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

Diplomová práce

Databáze léčiv a její využití v experimentálním medicínském systému

Plzeň 2016 Bc. Radim Štěpaník

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 8. května 2016

Bc. Radim Štěpaník

Abstract

Tato diplomová práce se zabývá vytvořením databáze léčivých přípravků poskytovaných v České republice. V rámci práce je implementována extrakce a transformace dat z různých datových úložišť a vytvoření RDF datového úložiště. V rámci ověření funkcionality je vytvořen webový server, který komunikuje s daným úložištěm, integruje další datové zdroje na aplikační vrstvě a poskytuje REST API pro klientské aplikace. Další částí práce je klientská aplikace Drugie, poskytující uživatelské rozhraní.

Klíčová slova: RDF, OWL, léčivé přípravky

This diploma thesis deals with creating a database of medicines which are available in the Czech Republic. Part of work is implementation of extraction and transformation data from different datasets and creation RDF data storage. As a confirmation of functionality there is a web server which communicates with data storage and integrates another datasets on application layer. Web server provide REST API for client application. The next part of thesis is client application Drugie. Drugie provides user interface.

Keywords: RDF, OWL, medicines

Obsah

1 Úvod

V dnešní době existuje velké množství datových zdrojů o léčivých přípravcích a jejich chemických, farmakologických a terapeutických vlastnostech. Tyto datové zdroje pocházejí zejména ze zahraničních datových zdrojů. Bohužel existuje pouze omezený počet datových sad zabývající se přímo léky v České republice.

Cílem této diplomové práce je popsat všechny možné datové zdroje a pokusit se integrovat několik těchto datových zdrojů za cílem vytvoření databáze léčivých přípravků poskytovaných v České republice.

V závislosti na vhodnosti se pokusím navrhnout nejlepší možné uložení těchto dat. Na základě daného úložiště vznikne nástroj, který integruje stávající datové zdroje do nově vzniklého úložiště dat o léčivých přípravcích v České republice.

Jako ověření práce bude implementováno řešení, které poskytne uživatelské rozhraní pro dotazování do tohoto datového zdroje a bude představovat možné využití tohoto nově vzniklého datové zdroje.

2 Sémantický web

World wide web, tak jak ho známe dnes, se skládá z obrovského množství dat. Tato data jsou strukturována jako webové stránky, které mají jednoznačný identifikátor. Na základě těchto identifikátorů je pak možné jednotlivé dokumenty mezi sebou spojovat. Tyto základy webu definoval Tim Berners-Lee v roce 1990 [1]. Jedna ze zásadních myšlenek celého konceptu je právě provázání dokumentů pomocí hyperlinků. Díky této revoluční myšlence bylo možné logicky propojovat jednotlivé webové stránky. Zároveň je možné pomocí webu odkazovat na další datové zdroje, jako jsou dokumenty, video soubory, hudební soubory a jiné. Tyto pospojované zdroje si můžeme představit jako orientovaný graf, kde uzly tvoří webové stránky a jiné datové zdroje. Hrany reprezentují skutečnost odkazování z jednoho dokumentu do druhého.

V souladu s dnešními standardy jsme pak schopni k jednotlivým uzlům grafu přiřadit metadata, která vychází ze specifikace [2] jazyka hypertext markup language, dále jako html. Tato metadata příslušící jednotlivým uzlům mohou specifikovat například datum vytvoření dokumentu, autora dokumentu, klíčová slova dokumentu nebo stručný popis dokumentu. V souladu s těmito informacemi byly vytvořeny algoritmy, které jsou schopny prohledávat mezi jednotlivými dokumenty právě na základě obsahu dokumentů, metadat a prolinkování mezi dokumenty. To má z dnešního pohledu však mnohé nevýhody.

Je nutné si uvědomit, že princip world wide webu vznikl jako aplikace hypertextu. Tato myšlenka byla daleko starší než world wide web a publikoval ji v roce 1945 pan Vannevar Bush. Prvotním cílem bylo tedy propojovat dokumenty mezi sebou.

Fakt, že i přestože součástí world wide webu jsou různé dokumenty a soubory, jedná se ve své podstatě stále o propojené dokumenty. V kontextu rozmachu internetu a world wide webu zde nastávají otázky, jak lépe využít grafové uspořádání webu. Bohužel vyvozování dalších skutečností ze současného stavu je velmi složité. Bez technologií a metod strojového učení nebo dokonce umělé inteligence nejsme schopni vyvozovat ze současného prolinkování jednotlivých dokumentů další poznatky.

Sémantický web staví na myšlence, že graf, který reprezentuje provázání dokumentů world wide web, se dá popsat daleko lepším způsobem. Hlavním problémem stávajícího řešení je fakt, že ze současného grafu jsme schopni

vyvodit pouze to, že jednotlivé dokumenty na sebe odkazují. Sémantický web rozšiřuje možnosti organizace jednotlivých uzlů, hran a přidávání metadat k jednotlivých uzlům a hranám. Pokud bychom mohli hovořit o tom, že world wide web modeluje strukturu jakési knihovny, sémantický web se snaží o bližší reprezentaci reality na základě vytvoření jednotlivých entit a vztahů mezi nimi. To je klíčová vlastnost. Je třeba si uvědomit, že dnešní world wide web není tvořen dokumenty, ale jedná se právě o reprezentaci a spojení různých entit. Typickým příkladem jsou sociální sítě, kde profil uživatele sice můžeme chápat jako dokument vytvořený autorem, ale ve své podstatě reprezentuje entitu daného uživatele. V rámci nástrojů sémantického webu by bylo možné jednotlivým uživatelům přiřadit vztahy mezi sebou. Jedná se vlastně o klasické prolinkování, ke kterému je přidána metainformace, jež daný vztah definuje.

Klíčovým důvodem pro zavedení sémantického webu obecně je strojové zpracování těchto dat. Zatímco v současné době je zapotřebí sběru velkého množství dat, použití umělé inteligence a vysokých nákladů k získání informací. Při aplikaci a rozšíření sémantického webu bychom výrazně snížili tyto vysoké náklady na získávání informací, protože by byly obsaženy při samotné implementaci. Zároveň bychom mohli vyvozovat z těchto dat další mnohem zajímavější skutečnosti, které mohou například plynout z hledání neobvyklých vzorů prolinkování jednotlivých uzlů mezi sebou.

2.1 Vývoj webu

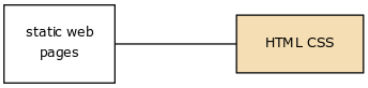
Způsob jakým získáváme z webu webové dokumenty v dnešní době je svým způsobem přímý. Pomocí webového vyhledávače, jakým je například Google, vzneseme dotaz v přirozeném jazyce [3]. Na základě toho nám tato webová služba vyhodnotí náš dotaz a vrátí relevantní výsledky. Vrácené výsledky reprezentují unikátní url adresy k jednotlivým webovým dokumentům. Prostřednictvím url vybraného výsledku nám prohlížeč vrátí relevantní dokument.

Princip sémantického webu jde o krok dále. Na našem počítači, mobilním zařízení je nainstalován personal assistant [3]. Tento assistant pak zprostředkovává komunikaci s řadou webových služeb a skládají dohromady informaci, která je pro uživatele co nejvíce relevantní. Personal assistent nakonec rozhoduje o tom, jakým způsobem bude informace uživateli prezentována.

2.1.1 Generace webových dokumentů

První generace webových dokumentů stavěla na použití statických html stránkách. To znamená, že pro každou unikátní url adresu byl vytvořen právě jeden dokument. Způsob jakým byly informace reprezentovány pak zajišťoval css dokument.

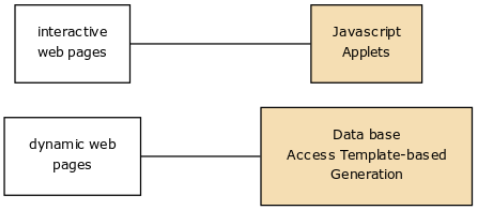
Jako dokumenty první generace se označují webové stránky z rané éry webu. Jedná se především o statické webové stránky. Obsah těchto dokumentů je vytvářen především znalými osobami tj. webmaster. Ostatní uživatelé se pak staví pouze do role konzumentů, nevytvářejí obsah webu [4]. Jako příklad takové stránky můžeme uvést například první webovou stránku `http://info.cern.ch/`.



Obrázek 2.1: web 1.0 [3]

Druhá generace webových dokumentů se vyznačuje dynamickými webovými stránkami. K lepší interakci s uživatelem se používají applety a javascript. Jednotlivé webové dokumenty jsou dynamicky generovány na základě připravených šablon a připojení k databázi.

Jedná se o dnešní podobu webu tak jak ji známe, vyznačuje se tak větší interakcí uživatelů. Uživatel už není pouze konzumentem obsahu, ale aktivně vytváří obsah. Jako aplikace webu 2.0 můžeme považovat sociální sítě, blogy, redakční systémy a jiné [4].

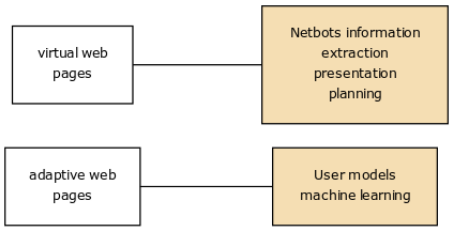


Obrázek 2.2: web 2.0 [3]

V sémantickém webu se využívá třetí generace webových dokumentů.

Hovoříme zde o osobních asistentech, kteří se spojují s webovými službami a na jejich základě vytváří virtuální webové stránky, které jsou personalizované pro daného uživatele.

Web 3.0 je nastupující generací webových dokumentů. Vyznačuje se přístupem k obrovskému množství informací, ke kterým přistupuje prostřednictvím osobního asistenta. Tvoří univerzální přístup k informacím napříč různými systémy. Využívá se umělé inteligence k vytváření relevantních výsledků a vytváření personalizovaných virtuálních webových stránek. Využívá se principů sémantického webu, tedy technologií jako RDF, SPARQL, RDFS a dalších, které jsou blíže popsány v kapitole Architektura sémantického webu.



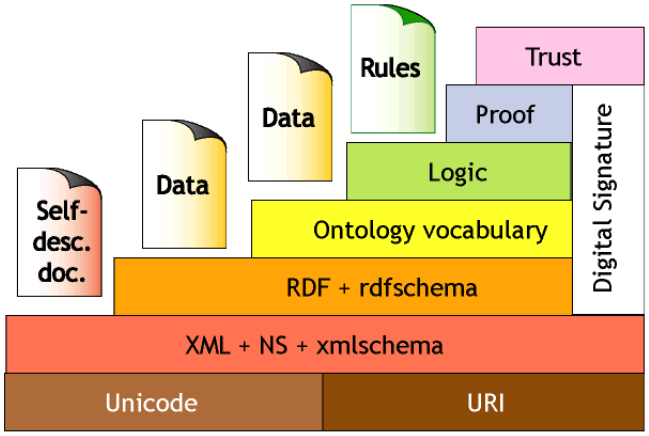
Obrázek 2.3: web 3.0 [3]

2.2 Architektura sémantického webu

Na obrázku [2.4] je vidět základní architektura sémantického webu. Architektura je rozdělena do 7 vrstev [5].

1. vrstva: URI a Unicode Unicode je použit jako univerzální standardizovaný kodovací systém. Na rozdíl od některých jiných umí zobrazovat znaky všech jazyků po celém světě, jako je čeština ale i arabština a čínština. URI, nebo-li uniform resource identifier, je jednoznačným identifikátorem jednotlivých zdrojů.

2. vrstva: XML a XML schema Používá se XML jako ideální jazyk pro reprezentaci struktury dat. XML schéma pak určuje gramatiku daných XML dokumentů.



Obrázek 2.4: Architektura sémantického webu `http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html`

3. vrstva: RDF a RDFS Resource description framework (RDF) poskytuje možnost reprezentace, výměny a znovupoužití metadat. RDF používá URI pro jednoznačnou identifikaci zdrojů. RDF využívá grafový model pro popis vztahů mezi jednotlivými zdroji. RDF Schema (RDFS) poskytuje základní modelovací možnosti, například modelování tříd.

4. vrstva: Ontologie, Slovníky Pomocí RDF a RDFS dokážeme popsat strukturu toho, jak jednotlivé entity jsou mezi sebou propojeny. Nicméně nedokážeme popsat co samotné entity ani vztahy znamenají. K tomuto účelu slouží ontologie, které mají za úkol definovat význam těchto entit a jejich vztahů.

5. vrstva: Logická vrstva Podle [6] je další vrstvou logická vrstva, která se dokáže do námi navrhnutého modelu dotazovat, vyhodnocovat dotazy a zároveň výsledky dotazů transformovat do námi čitelné podoby.

6. a 7. vrstva: Web of Trust [6] Tyto vrstvy by měly zabezpečovat důvěryhodnost poskytovaných výsledků. Zároveň by tyto vrstvy měly zajišťovat bezpečnost uchovávaných informací. V [5] se například uvádí příklad z oblasti medicíny. Kdy by příslušnou zdravotní dokumentaci měl mít přístupnou jen příslušný doktor.

Vertikální vrstva: Digitální podpis Digitální podpis by měl sloužit k jednoznačnému označení odkud byla data získána [7]. Tyto digitální podpisy jsou vytvořeny na základě specifikace [8] W3C XML Signature Syntax and Processing (Second Edition)

2.3 Uniform Resource Identifier

Uniform resource identifier nebo zkráceně uri tvoří základ architektury sémantického webu. Uri je nástrojem pro jednoznačné identifikování zdrojů. Uri je definováno dle RFC 3986 [9] jako jednoduché rozšiřitelné schéma pro celosvětově jednoznačnou identifikaci abstraktních nebo fyzických zdrojů. Uri můžeme použít pro identifikaci různých objektů například knížek v knihovně, automobilů nebo jiných fyzických či abstraktních zdrojů. Uri kombinuje Url a Urn.

Url [10] používáme k jednoznačné identifikaci adresy webových dokumentů na webu. Url označuje adresu, kde můžeme daný webový dokument stáhnout. Prostřednictvím protokolu HTTP pak získáváme daný webový zdroj. Tato url adresa se může během životního cyklu měnit. Na rozdíl od toho Urn [11] nebo-li uniform resource name je persistentní označení pro webové zdroje, které se během životního cyklu nemění.

Pokud bychom používali url a urn odděleně vypadalo by to následujícím způsobem. Chtěli bychom najít konkrétní knihu, která je identifikována urn:isbn:0451450523. Tuto konkrétní urn adresu bychom vložili do nějaké služby, která by zjišťovala adresu, kde daný zdroj můžeme nalézt. Uri tedy představuje oba tyto principy.

```
[schéma]://[userinfo]@[host]:[port]/[path]?[query]#[fragment]
```

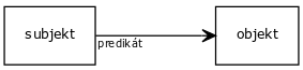
Obecný generický syntax uri se skládá z následujících částí [12]. Schéma představuje protokol jako například http, ftp, mailto. Userinfo obsahuje informace o uživateli a skládá se z hesla a uživatelského jména. Používá se k zabezpečení přístupu na danou uri. Host obsahuje doménové jméno nebo IP adresu verze 4 nebo 6. Obsahuje také informaci o portu. Path označuje cestu k danému dokumentu v rámci souborového systému daného webového serveru. Query obsahuje proměnné vstupní parametry, které daná webová aplikace může zpracovávat a nakonec fragment označuje jednoznačnou část daného dokumentu.

2.4 Resource Description Framework

Timmy Berners-Lee, zakladatel myšlenky world wide webu, hovoří dle [13] o resource description frameworku, dále RDF, jako o nástroji pro tvorbu sémantického webu. Jedná se o abstrakci jakým způsobem můžeme popisovat zdroje, tak aby byly lidsky čitelné a zároveň dobře zpracovatelné strojově. Berners-Lee zmiňuje RDF jako obecný rámec zachycení metadat a XML jako způsob jakým lze tyto informace zapsat.

