ČESKÉ VYSOKÉ
UČENÍ TECHNICKÉ
V PRAZE

F3

Fakulta elektrotechnická
Katedra počítačů

Bakalářská práce

Sémantické facetové vyhledávání na platformě React

Daniel Bourek
Softwarové inženýrství a technologie

Leden 2022
Vedoucí práce: Ing. Martin Ledvinka

Poděkování / Prohlášení

Tímto bych rád poděkoval vedoucímu této práce Ing. Martinovi Ledvinkovi za výborné vedení, mnoho doporučení a velkou ochotu na konzultacích při vypracovávání této práce. Dále bych chtěl poděkovat své rodině za jejich neustálou podporu při mých studiích.

Prohlašuji, že jsem předloženou práci vypracoval samostatně.

Daniel Bourek V Praze, 16. 1. 2022

Abstrakt / Abstract

Práce se zabývá problematiku facetového vyhledávání s přihlednutím na sémantická data. Zprvu čtenáře seznámi s pojmy a technologiemi sémantického webu. Pak zanalyzuje přístupy k facetovému vyhledávání a navrhne vhodný pro sémantická data. Součástí práce je pak návrh prototypu sémantického facetového vyhledávače na platformě React s důrazem na jeho přepoužitelnost. Ten je také výsledkem této práce.

The thesis deals with the issue of facet searching with regard to semantic data. First, it introduces the reader to the concepts and technologies of the Semantic Web. It then analyzes approaches to facet search and suggests suitable one for semantic data. Core of the work is the development of a semantic facet search prototype using Javascript framework React and emphasising on the reusabilty of this module. Thesis is successful in fulfilling its goals.

/ Obsah

Kapitola 1
Úvod

Málokterý vynález ovlivnil svět v takové míře jako vznik World Wide Web (zkráceně WWW či web). Za poměrně krátkou dobu své existence se web rozšířil téměř do každé části našeho života a dnes si bez něj lze svět jen těžko představit. Oproti ostatním ICT technologiím, které se často výrazně inovují a mění každých několik let, web funguje už 20 let téměř stejně. To se však začíná měnit s příchodem sémantického webu, který zásadně ovlivňuje, jak přistupujeme k datům v internetu – místo relací mezi dokumenty (hypertextové odkazy) můžeme vytvářet relace mezi fakty. Svět lze tak mnohem lépe popsat a stává se pro nás srozumitelnější. Navíc jsou tyto relace lépe strojově čitelné, takže se nestává srozumitelnější jen pro nás, ale i pro stroje. Ty pak můžou nad těmito daty mnohem přesněji vyhledávat informace či vykonávat automatizace.

Tato změna si žádá nové přístupy k ukládání, zpracování a vyhledávání dat. Právě vyhledáváním v sémantických datech se zabývá tato práce, konkrétně facetovým vyhledáváním. Facetové vyhledávání, tedy zatřídění vyhledaných výsledků do různých kategorií, je v současné době velmi rozšířené. Pomáhá nám upřesnit výsledky vyhledávání a najdeme jej tedy třeba skoro v každém větším e-shopu. Přístupů k facetovému vyhledávání je více, ne všechny jsou však vhodné pro sémantická data. Nad sémantickými daty tak existuje velmi málo řešení facetového vyhledávání a k tomu jsou často závislé na nějaké platformě. Cílem této práce je tak:

1. Srovnat existující přístupy k facetovému vyhledávání, především pak z hlediska využití sémantických technologií.
2. Navrhnout modul sémantického facetového vyhledávače, který bude umožňovat rozdělení vyhledávání a jeho vizualizace do samostatných modulů.
3. Naimplementovat prototyp modulu sémantického facetového vyhledávače na platformě React.

Kapitola 2
Sémantický web

V této kapitole se seznámíme s pojmem sémantický web a popíšeme si klíčové technologie týkající se tohoto pojmu. Ty jsou zásadní pro pochopení fungování světa sémantických dat, a tak tedy i k pochopení této práce.

