Vysoká škola ekonomická v Praze

Fakulta informatiky a statistiky

Katedra informačních technologií

Studijní program: Aplikovaná informatika

Obor: Informační systémy a technologie

Vyhledávání jako služba

DIPLOMOVÁ PRÁCE

Student: Bc. Luděk Veselý

Vedoucí: Prof. Ing. Zdeněk Molnár, CSc.

2017

University of Economics in Prague

Faculty of Informatics and Statistics

Department of Information Technologies

Study programme: Applied Informatics

Branch: Information Technologies

Search as a Service

DIPLOMA THESIS

Student: Bc. Luděk Veselý

Supervisor: Prof. Ing. Zdeněk Molnár, CSc.

2017

Tento list nahraďte
originálem zadání.

Prohlášení

Prohlašuji, že jsem diplomovou práci zpracoval samostatně a že jsem uvedl všechny použité prameny a literaturu, ze které jsem čerpal.

Datum:

Podpis:

Abstrakt

Tato diplomová práce popisuje návrh a tvorbu fulltextového vyhledávání poskytovaného jako služba.

Klíčová slova

Fulltext, Elasticsearch

Abstract

This diploma thesis describes creation of fulltext search service.

Key words

Fulltext, Elasticsearch

Poděkování

Rád bych poděkoval Ing. Ivanovi Jelínkovi za rady a pomoc při řešení.

Obsah

Seznam zkratek

URL	Uniform Resource Locator, adresa dokumentu na webu
XML	Extensible Markup Language, značkovací jazyk
EAN	European Article Number, mezinárodní číselná identifikace výrobků
ISBN	International Standard Book Number, číselná identifikace knižních vydání
SQL	Structured Query Language, jazyk určený pro práci s relačními databázemi
HTTP	Hypertext Transfer Protocol, protokol pro přenos souborů po síti internet
API	Application Programming Interface, rozhraní pro programování aplikací
REST	Representational State Transfer, architektura rozhraní pro komunikaci mezi počítačovými systémy
SOAP	Simple Object Access Protocol, protokol pro výměnu zpráv přes síť

1 Úvod

V této diplomové práci práci se zabývám návrhem a implementací nástroje, který umožňuje provozovatelům elektronických obchodů snadnou a rychlou implementaci plnotextového vyhledávání. Při jeho implementaci je totiž třeba mít určitou úroveň znalosti principů samotného vyhledávání včetně souvisejících oborů a technologií. Implementace kvalitního vyhledávání tak může být pro provozovatele elektronického obchodu náročná jak časově, tak finančně. Tento nástroj je poskytován jako služba se všemi výhodami i omezeními s tímto principem spojenými. Konkrétní cíle diplomové práce popisuji v následujících odstavcích.

1.1 Cíle práce

Cílem práce je vytvořit nástroj, který je nabízen jako služba a umožňuje snadnou implementaci plnotextového vyhledávání do elektronického obchodu. K dosažení tohoto cíle je třeba naplnit cíle dílčí.

Prvním dílčím cíem je provedení analýzy problému a to jak z pohledu byznysového, tak z pohledu samotné problematiky vyhledávání. Co se týče pohledu byznysového, tak je třeba analyzovat potřeby zákazníků, zjistit, jaká jsou specifika oblasti elektronického obchodování a definovat kritéria, jejichž naplnění je pro provozovatele elektronických obchodů klíčové. Z pohledu problematiky vyhledávání je pak třeba provést analýzu této disciplíny a utřídit tak znalosti potřebné k poskytnutí kvalitních výsledků plnotextového vyhledávání.

Dalším dílčím cílem je vytvořit návrh řešení problému jednak na základě znalosti potřeb zákazníků a také na základě znalosti problematiky plnotextového vyhledávání. Tento návrh musí být v dalším kroku implementovatelný.

Posledním dílčím cílem je samotná implementace aplikace dle jejího návrhu. Obnáší to výběr vhodných nástrojů, jejich nasazení do produkčního prostředí, naprogramování jednotlivých služeb a konečně také ověření samotné implementace. Tu je třeba porovnat vůči požadavkům a ověřit tak jejich naplnění.

1.2 Cílová skupina

První cílovou skupinou je vývojář, řešící problém plnotextového vyhledávání produktů při implementaci elektronického obchodu. Takovému čtenáři by měla práce poskytnout dostatečné teoretické znalosti potřebné pro implementaci vyhledávání.

Užitečná také může být konkrétní podoba implementace, která je v této práci popisována.

Další možnou cílovou skupinou je provozovatel elektronického obchodu, který přemýšlí, jakým způsobem zlepšit (případně zavést) plnotextové vyhledávání. V této práci získá přehled o složitosti samotné implementace, poskytne mu komentovaný soupis možných řešení a konečně také poskytne funkční službu, kterou může okamžitě na webový portál napojit.

Poslední cílovou skupinou budiž kdokoli, kdo se zajímá o problematiku plnotextového vyhledávání v českém jazyce. V této práci nalezne soupis problémů souvisejících s češtinou, které je třeba řešit. Dále zde nalezne konkrétní implementaci, kterou se může inspirovat při řešení obdobného problému.

1.3 Použité metody

V této části popisuji metody použité k naplnění jednotlivých cílů. Pro zkoumání problému oboru provedu analýzu trhu, nabízí se také možnost dotazování potenciálních uživatelů. Pro porozumnění problematice vyhledávání provedu rešerši literatury. Vzhledem k množství dostupné literatury bude třeba provést syntézu těchto informací. Při vytváření návrhu řešení bude použito modelování, výstupem by tedy měl být model řešení.

1.4 Struktura práce

V první části práce popisuji problematiku vyhledávání v prostředí elektronického obchodování. Definuji zde jednak kontext, v kterém se pohybuji a dále také popisuji problémy, které v tomto prostředí existují. Snažím se identifikovat potencionálního uživatele a definovat jeho požadavky na plnotextové vyhledávání. Toto prostředí má svá specifika, která také popisuji a vysvětluji, přoč jsem se na tuto oblast zaměřil. V závěru této části porovnávám existující nástroje umožňující implementaci vyhledávání a zjišťuji tak, proč má smysl vytvářet další službu, jaká je její přidaná hodnota.

V druhé části se zabývám teorií plnotextového vyhledávání. Popisuji zde celý proces od analýzy vstupních dat, přes jejich indexaci, až po samotné vyhledávání. Tyto poznatky budou následně využity k naplnění požadavků na vyhledávání, k zajištění kvalitních výsledků vyhledávání.

V další části práce navrhuji samotnou aplikaci tak, aby vyhovovala požadavkům a zároveň byla následně implementovatelná. Porovnávám zde dostupné nástroje, definuji případy užití aplikace a vytvářím model výsledné aplikace.

V poslední části diplomové práce popisuji konkrétní implementaci v jazyce Go s pomocí úložiště Elasticsearch. Výstupem této části je otestovaný spustitelný program, který umožňuje provádět indexaci produktů a jejich následné vyhledávání.

2 Analýza byznys požadavků na aplikaci

V první části diplomové práce popisuji problematiku plnotextového vyhledávání v prostředí elektronických obchodů. Nejprve je čtenář uveden do problematiky elektronického obchodování a obeznámen s základními pojmy a principy. Následně je popsán problém, který je řešen a jsou definovány konkrétní podmínky, za kterých bude výsledný nástroj přínosný a použitelný. Jsou zde také diskutovány stávající možnosti implementace plnotextového vyhledávání a porovnány alternativní služby.

2.1 Vysvětlení základních pojmů

Pro usnadnění orientace v této práci nejprve vysvětluji dále používané pojmy. Jedná se vesměs o známé termíny, přesto považuji za důležité uvést jejich význam na pravou míru.

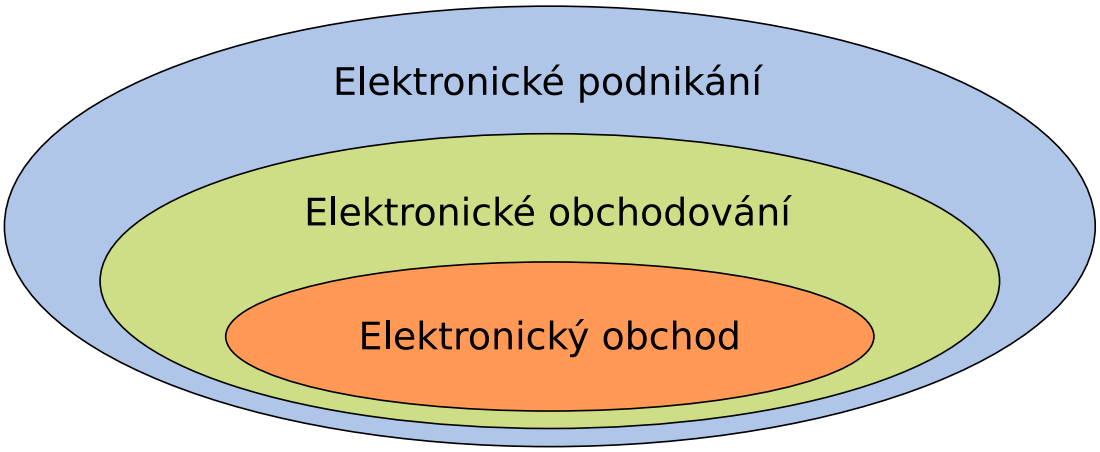
Elektronické obchodování

Termín elektronické obchodování (e-commerce) označuje veškerou činnost spojenou s obchodováním realizovaným prostřednictvím sítě internet [1, strana 11]. Spadá sem distribuce, nákup, prodej, marketing, servis produktů. Elektronické obchodování se dále podle dělí zaměření na cílové skupiny, z nichž nejpodstatnější jsou B2C (zaměřeno na koncové zákazníky) a B2B (zaměřeno na obchodníky) [1, strana 17]. Zároveň je však elektronické obchodování podmnožinou elektronického podnikání (e-business), které označuje veškeré obchodní a výrobní aktivity, zatímco elektronické obchodování se týká samotného prodeje zboží a služeb.

Elektronický obchod

Elektronický čí internetový obchod (e-shop) je webová aplikace, jejímž prostřednictvím provozovatel obchodu prodává zboží nebo služby zákazníkům [1, strana 16]. Jedná se o podmožinu elektronického obchodování, jde tedy o jeden z možných prodejních kanálů v prostředí internetu. Elektronický obchod je specifickou webovou aplikací, v níž se opakují určité vzorce. Konkrétně je to katalog produktů, kategorizace zboží, vkládání položek do košíku, platba objednávky, možnost kontaktu s zákaznickou podporou prostřednictvím on-line chatu přímo v aplikaci.

V elektronickém obchodě můžeme shledávat řadu podobností s obchodem kamenným, který zákazníci navštěvují osobně. Některé vlastnosti elektronického obchodu jsou zřejmě inspirovány obchody kamennými (procházení zboží podle kategorií, přidávání položek do košíku, uplatnění slevových poukázek při platbě). Často však elektronický obchod využívá možností, které webové prostředí nabízí (filtrace a řazení položek podle parametrů, vrácení se k naplněnému košíku, porovnávání zboží napříč obchody).



Obrázek 2.1: Vztah elektronického podnikání, obchodování a podnikání

Zcela zásadní výhodou elektronického obchodu oproti návštěvě obchodu kamenného je možnost vytvoření objednávky z pohodlí domova a její následné doručení zásilkovou službou. Tento přístup především šetří čas strávený cestou do obchodu a čas strávený v obchodě. Například v segmentu nákupu potravin může být výhodou to, že je takový nákup hygieničtější a zboží bude doručeno v lepším stavu díky uzpůsobeným vozům dopravců. Výhody ale vznikají i pro provozovatele – nemusí zřizovat prodejnu včetně jejího vybavení a zaměstnance, mohou efektivněji navrhnout sklad, nebo dokonce využít automatizace při kompletaci objednaného zboží.

