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

Květen 2022
Vedoucí práce: Ing. Martin Ledvinka

Draft: 4. 5. 2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Bourek** Jméno: **Daniel** Osobní číslo: **478425**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Sémantické facetové vyhledávání na platformě React

Název bakalářské práce anglicky:

React-based Semantic Faceted Search

Pokyny pro vypracování:

1. Srovnejte existující přístupy k facetovému vyhledávání, především pak z hlediska využití sémantických technologií.
2. Navrhněte modul sémantického facetového vyhledávače, který bude umožňovat rozdělení vyhledávání a jeho vizualizace do samostatných modulů.
3. Naimplementujte navržené řešení včetně vizualizačního modulu.
4. Ověřte funkčnost řešení srovnáním s existujícím facetovým vyhledávačem na stránkách https://slovník.gov.cz/prohlížeč.

Seznam doporučené literatury:

[1] D. Allemang, J. Hendler, Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL, Morgan Kaufmann, 2011
[2] R. Wieruch, The Road to learn React: Your journey to master plain yet pragmatic React.js, 2018
[3] G. M. Sacco, Y. Tzitzikas, Dynamic Taxonomies and Faceted Search: Theory, Practice, and Experience, Springer, 2009

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Martin Ledvinka, Ph.D. skupina znalostních softwarových systémů FEL

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

Datum zadání bakalářské práce: **02.02.2022** Termín odevzdání bakalářské práce: **20.05.2022**

Platnost zadání bakalářské práce: **30.09.2023**

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

Draft: 4. 5. 2022

Poděkování / Prohlášení

Rád bych 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, 15. 5. 2022

Abstrakt / Abstract

Práce se zabývá problematiku facetového vyhledávání se zaměřením na jeho použití pro 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 modulu sémantického facetového vyhledávače na platformě React s důrazem na jeho přepoužitelnost. Výsledkem je pak jeho implementace, která je také okomentovaná

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 ty existující 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 navržené řešení včetně vizualizačního modulu.
4. Ověřit funkčnost řešení srovnáním s existujícím facetovým vyhledávačem na stránkách `https://slovník.gov.cz/prohlížeč`.

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.

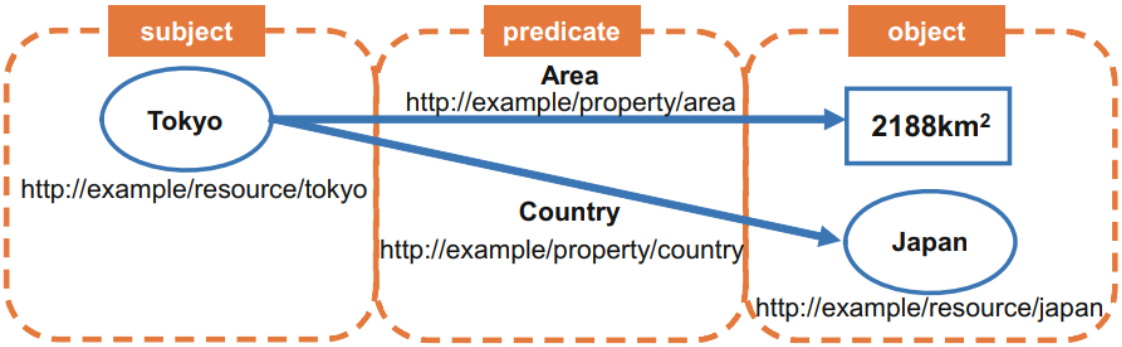
Sémantický web se tak má stát novým evolučním stupněm stávajícího webu (velmi často je nazýván také jako Web 3.0, ale fakticky je spíše jeho částí), kde jsou informace uloženy podle standardizovaných pravidel, což usnadňuje jejich vyhledávání a zpracování [2]. Ony standardizované pravidla jsou hlavně 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

V této sekci si popíšeme relevantní sémantické techonologie a formáty. Pochopení těchto technologií je nezbytné k návrhu a následné implementaci sémantického facetového vyhledáváče.

2.2.1 RDF

V sémantickém světě je standardem pro vytváření dat formát RDF [4]. RDF je standardizovaný strojově čitelný grafový formát, ve kterém se využívají tzv. triples, česky trojice, k popsání relací ve formátu subjekt - predikát - objekt. Tyto trojice lze znázornit jako orientované hrany v grafu.



Obrázek 2.1. Ukázka RDF trojice, kde je každá entita identifikovaná svou URI [5].