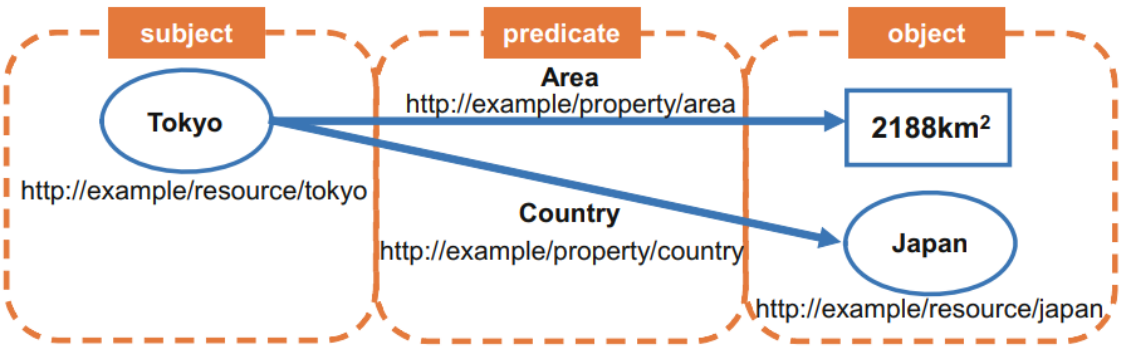
2.1 Co je sémantický web?

Myšlenka sémantického webu byla poprvé veřejnosti předvedena 17. května 2001, kdy v časopise Scientific American vyšel článek The Semantic Web [1]. Autory tohoto článku byli Tim Berners-Lee (zakladatel WWW), James Hendler a Ora Lassila, všichni tři jsou zásadními postavami ve vývoji sémantického webu. Na začátku tohoto článku popisují poměrně futuristickou scénku, kde po otevření webové stránky je zařízení schopné samo kompletně porozumět obsahu této stránky. Tedy veškerým informacím na ní napsané, včetně odkazů na jiné stránky a vztahů mezi nimi. Díky tomu pak pouze skrze komunikaci s dalšími stránkami naplánuje návštěvu lékaře, včetně toho, aby vyhovoval jejím časovým podmínkám, byl blízko domu či byl pokryt její pojišťovnou. Klíčové je zde to, že to zařízení zvládlo jen za pomocí webu, díky strojově čitelným standardizovaným datům na webových stránkách.

Sémantický web se tak má stát novým evolučním stupněm stávajícího webu (velmi často je nazýván také jako Web 3.0, ale fakticky je spíše jeho částí), kde jsou informace uloženy podle standardizovaných pravidel, což usnadňuje jejich vyhledávání a zpracování [2]. Ony standardizované pravidla jsou hlavně Resource Description Framework (RDF) a Web Ontology Language (OWL). Ty byly vyvinuty mezinárodním konsorciem W3C, které ve společnosti s veřejností vyvíjí i jiné webové standardy, pomocí nichž, chtějí rozvinout web do plného potenciálu. Pro ověření pravosti dokumentů (a jejich informací) využívá sémantický web také třeba digitální podpisy a šifrování [3].

2.2 Úvod do sémantických technologií

2.2.1 RDF



Obrázek 2.1. Ukázka RDF trojice, kde je každá entita identifikovaná svou URI [4].

V sémantickém světě je standardem pro vytváření dat formát RDF [5]. RDF je standardizovaný strojově čitelný grafový formát, ve kterém se využívají tzv. triples, česky trojice, k popsání relací (lze zapsat jako orientované hrany v grafu) ve formátu subjekt - predikát - objekt. Samotná syntaxe však definovaná není, často se však používá RDF/XML, která dokáže zapsat RDF graf jako XML dokument. Pro účely dnešních aplikací je nutné také zmínit existenci formátu RDFJS, který reprezentuje RDF data v jazyku Javascript [6].