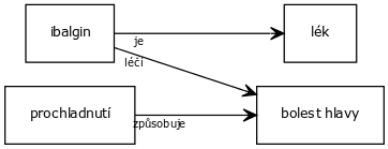
Dle [14] je RDF vysvětleno rozdělením jednotlivých slov názvu. Resource znamená zdroje, kdy se může jednat prakticky o jakoukoliv fyzickou či abstraktní entitu, která musí být jednoznačně identifikovatelná prostřednictvím uri. Description představuje popis zdrojů prostřednictvím popsání atributů a vztahů mezi jednotlivými zdroji. Tyto vztahy mohou být reprezentovány pomocí grafu. Framework pak představuje kombinaci používaných webových protokolů a technologií jako XML, URI, HTTP.

RDF využívá k uchovávání dat velmi jednoduchou myšlenku. Jedná se o vytváření tripletů. Tyto triplety vlastně reprezentují stavbu věty jednoduché přirozeného jazyka. Abych to upřesnil, musíme hovořit o jednotné stavbě takovéto věty. RDF trojice se skládá ze tří částí: subjektu, predikátu a objektu. Přeneseno na přirozený jazyk ukládáme věty složené z podmětu, přísudku a předmětu. Příkladem takovéto věty může být například: "Ibalgin léčí bolest hlavy".



Obrázek 2.5: Obecná struktura RDF tripletu

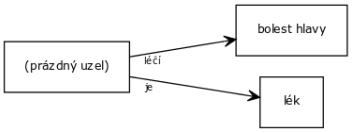
Tento způsob zápisu má několik vlastností důležitých pro tvorbu sémantického webu. Není závislý na struktuře. Tím je myšleno, že například relační databáze potřebují definovat jasně danou strukturu pro ukládaná data. Pokud by došlo ke změně struktury, mohlo by být velmi obtížné tuto strukturu změnit. Při častých změnách struktury by se pak tento úkol mohl stát nemožným. RDF staví na volnosti modelovaní. Zjednodušeně zvolíme jednotlivé entity, které představují jednotlivé subjekty a objekty a určíme vztahy, které mezi sebou mají. Výsledkem pak bude orientovaný graf reprezentující závislosti jednotlivých entit. Mějme například následující triplety: "Ibalgin je lék", "Ibalgin léčí bolest hlavy","Prochladnutí způsobuje bolest hlavy". Tyto triplety pak reprezentuje graf na obrázku 2.6.



Obrázek 2.6: Obecná struktura RDF tripletu

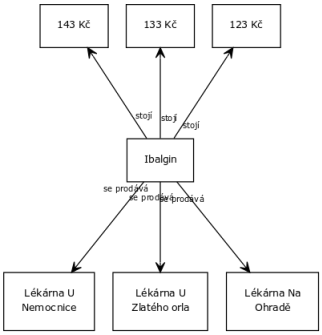
Na grafickém zobrazení je patrné, že mohou vznikat nové informace, které původně nebyly jasně definovány. V tomto konkrétním případě se jedná o informaci, že pokud prochladneme, můžeme některé symptomy odstranit za pomoci léků. Samozřejmě, že na rozsáhlejších strukturách mohou vznikat daleko zajímavější zjištění, která nebyla na první pohled patrná.

Jednotlivé zdroje v RDF grafu by měly být popsány pomocí uri adresy. Stejně tak by měl být popsán vztah, nebo-li predikát, mezi subjektem a objektem. Objekt může být popsán jak pomocí URI, tak se v některých případech hodí popsání pomocí literálu. Další možností jsou také prázdné uzly. Tyto prázdné uzly se používají například kvůli subjektu, který nedokážeme přesně identifikovat, ale můžeme k němu přiřadit jednotlivé atributy. Takovým příkladem může být informace, že na bolest hlavy existuje konkrétní lék. Tento konkrétní lék představuje prázdný uzel. Víme však o něm, že tato entita patří mezi léky a zároveň o tomto léku také víme, že léčí bolest hlavy. Výsledný graf by vypadal následujícím způsobem(obrázek: 2.7).



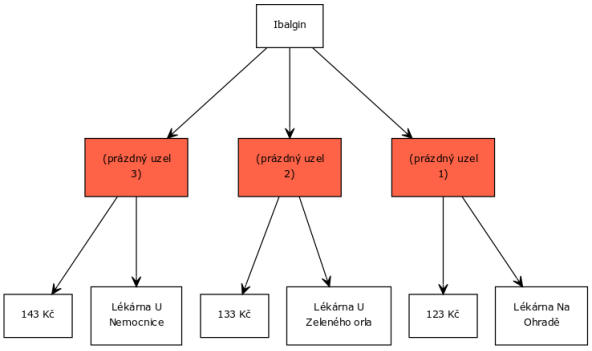
Obrázek 2.7: Prázdný RDF uzel

Tyto prázdné uzly jsou ale také nástrojem pro lepší modelování. Mohou působit jako agregátor většího množství atributů. Mějme následující příklad: ”Ibalgin je prodáván v různých lékárnách za různou cenu.”Jednoduchým modelováním tak dostáváme následující graf:



Obrázek 2.8: Příklad modelování prázdného uzlu 1

Z tohoto grafu jsme schopni získat informace, kde se daný lék prodává a za jakou cenu se lék prodává. Bohužel nejsme schopni se dotázat v jaké lékárně je daný lék nejlevnější. K modelování takovéto situace využijeme právě prázdného uzlu. Přetvořený graf by tedy vypadal následujícím způsobem 2.9:



Obrázek 2.9: Příklad modelování prázdného uzlu 2

K přetvoření modelu jsme využili prázdný uzel pro reprezentaci několikanásobného atributu. Díky spojení několika různých atributů bude možné vyhledávat lékárnu s nejnižší cenou ibalginu. Využili jsme prázdný uzel, protože tuto entitu nemůžeme jednoznačně identifikovat a je generována právě na základě atributů samotných. Další využití prázdných uzlů je modelování kontejnerů a kolekcí. Kontejner je otevřený list, ke kterému je možné přidávat další uzly. Naopak kolekce je uzavřený list, ke kterému nelze přidávat další možnosti [15].

2.4.1 Syntax

V předchozí kapitole jsem shrnul abstraktní principy resource driven framework. Dle architektury sémantického webu pak implementace využívá syntaxe XML dokumentů. Extensible markup language, zkráceně XML je obecně rozšiřitelný značkovací jazyk. XML se používá kvůli schopnosti zachytit strukturu dat. Jelikož je rdf založen na serializaci dat, je možné zapisovat rdf i jinými způsoby, jako je turtle, json, N3 notace. Výhodou těchto jazyků tak může být především lepší přehlednost zápisu.

```
<!-- Deklarace verze dokumentu -->
<?xml version="1.0"?>

<!-- Root element RDF dokumentu -->
<rdf:RDF
<!-- Definice jednotlivych namespace -->
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:drug="http://www.drugs.example/drug#">

<rdf:Description
<!-- Popis urciteho zdroje -->
rdf:about="http://www.drugs.example/drug#ibalgin">
  <drug:price>123</drug:price>
  <drug:producer>Zentiva</drug:producer>
</rdf:Description>

</rdf:RDF>
```

Zdrojový kód 2.1: ukázka rdf dokumentu

Základním principem zůstává, že pomocí xml zapisujeme trojici: zdroj, vlastnost, hodnota. Nejprve je zapotřebí definovat xml dokument. To zajistíme pomocí prvního řádku, xml version. Následně definujeme jmenné prostory nebo-li namespace. Jedná se vlastně o vytvoření zkratek tak, abychom nemuseli stále vypisovat celou URI. Tím se nám struktura daného dokumentu velmi zpřehlední a je čitelná jak strojově, tak i srozumitelná pro člověka. Značka xmlns:rdf tedy definuje jmenný prostor pro rdf syntax. Jedná se o základní sadu atributů, pomocí kterých vytváříme rdf dokument.

Následuje značka xmlns:drug, která definuje jmenný prostor pro popis,

v tomto případě léků. U složitějších příkladů je pak použito daleko větší množství těchto jmenných prostorů sloužících pro definici zdrojů, ontologií. Následuje pak popis určitého zdroje. V tomto konkrétním případě popisuji, že Ibalgin vyrábí společnost Zentiva a stojí 123 korun českých. Je také možné

```
<rdf:Description
rdf:about="http://www.drugs.example/drug#penicilin">
  <drug:manner>
    <rdf:Alt>
      <rdf:li>pill</rdf:li>
      <rdf:li>injection</rdf:li>
      <rdf:li>fluid</rdf:li>
    </rdf:Alt>
  </cd:format>
</rdf:Description>
```

Zdrojový kód 2.2: rdf kontejner

zapisovat skupinu věcí. K tomu používáme v rdf elementy rdf:Bag, rdf:Seq a rdf:Alt. Bag se používá pro specifikaci listů s neseřazeným pořadím položek [16], Seq se používá pro seřazené položky například abecedně nebo podle jiné vlastnosti, Alt pak specifikuje list alternativních hodnot. Jako příklad můžeme uvést skutečnost, že nějaký lék může být podáván v různých formách.

Atribut	**popis**
rdf:about	definuje popisovaný zdroj
rdf:Description	kontejner pro popis zdroje
rdf:resource	definuje zdroje jako atribut
rdf:datatype	definuje datový typ elementu
rdf:ID	definuje ID elementu
rdf:li	definuje list
rdf:_n	definuje uzel
rdf:nodeID	definuje ID uzlového elementu
rdf:parseType	definguje způsob jakým má být element parsován
rdf:RDF	kořen rdf dokumentu

Tabulka 2.1: RDF atributy [17]

Další možností jakým v rdf zapsat list hodnot je použití kolekce. Na rozdíl od kontejneru, což jsou předchozí varianty bag, seq a alt, není kolekce volně rozšiřitelná, tudíž její výčet hodnot je konečný. Tabulka 2.1 definuje další rdf atributy.

Jinou možností, jak zapisovat rdf dokumenty, je formát turtle [18]. Nejjednodušší příklad, jak zapsat triplet ve formátu turtle, je prosté zapsání uri ohraničené špičatými závorkami v pořadí subjekt, predikát a objekt a zakončené tečkou, která ukončuje daný triplet. Je možné k danému subjektu přidávat více atributů, aby došlo ke zkrácení zápisu. Jednotlivé atributy jsou odděleny středníkem. Zajištění zápisu jmenných prostorů se používá @prefix na začátku dokumentu. Příkladem turtle zápisu může být následující dokument.

```
@prefix drug: <http://www.drugs.example/drug/> .
@prefix type: <http://www.drugs.example/type/> .
@prefix price: <http://www.drugs.example/price/> .
@prefix producer: <http://www.drugs.example/producer/> .
@prefix rel: <http://www.drugs.example/relation/> .

drug:ibalgin rel:isProduced producer:zentiva.
drug:ibalgin rel:price price:ibalgin.
drug:ibalgin rel:type "painkiller".
```

Zdrojový kód 2.3: turtle

Jako podmnožina formátu turtle vznikl formát N-triples. Tento formát má jednodušší zápis lépe přizpůsobený strojovému zpracování. Na první pohled je tento způsob velmi podobný turtle, a dokonce by zapsání jednoduchých tripletů prošlo validátorem turtle [19]. Původním důvodem pro vznik tohoto formátu bylo testování, ale během času bylo dokázáno, že se jedná o vhodný výměnný formát pro rdf dokumenty.

Dalším způsobem zápisu rdf dokumentů je například za pomoci v dnešní době populárního JSON. Tento způsob zápisu se nazývá JSON-LD. JSON, nebo-li javascript simple object notation, má velmi jednoduchou syntax. Objevuje se vždy dvojice key, value, kdy value může být další vnořený json objekt.

```
{ "@context": "http://json-ld.org/contexts/person.jsonld",
  "@id": "http://dbpedia.org/resource/John_Lennon",
  "name": "John Lennon",
  "born": "1940-10-09"}
```

Zdrojový kód 2.4: Příklad json-ld [20]

2.5 RDF Schema

RDF schéma, zkráceně RDFs, je základním kamenem tvorby ontologií. RDF schéma definuje datový model nad daty. Model je uložen jako RDF dokument [16]. K datovému modelování se využívá několika základních vlastností. Vytváříme definici tříd, následně můžeme v RDF vytvářet instance těchto tříd. Definice omezení a konstrukce hiearchií prostřednictvím tvorby podtříd [21].

	Název třídy	**Popis**
Zdroj	rdfs:Resource	všechny entity RDF modelu jsou instancí této třídy
Třída	rdfs:Class	definuje abstraktní objekt, pomocí třídy se vytváří její instance
Vlastnosti	rdf:Property	základní třída pro vlastnosti
Literály	rdfs:Literal	třída pro literály

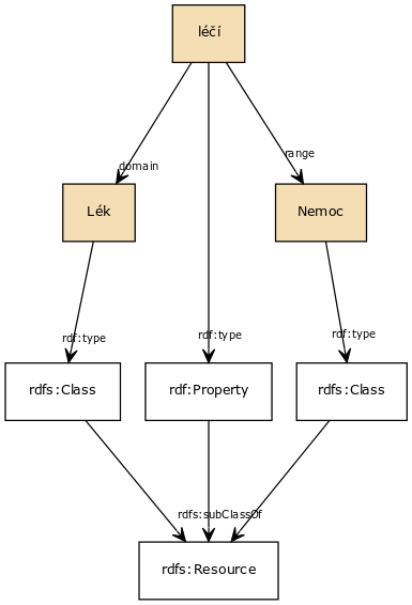
Tabulka 2.2: RDFs základní třídy [16]

V tabulce 2.2 je uveden přehled základních tříd RDF schema. Existují však ještě další třídy se specifickým využitím jako rdf:Datatype, rdf:XMLLiteral, rdf:Statement a jiné [16].

Kromě jednotlivých tříd, RDF schema definuje vlastnosti mezi jednotlivými třídami jako jsou subClassOf, subPropertyOf, domain a range. Výraz rdfs:subClassOf se používá pro implementaci dědičnosti. SubPropertyOf používáme pro vyjádření dědičnosti u vlastností. Domain definuje vztah mezi vlastností a subjektem a range mezi vlastností a objektem. Za pomocí těchto výrazů pak můžeme vytvářet složitější modely, ve kterých jsou jasně definovány vztahy mezi jednotlivými entitami. Pomocí schéma vytváříme také omezení o tom, které entity můžou mít mezi sebou nějaký vztah.

Mějme například RDF dle obrázku 2.6. Triviální schéma takového příkladu je vidět na obrázku 2.10. Všechny entity daného modelu jsou instancí

třídy rdfs:Resource. V datovém modelu také vystupují třídy Léky a Nemoci. Datový model definuje ”léky léčí nemoce”. To je definováno tím, že vlastnost ”léčí”může mít jako svojí domain lék a range nemoc. Tím vlastně předurčujeme povolenou strukturu RDF tripletu. Tím například zamezujeme tomu, že žádná nemoc nemůže léčit lék.



Obrázek 2.10: Příklad datového modelování RDFs

RDFs popisuje pouze strukturu daného datového modelu. Na předchozím případě jsme sice definovali, že léky léčí nemoci, ale fakticky nám RDF schéma neříká nic o tom, co tyto entity znamenají. Z lidského pohledu je jasné, že léky léčí nemoci, ale z pohledu strojového čtení jsou to pouze dvě entity reprezentované string hodnotou. K tomu, abychom mohli rozšířit strojově čitelné znalosti o daném modelu je zapotřebí zapojení dalších slovníků a ontologií. Tyto pojmy definuji v následujících sekcích.

2.6 Dotazování v sémantickém webu - SPARQL

SPARQL je nástrojem pro dotazování v technologiích sémantického webu. SPARQL je založen na RDF Turtle serializaci a hledání grafových vzorů [22].

SPARQL není pouze dotazovacím jazykem. SPARQL Protocol and RDF

Query Language je komplexní popsání systému vznesení požadavku, dotázání a vrácení výsledku. Hovoříme o třech částech.

SPARQL query language [23] zajišťující syntax podobnou jazyku xml. A možnostmi pro dotazování do grafové struktury. SPARQL - protocol pro RDF zaštiťující komunikaci mezi SPARQL klientem a dotazovacím procesorem [24]. Poslední částí je SPARQL Query Results XML Format [25] popisující XML formát a strukturu pro výsledky provedené na základě SPARQL dotazu.

