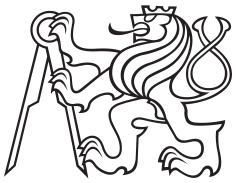
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Internetové stránky s podporou sémantického webu

Matěj Kulich

Školitel: Ing. Martin Ledvinka
Květen 2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Kulich** Jméno: **Matěj** Osobní číslo: **475740**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Internetové stránky s podporou sémantického webu

Název bakalářské práce anglicky:

Semantic Web-enabled Web Site

Pokyny pro vypracování:

1. Seznamte se s technologiemi sémantického webu, především RDF a principy Linked Data.
2. Analyzujte existující webové systémy pro správu obsahu. Zvláštní pozornost věnujte jejich rozšiřitelnosti o podporu technologií sémantického webu.
3. Navrhněte nasazení systému pro správu obsahu a jeho rozšíření o podporu sémantického webu.
4. Nasaďte Vámi vybraný systém, vytvořte v něm webovou prezentaci skupiny znalostních softwarových systémů.
5. Rozšiřte vytvořenou prezentaci o sémantická metadata.
6. Ověřte schopnost vytvořeného řešení automaticky poskytovat strojově čitelná data pomocí Linked Data crawlerů (např. LDSpider).

Seznam doporučené literatury:

[1] D. Allemang, J. Hendler, Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL, Morgan Kaufmann, 2011
[2] W. Morris: 8 Best CMS Platforms to Start a Website in 2020, 2020, https://www.hostinger.com/tutorials/best-cms
[3] R. Isele, J. Umbrich, Ch. Bizer and A. Harth: LDSpider: An open-source crawling framework for the Web of Linked Data, Proceedings of 9th International Semantic Web Conference (ISWC 2010) Posters and Demos, 2010

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Martin Ledvinka, skupina znalostních softwarových systémů FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **12.02.2021** Termín odevzdání bakalářské práce: **21.05.2021**

Platnost zadání bakalářské práce: **30.09.2022**

Ing. Martin Ledvinka podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____
Datum převzetí zadání

_____
Podpis studenta

Poděkování

Chtěl bych především poděkovat panu Ing. Ledvinkovi za cenné rady, konzultace a celkovou podporu. Dále bych chtěl také poděkovat své rodině za podporu a pevné nervy.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 20. května 2021

Abstrakt

Práce se zabývá sémantickým webem a systémy pro správu obsahu (CMS). Na úvod řeší kontext sémantického webu s popisem jednotlivých technologií. Dále je popsána rešerše CMS, jsou definovány kritéria pro výběr tohoto systému a je zmíněno několik nejpoužívanějších redakčních systémů. Následně řeší analýzu a návrh a nakonec se věnuje popisu implementace a následnému testování.

Klíčová slova: Sémantický web, CMS, RDF, XML, URI, metadata, Liferay

Školitel: Ing. Martin Ledvinka

Abstract

This thesis researches the topic of semantic web and content management systems (CMS). In the beginning thesis describes context of semantic web with description of individual technologies. Then research of CMS is presented, criteria of choosing right system is defined and a few these systems are discussed. Then thesis includes analysis and design of the application. Implementation with testing is described in the end.

Keywords: Semantic web, CMS, RDF, XML, URI, metadata, Liferay

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

1.1 Předmluva

Počet nových webových stránek stále narůstá a díky tomu se internet zaplňuje daty. Na jednu stranu je dobré, že se internet rozšiřuje a informace jsou tak více dostupné, avšak s tímto objemem dat nám přestává stačit fulltextové vyhledávání na nalezení relevantních dat, tudíž je potřeba datům dodat nějaký význam a tímto se právě zabývá sémantický web, který více přiblížím v této práci. Vytvořit kvalitní webové stránky od píky je poměrně časově náročná záležitost a proto existují různé systémy, které nám vývoj výrazně zjednodušují a zrychlují, mezi které patří i systémy pro správu obsahu, které jsou jedním z pilířů této práce a budou rozebrány kritéria, podle kterých zvolit správný systém.

1.2 Cíl projektu

Cílem této práce je zanalyzovat sémantický web a jeho technologie a následně udělat rešerši webových systémů pro správu obsahu (CMS). Přičemž v praktické části práce půjde o vytvoření webové aplikace, která je schopna vracet jak textová, tak strojově čitelná data a přitom bude snaha použít některý z existujících CMS na implementaci. Výstup této práce bude sloužit pro potřeby výzkumné skupiny KBSS.

Kapitola 2
Kontext sémantického webu

“Sémantický web není odděleným webem, je to rozšíření toho stávajícího. Informace v něm budou mít jasně definovaný význam, což umožní počítačům a lidem lépe spolupracovat.”[1] tuto definici řekl jeden z protagonistů W3C[1] a autor WWW Tim Berners-Lee, který tímto navazuje na jeho myšlenku, kde upozorňuje na fakt, že současný web je pouze směsice webových stránek, které na sebe nenavazují a nemají definovaný přesný význam. Slovo “sémantický” označuje strojově zpracovatelné nebo to, co je stroj schopen s daty dělat. Zatímco „web“ vyjadřuje myšlenku prostoru propojených objektů s mapováním z URI na zdroje [2].

2.1 Vize sémantického webu

Hlavní ambicí sémantického webu je tedy umožnit počítačům lépe manipulovat s informacemi a toho dosáhneme tím, že k datům přidáme metadata a tudíž je poté počítač schopný rozeznávat a do jisté míry i pochopit smysl těchto dat, avšak pouhá metadata nestačí. Jsou zapotřebí standardy pro syntaxi jejich zápisu a reprezentaci sémantických znalostí tak, aby mohly být efektivně a mnohostranně využity.

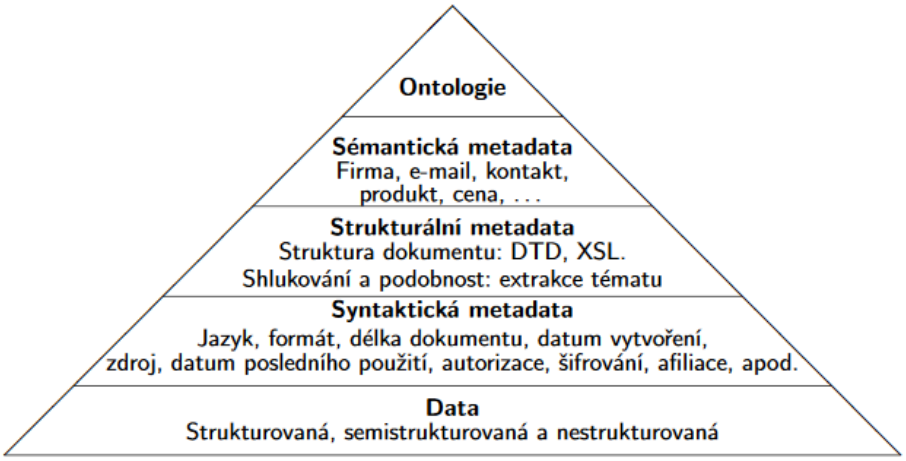
2.2 Metadata

Metadata jsou zkráceně data o datech a poskytují dodatečné informace nějakému obsahu. Když si vezmeme klasický obrázek, tak může obsahovat metadata například popisující velikost, datum vytvoření či barevnou hloubku. Webové stránky obvykle obsahují tyto metadata uložené ve formě meta tagů, díky kterým poté vyhledávač může indexovat tuto stránku [4]. Tato definice často bývá s metadaty spojována, avšak to není zas tak jednoduché. Metadata často nezahrnují jen základní informace o nějakém webovém obsahu, ale pojímají např. informace o tom, jak jsou spravována či uložena. Informace mohou být velmi rozmanité, může jít o popis nebo shrnutí obsahu, jindy se

[1] World Wide Web Consortium je mezinárodní komunita, kde členové pracují na vytvoření standardů pro web [3].

jedná o informace nezávislé na obsahu, jako je umístění nebo datum vytvoření. Metadata se dají rozdělit podle úrovně abstrakce nad daty následovně [5]:

- **Syntaktická metadata**
  Tyto metadata jsou především podrobnosti o zdroji dat a díky tomuto poskytují lepší pohled na data. Tato skupina se hodí na kategorizaci a katalogizaci.
- **Strukturální metadata**
  Podle názvu kategorie se dá usoudit, že se jedná primárně o strukturu dokumentu, která později může sloužit k ukládání, zpracování nebo prezentaci. Typickým příkladem je XML formát.
- **Sémantická metadata**
  Popisují kontextově podstatné informace vzhledem k doméně. Díky použití těchto metadat můžeme zajistit smysluplnou interpretaci dat a zvýšit schopnost interoperability[2].
- **Ontologie**
  Představují nejvyšší formu metadat a zároveň jsou základním kamenem sémantického webu. Podrobněji navážeme v další sekci 2.3.



Obrázek 2.1: Typy metadat [5]

[2]Spolupráce mezi více systémy, například poskytování služeb

2.3 Ontologie

První zmínka o *ontologii* se vyskytla ve filozofii, ve které označuje nauku o jsoucnu a bytí. V novém technickém pojetí však ontologie označuje všechny možné metody získávání znalostí o obsahu dat. Jednou ze základních definic v počítačovém světě je "Ontologie je explicitní specifikace konceptualizace" (T. Gruber, 1993), která popisuje to, že konceptualizace[3] musí být specifikována explicitně, to jest nikoliv jen v hlavě autora. Mezi hlavní účely ontologií řadíme podporu porozumění mezi lidmi, podporu komunikace mezi systémy a usnadnění návrhu aplikací, které se primárně soustředí na znalosti [6]. Ontologie jsou převážně formulovány pomocí nějakého reprezentačního jazyka, který je velmi blízký logickým formalismům. Tyto jazyky jsou obvykle založeny na predikátové nebo deskriptivní logice. Jedná se například o Common Logic, CycL nebo novější jako RDF či OWL, o kterých se zmíním níže 2.6.2 [7].

2.4 URI a IRI

Předtím, než se začneme zabývat další kapitolou s názvem "Linked data", je nutné si nejdříve objasnit, co je vlastně Uniform Resource Identifier (URI). URI identifikuje různé zdroje nejen na webu ale i v počítači. Tyto zdroje se můžou lišit, může to být například webová stránka nebo odesílatel/příjemce emailu. Dnešní aplikace používají jednoznačné označení k identifikaci zdroje nebo vyžádání dat z něj. Protokoly jako HTTP[4] nebo FTP[5] mohou na tomto základě fungovat, protože forma identifikace je předdefinována syntaxí URI [8]. Co se týče Internationalized Resource Identifier (IRI), tak je to nový prvek protokolu, který je doplňkem URI a může být namapován na URI, což znamená, že v nějakých případech lze použít IRI místo URI [9].