Jednotlivé uzly v grafu jsou pak identifikované pomocí IRI, což je nadmnožinou URI, která povoluje více Unicode znaků. IRI, potažmo URI, může pak vypadat například takto:

<http://www.w3.org/1999/02/22-rdf-syntax-ns#Class>

Samotná syntaxe RDF definovaná není, nejčastěji se však používá RDF/XML, která dokáže zapsat RDF graf jako XML dokument či Turtle, který je více podobný běžnému textu. Obě syntaxe jsou vytvořeny a doporučeny W3C. Pro účely dnešních aplikací je nutné také zmínit existenci formátu RDFJS, který reprezentuje RDF data v jazyku JavaScript [6].

2.2.2 OWL 2

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 se ontologií rozumí 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:[9]

- 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.

Později pak také Tim Berners-Lee definoval hodnocení kvality Linked Data od 1 do 5 hvězdiček. Pro splnění stupně hodnocení musí data splňovat i předchozí stupně. Toto hodnocení je definované takto:[9]

1. 1 hvězdička - Data jsou dostupné na webu (v jakýmkoliv formátu), ale s otevřenou licencí, jako Open Data.
2. 2 hvězdičky - Data jsou k dispozici jako strojově čitelná strukturovaná data (např. excel místo skenování obrázku tabulky)
3. 3 hvězdičky - Data jsou v nechráněnem formátu (např. CSV místo excelu).
4. 4 hvězdičky - Data používají otevřené standardy od W3C (RDF a SPARQL) k identifikaci věcí, aby mohli ostatní lidé na vaše data odkazovat.
5. 5 hvězdičky - Data jsou propojená s daty jiných lidí, aby poskytovaly kontext.

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

2.2.4 SPARQL

Primárním dotazovacím jazykem pro RDF je SPARQL [10]. 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

Obsahuje klauzule jako BIND k přiřazování hodnot k proměnným či OPTIONAL, který je podobný k nepovinnému JOIN z SQL. Proměnné jsou značeny prefixem „?“. Pro lepší čitelnost dotazu lze také využít klauzule „PREFIX“, která nahradí URI ontologie definovaným prefixem.

```
PREFIX ex: <http://example.com/exampleOntology#>
SELECT ?capital
       ?country
WHERE
  {
    ?x  ex:cityname       ?capital   ;
        ex:isCapitalOf    ?y         .
    ?y  ex:countryname    ?country   ;
        ex:isInContinent  ex:Africa  .
  }
```

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 [11]

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 [12]. 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]

[typy-facetu]
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, ale neovládá se posuvníkem, ale jsou nadefinovány a pojmenovány rozsahy, dle kterých se pak dají výsledky filtrovat.

3.2.5 Text facet

V některých knihovnách implementující facetové vyhledávání se také setkáme s „Text facetem“. Tím se rozumí běžné fulltextové vyhledávání. V našem prototypu a vzniklých knihovnách nebereme takové vyhledávání jako typ facetu.

3.3 Srovnání přístupů

Řešení jak implementovat facetové vyhledávání je více. Liší se hlavně podle toho, kde dochází k filtrování výsledků facety nebo jakým způsobem se získávají možnosti facetů. Například si můžeme vyžádat všechny možné výsledky, z těch sestavit možnosti facetů a poté podle aktivních facetů jen filtrovat stažená data. Toto je však u rozsáhlých výsledkových sad poměrně náročné na výkon, jelikož to všechnu zátěž převádí na klienta. Po rešerši současných řešení

3.3.1 Filtrování na straně klienta

Při filtrování na straně klienta nám server vždy zašle celou výsledkovou sadu, ze které zpracováním naplníme možnosti facetů. Po zvolení hodnot facetů pak můžeme jen filtrovat celou výsledkovou sadu a zobrazit jen výsledky, které vyhovují našim kritériím (zvoleným hodnotám facetů). Výhoda toho je, že sadu výsledků stačí teoreticky zaslat jen jednou a poté už s ní jen pracovat. 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 velmi rozsáhlá, což znamená velké množství dat, které musíme přenést internetem, uložit na klientovi a následně vyfiltrovat. To může při pomalejší rychlosti internetu nebo malém výpočetním výkonu na klientovi trvat velmi dlouho.

[1] Jejich HTML ovládací prvek má atribut disabled.