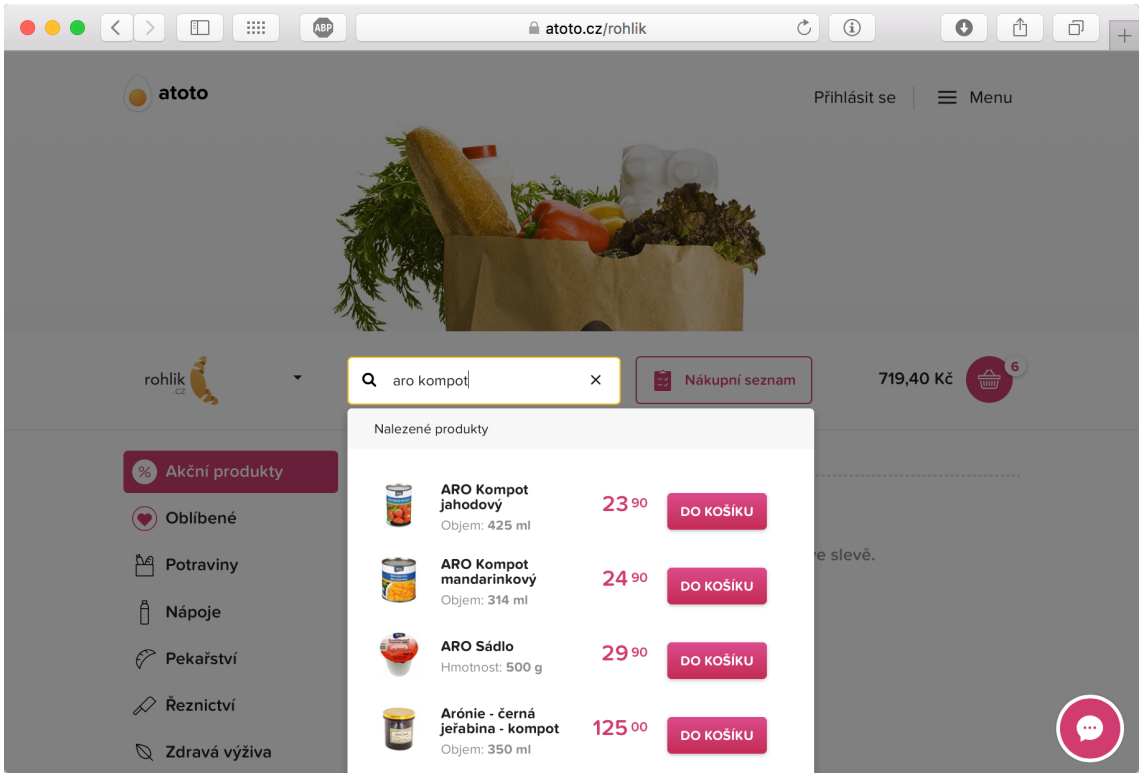
Plnotextové vyhledávání

Plnotextové vyhledávání (full-text search), je způsob vyhledávání v textových datech, kdy se vyhledává uživatelem formulovaný dotaz v invertovaném souboru (někdy také indexovém souboru nebo invertovaném indexu), tedy v souboru obsahující výrazy, podle nichž je daný záznam vyhledatelný [2, strana 15]. Proces, kterým vzniká indexový soubor se nazývá indexace. Tato problematika je podrobněji popisována v druhé kapitole.

2.2 Vysvětlení problému, který je řešen

Plnotextové vyhledávání je rychlým prostředkem nalezení konkrétního produktu v elektronickém obchodě. Typicky je využito zákazníky, kteří jsou na webu s konkrétní potřebou a jsou schopni formulovat výraz, podle kterého produkt hledají. Může jít o název tybu produktu, jeho značku, variantu nebo kategorii. Pro takového uživatele je procházení webu procházením kategorií zdlouhavé a právě plnotextové vyhledávání mu může zásadně zrychlit cestu k nalezení hledaného produktu.

Důležitost plnotextového vyhledávání navíc roste s rostoucím počtem produktů a kategorií, v kterých je složité se orientovat. Uživatel tak může vyhledávání využít už pro nalezení kategorie, která se nachází v dlouhém a nepřehledném menu.



Obrázek 2.2: Ukázka plnotextového vyhledávání v elektronickém obchodu

Implementace plnotextového vyhledávání však není triviální záležitostí. Vstup uživatele je v přirozeném jazyce, takže je třeba jej odpovídajícím způsobem zpracovat, aby bylo vůbec vyhledávání na webu použitelné. Studium této problematiky může být zdlouhavé a v konečném důsledku drahé. Pokud bude k dispozici hotové řešení, které bude snadno nasaditelné na elektronický obchod a bude umožňovat jednoduché uzpůsobení potřebám konkrétního webu, bude to výrazná úspora jak času, tak peněz provozovatele obchodu.

2.3 Specifika elektronického obchodování

V prostředí elektronického obchodování existují jisté vzorce které se opakují a odlišují toto prostředí od ostatních. Vyhledávají se zpravidla produkty (případně služby), které jsou zařazeny do určitých kategorií a disponují atributy jako cena, název, kód, popis, url, obrázek, dostupnost a dalšími parametry, které bývají číselné (hmotnost, objem) nebo výčtové (barva, značka).

Z toho vyplývá co a podle čeho se bude vyhledávat. Produkty mívají jednoznačné identifikátory, které jsou navíc standardizovány (EAN, ISBN). Pomocí těchto kódu je možné vyhledat produkty napříč elektronickými obchody nebo je párovat v cenových srovnávačích. Pro vyhledávání je dále důležitý název produktu, případně jeho varianty, která upřesňuje konkrétní verzi produktu. Méně častěji je nutné vyhledávat v popisu produktu, kde bývá uveden jak slovní popis, tak výpis parametrů.

U produktů dále bývá evidována cena, která může být interně uložena v kombinaci s marží. Obchody operující se zlevněným zbožím pak mívají uvedenou cenu jak před slevou, tak po této slevě, aby zákazník viděl, kolikaprocentní sleva je na produktu. I samotná cena produktu (případně marže prodejce) může hrát svou roli ve vyhledávání.

2.4 Definice požadavků na vyhledávání

Požadavky na samotné vyhledávání jsou ovlivněny prostředím, ve kterém vyhledávání probíhá a uživateli, kteří vyhledávání využívají. Primárně by mělo být možné nalézt daný produkt podle jeho názvu, nebo části názvu. To samé platí pro kód produktu. Produkt by měl být ale vyhledatelný i podle dalších atributů, jako je název kategorie nebo název značky výrobce. Někdy dokonce uživatel nemusí vyhledávat konkrétní produkt, ale jen produkty dané značky nebo kategorie, což by měl systém také umět rozpoznat.

Podoba vyhledávaných výrazů bude zadávána zákazníky v přirozeném jazyce, s čímž by si mělo vyhledávání také poradit. Konkrétně jde o tvarosloví, kdy může uživatel zadat výraz například v jiném pádu, než je uvedeno v názvu produktu. Dále jde o vyrovnání se s chybami, které mohou vzniknout při formulaci vyhledávaného výrazu – překlepy nebo pravopisné chyby. V neposlední řadě jde také o vztah slova k jeho významu, kdy může jedno slovo mít více významů (homonymum) nebo naopak více slov může odpovídat jednomu výrazu (synonymum).

Další problematikou při zadávání hledaného výrazu uživatelem, je poskytování relevantních výsledků už ve chvíli, kdy uživatel formuluje dotaz, tedy jej teprve píše. V takovém případě je třeba odhadnout, který výraz chce napsat a tuto informaci využít při zobrazení odpovídajících výsledků.

Ve chvíli, kdy systém zná produkty, které odpovídají hledanému výrazu by měl výsledky poskytovat ve vhodném pořadí, měl by tedy pracovat s relevancí výsledků vzhledem k zadanému výrazu. Toto může být velmi obtížné, protože každý uživatel má zájem o jiné produkty a tak pro něj mohou být ro jeden výraz relevantní jiné produkty, než pro někoho jiného. Dále do tohoto pořadí můžou vstupovat požadavky

provozovatele elektronického obchodu v případě, kdy má specifické požadavky na zboží, které chce nabízet přednostně. Důvodů pro takové chování může být více, může souviset s marží konkrétních produktů, nebo s filozofií samotného obchodu.

2.5 Technické požadavky na aplikaci

Na aplikaci je kladeno několik technických požadavků. Nejedná se o funkční požadavky, nýbrž o požadavky související s použitelností, výkonností, spolehlivostí a podporou. Naplnění těchto požadavků zajistí následný bezproblémový chod aplikace a uspokojení potřeb zákazníka.

Prvním technickým požadavkem na aplikaci je její rychlost. Ta je důležitá pro spokojenost koncového zákazníka, který provádí vyhledávání. Všechny části aplikace nejsou z hlediska rychlosti tak kritické, jedná se primárně o samotné vyhledávání. Akceptovatelná rychlost uživatelského rozhraní v tomto případě je 100 ms [14], v ideálním případě do 50 ms [15]. Pro naplnění tohoto požadavku je třeba, aby byla rychlost odezvy API co nejnižší. Rychlost by dále nemělo výrazně ovlivňovat množství současných požadavků. Aplikace by měla být navržena tak, aby ji bylo možné snadno škálovat při zvyšující se zátěži.

Dalším požadavkem je formát samotného API. Bude třeba použít takový formát, který bude co nejsnadněji implementovaný v používaných webových technologiích pokud možno bez instalace jakýchkoli rozšíření. To je důležité pro rychlé nasazení aplikace do provozu a odbourání možných vstupních bariér při rozhodování o využití nástroje. Pro načtení produktů z elektronických obchodů by bylo ideální využít stávajících exportů pro jiné systémy, kterými by obchody mohly implementovat. Pokud by nic takového neexistovalo, můsí být vytvoření export produktů pro provozovatele elektronického obchodu co nejsnažší.

Posledním technickým požadavkem je dokukemantace samotného API. Dokumentace musí být dobře pochopitelná, tedy psaná přehledně, ideálně s konkrétními ukázkami použití. Výhodou je také možnost vyzkoušet si práci s API vůči testovacímu prostředí, kde nebude riziko vzniku chyb na produkčních datech.

2.6 Popis oborů, kterých se práce dotýká

Samotné řešení problematiky vyhledávání je výrazně interdisciplinární obor. Je nutné mít značné povědomí o získávání informací, počítačovém zpracování přirozeného jazyka, což souvisí jak s počítačovou lingvistikou, tak úzce s teorií formálních jazyků a překladačů. Existuje také vztah mezi lingvistikou a logikou, jakožto vědě o myšlení, která se odehrává v kategoriích lidského jazyka. Podobný vztah lze nalézt také s umělou inteligencí, vzhledem k tomu, že používání přirozeného jazyka je inteligentní činností. Díky tomu, že probíhá ukládání dat, je důležitá znalost teorie datových modelů. Vzhledem k množství ukládaných dat se také můžeme dotýkat oboru Big Data. V neposlední řadě je třeba porozumnět potřebám potenciálních uživatelů, tedy mít určitou znalost podnikání a obchodování.

2.7 Stávající možnosti implementace vyhledávání

Na trhu existuje několik služeb, které umožňují implementaci plnotextového vyhledávání. Ty se však liší funkčností, obtížností implementace i cenou. Níže popisuji nejvýznamější z nich zejména vzhledem k požadavkům na aplikaci.

2.7.1 Využití relační databáze

První možností, jak vyhledávání implementovat je využití stávající databáze, kterou elektronický obchod disponuje. Ať už se jedná o open source (MySQL, Postgres) nebo konerční (SQL Server, Oracle) databázi, možnosti plnotextového vyhledávání jsou zde omezené. Výhodou je to, že jsou data stále v jednom úložišti, není třeba řešit správu (instalaci, konfiguraci nebo zálohování) dalšího nástroje. Ani vývojáři se nemusí učit nic nového, pouze využijí stávající databázi. Vyhledávání zde probíhá v nejjednodušší formě pomocí operátoru `LIKE`. Některé databázové systémy mají disponují pokročilejšími funkcemi, kterými lze vyhledávání zlepšit [16].

2.7.2 Elasticsearch, Solr, Sphinx

Využitím nástrojů přímo určených pro implementaci plnotextového vyhledávání lze dosáhnout nejlepších výsledků, ať už se jedná o Elasticseacrch či Apache Solr využívající Apache Lucene [17], nebo Sphinx. Jde o typický krok v okamžiku, kdy přestává funkčnost relační databáze pro potřeby vyhledávání dostačovat. Tyto nástroje umožňují pokročilé nastavení indexace a zároveň jsou dostatečně rychlé, aby byly schopny provádět vyhledávání v řádu milisekund. Zřejmou nevýhodou tohoto řešení je nutnost znalosti dalšího nástroje, jeho správa a řešení synchronizace se stávající databází.

