Jihočeská univerzita v Českých Budějovicích

Přírodovědecká Fakulta

Bakalářská práce:

Technologie faceted search a její praktické využití pro rozhraní agregátoru služeb

Vypracoval: Tomáš Honner

Školitel: Ing. Martin Čížek

České Budějovice 2014

Bibliografické údaje:

Honner T., 2014 : Technologie faceted search a její praktické využití pro rozhraní agregátoru služeb [The Faceted search technology and its practical use for aggregator of services interface] – 42 p., Faculty of Science, University of South Bohemia, České Budějovice, Czech Republic.

Anotace

Práce se zabývá rozborem vlastností a možností uplatněním technologie faceted search a jejího využití v aplikacích typu business intelligence (BI), popisem state-of-the-art implementace v podobě technologie Elastic Search a obsahuje praktický proof-of-concept s prezentací veřejně dostupných dat. Jako proof-of-concept bude sloužit aplikace založená na webových technologiích geolokace, NodeJS, Elastic Search a Google Maps API. Proof-of-concept prochází celý cyklus vývoje od rešerše a návrhu přes vytvoření modelu, use-case diagramů až po implementaci, testování a závěrečné vyhodnocení.

Klíčová slova

Faceted search, Elastic Search, NodeJS, geolokace, agregátor

Annotation

The bachelor‘s thesis deals with the analysis of properties and options of possible aplication facested search technology and its use in applications such as business intelligence (BI), state-of-the-art implementation description in the form of technology Elastic Search and contain practical proof-of-concept with public available data presentation. As a evidence of the proof-of-concept will serve application based on web technologies geolocation, NodeJS, Elastic Search and Google Maps API. The proof-of-concept passing the entire development cycle from the development of research, creating a patterns, use case diagrams until implementation, testing and final evaluation.

Key words

Faceted search, Elastic Search, NodeJS, geolocation, aggregator

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne 24.4.2014

……

Tomáš Honner

Poděkování

Rád bych poděkoval panu Ing. Martinu Čížkovi za vedení mé bakalářské práce a pomoc při jejím psaní.

Dále členům Ústavu aplikované informatiky za předané znalosti a odborné konzultace.

Panu Lukáši Ruczkowskému za vysvětlení základních principů fungování technologie NodeJS.

Rodičům Pavlovi a Marii Honnerovým za psychickou a morální podporu při studiu.

Obsah

1 Úvod

V současné době je silně na vzestupu technologie faceted search. Jedná se o rychlé vyhledávání na základě indexování dokumentů tvořených tzv. facety, což jsou aspekty vyhledávání. Tato technologie se hojně používá u databází s obrovským objemem dat, protože přináší podstatné zrychlení vyhledávání. Pro své internetové aplikace jí využívají takové korporace jako Amazon, Ebay nebo Twiter a GitHub. V internetových obchodech slouží pro vyhledávání zboží určitých aspektů jako např. „zboží této barvy, této značky, těchto rozměrů“, oproti tomu na Twitteru složí pro vyhledávání mezi stovkami miliard tweetů. Faceted search využívají i tuzemské firmy ve svých e-shopech jako Alza.cz nebo CZC.

Tato technologie má velké využití i v Business Intelligence (BI). Její součástí je i sběr dat od uživatelů a jejich analýza. Data jako nejprohlíženější článek, nebo konkrétní zboží, které si uživatel prohlížel nejdéle.

Významné je i použití faceted search v aplikacích s geolokačními prvky. Například aplikace Foursquare, která na základě geolokace zjistí vaší polohu a pak s použitím faceted search vyhledá restauraci či jiný podnik podle vašich požadavků. Nebo česká aplikace Český Benzín sloužící k nalezení nejbližší čerpací stanice se zadaným druhem benzínu či nafty.

Tato bakalářská práce se zabývá právě faceted search, jeho popisem, popisem základních technologií používaných pro faceted search a proof-of-concept aplikací, kde budou tyto technologie naimplementovány.

2 Cíle práce

Cílem práce je popsat technologii faceted search, její využití v BI a její praktické aplikace při prezentaci dat. Popsat state-of-art implementaci v podobě Elasticsearch. K doložení vyrobit proof-of-concept aplikaci. Aplikace bude agregátor služeb s podporou geografického vývěru.

Při zpracování tématu se zaměřte zejména na řešení následujících dílčích problémů.

1. Proveďte analýzu požadovaných funkčností, navrhněte konceptuální datový model, drátěné modely uživatelského rozhraní a use cases.

2. Navrhněte architekturu aplikace a konzultujte se školitelem. Pro kriteriální výběry zvažte techniky faceted search, pro prezentaci na mapě použijte Google maps. V oblasti faceted search Vám školitel navrhne konkrétnější řešení. Databázový stroj nemusí být nutně relační a dotazovacím jazykem nemusí být nutně SQL. Možná řešení jsou a) relační/SQL databáze, b) relační/SQL databáze + nerelační noSQL index nebo c) pouze nerelační noSQL. Volbu popište v doprovodné práci. Pro relační databáze preferujte stroj PostgreSQL, pro indexování pak řešení postavená nad Lucene (Elastic Search, Solr).

3. Návrh, použité návrhové vzory, technologická rozhodnutí včetně jejich zdůvodnění a projekt samotný zdokumentujte ve své práci

3 Základní pojmy

3.1 Agregátor

Je software, který shromažďuje jeden typ informace z více zdrojů. Funguje na manuální či automatické bázi a může koncovému uživateli předkládat jak konečný obsah, tak i výsledky hledání. [1]

3.2 ETL (extract, transform, load)

Je proces získávání dat z více zdrojů. Zahrnuje extrakci, transformaci a nahrání. Používá se například při vytvoření jednoho jednotného („korporátního“) systému z více („podnikových“) systémů například při tvorbě datových skladů.

1. Extrakce
   - extrakce dat z vnějších zdrojů, většinou nejobtížnější část ETL
   - většinou jde o periodickou činnost
   - musí se zohlednit nejenom data samotná, ale i jejich struktura, mechanizmy funkce nad daty prováděné
   - v další fázi zahrnuje parsing extrahovaných dat a kontrolu výsledku, zda má odpovídající strukturu.
2. Transformace
   - aplikuje se při ní série funkcí a pravidel na extrahovaná data, tak aby struktura dat odpovídala struktuře cíle, do kterého jsou data poté nahrána (pokud data nemají žádoucí strukturu už po extrakci a je nutné je upravovat
   - využívá metody jako filtrování, normalizace, konverze nebo i tvoření multidimenzionálních struktur či matematické operace.
   - může obsahovat i kontrolu kvality dodávaných dat.
3. Nahrání (load)
   - fáze nahrání do cílového systému
   - záleží na požadavcích a periodicitě jakou proces ETL má (každou minutu, hodinu, den, měsíc), jestli se předchozí data smažou a nahradí se nově vytvořenými nebo se data průběžně updatují [2]

Při tvorbě ETL lze postupovat dvěma způsoby. Buď si ETL navrhnout a vytvořit sám, nebo použít dedikovaný nástroj pro tvorbu ETL a jeho implementaci. Takovým nástrojem může být Talend nebo SSIS pro MS SQL Server.

3.3 Geolokace

Je metoda, která pomocí různých technik zjišťuje geografickou polohu objektu. Jejím výstupem je zpravidla adresa, ne zeměpisné souřadnice. [3]

Dvě hlavní metody pro zjišťování zeměpisné polohy u elektronických zařízení (počítače, mobily) jsou:

1. GPS – pouze u mobilních zařízení pokud GPS chip mají. Jde o nejpřesnější a nejspolehlivější metodu. Pokud uživatel zařízení povolí, tak zařízení zašle přesné informace o své aktuální poloze aplikaci.
2. IP adresa – zeměpisné souřadnice se získávají z databází IP adres providerů. Oproti GPS nepřesné. Často je jako zeměpisné poloha uživatele uváděna adresa serveru, ze kterého se připojuje, nebo adresa poskytovatele internetového připojení.

3.4 Faceted classification system

Systém dovolující přiřazení objektu více charakteristik (sad atributů), umožňující třídění několika způsoby, spíše než v jednom předem stanoveném pořadí. [4]

3.4.1 Vlastnosti faceted classification system

- Faceted classification system umožňuje přidání objektu více možností řazení a umožňuje těmto řazením být použity vyhledávači více směry než jedním předem určeným způsobem.
- Faceted classification system bere v úvahu kombinaci aspektů vyhledávacího filtru a rychleji vytváří sadu řazených objektů.
- Faceted classification system se zaměřuje na důležité, základní nebo trvalé charakteristiky obsahu objektů, čímž napomáhá při kategorizaci rychle se měnících úložišť.

- U faceted classification system nemusíme znát název kategorie ve které objekt primárně umístěn.
- Nové aspekty hledání mohou být vytvořeny bez narušení jakékoliv hierarchie nebo reorganizace ostatních aspektů.

Výhody:

- Systém má schopnost začlenit širokou škálu informací a katalogizovat je.

Nevýhody:

- Složitost, robustnost.

3.5 Faceted search

Faceted Search vyvinuto Association for Computing Machinery's Special Interest Group on Information Retrieval. Založeno na faceted classification system.