3.3.2 Filtrování na straně serveru

Filtrování na serveru většinu zátěži 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, je zvykem, že lze sémantická data dotazovat skrz publikovaný SPARQL endpoint. Jelikož můžeme na tento endpoint posílat požadavky i z klientské strany, nepotřebujeme vůbec server pro facetový vyhledávač. Z tohoto důvodu se zdá být vhodnější pro sémantická data 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.

[navrh]

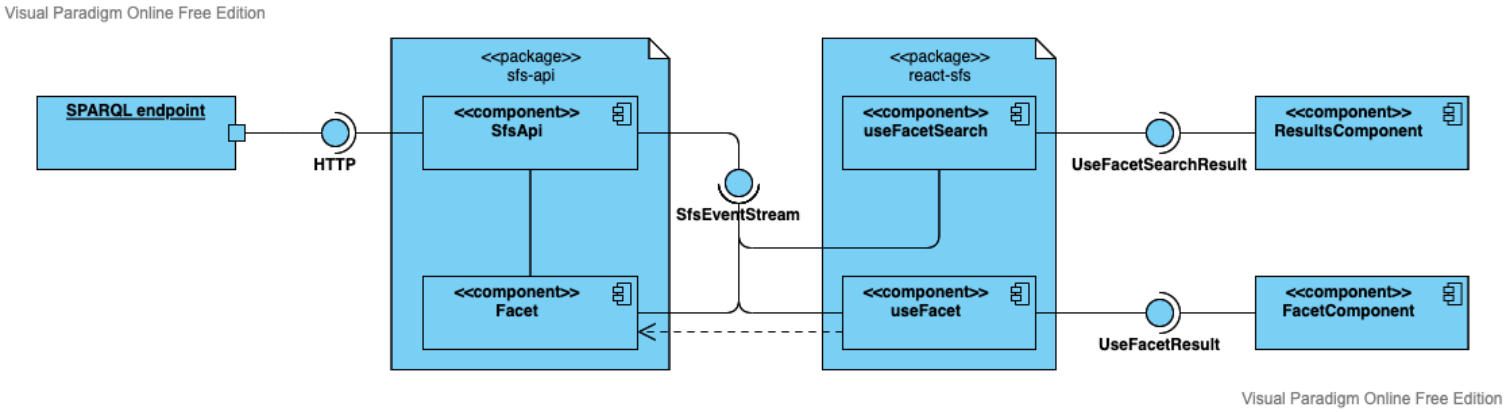
Kapitola 4
Návrh

Výsledkem této práce je sémantický facetový vyhledávač na platformě React. Tato kapitola se bude zabývat jeho návrhem. Nejprve navrhneme jak rozdělit moduly vyhledávače, poté popíšeme architekturu jednotlivých modulů a komunikaci mezi jednotlivými facety.

4.1 Moduly

Při návrhu je nutné dbát na to, aby byl vyhledávač rozdělený na moduly vyhledávání a jeho vizualizace. Toto rozdělení nám umožní, aby byl modul vyhledávání zcela nezávislý na použité platformě. Realizovat takový modul jde pak dvěmy způsoby - buďto jako samostatná serverová služba nebo jako modul bez využití konstruktů použité platformy, v našem případě React. Jelikož lze získávat sémantická data skrze SPARQL endpoint, který lze lehko volat přímo z prohlížeče, dává v našem případě větší smysl jít cestou modulu, který bude využíván přímo na frontendu.

Tento modul bude pojmenován *sfs-api* a bude koncipován tak, aby to bylo samostatná knihovna, kterou lze využít v jakémkoliv Javascript frameworku či prostředí. Jelikož však chceme implementovat vyhledávač primárně na platformě React, vytvoříme ještě jednu knihovnu jménem *react-sfs*, která bude usnadňovat použití *sfs-api* v React aplikacích. Výsledkem tak budou 3 projekty, knihovny *sfs-api* a *react-sfs* a finální projekt *sfs-react-demo*, který bude finálním sémantickým facetovým vyhledávačem a zároveň tak ukázkovým příkladem použití těchto knihoven. Knihovny *sfs-api* a *react-sfs* budou publikovány v repozitáři NPM[1], který je primárním repozitářem Node.js balíčků.



Obrázek 4.1. Diagram komponent sémantického facetového vyhledávače

Součástí publikovaných knihoven nejsou žádné komponenty či web elementy, které by přímo definovaly vizuální podobu samotných facetů či vyhledávání. Záměrně tuto

[1] https://www.npmjs.com

část necháváme na uživateli, abychom ho nijak neomezovali a nevnucovali mu vizuální podobu, která se mu nemusí líbit. Obě knihovny však poskytují rozhraní pro tyto web elementy, lze tedy využít i samostatně *sfs-api* pro jinou platformu než React.