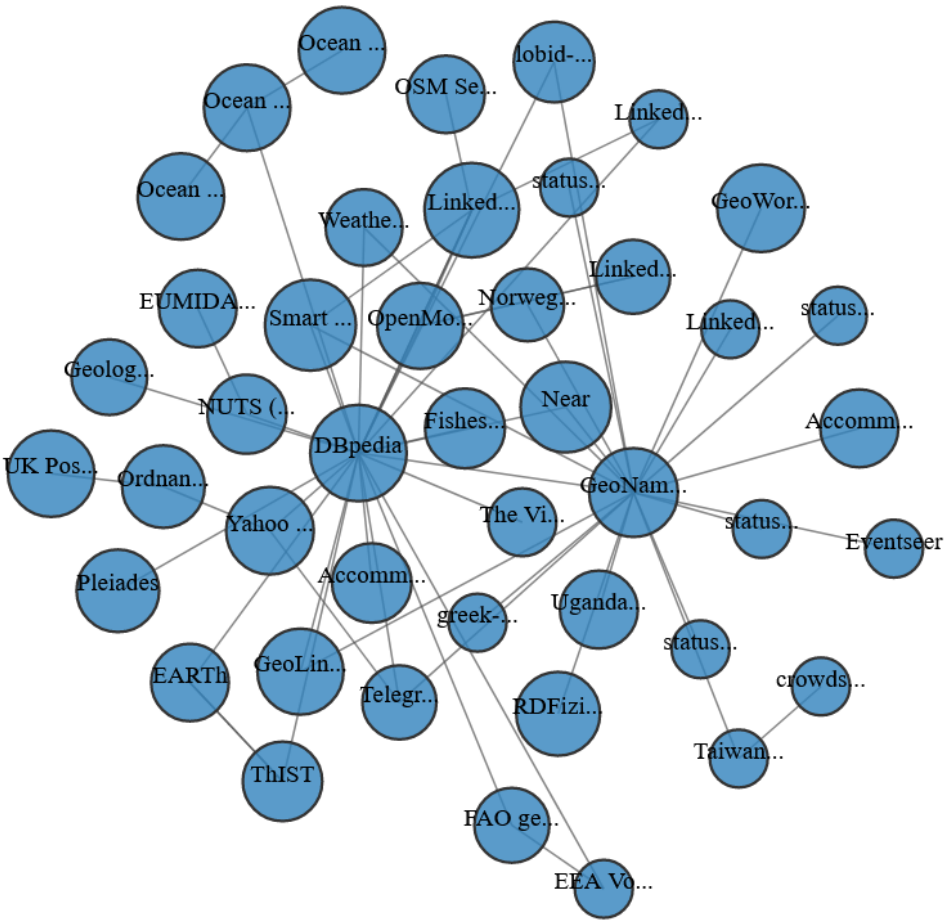
2.5 Linked data

Termín "Linked data" odkazuje na způsob vkládání strukturovaných dat na web tak, aby byly tyto data co nejužitečnější a mohly být znovu použity pro další účely. Důležité přitom je, aby byly dodrženy základní principy, které definoval Tim Berners-Lee a jsou následující [10]:

1. Používat URI jako název pro věci
2. Používat HTTP URI, aby si lidé mohli tyto názvy vyhledat
3. Když někdo vyhledá URI, poskytněte užitečné informace pomocí standardů (RDF*, SPARQL)
4. Zahrnout odkazy na jiné URI, aby bylo lehčí najít relevantní informace k dané věci

[3] systém pojmů, kterými lze popsat určitou část světa
[4] HyperText Transfer Protocol
[5] File Transfer Protocol

Hlavní myšlenkou těchto principů je především používat standardy reprezentace a přístupu dat na webu a dále také propojit hypertextovými odkazy[6] tyto data z různých zdrojů. Tyto odkazy pak spojují všechna propojená data do jedné globální databáze, díky které je pak snazší vyhledat relevantní informace [10]. Obrázek 2.2 ukazuje jednu skupinu sjednocených databází, které obsahují geografické informace. Mezi nejznámější patří DBpedia[7], která čerpá data z wikipedie a přetváří je na strukturalizované a prezentuje je jako součást sémantického webu.



Obrázek 2.2: Geografické linked data [11]

[6]Text, který odkazuje na jiný zdroj
[7]https://www.dbpedia.org/

2.6 Reprezentace dat a znalostí

V této sekci budou popsány základní technologie, kterými lze reprezentovat data.

2.6.1 XML

XML (eXtensible Markup Language) je jednoduchý dynamický a velmi flexibilní textový formát dat, který využívá tzv. tagy. Tyto tagy můžeme použít jako meta-data, tudíž je tento formát velmi vhodný pro účely sémantického webu. Je primárně určen k posílání dat. Hlavní cíle XML jsou především jednoduchost, obecnost a použití napříč Internetem [12]. Jeho formát je zobrazen v následující části 2.1.

Listing 2.1: Příklad XML

```
<?xml version = "1.0"?>
<person>
   <firstname> John </firstname>
   <lastname> Doe </lastname>
   <gender> male </gender>
</person>
```

2.6.2 RDF

RDF (Resource Description Framework) je standardní model pro výměnu dat na webu. RDF má funkce, které usnadňují slučování dat, i když se základní schémata liší, a konkrétně podporuje vývoj schémat v čase. RDF rozšiřuje strukturu propojení webu tak, aby používal URI k pojmenování vztahu mezi věcmi. Pomocí tohoto jednoduchého modelu umožňuje strukturovaná data slučovat, publikovat a sdílet napříč různými aplikacemi. Tato propojení si můžeme představit jako graf, který je reprezentován jako dva uzly spojené hranou [13]. V RDF je každý fakt nazvaný tvrzení, které se skládá ze tří částí, proto se tomu také občas říká trojice (triple). Tato trojice obsahuje subjekt, predikát a objekt, kde subjekt představuje zdroj, kterého se tvrzení týká. Může to být například nějaká osoba, věc či místo. Zatímco predikát reprezentuje nějakou vlastnost či vztah mezi zdroji. Objekt může být zase opět zdroj nebo nějaký literál [14].

Když si vezmeme větu “Autorem World Wide Webu je Tim Berners-Lee.”.

- Subjekt v tomto tvrzení je World Wide Web
- Predikát je autor
- Objekt je Tim Berners-Lee

V následující části 2.2 je vyobrazen RDF/XML syntaxe, která se používá na popsání grafu RDF ve formátu XML. První řádek je tzv. XML deklarace, která určuje, že následující text je ve formátu XML a také určuje verzi XML. Element rdf:RDF informuje o tom, že text v tomto tagu reprezentuje RDF. Dále je element xmlns, který určuje XML jmenný prostor. Všechny tagy v s předponou rdf stanovují, že jsou součástí jmenného prostoru na definované URI. Poté se definuje další jmenný prostor tentokrát s prefixem ex. O RDF tvrzeních můžeme mluvit jako o popisu, tudíž na řádce 4 označujeme element rdf:Description, který vytváří tvrzení o zdroji "http://www.example.org/leonard_nimoy". Je do něj vložen další tag, tentokrát určující vlastnost o zdroji. V našem případě to znamená, že herec Leonard Nimoy ztvárňuje jednu z hlavních rolí(vlastnost). Na řádku číslo 5 máme další tag, který určuje hodnotu této vlastnosti.

Listing 2.2: RDF/XML formát

```
1 <?xml version="1.0"?>
2 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-
    ns#"
3     xmlns:ex="http://www.example.org/">
4     <rdf:Description rdf:about="http://www.example.org/
    leonard_nimoy">
5         <ex:starred_in> Star Trek </ex:starred_in>
6     </rdf:Description>
7 </rdf:RDF>
```

2.6.3 RDF(S)

RDF Schema je sémantické rozšíření RDF. Poskytuje mechanismy pro popis skupin souvisejících zdrojů a vztahů mezi těmito zdroji. Tyto zdroje se používají k určení charakteristik dalších zdrojů, jako jsou domény a rozsahy vlastností. Tento systém skupin je podobný objektově orientovaným jazykům jako je například Java.

2.6.4 OWL

Web Ontology Language (OWL) je jazyk sémantického webu navržený tak, aby představoval bohaté a komplexní znalosti o věcech, skupinách věcí a vztazích mezi věcmi. Je to jazyk založený na výpočetní logice, takže znalosti vyjádřené v OWL mohou být využívány počítačovými programy např. k ověření konzistence těchto znalostí. Dokumenty OWL, známé jako ontologie, lze publikovat v síti WWW a mohou odkazovat nebo být odkazovány na jiné ontologie OWL. Také patří do skupiny technologií spravovanou W3C[8] stejně jako RDF.

[8] World Wide Web Consortium

Aktuální verze OWL je označovaná jako „OWL 2“ a je rozšířením verze z roku 2004. Mezi výstupy, které tvoří specifikaci OWL 2, patří přehled dokumentu, který slouží jako úvod do OWL 2 a popisuje vztah mezi OWL 1 a OWL 2 [17].

Kapitola 3

CMS systémy

CMS (Content management system), občas překládaný jako redakční systém, je software, který poskytuje snadné a rychlé vytváření webových stránek s obsahem jako například e-shopy, blogy nebo galerie. Jedná se o jedno z nejlepších a nejpoužívanějších řešení na tvorbu stránek. Dříve by člověk bez znalosti programování a dalších technických dovedností nedokázal vytvořit ani statickou stránku s pouhým textem a obrázky, to ale příchod CMS značně změnil. Jednou z hlavních vlastností a výhodou těchto systémů je online vestavěný WYSIWYG (what you see is what you get) editor, který se vyskytuje u většiny CMS a ukazuje v reálném čase přesně to, jak bude webová stránka vypadat, což dodává uživateli příjemný pocit interakce s webovou stránkou.

Avšak každá mince má dvě strany, tudíž ani CMS se nevyhne nějakým nevýhodám jako například pomalému načítání, a to díky tomu, že CMS je database driven, tudíž se musí čekat na zpracování stránek a to odhání návštěvníka, který musí čekat. Mezi další nevýhody patří určitě omezená optimalizace pro vyhledávače (SEO), což znamená že vyhledávací enginy jako google nebo seznam hůře hledají a neprioritizují tyto stránky ve výsledcích hledání. Je to způsobené tím, že stránky CMS jsou dynamicky generovány a to znamená, že URL těchto stránek je většinou automaticky generována a obsahuje dlouhé řetězce, které jsou kombinací slov, čísel a speciálních znaků. Mezi další nevýhody patří zajisté také nutnost seznámit klienta s prostředím CMS. Tím je myšleno vysvětlit klientovi, jak spravovat na webové stránce věci jako přidání obsahu nebo editaci práv ostatních autorů přes administrátorský panel, což může být stěžejní záležitost pro klienta [21].

3.1 Kritéria pro srovnání CMS

Než se přesuneme přímo na jednotlivé CMS, tak si nejprve shrneme podle jakých vlastností a kritérii si vybrat ten správný systém.

- **Účel a velikost**
  Nejprve je dobré si rozmyslet, co od daného CMS požadujete a jak velký projekt plánujete vytvořit. Některé systémy se hodí spíše pro malé a střední projekty (WordPress) a jiné zase pro rozlehlé (Drupal), tudíž je dobré se neukvapovat s rozhodnutím a vytyčit si funkční požadavky a následně vybrat odpovídající systém.

