Univerzita Pardubice

Fakulta ekonomicko-správní
Ústav systémového inženýrství a informatiky

Vyhledávání zboží v nabídkách e-shopů

Bc. Ondřej Zápotočný

Diplomová práce
2013

Univerzita Pardubice
Fakulta ekonomicko-správní
Akademický rok: **2012/2013**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Ondřej Zápotočný**
Osobní číslo:	**E100001**
Studijní program:	**N6209 Systémové inženýrství a informatika**
Studijní obor:	**Informatika ve veřejné správě**
Název tématu:	**Vyhledávání zboží v nabídkách e-shopů**
Zadávající katedra:	**Ústav systémového inženýrství a informatiky**

Zásady pro vypracování:

V současné době existuje celá řada vyhledávačů zboží v nabídkách e-shopů založených na rozličných technologiích. Cílem práce je navrhnout vhodný vyhledávač zboží a implementovat ho v prostředí Moodle.

Obsahem práce bude:

- analýza stávajících způsobů vyhledávání zboží v nabídkách e-shopů,
- identifikace nedostatků stávajících řešení,
- návržení řešení zjištěných nedostatků,
- implementace navrženého řešení v prostředí Joomla.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

KADLEC, Václav. Agilní programování: metodiky efektivního vývoje softwaru. 1. vyd. Brno: Computer Press, 2004. ISBN 80-251-0342-0.
KENNARD, James. Mastering Joomla! 1.5 : Extension and Framework Development. B: Packt publishing, 2007. ISBN 978-1-84719-282-0.
LACKO, L'uboslav. PHP 5 a MySQL 5: hotová řešení. Vyd. 1. Překlad Jan Pokorný. Brno: Computer Press, 2007. ISBN 978-80-251-1695-1.
RAHMEL, Dan. Joomla!: podrobný průvodce tvorbou a správou webů. Vyd. 1. Překlad Ondřej Gibl. Brno: Computer Press, 2010. ISBN 978-80-251-2714-8.

Vedoucí diplomové práce: **Ing. Miloslav Hub, Ph.D.**
Ústav systémového inženýrství a informatiky

Datum zadání diplomové práce: **1. října 2012**
Termín odevzdání diplomové práce: **30. dubna 2013**

L.S.

doc. Ing. Renáta Myšková, Ph.D.
děkanka

prof. Ing. Jan Čapek, CSc.
vedoucí ústavu

V Pardubicích dne 3. října 2012

PROHLÁŠENÍ

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 30. 4. 2013 Bc. Ondřej Zápotočný

PODĚKOVÁNÍ:

Tímto bych rád poděkoval svému vedoucímu práce doc. Ing. Miloslavu Hubovi, Ph.D. za metodické vedení a cenné rady, které mi pomohly při zpracování diplomové práce.

ANOTACE

Tato diplomová práce popisuje možnosti fulltextového vyhledávání v nabídkách e-shopů. Jsou popsány základní alternativy fulltextového vyhledávání v databázovém systému MySQL, včetně jejich porovnání vybranou metodou vícekriteriálního rozhodování. V práci je uveden popis návrhu struktury databáze pro realizaci vybraného fulltextového vyhledávače implementovaného v prostředí redakčního systému Joomla!.

KLÍČOVÁ SLOVA

Fulltextové vyhledávání, Sphinx, MySQL fulltext, Apache Solr, Joomla!

TITLE

Product search in e-shop offers

ANNOTATION

This thesis describes the full-text search in e-shop offers. It describes the basic alternatives of full-text search in MySQL database system, that is also compared with the selected method of the multi-criteria decision. The thesis deals with a description of the structure of database which will be used for realization of selected full-text search which is implemented to the editorial system Joomla!.

KEYWORDS

Full-text search, Sphinx, MySQL full-text, Apache Solr, Joomla!

