z: https://www.cambridgesemantics.com/blog/semantic-university/learn-sparql/sparql-vs-sql/

[35] W3C OWL Working Group. OWL 2 Web Ontology Language Document Overview (Second Edition). World Wide Web Consortium (W3C) [online]. 2012 [cit. 24.01.2018]. Dostupné z: https://www.w3.org/TR/2012/REC-owl2-overview-20121211/

[36] Welcome to the microformats wiki!. Microformats Wiki. Microformats [online]. 2017 [cit. 16.08.2018]. Dostupné z: http://microformats.org/wiki/Main_Page

7 Seznam obrázků, tabulek a zdrojových kódů

7.1 Seznam použitých obrázků

7.2 Seznam použitých tabulek

7.3 Seznam použitých zdrojových kódů

8 Přílohy

1) Zadání bakalářské práce
2) Přiložené CD, které obsahuje:
    - webovou aplikaci ve formě ASP.NET projektu,
    - výslednou ontologii serializovanou ve formátu RDF/XML,
    - databázi Virtuoso s nahranými daty.

Univerzita Hradec Králové
Fakulta informatiky a managementu
Akademický rok: 2017/2018

Studijní program: Aplikovaná informatika
Forma: Prezenční
Obor/komb.: Aplikovaná informatika (ai3-p)

Podklad pro zadání BAKALÁŘSKÉ práce studenta

PŘEDKLÁDÁ:	ADRESA	OSOBNÍ ČÍSLO
Nekolný David	Rybníky 50, Rybníky	I1500393

TÉMA ČESKY:

Webová aplikace pro podporu rozhodování v oblasti bylinné medicíny

TÉMA ANGLICKY:

Web application for decision making in herbal medicine

VEDOUCÍ PRÁCE:

Ing. Martina Husáková, Ph.D. - KIT

ZÁSADY PRO VYPRACOVÁNÍ:

Student vytvoří systém pro podporu rozhodování v oblasti bylinné medicíny. Zvolí vhodný způsob reprezentace, vizualizace a zpřístupnění informací (znalostí) z této oblasti pro nápomoc s řešením vybraných zdravotních obtíží.

Osnova:
1.Úvod
2.Sémantický web a jeho technologie
3.Vybrané aplikace
4.Vývoj webové aplikace
5.Závěry a doporučení
6.Seznam použité literatury

SEZNAM DOPORUČENÉ LITERATURY:

A developer's guide to the semantic web / Liyang Yu. Heidelberg : Springer, c2011. xix, 608 s. ISBN 978-3-642-15969-5 (váz.).

Learning SPARQL : querying and updating with SPARQL 1.1 / Bob DuCharme. Beijing : Sebastopol : O'Reilly, 2011. xiii, 235 s. ISBN 978-1-449-30659-5 (brož.).

Foundations of Semantic Web technologies / Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph. Boca Raton : CRC Press, c2010. 427 s. ISBN 978-1-4200-9050-5 (váz.).

Semantic web for the working ontologist : modeling in RDF, RDFS and OWL / Dean Allemang, James Hendler. Amsterdam : Morgan Kaufmann ; [Burlington] : Elsevier, c2008. xvii, 330 s. ISBN 978-0-12-373556-0.

Semantic web programming / John Hebeler ... [et al. ; foreword by Mike Dean]. Indianapolis : Wiley, c2009. xxix, 616 s. ISBN 978-0-470-41801-7 (brož.).

Programming the semantic web / Toby Segaran, Colin Evans, and Jamie Taylor. Beijing ; Sebastopol : O'Reilly, 2009. xv, 280 s. ISBN 978-0-596-15381-6 (brož.).

Topic maps-based ontology and semantic web : ontology-driven information retrieval system / Myongho Yi. Saarbrücken : VDM Verlag Dr. Müller, 2008. x, 164 s. ISBN 978-3-8364-3519-2 (brož.).

Relational database design and implementation : clearly explained / Jan L. Harrington. Burlington : Morgan Kaufmann, 2009 ; Amsterdam : Elsevier. xix, 420 s. ISBN 978-0-12-374730-3 (brož.).

PHP 6 : programujeme profesionálně / Ed Lecky-Thompson, Steven D. Nowicki ; [překlad Ondřej Gibl]. Brno : Computer Press, 2010. 718 s. ISBN 978-80-251-3127-5 (váz.).

Learning PHP, MySQL, and JavaScript / Robin Nixon. Beijing : Sebastopol : O'Reilly, 2009. xvii, 505 s. ISBN 978-0-596-

15713-5 (brož.).

Nardo, P. 2011. Bylinky našich babiček. Computer Press. ISBN: 978-80-251-2496-3.

Mayer, J. G., Saum, K., Uehleke, B. 2015. Bylinky z klášterní lékárny: Více než 100 léčivých rostlin. Knižní klub, ISBN: 978-80-242-4819-6.

Staňková-Kröhnová, M. 2009. Bylinky pro děti a maminky. Grada. ISBN: 978-80-247-2312-9.

Podpis studenta: Nekolný ..... **Datum:** 16. 8. 2018

Podpis vedoucího práce: ..... **Datum:** .....