4.2 Architektura

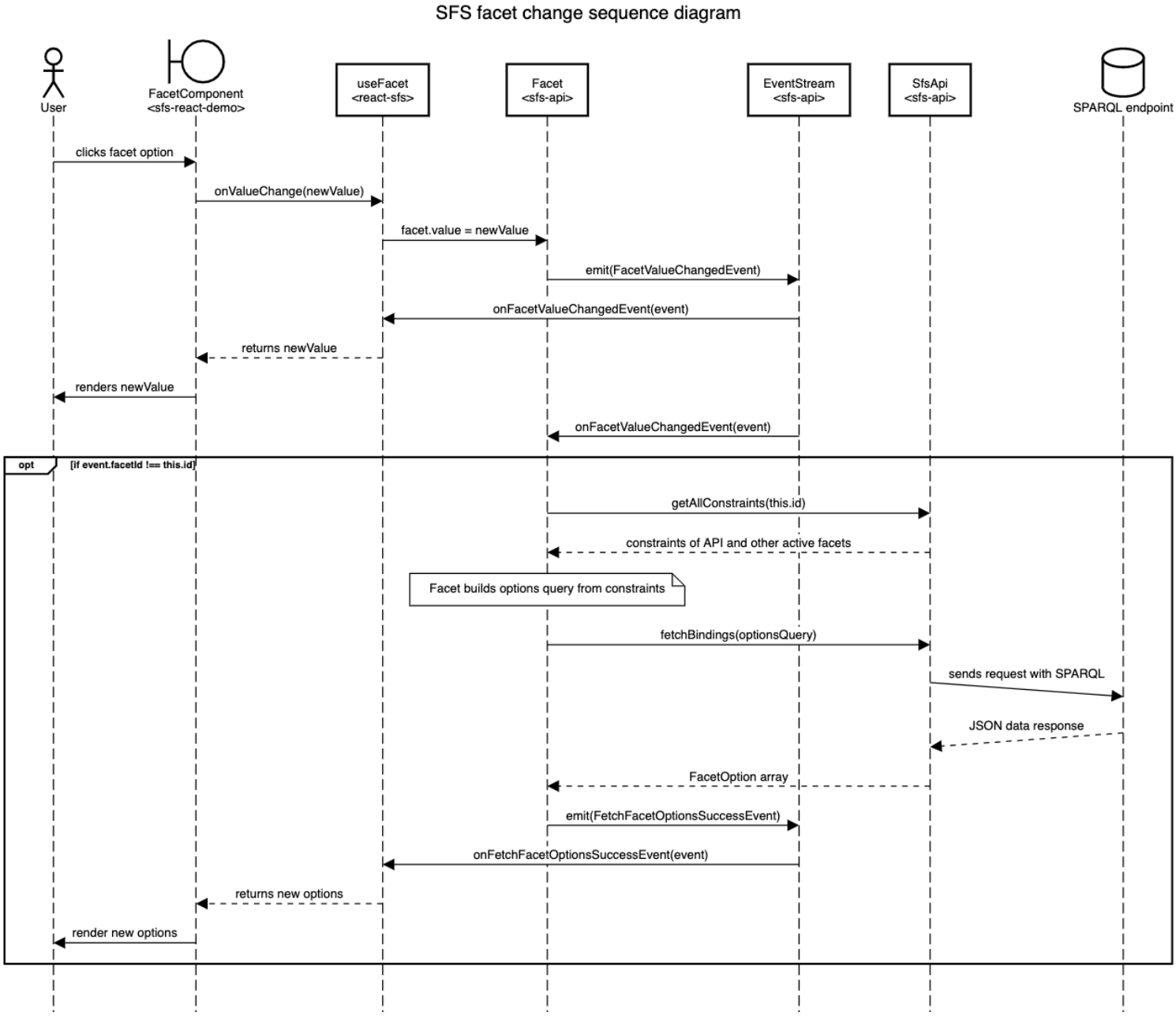
[constraints]
4.2.1 Constraints

Při návrhu bylo nutné vyřešit jakým způsobem omezíme možnosti facetů dle jiných aktivních. V návrhu je toto řešeno pomocí „contraints“. Constraints nazýváme v této práci (a i samotných knihovnách) SPARQL patterny, které omezí výsledky dle aktivních facetů. Tento nápad a termín byly inspirovány existující knihovnou *angular-semantic-faceted-search*[13], která je také implementací sémantického facetového vyhledávání.