2.2.2 OWL

Pro popsání základních ontologií vzniklo RDF Schema (RDFS), které obsahuje sadu základních tříd k použití [7]. Později se však vyvinul Web Ontology Language (OWL), který je mnohem bohatší a používá se tak pro popis informací o věcech a vztahů mezi nimi neboli ontologií [8]. Oba jazyky jsou standardem W3C.

V informatice je ontologie explicitní a formalizovaný popis určité problematiky. Datový model se sestává:

- Entita (objekt, jedinec, instance) je základní stavební prvek datového modelu ontologie. Entita může být konkrétní (člověk, tabulka, molekula) nebo abstraktní (číslo, pojem, událost).
- Kategorie (třída) je množina entit určitého typu. Podmnožinou kategorie je podkategorie. Kategorie může obsahovat zároveň entity i podkategorie.
- Atribut popisuje určitou vlastnost, charakteristiku či parametr entity. Každý atribut určité entity obsahuje přinejmenším název a hodnotu. Atribut je určen pro uložení určité informace vztahující se k dané entitě.
- Vazba je jednosměrné nebo obousměrné propojení dvou entit. Je možné říci, že vazba je určitým typem atributu, jehož hodnotou je jiná entita v ontologii.

2.2.3 Linked data

Výše popsané technologie umožňují vznik standardizovaným strukturovaným datům, které nazýváme Linked Data. Ty se řídí 4 principy:

- K identifikaci jednotlivých věci se používá URI.
- URI by měla být otevřená přes HTTP k vyhledání a interpretaci věci na internetu.
- HTTP URI by měla obsahovat užitečné informace.
- HTTP URI by měli být používané k odkazování na věci související (relace) s vaší URI.

Mezi největší Linked Data sety patří DBPedia.org - obdoba Wikipedia.org se sémantickými daty či GeoNames, popisující geografii.

2.2.4 SPARQL

Primárním dotazovacím jazykem pro RDF je SPARQL [9]. Ten je syntaxí velmi podobný SQL, funguje však spíše na porovnávání a dosazování oněch trojic - tedy potažmo orientovaných hran grafu. Má více druhů dotazů:

- SELECT – podobný SQL SELECT dotazu, tedy vrací data vyhovující dotazu
- CONSTRUCT – vrací výsledek dotazu jako nová data ve formátu RDF vyhovující dotazu
- ASK – vrací boolean hodnotu true/false odpovídající dotazu
- DESCRIBE – vráci RDF podobu toho, jak by vypadali data vyhovující dotazu

Dále obsahuje klauzule jako BIND k přiřazování hodnot k proměnným či OPTIONAL, který je podobný k nepovinnému JOIN z SQL.

Je zvykem, že stránky se sémantickými daty obsahují SPARQL endpoint, kam lze zadávat dotazy. Často se využívá řešení od firmy OpenLink jménem Virtuoso.

Kapitola 3
Facetové vyhledávání

V této kapitole si popíšeme, co je facetové vyhledávání a k čemu se primárně využívá. Zanalyzujeme a srovnáme pak různé přístupy k implementaci facetového vyhledávání, především z hlediska využití sémantických technologií. Abychom získali přehled o používaných řešení facetového vyhledávání, zanalyzujeme poskytované Facet Search APIs největších společností v této oblasti jako Elastic či Solr.

3.1 Popis

Obrázek 3.1. Ukázka facetů s vysvětlivkami [10].