- **Cena**
  Tyto systémy často bývají open-source, což znamená, že si neúčtují poplatky za provozování a funkce, tedy alespoň v základním balíčku, avšak čím větší cena CMS, tím větší robustnost, počet funkcí a možností systému. Cena v tomto případě není ukazatel kvality, ale spíše robustnosti, poukazuje na to i fakt, že nejrozšířenější a nejpoužívanější CMS je právě open-source WordPress, tudíž zde neplatí čím dražší, tím lepší.

- **Podpora sémantického webu**
  Vzhledem k povaze této práce je důležité také zhodnotit, jakým způsobem lze sémantická data přidat do webové stránky. Některé CMS mají podporu zabudovanou v základu systému jako třeba Drupal, avšak to bývá často velmi omezené a při nedostačující funcionalitě je nutné se přesunout na nějaký plugin.

- **Šablony a vzhled**
  Každý systém má několik předvytvořených šablon na úpravu a použití, avšak pro větší projekty jsou tyto šablony nedostatečné, tudíž je potřeba si nejprve rozmyslet, zda budou dostačovat základní šablony nebo bude potřeba vytvořit novou šablonu na míru a nebo koupit nějakou již udělanou šablonu třetí strany, čímž se zpátky vracíme k důležitosti výběru robustnosti systému.

- **Pluginy a rozšíření**
  Všechny webové stránky nejsou stejné, proto je nemožné přijít s jedním univerzálním řešením pro všechny uživatele najednou. Tento problém řeší pluginy a různá rozšíření. Je to vlastně taková nadstavba systému, která doplňuje a rozšiřuje stávající funkcionalitu o chybějící prvky [18].

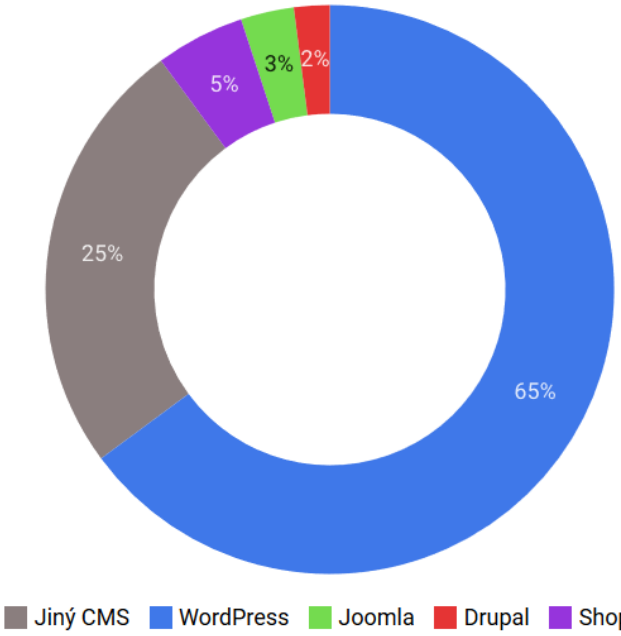
- **Support a fóra**
  Ačkoliv se CMS snaží být co nejjednodušší na použití, stále se najdou otázky, na které je potřeba najít řešení, například když si administrátor systému neví rady. Na to slouží téměř 24/7 podpora u nejpoužívanějších CMS, oproti tomu když se podíváme na menší, ne tak známé systémy, tak ty mívají většinou odpověď jenom na nejčastější dotazy a tudíž je nutné vyčkat na odpověď delší dobu [18].

3.2 Jednotlivé CMS systémy

Tabulka 3.2 popisuje statistiku použití CMS systémů každý rok od 2011 [19]. Je zde patrné, že CMS stále více narůstá na síle, když pouze 38.3% webových stránek nepoužívá žádný redakční systém. Největší procento k dnešnímu dni tvoří WordPress, což je nejznámější CMS. Celkem 39.5% veškerých webových stránek na internetu je ke dni 1.1.2021 tvořeno za pomocí WordPressu a to z něj dělá systém číslo jedna na vytváření webů s obsahem. Můžeme vidět, že již v roce 2011 zaujímal 13% podílu na trhu, které se postupně narůstaly až do dnešní podoby. Naopak když se podíváme na Shopify, který do roku 2014 nebyl zas tak používaný, tak v posledních letech se začal více využívat a v minulém roce vzrostl natolik, že předehnal Joomlu a stává se tak druhý nejpoužívanější. Když se podíváme na Joomlu a Drupal, tak jejich použití v posledních letech upadá, avšak to může být způsobené neskutečnou dominancí WordPressu, který zastiňuje ostatní.

Procentuální roční použití CMS, vždy k prvnímu dni roku [19]							
	2011	2014	2015	2016	2019	2020	2021
Žádný	76.4%	64.8%	61.7%	56.6%	45.3%	43.1%	38.3%
WordPress	13.1%	21.0%	23.3%	25.6%	32.7%	35.4%	39.5%
Shopify		0.1%	0.3%	0.4%	1.4%	1.9%	3.2%
Joomla	2.6%	3.3%	3.3%	3.3%	3.0%	2.6%	2.2%
Drupal	1.4%	1.9%	2.0%	2.1%	1.9%	1.7%	1.5%

Graf na obrázku 3.1 ukazuje rozdělení jednotlivých CMS použitých na webové stránky tvořených pouze pomocí redakčního systému. Na grafu je vidět znatelná převaha WordPressu s 65% zastoupením, dále následuje Shopify se svými 5%, Joomla se 3% a nakonec Drupal se 2%. Zbytek tvoří, ne tak používané, redakční systémy, které mají zastoupení méně než 1%. Celkem tyto systémy tvoří 25%, což ukazuje na to, že je těchto systémů mnoho a stále přibývají.



Obrázek 3.1: Distribuce webových stránek vytvořených pomocí CMS [20]

3.2.1 WordPress

WordPress je nejpoužívanější open-source redakční systém napříč všemi CMS, vytvořen pomocí PHP a MySQL. Po jeho spuštění v roce 2003 se rychle rozrostl a dodnes se drží prvenství. WordPress má velkou komunitu, která vytvořila velké množství nejrůznějších widgetů, šablon a pluginů, které pomáhají k rychlejšímu rozvoji stránky. Je vytvořený tak, aby byl vysoce přizpůsobitelný a mohl se dát lehce změnit. Vše od databáze až po modifikace šablon je pokryto API[1], díky této standardizaci mohou vývojáři vytvářet funkcionalitu nad rámec WordPressu [22]. Je tak populární díky možnosti rychlého vytvoření jednoduchého webu a jednoduchosti ovládání.

Když se podíváme na podporu sémantického webu, tak WordPress sám o sobě nepodporuje sémantický web, avšak je možné přidat pluginy, které disponují technologiemi sémantického webu. Jelikož je nejpopulárnější, tak má spoustu již vytvořených pluginů, z kterých lze vybírat. Například plugin jménem "wp-linked-data" podporuje linked data a můžete v něm zadat libovolné trojice RDF v různých formátech jako RDF/XML, Turtle nebo N3 [23]. Pluginy "Rich Meta in RDFa", "Dublin Core Metadata Generator" a "Dublin Core Metadata" umožňují uživatelům přidávat Dublin Core data[2] ve formě RDFa přímo do HTML [24]. Všechny tyto pluginy jsou open-source.

[1] Application Programming Interface
[2] je standard pro reprezentaci metadat používaný na základní informace o dokumentu

3.2.2 Shopify

Shopify je špička na poli e-commerce a oproti třeba WordPressu je dodáván jako SaaS[3]. Je vytvořen prostřednictvím open-source webové aplikace Ruby on rails, avšak Shopify jako takový není open-source, musí se platit měsíční poplatek, který záleží na počtu dodávaných funkcí jako je například správa hostingu, bezpečnosti či SSL certifikátů. Tento poplatek se pohybuje od 29$ do 299$ za měsíc [25]. Pro zajímavost zakladatel Shopify spustil jejich e-shop pouze po 2 měsících vývoje [26]. Shopify přímo nenabízí žádný veřejný plugin na podporu sémantického webu.

3.2.3 Joomla!

Mezi další CMS systémy řadíme i open-source Joomlu. Byla vytvořen v roce 2005, takže jako WordPress už je tady s námi mnoho let. Je napsaná v PHP a pro ukládání dat používá MySQL databázi. Jedná se o jeden z nejpopulárnějších systémů pro správu obsahu díky jeho funkcím, jako je ukládání stránek do mezipaměti , podpora více jazyků, doplňky a rozšíření [27]. Joomla přináší mnoho různých šablon a rozšíření, avšak je nutné nastavit hosting a doména. Ačkoliv je Joomla! téměř ideální systém pro vývojáře a zkušené tvůrce webu, tak není tolik vhodný pro začátečníky vzhledem ke komplexnosti systému [18].

Podpora sémantického webu je v tomto případě slabší než v případě WordPressu. Jako první připadá v úvahu placené rozšíření " Metadata od developera eorisis", které poskytuje rozhraní k přidávání Dublin Core dat ve formátu RDFa stejně tak jako WordPress a mnoho dalších funkcí. Toto rozšíření se platí každé tří měsíce a částka činí 37 EUR. Jako další rozšíření počítáme i "Dublin Core Extended Metadata", které také poskytuje integraci Dublin Core metadat. Toto rozšíření je velmi snadné na instalaci a je zdarma. Dále lze najít na githubu repozirář "Joomla-rdf", který implementuje opět Dublin Core.

3.2.4 Drupal

Tento redakční systém, vytvořený na platformě PHP, je také open-source a v posledních letech začal být velmi populární. Nejdříve byl vytvořen speciálně pro sociální sítě, avšak po integraci frameworku Symfony[4] si ho mnoho uživatelů oblíbilo. Drupal je díky velké schopnosti modifikace velmi vhodný pro vývojáře a tím je také skvělý pro velké a rozsáhlé projekty, které vyžadují řešení na míru s mnoha integracemi. Na práci s tímto systémem je vhodné disponovat alespoň základní technickou znalostí [28].

Od verze 8 byla sémantická technologie zahrnuta do jádra systému. Dostupnost schématu v jádru zajistila, že všechny informace o webu lze prezentovat strukturovaným způsobem, jak to preferují vyhledávače. Další modul v Drupalu, "RDF", lze použít k přidání strukturovaných metadat na všechny druhy

[3] Software as a Service
[4] PHP framework

webových stránek [29]. Existuje spoustu dalších modulů na podporu RDF jako například “RDF Extensions”, “Simple RDF” nebo “RDF entity”. Všechny tyto moduly jsou zdarma a stačí je stáhnout a vložit přímo do Drupalu. Mezi moduly podporující Dublin Core poskytuje webová stránka Drupalu také “Dublin Core to CCK”, avšak je to staršího data. “Ontology” a “Drupal2RDF” jsou další moduly, které integrují OWL a publikují strukturu webu jako OWL ontologii.