Mezi schopnosti SPARQL 1.0 patří [22]: Extrakce dat jako URI, Blank nodes, literály. Subgrafy RDF. Prozkoumávání dat prostřednictvím dotazování na neznámé vztahy. Komplexní spojovací operace na heterogenní databáze v jednom dotazu. Transformovat RFD data z jednoho do jiného slovníku. Kontstrukce nového RDF grafů na základě dotazovaného grafu.

Rozšířené funkce SPARQL 1.1 jsou: Rozšířené možnosti dotazování jako agregační funkce, subqueries, negace a jiné. Dovoluje například aktualizaci RDF grafu a plnou manipulaci s daty.

Syntax SPARQL je odvozená a vizuálně podobná jazyku SQL. Pro dotázání se na některé proměnné grafu používáme, stejně jako u SQL, výraz SELECT, který doplňujeme o výčet proměnných. Následuje výraz WHERE, který upřesňuje hledaný grafový vzor.

```
<!-- Deklarace namespace -->
PREFIX drug: <http://www.drugs.example/drug/>
PREFIX type: <http://www.drugs.example/type/>
PREFIX price: <http://www.drugs.example/price/>
PREFIX producer: <http://www.drugs.example/producer/>
PREFIX rdf <xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">


SELECT ?drug, ?price
WHERE
       ?drug rel:type "painkiller";
              rel:price ?price.
ORDER BY ASC (?price)
LIMIT 10 OFFSET 0
```

Zdrojový kód 2.5: Základní SPARQL dotaz [22]

Na příkladu 2.5 je vidět základní struktura SPARQL dotazu. Tento konkrétní příklad je založen dle příkladu zdrojového kódu dle ukázky 2.3. Tento konkrétní dotaz vrací deset nejlevnějších léků, které označujeme jako "painkiller". V horní části dotazu jsou definovány namespace, které se v dotazu používají. Dále následuje popis struktury výsledků dotazu pomocí výrazu SELECT. Tento příklad požaduje vrácení léků a cen. Následuje výraz WHERE, ve kterém specifikujeme grafový vzor daného dotazu, tedy že hledáme všechny zdroje, které jsou typu "painkiller"a mají uvedenou cenu. Na konci je uvedeno řazení ORDER BY a počet vrácených výsledků LIMIT, OFFSET, podobně jako u SQL dotazů.

Mimo SELECT můžeme ve SPARQL provádět ještě další operace. ASK kontroluje, zda uvedený dotazovaný vzor v grafu existuje. Vrací pouze hodnotu true nebo false. CONSTRUCT vrací nový RDF graf vytvořený dle vzoru z původního grafu [23]. DESCRIBE vrací jediný výsledek, který obsahuje pouze RDF data o zdrojích [23] .

2.7 Ontologie

Dle [26] je princip sdílení znalostí založen na použití společné sady znaků a konceptů jazyka - syntax. Dohoda o tom, co koncepty a jednotlivé znaky znamenají - sémantika. Klasifikace jednotlivých konceptů do hierarchické struktury - taxonomie. Asociace a vztahy mezi koncepty definované ve slovnících a pravidla a znalosti o vlastnostech a jejich vztazích - ontologie.

RDF schéma, o kterém se zmiňuji v kapitole 2.5, je základním kamenem pro tvorbu ontologií. RDFs nám dovoluje tvořit datový model. Nicméně tento model sice dokáže postihnout strukturu grafu, ale není schopen vyjádřit složitější omezení. Jednoduchým příkladem je to, že RDFs vyjádří to, že člověk má svého rodiče, ale už pomocí tohoto schéma nedokážeme vystihnout to, že jeden člověk má právě dva rodiče.

K vyjádření těchto složitějších vztahů používáme ontologie. Ontologie je ve filozofickém pojetí chápana jako nauka o bytí, případně jako univerzální soustava znalostí popisující objekty, jevy a zákonitosti světa "tak, jak je"[27]. Jak uvádí [27] je definice ontologie v kontextu světa informatiky možná formulace T. Grubera : "Ontologie je explicitní specifikace konceptualizace", kdy konceptualizace je systém pojmů modelující určitou část světa. Tato definice byla dále rozšířena W. Borstem: "Ontologie je formální specifikace sdílené

konceptualizace".

2.7.1 OWL

Pro rozšířenou tvorbu ontologií se používá OWL - Web ontology language. Jazyk OWL 2 se může zapisovat v několika variantách - funkcionální syntax, který nahrazuje abstraktní syntax OWL 1. RDF syntax, XML syntax, Manchester syntax - vhodný pro strojové čtení, turtle. Nejjednodušším způsobem zápisu ontologií je dle [28] turtle.

```
<!-- Definice tridy -->
<owl:Class rdf:about="http:/drug.ex/drug#drugtype">
        <rdfs:label>Lek</rdfs:label>
        <rdfs:comment>Trida vsech leku</rdfs:comment>
</owl:Class>
<!-- Definice podtridy Painkiller -->
<owl:Class rdf:about="http://drug.ex/drug#painkiller">

<!-- Definice vlastnosti -->
<owl:DatatypeProperty rdf:about="http://drug.ex/drug#atc"/>

<owl:ObjectProperty rdf:about="http://drug.ex/drug#similar"/>

        <!-- Painkiller je podtridou leku -->
        <rdfs:subClassOf rdf:resource="http://drug.ex/drug#drugtype"/>
        <rdfs:label>Prasky proti bolesti</rdfs:label>
</owl:Class>
```

Zdrojový kód 2.6: OWL ukázka

Na ukázce kódu 2.6 je vidět použití OWL. Příklad ukazuje definování třídy léků a léků proti bolesti. Dle této ontologie jsou léky proti bolesti podtřídou léků. Zároveň zde definujeme atribut o ATC skupině léku a o podobných léčivech.

Na ukázce 2.7 je pak vidět konkrétní použití v rdf. Příklad popisuje konkrétní lék - ibalgin. Definuje třídu daného léku, tedy že se jedná o lék proti bolesti. Dále vidíme zařazení do konkrétní ATC skupiny a možné náhrady daného léku.

```
<!-- priklad RDF statement -->
<rdf:Description rdf:about="http://drug.ex/drug#ibalgin">
        <!-- Ibalgin je prasek proti tlumeni bolesti -->
        <rdf:type rdf:resource="http://drug.ex/drug#painkiller"/>
        <!-- Definice skupiny ATC -->
        <drug:atc>M01AE01</drug:atc>
        <!--Misto ibalginu lze pouzit paralen-->
        <drug:similar rdf:resource="http://drug.ex/drug#paralen"/>
</rdf:Description>
```

Zdrojový kód 2.7: ukázka použití

3 Dostupné datové zdroje

V této kapitole bych chtěl představit zdroje dat o léčivech, které jsou dostupné. Jedná se o přehled dostupných informací a zhodnocení jejich strojového zpracování.

3.1 Státní ústav pro kontrolu léčiv

Státní ústav pro kontrolu léčiv, dále zkráceně SÚKL, [29] je organizační složkou státu. SÚKL je zřízen Ministerstvem zdravotnictví České republiky.

Dle [30] jsou hlavními povinnostmi SÚKL kontrola jakosti léčiv, kontrola jakosti přípravků ze kterých se léky vytvářejí. Dále se ústav zabývá tím, aby byly používány pouze zdravotnické prostředky, které jsou podle dostupných vědeckých poznatků bezpečné a funkční. Zabývá se cenovou dostupností léků. Zodpovídá za popis léků, jeho pravdivost a objektivnost. Zajišťuje věrohodnost údajů získaných při výzkumu léčiv a zdravotnických prostředků a kontroluje etickou správnost jejich získání. Poskytuje nestranné informace, aby ústav přispěl k tomu, aby byly léčiva a zdravotnické prostředky správně používány. Zajišťuje v rámci své působnosti kontrolní činnost a přispívá k vytváření prostředí důvěry v kontrolní systém České republiky.

Dle [30] článek 4, část 2 zmíním pouze činnosti relevantní k této diplomové práci. Zajišťuje správu datového úložiště pro elektronickou preskripci. Rozvíjí informační podporu státní správy a veřejnosti s cílem odstranit neznalosti o lékové politice a reálném stavu zacházení s léčivy. V oblasti informačních technologií je používáno takové řešení, které zajistí funkčnost a efektivnost ve prospěch občanů.

SÚKL provozuje portál www.sukl.cz [29]. Tento portál obsahuje databázi léků s aktivním výskytem na českém trhu dle pravidelného hlášení dodávek distributorů. Je zde také vedena databáze lékáren v České republice.

3.1.1 Otevřená data sukl

SÚKL poskytuje tato data prostřednictvím otevřených dat [31]. Dle [32] je definice otevřených dat: Otevřená data jsou informace a data zveřejněná na internetu, která jsou úplná, snadno dostupná, strojově čitelná, používající standardy s volně dostupnou specifikací, zpřístupněná za jasně definovaných podmínek užití dat s minimem omezení a dostupná uživatelům při vynaložení minima možných nákladů. Konkrétně jde o různé statistiky, rozpočty, přehledy, databáze apod.

Podmínky užití těchto dat jsou dostupné na [33]. Tato data je povoleno šířit - kopírovat, distribuovat a sdělovat veřejnosti, využívat a citovat, využívat komerčně. Toto použití je povoleno za podmínek, že jako zdroj musí být uveden SÚKL a musí být použity způsobem, který nemění význam těchto dat.

3.1.2 Dostupné formáty dat

Na portále SÚKL je možné získat datové sady ve dvou základních formátech. Dříve byla data exportována do formátu xls. Pro zažádání o datovou sadu je nutné na portálu zadat svoji emailovou adresu. Poté daný uživatel obdrží emailovou zprávu s unikátním odkazem na stažení datové sady. Tato datová sada je poskytnuta v komprimovaném formátu .zip. Po dekomprimaci jsou tato rozbalená data ve formátu .xls.

Po bližším prozkoumání datové sady jsem zjistil, že i přesto, že má daný soubor příponu pro soubory aplikace Microsoft Excel .xls, je pravý formát daného souboru html. Pomocí html elementů je pak vytvořena tabulka. Takto zpracovaný soubor je možné otevřít v aplikacích Microsoft Excel. Nepochybnou výhodou toho, že daný soubor je ve formátu html, je strojové zpracování. Daný soubor obsahuje všechny data v jedné tabulce a jedná se o popis léčivých přípravků v České republice. K této datové sadě nejsou dostupná metadata ohledně významu jednotlivých sloupců, proto není význam některých sloupců v této datové sadě zcela jasný.

Zároveň SÚKL poskytuje data ohledně léků v České republice prostřednictvím otevřených dat na stránkách opendata.sukl.cz [31]. Tato data jsou poskytována na portálu SÚKL od 1. 1. 2016. Tuto informaci jsem získal na základě přímého dotazu na kontakt uvedený na portále. Na tomto webovém

portálu, který je spravován právě SÚKL, jsou dostupné datové sady popisující aktuální stav z oblasti léčiv v České republice. Jednotlivé datové sady jsou přehledně zpracované do konkrétních oblastí. Tyto datové sady jsou pak dostupné ve formátu csv.

3.1.3 Dostupné datové zdroje

Databáze SÚKL představuje především hlavní datový zdroj o lécích v České republice. Do 1. 1. 2016 poskytovala pouze data přímo související s léčivými přípravky v České republice. Jednalo se pouze o data o léčivých přípravcích, bez bližšího popisu této datové sady. Tato datová sada obsahovala pouze jednu tabulku, která obsahovala mimo jiné tyto informace: kód sukl, název přípravku, doplněk názvu, cesta podání, léková forma, velikost balení, síla přípravku, typ balení, země držitele, aktuální držitel rozhodnutí o registraci, ATC skupina, registrační číslo, souběžný dovozce, země souběžného dovozce, registrační procedura, režim prodeje a další.

Tato datová sada neobsahovala bližší metadata, která by poskytovala lepší představu o obsahujících datech. Některé sloupce, jako například režim prodeje nebo registrační procedura, obsahovaly pouze zkratky hodnot. Bylo by tedy nutné provést analýzu jednotlivých sloupců a zpětně nalézt všechny hodnoty kterých nabývají a zjistit příslušná metadata, co sloupce znamenají.

Od 1. 1. 2016 je však na portále SÚKL dostupná prostřednictvím projektu Otevřených dat datová sada ve formátu csv [31]. Tato datová sada je členěna na několik částí. Hlavní část tvoří především Databáze léčivých přípravků a Datové rozhraní k Databázi léčivých přípravků.

Databáze léčivých přípravků se skládala v době exportu, 4. 1. 2016, z celkem 34 csv souborů. Každý z těchto souborů představuje entitu související s léčivými přípravky dostupných v České republice. Dohromady tyto soubory představují klasický relační model. Pro zjednodušenou představu tato datová sada obsahuje informace o léčivých přípravcích. Tyto informace jsou blíže vysvětleny prostřednictvím číselníků. Datová sada tak obsahuje například číselníky o výrobcích léčivých přípravků, léčivých látkách, dokumentech a dalších.

Všechny tyto číselníky jsou popsány prostřednictvím Datového rozhraní k Databázi léčivých přípravků [31]. Tento datový číselník ve formátu csv obsahuje 3 sloupce. První sloupec definuje název datové sady, druhý slou-

pec obsahuje pořadí sloupce v daném souboru, třetí sloupec název sloupce a nakonec poslední, čtvrtý sloupec, definuje význam.

3.1.4 Popis několika významných datových zdrojů

V této pasáži budou popsány nejdůležitější součásti datové sady a některé jejich sloupce. Jednotlivé popisy jsou převzaty a doplněny z Datového rozhraní k Databázi léčivých přípravků a dalších zdrojů [31].

Léčivé přípravky

Hlavní datový zdroj. Obsahuje názvy jednotlivých léčivých přípravků. Obsahuje bližší specifikaci jednotlivých léčivých přípravků. Odkazuje na další číselníky. Obsahuje identifikátor "kod sukl". Jedná se o kód léčivého přípravku, který přiděluje SÚKL dané variantě léčivého přípravku. Tento identifikátor je získán v rámci rozhodnutí o registraci daného léčivého přípravku.

Dále obsahuje název léčivého přípravku, doplněk názvu léčivého přípravku. Doplněk názvu jednoznačně určuje variantu léčivého přípravku. Určuje se dle cesty podání léčivého přípravku, lékové formy, velikosti balení a síly.

Datová sada obsahuje data ohledně síly léčivého přípravku. Tato síla léčivého přípravku je vyjádřena kvantitativně vzhledem k jednotce dávky, objemu nebo hmotnosti podle lékové formy.

Anatomicko-terapeuticko-chemická skupina nebo-li WHO index [34]. ATC poskytuje standard pro klasifikaci léčivých látek a slouží jako nástroj pro výzkum využití léčiv. ATC kódy jsou zahrnuty v mezinárodních katalozích léků a mohou tak být použity pro srovnání jednotlivých léčivých přípravků [35].

Obsahuje také označení přípravků s anabolickým a jiným hormonálním účinkem, dopingových látek, skupiny omamných nebo psychotropních látek (návykové látky).

ATC skupiny

Datový zdroj obsahuje popis anatomicko-terapeuticko-chemických-skupin. Tato datová skupina obsahuje kód dané skupiny, český a anglický název.

Jednotlivé kódy jsou přidělovány na základě skupin dle účinků na jednotlivé orgány a jejich systémů na které působí, na základě chemických skupin, farmakologických a terapeutických vlastností [36].

Existuje celkem čtrnáct hlavních skupin - 1. stupeň, které mají jednu farmakologicko terapeutickou podskupinu - 2. stupeň. Třetí a čtvrtý stupeň je chemicko farmakologicko terapeutická podskupina, a nakonec pátý stupeň reprezentuje chemickou sloučeninu [36].

Kód skupiny může být tvořen maximálně sedmimístným kódem. Tento kód představuje jednotlivé stupně a je tvořen písmeny a číslicemi. První úroveň je tvořena jedním písmenem. Druhá úroveň je tvořena dvěma číslicemi. Třetí i čtvrtá úroveň je označena jedním písmenem. Pátá úroveň je označována dvěma číslicemi.

Dokumenty

Datový zdroj obsahuje informace o názvech dokumentů související s daným léčivým přípravkem. Tyto soubory jsou pak dostupné přímo na portálu SÚKL. Dokumenty jsou na portále uloženy ve formátu pdf.