2.7.3 Algolia

Algolia je pokročilý nástroj umožňující imlementaci plnotextového vyhledávání [10]. Zaměřuje se na poskytování kvalitních výsledků v co nejkratším možném čase (v řádu milisekund). Snaží se usnadnit práci programátorům tím, že nabízí řadu připravených integrací pro konkrétní programovací jazyky a frameworky. Kromě výsledků vyhledávání umožňuje faceting, tedy dodání dat pro tvorbu filtrů, podporuje řadu jazyků nebo vyhledávání podle geografické lokality. V neposlední řadě je k dispozici dashboard zobrazující stav systému a statistiky vyhledávání.

Pokud si zákazník vystačí s 10 000 vyhledatelnými produkty, může nástroj Algolia používat zdarma, pouze je povinnen uvést ve výsledcích vyhledávání logo firmy. Placené verze začínají na 59 dolarech za měsíc za 100 000 produktů.

2.7.4 Swiftype Site Search

Swiftype Site Search je nástroj podobný nástroji Algolia, zaměřuje se však více na uživatelské rohraní a celkově na snadnost použití [11]. Mezi jeho hlavní funkce pat-

ří vyhledávání v okamžiku formulace hledaného výrazu, možnost filtrace výsledků, dodatečná úprava pořadí výsledků ručním zásahem na úrovni konkrétních výsledků, nebo na základě některého z atributů produktu. V nesposlední řadě je také k dispozici přehledná analytika proběhlého vyhledávání. Cena za provoz služby začíná na 299 USD měsíčně.

2.7.5 AWS CloudSearch

Firma Amazon poskytuje v rámci svého cloudu službu CloudSearch [12]. Jejími přednostmi jsou především vysoká výkonnost a škálovatelnost. Velkou vstupní bariérou je však její počáteční složitost. Uživatel musí nejprve proniknout do některých základních konceptů Amazon Web Services, až poté může začít pracovat se samotným vyhledáváním. Co se funkčnosti týče, nabízí služba podporu 34 jazyků, možnost nastavení váhy jednotlivých atributů, nebo doplňování texu během psaní. Cena záleží na využití výpočetní kapacity, její určení je tedy složitější.

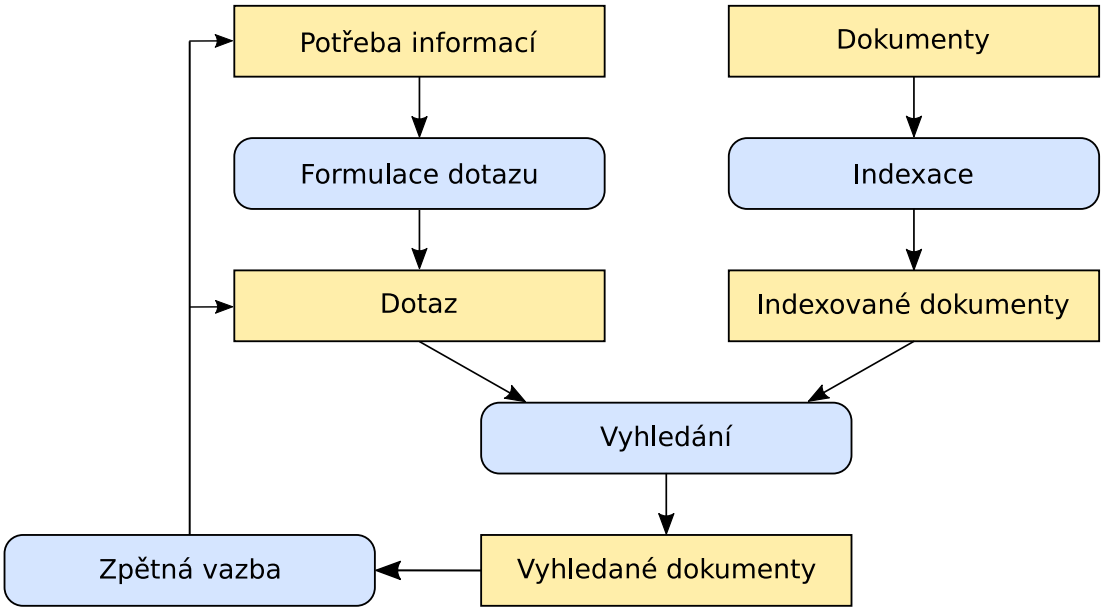
2.7.6 Google Custom Search Engine

Google Custom Search Engine [13] se nejvíc odlišuje od ostatních služeb. Vyhledávání je plně řízeno algoritmem společnosti Google a také výsledky vyhledávání vypadají obdobně, jako výsledky vyhledávání na www.google.com. Odlišné je i samotné napojení na službu – výsledky vyhledávání jsou na web vloženy jako samostatná stránka, na které je vidět logo společnosti Google. Základní varianta je však zdarma, je to tedy levná cesta, jak rychle zprovoznit vyhledávání na webu. Pokročilejší varianta umožňující konfiguraci stojí 100 USD ročně.

3 Plnotextové vyhledávání

Tato kapitola popisuje problematiku textového vyhledávání a poukazuje tak na konkrétní problémy, které je při vyhledávání nutno řešit. Problematika je rozebírána v kontextu elektronického obchodování, jsou tedy uváděny pouze principy aplikovatelné v tomto oboru.

Vyhledávání je proces sestávající jednak z ukládání dokumentů určených k vyhledávání a dále z provedení vyhledávání uživatelem. Obecné schéma vyhledávání vypadá následovně:



Obrázek 3.1: Proces vyhledávání

Ukládání dokumentů probíhá tak, že se upraví pro potřeby vyhledávání a uloží do indexu, přičemž celý tento proces se označuje jako indexace. Ve chvíli, kdy jsou dokumenty připraveny k vyhledávání, může uživatel mající potřebu v těchto dokumentech vyhledávat (získávat z nich informace) formulovat dotaz, kterým chce získat odpovídající dokumenty. Z něj je vytvořen dotaz, podle nějž bude vyhledávání provedeno. Následně jsou uživateli dodány dokumenty vyhovující dotazu, ten pak může dotaz na základě získaných dokumentů a pochopení způsobu vyhledávání dotaz upravit.

3.1 Analýza dat, v kterých bude vyhledáváno

Nejprve je třeba prozkoumat data, ve kterých bude vyhledávání prováděno. Vycházím ze skutečnosti, že naprostá většina elektronických obchodů své zboží nabízí také prostřednictvím srovnávačů zboží, přičemž v počtu návštěv a tedy celkovém provozu v rámci českého internetu jsou nejdůležitější Heureka.cz a Zboží.cz [18].

Data pro tyto srovnávače musí být dodávána prostředníctvím XML feedu, který odpovídá specifikaci jednotlivých systémů. Při porovnání obou specifikací lze nalézt jistou podobnost v požadovaných údajích. Samotný XML feed obsahuje produkty obchodu, přičemž u každého produktu je možné uvést několik atributů, z nichž některé jsou povinné. Konkrétně v případě Zboží.cz musí mít každý produkt uveden název, popis, URL, cenu a dostupnost. Dále lze využít volitelných parametrů jako název kategorie, název značky a výrobce, EAN, ISBN, produktový kód výrobce, případně další doplňkové informace a parametry [20]. V případě Heuréka.cz je podstatný název obsahující případně výrobce a produktové číslo či kód, název kategorie, dostupnost a cena dopravy. Informace o produktu mohou dále obsahovat popis, EAN, kód produktu nebo soupis parametrů [19]. Minimální XML soubor s jedním produktem ve formátu Heuréka by vypadal následovně:

```
<?xml version="1.0" encoding="UTF-8"?>
<SHOP>
 <SHOPITEM>
  <ITEM_ID>40272131</ITEM_ID>
  <PRODUCT>
   Matrace MYRBACKA - Latexová matrace, střední tvrdost, bílá
  </PRODUCT>
  <PRODUCTNAME>Matrace MYRBACKA</PRODUCTNAME>
  <DESCRIPTION>
   <![CDATA[Latex vám umožní lépe odpočívat a to tak, že sleduje
    kontury vašeho těla, uvolňuje tlak a poskytuje podporu.
     - Potah: 64% polyester, 36% bavlna
     - Vnitřní látka: Netkaný polypropylen, 100% jehněčí vlna
     - Komfortní materiál: syntetický latex, pěna polyuretan]]>
  </DESCRIPTION>
  <URL>http://ikea.com/cz/cs/catalog/products/40272</URL>
  <IMGURL>http://ikea.com/cz/cs/images/products/40272.JPG</IMGURL>
  <PRICE>9087</PRICE>
  <PRICE_VAT>10990</PRICE_VAT>
  <CATEGORYTEXT>Dům a nábytek | Nábytek | Matrace</CATEGORYTEXT>
  <MANUFACTURER>IKEA</MANUFACTURER>
 </SHOPITEM>
</SHOP>
```

Na základě znalosti struktury feedů lze zjistit, že při využití stávajících XML exportů internetových obchodů bude každý produkt obsahovat název, popis, URL,

cenu, dostupnost a název kategorie, přičemž pravděpodobně bude k dispozici řada dalších atributů, dle použetého formátu feedu a dle pečlivosti provozovatele obchodu.

3.2 Automatické indexování textů

Indexace je proces, kdy jsou ukládány textové dokumenty určené k vyhledání. Ukládány jsou pouze výrazy, podle nichž by měl být ukládaný dokument vyhledatelný, místo kam jsou tyto výrazy ukládány se nazývá index. Jedná se vlastně o klíče, podle kterých bude bude možné rychle hledaný záznam nalézt. To je důležité z hlediska rychlosti vyhledávání – není možné všechny uložené záznamy procházet a analyzovat až ve chvíli vyhledávání.

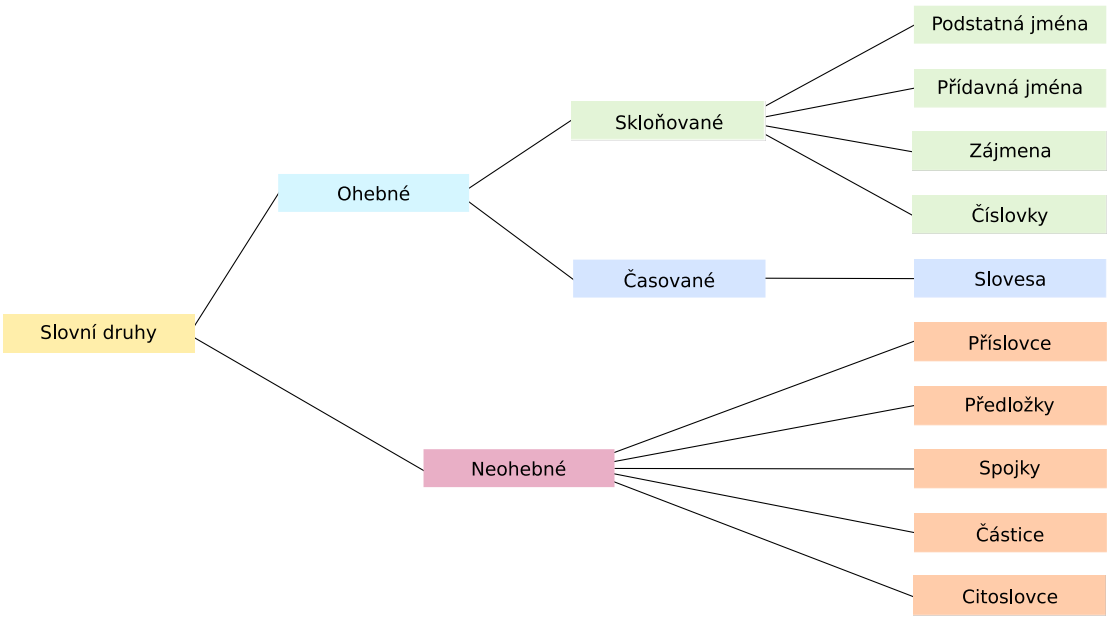
Při indexování je třeba analyzovat každý výraz, který by měl být uložen do indexu. V prvé řadě je třeba rozhodnout, zda je třeba jej do indexu ukládat, je tedy třeba řešit významnost jednotlivých výrazů v textu. Pokud daný výraz není pro vyhledávání užitečný, nemá smysl jej ukládat. Dalším problémem je tvarosloví, tedy to, že slova mohou měnit tvar (jedná se například o skloňování podstatných jmen), stále se přitom jedná o jedno slovo. V neposlední řadě je také třeba zohlednit význam slov v daném oboru, kdy mohou stejná slova vyjadřovat jiné věci (homonymie – například myš ve smyslu počítačového příslušenství a myš jako zvíře), nebo může být jeden význam vyjádřen různými slovy (synonymie – lednička i chladnička označuje totéž). Konečně je také třeba brát v úvahu další vztahy mezi slovy, jako nadřazenost a podřazenost. Například banán a ovoce mohou označovat totéž, přičemž ovoce to činí v širším slova smyslu, toto označení lze použít i pro jiné ovoce, například jablko.