OBSAH

SEZNAM TABULEK

SEZNAM OBRÁZKŮ

SEZNAM ZKRATEK

AHP	Analytický hierarchický proces
API	Rozhraní pro programování aplikací
DPH	Daň z přidané hodnoty
DIČ	Daňové identifikační číslo
DTD	Definice typu dokumentu
EAN	Čárový kód používaný k označování jednotlivých druhů zboží
HTTP	Internetový protokol pro přenos objektů libovolného typu
IČ	Identifikační číslo osoby
ISBN	Mezinárodní standardní číslo knihy
LAMP	Sada svobodného softwaru Linux, Apache, MySQL a PHP
MVC	Softwarová architektura, která rozděluje datový model aplikace, uživatelské rozhraní a řídicí logiku do tří nezávislých komponent
MySQL	Systém pro řízení databází
PHP	Skriptovací programovací jazyk
PSČ	Poštovní směrovací číslo
SQL	Standardizovaný dotazovací jazyk používaný pro práci s daty v relačních databázích
TCP/IP	Komunikační protokol
URL	Řetězec znaků, který slouží k přesné specifikaci umístění zdrojů informací
VPS	Virtuální privátní server
XML	Rozšiřitelný značkovací jazyk

SEZNAM SYMBOLŮ

CI	Index konzistence
CR	Konzistenční poměr
i	Index řádkového kritéria
j	Index sloupcového kritéria
m	Počet kritérií
RI	Hodnota náhodného konzistenčního indexu udávaná v tabulkách
S	Saatyho matice
s_{ij}	Prvek Saatyho matice
v_i	Váha i-tého (řádkového) kritéria
v_j	Váha j-tého (sloupcového) kritéria
λ_{max}	Největší vlastní číslo matice vlastních čísel

Úvod

Nakupování pomocí internetu se stává v dnešní době přirozenou součástí chování spotřebitelů všech věkových generací, které přináší řadu výhod, ale též i některé nevýhody.

Prvotním impulzem, proč spotřebitel nakupuje na internetu, je cena. Cenový rozdíl při nákupu na internetu je dán tím, že internetový prodejce nemá tak vysoké náklady za pronájem prostorů pro prodejnu a sklad. Dále ušetří nemalé finanční prostředky za platy zaměstnanců, není jich potřeba takové množství. Z těchto důvodů si může internetový prodejce dovolit nižší marže a následně nižší cenu pro zákazníka. Na stranu druhou zákazník přichází o kontakt s prodejcem a např. reklamace je pak nucen z větší části si řešit sám. S nákupem zboží v internetovém obchodě je dosti často spojen poplatek za přepravu a za zabalení objednávky. Po dosažení určité částky mnohdy tyto poplatky odpadají. Objednané zboží pak můžete zaplatit několika způsoby, většinou se jedná o platbu dobírkou, převodem z bankovního účtu, kartou po internetu nebo internetovým platebním systémem jako je velmi rozšířený PayPal. Některé internetové obchody mají i síť poboček, kde si zákazník může osobně vyzvednout a v hotovosti zboží zaplatit.

Druhým impulzem je otevírací doba, internetové obchody mají otevřeno 24 hodin denně 7 dní v týdnu a jsou tak zákazníkům k dispozici neustále.

Nabídka zboží dostupného na internetu je velmi rozsáhlá oproti kamenným obchodům. Informace o produktech jsou velmi často aktualizovány, naproti tomu např. reklamní prospekty není možné takto rychle aktualizovat.

Vyvstává otázka, jak může spotřebitel takové velké množství nabídek porovnat a rozhodnout se. Zde je potřeba porovnávat zboží z více obchodů a třeba i od více výrobců. Odpovědí jsou internetové katalogy zboží, které hlídají ceny ve stovkách nebo dokonce tisících internetových obchodů.

Hlavním cílem této diplomové práce je popis a porovnání běžně používaných způsobů fulltextového vyhledávání v databázovém serveru MySQL včetně identifikace nedostatku stávajících řešení.

Dalším cílem této diplomové práce je implementace vhodného fulltextového vyhledávače v prostřední redakčního systému Joomla!. Vyhledávač bude doplněn o nezbytnou funkcionalitu pro otestování a budoucí další rozšíření.

1 VYHLEDÁVAČE ZBOŽÍ

Vyhledávače zboží jsou specializované webové stránky zaměřené na vyhledávání a porovnávání zboží dle různých parametrů, zejména dle ceny. Jsou tak předurčeny k tomu, aby přitahovaly zákazníky, kteří mají zájem nakoupit.