3.2.5 Liferay

Poslední jmenovaný CMS sice není tak známý a používaný jako předešlé systémy, avšak má velkou výhodu díky portletům, o kterých se zmíním později. Liferay je vytvořen nad platformou Java a je také open-source, který má na výběr ze dvou verzí, a to komunitní Standard Edition a komerční Enterprise Edition. Komerční verze je placená a klade velký důraz na stabilitu a podporu, ale není tak oblíbená jako komunitní, která se používá především na podnikové portály [31]. Předností tohoto CMS je schopnost přizpůsobit či přepsat vše, co vás jenom napadne. Zatímco většina redakčních systémů je limitována API integracemi, tak s Liferayem dostanete i zdrojový kód, který můžete snadno přepsat k obrazu svému [30].
Dále mezi hlavní výhody patří podpora téměř 32 jazyků v základním balíčku a kompatibilita s hojně využívanými databázemi jako MySQL, PostgreSQL či Oracle. Liferay je v tomto ohledu velmi univerzální a flexibilní a nestane se, že by bránily technické podmínky v jeho nasazení [31].

Kapitola 4

Analýza

Nyní přejdeme k aplikaci jako takové a nejprve je důležité si položit otázku, co by měl systém poskytovat za funkce uživateli. Tyto požadavky jsem konzultoval s vedoucím práce, ze kterých jsem později vytvořil use case diagram, který tyto požadavky graficky zobrazuje.

4.1 Stav AS-IS

Aktuální webové stránky skupiny mají problémy se stabilitou serveru. Server často padá z důvodu starší verze Liferaye a tudíž je nutné častý restart. Dále je zde vidět zastaralejší a jednodušší design stránek.

4.2 Stav TO-BE

Výsledná aplikace by měla být stabilní, poskytovat uživatelům webovou prezentaci a zároveň podporovat sémantická data. Vzhled této webové prezentace by měl odpovídat modernějšímu designu.

4.3 Funkční požadavky

Jelikož tato webová aplikace bude sloužit výzkumné skupině, tak zde bude uživatel brán jako host, zatímco členové skupiny budou mít možnost přihlášení a dalších funkcí. Člen skupiny zde bude mít stejné funkce jako host a k nim ještě některé navíc.

1. **Host**
   - Zobrazit blogové příspěvky
   - Zobrazit detail příspěvku
   - Zobrazit členy skupiny
   - Zobrazit předešlé a aktuální projekty
   - Požádat o sémantická data
2. **Člen skupiny**

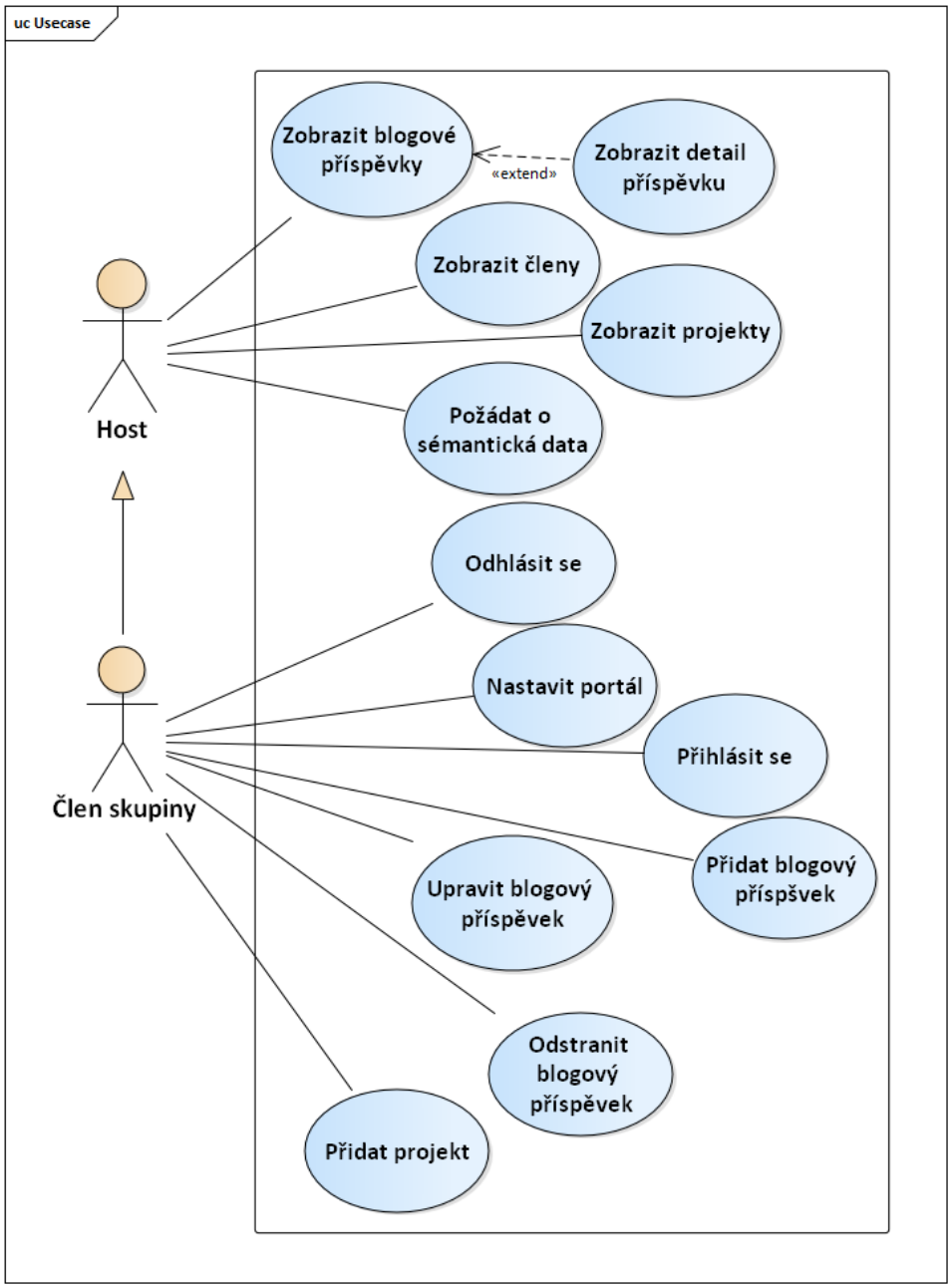
- Přihlásit se
- Odhlásit se
- Přidat projekt
- Přidat blogový příspěvek
- Odstranit blogový příspěvek
- Upravit blogový příspěvek
- Nastavit portál

4.4 Nefunkční požadavky

- **Server** - Aplikace bude nejspíše nasazena jako kontejner do Dockeru, což je nástroj, který velmi zjednodušuje nasazování a spouštění aplikací, tudíž je nutné prozkoumat možnosti integrace Liferay do Dockeru.
- **Cachování** - Data se sice budou načítat z databáze, ale není potřeba při každém zobrazení stránky znovu poslat dotaz do databáze, protože se data nebudou měnit tak často, proto bude výhodné data ukládat do cache.
- **Databáze** - Skupina má k dispozici PostgreSQL databázi, kam se budou ukládat data o uživatelích a blogových příspěvcích a poté ještě grafovou databázi GraphDB, kam se budou ukládat sémantická data ve formě RDF dat.
- **Podpora prohlížečů** - V dnešní době je víceméně nutné, aby se stránka dala zobrazit v nejpoužívanějších prohlížečích jako Google Chrome, Mozilla Firefox, Safari a Internet Edge.

4.5 Use case diagram

Na následujícím obrázek je vidět usecase diagram, který ukazuje případy užití aplikace. Diagram obsahuje 2 aktéry hosta a člena skupiny, přičemž si lze povšimnout, že člen skupiny má stejné funkce jako host a dále ještě vlastní, které se týkají úpravy obsahu či přizpůsobit webové stránky.



Obrázek 4.1: Diagram případů užití

4.6 Výběr CMS

Při výběru redakčního systému jsem se rozmýšlel mezi WordPressem a Liferayem. Myšlenka byla taková, že WordPress je nejpoužívanější CMS, tudíž bude mít velkou komunitu a podporu a navíc je pravděpodobné, že bude existovat nějaké podobné řešení mého problému, takže bych se mohl inspirovat. Na druhou stranu je WordPress napsaný v PHP, což není můj nejoblíbenější

jazyk. Když se podíváme na Liferay, tak nám přináší následující výhody. Jelikož aktuální webové stránky byly vytvořeny v tomto systému, tak s ním má skupina již nějaké zkušenosti a tudíž je zde možnost pomoci. Dále je nutné zmínit, že Liferay je napsaný v jazyce Java, což hraje velkou roli při rozhodování a navíc je open-source, tudíž lze většina věcí upravit. Nakonec jsem se rozhodl pro Liferay, především kvůli programovacímu jazyku a doporučení.

Liferay nabízí balíček DXP, který se od předchozích verzí liší znatelně větší modularitou, díky použití OSGi standardu [1]. Toto přináší velkou výhodu při deployování různých modulů, které lze nasadit rychle a bezpečně za běhu aplikace, tzn. že se nemusí restartovat server, který většinou nabíhá pár minut [32].

1. **Liferay Tomcat Bundle**
   Tento balíček je dodáván rovnou s Tomcat serverem, na kterém je již nainstalovaný Liferay DXP. Toto řešení je velmi vhodné, pokud při vývoji není k dispozici žadný server. Toto bude ideální řešení v mém případě.

2. **Liferay WAR** War je vlastně archiv obsahující soubory jako například jary, jsp, html a další soubory potřebné k vytvoření webové aplikace. Archiv se hodí v případě, že máme k dispozici nějaký aplikační server, na který můžeme rovnou naší aplikaci nahrát.

[1] https://help.liferay.com/hc/en-us/articles/360018162431-OSGi-and-Modularity

Kapitola 5

Návrh

Tato kapitola se bude zabývat návrhem z hlediska architektury, přičemž nejdříve bude popsána z hlediska podpory sémantického webu a poté naznačím samotnou architekturu. Ke konci tato kapitola řeší otázku uživatelského rozhraní.

5.1 Architektura řešení

Nejdříve zde popíši, jakým způsobem lze zakomponovat sémantická data do řešení a následně osvětlím myšlenku architektury a koncept portletu.