3.3 Specifika českého jazyka

Vzhledem k tomu, že je vyhledávání vytvářeno pro české internetové obchody, bude veškeré vyhledávání probíhat v českém jazyce, a proto nyní popíši jeho specifika. Český jazyk pracuje se slovy odlišně než jiné jazyky. Dochází i k drobným odchylkám i v rámci českého jazyka na základě geografické oblasti (nářečí) nebo na základě zaměření dané skupiny lidí (hantýrka). Dále se můžeme zabývat grafickou podobou jazyka (morfologií) nebo jeho zvukovou podobou (fonetikou). Vzhledem k tomu, že vyhledávání probíhá na textové bázi, zabývám se dále právě grafickou podobou jazyka.

Slova jsou v českém jazyce řazena do slovních druhů, kterých je celkem deset. Slovní druh pomáhá určit chování slova – slova stejného slovního druhu se ohýbají a vyskytují v textu podle podobných pravidel. Slovní druhy lze dělit podle toho, zda jsou ohebné (tedy zda se jejich tvar může měnit) a neohebné. Neohebné slovní druhy jsou zpravidla používány ve větě společně s obehnými, protože samy o sobě nenesou informační hodnotu. Naopak ohebné slovní druhy samy o sobě nesou informační hodnotu, jsou proto vhodnými kandidáty pro to, aby se podle nich vyhledávalo. Výjimkou jsou zde zájmena, která nesou informaci nepřímo.

V rámci ohebnosti lze rozlišovat jakým způsobem jsou ohýbána. Slovesa mohou být časována, příslovce mohou být stupňována, podstatná jména, přídavná jména,



Obrázek 3.2: Dělění slovních druhů dle jejich ohebnosti

zájmena a číslovky mohou být skloňována. Dále v češtině rozlišujeme tři jmenné rody (mužský, ženský a střední) a dvě čísla (jednotné a množné). U sloves navíc rozlišujeme jejich vid.

3.4 Problematika tvarosloví

První problém související s zpracováním přirozeného jazyka je vypořádání se s tvaroslovím (morfologií). Jednotlivá slova se v českých textech objevují v různých slovních tvarech, málokdy jsou všechna slova v základním tvaru. Například jeden produkt může mít název **Alex Čistič na laminát s pomerančovým olejem** i **Alex čistič na laminát pomerančový olej**. V obou případech je uveden **pomeranč**, jen v různém tvaru. Při vyhledávání však chceme docílit toho, aby byla obě slova vyhledatelná shodně, ať už bude slovo pomeranč zadáno v kterémkoli pádě. Typické řešení je vřazení modulu do procesu indexace dokumentů nebo dotazů, který sjednocuje podobu zadaných slov.

3.4.1 Základní pojmy tvarosloví

Nejprve vysvětlím základní pojmy rozebíraného oboru, jejichž znalost je nutná k pochopení následujícího textu.

Morfém

Morfém je minimální funkční jednotka s vlastním významem. Jde o základní jednotku vědy, která se zabývá tvorbou slov, tedy morfologie.

Lexém

Lexém je základní stavební jednotka slovníku, znaková jednotka vyjadřující pojem nesoucí význam. Je to základní jednotka vědy zabývající se slovní zásobou – lexikologie. Ta je stejně jako morfologie odvětvím lingvistiky. Každý lexém je pak možné dále dělit na koncovku, kmen, kořen, prefix a sufix.

Foném

Foném je hláska umožňující rozlišit význam. Obory, pod které spadá, se zabývají zvukovou stránkou jazyka a nazývají se fonetika a fonologie.

Flexe

Flexe neboli ohýbání popisuje změnu slov vzniklou při skloňování, časování nebo ohýbání. Jde o grafickou změnu lexému pro vyjádření odpovídajícího pádu, rodu a dalších gramtických kategorií.

3.4.2 Operátor pravostranného rozšíření

Nejjednodušším řešením problematiky ohýbání slov je operátor pravostranného rozšíření. Mějme slovo **svíčka**, které se při skloňování v jednotném čísle mění na svíčky, svíčce, svíčku, svíčko, svíčce, svíčkou. V tomto příkladu lze vypozorovat, že se mění pouze konec slova, zatímco levá část **svíč-** zůstává stejná. Tento přístup bude sice poměrně úspěšný, nicméně se nevyhneme případům, kdy bude mít více slov stejný základní tvar. Mohli bychom tak při vyhledání výrazu **svíčk** obdržet i dokumenty obsahující slovo **svíčková**.

Vzhledem k jednoduchosti a zároveň poměrně vysoké úspěšnosti bývá tento přístup často využíván jako jedna z prvních implementací plnotextového vyhledávání přímo v relační databázi pomocí operátoru `LIKE`. Konkrétně vyhledávání v jazyce SQL výše diskutovaného výrazu by bylo provedeno jako `LIKE 'pomeranč%'`.

Kromě problému s možným shodným základem pro různá slova je další komplikací při použití této metody skutečnost, že při ohýbání slov nedochází pouze ke změně koncovek, ale často se mění také některá písmena základu slova. Uvažujme slovo **kůň**, které má v druhém pádu podobu **koně**. Pro tento případ by bylo možné rozšířit možnost pravostranného rozšíření o nahrazení pouze jednoho znaku. Pokud bychom tak měli pro nahrazení více písmen využít znak ***** a pro nahrazení právě jednoho písmene znak **?**, byl by základní tvar tohoto slova **k?ň***. To je ale problém, protože takový výraz odpovídá například i slovu **kaňka**.

3.4.3 Derivátor slovních tvarů

Derivátor slovních tvarů je nástroj, který namísto daného slova vygeneruje všechny jeho gramatické tvary, případně jeho použitelné odvozeniny. Ty jsou následně použity jako jejich logická disjunkce. Například místo slova `svíčka` by se použilo:

```
(svíčky OR svíčce OR svíčku OR svíčko OR svíčkou
  OR svíček OR svíčkám OR svíčkami OR svíčkách)
```

Takový nástroj musí obsahovat co nejpřesnější sadu pravidel, jak vytvářet veškeré tvary zadaného slova. Dále je třeba k expandovanému slovu zadat jeho veškeré informace, aby mohlo být vybráno správné pravidlo.

3.4.4 Stematizace

Stematizace je proces, kdy je slovo převáděno na jeho kmen (stem). Toho je docíleno odstraněním předpon, přípon a koncovek. Její použití je vhodné pro jazyky, které mění slova jen tímto způsobem (např. Angličtina). Pro češtinu je však stematizace hůře aplikovatelná, protože se slova ohýbají podle složitějších pravidel, ne jen změnou předpon, přípon nebo koncovek. Stematizaci tedy lze využít při hledání základního tvaru slova i v českém jazyce, je ale nutné ji rozšířit o další procesy, které jsou popisovány dále.

3.4.5 Lematizace

Lematizátor je nástroj, který převádí slovo na jeho základní gramatický tvar, tzv. lemma. Základním gramatickým tvarem se roumí první pád jednotného čísla u podstatných jmen nebo infnitiv u sloves. Celý proces se nazývá lematizace a kromě převodu na základní tvar může lematizátor disponovat doplňkovou funkcí, kdy pro daný tvar poskytne i jeho vlastnosti, například slovní druh, pád nebo číslo. Lematizátor pracuje v podstatě opačně ve srovnání s derivátorem. Lematizace zároveň není totéž co stematizace, které předpokládá, že dochází jen k přidávání předpon a přípon. Lematizátor jej vlastně rozšiřuje, odstranění předpon nebo přípon však může být jednou jeho z činností. Především ale počítá s tím, že se koncovky nebo kmeny slov mohou měnit, což je důležité pro češtinu. Dalším rozdílem je výsledek daného procesu – po stematizaci může vzniknout díky ořezání předpon a přípon neexistující slovo, výstupem lematizace je však vždy slovo existující.

Algoritmů pro implementaci lematizátoru je více, triviálním řešením může být použití slovníku, který obsahuje veškerou slovní zásobu daného jazyka a všechny tvary těchto slov. V něm je pak nalezeno slovo v daném tvaru a k němu odpovídající základní tvar. Výhodou takového přístupu je přesnost lematizace, pokud je již daný tvar slova znám, je triviální najít přesný základní tvar. Jediný problém může způsobit duplicita daného slova, kdy se musí lematizátor rozhodnout, na kteroý základní tvar

jej upraví. Nevýhodou je poté nutnost vytvoření takového slovníku, kdy je třeba popsat veškerou slovní zásobu.

Další možností je algoritmická lematizace, kdy jsou definována pravidla, podle kterých jsou slova ohýbána a ta jsou používána pro hledání základního tvaru. Využití takového přístupu znamená daleko menší soubor pravidel, podle kterých je lematizace prováděna ve srvnání s kompletním slovníkem slovní zásoby. Je tedy rychlejší takový lematizátor vybudovat od počátku a ve srovnání s použitím slovníku je velká šance, že bude správně zpracováno neznámé slovo. Pokud chybí ve slovníku, nelze rozhodnout, jak lema vytvořit. Pokud však budeme mít definovánu jen sadu pravidel, spíše bue lematizace provedena úspěšně. Zřejmou nevýhodou ve srovnání s využitím slovnku je kvalita takové lematizace. Čeština je dosti nepravidelná a existuje mnoho výjimek, které se při ohýbání slov vyskytují.

Konečně lze také pro lematizaci využít stochastických metod, respektive metod strojového učení. Ručně se vytvoří trénovací data, která se následně použijí pro vytvoření modelu. Na základě tohoto modelu lze provádět samotnou lematizaci, přičemž použitelných metod strojového učení je celá řada.

3.5 Problematika významnosti

Další problém, který je třeba řešit je významnost výrazů, podle nichž je vyhledáváno. Ne všechna slova jsou pro vyhledávání využitelná, protože nenesou žádnou užitečnou hodnotu, nebo jsou tak častá, že je jejich využití degradováno.

Zároveň však ne všechna slova charakterizují daný dokument stejnou mírou. Nelze se tedy jen rozhodovat, zda dané slovo ukládat do invertovaného indexu, je také třeba pamatovat na rozdílnou informační hodnotu slov vůči dokumentům – ať už se jedná o indexovaný dokument, nebo o celou jejich sadu.

3.5.1 Negativní slovník

Negativní slovník, někdy označovaný také jako stop-slovník nebo slovník stop-slov, je datová struktura obsahující slova, která jsou v daném případě považována za bezvýznamná z pohledu vyhledávání. Bývají to slova určitých slovních druhů, nebo obecně slova, která nenesou žádný význam. Obvykle jde o předložky, spojky nebo zájmena. Někdy také může jít o podstatná jména, která se v dané oblasti vyskytují tak často, že jsou pro vyhledávání nepoužitelná.

Kromě výše uvedených slovních druhů by také bylo možné použít některá nejčastěji používaná slova v českém jazyce [21] . Pokud se podíváme na prvních 5 nejpoužívanějších přídavných jmen (jiný, určitý, další, nový, velký), zjistíme, že právě toto jsou slova, která pravděpodobně nepůjdou pro vyhledávání dobře použít. Pro vytvoření negativního slovníku tak bude potřeba existující slovník obsahující slova potřebných slovních druhů a některá tato nejpoužívanější slova.