Uživatelé internetu si tak velmi rychle zvykli při nákupech porovnávat ceny na jednom místě. Pro uživatele je to rychlé a pohodlné. Hlavní důvod adaptace uživatelů je to, že na jediném webu najdou agregované informace ze stovek či tisíců internetových obchodů. Návštěvník stránky tak ihned ví, ve kterém internetovém obchodě je daný produkt nejlevnější, dostupný skladem nebo se může rozhodovat na základně hodnocení daného internetového obchodu. Tím se uživateli zjednoduší a i zkrátí hledání. Takto lze snadno najít internetový obchod, kde uživatel koupí co potřebuje. Nemusí tak hledat přímo v jednotlivých internetových obchodech, to je velmi zdlouhavé. [20]

Pokud se na vyhledávače podíváme z pohledu provozovatelů internetových obchodů, tak vyhledávače jsou pro ně cenným zdrojem návštěvnosti (přivádí uživatele, kteří chtějí nakupovat). Každý vyhledávač zboží řadí produkty podle nějakého algoritmu a bere tak v úvahu různé kritéria (nejčastěji cenu nebo hodnocení spokojenosti zákazníkem), takže lze omezeně protlačit své produkty nahoru i ve vyhledávači zboží a zpětně podle měření konverzí si kontrolovat úspěšnost. [15]

Známé vyhledávače zboží:

- http://www.zbozi.cz
- http://www.heureka.cz
- http://www.srovnanicen.cz
- http://www.hledejceny.cz
- http://www.monitor.cz

Obecné výhody pro uživatele jsou:

- velmi rychle srovnávat zvolené produkty napříč mnoha internetovými obchody,
- uživatel najde důležité informace pro nákup, to je např. cena, hodnocení internetových obchodů, dostupnost, termín dodání a další.

Výhody pro provozovatele e-shopů (prodejce) jsou:

- návštěvník přichází na vyhledávač zboží se záměrem nakoupit hledaný produkt. Vyhledávače zboží tak přivádějí na e-shop mnohem cílenější návštěvnost a v porovnání s ostatními zdroji návštěvnosti dosahují i několikanásobně větších konverzních poměrů,
- každý e-shop, který chce využít vyhledávače zboží, musí nejprve vytvořit exportní XML feed obsahující nabízené produkty. Tím se zrychluje výměna informací mezi vyhledávačem a e-shopem. Změny v nabídce zboží (nové zboží, slevy a jiné akční nabídky) se tak projevují velmi rychle,
- u některých vyhledávačů zboží je možno dokoupit různé služby pro zvýhodnění svého zboží, např. přednostní výpis. Produkty tak budou zobrazovány na prvních místech.

Tabulka 1 - Návštěvnost dle http://www.netmonitor.cz ze dne 01. 08. 2012, zdroj: [vlastní]

Webová stránka	Reální uživatelé	Zobrazení	Návštěvy celkem	Průměrná délka návštěvy
http://www.zbozi.cz	131 222	2 496 973	260 225	5 m 36 s
http://www.heureka.cz	127 945	1 916 303	251 706	5 m 32 s
http://www.srovnanicen.cz	22 034	104 628	38 038	2 m 15 s
http://www.hledejceny.cz	7 405	41 197	12 711	1 m 59 s
http://www.monitor.cz	1 284	4 152	2 006	-

Popis hodnot uvedených v tabulce:

- Reální uživatelé - uživatelé, kteří navštívili web.
- Zobrazení - průměrný počet zobrazených stránek.
- Návštěvy celkem - celkový počet návštěv na webu (jeden uživatel může udělat více návštěv, web navštěvují i internetoví roboti).
- Průměrná délka návštěvy - čas strávený uživatelem během jedné návštěvy.

Český trh s vyhledáváním zboží patří dle tabulky 1 webům http://www.zbozi.cz a http://www.heureka.cz. Tyto dva weby se odlišují několikrát vyšší návštěvností a průměrnou dobou, kterou návštěvník tráví na vyhledávači zboží. Hlavně tato průměrná délka návštěvy nám říká, že na těchto webových stránkách uživatel důkladně prochází obsah.

1.1 Zboží.cz

Tento vyhledávač provozuje společnost Seznam.cz a. s., je velmi jednoduchý. Umožňuje řadit podle relevance a ceny. Je propojen s webem http://www.firmy.cz.