Tento zdroj poskytuje informaci o názvu dokumentu příbalového letáku. SPC nebo-li souhrnu údajů o léčivém přípravku a název souboru s textem na obalu.

Látky

Datová sada obsahuje základní informace o látkách, které se vyskytují v léčivých přípravcích a jsou v databázi SÚKL. Tento soubor poskytuje informaci pouze o názvu dané látky a to v české i anglické variantě. Dále se tento datový zdroj odkazuje na číselníky doping, NARVLA - označení látek dle Nařízení vlády 454/2009 Sb. a číselník závislostí, tedy jestli daná látka způsobuje závislost.

Složení léčivých přípravků

Datová sada spojuje látky s léčivými přípravky. Jsou zde specifikovány léčivé látky, jež daný léčivý přípravek obsahuje. U tohoto vztahu je také definováno minimální a maximální možné obsažení dané léčivé látky v daném léčivém přípravku.

Další datové zdroje

Celkem datové zdroje SÚKL obsahují 34 datových sad. Tyto další datové sady obsahují informace o výrobcích, synonyma léčivých látek, stavy registrací léku a další. Jedná se především o číselníky, které blíže specifikují daný léčivý přípravek. Tyto datové sady jsou pak využity v hlavní datové sadě - léčivé přípravky.

Dále jsou v katalogu otevřených dat SÚKL dostupné další datové sady. Jsou zde k dispozici ke stažení přímo dokumenty k léčivým přípravkům, jako souhrn údajů o léčivém přípravku, příbalové informace a obaly na textu. Dále je zde datová sada se seznamem lékáren v České republice a dále soubory obsahující agregovaná data z hlášení o dodávkách do lékáren a zdravotnických zařízení.

3.2 Ministerstvo zdravotnictví České republiky

Ministerstvo zdravotnictví poskytuje na svém portále informace o úhradách a cenách a výši případného doplatku léčivých přípravků hrazených z veřejného zdravotního pojištění. Tato databáze používá jako hlavní identifikátor kód léčivého přípravku z databáze SÚKL.

O každém léčivém přípravku je možné dále zjistit doplněk léčivého přípravku, anatomicko-terapeuticko-chemickou skupinu. Ohledně cen léčiv dostupných v České republice poskytují nejvyšší možnou cenu v lékárně při výdeji na recept, nejvyšší možný doplatek pacienta, průměrnou aktuální cenu v lékárně a průměrný aktuální doplatek pacienta. Dále jsou na tomto portále dostupné dokumenty příslušící k danému léčivému přípravku.

3.3 U.S. National Library of Medicine

Národní knihovna medicíny (NML) se nachází ve městě Bethesda ve státě Maryland. Je součástí Národního institutu zdraví (National Institutes of Health), amerického ministerstva zdravotnictví a sociálních služeb (HHS). Je to největší biomedicínská knihovna na světě a jejím úkolem je zavádění biomedicínských výzkumů do praxe. Tato knihovna byla založena v roce 1836 a od té doby se stala úložištěm neuvěřitelného množství dat pro miliony uživatelů každý den. Elektronická informační služba nabízí až biliony bajtů s daty. Do elektronické knihovny mají přístup vědci, zdravotníci i veřejnost po celém světě a knihovna je využita až v počtu jedné miliardy vyhledání za rok.

Knihovna má obrovskou škálu služeb, která obsahuje téměř 19 miliónů knih, časopisů, rukopisů, ROM a jiné formy lékařských informací.

Svou otevřeností pro veřejnost dopomáhá v dnešní digitální době k většímu povědomí o zdravém chování, péči o zdraví, podporuje veřejné zdraví, které stojí na základech biomedicínských výzkumů.

NML se zabývá získáváním, organizováním a zachováváním světové biomedicínské literatury. Spolupracuje s národní sítí knihoven, kterých je po celém světě 5600. Umožňuje přístup nejen k oblasti biomedicíny, ale také k molekulární biologii, toxikologii, enviromentálnímu zdraví, k výzkumům úrovní zdravotnických služeb a veřejného zdraví.

3.3.1 Datové zdroje

Na portálu U. S. Library of Medicine bylo v dubnu 2016 dostupných celkem 291 datových zdrojů. Tyto datové zdroje jsou dostupné na portálu National Library of Medicine [37]. Jednotlivé datové zdroje lze třídit podle několika parametrů. Prvním kritériem vyhledávání datových sad je oblast vyhledávání, jako jsou chemické vlastnosti, lékařské publikace, nemoci, genetika, toxikologie a další.

Dalším kritériem je poskytovaný formát datové sady. Datové zdroje jsou například dostupné prostřednictvím například API, ve formátu xls nebo csv, ve stažitelném formátu txt, pdf, html.

3.3.2 Vybrané datové zdroje

Všechny tyto datové zdroje nesouvisejí přímo s léčivými přípravky a léky. Jedná se o datové zdroje související s lékařstvím obecně. Ze všech datových zdrojů, které nabízí National Library of Medicine vyberu některé datové sady, které souvisej í s tématem diplomové práce. Blíže se pak dále zaměřím na datovou sadu RxNav, kterou blíže popíši v následující sekci.

AIDSInfo API Poskytuje přístup k lékům, které se používají pro léčbu nemoci AIDS. Tento datový zdroj poskytuje API pro extrakci dat.

Clinical Trials Poskytuje databázi studií ohledně léků a zacházení s nimi.

DailyMed DailyMed je webový server vytvořený v listopadu 2005 U.S. lékařskou knihovnou (NLM). Server spolupracuje se Správou léčiv a potravin (Food and Drug Administration - FDA). Obsahuje oficiální příbalové letáky léčiv, které jsou schválené FDA. Informace si uživatel může nejen přečíst, ale i stáhnout. Je vhodný pro lékaře i pacienty. Dokonce si uživatelé mohou nastavit automatické zasílání informací o aktualizacích nebo zrušení určitých léků. Výhodou je pomoc při vytváření osobních zdravotních záznamů v elektronické podobě.

Drug Information Tento informační portál je vstupní branou do aktuálních informací o léčivech z NLM a dalších vládních agentur. Drugs.com poskytuje informace o více než 24.000 lécích na předpis, bez předpisu a o přírodních produktech. Obsahuje vysvětlení interakcí mezi léky a možné reakce mezi léky a potravinami.

3.3.3 RxNav API

Služba RxNav API [38] vznikla v Lister Hill National Center for Biomedical Communications.

Národní centrum pro biomedicínskou komunikaci, je součástí NLM (National Library of Medicine), které se snaží zlepšit přístup k velice kvalitním a

odborným informacím z oboru biomedicíny a tím podpořit snahy NLM o celosvětovou přístupnost ke zdravotnickým informacím.

Mezi zdroje tohoto centra patří ClinicalTrials.gov, které podává informace o dostupných klinických výzkumných studiích pro širokou škálu onemocnění. Profiles.nlm.nih.gov (Profily ve vědě) obsahují sbírky, knihy, časopisy, rukopisy a mnohé další k osvětlení historie zdravotnictví. Pokud se uživatel snaží najít něco o genetických výzkumech, jehož vysvětlení je podané pro laiky, pak využije zdroj z Genetics Home Reference. Byla také vytvořena databáze lékařských článků pod názvem: Medical Articals Record System. Mezi největší novinky patří výuka medicíny pomocí podrobného vytváření 3D obrazů a tím názorným ukázkám lidského těla.

RxNav je REST API služba, která sdružuje data o léčivých přípravcích z několika různých zdrojů. Sjednocuje mimo jiné datové sady RxNorm [39], NDF-RT [40] a RxTerms [41].

RxNorm RxNorm je produkována NLM. Je to normalizovaná databáze léků, která obsahuje informace z odborné terminologie a databáze lékáren. Pro řešení problémů s různými sadami stejných léků, jejichž názvy se mohou lišit, používá RxNorm normalizované názvy a identifikátory léčiv. Cílem RxNorm je, aby počítačové systémy efektivně a jednoznačně sdělovaly informace o jednotlivých lécích. Umožňuje zjistit, které léky mají podobné složení nebo názvy výrobce či velikost balení. Pokud se vyhledávaný lék již neprodává, RxNorm vytvoří graf zastoupených léčivých látek, a tím se snaží vyhledat co nejvhodnější náhradu na již neexistující lék. RxNorm obsahuje léky na předpis nebo celé balíčky léků, které obsahují více druhů léků nebo léky určené k podání ve stanoveném pořadí.

NDF-RT NDF-RT je produkováno americkým Ministerstvem pro záležitosti veteránů (VHA). NDF-RT je rozšíření souboru NDF. Upravuje seznam léčiv do formální reprezentace. Tento server je důležitý pro vytváření charakteristiky léčiv. Obsahuje informace o složkách léků, chemickém složení, formě léku, o fyziologických účincích, mechanismu účinku a farmakokinetice. NDF-RT kombinuje hierarchickou klasifikaci léčiv serveru NDF s referenčním modelem multi-kategorií. Mezi tyto kategorie patří: buněčné nebo molekulární interakce, chemické přísady a složky, klinická kinetika, nemoci, dávkování, farmaceutické přípravky, fyziologické účinky, terapeutické kategorie nebo interakce mezi léky. NDF-RT je měsíčně aktualizován v Metathesauru jako

součást RxNorm.

RxTerms RxTerms je databáze léků odvozená od RxNorm, která obsahuje databázi léků na předpis. Data o léčivech čerpá ze serveru RxNorm, ale na rozdíl od něj má pokročilejší formu vyhledávání a vylučuje z této databáze léky zastaralé nebo ty, které nejsou v USA k dispozici. Výhodou je navádění uživatele ke správnému psaní hledané látky předepsanými možnostmi, a tím zabraňuje chybám ve vyhledávání. Na základě seznamu léků, které jsou na předpis v USA není vhodný pro vyhledávání pro ostatní země. RxTerms se používá i v NML Personal Helath Record.

3.4 Drugbank

Databáze DrugBank je zdroj, který obsahuje širokou škálu informací z oblasti bioinformatiky a chemoinformatiky. Kombinuje podrobné chemické, farmakologické a farmaceutické informace o léčivých přípravcích s komplexními cíli léčiv. Každá položka DrugCard obsahuje datová pole s informacemi, z nichž polovina je věnována chemickým informacím o léku a druhá polovina jeho léčivým účinkům.

3.5 Open PHACTS

V současné době se farmaceutické společnosti snaží sestavit vlastní databáze farmakologických a fyzikálně chemických datových zdrojů. PHACTS integruje data z různých veřejně dostupných databází, a to jim umožňuje vytvořit stabilní infrastrukturu plnou různých databází, které jsou navzájem propojeny. To napomáhá vědcům ke zvýšení efektivity pracovních postupů v oblasti výzkumu a analýzy, jelikož podrobně informuje o složení léků a jeho interakcích. PHACTS platforma integruje data spojená s každým identifikátorem jednotlivých léků, takže uživatel může procházet informace o léčivech z různých zdrojů na jednom místě.

4 Návrh datového modelu

Jak jsem deklaroval v předchozí kapitole, existuje velké množství datových sad, které se týkají léků, léčivých prostředků, chemického složení. Hlavním cílem této diplomové práce bude vytvoření databáze léku v České republice a pokusit se nalézt způsob integrace dalších zdrojů.

Nejdříve se pokusím nastínit hlavní cíle, které by měla datová sada obsahovat. V následující sekci otevřu diskuzi ohledně vhodného výběru způsobu ukládání těchto dat. Tím myslím buď rozhodnutí mezi uložením dat prostřednictvím nějaké relační databáze, nebo využití spíše principů NoSQL databází, jako RDF úložiště nebo databáze typu key - value.

Následně jsem se pokusil analyzovat dostupné datové zdroje popsané v předchozí kapitole. Na základě vhodnosti využitelnosti jednotlivých dat rozhodnu, které datové sady využiji pro tvorbu integrovaného datového zdroje.

Cílem diplomové práce by nemělo být pouze vytvoření datového zdroje dostupných léčivých přípravků v České republice. Mým cílem je centralizovat datové sady dostupné o léčivých přípravcích v České republice a nalézt vhodný způsob, jakým by bylo možné tuto databázi integrovat se zahraničními datovými zdroji. Pokud by se podařilo nalézt vhodný identifikátor propojení Českých datových sad se zahraničními datovými zdroji, bylo by teoreticky možné tuto datovou sadu dále rozšiřovat o mnoho dalších datových sad. Možnosti rozšiřování, jak jsem deklaroval v předchozí kapitole, jsou obrovské. Jenom U. S. National Library spravuje kolem 291 zdrojů.

V rozsahu této diplomové práce není možné integrovat velké množství datových zdrojů, ale spíše nastínit způsob, jakým by bylo možné toho dosáhnout a zároveň poskytnout nástroje a datové sady. Základním parametrem nově vzniklé datové sady a vůbec způsobu jejího použití by měla být další rozšiřitelnost s co nejmenšími prostředky. Ideálním takovým případem by bylo, kdyby nebylo zapotřebí pro další účely provádět s datovou sadou žádné další extrakce a transformace.

4.1 Způsob uchovávání dat

Z požadavků, které vycházejí ze zadání mé diplomové práce a které jsem blíže specifikoval výše, jsem se pokusil nalézt vhodný způsob pro vytvoření modelu. Zjednodušeně jsem se nejdříve musel rozhodnout, zda zvolím klasický přístup uložení datových zdrojů v nějaké relační databázi nebo použiji některý z novějších přístupů, jako jsou grafové databáze nebo key value databáze.

Z tohoto rozhodnutí pak bude vycházet to, jakým způsobem budu přistupovat k vytvoření datovému modelu. Zatímco relační principy směřují především k přesné specifikaci celého modelu předem, NoSQL pojetí vede spíše k volnějšímu přístupu ke schématu.

Koncept relačního schématu vychází z detailního popsání a modelování před samotným využíváním databáze. Je nutné detailně identifikovat jednotlivé entity, které mají být do daného schématu ukládány a zejména určit vztahy mezi jednotlivými entitami. Pozdější úpravy relačního schématu mohou být velmi náročné, zejména kvůli jasně definovaným vztahům mezi jednotlivými entitami. Návrh relačního schématu jasně vede k detailnímu vymodelování současného stavu. Nemá tedy moc prostředky na to se adaptovat nově vzniklým okolnostem, které je zapotřebí do schéma zanést.

Vytvoření relačního schématu pro databázi léků s ambicemi integrace dalších datových zdrojů je velmi obtížné. Bylo by zapotřebí zanalyzovat a pochopit problematiku nejenom kolem léků, ale také jejich chemického složení a návaznosti na problematiku z oblastí lékařství, farmakologie, chemie. Samozřejmě pokud se snažíme modelovat nějakou část reality, vždy se pouze snažíme přiblížit dané realitě co nejblíže. Bylo by tak možné vytvořit relační model pouze pro určitou část. Nicméně vzniklá datová sada by měla být připravena integrovat různé další transdisciplinární datové sady. Zde pak můžou vznikat konflikty ve vytváření jednotlivých vztahů mezi entitami a mohlo by být obtížné udržet třetí normální formu.

Dalším problémem, kvůli datové správnosti a zároveň kvůli udržení třetí normální formy, je nutnost detailně analyzovat každý atribut dané entity. Je zapotřebí u každé datové sady zjistit výskyt hodnot, a podle toho určit daný datový typ.

Druhou možností, jak daná data uchovávat, je využití některého z principů NoSQL databází. Pojem NoSQL databáze označuje hned několik různých přístupů jak uchovávat data. Prvním přístupem jsou key- value data-

báze, kdy se ukládají pouze páry klíč hodnota. Dále existují dokumentové databáze, které ukládají jednotlivé položky jako objekty, dokumenty, podobné json objektům. Posledním možným principem je grafová databáze, kde modelovanou realitu reprezentuje graf.

Principy ukládání do dokumentové nebo key - value databáze se pro případ vytvoření rozsáhlého datového úložiště nehodí. Důvodem je, že tyto přístupy jsou dle anglického výrazu "schemeless", nebo-li bez předepsaného schématu. To představuje velký problém, protože kromě toho, že potřebujeme data ukládat, je zapotřebí zároveň uchovávat i metadata o tom, co znamenají.