Je technika přístupu k informacím organizovaným podle faceted classification system, umožňující uživatelům prohlížet kolekce informací s použitím mnohonásobných filtrů. Aspekty vyhledávání odpovídají vlastnostem hledané informace, často jsou odvozeny analýzou textu za pomoci extrakcí entit nebo předdefinovaných položek v databázi (např. autor, popis, jazyk atd.), proto o faceted search můžou být rozšířeny i stávající aplikace nebo webové stránky. [5]

3.5.1 Příklad faceted search

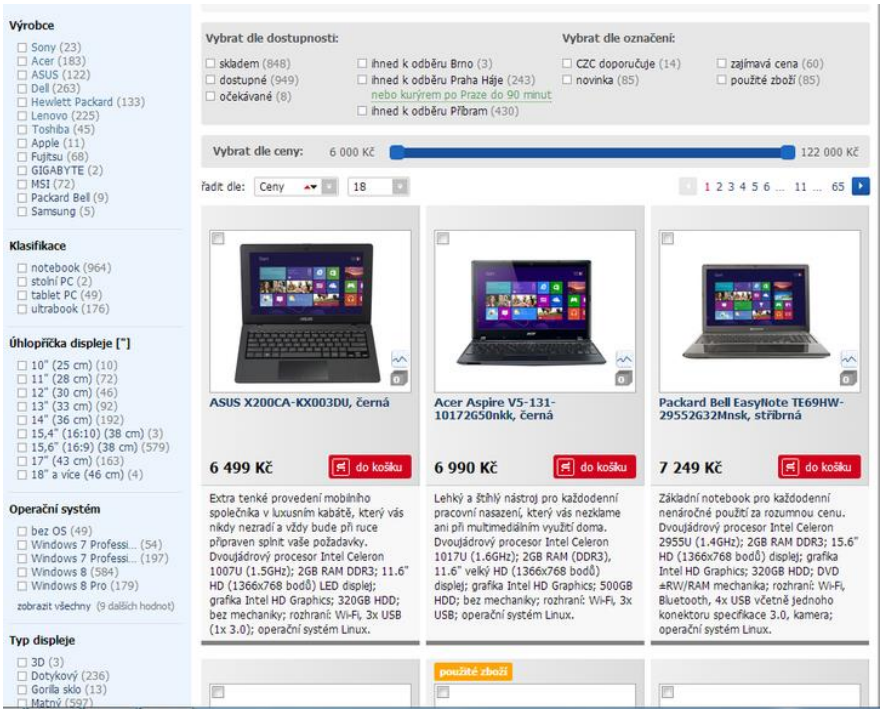
Klasický příklad faceted search je jakákoli aplikace nabízející víceúrovňové filtrování hledaných výsledků např. moderní e-shop, který při výběru zboží nabízí volbu výrobce, specifikace (u počítače např. procesor, velikost disku, druh grafické karty atd.), cenové rozmezí. Po aplikaci na práci může faceted search filtrace výsledků hledání vypadat následovně. Benzínová pumpa nabízející benzín s oktanovým číslem 98, v jihočeském kraji, u pumpy je myčka, pumpa má 24 hodinový provoz.

4 Přehled současného stavu řešené problematiky

Faceted search využívají velké společnosti s obrovskými databázemi například internetové obchody Amazon či ebay nebo Twitter a jiné internetové služby jako Foursquare a GitHub. Využívá jak pro upřesnění parametrů hledaného zboží, tak pro vyhledání nejbližší restaurace nebo knihovny na GitHubu.

4.1 Faceted search na Amazonu, ebay a jiných internetových obchodech

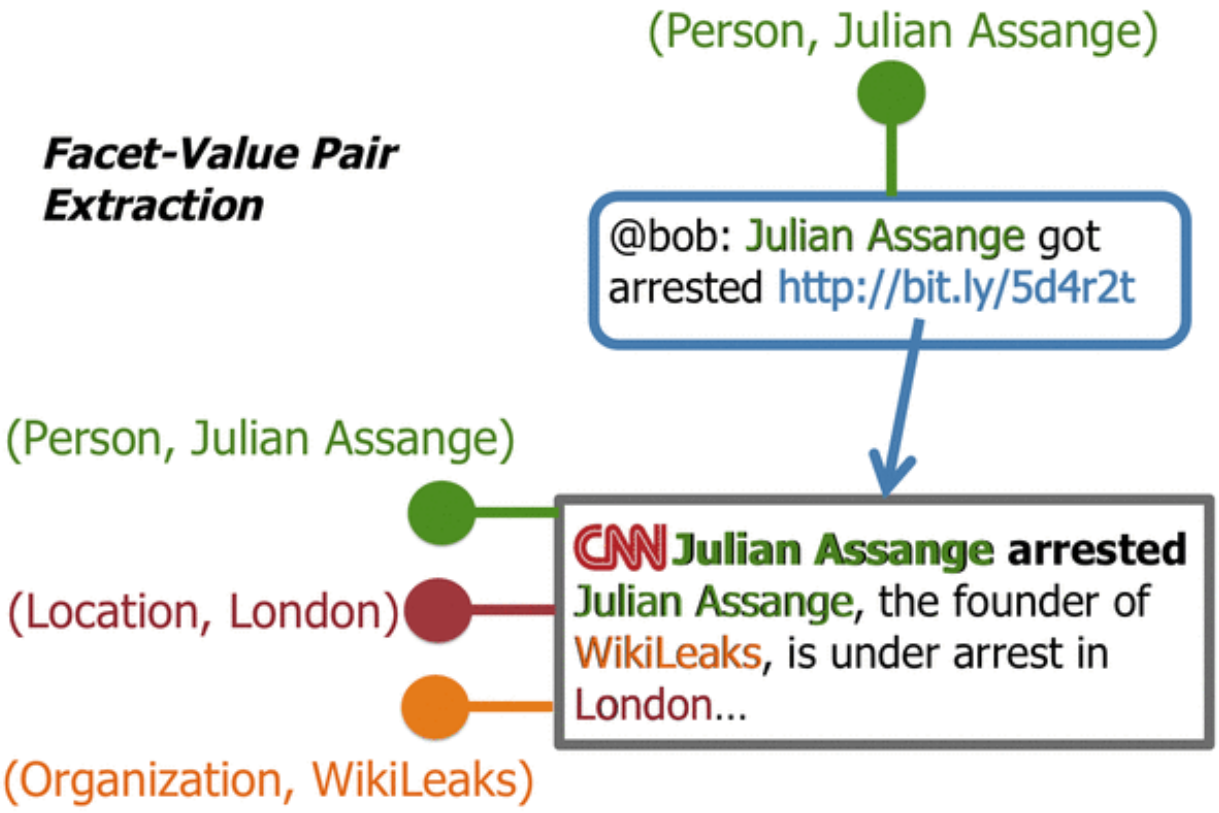
Přínos faceted search pro internetové obchody není jenom v tom, že je rychlý nad velkými databázemi atd. Jeho přínos je i ve sběru business intelligence (viz. obrázek č.1), ale napomáhá i při výběru zboží zákazníkům obchodu, kteří ještě nejsou rozhodnutí pro konkrétní typ. Na obrázku níže, který slouží jako příklad implementace faceted search implementovaného v internetovém obchodě Amazon.com. Zde jsou vidět facety jako Výrobce, Klasifikace, Operační systém atd.



Obrázek 1 Příklad facet na internetovém obchodě CZC.cz [6]

4.2 Faceted search na Twitteru

Další velká společnost používající faceted search je Twitter. Zde takzvané facety (aspekty vyhledávání) popisují vlastnosti zprávy nahrané na twitter. Viz. Obrázek