Facetové vyhledávání je zatřídění vyhledaných výsledků do různých kategorií (facetů) dle kterých se dá sada výsledků dále filtrovat. Dá se ním tak obohatit každé vyhledávání, ale často bývá spojeno s fulltextovým vyhledáváním, aby uživateli umožnilo jeho dotaz dále upřesnit. Hojně se využívá třeba v e-commerce sektoru, kde podle studie Nielsen Norman Group (NNG) z roku 2018 jsou e-shopy bez facetového vyhledávání výjimkou [11]. Jelikož není definovaný žádný standard facetového vyhledávání, zadefinujeme si, co by měl takový modul facetového vyhledávání splňovat:

- facet obsahuje hodnotu pro každý výsledek ze sady výsledků
- jednotlivé facety lze kombinovat mezi sebou
- mezi kritérii facetů platí logický AND (ne pouze OR), tzn. aby se výsledek objevil v sadě výsledku, musí vyhovět všem aktivním facetům
- hodnoty facetů ukazují počet výsledků, které aplikování facetu s danou hodnotou v aktuálním stavu vrátí

- hodnoty facetů, které by vrátily prázdnou sadu výsledků se nezobrazují nebo jsou „disabled“[1]

3.2 Typy facetů

Facety si můžeme rozřadit dle toho jakým způsobem se volí jejich hodnoty. V této práci budeme tyto kategorie nazývat jako typy facetů.

3.2.1 Select facet

Facet s možností volby nejvýše jedné hodnoty podle které je pak sada výsledků filtrována. Ovládacím prvkem bývá select element.

3.2.2 Checkbox facet

Facet s možností volby více hodnot skrz zaškrtávání checkboxů.

3.2.3 Range facet

Facet pro číselná data s možností nastavení rozsahu. Ovládacím prvkem bývá posuvník (input element s hodnotou atributu type range).

3.2.4 Bucket facet

Podobné jako range facet, akorát se neovládá posuvníkem, ale jsou nadefinovány rozsahy, do kterých se pak výsledky roztřídí.

3.3 Srovnání přístupů

Řešení jak implementovat facetové vyhledávání je více. Obecně je lze rozdělit podle toho, kde dochází k filtraci výsledků aktivními facety, tedy jestli na straně klienta či na straně serveru. U implementace facetového vyhledávání je také nutné myslet na to jakým způsobem se budou plnit data facetů. Nejčastěji se setkáváme s tím, že se hodnoty facetů posílají ve stejné response jako sada výsledků.

3.3.1 Filtrování na straně klienta

Při filtrování na klientu nám server vždy zašle celou výsledkovou sadu, kterou při zpracování vyfiltrujeme dle aktivních facetů a zobrazíme jen vyfiltrované výsledky. To znamená, že sadu výsledků stačí teoreticky zaslat jen jednou a případně lehce „zacachovat“. Nemusíme tak také řešit zasílaní facetů v komunikaci se serverem a ta se tak stává vcelku přímočará a přehledná. S tím je však ale spojená i hlavní nevýhoda – sada výsledků může být velká, což znamená velké množství dat, které musíme přenést internetem, uložit na klientovi a následně vyfiltrovat. To může při pomalejším spojení nebo malém výpočetním výkonu na klientovi trvat delší dobu.

[1] Jejich HTML ovládací prvek má atribut disabled.

3.3.2 Filtrování na straně serveru

Filtrování na serveru většinu „tvrdé dřiny“ přenáší na server, což přirozeně zvyšuje jeho zatížení. Musíme serveru spolu s požadavkem předat jaká data chceme (aktivní facety) a on nám musí zpátky poslat již vyfiltrovanou sadu výsledků a nové hodnoty pro tyto facety (společně s počtem jejich výskytů). Zároveň to ale znamená, že pokud je server správně implementován, může přidat filtrování už do dotazu do databáze a celý proces tak výrazně zrychlit. Pokud k tomu ještě přidáme fakt, že server vrací vyloženě jen data, které požadujeme a kterých velikost tak může být značně nižší (a přenos rychlejší) jedná se tak jednoznačně o rychlejší metodu. Zároveň je toto řešení i spolehlivější a lépe škálovatelné.