Tuto možnost však implementují grafové databáze. Zejména pak RDF úložiště, které jsem blíže popsal v první kapitole. Díky tomuto úložišti bude možné zároveň docílit neomezené rozšiřitelnosti, protože jednotlivé entity si můžeme představit jako uzly grafu. Při nalezení správných identifikátorů není problém spojit jednotlivé uzly dvou zdánlivě nesouvisejících grafů.

Zároveň tyto úložiště mají silný nástroj k vytváření metadat v podobě vytváření ontologií pomocí OWL. Tyto principy jsou blíže vysvětleny v první kapitole. Nicméně je možné jednoduše rozšiřovat stávající datovou sadu v podobě grafu o další datové sady bez ztráty metadat. Zároveň je velmi jednoduché sdílet i jednotlivé ontologie napříč různými datovými sadami.

Rozhodl jsem se vytvořit datový model pomocí principů RDF a popsání ontologií potřebných ontologií pomocí OWL. Tento přístup by měl zaručit jednoznačný popis datové sady a zároveň efektivní možnosti v rámci rozšiřování původní datové sady.

4.2 Výběr vhodných datových sad

V předchozí sekci jsem deklaroval způsob jakým budu vytvářet model databáze. V této pasáži popíši datové zdroje, které jsem vybral pro tvorbu základní datové sady. Tedy datové sady, které počítají s další použitelností a rozšiřitelností.

Jako hlavní datový zdroj pro léčivé přípravky dostupné v České republice jsem zvolil prakticky jedinou možnou variantu. Touto variantou je katalog otevřených dat SÚKL. Tuto datovou sadu jsem blíže popsal v předcháze-

jící kapitole. Jedná se o 34 souborů ve formátu csv. Zároveň jsou tato data opatřena metadaty v podobě popsání datového rozhraní k databázi léčivých přípravků.

Pro vyhledávání aktuálních informací o cenách léčivých přípravků v České republice jsem zvolil integraci s portálem Ministerstva zdravotnictví České republiky, které poskytuje aktuální informace právě o cenách jednotlivých léčivých přípravků a výši případného doplatku hrazených z veřejného zdravotního pojištění.

Otevřená data SÚKL, jak jsem popsal v předchozí kapitole, obsahují také informace o složení daných léčivých přípravcích, ATC skupině a dalších vlastnostech léků. Mým cílem bylo integrovat do nově vytvořené datové sady také další data, z nichž by bylo možné vyvozovat důležité informace, například o účinku daných léků.

K rozšířenému pohledu na jednotlivé léčivé přípravky jsem hledal potřebný identifikátor, pomocí kterého bych mohl integrovat další zejména zahraniční datové sady. Cílem bylo nalézt datové sady, které by přinesly lepší vhled do problematiky farmakologických a chemických účinků jednotlivých léčivých látek. Jako příklad mohu uvést nalezení interakcí mezi léčivou látkou daného léčivého přípravku s dalšími látkami.

Rozšířená data o léčivých látkách by pak díky nově vzniklému grafu, který je reprezentován RDF úložištěm, mohla poskytovat další cenné informace o léčivých přípravcích. Jednou z takových zajímavých otázek je otázka různé účinnosti léčivých přípravků se stejnou účinnou látkou.

Problémem byla především různorodost jednoznačných identifikátorů jednotlivých datových sad, protože většina datových sad používala vlastní sadu identifikátorů rozdílných od identifikátorů používané v sadě otevřených dat poskytované SÚKL.

Nakonec jsem zvolil integraci jednotlivých léčivých látek prostřednictvím identifikátoru anatomicko-terapeuticko-chemické klasifikace léčiv. Tento identifikátor je spravován a vytvářen Světovou zdravotnickou organizací Who a slouží mimo jiné k porovnávání léčivých látek na mezinárodní úrovni. ATC skupina daného léčivého přípravku totiž ve svém nejvyšším stupni jasně identifikuje léčivou látku, která je v daném léku použita.

Jako datový zdroj, který jsem integroval na základě anatomicko-terapeuticko-chemické klasifikace, jsem připojil službu RxNav, která je spravována U. S.

National library of medicine. Tento zdroj jsem blíže popsal v předcházející kapitole. Rozhodnutí pro integraci právě tohoto datového zdroje jsem zvolil na základě několika kritérií. Služba RxNav API poskytuje sama o sobě velké množství dat týkajících se léčivých přípravků, látek, jejich chemického složení, účinků, interakcí a mnohých dalších. Zároveň poskytuje rozhraní pro namapování dalších identifikátorů datových zdrojů, například datové sady Drugbank. Dalším důležitým kritériem byl formát poskytovaných dat. Datová sada je dostupná prostřednictvím REST API a poskytuje data ve formátu JSON. Formát JSON je velmi dobrým pro parsování dat, což je pro případ integrace datové sady ideální. Celé API je také velmi dobře popsáno na portálu RxNav API, což znamená, že je možné vytvořit potřebné ontologie.

5 Návrh řešení a architektury

V předchozích kapitolách jsem popsal dostupné datové zdroje a prostředky jakými budu daná data uchovávat. V této kapitole popíši konkrétní způsob implementace. Obecně bylo mým cílem vytvořit datové úložiště, které poskytuje informace o léčivých přípravcích dostupných v České republice, zjišťování interakcí dalších látek s danými léčivými přípravky, informace o aktuálních cenách léčivých přípravků, dokumentech související s léčivými přípravky. Dalším cílem bylo vytvořit možnost rozšiřitelnosti daného zdroje a zároveň možnosti využití tohoto datového zdroje pouze přes rozhraní.

Implementační část práce se tedy skládá ze tří logických částí. První částí je vytvoření RDF úložiště. V této části bylo zapotřebí extrahovat data ze stávajících zdrojů. Poté bude následovat transformace těchto dat do potřebné podoby. Z výsledků těchto transformací by bylo možné vytvořit příslušné ontologie a nakonec vytvořit RDF dokumenty, které je možné uložit do vybraného úložiště.

Druhou implementační částí je vytvoření webového serveru, který se primárně stará o komunikaci s RDF úložištěm. Tento webový server vytváří dotazy SPARQL, kterými se dotazuje RDF úložiště na potřebná data. Další funkcionalitou je integrace dalších možných datových zdrojů na aplikační vrstvě. Zároveň tento webový server bude zajišťovat komunikaci s klientskými aplikacemi. To znamená nejen sestavování a transformace výsledků z dotazů do RDF úložiště a dalších REST API, ale také bude poskytovat služby zajišťující například autentizaci uživatelů jednotlivých systému. Tento webový server bude mít vlastní provozní databázi, kde bude uchovávat potřebná data například právě o autentizaci jednotlivých uživatelů. Komunikace pomocí s tímto webovým serverem bude probíhat prostřednictvím rozhraní REST API, které bude vracet výsledky ve formátu JSON. Tento univerzální formát jsem zvolil zejména kvůli výhodám jednoduchého strojového zpracování vrácených výsledků. Tento server je postaven tak, aby byl schopen zajišťovat provoz různých dalších klientských aplikací ať už různých desktopových, mobilních či webových aplikací.

Třetí částí mé implementace diplomové práce bude ověření funkcionality RDF úložiště a webového serveru. Kvůli demonstraci nejen stanovených cílů mé práce, ale i současných možností vývoje webových aplikací se bude jednat o čistě klientskou aplikaci funkcionálně nezávislou na webovém serveru.

Klientská aplikace tedy nevyužívá robustního serverového řešení, jako je to například u webových aplikací vytvořených pomocí architektury JSP (technologie pro vývoj dynamických HTML stránek) + servlety nebo například standardní architektury PHP aplikací využívající framework Nette. Bude se jednat o čistě klientskou aplikaci, která bude řešit především vizualizaci dat.

Podobnou architekturu popisuje Tim Bernes Lee ve svém článku The Semantic web [13], kde předpovídá vývoj webových aplikací směrem k propojeným zdrojům dat prostřednictvím RDF a následné zpracování jednotlivých výsledků pomocí agentů [13] - klientských aplikací.

5.1 Zvolení vhodných technologií

Po deklarování cílů mé diplomové práce bylo zapotřebí zvolit odpovídající technologické prostředky pro jejich dosažení. Výběr použitých technologií se skládal z výběru vhodného nástroje nebo programovacího jazyka pro extrakci datových zdrojů, jejich transformaci a vytvoření RDF dokumentů. Dále jsem musel nalézt vhodné RDF úložiště. Dále jsem musel nalézt vhodnou technologii pro vytvoření webového serveru a jeho provozní databázi. A nakonec jsem hledal vhodný prostředek pro vytvoření klientské webové aplikace, kde budou prezentovány výstupy a zajišťována interakce s uživatelem.

Před začátkem implementace jsem si určil cíl co možná největší nezávislosti jednotlivých aplikací, tedy: webového serveru, klientské aplikace a databázového serveru. Chtěl jsem demonstrovat přístup k vývoji moderních webových aplikací s podobným přístupem, jako byla představa Tim Bernes Lee, tedy oddělovat webové služby, které data zpracovávají, vyhodnocují od klientských aplikací, které je zpracovávají. Samozřejmě nelze hledat naprosto přesnou paralelu s tím co popisoval Tim Bernes Lee, protože jeho popis hovoří o integraci z mnoha různých zdrojů od osobních po veřejně dostupné služby. V rámci rozsahu diplomové práce bylo možné dosáhnout pouze omezeného počtu integrací datových zdrojů.

Další aspekt, který určoval použité technologie byla komunikace. Jednotlivé webové služby mezi sebou komunikují. Bylo tedy zapotřebí zvolit vhodný komunikační protokol. Zde jsem určil, v rámci standardů vývoje webových aplikací, že komunikace mezi jednotlivými aplikacemi bude probíhat prostřednictvím REST API poskytující zdroje ve formátu JSON. Tento druh

komunikace byl zvolen vzhledem k možnostem vývoje nezávislých webových klientských aplikací ve smyslu jakým jsem deklaroval, tedy že bude nezávislá na nějakém robustnějším serverovém řešení.

Při výběru vhodného programovacího jazyka nebo jazyků, jsem musel zohlednit to, že by měl být schopen zpracovávat velké množství dat a provádět nad nimi další operace. Musel jsem také zhodnotit znovupoužitelnost zdrojových kódů na jednotlivých částech systému. Protože jednotlivé části systému budou určitě zpracovávat podobné entity.

V závislosti na uvedených okolnostech jsem se rozhodl serverovou aplikaci a aplikaci RDF builder implementovat v Node.js [42]. Webového klienta jsem se rozhodl implementovat pomocí javascriptového frameworku React.js [43]. V závislosti na uvedených technologiích jsem zvolil jako datové RDF úložiště Stardog [44]. Toto datové úložiště jsem zvolil vzhledem k tomu, že poskytuje knihovnu pro komunikaci přímo pro Nodejs. Ostatní datová úložiště neposkytovala oficiální knihovny pro komunikaci nebo nebyly v době tvorby rozhodnutí k dispozici. Nicméně architektura aplikace je navržena tak, že SPARQL dotazy jsou odděleny od zdrojového kódu. Díky tomu je možné vyměnit datové úložiště za jiné a pouze vytvořit vrstvu pro komunikaci s daným úložištěm.

5.1.1 Stardog

Stardog je datové RDF úložiště. Toto úložiště podporuje RDF 1.1 a OWL2. Toto datové úložiště je mimo jiné využíváno organizacemi jako Nasa, Viacom a nebo univerzitami jako Columbia College Chicago a The University of Arizona. Pro spuštění tohoto úložiště je zapotřebí Java.

Stardog poskytuje přehledné grafické rozhraní, pro vytáření jednotlivých databází a jejich spravování. Poskytuje knihovny pro jazyky Java, Node.js, Framework Spring, Groovy, Clojure. Kompletní dokumentace je dostupná na internetových stránkách Stardog [44].

Ve své diplomové práci využívám toto datové úložiště jako databázové RDF úložiště. Prostřednictvím webového serveru jsou pomocí knihovny Stardog zasílány SPARQL dotazy. Stardog pak vrací jako odpověď JSON objekt.

5.1.2 Node.js

Node js je asynchronní událostmi řízený Javascriptový runtime. Je vytvořen pro návrhy vysoce škálovatelných síťových aplikací. Je postaven na asynchronním zpracování událostí pouze na jednom vlákně oproti běžnějšímu více vláknovému přístupu, který využívají například PHP nebo Java webové aplikace [42].

V mé diplomové práci jsem Node.js použil pro vytvoření webového serveru, který komunikuje se Stardog úložištěm a dalšími webovými službami. Jako provozní databázi používá úložiště Mongo DB. Dále jsem využil Node.js pro proces vytváření RDF a OWL dokumentů.

5.1.3 Mongo DB

Mongo DB je open source dokumentová databáze. Jednotlivé záznamy - dokumenty uvnitř databáze jsou uloženy ve formátu velmi připomínající JSON objekty. Hlavními vlastnostmi této databáze je vysoký výkon, bohatý dotazovací jazyk, vysoká dostupnost, horizontální škálovatelnost. [45]

Ve své implementační části používám toto úložiště jako provozní databázi, kde ukládám například data o přihlášených uživatelích.

5.1.4 React.js

Jedná se o Javascriptovou knihovnu pro tvorbu uživatelských rozhraní. Knihovna odstiňuje programátora od přímé manipulace s DOM strukturou jako bylo zvykem například u javascriptové knihovny Jquery. Vytváří virtuální DOM, který následně porovnává s aktuální vykreslenou DOM strukturou. Na základě rozdílu mezi nimi pak překreslí DOM strukturu [43].

Pomocí této knihovny jsem vytvořil klientskou webovou aplikaci pro koncového uživatele. Díky tomu, že se jedná o čistě Javascriptovou aplikaci, která je stažena po požadavku přímo do webového prohlížeče, aplikace lépe reaguje na uživatelské interakce, protože se neprovádí zbytečný dotaz na server, který pouze změní pohled aplikace, tak jako je to například u klasických aplikací PHP.

5.2 Architektura

Na obrázku 5.1 je vidět výsledná architektura. Ústředním bodem celého systému je webový server. Ten komunikuje s RDF úložištěm, odkud získává na základě SPARQL dotazů výsledky. Toto RDF úložiště tedy poskytuje data o léčivých přípravcích v České republice, zároveň poskytuje informace o zdrojích, kde je možné nalézt aktuální ceny léčivých přípravků (portál Ministerstva zdravotnictví České republiky) a integruje další zahraniční datové zdroje.

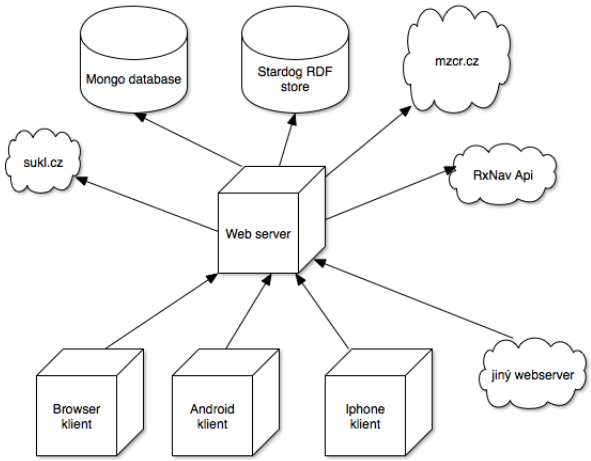
Webový server pak výsledky SPARQL dotazů vyhodnocuje a případně provádí další akce. Například pokud je požadováno po webovém serveru vrátit detailní data o léku Ibalgin, pošle se dotaz do databázového úložiště. Vrácený výsledek obsahuje data o daném léku a také o tom, kde je možné nalézt aktuální informace o ceně daného léku. Na základě těchto dat se vykoná požadavek na portál Ministerstva zdravotnictví České republiky, ze které se získají požadovaná data. Webový server pak transformuje získaná data a vytvoří z nich JSON objekt, který pošle jako odpověď.

Na obrázku je také vidět, že webový server využívá jako provozní databázi Mongo [45]. Do této databáze jsou ukládáni například přihlášení uživatelé a další data, které nesouvisejí s hlavním datovým zdrojem. Chtěl jsem tím ukázat trend současných webových aplikací a to, že není nutné mít pouze jeden centrální datový zdroj. Naopak se v mnohých případech stává výhodnější používat několik databázových systémů podle způsobu jejich využití.