Abstraktní třída `Facet` představuje stav facetu. V době psaní této práce nabízí knihovna *sfs-api* dva typy facetů (dle naší definice z 3.2) - checkbox facet a select facet. Uživatel může z této třídy dědit a naimplementovat si vlastní facet. Toho lze docílit implementováním abstraktních metod `buildOptionsQuery()`, která by měla vracet query k dotazování možností facetů a `getFacetConstraints()`, která by měla vracet constraints dle [constraints].

4.3 Popis eventů

Facety i API komunikují svůj stav mezi sebou pomocí eventů. Pomocí těchto eventů můžeme komunikovat stav i se samotným uživatelským prostředím. Jsou tedy klíčové k integraci knihovny do jakékoliv aplikace. Abychom si to lépe přiblížili, ukážeme si proces změny hodnoty facetu a reakci ostatních facetů na sekvenčním diagramu níže.



Obrázek 4.2. Sekvenční diagram změny hodnoty facetu

4.4 Konfigurace (tohle spíš do implementace asi)

```
export const sfsApi = new SfsApi({
  endpointUrl: "https://xn--slovnk-7va.gov.cz/sparql",
  facets: [glosaryFacet, subClassOfFacet],
  queryTemplate:
    `SELECT DISTINCT ?_id ?_label
WHERE
  { ?_id a <http://www.w3.org/2004/02/skos/core#Concept> .
    FILTER isIRI(?_id)
    OPTIONAL
```

```
      { ?_id rdfs:label ?rdfsLabel
        FILTER langMatches(lang(?rdfsLabel), "${language}")
      }
    OPTIONAL
      { ?_id skos:prefLabel ?prefLabel
        FILTER langMatches(lang(?prefLabel), "${language}")
      }
      BIND(coalesce(?rdfsLabel, ?prefLabel, ?_id) AS ?_label)
  }
ORDER BY ASC(?_label)`,
  prefixes: {
    rdfs: "http://www.w3.org/2000/01/rdf-schema#",
    skos: "http://www.w3.org/2004/02/skos/core#",
    dct: "http://purl.org/dc/terms/",
    dbp: "http://dbpedia.org/property/",
    dbo: "http://dbpedia.org/ontology/",
  },
  language,
});
```

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 *react-sfs*. 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
Implementace

V této kapitole si popíšeme postup implementace návrhu *sfs-api* z předešlé kapitoly 4. Zdůvodníme si některé rozhodnutí při vývoji a použité knihovny.

5.1 Popis implementace

Při implementaci jsme

5.2 Použité knihovny

Při vývoji jsme se snažili, aby knihovny měly co nejméně závislostí, aby byly nenáročné na místo a neinstalovali uživateli závislosti, které nepotřebuje nebo nechce. Zde si popíšeme využité knihovny. Všechny tři projekty jsou napsané v TypeScriptu, což je nadstavba JavaScriptu, která jej rozšiřuje o statické typování a předchází tak spoustě chyb.

5.2.1 sfs-api

Na balíček *sfs-api* jsme si vystačili pouze s třemi závislostmi. Využíváme knihovnu *fetch-sparql-endpoint*[1]. 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 poměrně aktuální.

Balíček datového modelu RDFJS je druhou závislostí. *sfs-api* využívá ještě knihovnu *sparqljs*[2] ke parsování SPARQL dotazů do JavaScript objektů. Skrze tyto objekty můžeme pak query lépe upravovat.

5.2.2 react-sfs

Knihovna *react-sfs* má jako závislost pouze *sfs-api*.

5.2.3 sfs-react-demo

Jako součást zadání této práce bylo rozhodnuto, že na implementaci vyhledávače bude použit JavaScript framework React. K vytvoření projektu vyhledávače, tedy *sfs-react-demo* jsme použili doporučenou metodu vytváření React projektů *create-react-app*. Ta vytvoří předpřipravený projekt s minimální konfigurací a nejužitečnějšími závislostmi.