Tento vyhledávač fungoval dlouhou dobu pouze jako jednoduchý, nepříliš inteligentní vyhledávač zboží a tak si svojí pozici držel převážně díky umístění na předních pozicích na http://www.seznam.cz a velikosti svojí databáze. Od července 2012 došlo k rozšíření funkcionality o vyhledávání podle parametrů, jako je například hmotnost, značka či velikost. Cílem tohoto rozšíření bylo zjednodušit uživateli práci v hledání mezi milióny nákupních položek, které nabízí více než 25 000 internetových obchodů. Pokud uživatel přesně ví, co hledá, stačí zadat několik směrodatných parametrů a během chvíle se uživateli zobrazí odpovídající zboží, které ho zajímají. [16], [20]

Kromě klasických technických parametrů, jakými jsou například značka, váha nebo velikost displeje, je u vybraných výrobků možnost filtrovat podle tzv. fuzzy parametrů. Ty ukazují, kam či na co se daný produkt hodí. U notebooků to mohou být funkce stylový, herní, profesionální a tak dále. Zajímavou možností hlavně pro ženy je možnost filtrování vybraného zboží podle barev.

Při využívání placených služeb je datový zdroj ve formátu XML obsahující informace o nabízeném zboží (XML feed) stahován 2x denně v ranních a odpoledních hodinách (přesný interval není určen). [16]

1.2 Heureka!

Tento vyhledávač vstoupil na trh na konci roku 2007, provozovatelem je firma Miton Media a. s. Stránky http://www.heureka.cz nefungují pouze jako obyčejný vyhledávač zboží, ale přináší další důležité informace o zboží a jednotlivých obchodech a to parametrické vyhledávání, srovnání zboží, vývoj ceny zboží, názory, recenze uživatelů, uživatelskou poradnu (radí návštěvníkům podle čeho vybírat), hodnocení a recenze jednotlivých internetových obchodů „Ověřeno zákazníky“. Služba „Ověřeno zákazníky“ využívá názory skutečných zákazníků. Hodnocení obchodu se počítá za posledních 90 dnů. Starší hodnocení se sice zobrazí v celkovém počtu všech hodnocení, ale jejich váha se již do procentuálního hodnocení obchodu nepočítá. Tím je zajištěna aktuálnost a reálná informace o aktuálním chování obchodu. [15], [17]

Vyhledávač http://www.heureka.cz se od své velké konkurence liší hlavně uživatelskou poradnou, kde pomáhá lidem s výběrem vhodného produktu, který hledají.

Při využívání placených služeb je XML feed se zbožím importován každé 2 hodiny a výsledky jsou zobrazovány na následujících stránkách:

- http://www.heureka.cz
- http://www.zbozi.centrum.cz
- http://www.zbozi.atlas.cz
- http://www.nejlepsiceny.cz
- http://www.seznamzbozi.cz

1.3 Porovnání

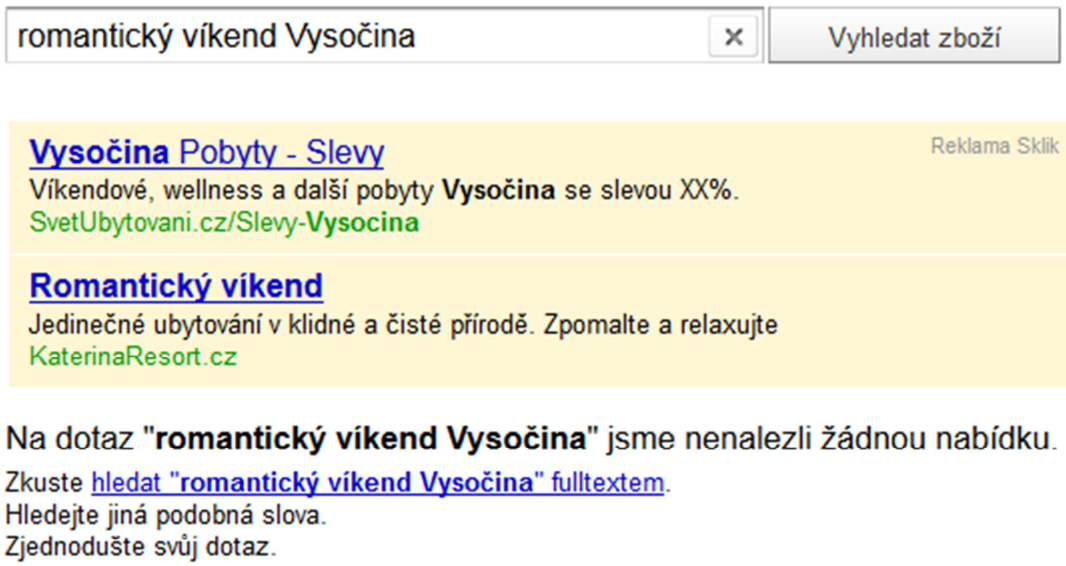
Http://www.zbozi.cz bojuje jednoduchostí a přehledností. Pro nákup pevně vybraného zboží, při kterém hledáme nízkou cenu, je toto místo vhodné. Když více informací nepotřebujeme.