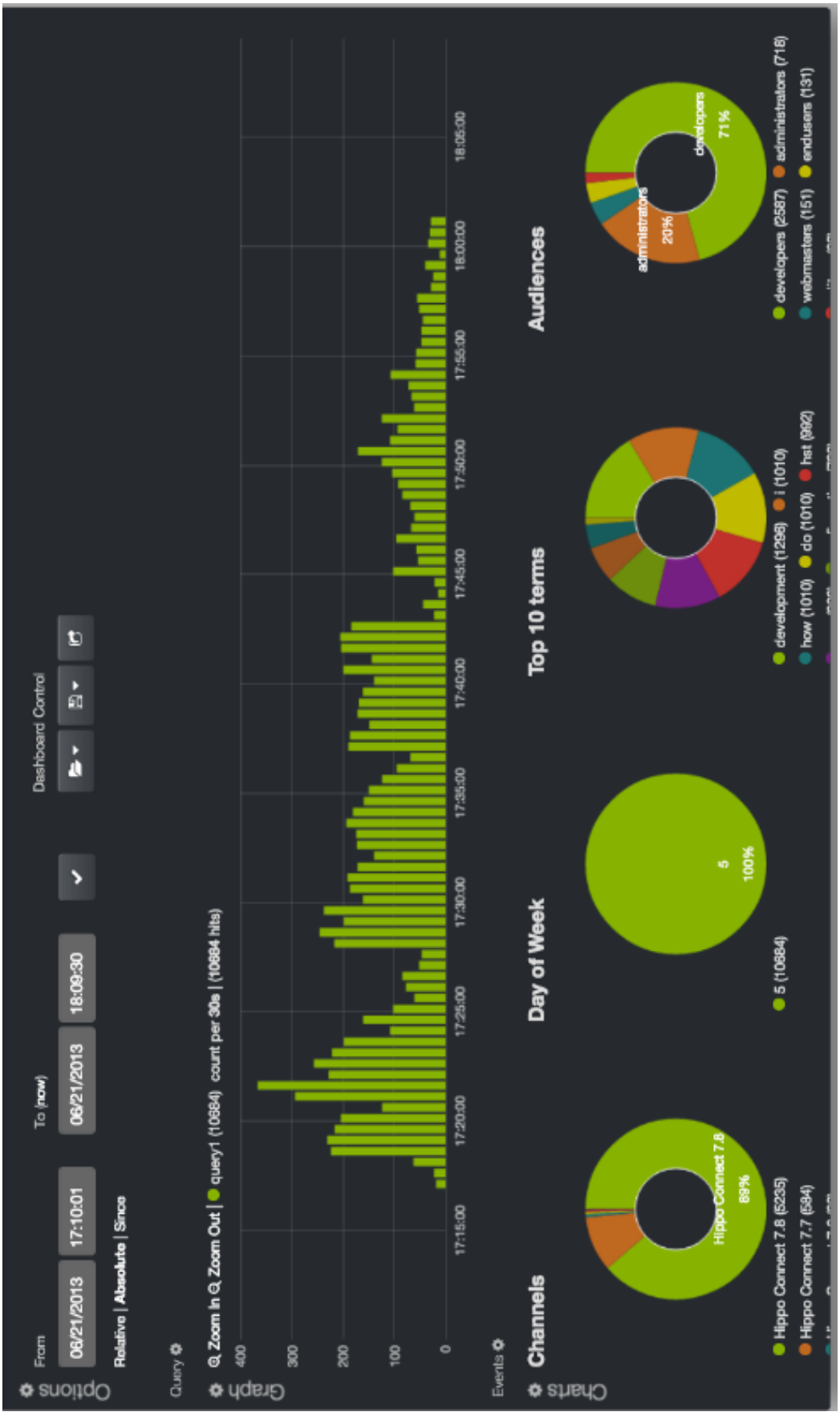


Obrázek 2 Tvoření facet na Twitteru [7]

4.3 Faceted search jako Business Intelligence (BI)

Faceted search se dá snadno využít i jako business intelligence. Většina jeho implementací jako vedlejší produkt své činnosti sbírá a ukládá statistická data. Jako příklad můžeme použít výše uvedený twitter. Zde lze snadno z takto nasbíraných dat tvořit analýzy a grafy.

Například, co bylo na sítí twitter nejvíce hledáno za poslední dvě hodiny, nebo jaký je nejvíce čtený tweet, nebo kolik uživatelů celkem daný tweet vidělo. Na obrázku číslo 3 je vidět vizualizace faceted search s BI v programu Kibana pro Elasticsearch.



Obrázek 3 Ukázka BI v programu Kibana [8]

5 Proof-of-concept a jeho cíl

Cílem proof-of-conceptu je vytvořit aplikaci, která bude mít implementovaný faceted search.

5.1 Parametry proof-of-concept

Jako proof-of-concept byla po konzultaci s vedoucím bakalářské práce vybrána aplikace s těmito parametry:

1. Proof-of-concept bude mít naimplementovaný faceted search.
2. Proof-of-concept bude jako implementaci faceted search používat Elasticsearch.
3. Business logika proof-of-concept bude implementovaná v technologii NodeJS.
4. Front-end proof-of-concept bude ve webovém prohlížeči.
5. Data se budou zobrazovat na mapě, kterou jako službu poskytuje Google

5.2 Upřesnění proof-of-concept

Jako proof-of-concept bude sloužit aplikace s vyhledáváním čerpacích stanic zohledňujícím geografické údaje o poloze uživatele aplikace a čerpacích stanic. Jako aspekty vyhledávání (faceted search) budou sloužit kraje, vzdálenost od uživatele na základě geolokačních údajů, nebo nabízené druhy benzínu a jejich cena.

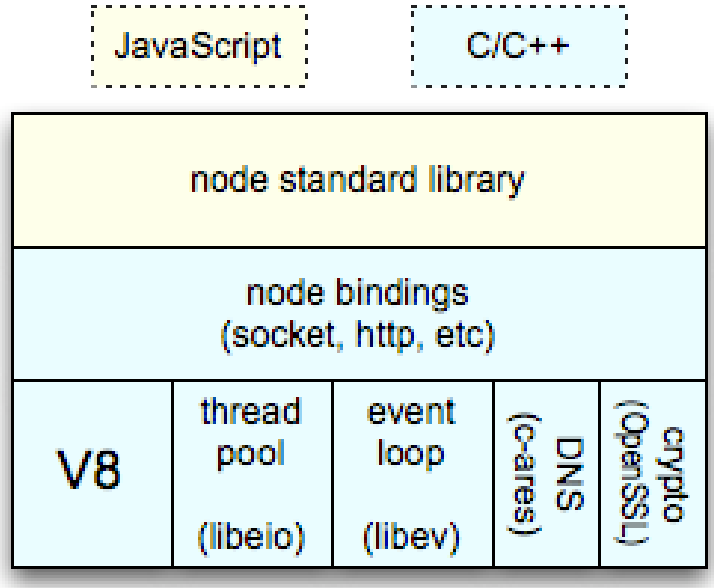
Jako vstupní data budou pro aplikaci sloužit data z veřejného projektu, kam data o benzínových stanicích a cenách benzínu ukládají jeho uživatelé.

6 Přehled součástek, technologií a standardů pro řešení

6.1 NodeJs

Je serverová platforma postavená na javascriptovém interpreteru V8 googlovského prohlížeče Chrome s vrstvou C++ nad ním, sloužící pro vývoj rychlých a škálovatelných webových aplikací.

Jak funguje. Na rozdíl od ostatních podobných technologií není založen na vláknech ale na událostech. Takzvaný event-loop nabízí oproti přístupu ve vláknech škálovatelnost až milionů sobě konkurujících operací. Navíc NodeJS používá asynchronní přístup, takže nečeká na žádné I/O operace, ale pokračuje ve vyřizování dalších událostí. [9]



Obrázek 4 NodeJS Architektura [10]