3.4 Převedení do sémantického světa

Jak jsme zmínili v minulé kapitole, sémantické data jsou dotazovány pomocí SPARQL. V praxi to znamená, že k získání dat musíme zkonstruovat dotaz v syntaxi SPARQL a poslat požadavek na publikovaný SPARQL endpoint. Jelikož se tento endpoint dá provolit i z klienta, nepotřebujeme vůbec server pro facetový vyhledávač. Vyplývá nám z toho, že pro sémantická data je lepší použít filtrování už jako součást SPARQL dotazu. Při konstrukci dotazu je nutné myslet, ale i na to, aby se nám vrátila data s hodnotami facetů. Toho lze docílit použití klauzule OPTIONAL. Výsledné řešení pak bude něco mezi oběma popsanými přístupu v minulé sekci.

Kapitola 4
Implementace

Výsledek této práce je prototyp sémantického facetového vyhledávače na platformě React. Tento prototyp je popsán v této kapitole, společně s návrhem jeho architektury a zdůvodněním některých rozhodnutí při vývoji. Prvně však zmíníme použité technologie a knihovny.

4.1 Technologie a knihovny

Jako součást zadání této práce bylo rozhodnuto, že na implementaci prototypu bude využit javascriptový framework React. Ten je v současné době velmi populární a pro naše potřeby vhodný, díky jeho modularizaci. Použili jsme tedy doporučovanou metodu create-react-app k vytvoření základních modulů. Abychom se nemuseli zaobírat UI prvky prototypu, využili jsme knihovnu MUI (dříve známá také jako Material-UI) a použili připravené komponenty z ní. Druhá knihovna, kterou využíváme je fetch-sparql-endpoint [12]. Ta nám zjednodušuje volání SPARQL endpointu a serializaci příchozích dat do formátu RDFJS. Vybrali jsme ji hlavně proto, že je skutečně jednoduchá, neimplementuje nepotřebné věci navíc a je udržována aktuální. Všechen kód je napsaný v Typescriptu, což je nadstavba Javascriptu, která jej rozšiřuje o statické typování a předchází tak spoustě chyb.

4.2 Architektura

Při návrhu architektury je nutné dbát na to, aby byl modul vyhledávací logiky nezávislý na použitou platformu. Toho lze docílit buďto jeho realizací jako samostatné serverové služby nebo jako modul bez využití konstruktů využité platformy. Rozdělili jsme tedy vyhledávač na modul vyhledávací logiky napsaný v čistém Typescriptu se jménem sfs-api a modul napojení na prezentační vrstvu (web elementy) určený pro React nazvaný sfs-ui. Prototyp pak oba tyto moduly importuje, aby je ukázal na příkladu.

Pro použití modulu sfs-api musíme vytvořit instanci typu FacetSearchApi, kde předáváme konfiguraci požadovaných facetů a SPARQL endpointu.

```
export const sfsApi = new FacetSearchApi({
  endpointUrl: "https://dbpedia.org/sparql",
  facetConfigs: facetConfigs,
  prefixes: `
        PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
        PREFIX dbp: <http://dbpedia.org/property/>
        PREFIX dbo: <http://dbpedia.org/ontology/>
        PREFIX foaf: <http://xmlns.com/foaf/0.1/>
      `
});
```

Facet je reprezentován rozhraním Facet a má své unikátní facetId, dle kterého je dále identifikován. Jednotlivé typy facetů pak toto rozhraní rozšiřují a implementují jeho metodu „generateSparql“, která se volá při sestavení výsledného SPARQL dotazu. Vracet by měla část SPARQL týkající se tohoto facetu. Podpora dalších typů je plánovaná s dalším vývojem vyhledávače. Tento design umožňuje uživateli vytvořit si vlastní typ facetu (implementováním interfacu Facet) pro případy nepokryté knihovnou, s tím, že ho ale stále může kombinovat s ostatními facety.

K připojení jednotlivých facetů používáme modul sfs-ui. Ten za pomoci implementovaného React hooku spojí prezentační vrstvu se stavem facetu a je tak bodem komunikace s druhým modulem. K tomu se používá hlavně subscriber pattern, kde je identifikátorem právě ono facetId.