Další možností, která může jednoduše vyloučit spojky a předložky, je odfiltrování krátkých slov. Vzhledem k tomu, že většina takových slov má délku jeden až dva, nejvýše tři znaky, je možné to v filtru zohlednit. Hlavní výhodou tohoto přístupu je

rychlost oproti vyhledávání slova v slovníku. Je tedy optimální oba přístupy kombinovat, nejprve odfiltrovat krátká slova a poté využít slovníku stop-slov, která se tím výrazně zmenší (je možné krátká slova vypustit) a celá indexace se tím zrychlí.

3.5.2 Významnost výrazů

Důležitá veličina při indexaci výrazu je jeho významnost, neboli míra reprezentace indexovaného dokumentu. Některé výrazy totiž vystihují daný dokument přesněji než jiné. Uvažujme produkt s názvem **Jablko Idared červené**. Slovo **Jablko** tento produkt vyjadřuje poměrně dobře, ale stále ještě příliš obecně. Pokud uživatel vyhledává podle tohoto výrazu, stále ještě nelze rozhodnout, že je tento produkt ten, který hledal. Slovo **Idared** již reprezentuje konkrétní produkt daleko přesněji, žádný jiný produkt se takto pravděpodobně nejmenuje. Opakem je pak slovo **červené**, které vystihuje produkt nejméně, červené mohou být i jiné potraviny (rajčata) nebo dokonce úplně jiné produkty (například petrklíč červený).

Z uvedeného příkladu lze vypozorovat, že významnost výrazu souvisí s jeho výskytem v celém souboru indexovaných dokumentů. Přesněji jde o vztah mezi frekvencí výrazu (TF – term frequency) a frekvencí v invertovaných dokumentech (IDF – inverse document frequency) [2, strana 21]. Ty lze vypočítat pomocí vzorců:

$$\mathrm{tf}(t,d) = 0.5 + 0.5 \cdot \frac{f_{t,d}}{\max\{f_{t',d} : t' \in d\}}$$

$$\mathrm{idf}(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

Samotná váha daného výrazu (TF-IDF) je definována jako součin těchto hodnot:

$$\mathrm{tfidf}(t,d,D) = \mathrm{tf}(t,d) \cdot \mathrm{idf}(t,D)$$

Pro zpřesnění tohoto výpočtu je možné dále pracovat s váhou a to jak u frekvence výrazu, tak u frekvence v invertovaném indexu. Hodnoty bývají nejčastěji normalizovány nebo logaritmovány.

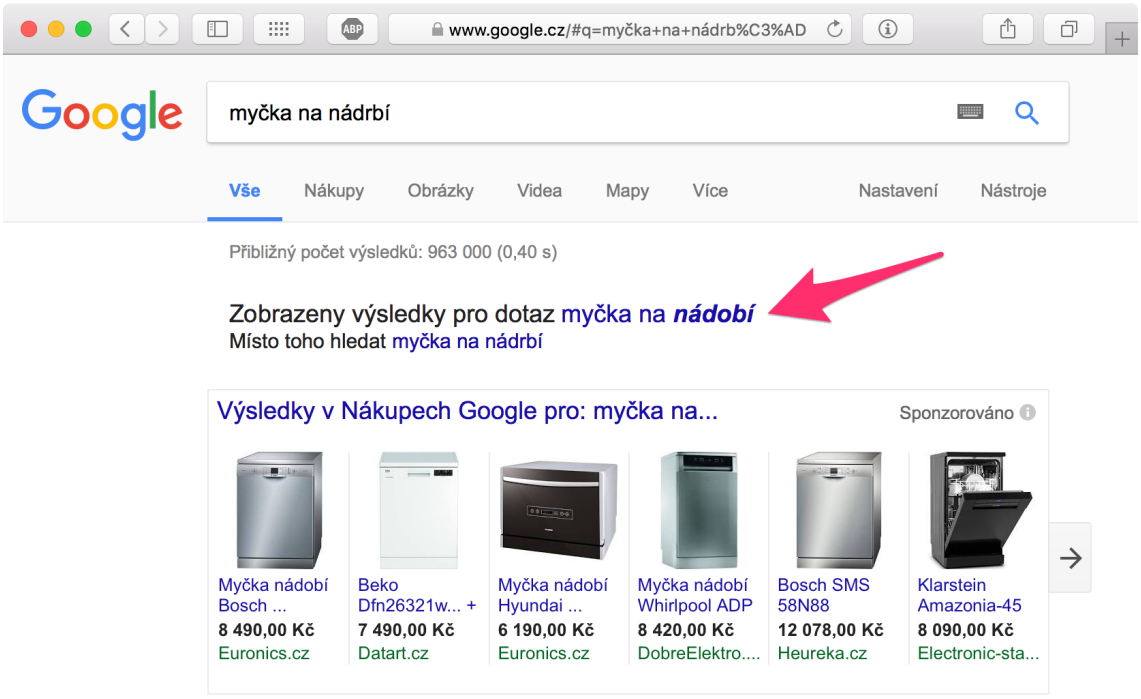
3.6 Tezaurus

Tezaurus označuje slovník, který charakterizuje vztahy mezi slovy. Zachycuje podobnost, nadřazenost nebo podřazenost mezi slovy. Lze jej využít při indexaci nebo formulaci dotazu a s jeho pomocí je možné daná slova rozšířit o slova s obdobným významem, přičemž význam může být obecnější nebo naopak přesnější. Tezaurus je díky množství vztahů které popisuje propracovanější než jen seznam synonym. Lze jej využít i pro neobvyklá slova, jako je žargón nebo další expresivní výrazy.

Vytvoření takového slovníku je závislé na oblasti, pro kterou je vytvářen. Například slovo **oko** bude nahraditelné jinými slovy v kontextu gastronomie a jinými slovy v oboru biologie. Nelze tedy bezmyšlenkovitě používat jeden takový slovník globálně pro veškeré vyhledávání, je třeba zohlednit obor, ve kterém bude využíván.

3.7 Přibližné vyhledávání

Dalším komplexním problémem, kterým je třeba se zabývat je přibližné vyhledávání, tedy vyrovnání se s překlepy nebo schopnost vyhledat dokumenty už v okamžiku, kdy je formulován vyhledávaný dotaz. Tato funkčnost je označována jako search-as-you-type nebo také incremental search.



Obrázek 3.3: Odhalení překlepu ve vyhledávání na webu google.com

U přibližného vyhledávání je třeba se vypořádat s měrou, jak přibližný daný výraz je vůči výryzu hledanému, což lze vyjádřit pomocí editační vzdálenosti. Přistoupit k této problematice lze i jednoduším přístupem – generováním n-gramů.

3.7.1 Editační vzdálenost

Editační vzdálenost je způsob, jak vyjádřit podobnost dvou textových řetězců. Je vyjádřena jako celé číslo, které udává počet operací nutný pro transformaci z jednoho textového řetězce na druhý. Editačních vzdáleností pro textové vyhledávání je více, podle toho, jaké operace na textovém řetězci umožňují.

V roce 1965 Vladimir Levenshtein definoval **Levenshteinovu vzdálenost** jako počet změn znaků vedoucí k transformaci z jednoho textu na druhý [22]:

- Nahrazení jednoho znaku jiným: **jxblko** → **jablko**
- Vložení jednoho znaku: **jaxblko** → **jablko**
- Odstranění jednoho znaku: **jblko** → **jablko**

Frederick Damerau ji později rozšířil o další transformaci, která má shodnou váhu:

- Prohození dvou znaků: **jbalko** → **jablko**

Tato transformace vlastně nahrazuje více dílčích operací definovaných výše. Vzdálenost s touto transformací o hodnotě 1 je označována jako **Damerau–Levenshteinova vzdálenost**. Počtem transformací se zvyšuje editační vzdálenost a zároveň s ní se snižuje pravděpodobnost, že daný textový řetězec je překlepem druhého řetězce. Damerau vypozoroval, že 80% překlepů má editační vzdálenost rovnou jedné [23]. Je tedy výrazně nižší pravděpodobnost, že v textu bude tolik chyb, aby byla vzdálenost vyšší, což lze zohlednit při konstrukci vyhledávacího algoritmu a bude mít zřejmě pozitivní vliv na rychlost vyhledávání.

Další vzdáleností je **Hammingova vzdálenost**, která počítá pouze s náhradou znaku v textovém řetězci. Z tohoto omezení je však patrné, že je použitelná pouze pro řetězce shodné délky. V praxi textového vyhledávání tedy budeme pracovat pravděpodobně s vzdálenostmi definovanými výše.

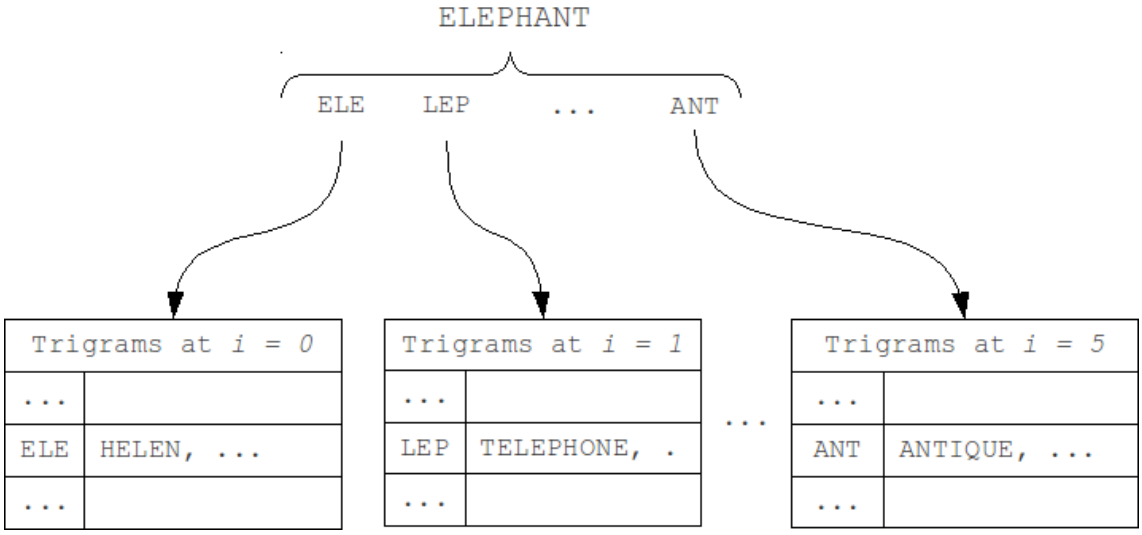
Na mírně odlišném principu funguje **vzdálenost nejdelšího společného podřetězce** (nebo také posloupnosti – LCS distance). Jejím úkolem je nalézt nejvyšší možný za sebou jdoucí počet znaků, který je společný oběma řetězcům. Tento přístup je ale méně chodný pro vyrovnýní se s chybami uprostřed slov, navíc je výpočetně náročnější.

Další z možných vzdáleností je **Jarova vzdálenost**, která kromě počtu změn na znacích počítá i počet shodných vzdáleností. Jejím rozšířením je **Jaro-Winklerova vzdálenost**, která navíc počítá s tím, že shodnost na začátku řetězce má vyšší váhu, než na jeho konci – vychází totiž z pozorování, že na začátku textu se vyskytuje méně chyb [25]. Tento přístup také dosahuje lepších výsledků u krátkých slov.

3.7.2 N-gramová podobnost

Vytváření n-gramů je používáno při indexaci, kdy jsou slova dělena na části dlouhé `n` znaků. Pokud bychom měli vygenerovat n-gramy o délce 3 (trigramy) pro slovo **telefon**, byly by to výrazy: **tel**, **ele**, **lef**, **efo** a **fon**. Lze si všimnout, že jednotlivé vygenerované výrazy se překrývají. Pokud pak uživatel zadá jako hledaný výraz **telef**, bude odpovídat trigrmům **tel**, **ele** a **lef**, tedy třem z celkem pěti. Trigramy jsou společně s digramy (n-gramy o délce dvou znaků) nejvhodnější vzhledem k obvyklé délce slov. N-gram o délce 1 pak ztrácí smysl výše nastíněné logiky.