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
}).listen(1337, '127.0.0.1');
console.log('Server running at http://127.0.0.1:1337/');
```

Obrázek 5 Jednoduchý http server v NodeJS [11]

6.1.1 Express Framework

Serverový framework vyvinutý pro snadnější programování v NodeJS. Nabízí rychlou a jednoduchou tvorbu serverových aplikací se zaměřením na RESTové rozhraní.

6.2 Implementace Faceted search

Existuje několik implementací technologie faceted search. Často jsou tyto implementace postaveny na technologii Apache Lucane. Dvě nejpoužívanější z nich jsou Elasticsearch a Apache Solr. Obě jsou multiplatformní, obě jsou implementovány v Javě, obě pracují na principu, že usnadňují práci s knihovnou Lucane skrze uživatelsky přátelské API a přidávají k ní další funkcionality. Tím, že jsou obě postaveny na Apache Lucane, mají hodně společných vlastností a liší se od sebe pouze v přidaných funkcionalitách.

6.2.1 Dalšími technologiemi pro implementaci faceted search

Sphinx je Open-Source multiplatformní fultextový vyhledávací server implementovaný v technologii C++. Vytváří index nad SQL databázemi nebo NoSQL úložišti. Dotazování je prováděno skrze dotazovací jazyk SphinxQL. Na běžném desktopovém počítači s dvěma jádry dokáže Sphinx provést více jak 500 dotazů za sekundu a indexovat dokumenty rychlostí 60MB za sekundu. [12]

Xapian je Open-Source multiplatformní vyhledávací knihovna implementovaná v C++. Je vysoce adaptabilní nástroj, umožňující vývojářům snadné použití pokročilého indexování a

použití vyhledávacích nástrojů v aplikacích. Také podporuje tzv. „Probabilistic Information Retrieval model“[1] a také podporuje rozšířenou sadu logických operátorů při dotazování.

6.3 Elasticsearch

Open source nástroj na real-time vyhledávání a analytiku. Elasticsearch vznikl v roce 2010 a je postavený na Apache Lucane, knihovně implementované v Javě sloužící pro vyhledávání ve fulltextu. Je licencován po Apache Licence.

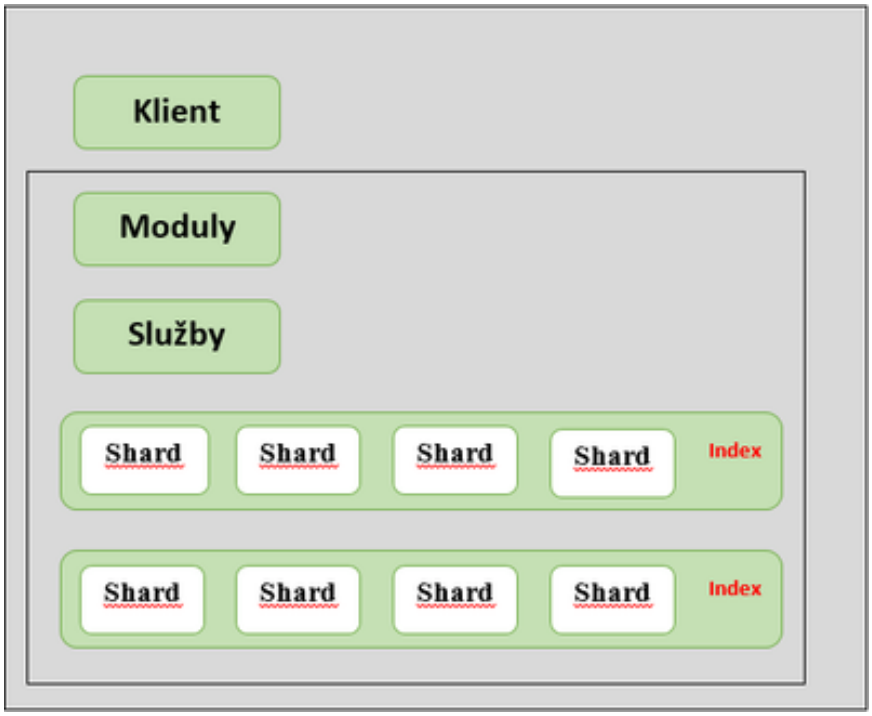
Vlastnosti:

- Snadno škálovatelný.
- Vysoce výkonný.
- Multiplatformní.
- Schema free
- Document based.
- Jako rozhraní používá webové služby typu REST a komunikace se předává ve formátu JSON

Elasticsearch může běžet jako cluster pro několik strojů, které se nazývají uzly a jednat jako jedna entita. Data jsou rozdělena do několika bloků nazvaných „shards“, které jsou automaticky dány nad uzly v clusteru. Každý shard může mít v sobě žádnou, jednu nebo více informací, čímž se zvýší výkon procházení (viz. Obrázek5). Informace jsou po clusteru rozděleny tak, že když dojde k poruše či vypnutí uzlu, jsou informace stále dostupné z primárního shardu. [13]

Elasticsearch je také velice snadné používat. Nepotřebuje žádnou konfiguraci. Stačí ho stáhnout a přes příkazový řádek spustit. Primárně se vyvíjí pro Linux, ale už existuje i verze pro Windows.

[1] Model z oboru pravděpodobnosti. Nastavuje u slov jejich váhu. Důležitější slova mají vyšší váhu, méně důležitá mají nižší váhu.



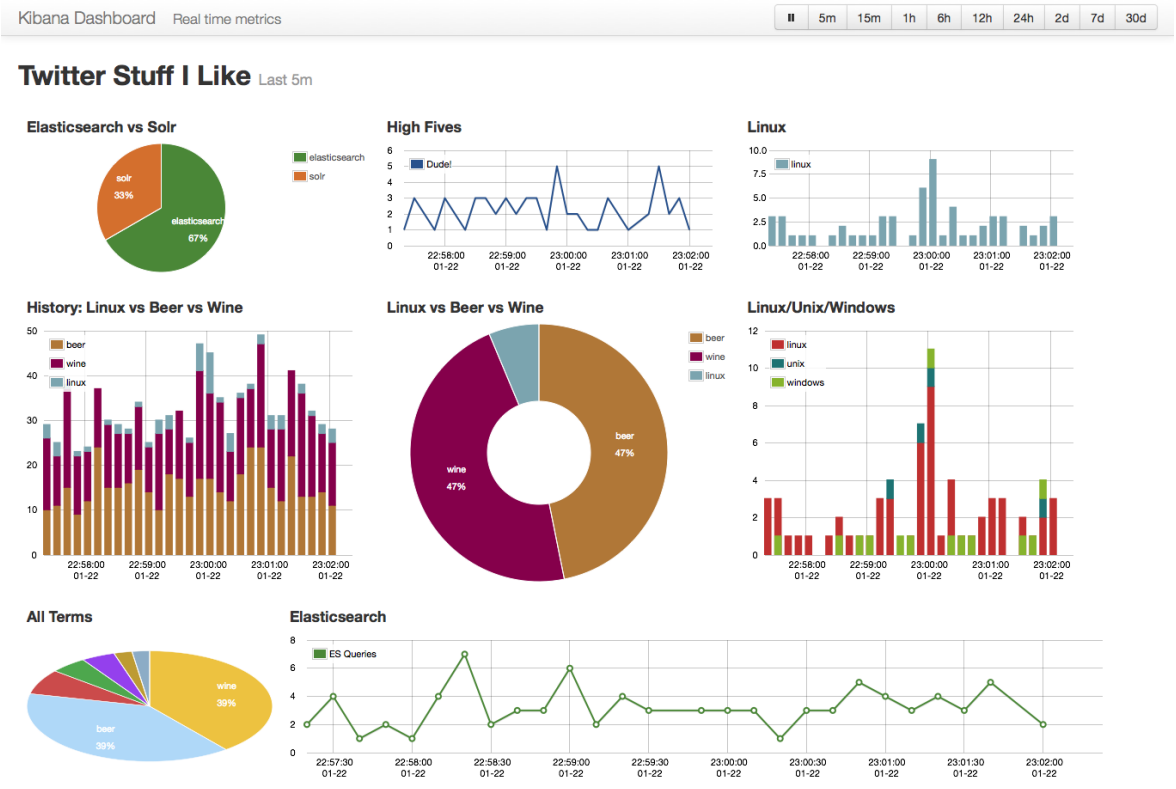
Obrázek 6 Architektura Elasticsearch