V tomto konkrétním případě je to velmi užitečné. Odděleně vytvářím a spravuji hlavní databázový zdroj (RDF úložiště), vytvářím komplikované ontologie týkající se léčivých přípravků. Zatímco provozní Mongo databáze pracuje pouze s daty zajišťující funkcionalitu klientských aplikací.

Díky vlastnostem Mongo databáze je velmi snadné měnit schéma a model této databáze. Vzhledem k násobně nižší složitosti takovéto databáze by bylo zbytečné vytvářet ontologie pro popis uživatele (v případě, že bychom uvažovali o ukládání všech dat do centrálního RDF úložiště).

Pro ilustraci využití tohoto webového serveru je na obrázku 5.1 zakresleno několik klientů jako je Android aplikace, Javascriptová aplikace v prohlížeči nebo Iphone klient. Je možné také propojit tento webový server s jiným webovým serverem, který bude využívat pouze část funkcionality.



Obrázek 5.1: Architektura webové aplikace

Pro demonstraci principů napojení klientské aplikace jsem vytvořil pouze klientskou aplikaci běžící v prohlížeči, kterou jsem pojmenoval Drugie. Nicméně stejné principy komunikace s webovým serverem by fungovaly i pro ostatní platformy.

Aplikace Drugie v mé diplomové práci je určitým ověřením, že daná architektura funguje. Je zde proto implementován pouze omezený počet případů užití.

6 Implementace

V této části diplomové práce popíši konkrétní implemetaci jednotlivých aplikací. Stručně popíši principy, nástroje a algoritmy se kterými jsem pracoval.

6.1 RDF Builder

Aplikace RDF builder je konzolovou aplikací vytvořenou pro potřeby tvorby RDF úložiště. Její hlavním úkolem je extrahovat data z daných zdrojů, transformovat a následně vytvářet předepsané ontologie a RDF dokumenty. Aplikace je, jak jsem již popsal výše, postavena na technologii Node.js.

Potřeboval jsem nástroj, pomocí kterého bude možné v co možná nejkratší době vytvářet poměrně velké (některé dokumenty mají i více než 200 MB) RDF dokumenty. Dalším důležitým kritériem bylo dosáhnout vysoké flexibility při tvorbě ontologií a RDF dokumentů. Tím mám především na mysli, aby bylo možné vytvářet jednotlivé dokumenty v přehledné formě - xml, turtle...

Ve složce jsou data uložena vyexportovaná data z katalogu otevřených dat SÚKL. Jednotlivé verze exportů jsou ukládány do složek ve formátu $DLP < Y >< m >< d >$, kde Y představuje rok, m měsíc a d den exportu dat.

Na ukázce adresářové struktury 6.1 je vidět obecná struktura aplikace uložené v adresářové struktuře pod názvem RDF builder. Ve složce app jsou uloženy zdrojové kódy, které souvisejí přímo s danou problematikou. Jsou zde tedy různé controllery a také modelové třídy. Ve složce modules jsou obecné knihovny, které přímo nesouvisejí z danou problematikou a jsou znovupoužitelné. Jedná o knihovnu pracující se soubory, se šablonovacím systémem a nakonec pomocná knihovna pro opakující se funkce. Složka node_modules obsahuje použité externí knihovny. Ve složce temp jsou pak uloženy výsledky daných transformací. Nakonec jsou zde soubory config.js, které slouží jako nastavení dané aplikace, index.js pro spouštěcí skript celé aplikace a nakonec package.json, kde jsou uloženy všechny závislosti na externí knihovny potřebné v dané aplikaci.

V závislosti, že je zapotřebí zpracovávat desítky dokumentů, které po-

```
data
└── sukl
    └── DLP20160104
        ├── dlp_lecivepripravky.csv
        └── *.csv
RDF builder
├── app
│   ├── presenters
│   │   └── templates
│   │       ├── dlplecivepripravky
│   │       │   ├── defaulthtml
│   │       │   └── rdf.html
│   │       ├── dlp_latky
│   │       └── ...
│   ├── dlp_lecivepripravky_presenter.js
│   ├── dlp_latky_presenter.js
│   ├── rxnav
│   └── sukl
├── modules
│   ├── template_system
│   └── utils
├── node_modules
├── temp
├── config.js
├── index.js
└── package.json
```

Zdrojový kód 6.1: Adresářová struktura projektu

cházejí z různých zdrojů, jsou různě strukturované, jsem vytvořil spouštěcí mechanismus pro vytváření konkrétního RDF dokumentu a související ontologie. Všechny dokumenty jsou vygenerovatelné prostřednictvím příkazu node index.js a název příslušného datového zdroje. Pro případ vytvoření RDF dokumentu léčivých přípravků vypadá spouštěcí příkaz takto:

```
$ node index.js dlp_lecivepripravky
```

Obecné fungování RDF builderu je vidět na sekvenčním diagramu 6.1. V index.js se zpracují předáváné parametry aplikace. Následně se zavolá metoda router v app.js. App.js rozhoduje podle vstupních parametrů, který z presenterů zpracuje volanou akci. Daný presenter pak extrahuje a transformuje

data a vytvoří z nich pomocí model mapperu objekt, který představuje danou entitu. Na základě tohoto modelu se pak pomocí šablonovacího systému vytvoří RDF dokumenty.

6.1.1 Moduly

V rámci RDF builderu jsem musel vytvořit několik modulů. Moduly jsou v prostředí Node.js knihovny. Jednou z výhod využívání právě tohoto prostředí je velká databáze knihoven. Tyto knihovny jsou dostupné prostřednictvím balíčkovacího nástroje npm [46].

Obvyklá architektura Node.js aplikací nestaví na využití robustních frameworků, jako je tomu známo například z prostředí Javy (Spring, Wicket) nebo PHP (Nette, Symfony, Laravel). Většina aplikací je spíše sestavena z různých na sobě nezávislých knihoven. Výsledná aplikace, tedy kvůli menší funkcionalitě, nevyužívá nějaký robustní framework, ale je složena z množství různých modulů.

V rámci zachování tohoto přístupu jsem se snažil využít některé externí knihovny. Nicméně bylo zároveň zapotřebí kvůli konkrétnímu případu užití vytvořit vlastní moduly, které redukují duplicitní kód aplikace a zvyšují její čitelnost.

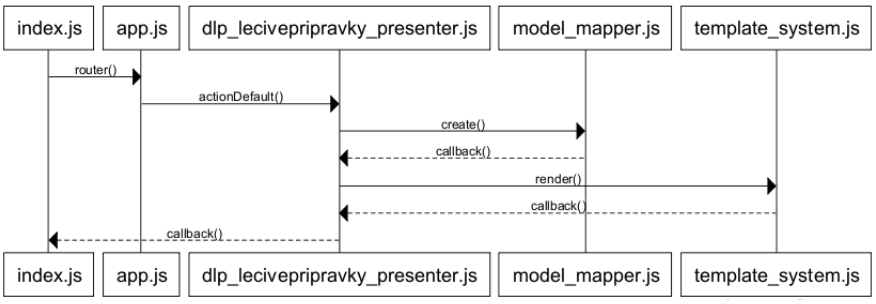
Pro aplikaci RDF builder jsem vytvořil tři moduly: šablonovací systém, práci s kódováním souborů a nakonec javascriptovou knihovnu pro běžné využití. Tyto vytvořené moduly nejsou přímo závislé na konkrétní implementaci aplikace a jedná se vždy prakticky o funkce, které zpracovávají nějakým způsobem vstup.

Modul se šablonovacím systémem vznikl z potřeby vytváření velkých RDF souborů. Tyto soubory mají až desetitisíce řádků, proto by bylo časově velmi náročné tyto dokumenty tvořit manuálně. Zároveň při změnách nebo chybách ve schématu by byly velmi komplikované i úpravy. Daný modul má za úkol načíst soubor, kde je uložena šablona daného RDF dokumentu a uložit vygenerovaný výstup do nového souboru. K samotnému šablonování je využití již existujícího šablonovacího nástroje Mustache [47].

Modul pro změnu kódování souborů vznikl z potřeby měnit kódování souborů. V rámci vytváření RDF dokumentů jsem se setkal s nekonzistentním uložením původních datových sad. Proto bylo zapotřebí všechny datové

zdroje sjednotit na jednom kódování. Jako standardní kódování do kterého jsem převáděl všechny soubory je UTF-8. Tento modul tedy obsahuje pouze jednu funkci, která má na vstupu cestu k původnímu dokumentu, cestu nově vytvořeného dokumentu s novým kódováním, původní kódování a cílové kódování.

Posledním modulem, který jsem v rámci této aplikace vytvořil, je jsutils. Jedná se pouze o malou knihovnu poskytující funkce zkracující některé mnou používané výrazy v javascriptu. Například testování zda daná proměnná je pole.



Obrázek 6.1: Sekvenční diagram fungování RDF builder

6.1.2 Model

Pro potřeby vytváření jednotlivých RDF a dokumentů a ontologií jsem pro lepší práci s daty vytvořil model. Místo toho, abych například extrahovaná data ze souboru csv uchovával v dvourozměrném poli, využívám mapování jednotlivých řádků csv souboru na mnou vytvořenou entitu. Rozdíl ve využití je vidět na ukázce kódu 6.2. Řešení při vytváření dokumentu za použití pouze pole surových dat je nečitelné. Existuje zde daleko větší riziko chybovosti vzhledem k tomu, že daný atribut identifikuje pouze číslo. Zatímco díky využití namapovaného objektu je výsledná šablona daleko čitelnější.

Tato modelová vrstva je tvořena vždy dvěma soubory. První reprezentuje samotnou entitu a definuje datový obal. Zároveň tato entita zaručuje správnost ukládaných dat. Jako příklad mohu uvést validaci na objektu reprezentující ATC skupinu. ATC skupiny tvoří hiearchickou skupinu, kdy kromě prvního stupně kategorie ATC mají všechny ostatní ATC nějakého předka. Tento předek lze zjistit přímo z literálu kódu příslušící dané ATC skupině,

```
<!-- pole -->
<dlp_lecivepripravky:vydej resource="{csvLine[5]}" />

<!-- objekt -->
<dlp_lecivepripravky:vydej resource="{lecivy_pripravek.vydej}" />
```

Zdrojový kód 6.2: OWL ukázka

protože její kód má předepsanou strukturu. Nicméně jsem zjistil, že u některých kódů nelze určit přímo daného předka, protože tato ATC skupina v datech poskytovaných SÚKL neexistovala.

U každé takovéto entity jsem také definoval mapper, který má za úkol pole dat namapovat na danou entitu. Tímto přístupem jsem odstínil samotné datové objekty od logiky vytváření jejich instancí. Tyto datové obaly je poté možné dále využít pro jiné potřeby v rámci aplikace.

6.1.3 Presentery

Pro vytváření každého RDF dokumentu a jeho ontologií existuje jeden presenter, který řídí toto zpracování. V mé konkrétní implementaci má každý presenter dvě metody. Presenter totiž řídí vytváření RDF dokumentu a ontologií právě pro jeden datový zdroj, soubor. Jedna metoda zajišťuje vytvoření RDF dokumentu a druhá metoda danou ontologii. Obecně presenter představuje agregaci všech akcí potřebných k vytvoření příslušného dokumentu.

Nejdříve daný presenter získá surová data, která namapuje na určitou entitu. Výsledkem je pole objektů, které jsou předány šablonovacímu systému, který zajistí samotné vytvoření daného dokumentu. Během implementace jsem narazil na problém s velikostí vytvořených souborů. Databáze Stardog totiž přes své webové rozhraní přijímá pouze soubory omezené velikosti. V rámci presenterů jsem tedy musel ošetřit případ, kdy se vytváří velký dokument (řádově ve stovkách MB). Omezil jsem tedy počet možných zapsání entit do jednoho souboru na 4000. Při tomto omezení se pak generovaly soubory o maximální velikosti řádově několik MB.

6.1.4 Šablony

Každá z šablon funguje pro vytvoření jednoho RDF dokumentu. Pro zápis RDF jsem si vybral klasický xml zápis. Výhodou tohoto zápisu je především lepší podpora kontroly správnosti zápisu přímo ve vývojových prostředích. To mi při zpracování všech datových zdrojů velmi pomohlo k minimalizaci chybovosti v daných šablonách.

Tyto šablony využívají jako šablonovací jazyk knihovnu Mustache [47]. Každé šabloně tak předám pole vytvořených objektů z presenteru. Zápis jednotlivých atributů objektů do šablony pak provádím dle zápisu na ukázce kódu 6.2.

6.2 Webový server

Webový server má dle předchozích kapitol za úkol dělat prostředníka mezi datovými zdroji a klientskými aplikacemi. S klientskými aplikacemi komunikuje prostřednictvím REST API. Tento webový server je postaven na minimalistickém frameworku Express [48], který poskytuje nástroje pro vytváření REST API.

```
/
├── mongo_model
├── node_modules
├── routes
├── sparql
├── stardog_api .2 app.js
└── package.json
```

Zdrojový kód 6.3: Adresářová struktura projektu

Adresářová struktura 6.3 projektu je podobná jako u jiných aplikací vytvořených pomocí frameworku express. Hlavním místem aplikace je soubor app.js. Jedná se o konfigurační místo celého projektu. Je zde nastavení, připojení do databáze, základního routování a nastavení různého midleware.

Jednotlivé zpracování dotazů na server je pak prováděno prostřednictvím skriptů ve složce routes. Tyto skripty jsou registrovány v app.js pod nějakou url adresou. Dále aplikace obsahuje modul stardog_api, který komunikuje se

Stardog databází, zasílá jí dotazy v jazyku SPARQL. Tato aplikace je napojena na provozní databázi Mongo DB. Definice schémat související s modelem Mongo databáze jsou uloženy v adresářové struktuře ve složce mongo_model.

6.2.1 Stardog api

Tento modul jsem vytvořil kvůli potřebě oddělit SPARQL dotazy od logiky aplikace. Kvůli oddělení SPARQL dotazů jsem vytvořil modul query_loader.js, který zajišťuje načtení požadovaného souboru, ve kterém je uložena šablona k SPARQL dotazu, a následně na základě vstupních parametrů zkompletuji požadovaný dotaz. Ke přímé komunikaci s databází Stardog je využito vytvořené knihovny dostupné a dokumentované na portále Stardog [44].

Zároveň tento modul poskytuje při jeho použití lepší a srozumitelnější rozhraní. V rámci diplomové práce je implementován pouze jeden logický celek týkající se léčivých přípravků. Nicméně při rozšiřování množství služeb, které server poskytuje je modul připraven na další rozšíření. Příklad použití rozhraní tohoto modulu je vidět ve zdrojovém kódu 6.4.

```
stardog_api.dlp_lecivepripravky().findById(id, callback);
```

Zdrojový kód 6.4: Použití Stardog api

6.2.2 Mongo model

Pro uchovávání provozních dat, jako jsou uživatelské účty aplikace jsem použil databázi Mongo. Ke komunikaci s touto databází jsem použil knihovnu Mongoose [49]. Jelikož jsou jednotlivé dokumenty ukládány v Mongo databázi ve struktuře velmi podobné formátu JSON, je tato databáze velmi vhodná v kombinaci s použitím prostředí Node.js. Javascriptové prostředí totiž poskytuje velmi dobré nástroje pro zpracovávání objektů. Používání objektů pocházející z Mongo databáze se tak stává velmi pohodlné, protože není zapotřebí robustních nástrojů pro objektově relační mapování. Lépe vysvětlím na příkladu vytvoření schéma.

Ve zdrojovém kódu 6.5 je vidět vytvoření schéma uživatele. Tímto způsobem je možné prostřednictvím knihovny Mongoose definovat schéma uživa-

```
var userSchema = new Schema({
    username: {
        type: String,
        unique: true
    },
    firstName: String,
    lastName: String,
    password: String,
    accessToken: {
        type:String,
        index: true
    },
    pharmacy: [{
            type: Schema.Types.ObjectId,
            ref:DRUG_COLLECTION_NAME
    }],
});
```

Zdrojový kód 6.5: Mongo DB schéma

tele včetně vazeb M:N. Zde je konkrétně M:N vazba reprezentována atributem pharmacy, který definujeme jako pole hodnot ID schématu léků.