Porovnávání slov pomocí n-gramů je znázorněno na následujícím schématu. Zde je zadané slovo rozděleno na trigramy, z nichž každý je dále porovnávám s trigramy indexovaných slov.



Obrázek 3.4: Vyhledávání pomocí n-gramů

Tento mechanismus lze použít pro našeptávání i pro odhalení překlepů. Je snadno implementovatelný a poměrně rychlý, selhává však u překlepů v krátkých slovech [24]. Předpokládejme slovo o délce 5 znaků – **mobil**. Vzniklé trigramy jsou **mob**, **obi**, **bil**. Pokud uživatel vyhledá slovo s překlepem přesně uprostřed (**movil**), nebude nalezena shoda v žádném trigramu.

Řešením by mohlo být snížení počtu znaků v n-gramu, což může na druhou stranu mít negativní vliv na rychlost. Možným kompromisem tak může být různá délka n-gramů, kdy by vznikly kratší n.gram pro začátek a konec slov. Konkrétně pro slovo **mobil** by tak vznikly n-gramy **m**, **mo**, **mob**, **obi**, **bil**, **il**, **l**.

3.8 Relevance

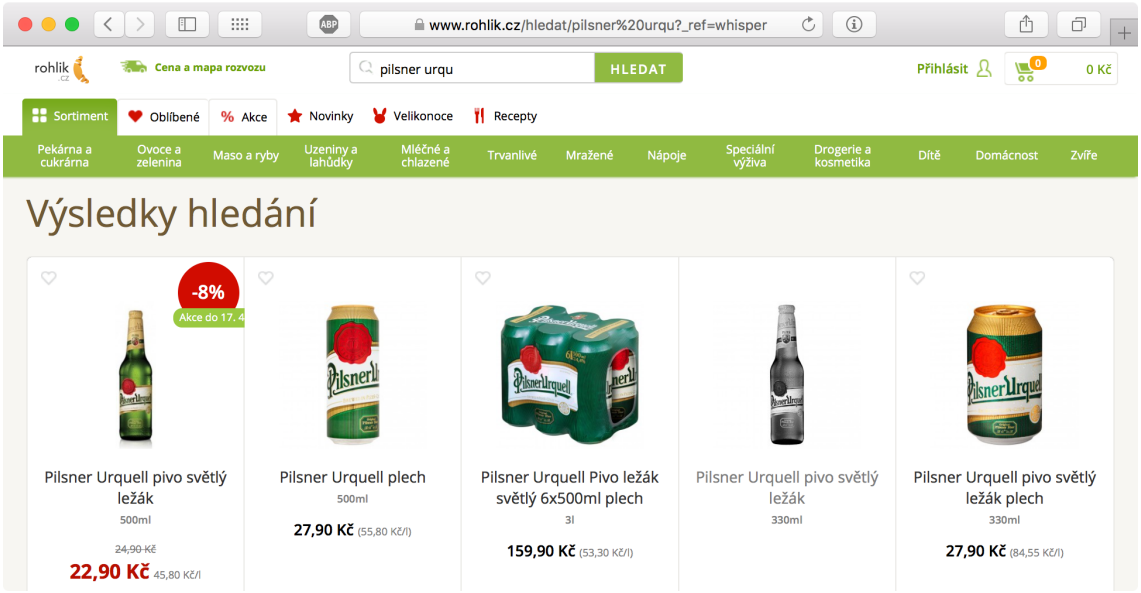
Relevancí rozumíme míru, jakou odpovídá nalezený dokument zadanému zadanému dotazu. Je použitelná pro řazení nalezených dokumentů, kdy jsou nejprve zobrazeny dokumenty více odpovídající hledanému výrazu. Dále je možné se na základě relevance rozhodnout, zda ještě nalezený dokument hledanému výrazu odpovídá, zda je vůbec třeba jej zobrazit.

Relevance je do jisté míry subjektivní, protože každý zákazník vyhledává na webu s jiným očekáváním, přestože třeba vyhledáva pomocí stejných výrazů. Objektivně lze však relevanci vyjádřit pomocí TF-IDF, tedy frekvence nalezených výrazů vůči jejich frekvenci v invertovaném indexu.

V praxi elektronických obchodů však často do řazení vstupují další veličiny, jako je cena produktu, skladovost nebo výše marže. Pokud například uživatel vyhledá

iPhone, může být žádoucí nejprve zobrazit produkt **iPhone SE 64GB Vesmírně černý**, než jeho příslušenství, například **Sportovní obal na iPhone SE** z důvodu několikanásobně vyšší marže na tomto produktu.

Elektronické obchody často také operují s akčním zbožím. Zejména v České republice je obchodování se zlevněným zbožím populární, může bý tak žádoucí zobrazit takové produkty dříve, než produkty prodávané za standardní cenu.



Obrázek 3.5: Řazení výsledků – upřednostnění akčního zboží

Pořadí výsledků nemusí být ovlivňováno jen zájmy provozovatele, mohou jej formovat sami zákazníci svým chováním. Konkrétně v případě vyhledávání lze pozorovat proklikovost jednotlivých produktů vzhledem k zadanému dotazu. Pokud se ukáže, že na základě daného dotazu zákazníci nejčastěji klikají na konkrétní produkt, je možné jej zobrazit ve výsledcích vyhledávání pro tento dotaz dříve.

S tímto přístupem je však možné jít ještě dál a začít vytvářet uživatelské profily zákazníků podle toho, o jaké zboží se zajímají, co vyhledávájí, jaké jsou jejich údaje zadané při registraci a objednávce (lokalita, pohlaví), nebo v jaké časy nakupují prostřednictvím jakých zařízení. Tato problematika je již složitější, nicméně v posbíraných datech a vytvořených uživatelských profilech lze hledat další souvislosti pomocí statistických metod a strojového učení. Konkrétně lze zákazníka zařadit do určité skupiny pomocí shlukové analýzy a podle toho k němu přistupovat (upravit řazení výsledků vyhledávání) nebo využít algoritmů doporučovacích systémů.

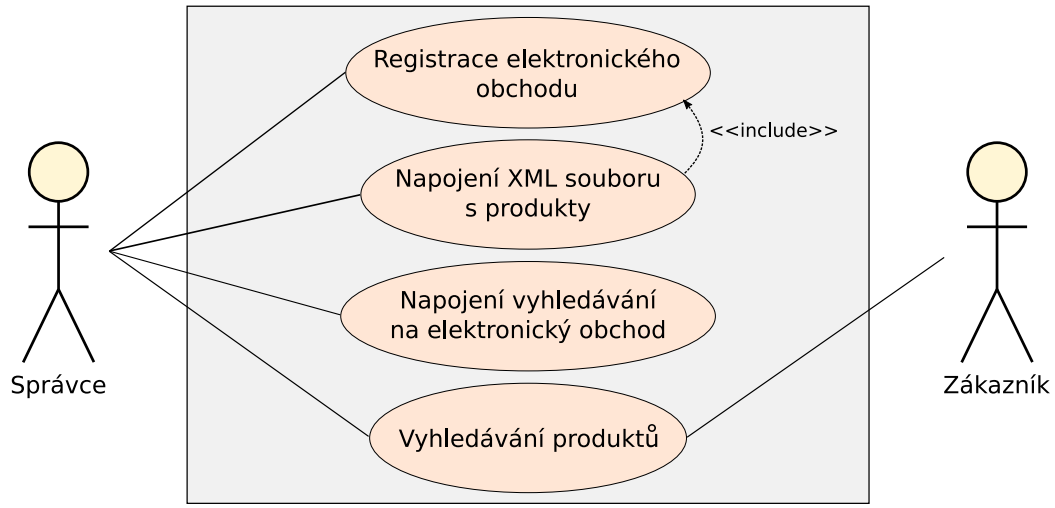
Samotné řazení všech uložených dokumentů může být výpočetně náročné, v tu chvíli lze řadit dokumenty ve dvou průchodech. V prvním průchodu jsou nalezeny dokumenty odpovídající zadanému dotazu a případně jřazeny s pomocí nenáročného výpočtu. Poté lze provést druhé řazení nad výběrem prvních dokumentů. Díky tomu, že už probíhá řazení nad malou množinou dat, je možné využít výpočetně náročnějších algoritmů.

4 Návrh řešení

V této kapitole vytvářím návrh aplikace umožňující implementaci plnotextového vyhledávání do elektronického obchodu, které bude možné provozovat jako službu. Popisuji zpsob, jakým bude aplikace používána, aktéry, kteří v užívání aplikace vystupují a vytvářím její model, který bude požadavkům nejlépe odpovídat.

4.1 Definice případů užití

Aplikaci používají dva typy uživatelů. Správce (provozovatel elektronického obchodu), který zajišťuje její konfiguraci (například napojení vstupních dat) a zákazník elektronického obchodu, který provádí samotné vyhledávání. Z toho vyplývá, že ty části aplikace, které používá správce musí být zabezpečeny, aby k nim neměl přístup nikdo jiný. Navíc musí aplikace umět pracovat s více správci zároveň, kdy má každý přístup pouze ke svým datům. Pro zákazníka musí být naopak jeho část aplikace veřejně přístupná, musí být také co nejrychlejší. Právě zákazník bude provádět vyhledávání a rychlost, s kterou obdrží výsledky je proto klíčová.



Obrázek 4.1: Případy užití aplikace

Toto jsou základní případy užití z nejméně podrobného pohledu. Jednotlivé případy užití popisuji podrobně níže.

4.1.1 Registrace do aplikace

Toto je první případ užití a jeho splnění je předpokladem pro další případy užití. Budoucí správce, který chce aplikaci používat musí mít vytvořen účet, který je přístupný po zadání jeho přihlašovacích údajů.

- Správce zobrazí registrační formulář
- Správce zadá údaje pro svůj účet (e-mailová adresa a heslo)
- Správce zadá adresu souboru XML s produkty a odešle formulář
- Správce je přihlášen pod novým účtem, je spuštěn import produktů

Výsledkem tohoto případu je vytvořený nový účet. Při ukládání systém kontroluje unikátnost zadaného uživatele a na základě zadané URL produktového XML vytváří novou konfiguraci daného uživatele a na pozadí spouští import produktů.

4.1.2 Napojení XML souboru s produkty

Tento případ užití lze jednak provést samostatně ve chvíli, kdy je správce přihlášený do aplikace, je ale také dílčí operací již při registraci.

- Správce zobrazí formulář pro zadání URL produktového XML
- Správce zadá adresu souboru XML s produkty a odešle formulář
- Je spuštěn import produktů na pozadí, je zobrazena informace správci

Po provedení tohoto případu užití je aktualizována (nebo vytvořena pokud neexistuje) konfigurace daného uživatele. Zároveň je na pozadí spuštěn import produktů, aby byla zajištěna aktuálnost produktové databáze.

4.1.3 Napojení vyhledávání na elektronický obchod

Ve chvíli, kdy má správce v aplikaci uložené produkty, může již využít samotné vyhledávání, které je poskytované jako služba. Toto napojení musí být maximálně intuitivní.

- Správce se přihlásí do aplikace
- Správce přejde na stránku zobrazující manuál k implementaci napojení
- Správce upraví elektronický obchod tak, aby byly požadavky na vyhledávání směrovány na URL uvedenou v manuálu

4.1.4 Vyhledávání produktů

Nejčastějším případem užití je vyhledávání prováděné zákazníkem. To je možné až ve chvíli, kdy je implementovaný jak import produktů, tak samotné vyhledávání.

- Uživatel v elektronickém obchodu začne psát dotaz do textového pole pro vyhledávání
- Při každém napsaném nebo smazaném písmenu je odeslán HTTP požadavek na API vyhledávání
- Pro každá odeslaný požadavek je aktualizován výpis nalezených produktů a tento požadavek
  je na serveru zaznamenán

Vzhledem k důležitosti tohoto případu je nutné aby bylo vyhledávání rychlé a stabilní. Každé zpoždění nebo výpadek totiž snižuje důvěru klienta používající vyhledávání.