```
client.search({
  index: 'myindex',
  body: {
    query: {
      match: {
        title: 'test'
      }
    },
    facets: {
      tags: {
        terms: {
          field: 'tags'
        }
      }
    }
  }
}, function (error, response) {
  // ...
});
```

Obrázek 7 Ukázka dotazu ve formátu JSON pro Elasticsearch [8]

6.3.1 Elasticsearch jako BI (Business intelligence)

Elasticsearch při vyhledávání dokumentů sbírá různá statistická data. Toho jde snadno využít při implementaci business intelligence. Významným projektem pro BI nad elasticsearchem je projekt Kibana. Jde o aplikaci, která po nahrání na webový server zobrazuje ve webovém prohlížeči. Uživatel má tak okamžitý přístup k datům pro BI. Kibana zobrazuje například grafy s porovnáním počtu dotazů na určité facety v časové ose, nebo tvoří grafy z logů databáze ES. Pokud použijeme jako příklad internetový obchod CZC.cz uvedený víše (viz. obrázek č.1), tak bychom mohli s pomocí Kibany například generovat grafy dotazů na určité výrobce během posledního měsíce nebo grafy dotazů na konkrétní zboží během dne, nebo jaký dotaz během dne byl nejvíce používán. Další možnosti užití Kibany nad daty ze sítě Twitter (viz. obrázek č.8).



Obrázek 8 Ukázka projektu Kibana na datech ze sítě Twitter [14]

6.4 Bootstrap

Je volně distribuovaný frond-end framework pro webové aplikace sloužící k tvoření grafického rozhraní a layoutů nezávislých na velikosti zobrazovacího zařízení. Bootstrap v sobě kombinuje tři technologie a to HTML5, Javascript a CSS3.

Boostrap vyvinuli Mark Otto a Jacob Thorton ve firmě Twitter Inc. pro interní použití. V roce 2011 ho ale twitter vydal pod open-source licencí.

Proč právě Bootstrap:

1) Umožnuje i graficky nenadaným jedincům vytvářet solidní grafické návrhy.
2) Jednou z jeho hlavních vlastností je automatická úprava layoutu aplikace podle velikosti displeje zobrazovacího zařízení.
3) Jeho dokumentace a popis prostředí je na profesionální úrovni, což umožňuje rychlejší vývoj aplikace

6.5 Jquery

Je javascriptovská knihovna usnadňující používání javascriptu na straně klienta. Knihovna je open-source s licencí MIT. JQuery nabízí snadnější práci s DOM objekty, zpracovávání událostí, tvorbu animací a vývoj aplikací používajících AJAX. [15]

Právě pro tyto vlastnosti je použit v práci na zajištění komunikace se serverem skrze AJAX a pro snadnější manipulaci s objekty DOM. Taky je zde nutný jako prerekvizita pro některé knihovny (např. Gmaps.js).

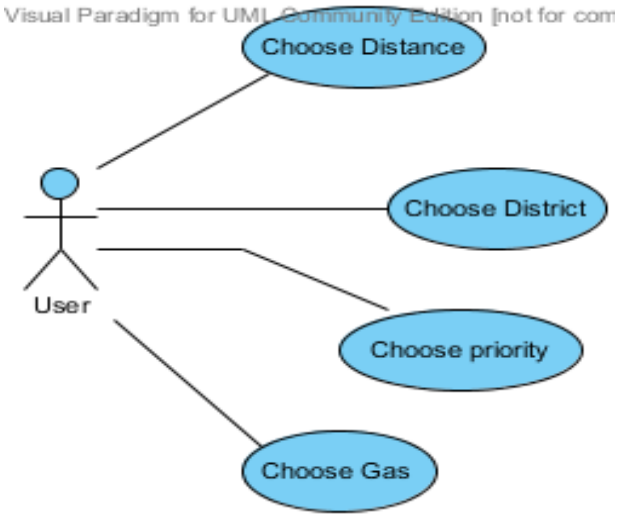
6.6 Gmaps.js

Je volně distribuovaná knihovna sloužící k jednoduššímu použití Google Maps API. [16] Knihovna například usnadňuje tvoření markerů[2] na mapách a to včetně vytváření jejich kolekcí. Dále podporuje vytváření tras či jednoduché grafiky ve formě polygonů. Jsou zde přidány funkcionality ve formě spolupráce s geolokací, vytváření kontextových menu, grafické úpravy mapy a jejich vlastností, nebo přidávání vlastních ovládacích prvků přímo do mapy. Jeden z hlavních důvodů proč byla knihovna vybrána pro použití v práci je přímá podpora formátu JSON pro předávání parametrů pro mapu od serveru nebo jiných aplikací.

[2] Marker je grafické znázornění významného bodu na mapě. Defaultně má tvar kruhové výseče vyplněné červenou barvou.

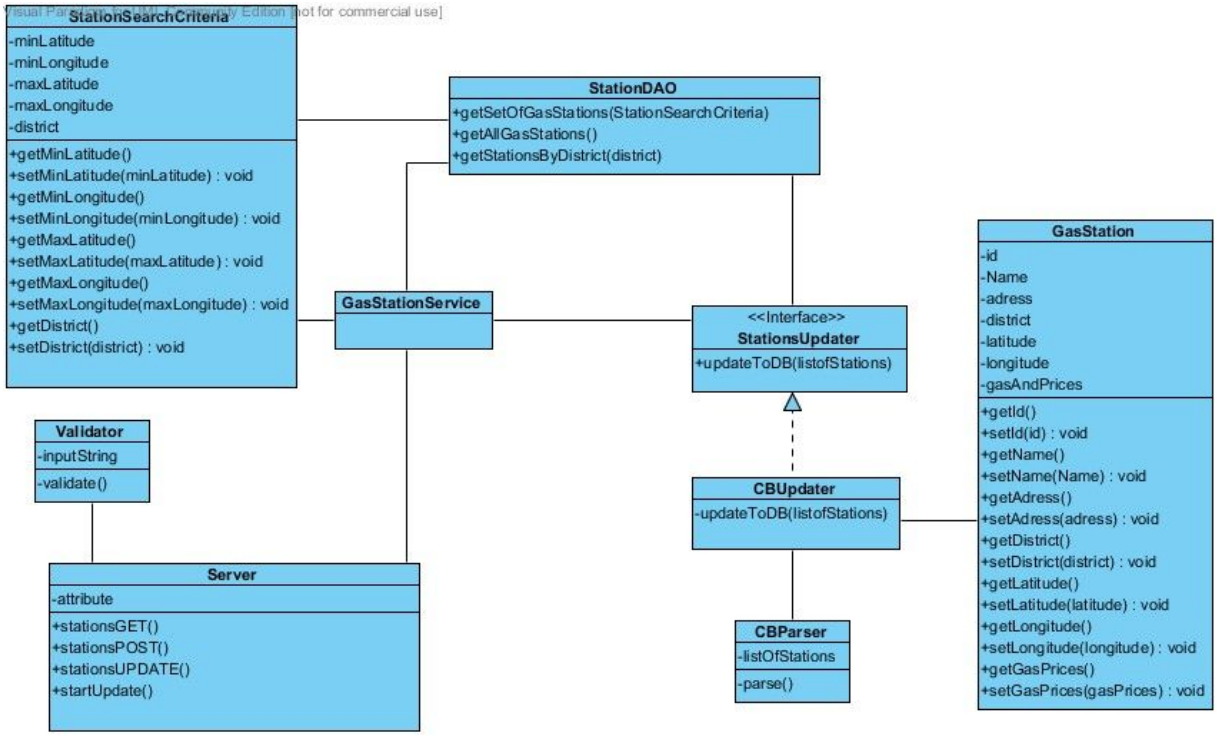
7 Proof-of-concept

7.1 Use Case diagram



Obrázek 9 Use Case diagram

7.2 Class diagram

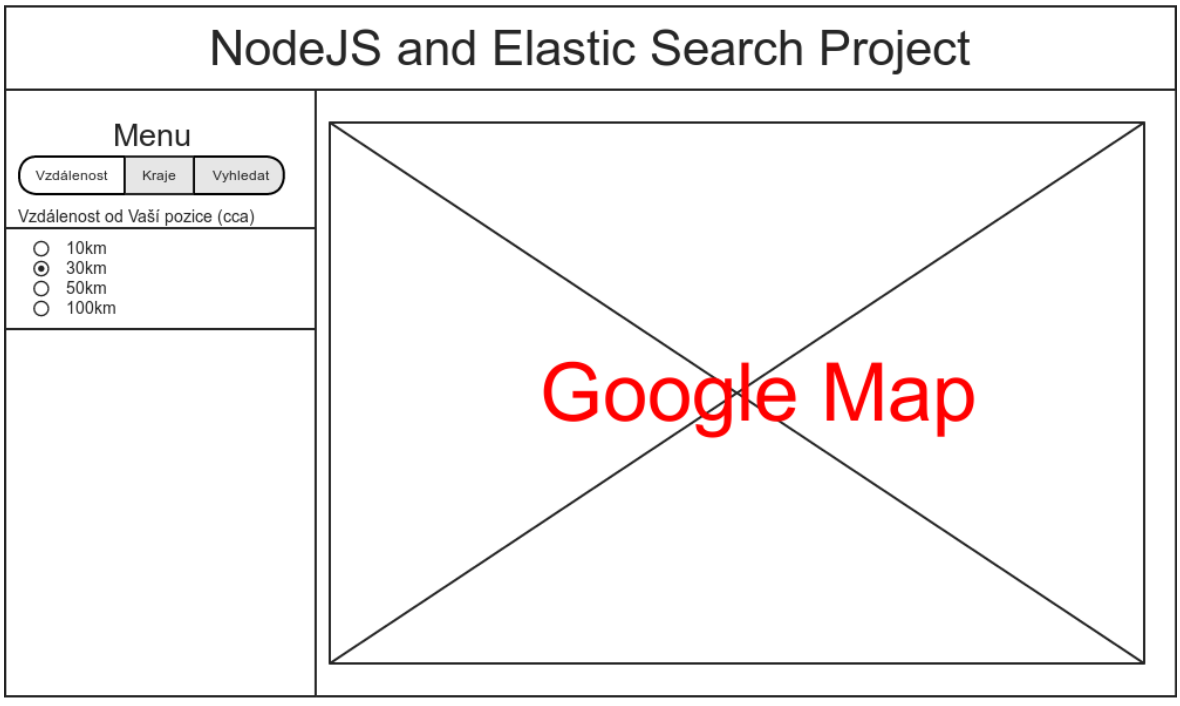


Obrázek 10 Class diagram

7.3 Model

Obrázek 11 Datový model

7.4 Wireframe a grafické rozhraní



Obrázek 12 Wireframe

Na ukázku uveden pouze jeden wireframe, ostatní jsou součástí přílohy.

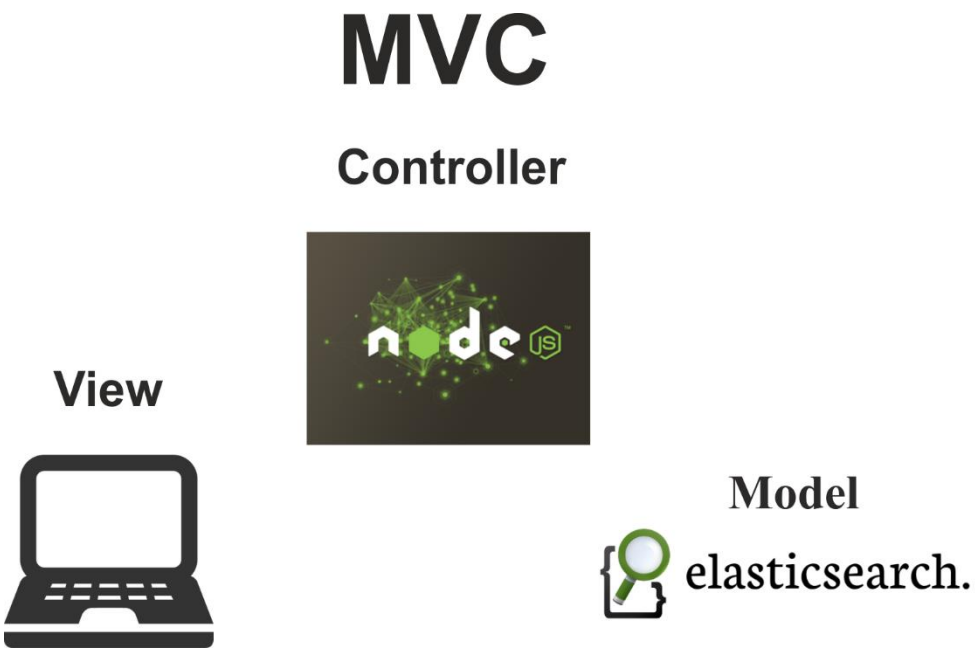
Grafická návrh byl uzpůsoben dvěma vlastnostem:

1) Aby bylo ovládání aplikace co nejjednodušší a intuitivní
2) Aby aplikace svou grafickou úpravou podporovala zobrazení jak na desktopových zařízeních, tak na mobilních zařízeních (smartphonech, tabletech), proto byl na klientovy použit framework Bootstrap.

Pro návrh barevného schématu aplikace byla použita webová stránka http://colorschemedesigner.com/.

7.5 MVC

Pro proof-of-concept byla vybrán návrhový vzor MVC pro jeho snadnou aplikovatelnost v javascriptu, jednoduchost a přehlednost. Jako view bude v tomto případě sloužit klient ve webovém prohlížeči PC nebo mobilního zařízení. Model bude indexovací databáze Elasticsearch a controller NodeJS server.



Obrázek 13 MVC architektura proof-of-concept

8 Implementace

Implementace je rozdělena na dvě části. A to implementace klienta a implementace serveru.

Server je pak rozdělen do dvou částí. První z nich je samotný server s implementovaným faceted search a druhá je agregátor sbírající data pro Elasticsearch.

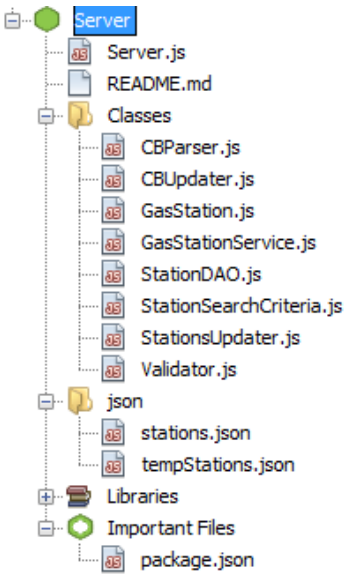
8.1 Vývojové prostředí

Jako vývojové prostředí bylo vybráno programátorské prostředí Netbeans IDE pro jeho robustnost, stabilnost a podporu všech použitých programovacích jazyků. Nejdříve bylo použito ve verzi 7.4 a potom ve verzi 8.0. Toto prostředí bylo použito jak na vývoj klienta v HTML, CSS a javascriptu, tak na vývoj serveru v technologii NodeJS. Pro usnadnění vývoje byl do prostředí Netbeans doinstalován plugin nodejs, který přidává do Netbeans podporu pro NodeJS projekty. Dále byl pro vývoj použit NodeJS server ve verzi 0.10.26 a Elasticsearch ve verzi 1.1.0.

8.2 Adresářová struktura

8.2.1 Server

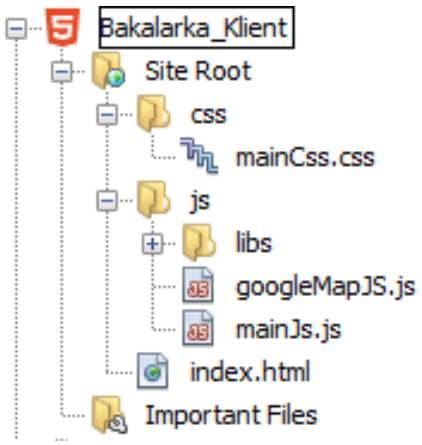
V kořenovém adresáři je umístěn spouštěcí skript celé aplikace Servers.js a README soubor. V adresáři Classess jsou umístěny skripty se třídami a v adresáři json pomocné soubory s daty typu json.



Obrázek 14 Adresářová struktura server

8.2.2 Klient

Má klasickou adresářovou strukturu pro HTML stránku. V kořenovém adresáři je umístěn soubor index.html. V adresáři css soubor se styly. V adresáři js skripty a v js/libs frameworky JQuery a Boostrap.



Obrázek 15 Adresářová struktura klient

8.3 Server

8.3.1 Třída Server

Třída Server slouží jako spouštěcí skript celé aplikace. Je zde nadefinována konfigurace aplikace s řešením pro CORS (viz. obrázek č.16).