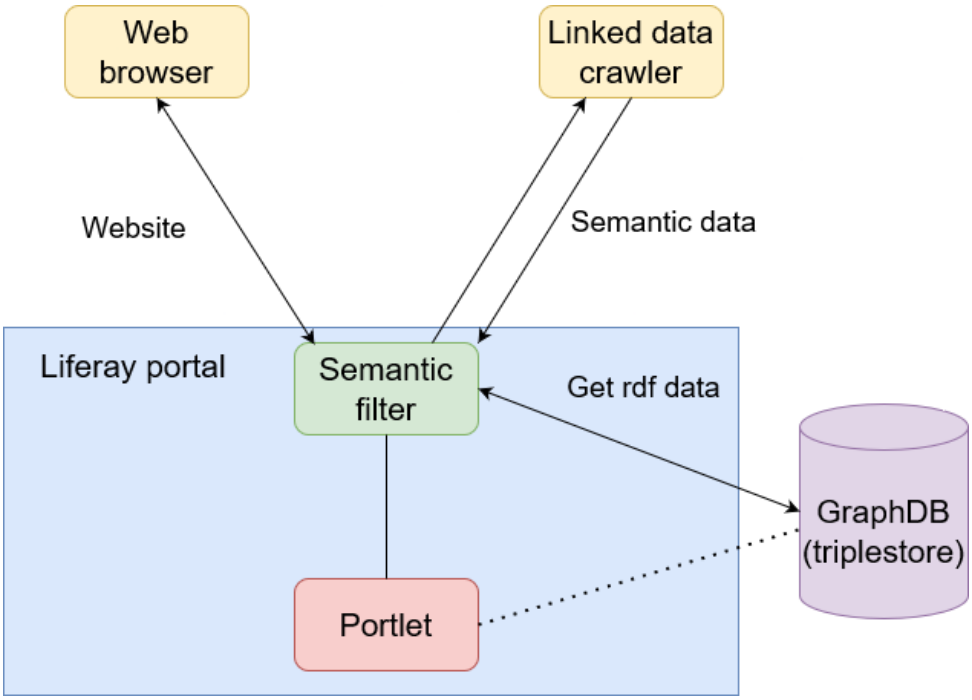
5.1.1 Podpora sémantického webu

Pro přidání sémantických dat se u webových stránek nabízí dvě schůdná řešení. První z nich funguje tím způsobem, že se stránky obohatí o atributy, ve kterých jsou uloženy metadata. To se většinou dělá přes nějaký RDFa editor, který ukládá RDF data do atributů XHTML nebo HTML. Tento způsob je výhodný v tom, že jsou data uložena přímo ve stránce a není potřeba se dotazovat na jiné místo, ale je nutné najít vhodný editor. Druhý způsob potřebná metadata ukládá do triplestoru, avšak za cenu častého dotazování a nutnosti implementace další části, která rozhoduje, kdy poskytnout sémantická data a kdy jenom webovou stránku. Triplestore je RDF databáze, do které se ukládají trojice. V tomto případě mi přišlo lepší zvolit RDFa editor, kvůli přímočařejší implementaci a vyhnout se tak SPARQL dotazům. Ale vzhledem k tomu, že po delší době nebyl nalezen žádný vyhovující editor, který by přidával sémantická data přímo do stránky a zároveň se dal integrovat do Liferay, tak jsem musel přistoupit ke druhé variantě.

5.1.2 Architektura a portlety

Myšlenka této architektury je naznačena na obrázku 6.1. Můžeme vidět, že nejdříve přijde požadavek do filtru, který se na základě Accept hlavičky rozhodne, jestli vrátí webové stránky nebo sémantická data. Pokud má tedy vrátit sémantická data, tak filtr nejdříve získá výsledná data z triplestoru a následně je vrátí klientovi, avšak v druhém případě se požadavek pošle do

portletu, což je webová aplikace v Liferay DXP. Koncept portletu lze chápat jako nějakou komponentu, která poskytuje přidanou funkcionalitu. Přičemž v tomto případě bude nutné vytvořit nějaké vlastní portlety, protože žádný z existujících pluginů neposkytoval požadované funkce.



Obrázek 5.1: Návrh architektury

5.2 Uživatelské rozhraní

Prvotní inspirace na vzhled webových stránek přišla ze stávajícího webu, která vyústila k vytvoření jednoduchého návrhu vzhledu pomocí Liferay WYSIWYG editoru. Tento návrh byl později přepracován na modernější vzhled.

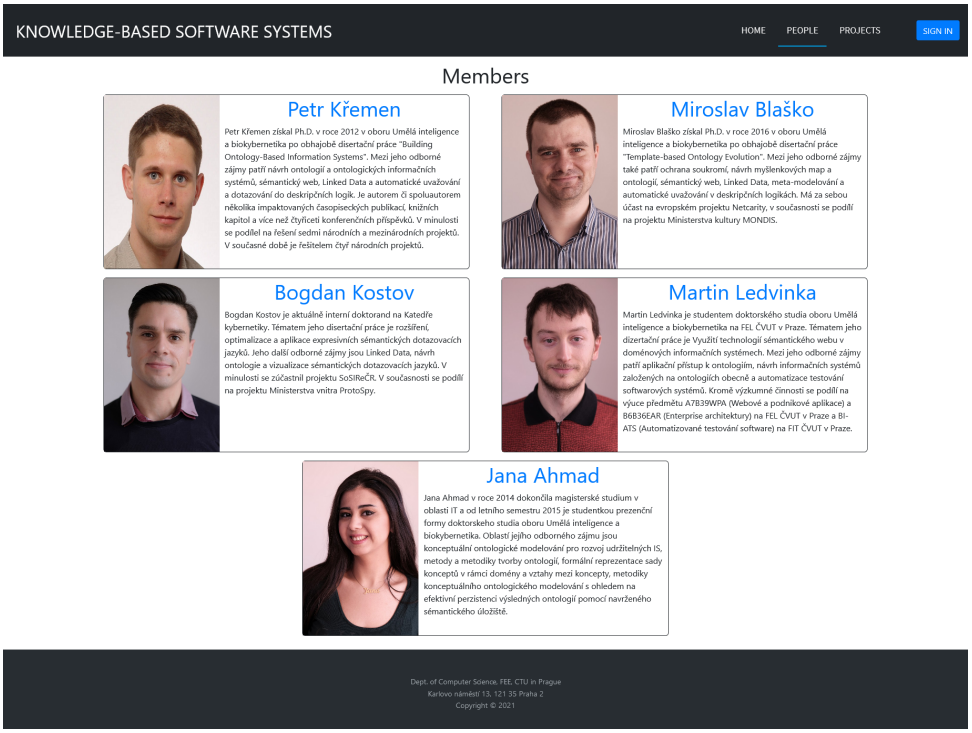
5.2.1 Hlavní stránka

Na hlavní stránce 5.2 je velký obrázek, který popisuje veškeré oblasti a témata, kterým se KBSS skupina věnuje. Dále se zde nachází v pravé části blog, do kterého mohou přidávat příspěvky pouze členové skupiny. Navíc je zde vidět navigace, která je stejná pro všechny stránky.

Obrázek 5.2: Domovská obrazovka

5.2.2 Členové

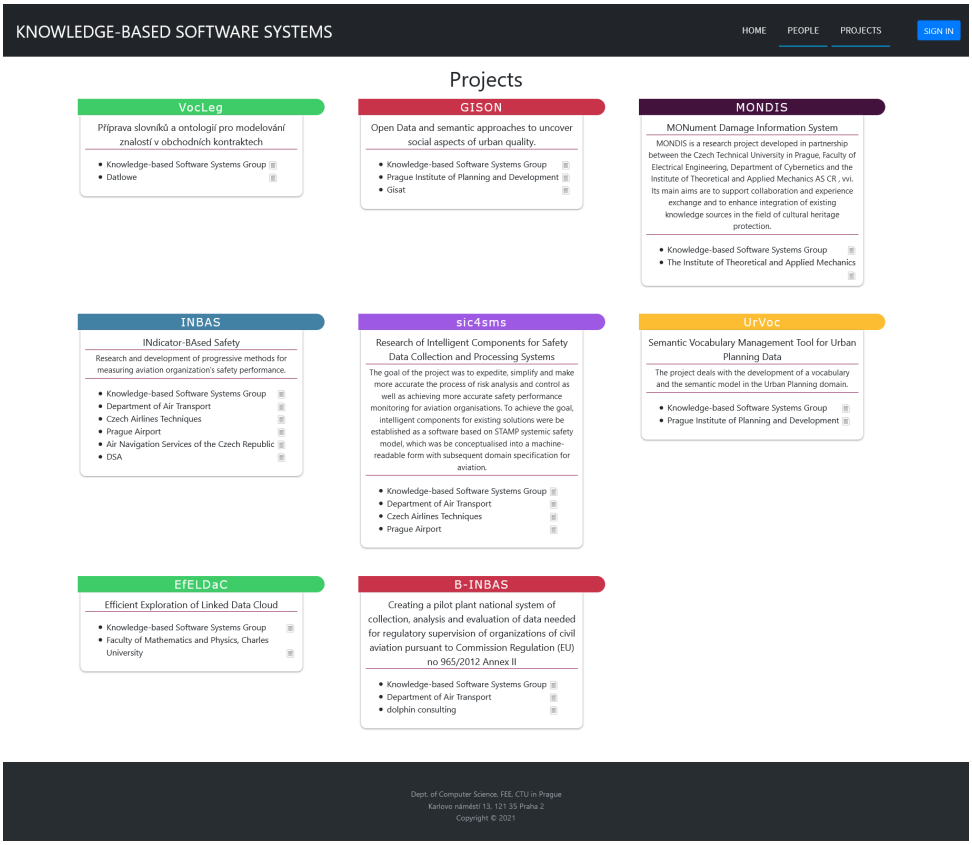
Co se týče této stránky 5.3, tak je poměrně jednoduchá, zobrazuje pouze jednotlivé členy skupiny, u kterých jsou uvedeny základní údaje včetně popisu a odborného zájmu.



Obrázek 5.3: Stránka se členy skupiny

5.2.3 Projekty

Tato stránka 5.4 zobrazuje veškeré projekty, na kterých skupina pracuje. U každého projektu je vypsán jejich akronym, název a podrobný popis, čím se projekt zabývá. Dále jsou k jednotlivých projektům připojeni i partneři, kteří na projektu spolupracují.



Obrázek 5.4: Stránka s projekty

Kapitola 6
Implementace

V této kapitole bude popsána implementace řešení. Nejprve zde popíši zvolenou architekturu a následně udělám shrnutí nasazení, výběru databáze a dalších použitých technologií.