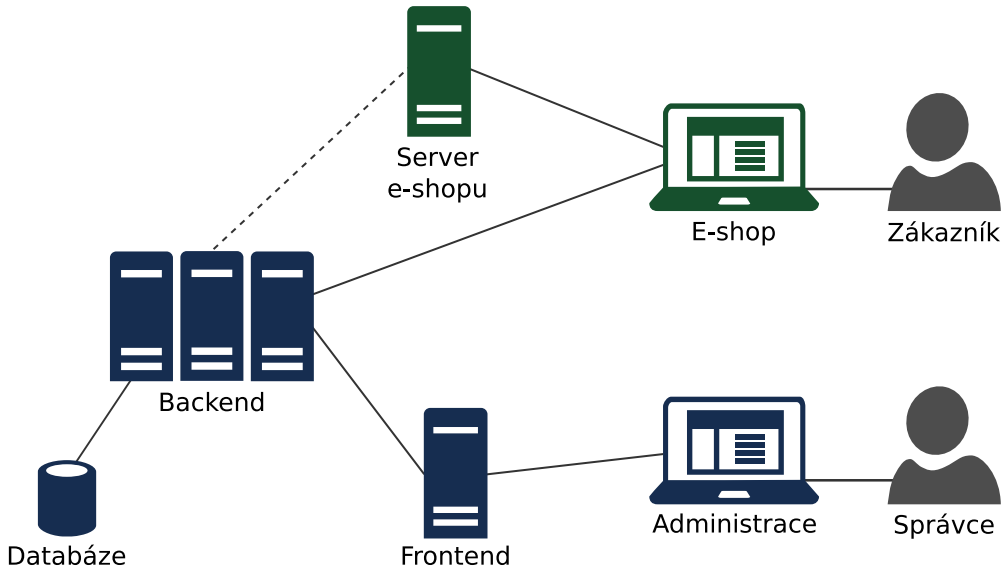
4.2 Návrh architektury aplikace

S aplikací bude komunikovat správce prostřednictvím grafického webového rozhraní a dále zákazník, který s aplikací komunikuje prostřednictvím grafického rozhraní elektronického obchodu. Ten musí s vyvíjenou aplikací komunikovat prostřednictvím jasně definovaného a jednoduše implementovatelného API.

Z tohoto důvodu se jeví jako výhodné rozdělit aplikaci na dvě části – první část (**backend**) bude obstarávat veškerou logiku, perzistenci dat a bude s ní možné komunikovat prostřednictvím API. Druhou částí (**frontend**) by byla samostatná webová aplikace s grafickým rohraním, která sama o sobě nepracuje s žádným úložištěm, ale všechny tyto požadavky nechává vyřídit backend [26]. Rozdělení na frontend a backend s sebou nese řadu dalších výhod – frontend bůže být vyvíjen v jazyce, který je pro to vhodnější, na obou částech můžou v budoucnu pracovat specializovaní vývojáři, obě aplikace je možné nezávisle škálovat. Zpřístupněním celého API navíc mohou vývojáři implementovat celé napojení plně automatizovaně.

Dále je třeba, aby byla architektura navržena tak, aby bylo možné aplikaci snadno škálovat. To se týká databáze i samotné aplikace dostupné prostřednictvím API. Databázi je tedy vhodné volit takovou, která umožní také horizontální škálování [28]. Dá se předpokládat, že s rostoucím počtem uživatelů služby bude rovnoměrně růst i potřeba na objem zpracovávaných dat. S horizontálním škálováním souvisí také možnost vytváření replik dat v úložišti, v takovém případě by byl cluster schopný se vyrovnat s případnou ztrátou některých dat. S možností snadného vertikálního škálování aplikace je třeba počítat při jejím nasazování. Prostředí, do kterého bude aplikace nasazována by mělo být schopné aplikaci škálovat obdobně jako databáze. Díky tomu, že veškerá data jsou persistována v databázi, je možné průběžně zapínat a vypínat jednotlivé instance aplikace bez obavy o ztrátu dat.

Na následujícím obrázku je znázorněn vztah mezi jednotlivými komponentami aplikace včetně napojovaného elektronického obchodu. Dále jsou zobrazeni oba aktéři.



Obrázek 4.2: Architektura aplikace – propojení dílčích komponent

Ze schématu je zřejmé, že zákazník si zobrazuje elektronický obchod, který je zobrazen na základě dat z serveru obchodu. Data pro vyhledávání jsou načtena prostředníctvím API aplikce, přičemž požadavek může být odeslán přímo z webového prohléžeče zákazníka (asynchronní HTTP požadavek) nebo prostředníctvím serveru obchodu (pokud probíhá vykreslení HTML stránky na serveru). Z druhé strany s aplikací operuje správce, který ji ovládá skrze administraci, která disponuje grafickým webovým rozhraním. Ta však veškeré požadavky směřuje na backend. Konečně backend používá pro persistenci dat (produktů i konfigurací) databázi, ostatním službám je pak dostupný skrze API.

4.3 Datový model

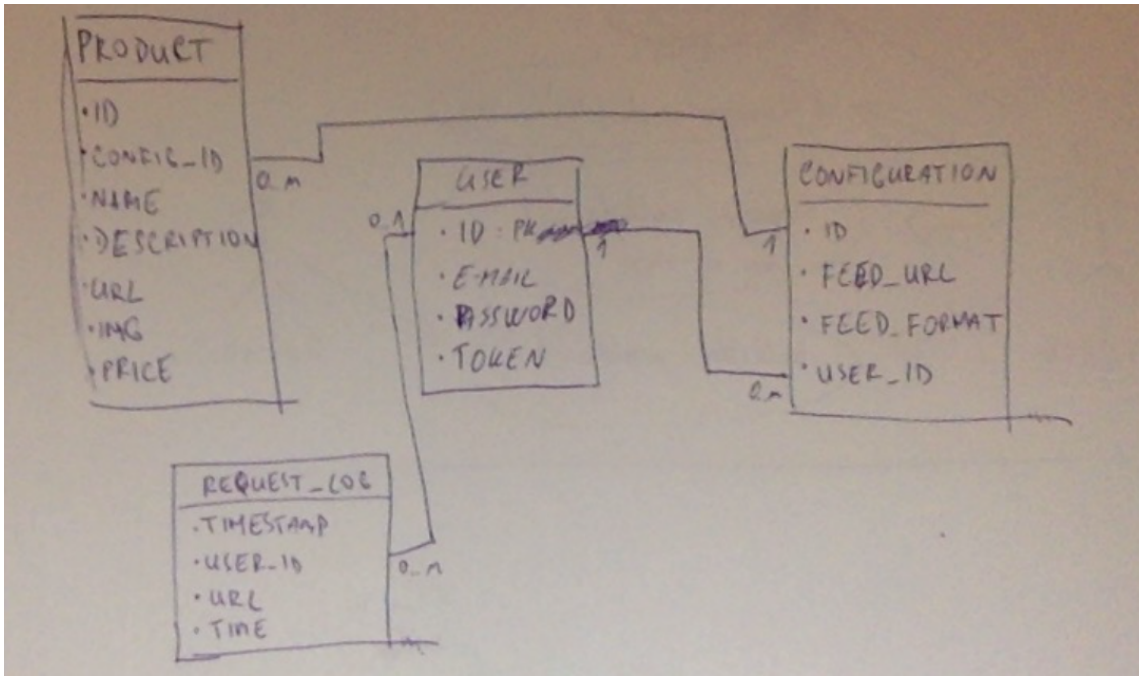
Nyní definuji entity, které v aplikaci vystupují, jejich atributy a vztahy mezi nimi. V systému je primárně definován **uživatel** (správce), který se do aplikace registroval a vytváří konfiguraci pro import produktů. Pro každou tuto konfiguraci jsou do systému importovány produkty, jejichž struktura byla rozebrána v analýze dat, v kterých bude vyhledáváno.

U uživatele je zvolen jako jeho přirozený jednoznačný identifikátor jeho e-mailová adresa, kterou zadává při registraci. Dále je třeba uložit jeho heslo (respektive jeho hash) pro umožnění přihlášení. Posledním atributem uživatele je náhodně vygenerovaný token, který bude využit při přístupu k API klientem. Je totiž třeba zajistit,

aby nebylo možné triviálním způsobem procházet produkty dalších uživatelů [5, strana 16].

Uživatel si dále vytváří konfiguraci, ve které je uložena adresa s XML souborem obsahujícím produkty, informace o jeho formátu (v případě že bude podporováno více formátů, je to vhodné pro detekci formátu a použití vhodného parseru). Tato **konfigurace** náleží právě jednomu uživateli, v jiných případech nemá její existence význam.

Další entitou, s kterou je třeba pracovat je **produkt**. Právě ten bude vyhledáván. Jeho atributy v nejjednodušší podobě jsou jeho jednoznačný identifikátor nebo kód (v rámci daného obchodu), název, popis, URL, adresa obrázku a cena. Produkt je možné rozšířit o další atributy, nicméně toto jsou atributy, které by měly být u každého produktu známy.



Obrázek 4.3: Datový model aplikace

Poslední entitou znázorněnou v diagramu je zalogovaný požadavek. Vzhledem k tomu, že chceme mít jednak kontrolu nad rychlostí vyhledávání, tak je dobré mít přehled o využití aplikace ať už jako celku, tak vzhledem k jejímu využití jednotlivými obchody. Každý takový záznam musí obsahovat datuma čas, kdy byl zaznamenán, URL, na kterou byl požadavek odeslán a čas, který trvalo jeho vyřízení.

4.4 Návrh API

Klíčový je návrh API, jehož prostřednictvím je dostupná veškerá logika aplikace. Při jeho návrhu jsem se rozhodl využít přístupu design-first [27], kdy je nejprve navrženo

samotné API (vytvořena jeho dokumentace) a až poté je přistoupeno k samotné implementaci. Pro jeho návrh použiji nástroj Apiary [6], který umožňuje pracovat s API pomocí tohoto přístupu.

Před konkrétním návrhem je třeba rozmyslet protokol a další principy, kterých API využívá. Přestože by se mohlo zdát, že to je až záležitost samotné implementace, tento krok zásadně ovlivňuje již způsob návrhu API. Dále je to důležité k použití přístupu design-first.

Pro naplnění požadavků vycházejících z informací o problematice elektronického obchodování je třeba volit takový protokol, který je běžně dostupný napříč všemi běžnými webovými prohlížeši a je možné jej využít v používaných webových technologiích. Takovým protokolem je HTTP (případně novější verze HTTP/2). V Nejnvějších prohlížečích by sice bylo možné používat pokročilejší protokoly (například WebSockets [30]), aktuálním cílem je však maximální možná kompatibilita a snadnost implementace. Samotná komunikace probíhá dle principu REST, data jsou zasílána ve formátu JSON, což jsou nejpřímočařejší způsoby návrhu a implementace [5]. Bylo by možné využít jiných mechanismů (XML-RPC nebo SOAP), nicméně pro tento konkrétní případ jsou méně vhodné vzhledem k jejich snadnosti implementace, pochopitelnosti a rychlosti [5, strana 31].

4.4.1 Datové struktury

Nejprve je třeba identifikovat zdroje (resources), což jsou datové struktury, které budou prostřednictvím API předávány. Dále je třeba definovat operace, které lze nad jednotlivými zdroji provádět (čtení, vytváření, mazání, aktualizace) a nakonec také vhodně navrhnout koncové body (endpoints) [5, strana 12], pod kterými bude každá operace dostupná.

Základní zdroje, s kterými API operuje jsou **produkt** a **konfigurace uživatele**. Tyto struktury nesou důležitou informační hodnotu, první veškeré informace o produktu, které může být třeba v elektronickém obchodu zobrazit, druhá pak informace o nastavení daného uživatele. Tyto struktury lze popsat pomocí syntaxe API Blueprint [5, strana 131] následovně:

```
# Data Structures

## Product (object)
- id: `1mWa9` (string, required) - ID of product
- userId: `01dsdmmwv2dwkdlam8di8` (string, required) - User ID
- name: `Postel bílá` (string, required) - Name of product
- description: `Postel vyrobena z masivu...` (string, required) -
 ↪  Product description
- url: `http://shop.tld/1mWa9` (string, required) - detail URL
- img: `http://shop.tld/1mWa9.jpg` (string, required) - image URL
- price: `1` (number, required) - Current price of product incl. VAT
- updated: `2016-12-04T23:22:16+01:00` (string, optional) -
 ↪  Timestamp of last update
```

```
## UserConfig (object)
- id: `01dsdmmwv2dwkdlam8di8` (string, required) - User ID
- token: `7Tnl6xPQhVPiv8Oryboo4iTckqXVuG4xjWFXTC1mzU3EaGY5`
 ↪  (string, required) - security token for API access
- email: `user@email.com` (string, required) - e-mail address
- password: `p4ssw0rd` (string, optional) - new password
- feedUrl: `http://shop.tld/feed.xml` (string, required) - XML feed
- feedFormat: `heureka` (enum[string], required)
    - `heureka`
```

Zde struktura `Product` bude používána pouze pro čtení, produkty jsou vytvářeny na základě importovaného XML souboru. Konfigurace uživatele je používána jak při registraci (vytvoření nové konfigurace), tak při aktualizaci této kanfigurace (čtení i změna).