```
app.configure(function ()
        {
          app.use(express.json());
          app.use(express.methodOverride());
          app.use(express.bodyParser());
          app.use(app.router);

        });


app.all('*', function(req, res, next) {
          res.header("Access-Control-Allow-Origin", "*");
          res.header('Access-Control-Allow-Methods', 'HEAD, GET, POST, PUT, DELETE');
          res.header("Access-Control-Allow-Headers", "Content-Type");
          next();
         });
```

Obrázek 16 Třída Server nastavení aplikace

Má v sobě implementované RESTové rozhraní přes které přijímá požadavky od klienta. Ty pak validuje, jestli parametry požadavků odpovídají implementovanému standardu pro

daný požadavek. Potom je předává instanci třídy `GasStationService`, ve které je uložena business logika aplikace k dalšímu zpracování. Předání probíhá funkcemi `getStationsByDistance, getStationsByDistrict, getStationsByFuel`, podle toho na jakou url požadavek přišel. Poté co se vrátí výsledek po zpracování, ho ve formátu JSON odesílá klientovy.

8.3.2 Třída GasStationService

Je v ní implementována business logika aplikace. Například se zde počítají geolokační údaje pro vyhledávání nebo jsou zde uloženy konstanty pro tyto výpočty. Tato třída také vytváří instanci třídy `StationSearchCriteria` a přes funkci `createSearchCriteria` zadává parametry daného vyhledávání.

```
GasStationService.prototype.createSearchCriteria = function(d, param) {
    var criteria = new StationSearchCriteria();
    if (d == 1)
    {
        this.calculateDistance(param);
        criteria.setMinimumLatitude(this.minLat());
        criteria.setMaximumLatitude(this.maxLat());
        criteria.setMinimumLongitude(this.minLong());
        criteria.setMaximumLongitude(this.maxLong());
        criteria.setSearch("distance");
    }
```

Obrázek 17 Příklad vytvoření parametrů pro vyhledávání

Tento objekt pak předává instanci třídy StationDAO, která pak provede samotné vyhledání v databázi Elasticsearch.

8.3.3 Třída StationDAO (Station Data Access Object)

Vytváří spojení s databází Elasticsearch a provádí nad ní vyhledávací funkce a funkce typu CRUD. K tomu používá implementovaný modul elasticsearch pro NodeJS. Spojení je vytvořeno na bázi skrze RESTového rozhraní databáze Elasticsearch a veškeré dotazy se odesílají ve formátu JSON na port databáze. Třída v konstruktoru přijímá jako parametr objekt typu `StationSearchCriteria`, podle kterého pak spustí daný dotaz nad databází. Tato třída je také využita agregátorem pro vkládání a update dat.

8.3.4 Třída StationSearchCriteria

Třída sloužící pouze jako storage pro parametry vyhledávání.

8.3.5 Třída Validator

Třída slouží k validaci vstupů z klienta. Pro tuto funkci je v ní implementovaný modul validator pro NodeJS. Třída má hlavní funkci `validate`, která přijímá řetězec k validaci a vrací proměnou typu boolean s hodnotou true nebo false podle výsledku validace.

8.4 Server – část agregátor

8.4.1 Třída StationsUpdater

„Interface“ pro update databáze Elasticsearch. Jazyk javascript tudíž i NodeJS strukturu typu interface přímo nepodporuje, v důsledku se tedy jedná o normální třídu, navrženou ale tak aby fungovala jako interface.

8.4.2 Třída CBParser

Slouží k získávání dat ze stránek projektu ceskybenzin.cz. Tyto data parsuje, předělává na objekty a ukládá do souboru typu json. Parsování se spouští jednou ze dvou hlavních metod `parse`, která nejdříve vygeneruje pole s url adresami na kterých se nacházejí data pro aplikaci. Pak pošle na všechny tyto url http request a odpověď rozparsuje na třídy typu `GasStation` a uloží do souboru `tempStations.json`. Tento soubor za použití metody `getLatLongToStations` znovu načte a ke každému objektu typu GasStation za pomoci metody geocode přiřadí geolokační údaje. Tyto aktualizované objekty nakonec uloží do dalšího souboru `stations.json`.

```
CBParser.prototype.getLatLongToStations = function() {
    var final = [];
    var file = '../json/tempStations.json';
    var f = jf.readFileSync(file);
    f.forEach(function (obj){
            var gs = new GasStation();
            gs.setName(obj.name);
            gs.setAddress(obj.address);
            gs.setDistrict(obj.district);
            gs.setGasAndPrices(obj.gasAndPrices);
            geocoder.geocode(obj.address+ '  Česká republika')
            .then(function(res) {
                gs.setLatitude(res[0].latitude);
                gs.setLongitude(res[0].longitude);
            console.log(JSON.stringify(gs));
            final.push(gs);
            console.log(JSON.stringify(final));

            fs.writeFile('../json/stations.json', JSON.stringify(final, null, 4), function (err) {
              if (err) throw err;
              console.log('It\'s saved!');
            });
    });
```

Obrázek 18 Ukázka metody getLatLongToStations

```
{
    "name": "Agip",
    "address": " Brno - Bauerova",
    "district": "JHM",
    "gasAndPrices": {
        "  Natural 98": 38.5,
        "  Natural 95": 36.5,
        "  Nafta": 36.5
    },
    "latitude": 49.18647,
    "longitude": 16.5736085
},
```

Obrázek 19 Ukázka ze souboru stations.json

8.4.3 Třída CBUpdater

Spouští její metodou `update` update dat v databázi. Načte soubor `stations.json`, vytvoří instanci třídy `StationDAO` a pro každý objekt typu `GasStation` zavolá metodu `createGasStation` která objekt uloží do databáze.