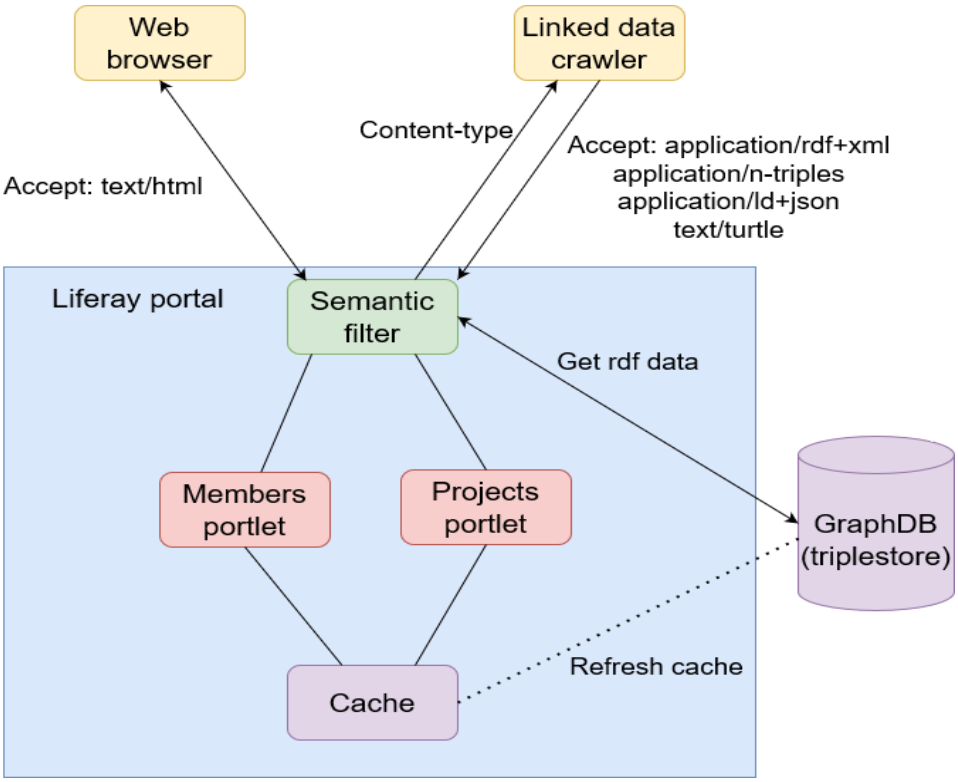
6.1 Architektura

Pro tuto aplikaci byla vybrána architektura klient-server, která funguje tak, že server čeká na požadavek od klienta, který poté zpracuje a vrátí odpověď klientovi. Tento styl se používá hlavně u webových aplikací. Co se týče struktury aplikace, tak je rozdělena do modulů, které jsou ukázány na obrázku 6.1. Díky modularizaci lze velmi snadno a rychle nasazovat moduly a také je velmi snadné rozšiřovat o další.

Nyní se podrobněji podíváme, jak přesně funguje zvolená architektura v tomto případě. Klient pošle požadavek, který nejdříve zpracuje sémantický filter, jenž rozhodne, zda požadavek zpracuje, nebo ho pošle dále do portletu. V tomto řešení jsou zakomponovány celkem čtyři moduly, které jsou následně podrobněji popsány z hlediska funkcionality:

- **Semantic filter** - V tomto modulu je implementován hook, který se spustí, pokud URL obsahuje buď /web/kbss/projects nebo /web/kbss/-projects, což jsou dvě jediné dovolené adresy, kam lze poslat požadavek na získání sémantických dat. Následně se na základě Accept hlavičky požadavku vyhodnotí, jestli aplikace vrátí klasické webové stránky nebo jestli vrátí sémantická data. Tedy pokud Accept hlavička je jedna z následujících MIME typů: application/rdf+xml, application/n-triples, application/ld+json nebo text/turtle, tak filter vytvoří Sparql dotaz, který pošle do triplestoru, kde získá RDF data, která po načtení převede do požadovaného formátu, nastaví Content-type responsu na odpovídající MIME type a vrátí tyto data klientovi. Ovšem pokud bude hlavička Acceptu text/html, tak zpracuje požadavek standardně, tzn. pošle tento požadavek do odpovídajícího portletu, který už má na starosti zpracování a zobrazení požadované stránky. Jestliže Accept type bude prázdný, tak požadavek bude zpracovávat stejně jako kdyby obsahoval text/html.

- **Members portlet** - Jediná funkce tohoto modulu je zobrazení stávajících členů. Portlet nejprve zkontroluje, zda nemá členy uložené v cache, pokud ano, tak se načtou rovnou do stránky a zobrazí uživateli. Pokud tomu ovšem tak není, tak je nutné načíst nová data a obnovit cache, na to je již připravena funkce, která upraví Sparql dotaz podle požadovaného jazyka a pošle tento dotaz do triplestoru, odkud vezme data, uloží je do cache a zobrazí je klientovi.
- **Projects portlet** - Vzhledem k tomu, že i tento modul má zobrazovací funkci, tak funguje v principu stejně jako předchozí, avšak u tohoto modulu má přihlášený uživatel dvě další funkce. Jedna z nich pouze vyčistí cache, což způsobí při dalším požadavku, že se z triplestoru přečtou aktuální data a uloží se znovu do cache. Jako druhou funkci má přihlášený uživatel možnost přidat nový projekt. Nejdříve se z triplestoru načtou dostupní partneři, kteří se zobrazí ve formuláři. Po vyplnění potřebných dat ve formuláři se data zpracují a vytvoří se z nich nový Sparql dotaz, který se pošle do triplestoru, kam přidá nový projekt.
- **KBSS theme** - Tento modul se stará o vytváření šablony pro portál, je to vlastně základní šablona použitá na všechny stránky, která obsahuje navigaci, patičku a rozložení portletů. Jelikož je telefon a tablet v dnešní době již standardem, tak šablona podporuje responzivitu, tzn.že je přizpůsobená zobrazení na menších zařízení.



Obrázek 6.1: Model architektury

6.2 Sémantická data

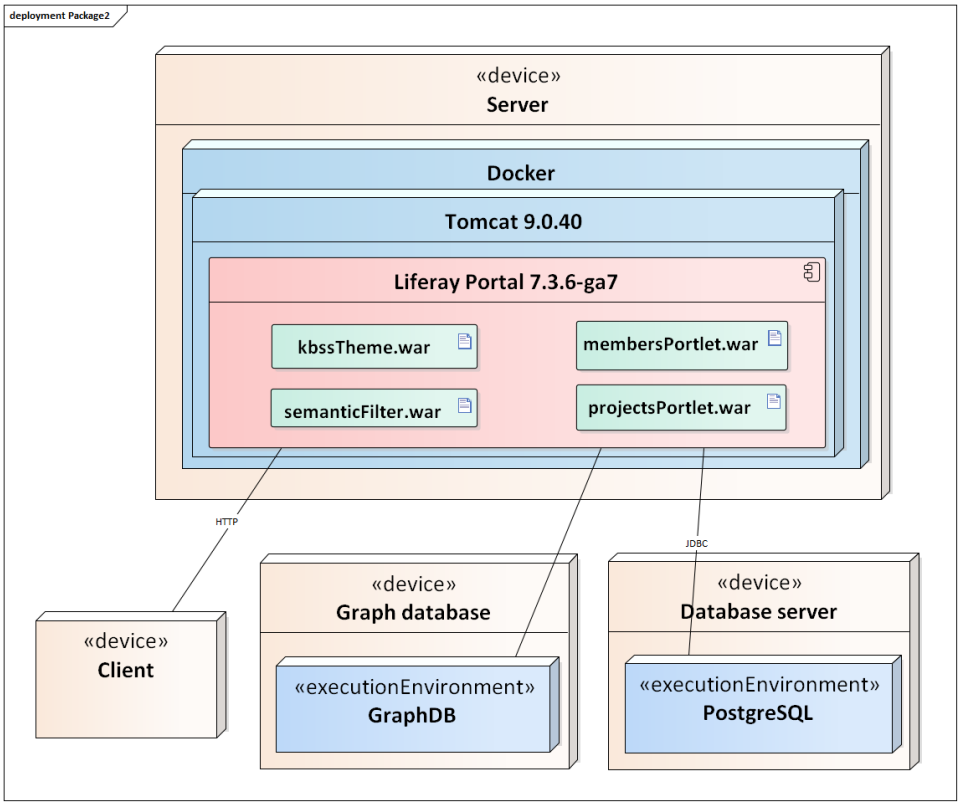
Na práci se sémantickými daty jsou použity Sparql dotazy, které se posílají do triplestoru, přičemž lze v následující části 6.1 vidět příklad na použití dotazu. Jedná o construct dotaz, který vrací nový RDF graf, podle specifikovaných podmínek. V tomto případě tento dotaz vrací všechny členy skupiny.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX p:    <http://purl.org/dc/terms/>

construct{
    ?person foaf:firstName ?firstname.
    ?person foaf:lastName  ?lastname.
    ?person p:description  ?description.
    ?person foaf:img       ?img.
    ?person foaf:workInfoHomepage   ?homepage
}WHERE{
    ?person foaf:firstName  ?firstname.
    ?person a               foaf:Person.
    ?person foaf:lastName   ?lastname.
    ?person p:description   ?desc.
    ?person foaf:img        ?img.
    ?person foaf:workInfoHomepage ?homepage.
    ?person foaf:member <http://onto.fel.cvut.cz/ontologies/
   kbss#KBSS>