Kapitola 5
Závěr

V práci jsme se seznámily s pojmem sémantický web a popsali si jeho klíčové technologie. Poté jsme si představily problematiku facetového vyhledávání, kde jsme si zadefinovali jak by to měl vypadat faceotový vyhledávač a zanalyzovali způsoby jak jej implementovat. S těmito poznatky jsme navrhli a následně implementovali prototyp sémantického facetového vyhledávače. Celý proces návrhu jsme zdokumentovali. Postupně jsme tak splnili všechny cíle této práce.

Implementovaný prototyp je však stále doopravdy jen prototypem a do finální podoby má daleko. To se ale v budoucnosti změní, jelikož bude dále rozvíjen jako součást bakalářské práce a následně publikován jako open source balíček. Pokud bude finální vyhledávač implementovaný dobře, mohl by se začít používat jako primární dostupné řešení facetového vyhledávání pro sémantická data. Nahradit by mohl i vyhledávač pro Sémantický slovník pojmů udržovaný Ministerstvem Vnitra České Republiky (MVČR) ve společnosti s FEL ČVUT, který je nevyhovující.

Ačkoliv vývoj sémantického webu v minulém desetiletí spíše stagnoval a nepodařilo se mu rozšířit tak jak si jeho autoři přáli, v posledních letech se to značně mění. S průlomy v posledních v oblasti umělé inteligence či internetu věcí se opět o sémantickém webu mluví jako o jedné z přicházejících klíčových technologií ve vývoji webu a internetu a šlo by tak očekávat jeho výrazný vývoj přímo v následující letech. Jestli se predikce potvrdí ukáže čas, ale je možné, že i tato práce pak bude využívána v nově vzniklých aplikacích jako řešení sémantického facetového vyhledávání.

Příloha A
Slovníček

API	Application programming interface
ČVUT	České vysoké učení technické v Praze
FEL	Fakulta elektrotechnická ČVUT
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
ICT	Informační a komunikační technologie
OWL	Web Ontology Language
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
SPARQL	SPARQL Protocol and RDF Query Language
SQL	Structured Query Language
URI	Uniform Resource Identifier
WWW	World Wide Web
W3C	World Wide Web Consortium
XML	Extensible Markup Language

Literatura

[1] Sir Tim Berners-Lee, James Hendler a Ora Lasilla. The Semantic Web. *Scientific American*. 2001, 2001 (Vol. 284. No. 5), 34-43. DOI https://www.jstor.org/stable/10.2307/26059207.

[2] *Semantic University*. 2021. `https://cambridgesemantics.com/blog/semantic-university/intro-semantic-web`.

[3] *The Security of the Semantic Web - Secrecy, Trust and Rationality*. 2003. `https://www.w3.org/People/n-shiraishi/work/Security-of-RDF.html`.

[4] *Understanding Linked Data Formats*. 2019. `https://rdf.community/understanding-linked-data-formats`.

[5] *RDF 1.1 Concepts and Abstract Syntax*. 2014. `https://www.w3.org/TR/rdf11-concepts`.

[6] *RDF/JS: Data model specification*. 2020. `https://rdf.js.org/data-model-spec`.

[7] *RDF Schema 1.1*. 2014. `https://www.w3.org/TR/rdf-schema`.

[8] *OWL 2 Web Ontology Language Document Overview (Second Edition)*. 2012. `https://www.w3.org/TR/owl2-overview`.

[9] *SPARQL 1.1 Query Language*. 2013. `https://www.w3.org/TR/sparql11-query`.

[10] *What is faceted search*. 2018. `https://stackoverflow.com/questions/5321595/what-is-faceted-search`.

[11] *The State of Ecommerce Search*. 2018. `https://www.nngroup.com/articles/state-ecommerce-search`.

[12] *Fetch SPARQL Endpoint repository*. 2021. `https://github.com/rubensworks/fetch-sparql-endpoint.js`.