```
// FCE pro vložení stanice do DB
StationDAO.prototype.createGasStation = function(gasStation) {
        var gs = gasStation;
        client.create({
                index : "stations",
                type : "station",
                body : {
                        name : gs.name,
                        adress : gs.address,
                        district : gs.district,
                        latitude : gs.latitude,
                        longitude : gs.longitude,
                        gasAndPrices : gs.gasAndPrices

                }
        }
        ,function (error, response)
        {
                console.log(error);
        });
};
```

Obrázek 20 Ukázka metody createGasStation

8.4.4 Třída GasStation

Definuje objekt typu GasStation do kterého parser ukládá data o benzínových stanicích a zároveň slouží jako datový model pro databázi.

```
function GasStation()
{
        this.name = '';
        this.address = '';
        this.district = '';
        this.latitude;
        this.longitude;
        this.gasAndPrices = new Object();
};
```

Obrázek 21 Ukázka property třídy GasStation

8.5 Klient

Jedná se o standartního klienta v podobě webové aplikace vytvořené v technologiích html, javascript a za použití frameworků JQuery, Bootstrap, iCheck a Gmaps.js.

8.5.1 Design

Design aplikace byl navržen tak, aby podléhal těmto kritériím.

- Přehlednost a jednoduchost
- Snadná obsluha
- Vysoký kontrast a s tím spojená dobrá čitelnost
- Přijatelné zobrazování na mobilních zařízeních

Toho bylo dosaženo tím, že v samotné aplikaci je menu pouze s třemi položkami. Dále je design aplikace navržen tak aby podporoval jednoduchý princip založený na „Vyber možnost a potvrď výběr“. Pro vysoký kontrast bylo vybráno velmi světlé pozadí a tmavý font písma.

8.5.2 Javascript na straně klienta

Javascript na straně klienta je rozdělen do tří kategorií

1. Obsluhující webové rozhraní – obsahuje funkce pro vykreslování menu a funkce pro zjišťování aspektů vyhledávání zadaných uživatelem. Je uložen v souboru `mainJS.js`.

2. Ajax pro komunikaci se serverem NodeJS – zajišťuje odesílání parametrů na server NodeJS ve formátu JSON. K tomu používá funkci frameworku JQuery `ajax`. Je uložen v souboru `mainAJAX.js`.
3. Skripty pro vykreslování mapy – zajišťují vykreslování údajů ze serveru na mapu tím, že benzínové stanice odeslané serverem předělávají na markery v mapě. Také zajišťují překreslování mapy a její aktualizace. Uloženy jsou v souboru `googleMapJS.js`.

8.5.3 Google Maps

Jako jeden z hlavních parametrů v klientu je implementace Google Maps API pro vykreslení výstupních dat ze serveru na mapě. Google map je implementována přes knihovnu Gmaps.js, která posílá request na mapu zadaných parametrů a následně na ní vykreslí data.

Tato serverová služba je zde využívána verzi Free Non-Commercial. Pro lepší funkcionalitu je zde implementována knihovna Gmaps.js.

9 Testování

Testování probíhalo formou ručního testování vývojáře. Pro testování klienta byly použity webové prohlížeče Chrome a Firefox s pluginem firebug. Pro testování serverové části byl použit modul pro server NodeJS node-inspector.

Testování probíhalo na několika úrovních:

1. Kontrola funkčnosti kódu.
2. Kontrola vstupních a výstupních hodnot funkcí. Zvláště bylo nutno kontrolovat, jestli vstupní hodnoty funkcí v daném čase existují nebo je vstupní hodnotou proměnná typu undefined
3. Kontrola chybovosti agregátoru.
4. Kontrola funkčnosti aplikace

9.1 Příklady řešených problémů

9.1.1 Charset

Vstupní data pro agregátor, který je dále zpracovával podle modelu ETL přicházela ve špatném charsetu windows1250 namísto vyžadovaného UTF8. Problém byl vyřešen tak, že modul zpracovávající http response byl použit tak aby data automaticky nepřeváděl na proměnou typu string, ale data ponechal jako pole bytů, které se pak modulem iconv konvertuje do proměnné typu string s už správným charsetem.

9.1.2 Asynchronnost u agregátoru

Jedna z hlavních vlastností serveru NodeJS asynchronnost se ukázala u agregátoru jako velký problém. Agregátor zpracovává větší množství dat, což platí jak pro parser, tak pro updater. Toto zpracovávání může trvat dohromady i několik minut, například parser zpracovává data průměrně 45 sekund. Tím dochází k tomu, že se funkce, které se vyvolávají nad daty, spustí v době, kdy data ještě neexistují, nebo se nespustí, protože jim vypršel timeout.

Tento problém byl po zdlouhavé době testování vyřešen tak, že do té doby robustní agregátor byl rozdělen na menší, na sobě nezávislé úseky, které se spouštějí samostatně a jejichž výstupem jsou data ve formátu json, sloužící jako vstupní data pro další úsek.

10 Vyhodnocení

10.1 Užití faceted search v praxi

Faceted search se vyplatí použít u aplikací pracujících s velkým objemem dat. Jak v ohledu indexování záznamů v obrovských databázích, tak zpracovávání rozsáhlých dokumentů. V tomto případě má smysl ho použít. Dotazy nad databázemi budou zpracovány s větší rychlostí v řádu viz. tabulka č.1, než u konvenčních databází sql. Naopak faceted search nemá význam implementovat nad malými databázemi v řádu stovek záznamů. Zde jsou přístupové rychlosti k databázi na srovnatelné úrovni.

Pro přehled je uvedena tabulka s rychlostmi databáze Elasticsearch. Prostředí, ve kterém byly tyto rychlosti naměřeny byl notebook vývojáře, který měl SSD disk a dostatečnou velikost pamětí RAM. Jako data bylo použito 2.1 milionů stránek na wikipedii

	completion	fuzzy 1	fuzzy 2	prefix query	match query	FST size in-memory
standard	0.72ms	1.14ms	4.14ms	4.41ms	5.80ms	82mb
optimized	0.53ms	0.69ms	2.46ms	3.33ms	1.63ms	80mb
payload	0.65ms	1.16ms	4.32ms	4.22ms	6.18ms	166mb
payload + optimized	0.62ms	0.75ms	2.72ms	3.30ms	1.60ms	163mb

Tabulka 1 Rychlost Elasticsearch [8]

10.2 Implementace faceted search

Jsou zhruba čtyři hlavní implementace faceted search Elasticsearch, Solr, Sphinx a Xapian. Elasticsearch a Solr jsou oba založeny na knihovně Apache Lucane a implementovány v Javě a oba pro komunikaci používají RESTové rozhraní, z čehož vyplívá, že jsou multiplatformní. Sphinx a Xapian jsou implementovány v C++ a pro komunikaci používají vlastní protokol. Implementace se ani moc neliší v podporovaných programovacích jazycích, například všechny podporují Javu, PHP, Ruby nebo Perl.