6.2.3 REST API

Webový server poskytuje REST API, prostřednictvím kterého mohou se serverem komunikovat klientské aplikace. Pro fungování aplikace jsem se snažil udělat strukturu url adres, na kterých jsou dostupné jednotlivé POST a GET požadavky. Tato struktura je dělena tak, aby přístup k logicky podobným zdrojům a akcím byl dostupný pod stejnou adresou.

Základní funkce pro získávání dat ze Stardog databáze jsou dostupné na url adrese "/dlp_lecivepripravky". Tato url adresa slouží jako rozhraní pro pokládání dotazů do databáze. Přístup na tuto url adresu není nijak zabezpečen. Na této url adrese je možné na url "/findAllByNamePattern"získat seznam všech léků obsahující hledanou sekvenci písmen. S tím souvisí metoda "/countByNamePattern", která vrací počet pozitivních výsledků hledání sekvence písmen v názvu léčivých přípravků. Url adresa "/findAllSimilar-

DrugByATC"poskytuje seznam všech podobných léků. Dále API poskytuje službu pro získání složení daného léku.

Na další url adrese jsou dostupné zdroje z RxNav API (díky integraci identifikátoru z RxNav na datové vrstvě). V rámci diplomové práce a aplikace Drugie jsem implementoval pouze jednu metodu a to "/findInteractionById", kdy je na vstupu očekáván identifikátor léčivého přípravku z databáze SÚKL. Na základě dotazu do Stardog databáze je zjištěn identifikátor v rámci databáze RxNav a poté je zaslán dotaz přímo do API RxNav. Na dané url jsou dostupné látky, které interagují s daným léčivým přípravkem.

Součástí REST API je také možnost získat informace o cenách a výši případného doplatku léčivých přípravků hrazených z veřejného zdravotního pojištění. Po zaslání požadavku na url "/prices"s identifikátorem definující daný léčivý přípravek jsou uživateli zaslány aktuální informace o cenách daného léčivého přípravku. Toho je dosaženo strojovým zpracováním příslušného internetového zdroje, který známe díky uložení v RDF databázi, v reálném čase. Zašle se požadavek na příslušnou stránku a následně jsou odsud vytažena potřebná data, která jsou transformována do odpovědi ve formátu JSON.

Další API souvisejí s přihlášenými uživateli do aplikace. Základem je tedy url adresa "/users". Na této adrese je možné registrovat, přihlásit a autorizovat uživatele. Zabezpečení funguje na jednoduchém principu. Uživatel se prostřednictvím url "/register"může do systému zaregistrovat. Po přihlášení prostřednictví, "/login", pokud jsou přihlašovací údaje validní, je uživateli zaslán přístupový kód, token. Tento současný přístupový token je zároveň uložen v databázi. Pro přístup k zabezpečeným zdrojům v aplikaci pak uživatel používá právě tento token. Tím se minimalizuje počet zaslaných uživatelských jmen a hesel skrz internet. Nicméně hesla se nezasílají z klientských aplikací v čisté formě, ale je použit hashovací algoritmus. Díky tomu, že se daný token obnovuje po každém přihlášení se zvyšuje bezpečnost. Samozřejmě je možné implementovat v rámci bezpečnosti další ochranné mechanismy jako párovat daný token s IP adresou klienta, požadovat po klientské aplikaci obnovu tokenu po nějakém časovém úseku a další. V uživatelské části webového serveru je v současné době implementována možnost hodnocení jednotlivých léčivých přípravků a přidání komentáře.

6.2.4 Zpracování požadavků na webovém serveru

Hlavní rozdíl mezi programováním klasického webového serveru, například za pomocí ve PHP, je ve zpracování požadavků serverem. Zjednodušeně, klasický server na kterém běží PHP aplikace, vytváří automaticky nové vlákno, pro každý nový požadavek, na kterém je požadavek zpracován. Zatímco server Node.js přijímá všechny požadavky v jednom vlákně. Pokud je to zapotřebí (například dotaz do databáze), je použito neblokujících workerů, kde se mohou vytvářet další vlákna dle potřeby. Nejde však o vytváření nového vlákna nebo procesu pro každý zpracovávaný požadavek [42].

Proto se v programování v prostředí Node.js velmi často používá asynchronních volání, které využívají anonymní funkce. Tyto funkce jsou spuštěny, a pokud je to zapotřebí vytvoří se pro jejich zpracování neblokující nové vlákno. V momentě kdy je daný proces dokončen, je pomocí callback funkce vrácena odpověď do hlavního vlákna. Prostředí Node.js odstiňuje programátora od vytváření nebo synchronizace vláken.

Pro správné fungování serveru je však nutné zajistit neblokující frontu událostí. Jak již jsem psal výše, je třeba využívat asynchronního volání pomocí anonymních funkcí. Problém pak nastává při serializaci těchto asynchronních požadavků. Zejména v případech, kdy jsou jednotlivá volání na sobě závislá.

Jako příklad v mé aplikaci mohu uvést proces přidání nového hodnocení, kdy je zapotřebí nejdříve získat z Mongo databáze přihlášeného uživatele. Poté se také z Mongo databáze zjišťuje, jestli daný lék již není vytvořen (bylo mu přiděleno nějaké hodnocení). Pokud není v Mongo databázi daný lék vytvořen, je zapotřebí získat jeho základní atributy ze Stardog databáze. Až poté je možné vložit nový lék (přidat stávájícímu nové hodnocení).

K práci s větším množstvím asynchronních funkcí a jejich serializaci využívám knihovnu async [50]. Tato knihovna umožňuje pracovat se sériově, tedy po sobě navazujícími funkcemi. To je případ, který jsem popsal výše. Také podporuje spuštění paralelních funkcí zároveň. Toho využívám především v RDF builderu u vytváření větších RDF dokumentů, kdy rozděluji dokumenty do několika souborů. Spustím tedy paralelně všechny volání na vytvoření nového souboru.

6.3 Aplikace Drugie

Aplikace Drugie představuje v mé diplomové práci ověření předsevzatých cílů, demonstraci funkcionality RDF úložiště a webového serveru. Zároveň tato aplikace může sloužit pro praktické využití nejen v lékařství, farmakologii nebo pro běžné uživatele hledající rozšířené informace o léčivých přípravcích.

Hlavní funkcionalitou je vyhledávání mezi léčivými přípravky na základě části názvu daného léčivého přípravku. Je možné získat detail daného léčivého přípravku. Tento detail léčivého přípravku je složen z několika datových zdrojů. Jsou zde informace pocházející přímo z datového zdroje SÚKL, dále jsou zde k dispozici důležité dokumenty o léčivém přípravku. Dále aplikace poskytuje informace o cenách a výši případného doplatku léčivých přípravků hrazených z veřejného zdravotního pojištění.

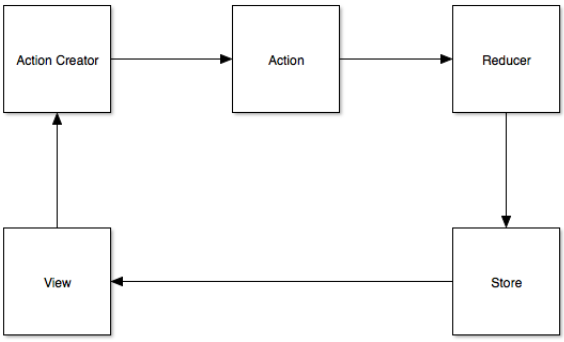
Na detailu daného léčivého přípravku je možné získat složení - seznam látek, které obsahuje. Detailní pohled na lék dále poskytuje návrhy podobných léčivých přípravků. V tomto pohledu je dostupný také seznam interakcí mezi dalšími léky.

Další funkcionalitou je uživatelská část aplikace. Uživatel se může registrovat, přihlásit. Na detailu léčivého přípravku je pak možné přidat hodnocení daného léčivého přípravku. Uživatel si také může ukládat léky do vlastního seznamu léků - lékárny.

Tato klientská aplikace je vytvořena pomocí knihovny React.js. V pomyslné MVC architektuře pomáhá tato knihovna vytvářet view. React vytváří abstrakci pomocí virtuálního DOM a nepracuje přímo s DOM stromem jako například knihovna jquery. Zároveň React implementuje jednosměrný tok dat, díky tomu se snaží být lépe pochopitelný [43].

K uchovávání a změně stavu aplikace používám knihovnu Redux.js [51]. Způsob jakým přistupuje k toku dat je implementace Flux architektury [52]. Celý proces začíná nějakou uživatelskou akcí z view. Může se jednat například jen o načtení dané stránky. Tato událost vyvolá nějakou akci, typicky se jedná o funkce. Poté co daná funkce vrátí požadovaný výsledek, předá se výsledek Reduceru. Ten tento výsledek zpracuje a uloží do Store, který představuje aktuální stav aplikace. Při změně stavu aplikace je pak příslušná část aplikace, která má být díky změně stavu změněna, znovu vykreslena.

Díky použití těchto knihoven vznikla následující adresářová struktura 6.6.



Obrázek 6.2: Flux

Ve složce assets a style jsou pomocné css soubory sloužící k definici vzhledu aplikace. Přes soubor index.html se spouští aplikace. Jedná se o klasický html soubor, který má v sobě odkaz na javascriptový soubor.

Všechny zdrojové kódy k této aplikaci jsou ve složce src. Jednotlivé zdrojové kódy jsou pak členěny dle výše popsané architektury do složek actions, components a reducers. Soubor index.js je spouštěcím a konfiguračním skriptem v aplikaci.

```
/
├── assets
├── node_modules
├── src
│   ├── actions
│   ├── components
│   ├── reducers
│   ├── services
│   ├── index.js
│   └── routes.js
├── style
├── index.html
└── webpack.config.js
```

Zdrojový kód 6.6: Adresářová struktura Drugie

Ve složce components jsou uloženy všechny skripty související s view aplikace. Jedná se o vizuální komponenty. Návrh architektury pomocí knihovny

React.js vede k rozdělování logiky do malých nezávislých komponent. Každá z komponent má vlastní stav na základě kterého se vykresluje. Komponenty mohou získat svůj stav prostřednictvím vstupních parametrů. Při použití knihovny Redux pak mohou získávat stav přímo ze stavu celé aplikace. Každá komponenta má definovanou metodu render, ve které se definuje vzhled prostřednictvím html nebo dalších komponent. Další metody pak obvykle reagují na nějakou uživatelskou akci, typicky kliknutí na odkaz, tlačítko.

Při použití knihovny Redux jsou tyto akce předávány akcím (Actions - viz. obrázek 6.2). Prostřednictvím této akce dojde k vytvoření nového stavu. To je typická vlastnost pro Flux architekturu. Jedná se o jednosměrný tok dat, proto nedochází ke redeklaraci proměnných objektu představující stav, ale dojde k vytvoření nového stavu. Samotné soubory ve složce actions obsahují pouze funkce. Žádné dvě z těchto funkcí by neměly být na sobě závislé. Tento návrh se využívá v Redux aplikacích kvůli snížení závislosti jednotlivých částí kódu na sobě. Návrh aplikací Redux je silně ovlivněn funkcionálním návrhem, čímž se snaží docílit odstraněním vedlejších účinků. Samotná funkce tak nemění stav aplikace jako takové. Každá funkce má a nebo nemusí mít vstupní parametry. Výstupem každé této funkce je objekt, který má dvě proměnné. První proměnnou je type, tato proměnná určuje název dané akce. Tento název je dále odchytáván reducerem. Druhou proměnnou je payload, náklad. Pomocí této proměnné objekt přenáší výsledek do reduceru.

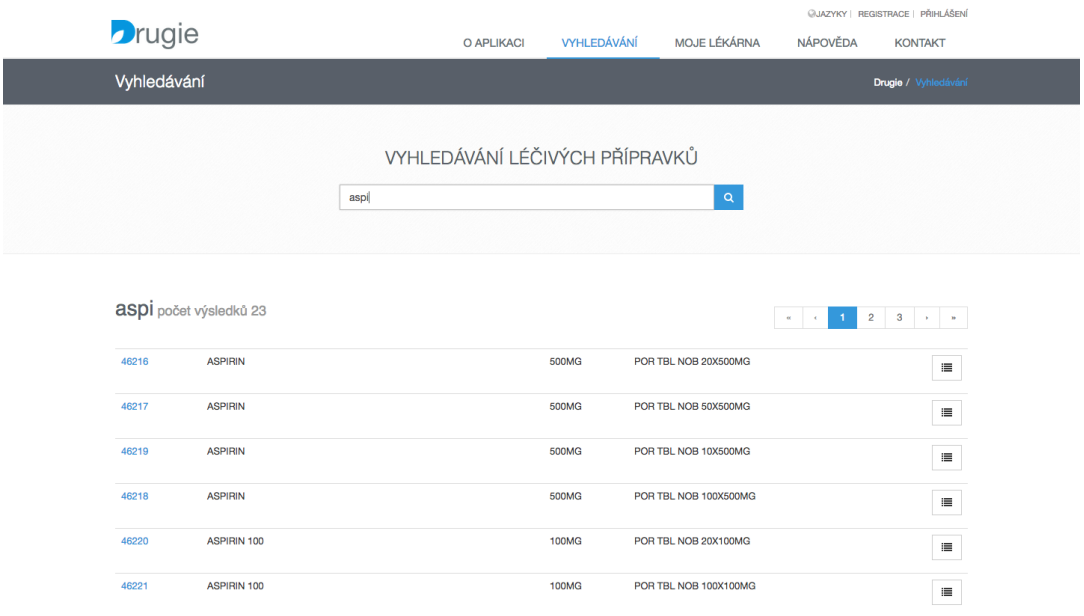
Ve složce reducers pak probíhá samotné zpracování nového stavu aplikace. Každý z reduceru reprezentuje jen jednu logickou část celkového stavu. Například v mé aplikaci používám 7 reducerů. Každý reducer je funkce, která má na vstupu stávající stav aplikace a nějakou akci, tedy objekt složený z type a payload proměnných. Na základě těchto vstupních parametrů pak reducer vytvoří nový stav aplikace.

6.3.1 Ukázka aplikace

Na obrázku 6.3 je vidět jedna z hlavních obrazovek aplikace. Na této obrazovce probíhá vyhledávání léků na základě shody části názvu. Vyhledávání léčivých přípravků se začne provádět při vložení prvních třech znaků názvu automaticky. Při každé další změně vyhledávacího dotazu se zašle požadavek na webový server, který zpět zašle požadovaný seznam léků. Zároveň se zasílá další nezávislý požadavek na server na celkový počet léčivých přípravků odpovídající hledanému výrazu.

Na této obrazovce je také možné listovat mezi nalezenými léčivými přípravky. Je zde uveden také počet výsledků, které vyhovují zadané části názvu. V tomto seznamu je dále možné se prokliknout na detail daného léčivého přípravku.

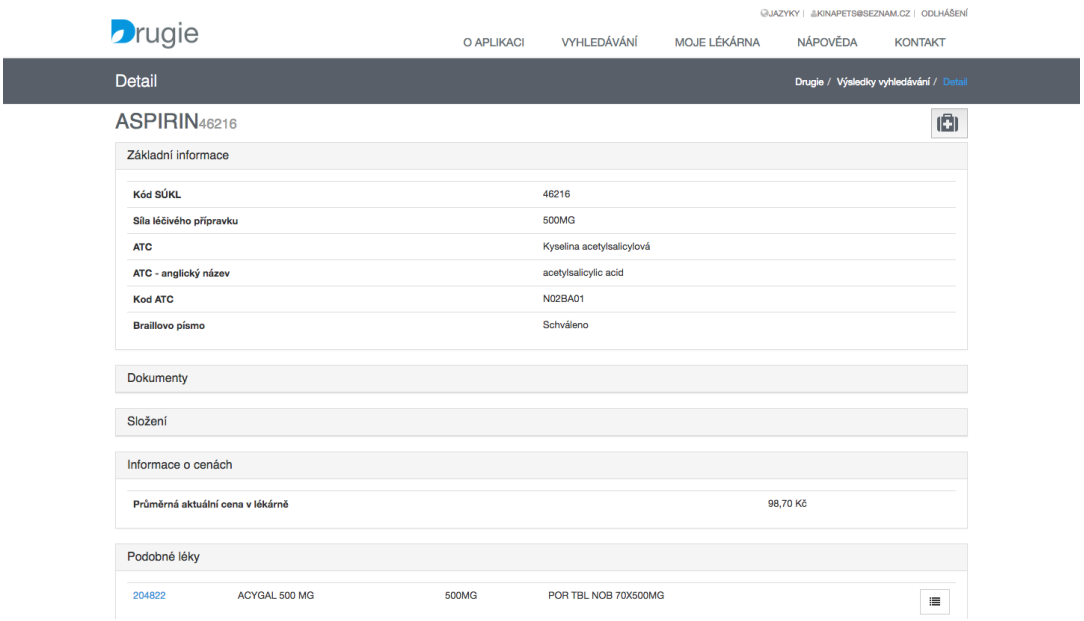
V aplikaci jsou implementovány dvě jazykové mutace - česká a anglická. Aplikace při přenastavení aktuálního jazyka automaticky překreslí všechny texty na aktuální jazykovou mutaci. Přepínání jazyků se nachází v pravém horním rohu aplikace. V tomto horním menu je dále možnost registrovat nebo přihlásit uživatele.