    OPTIONAL {
        ?person p:description ?lang_d .
        FILTER (LANGMATCHES(LANG(?lang_d), language))
    } BIND(COALESCE(?lang_d, ?desc) as ?description)
}
```

Listing 6.1: Sparql dotaz na získání všech členů KBSS

6.3 Nasazení

Při implementaci byla aplikaci spouštěna a testována na localhostu, protože nebylo jasné, jestli bude nasazena přímo na server nebo do Dockeru. Tento způsob byl velmi výhodný především díky tomu, že Liferay balíček obsahuje Tomcat server, který stačí jenom spustit a Liferay je připravený k použití. Co se týče finálního nasazování na server KBSS, tak je nutné zmínit, že skupina používá Docker a tudíž je velmi pravděpodobné, že se bude nasazovat rovnou do Dockeru. Tento způsob nasazení popisuje obrázek 6.2 .



Obrázek 6.2: Diagram nasazení

6.4 Databáze

Na ukládání dat jsou použity celkem dvě databáze, a to relační databáze Postgres a grafová databáze GraphDB. Do relační databáze Liferay ukládá informace o portálu jako takovém, což zahrnuje uložení blogových příspěvků, údajů o uživatelích a nastavení portálu, do kterého patří šablony, název nebo přiřazení jednotlivých portletů ke stránce. Navíc díky použití Liferay je možné využít stávající data v databázi, ke které se nový portál připojí a tyto data zůstanou zachována. Co se týče druhé databáze, jedná se o grafovou databázi, ze které čerpá webová aplikace data na hlavní obsah stránek, tzn. na data o členech skupiny a projektech.

6.5 Použité technologie

- **Maven** - Tento nástroj je jeden z nejznámějších nástrojů pro infrastrukturu projektu, který se stará o správu projektu ve smyslu vytvoření, dokumentace, distribuce a hlavně závislostí jiných knihoven a frameworků[33]. Byl použit jak na celý projekt, tak i na jednotlivé portlety a díky tomu bylo velmi snadné znovu zkompilovat všechny projekty.

- **RDF4J** - Tento framework slouží k přístupu do grafové databáze GraphDB na základě Sparql dotazů, které se v tomto případě posílají z jednotlivých portletů a sémantického filtru. V následující části bude naznačen popis řešení práce s tímto frameworkem. S repozitářem se nejdříve vytvoří spojení a následně jsou získány data pomocí Sparql dotazu. Tyto data jsou pak převedena do požadovaného formátu a jsou odeslány klientovi.

- **Freemarker** - Je to šablonovací systém založený na Javě, který se používá pro MVC architekturu. Freemarker byl použit na vytvoření základního vzhledu stránky, tzn. na rozložení, navigaci, patičku a obecné vlastnosti portletu jako konfigurační ikony či menu.

- **JavaServer Pages** - Na vzhled jednotlivých portletů byly použity JavaServer Pages, což je technologie na serverové straně, která se v tomto případě stará o zobrazení obsahu stránek. Je to vlastně technologie nezávislá na platformě, která umožňuje dynamické vytváření webových aplikací. Vzhledem k tomu, že bylo nutné dynamicky měnit obsah stránky, tak je tato technologie ideální.

- **IntelliJ IDEA** - Jako vývojové prostředí jsem používal IntelliJ IDEA od firmy Jetbrains, se kterým mám již nějaké zkušenosti. Navíc lze do tohoto prostředí nainstalovat pluginy, přičemž jeden z nich je přímo napsán pro Liferay. Tento plugin umožňuje velmi dobrou spolupráci vývojového prostředí se samotným portálem jako nasazování archivů na server nebo prvotní inicializaci portálu. Ještě zde byla možnost použití vývojového prostředí Liferay Dev Studio DXP, což je prostředí šité přímo na míru pro Liferay portál, které je nadstavbou vývojového prostředí Eclipse.

Kapitola 7

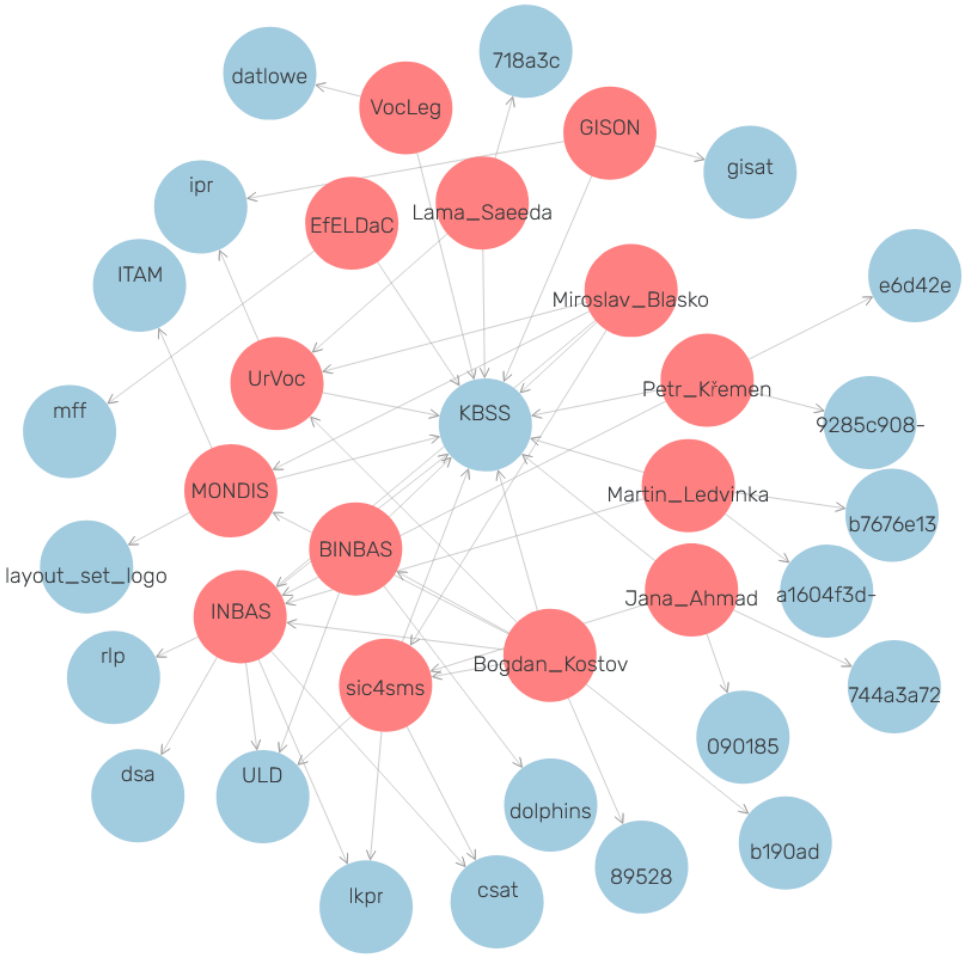
Testování

V této kapitole bude popsáno, jakým způsobem lze otestovat tuto aplikaci z hlediska sémantických dat. V tomto případě se však nejedná o typické uživatelské testování, neboť většinou běžný uživatel nepožaduje sémantická data. Tudíž je nutné ukázat, že aplikace poskytuje linked data crawlerům strojově čitelná data. Termín crawler označuje naprogramovaného bota, který systematicky prochází web a získává data. Pro tento případ byl vybrán LDSpider [1] díky doporučení a také implementaci v jazyce Java. Tento crawler nabízí práci s RDF/XML formátem a poskytuje následující strategie prohledávání:

- **The breadth-first strategy** - Je důležité zmínit, že se prohledávání provádí v cyklech a v tomto případě se nejdříve načtou všechny URI z předchozího cyklu ještě předtím, než se přejde na další cyklus. Vždy se nejdříve zpracují všechny sousední URI a až poté se jde více do hloubky každého uzlu. Tato strategie potřebuje tři parametry, přičemž první udává hloubku, jak moc se má při prohledávání ponořit. Druhý parametr limituje počet prohledávaných URI a poslední určuje limit adres, za které se obvykle musí platit, jsou to většinou nějaké subdomény domény nejvyššího řádu. Použití této strategie se doporučuje v situaci, kdy chceme získat nějaký omezený graf.
- **The load-balancing strategy** - Tato strategie potřebuje jeden parametr a to je maximální počet URI. Snaží se co nejrychleji načíst právě zadaný počet URI a je užitečná v případě, když by načtené dokumenty měly být distribuovány mezi doménami bez přetížení konkrétního serveru [34].

Vzhledem k tomu, že aplikace v době testování ještě nebyla nasazena na serveru, tak jsem ji testoval lokálně. Z výstupu LDSpideru byl vytvořen graf, který je vizualizován na obrázku 7.1, přičemž jako vstupní URI byly použity dvě následující adresy http://localhost:8080/web/kbss/people a http://localhost:8080/web/kbss/projects. LDSpider vrací primárně data ve formě N-Quads, což je formát zápisu rdf dat, který má následující syntaxi. Nejdříve začína jako standardní trojice <subjekt> <predikát> <objekt> a na čtvrté pozici je buď IRI, která identifikuje graf, do kterého předchozí trojice patří nebo prázdné místo. Tento výstup je zobrazen v příloze C.1.

[1] LDSpider [34]



Obrázek 7.1: Linked data vytvořené

Kapitola 8

Závěr

Tato práce byla zaměřena na sémantický web a systémy pro správu obsahu, přičemž byl v první řadě popsán kontext a koncept sémantického webu a poté následovalo přiblížení reprezentace dat se zaměřením na jednotlivé technologie. Následně se práce věnuje CMS systémům z hlediska kritérií pro srovnání systémů a podpory sémantického webu, načež se poté porovnají existující CMS. Další kapitola obsahuje analýzu webové aplikace, kde jsou popsány funkční a nefunkční požadavky a následně je vybrán CMS. Následující kapitola popisuje návrh z hlediska vybrané architektury, uživatelského rozhraní a zachycuje myšlenku, jakým způsobem by měly být sémantická data zakomponována do řešení. V další kapitole je popsána implementace, ve které je podrobněji rozebrána architektura a je zmíněna otázka nasazení a databází. Poslední kapitola obsahuje průběh testování a výsledky testování.

Praktický výstup této práce je webová aplikace, která nahradí stávající webové stránky výzkumné skupiny a bude zároveň podporovat sémantická data. Jelikož je aplikace implementována pomocí modulů, tak není problém ji do budoucna rozšířit o nějakou další funkcionalitu ve formě nových portletů. Co se týče přidání nových statických stránek, o to se postará Liferay.

Příloha A

Literatura

[1] *The Semantic Web Made Easy* [online].[cit. 2021-05-17] Dostupné z: https://www.w3.org/RDF/Metalog/docs/sw-easy

[2] *What Is the Semantic Web?* [online].[cit. 2021-01-22]. Dostupné z: https://www.ontotext.com/knowledgehub/fundamentals/what-is-the-semantic-web/

[3] About W3C [online].[cit. 2021-01-22]. Dostupné z: https://www.w3.org/Consortium/

[4] Christensson, P. (2006). *Metadata Definition* [online].[cit. 2021-01-16]. Dostupné z: https://techterms.com/definition/metadata

[5] Ing. Vilém Sklenák, CSc. *Metadata, sémantika a sémantický web.* [dokument].[cit. 2021-01-16]. Praha VŠE, 2004

[6] Svátek, Vojtěch. *Ontologie a sémantický web* [online].[cit. 2021-01-17]. Dostupné z: http://nb.vse.cz/~svatek/onto-www.pdf

[7] ŠMERAL, Ron. *Building of domain-specificsemantic networks from webpages* [online].[cit. 2021-01-4]. Bakalářská práce. Dostupné z: https://is.muni.cz/th/mpuyl/bc.pdf

[8] URI: The Uniform Resource Identifier Explained: *What is the Uniform Resource Identifier (URI)?* [online]. 2020 [cit. 2021-01-20]. Dostupné z: https://www.ionos.com/digitalguide/websites/web-development/uniform-resource-identifier-uri/

[9] *Internationalized Resource Identifiers (IRIs)* [online]. [cit. 2021-05-17]. Dostupné z: https://www.w3.org/International/O-URL-and-ident.html

[10] *LinkedData* [online].[cit. 2021-01-19]. Dostupné z: https://www.w3.org/wiki/LinkedData

[11] The Linked Open Data Cloud: *Subclouds by domain* [online].[cit. 2021-01-19]. Dostupné z: https://lod-cloud.net/

[12] HTML vs XML: *XML* [online].[cit. 2021-01-22]. Dostupné z: https://www.geeksforgeeks.org/html-vs-xml/

[13] Resource Description Framework (RDF): *Overview* [online].[cit. 2021-01-22]. Dostupné z: https://www.w3.org/RDF/

[14] Getting Started With RDF4J: *Introducing RDF* [online].[cit. 2021-01-22]. Dostupné z: https://rdf4j.org/documentation/tutorials/getting-started/