Vzhledem k podobným vlastnostem implementací faceted search, spíše než na vlastnostech daných implementací záleží na tom, v jakém programovacím jazyce chceme projekt implementovat, a na druhotných vlastnostech projektu.

11 Závěr

V rámci bakalářské práce bylo provedeno rešeršní šetření v oblasti technologií faceted search a tato technologie byla popsána v bakalářské práci. Byl popsán Elasticsearch jako state-of-art implementace technologie faceted search. Byl vyroben proof-of-concept v podobě agregátoru služeb s podporou geografického výběru s naimplementovanou technologií faceted search. U proof-of-concept byl navržen datový model, wireframe uživatelského rozhraní, class diagram a use cases. Všechny tyto návrhy byly prokonzultovány s e školitelem. Návrhy, použité návrhové vzory a technologické rozhodnutí byly zdokumentovány v bakalářské práci. Všechny cíle stanovené pro proof-of-cocept byly splněny. Proof-of-concept je funkční, nikoliv však dokončený. Jeho termín dokončení je naplánován až po termínu odevzdání bakalářské práce.

11.1 Splnění cílů bakalářské práce

Cíl	***Výsledek***	***Poznámka***
Popsat technologii faceted search	splněno	
Popsat využití technologie faceted search v BI	splněno	
Popsat state-of-art implementaci v podobě Elasticsearch	splněno	
Vyrobit proof-of-concept aplikaci	splněno	
Navrhnout datový model, wireframe, use cases	splněno	
Navrhnout architekturu proof-of-concept a konzultovat se školitelem	splněno	
Návrh, návrhové vzory a technologická rozhodnutí zdůvodnit a zdokumentovat v práci	Splněno	

Tabulka 2 Splnění cílů bakalářské práce

V rámci bakalářské práce se také student seznámil s novými webovými technologiemi a postupy a naučil se je používat a implementovat v projektech.

Reference

1. **Wikipedia.** Agregátor. *Wikwpedia.* [Online] Wikimedia Foundation, Inc., 2014. http://cs.wikipedia.org/wiki/Agreg%C3%A1tor.

2. —. Extract, transform, load. *Wikipedia.* [Online] Wikimedia Foundation, Inc. http://en.wikipedia.org/wiki/Extract,_transform,_load.

3. —. Geolokace. *Wikipedia.* [Online] Wikimedia Foundation, Inc., 2014. http://cs.wikipedia.org/wiki/Geolokace.

4. —. Faceted classification system. *Wikipedia.* [Online] Wikimedia Foundation, Inc., 2014. http://en.wikipedia.org/wiki/Faceted_classification.

5. —. Faceted Search. *Wikipedia.* [Online] Wikimedia Foundation, Inc., 2014. http://en.wikipedia.org/wiki/Faceted_search.

6. **CZC.** CZC. *CZC.* [Online] Czech Computer, 2014. czc.cz.

7. **Delft University of Technology.** Twitter Faceted Search. *Twitter Faceted Search.* [Online] Web Information Systems, Delft University of Technology, 2014. http://www.wis.ewi.tudelft.nl/research-lines/faceted-search/.

8. **Elasticsearch.** Elasticsearch. *Elasticsearch.* [Online] Elasticsearch BV, 2014. http://www.elasticsearch.org/.

9. **Nešetřil, Jakub.** NodeJS. *Zdroják.* [Online] Devel.cz Lab s.r.o., 2014. http://www.zdrojak.cz.

10. **Letaifa, Nagi.** A Full Javascript Architecture, Part One - NodeJS. *Zenika.* [Online] Zenika, 2014. http://blog.zenika.com/index.php?post/2011/04/10/NodeJS.

11. **Joyent, Inc.** API. *Node.js.* [Online] Joyent, Inc., 2014. http://nodejs.org/.

12. **Sphinx Technologies Inc.** About. *Sphinx.* [Online] Sphinx Technologies Inc., 2014. http://sphinxsearch.com/about/sphinx/.

13. **Persyn, Jurriaan.** Introduction to Elastic search. *jurriaanpersyn.* [Online] 2013. http://www.jurriaanpersyn.com/archives/2013/11/18/introduction-to-elasticsearch/.

14. **Sissel, Jordan.** Logtash. *Logtash.* [Online] 2014. http://semicomplete.com/.

15. **Wikipedia.** Jquery. *Wikipedia.* [Online] Wikimedia Foundation, Inc., 2014. http://en.wikipedia.org/wiki/JQuery.

16. **Leon, Gustavo.** gmaps.js. *GitHub.* [Online] MIT, 2014. http://hpneo.github.io/gmaps/.

17. **MIT.** Getbootstrap. *Getbootstrap.* [Online] MIT, 2014. http://getbootstrap.com/css/.

18. **Refsnes Data.** W3Schools. *W3Schools.* [Online] Refsnes Data, 1999-2014. http://www.w3schools.com/.

19. **Wikipedia.** Twitter Bootstrap. *Wikipedia.* [Online] Wikimedia Foundation, Inc., 2014. http://cs.wikipedia.org/wiki/Twitter_Bootstrap.

20. **The Apache Software Foundation.** Apache Solr. *Apache.org.* [Online] The Apache Software Foundation, 2014. https://lucene.apache.org/solr/.

21. **Xapian.** Xapian Project. *Xapian.* [Online] Xapian Project, 2014. http://xapian.org/.

Přílohy

A Aplikace

Aplikace se nachází na přiloženém cd.

B Požadavky a postup instalace

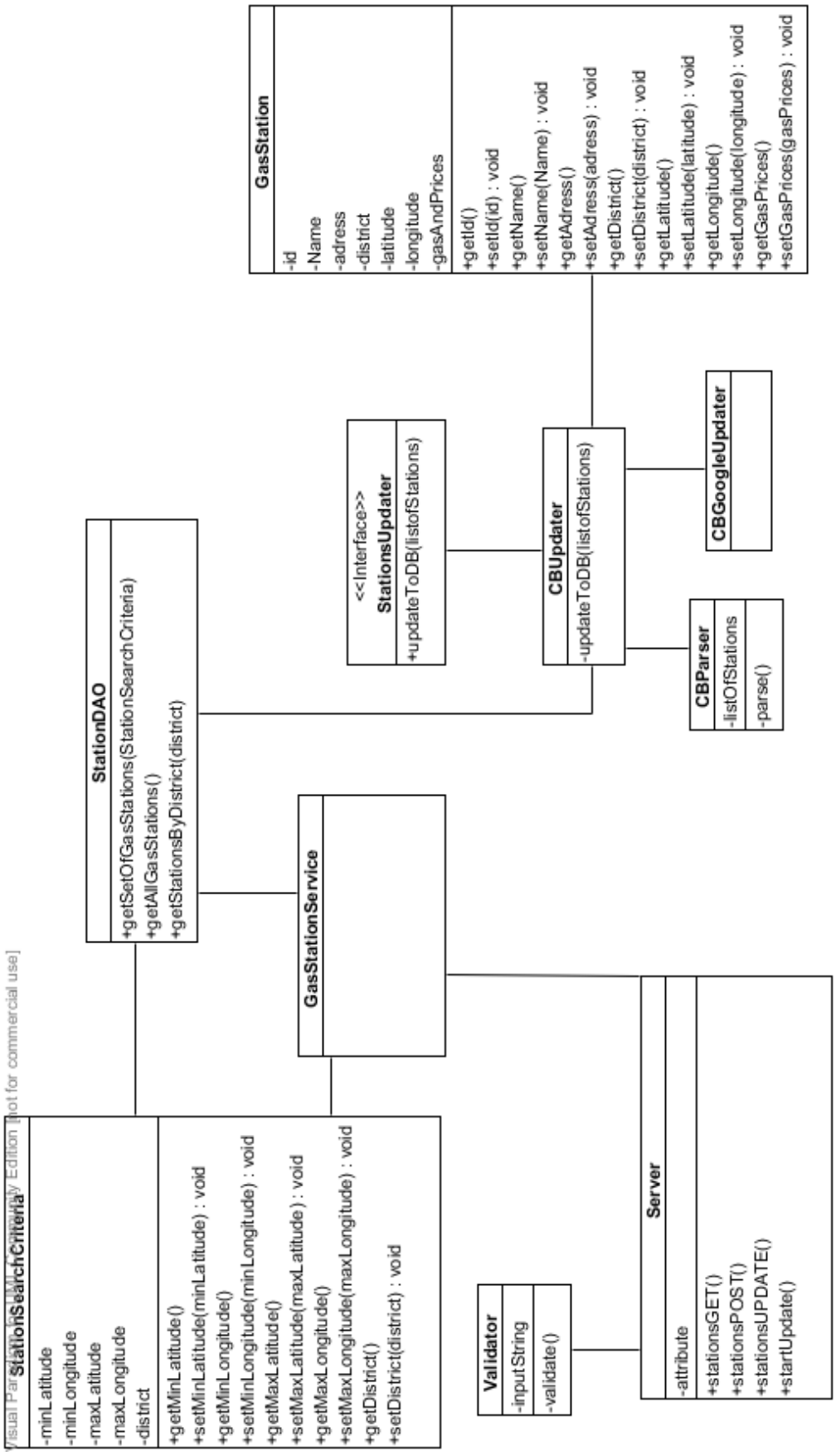
Požadavky

- NodeJS server 0.10.26 a vyšší
- Elasticsearch 1.1.0 a vyšší
- Apache2

Postup instalace

1. Rozbalte soubor BP-TomasHonner-program.rar uložený na cd.
2. Složku Klient zkopírujte do adresáře určeného pro weby http serveru Apache.
3. V příkazové konzoli systému spusťte službu elasticsearch.
4. V příkazové konzoli systému změňte aktuální adresář na Server\Classes a příkazem node spusťte soubor CBParser.js a vyčkejte, než se provedou všechny operace.
5. V příkazové konzoli systému příkazem node spusťte soubor CBGoogleUpdater.js a vyčkejte, než se provedou všechny operace.
6. V příkazové konzoli systému příkazem node spusťte soubor CBUpdater.js a vyčkejte, než se provedou všechny operace.
7. V příkazové konzoli systému změňte adresář na Server a příkazem node spusťte soubor Server.js, čímž se spustí server a jeho RESTové rozhraní.
8. Spusťte webový server Apache a ve webovém prohlížeči zadejte url adresu klienta.

C Class Diagram



D Seznam použitých zkratek

API	Application Programming Interface
AJAX	Asynchronous Javascript And XML
BI	Business Intelligence
CORS	Cross-Origin Resource Sharing
CRUD	Create Read Update Delete
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
MIT	Massachusetts Institute of Technology
NoSQL	No Structured Query Language
REST	Representational State Transfer
SQL	Structured Query Language
SSD	Solid State Drive

E Seznam použitých obrázků a tabulek