Obrázek 6.3: Ukázka aplikace Drugie

Další ukázková obrazovka je vidět na obrázku 6.4. Jedná se o obrazovku s detailem daného léčivého přípravku. V horní části obrazovky se zobrazují základní informace o daném léčivém přípravku. Pod tímto blokem jsou další bloky, které se rozbalí po kliknutí. Je zde blok informací o dokumentech o daném léčivém přípravku, o složení daného léku. Dále pak informace o aktuálních cenách léčivého přípravku a alternativy k léčivému přípravku. Posledními bloky je blok s informacemi o interakcích s daným léčivým přípravkem a blok s hodnocením a komentáři léčivého přípravku. V pravém horním rohu se zobrazuje tlačítko přihlášeným uživatelům pro přidání léčivého přípravku do vlastní lékárny.

Každý z těchto bloků představuje samostatnou React komponentu využitelnou mnohonásobně v aplikaci.



Obrázek 6.4: Ukázka aplikace Drugie

7 Ověření a diskuze

Diplomová práce se zabývá vytvořením databáze léčivých přípravků v České republice. Cílem bylo vytvořit databázi léčiv v České republice, kterou jsem následně integroval s dalšími datovými zdroji, zejména jsem se pokusil integrovat databázi se zahraničními datovými zdroji. Zahraniční datové zdroje totiž poskytují mnohem více datových zdrojů. Tyto zahraniční zdroje poskytují data o složení, interakcích, chemickém složení a mnohá další.

Mým dalším cílem bylo vytvořit architekturu na sobě co nejvíce nezávislých aplikací. Chtěl jsem demonstrovat architekturu aplikace, kde existuje sémantické úložiště, se kterým komunikuje webový server. S tímto webovým serverem pak mohou komunikovat různé klientské aplikace nebo další webové servery.

V praktické části práce jsem tak nejdříve implementoval potřebné procesy pro extrakci a transformaci dat z externích datových zdrojů a převedení do RDF dokumentů. Za pomocí těchto vytvořených RDF dokumentů jsem vytvořil RDF úložiště Stardog. S tímto datovým úložištěm pak pracuje mnou implementovaný webový server. Tento webový server jednak zasílá SPARQL dotazy do RDF úložiště, zároveň také integruje na aplikační vrstvě další datové zdroje. Posledním článkem je pak klientská aplikace Drugie. Tato klientská aplikace komunikuje prostřednictvím REST API s webovým serverem.

Všechny tyto tři aplikace byly napsány za pomocí jazyka Javascript. Zejména pak díky prostředí Node.js, který poskytuje běhové prostředí jak pro aplikaci RDF builder, tak pro webový server. Ve své diplomové práci jsem chtěl ukázat možný moderní přístup k vývoji webových aplikací. Nepoužil jsem žádné robustní architektury nějakého frameworku, které by vedlo spíše k vytvoření jedné velké aplikace namísto tří. Využil jsem možnosti využití obrovského množství javascriptových knihoven [46] a vytvořil jsem tak pro každou aplikaci speciální sadu nástrojů. Pro RDF builder jsem potřeboval nástroje pro vytváření RDF dokumentů, zatímco webový server potřeboval knihovny pro vytvoření REST API a komunikaci s RDF úložištěm. Aplikace Drugie pak využívá knihovny pro tvorbu uživatelských rozhraní, která je velmi vhodná pro vytváření klientských javascriptových aplikací komunikující se serverem na základě REST API.

Na základě této implementované architektury jsem ověřil její funkčnost a nastínil potenciální využití.

7.1 Současné zpracování a návrhy do budoucnosti

V této části bych rád popsal další možné navázání na mou diplomovou práci. Má diplomová práce se zabývá zejména vytvořením daného databázového úložiště a ověření jeho funkcionality. Využitím daných dat se má práce zabývá jen omezeně. Do implementační části byly vybrány některé možné případy užití, jako hledání léčivého přípravku a také další interakce s uživatelem, jako je přihlášení, osobní lékárna uživatele (uživatel si ukládá léčivé přípravky, které využívá) nebo možnost hodnotit jednotlivé léčivé přípravky. V rámci rozsahu této diplomové práce také byla provedena pouze omezená integrace z jiných datových zdrojů. Celkově implementované části, zejména webový server a klientská aplikace, spíše demonstrují možnosti dalšího rozvoje aplikací.

RDF úložiště obsahuje zejména data o léčivých přípravcích pocházející z katalogu otevřených dat, který je poskytován SÚKL. Dále pak obsahuje data o cenách léčivých přípravků pocházející z Ministerstva zdravotnictví České republiky. Dalším datovým zdrojem je služba RxNav, která se stala mimo jiné prostředníkem mezi českými a zahraničními zdroji. Na základě integrace identifikátoru této datové sady, který je integrován do RDF databáze, je pak možné prostřednictvím této služby integrovat další datové zdroje, například databázi zabývající se zejména chemickými vlastnostmi léků - Drugbank. Dále by bylo možné využít data ze služby Open Phacts, která integruje další datové zdroje z oblasti léčivých přípravků, jejich chemického složení, terapeutických a farmakologických účinků.

Dalším zajímavým směrem, kam by se mohl vývoj aplikací ubírat, je zpravování textů léčivých přípravků. Každý léčivý přípravek obsahuje SPC, nebo-li souhrn údajů o přípravku, PIL - příbalové informace a text na obalu. Zahraniční datové zdroje obsahují informace o kontraindikacích, o užívání, nicméně jsou především v anglickém jazyce. Tyto doprovodné texty však obsahují potenciálně obrovské množství dat v českém jazyce. Bylo by zapotřebí strojového zpracování těchto textů. Z dané datové sady by bylo možné extrahovat různé informace. Například určit klíčová slova každého dokumentu, na základě by byla léčiva dohledatelná například na základě názvu nemoci. Bylo by také možné extrahovat kontraindikace a jejich popis přímo z těchto dokumentů.

Mimo další rozšíření datové vrstvy by bylo také možné rozšířit škálu kli-

entských aplikací. Momentální řešení poskytuje pouze webovou klientskou aplikaci. Nicméně na základě architektury není problém vytvořit další klientské aplikace. Zejména by se mohlo jednat o mobilní aplikace pro systémy Android a iOS. Implementace těchto dalších aplikací by nevyžadovala výraznější zásahy do stávající architektury, protože je navržena takovým způsobem, že počítá právě s touto možností rozšiřitelnosti.

Jako další možnost, kde vidím potenciální rozšíření aplikací, je získávání rozšířených informací z vytvořené datové sady. Budoucí směřování aplikace například vidím v propojení datových zdrojů s uživateli. Zde je prostor pro vytváření nových informací ohledně účinků léčivých přípravků. Aplikace by mohla fungovat jako sběrna dat. Bylo by tak možné zkoumat například účinky léčivých přípravků na lidi na základě věku, pohlaví, váhy ale také na základě využívání jiných léčivých přípravků. Samozřejmě by se jednalo o chemicky a farmakologicky nepodložené informace. Nicméně by tato aplikace mohla poskytnou náměty a podklady pro další směřování v oblasti farmacie a chemie.

Právě díky tomuto předpokladu jsem v aplikaci Drugie implementoval možnost přihlašování uživatelů a hodnocení jednotlivých léků. Samozřejmě se jedná o naivní přístup, ze kterého by pravděpodobně nemohly být vyvozovány další závěry. Tento postup jsem však zvolil, abych prezentoval tuto možnost. Kdyby aplikace směřovala k tomu, aby uživatelé o sobě vyplnili více dat jako věk pohlaví a další, bylo by pak možné tyto data statisticky zpracovávat a vyvozovat z nich informace. Už i na základě současné implementace by bylo možné sledovat chování jednotlivých uživatelů. Například sledování přidávání léků do osobních lékáren. Bylo by možné pozorovat, zda dochází například k výměně některého léku za jeho alternativu. Na základě tohoto zjištění by pak bylo možné srovnat jejich chemické složení.

8 Závěr

Diplomová práce Databáze léčiv a její využití v experimentálním medicínském systému vznikla kvůli vytvoření datového zdroje o léčivých přípravcích dostupných v České republice a o jejich chemických, farmakologických a terapeutických vlastnostech.

Vytvoření tohoto datového úložiště je demonstrováno na implementaci webového serveru a klientské aplikace Drugie. Tato aplikace ověřuje dosažení požadovaných cílů této diplomové práce. Je zde provedena implementace několika případů užití. Zejména vyhledávání léčivých přípravků a zobrazování detailních informací o nich.

Práce se nejdříve zabývá teoretickou rovinou sémantického webu a využití sémantických databází. V další části popisuji potenciální datové zdroje pro tvorbu tohoto datového úložiště. Kapitola Návrh řešení a architektury popisuje využité technologie a principy při implementaci. Následuje popis samotné implementace jednotlivých částí a nakonec provádím diskuzi nad stávajícím řešením a popisuji možnosti další rozšiřitelnosti.

Práce se drží zadání pro tuto diplomovou práci. Jako ověření funkcionality této diplomové práce je vytvořena klientská aplikace Drugie, která demonstruje možné využití tohoto datového zdroje.

Literatura

[1] *Tim Berners-Lee.* https://www.w3.org/People/Berners-Lee/, 2015.

[2] D. C. T. Berners-Lee, *HTML RFC 1866.* w3c, 1995.

[3] D. H. Sack, "Semantic web technologies."

[4] J. M. Silva. Web 3.0: A vision for bridging the gap between real and virtual. [Online]. Available: http://csis.pace.edu/ctappert/dps/d861-09/team3-read3.pdf

[5] A. Medić, *Making secure Semantic Web.* Universal Journal of Computer Science and Engineering Technology, 2010.

[6] M. Sintek, *The Vision of the Semantic Web.* Stanford University, 2002.

[7] C. Meltzer. Semantic web: the devil is in the details. [Online]. Available: http://www.sajim.co.za/index.php/SAJIM/article/viewFile/352/342

[8] J. B. Mark Bartel. Xml signature syntax and processing. [Online]. Available: http://www.w3.org/TR/xmldsig-core/

[9] T. Berners-Lee. Uniform resource identifier (uri): Generic syntax. [Online]. Available: https://www.ietf.org/rfc/rfc3986.txt

[10] ——. Uniform resource locators (url). [Online]. Available: https://www.ietf.org/rfc/rfc1738.txt

[11] K. Sollins. Functional requirements for uniform resource names. [Online]. Available: https://tools.ietf.org/html/rfc1737

[12] D. H. Sack. How to identify things? - uris. [Online]. Available: https://www.youtube.com/watch?v=wG2c2YBIWB0

[13] J. H. Tim Berners-Lee and O. Lassila, *The Semantic Web.* Scientific American, 2001.

[14] D. H. Sack. Semantic web technologies. [Online]. Available: https://www.youtube.com/watch?v=0uZOdFc8sYY

[15] ——. Semantic web technologies. [Online]. Available: https://www.youtube.com/watch?v=qbkJi_rzcc8

[16] D. Brickley. Rdf schema 1.1. [Online]. Available: http://www.w3.org/TR/rdf-schema/#ch_containervocab

[17] Xml rdf. [Online]. Available: http://www.w3schools.com/xml/xml_rdf.asp

[18] T. B.-L. David Beckett. Terse rdf triple language. [Online]. Available: http://www.w3.org/TR/turtle/

[19] D. Beckett, "A line-based syntax for an rdf graph," 2014. [Online]. Available: http://www.w3.org/TR/n-triples/

[20] Json for linking data. [Online]. Available: http://json-ld.org/

[21] D. H. Sack. How to model classes and relations? -. [Online]. Available: https://www.youtube.com/watch?v=0xT7AvDRHcs

[22] ——. Rdf based knowledge representation. [Online]. Available: https://open.hpi.de/files/84f8c9f7-6261-4311-8025-a4ba042e7e16

[23] S. Q. L. for RDF. Sparql query language for rdf sparql query language for rdf. [Online]. Available: http://www.w3.org/TR/rdf-sparql-query/

[24] K. G. Clark. Sparql protocol for rdf. [Online]. Available: http://www.w3.org/TR/rdf-sparql-protocol/

[25] Sparql query results xml format. [Online]. Available: http://www.w3.org/TR/rdf-sparql-XMLres/

[26] D. H. Sack. Ontology as central concept in philosophy. [Online]. Available: https://www.youtube.com/watch?v=mXdswAsFxO0

[27] V. SVÁTEK. Ontologie a www. [Online]. Available: http://nb.vse.cz/~svatek/onto-www.pdf

[28] D. H. Sack. Web ontology language - owl. [Online]. Available: https://www.youtube.com/watch?v=LdsYkpFvYxU

[29] Súkl. [Online]. Available: http://www.sukl.cz/

[30] Status sukl. [Online]. Available: http://www.sukl.cz/sukl/statut-sukl

[31] Otevřená data súkl. [Online]. Available: https://opendata.sukl.cz/

[32] [Online]. Available: http://www.mvcr.cz/clanek/otevrena-data.aspx

[33] Podmínky užití otevřených dat. [Online]. Available: https://opendata.sukl.cz/?q=podminky-uziti-otevrenych-dat

[34] Atc/ddd index 2016. [Online]. Available: http://www.whocc.no/atc_ddd_index/

[35] Atc - anatomical therapeutic chemical - introduction. [Online]. Available: http://sydney.edu.au/medicine/fmrc/atc/index.php

[36] T. U. of Sydney. Atc - structure. [Online]. Available: http://sydney.edu.au/medicine/fmrc/atc/structure/index.php

[37] (2016, 2) Databases, resources apis. [Online]. Available: https://wwwcf2.nlm.nih.gov/nlm_eresources/eresources/search_database.cfm

[38] Rxnav. [Online]. Available: https://rxnav.nlm.nih.gov/index.html

[39] (2014) Rxnorm. [Online]. Available: https://www.nlm.nih.gov/research/umls/rxnorm/

[40] (2016) Ndf - rt. [Online]. Available: https://www.nlm.nih.gov/research/umls/sourcereleasedocs/current/NDFRT/

[41] Rxterm. [Online]. Available: https://wwwcf.nlm.nih.gov/umlslicense/rxtermApp/rxTerm.cfm

[42] Node js. [Online]. Available: https://nodejs.org/en/

[43] React js. [Online]. Available: https://facebook.github.io/react/

[44] Stardog. [Online]. Available: http://stardog.com/

[45] Mongo db. [Online]. Available: https://www.mongodb.org/

[46] Npm js. [Online]. Available: https://www.npmjs.com/

[47] Mustache. [Online]. Available: https://mustache.github.io/

[48] Express js. [Online]. Available: http://expressjs.com/

[49] Mongoose. [Online]. Available: http://mongoosejs.com/

[50] Higher-order functions and common patterns for asynchronous code. [Online]. Available: https://www.npmjs.com/package/async

[51] Redux. [Online]. Available: http://redux.js.org/

[52] Flux. [Online]. Available: https://facebook.github.io/flux/

Seznam obrázků

Seznam tabulek

Ukázky zdrojového kódu