Abychom se nemuseli zaobírat UI prvky prototypu, využili jsme knihovnu MUI (také známou také jako Material-UI) a použili připravené komponenty z ní. Druhá knihovna, kterou využíváme je fetch-sparql-endpoint [14]. Ta nám zjednodušuje volání SPARQL endpointu a serializaci příchozích dat do formátu RDFJS. Vybrali jsme

[1] https://www.npmjs.com/package/fetch-sparql-endpoint
[2] https://www.npmjs.com/package/sparqljs

ji hlavně proto, že je skutečně jednoduchá, neimplementuje nepotřebné věci navíc a je udržována aktuální.

Za zmínku pak ještě stojí knihovna *react-virtuoso*, kterou využíváme na virtualizaci listů s výsledky a možnostmi facetů. Virtualizace listu znamená, že se renderuje vždy jen část listu, která je zobrazená a ne ostatní položky, které se zrovna nezobrazují. List totiž může obsahovat velmi velké množství dat a renderování všech položek by mohlo znatelně zpomalit naší aplikaci.

5.3 Srovnání s existujícím vyhledávačem

Na adrese https://slovník.gov.cz/prohlížeč v nynější době běží facetový vyhledávač nad databází sémantických dat spravovanou Ministerstvem Vnitra České Republiky (MVČR) ve spolupráci s ČVUT. Tento stávající, a v některých ohledech nevyhovující, vyhledávač bychom chtěli nahradit nově vzniklým v rámci této práce. Nyní tedy, porovnáme tyto dva vyhledávače, čímž zároveň ukážeme funkčnost našeho nově vzniklého.

rychlost načítání, virtualizace, jazyk, nezávislost na vizuální podobě, možnost implementování vlastního facetu

Kapitola 6
evaluace

Kapitola 7
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 facetový vyhledávač a zanalyzovali způsoby jak jej implementovat. S těmito poznatky jsme navrhli a následně implementovali knihovny pro implementaci facetového vyhledávače a s jejich pomocí i vyhledávač samotný. Postupně jsme tak splnili všechny cíle této práce.

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 však 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.

Literatura

ic-american-article]

[1] Berners-Lee, Sir Tim, James Hendler a Ora Lasilla. The Semantic Web. *Scientific American*. Scientific American, a division of Nature America, Inc., 2001, ročník 2001, č. Vol. 284. No. 5, s. 34-43. Dostupné na DOI https://www.jstor.org/stable/10.2307/26059207.

cambridge-semantics]

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

mantic-web-security]

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

[w3c-rdf]

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

[rdf-community]

[5] *Understanding Linked Data Formats*. Dostupné na `https://rdf.community/understanding-linked-data-formats`.

[rdfjs-data-model]

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

[w3c-rdfs]

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

[w3c-owl]

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

[w3c-linked-data]

[9] *Linked Data*. Dostupné na `https://www.w3.org/DesignIssues/LinkedData.html`.

[w3c-sparql]

[10] *SPARQL 1.1 Query Language*. Dostupné na `https://www.w3.org/TR/sparql11-query`.

ets-explained-image]

[11] *What is faceted search*. Dostupné na `https://stackoverflow.com/questions/5321595/what-is-faceted-search`.

[nng-study]

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

ntic-faceted-search]

[13] *SPARQL Faceter - Client-Side Faceted Search Using SPARQL*. Dostupné na `http://semanticcomputing.github.io/angular-semantic-faceted-search`.

tch-sparql-endpoint]

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

Příloha A
Slovníček

API	Application programming interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
ICT	Informační a komunikační technologie
IRI	Internationalized Resource Identifier
OWL	Web Ontology Language
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
SPARQL	SPARQL Protocol and RDF Query Language
SQL	Structured Query Language
UI	uživatelské rozhraní
URI	Uniform Resource Identifier
WWW	World Wide Web
W3C	World Wide Web Consortium
XML	Extensible Markup Language

Requests for correction

[rfc-1] předělat poděkování

[rfc-2] předělat abstrakty

[rfc-3] přendat do jine kapitoly asi

[rfc-4] se docela opakuju co

[rfc-5] možná dát pryč

[rfc-6] revisit

[rfc-7] pridat listing label

[rfc-8] je ok začít obrázkem?

[rfc-9] možná někde vysvětlit co to je

[rfc-10] vysvětlit nějak klientskou stranu vs server

[rfc-11] dodelat

[rfc-12] možná o ní budu psát už někde předtím

[rfc-13] pridat listing label

[rfc-14] přendat do jine kapitoly asi

[rfc-15] TODO

[rfc-16] přendat do jine kapitoly asi