Dalšími pomocnými strukturami jsou data pro provedení **přihlášení** správce do administrace. Ten při přihlášení zadává přihlašovací jméno a heslo, jako odpověď obdrží své ID a dále token, identifikující jeho sezení a použitelný pro vykonání dalších požadavků.

```
## LoginRequest (object)
- email: `user@email.com` (string, required) - e-mail address
- password: `p4ssw0rd` (string, required) - password

## LoginResponse (object)
- userId: `01dsdmmwv2dwkdlam8di8` (string, required) - id of user
- token: `JT0n0rws8rRMx8P205hmc68c9yptDW91PkuLNcHe2J2NxQdyO`
 ↪  (string, required) - auth token
```

Poslední struktury, které API používají, definují formát odpovědi pro operace, které neočekávají žádná data, pouze je modifikují (mutace). Po provedení takových operací API informuje o úspěšnosti jejich provedení odpovídajícím stavovým kódem. V případě neúspěchu přidává srozumitelnou hlášku včetně detailního popisu o důvodu neúspěchu, což usnadňuje práci vývojářům napojovaných aplikací a odstraňování chyb [5, strana 42].

```
## Ok (object)
- status: `200` (number, required) - HTTP status code
- message: `User created` (string, optional) - Result message

## Error (object)
- status: `400` (number, required) - HTTP status code
- message: `Unable to create user` (string, required) - Error
 ↪  message
- description: `Email already exists` (string, optional) -
 ↪  Additional description to message
```

4.4.2 Koncové body

Pro každou operaci je definován kocnový bod (endpoint) a metoda, pomocí které je operace dostupná, přičemž koncový bod je URL, pod kterou je dostupný. Ta je uváděna relativně vzhledem k doméně, pod kterou je aplikace provozována. Dále jsou všechny mají adresy prefix `/api/v1`, který usnadní možné budoucí aktualizace, které nejsou zpětně kompatibilní.

Koncové body jsou pojmenovány podle zdrojů, s kterými pracují, aby bylo jejich používání intuitivní. Prvními jsou body pro provedení přihlášení a odhlášení, které operují s zdroji **LoginRequest** a **LoginResponse**, respektive **Ok** a **Error**.

- `POST /api/v1/sign/in` – přihlášení do administrace
- `POST /api/v1/sign/out/{token}` – odhlášení z administrace

Další množina bodů provádí operace se zdrojem **UserConfig**. Slouží pro registraci nového uživatele, aktualizaci údajů stávajícího uživatele, zobrazení registračních údajů a případnou deaktivaci účtu stávajícího uživatele.

- `GET /api/v1/user/{token}` – získání konfigurace uživatele
- `PUT /api/v1/user/{token}` – vytvoření nebo aktualizace konfigurace
- `DELETE /api/v1/user/{token}` – deaktivace uživatele

Posledním koncovým bodem je vyhledávání produktů. Tento bod je jediný veřejný a použitý v elektronickém obchodě. Tento bod přejímá dva paramtery, kterými jsou ID uživatele (obchodu) a hledaný výraz. Data jsou získávána jako pole objektů **Product**, které je ještě zanořeno v dalším poli, kde je dostupné pod klíčem `products`. To usnadní budoucí možné rozšiřování o další data (výsledky agregačních dotazů nebo například stav stránkování) s zachováním zpětné kompatibility.

- `GET /api/v1/search/{userId}/{query}` – vyhledání produktů dle `query`

Tento koncový bod musí být v administraci aplikace zadokumentován a uveden s názornými příklady, jak jej napojit do elektronického obchodu na základě konkrétní použité technologie.

Celé API je dokumentováno v formátu API Blueprint a je distribuováno společně s zdrojovými kódy aplikace, které jsou přílohou této práce. Současně je vytvořen projekt v nástroji Apiary, ve kterém je dokumentace uložena [31]. Ta obsahuje i vzorová data, je tedy možné využít serverů proxy, které umožňují přistupovat k API, které vrací ukázková data. Je tedy možné začít nezávisle na sobě navrhovat a vyvíjet stranu, která API poskytuje (backend) i konzumuje (frontend). Vytvořený soubor ve formátu API Blueprint budiž smlouvou, která mezi těmito aplikacemi vznikla.

5 Implementace

- Vyber technologii, nastroju, frameworku; programovani, testovani, deployment...

5.1 Výběr nástroje pro vyhledáváni

- Popis a porovnani vhodnych nastroju
  - MySQL
  - Elasticsearch
  - Sphinx
  - Postgres
  - ...

5.2 Nastavení a nasazení Elasticsearch

- Implementace indexace
- Implementace vyhledavani

5.3 Výběr nástroje pro backend a frontend

- Popis a porovnani vhodnych nastroju
  - Java
  - Go
  - PHP
  - ...

5.4 Backend

- Popis trid, rozhrani, implementacni detaily

5.5 Frontend

- Frontend: UI, komunikace s backendem

5.6 Deployment

- Deployment – CI, Docker

5.7 Testování, ověření

- Overeni funkcnosti a prinosu
- Testovani, zda jsou splnena akceptacni kriteria
- Porovnani s stavajicim resenim vyhledavani na konkretnim priklade
- Metriky, tabulky, grafy...

6 Závěr

V této diplomové práci jsem...

6.1 Dosažení vytčených dílů

...

6.2 Diskuze možného budoucího rozšiřování

...

Literatura

[1] SUCHÁNEK Petr. *E-commerce*. Vyd. 1. Praha: Ekopress, s.r.o., 2012 144 s. ISBN 978-80-86929-84-2.

[2] STROSSA, Petr. *Počítačové zpracování přirozeného jazyka*. Vyd. 1. Praha: Oeconomica, 2011 316 s. ISBN 978-80-245-1777-3.

[3] AYSE Göker a DAVIES John. *Information Retrieval: Searching in the 21st Century*. Vyd. 1. Library of Congress Cataloging-in-Publication Data, 2009 295 s. ISBN: 978-0-470-02762-2.

[4] AGGARWAL Charu C., ZHAI ChengXiang. *Mining Text Data*. Springer New York Dordrecht Heidelberg London, 2000 522 s. ISBN 978-1-4614-3222-7.

[5] STURGEON Phil. *Build APIs You Won't Hate*. Vyd. 1. Philip J. Sturgeon, 2015 188 s. ISBN 978-0692232699.

[6] Apiary. *Platform for API Design, Development & Documentation* [online]. 2017 [cit. 2017-03-11]. Dostupné z: https://apiary.io.

[7] KENNEDY William. *Go in Action*. Manning Publications Co, 2016 264 s. ISBN 978-1-6172-9178-4.

[8] CLINTON Gormley, ZACHARY Tong. *Elasticsearch: The Definitive Guide*. O'Reilly Media, 2015 724 s. ISBN 978-1-4493-5854-9.

[9] Elasticsearch. *Elasticsearch Reference* [online]. 2017 [cit. 2017-03-03]. Dostupné z: https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html

[10] Algolia. *Hosted cloud search as a service* [online]. 2017 [cit. 2017-03-03]. Dostupné z: https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html

[11] Swiftype. *Site Search by Swiftype* [online]. 2017 [cit. 2017-03-30]. Dostupné z: https://swiftype.com/site-search

[12] AWS. *Amazon CloudSearch* [online]. 2017 [cit. 2017-03-30]. Dostupné z: https://aws.amazon.com/cloudsearch/

[13] Google. *Custom Search Engine* [online]. 2017 [cit. 2017-03-30]. Dostupné z: https://cse.google.com/cse/

[14] GigaSpaces Blog. *Amazon found every 100ms of latency cost them 1% in sales* [online]. 2017 [cit. 2017-03-26]. Dostupné z: https://blog.gigaspaces.com/amazon-found-every-100ms-of-latency-cost-them-1-in-sales/

[15] StackShare, Inc. *How Algolia Reduces Latency For 21B Searches Per Month* [online]. 2017 [cit. 2017-04-05]. Dostupné z: https://stackshare.io/algolia/how-algolia-reduces-latency-for-21b-searches-per-month

[16] PostgreSQL: Documentation. *Full Text Search* [online]. 2017 [cit. 2017-03-26]. Dostupné z: https://www.postgresql.org/docs/9.5/static/textsearch.html

[17] The Apache Software Foundation. *Apache Lucene* [online]. 2017 [cit. 2017-03-26]. Dostupné z: https://lucene.apache.org

[18] Sdružení pro internetový rozvoj. *NetMonitor* [online]. 2017 [cit. 2017-04-01]. Dostupné z: http://www.netmonitor.cz

[19] Heureka Shopping s.r.o. *Specifikace XML souboru* [online]. 2017 [cit. 2017-04-01]. Dostupné z: https://sluzby.heureka.cz/napoveda/xml-feed/

[20] Seznam.cz, a.s. *Specifikace XML feedu pro internetové obchody* [online]. 2017 [cit. 2017-04-01]. Dostupné z: https://napoveda.seznam.cz/cz/zbozi/specifikace-xml-pro-obchody/specifikace-xml-feedu/

[21] TĚŠITELOVÁ Marie. *O češtině v číslech*. Praha: Academia, 1987 205 s.

[22] Elasticsearch. *The Definitive Guide - Fuzziness* [online]. 2017 [cit. 2017-04-08]. Dostupné z: https://www.elastic.co/guide/en/elasticsearch/guide/current/fuzziness.html.

[23] DAMERAU, Fred J. *A Technique for Computer Detection and Correction of Spelling Errors*. Communications of the ACM (ACM), 1964, s. 171–176. ISSN 0001-0782. doi: 10.1145/363958.363994. Dostupné z: http://doi.acm.org/10.1145/363958.363994

[24] SMETANIN Nikita. *Fuzzy string search* [online]. 2017 [cit. 2017-04-08]. Dostupné z: http://ntz-develop.blogspot.cz/2011/03/fuzzy-string-search.html.

[25] CHRISTEN, Peter. *A Comparison of Personal Name Matching: Techniques and Practical Issues*. ICDMW '06 Proceedings of the Sixth IEEE International Conference on Data Mining, 2006, s. 290-294, IEEE Computer

Society Washington, DC, USA. doi: 10.1145/363958.363994. Dostupné z: http://users.cecs.anu.edu.au/~Peter.Christen/publications/tr-cs-06-02.pdf. ISBN: 0-7695-2702-7.

[26] HUS Maarten. *The case for separating front- and back-end* [online]. 2014 [cit. 2017-04-11]. Dostupné z: http://dontpanic.42.nl/2014/10/the-case-for-separating-front-and-back.html.

[27] Apiary. *How to Build an API* [online]. 2017 [cit. 2017-04-11]. Dostupné z: https://apiary.io/how-to-build-api.

[28] IBM. *How to explain vertical and horizontal scaling in the cloud* [online]. 2014 [cit. 2017-04-11]. Dostupné z: https://www.ibm.com/blogs/cloud-computing/2014/04/explain-vertical-horizontal-scaling-cloud/.

[29] API Blueprint. *A powerful high-level API description language for web APIs* [online]. 2017 [cit. 2017-04-12]. Dostupné z: https://apiblueprint.org.

[30] Mozilla Developer Network. *WebSockets – Web APIs* [online]. 2017 [cit. 2017-04-12]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API.

[31] Apiary. *Apisearch – Full text search service* [online]. 2017 [cit. 2017-04-12]. Dostupné z: http://docs.apisearch.apiary.io/.

Rejstřík

...

Terminologický slovník

Pojem - zkratka - popis

A Obsah přiloženého DVD

- Soubor Diplomova_prace_2017_Ludek_Vesely.pdf
  - Text diplomové práce