[15] Building the Semantic Web [online].[cit. 2021-01-23]. Dostupné z: https://www.xml.com/pub/a/2001/01/24/rdf.html/

[16] RDF Primer: *An XML Syntax for RDF: RDF/XML* [online].[cit. 2021-01-23]. Dostupné z: https://www.w3.org/TR/rdf-primer/

[17] OWL: *Overview* [online].[cit. 2021-01-22]. Dostupné z: https://www.w3.org/OWL/

[18] *15 Best and Most Popular CMS Platforms in 2021 (Compared)* [online]. [cit. 2021-01-10]. Dostupné z: https://www.wpbeginner.com/showcase/best-cms-platforms-compared/

[19] *Historical yearly trends in the usage statistics of content management systems* [online]. [cit. 2021-01-10]. Dostupné z: https://w3techs.com/technologies/history_overview/content_management/all/y

[20] *Usage statistics of content management systems* [online]. [cit. 2021-01-03]. Dostupné z: https://w3techs.com/technologies/overview/content_management

[21] *The Disadvantage of a Content Management System* [online]. [cit. 2021-01-14]. Dostupné z: http://www.drostdesigns.com/the-disadvantage-of-a-content-management-system/

[22] Wordpress (CMS): *What is Wordpress (CMS)* [online]. [cit. 2021-01-14]. Dostupné z: https://www.investopedia.com/terms/w/wordpress-cms.asp

[23] wp-linked-data: *Description* [online]. [cit. 2021-01-22]. Dostupné z: https://wordpress.org/plugins/wp-linked-data/#description

[24] Rich Meta in RDFa: *Description* [online]. [cit. 2021-01-22]. Dostupné z: https://wordpress.org/plugins/rich-meta-in-rdfa/

[25] Popular CMS by Market Share: *The 10 Most Popular CMS Platforms: Shopify* [online]. [cit. 2021-01-14]. Dostupné z: https://websitesetup.org/news/popular-cms/

[26] What is Shopify?: *Shopify Technology* [online]. [cit. 2021-01-14]. Dostupné z: https://dropshipnews.com/what-is-shopify-all-about-and-how-does-it-work/

[27] Joomla: *Definition - What does Joomla mean?* [online]. [cit. 2021-01-14]. Dostupné z: https://www.techopedia.com/definition/3276/joomla

[28] Výběr toho správného redakčního systému pro Vaše stránky: *Drupal* [online]. [cit. 2021-01-08]. Dostupné z: https://www.ovh.cz/webhosting/website/porovnani-cms/

[29] A Little About Drupal Semantic Web: *Drupal And Schema: The Story So Far* [online]. [cit. 2021-01-23]. Dostupné z: https://opensenselabs.com/blogs/articles/little-about-drupal-semantic-web

[30] Liferay, lídr mezi portály *Liferay, to je především open-source* [online]. [cit. 2021-01-13]. Dostupné z: https://orchi.tech/blog/liferay-lidr-mezi-portaly/

[31] Liferay: *Choosing Liferay* [online]. [cit. 2021-01-16]. Dostupné z: https://www.omegabit.com/liferay

[32] Liferay: *OSGi and Modularity* [online]. [cit. 2021-05-4]. Dostupné z: https://help.liferay.com/hc/en-us/articles/360018162431-OSGi-and-Modularity-

[33] Maven: *What is maven?* [online]. [cit. 2021-05-7]. Dostupné z: https://maven.apache.org/guides/getting-started/index.html

[34] LDspider: *Isele, Robert & Umbrich, Jürgen & Bizer, Christian & Harth, Andreas. (2010). LDspider: An Open-source Crawling Framework for the Web of Linked Data.. 658* [online]. [cit. 2021-05-12]. Dostupné z: https://www.researchgate.net/publication/221467189_LDspider_An_Open-source_Crawling_Framework_for_the_Web_of_Linked_Data

Příloha B

Seznam použitých zkratek

HTTP	HyperText Transfer Protocol
KBSS	Knowledge-Based Software Systems
XML	Extensive Markup Language
JAR	Java archive
WAR	Web archive
CMS	Content management system
WYSIWYG	What you see is what you get
URI	Uniform Resource Identifier
MIME	Multipurpose Internet Mail Extensions
HTML	Hypertext Markup Language
XHTML	Extensible Hypertext Markup Language
SaaS	Software as a Service
SSL	Secure Sockets Layer

Příloha C

Výstup z LDSpidera

Listing C.1: Zkácený výstup z LDSpidera

<http://onto.fel.cvut.cz/ontologies/kbss#Miroslav_Blasko> <http://www.w3.org/2002/07/owl#sameAs> <http://onto.fel.cvut.cz/ontologies/kbss#Miroslav_Blasko> <http://localhost:8080/web/kbss/people> .

<http://onto.fel.cvut.cz/ontologies/kbss#Miroslav_Blasko> <http://xmlns.com/foaf/0.1/member> <http://onto.fel.cvut.cz/ontologies/kbss#KBSS> <http://localhost:8080/web/kbss/people> .

<http://onto.fel.cvut.cz/ontologies/kbss#Miroslav_Blasko> <http://xmlns.com/foaf/0.1/pastProject> <http://onto.fel.cvut.cz/ontologies/kbss#INBAS> <http://localhost:8080/web/kbss/people> .

<http://onto.fel.cvut.cz/ontologies/kbss#Miroslav_Blasko> <http://xmlns.com/foaf/0.1/pastProject> <http://onto.fel.cvut.cz/ontologies/kbss#MONDIS> <http://localhost:8080/web/kbss/people> .

<http://onto.fel.cvut.cz/ontologies/kbss#Miroslav_Blasko> <http://xmlns.com/foaf/0.1/pastProject> <http://onto.fel.cvut.cz/ontologies/kbss#UrVoc> <http://localhost:8080/web/kbss/people> .

<http://onto.fel.cvut.cz/ontologies/kbss#Miroslav_Blasko> <http://xmlns.com/foaf/0.1/pastProject> <http://onto.fel.cvut.cz/ontologies/kbss#sic4sms> <http://localhost:8080/web/kbss/people> .

<http://onto.fel.cvut.cz/ontologies/kbss#Miroslav_Blasko> <http://xmlns.com/foaf/0.1/firstName> "Miroslav" <http://localhost:8080/web/kbss/people> .

<http://onto.fel.cvut.cz/ontologies/kbss#Miroslav_Blasko> <http://xmlns.com/foaf/0.1/lastName> "Bla\u0161ko" <http://localhost:8080/web/kbss/people> .

<http://onto.fel.cvut.cz/ontologies/kbss#Miroslav_Blasko> <http://xmlns.com/foaf/0.1/img> <http://cs.fel.cvut.cz/upload/persons/5bac11b845c7e4e12fb719c8ca2346d46d892f9b.jpg> <http://localhost:8080/web/kbss/people> .

<http://onto.fel.cvut.cz/ontologies/kbss#Miroslav_Blasko> <http://xmlns.com/foaf/0.1/workInfoHomepage> <https://usermap.cvut.cz/profile/18c18d68-6d75-475e-a3f5-b57e7f906e1d> <http://localhost:8080/web/kbss/people> .

<http://onto.fel.cvut.cz/ontologies/kbss#Bogdan_Kostov> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.w3.org/2002/07/owl#Thing> <http://localhost:8080/web/kbss/people> .

<http://onto.fel.cvut.cz/ontologies/kbss#Bogdan_Kostov> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.w3.org/2002/07/owl#NamedIndividual> <http://localhost:8080/web/kbss/people> .

<http://onto.fel.cvut.cz/ontologies/kbss#Bogdan_Kostov> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://xmlns.com/foaf/0.1/Person> <http://localhost:8080/web/kbss/people> .

<http://onto.fel.cvut.cz/ontologies/kbss#Bogdan_Kostov> <http://www.w3.org/2002/07/owl#sameAs> <http://onto.fel.cvut.cz/ontologies/kbss#Bogdan_Kostov> <http://localhost:8080/web/kbss/people> .

<http://onto.fel.cvut.cz/ontologies/kbss#Bogdan_Kostov> <http://xmlns.com/foaf/0.1/member> <http://onto.fel.cvut.cz/ontologies/kbss#KBSS> <http://localhost:8080/web/kbss/people> .

<http://onto.fel.cvut.cz/ontologies/kbss#Bogdan_Kostov> <http://xmlns.com/foaf/0.1/pastProject> <http://onto.fel.cvut.cz/ontologies/kbss#INBAS> <http://localhost:8080/web/kbss/people> .

<http://onto.fel.cvut.cz/ontologies/kbss#Bogdan_Kostov> <http://xmlns.com/foaf/0.1/pastProject> <http://onto.fel.cvut.cz/ontologies/kbss#MONDIS> <http://localhost:8080/web/kbss/people> .

<http://onto.fel.cvut.cz/ontologies/kbss#Bogdan_Kostov> <http://xmlns.com/foaf/0.1/pastProject> <http://onto.fel.cvut.cz/ontologies/kbss#BINBAS> <http://localhost:8080/web/kbss/people> .

<http://onto.fel.cvut.cz/ontologies/kbss#Bogdan_Kostov> <http://xmlns.com/foaf/0.1/pastProject> <http://onto.fel.cvut.cz/ontologies/kbss#UrVoc> <http://localhost:8080/web/kbss/people> .

<http://onto.fel.cvut.cz/ontologies/kbss#Bogdan_Kostov> <http://xmlns.com/foaf/0.1/pastProject> <http://onto.fel.cvut.cz/ontologies/kbss#sic4sms> <http://localhost:8080/web/kbss/people> .

<http://onto.fel.cvut.cz/ontologies/kbss#Bogdan_Kostov> <http://xmlns.com/foaf/0.1/firstName> "Bogdan" <http://localhost:8080/web/kbss/people> .

<http://onto.fel.cvut.cz/ontologies/kbss#Bogdan_Kostov> <http://xmlns.com/foaf/0.1/lastName> "Kostov" <http://localhost:8080/web/kbss/people> .

Příloha D

Obsah elektronické přílohy

- `git_repozitar` ..... Odkaz na zdrojové kódy
- `kbss_website`
  - `semanticFilter` ..... Sémantický filtr
    - `src/main/java/filters` ..... KBSS šablona pro portál
      - `Queries` ..... Sparql dotazy
      - `SemanticFilter` ..... Implementace filtru
    - `src/main/webapp/WEB-INF`
      - `liferay-hook.xml` ..... Konfigurace hooku
      - `liferay-plugin-package.properties` ..... Popis hot deploye
      - `web.xml` ..... Deskriptor webové aplikace
  - `kbssTheme` ..... KBSS šablona pro portál
    - `src/main/webapp/css` ..... Kaskádové styly šablony
    - `src/main/webapp/templates` ..... Šablony stránek pro portál
  - `membersPortlet` ..... Portlet členů skupiny
    - `src/main`
      - `java/membersportlet` ..... Zdrojové kódy portletu
      - `webapp` ..... Konfigurace a vzhled portletu
  - `projectsPortlet` ..... Portlet projektů skupiny
    - `src/main`
      - `java/projectsportlet` ..... Zdrojové kódy portletu
      - `webapp` ..... Konfigurace a vzhled portletu
  - `wars` ..... Vygenerované webové archivy
  - `portal_settings.lar` ..... Exportované nastavení portálu KBSS
  - `README.md` ..... Popis návodu nasazení na Docker