Heureka.cz je už o něco dále, nabízí nejen cenové porovnání mnoha e-shopů nabízející požadované zboží, ale rovněž umožňuje sledovat oblíbenost právě zvoleného kusu, číst uživatelské recenze, případně se zeptat na nějakou otázku k produktu. Je proto vhodný nejen k samotnému nákupu vybrané věci, ale také jako "křižovatka", pokud máme přibližnou představu o poptávaném produktu nebo při výběru ze dvou různých typů. Uživatelské recenze, hodnocení i další aspekty nám tak mohou pomoci upřesnit náš výběr nebo i pomoct vybrat něco lepšího, alternativního.

Cílem vyhledávačů zboží kromě hledání v databázi je také maximalizovat zisky. Proto nabízejí několik placených funkcí pro lepší zviditelnění daného e-shopu. První možností je úprava výsledků fulltextového vyhledávání, aby např. první 2 místa obsadili produkty se zaplaceným zvýhodněním. Druhou základní možností pro finanční příjem je cena prokliku z vyhledávače zboží do e-shopu, který zboží nabízí. Cena jednoho prokliku je v základu pro obě služby Heureka! a Zboží.cz 1 Kč bez DPH. U dražších položek nebo v různých kategoriích se může cena lišit.

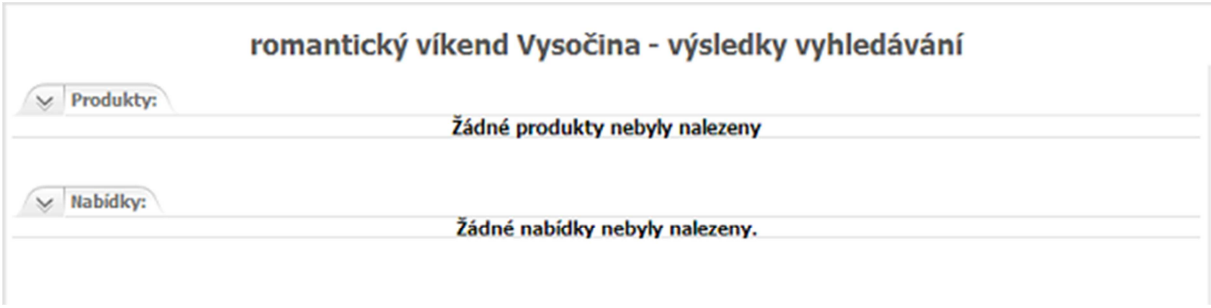
Vyhledávače se delší dobu snaží přesně identifikovat shodné produkty v různých obchodech. Toto je prováděno pomocí párování. Z tohoto důvodu je nezbytné jednoznačně identifikovat jednotlivé načítané položky například pomocí názvu, EAN kódu, dodavatele, ISBN nebo také ceny s již načtenými položkami. V případě shody, jsou položky interně shodně označeny. Vyhledávání upřednostňuje spárované položky a hledá dle parametrů, kterými se položky liší. Pro položky, které lze jednoznačně identifikovat tento způsob

funguje. Ale pokud hledáme zážitky, které není možné jednoznačně identifikovat např. romantický víkend, pobyt v lázních, jízda na čtyřkolkách a další, může se velmi snadno stát, že nenajdeme žádný výsledek.



Obrázek 1 - Hledání na http://www.zbozi.cz, zdroj: [vlastní]

Na obrázku 1 je vidět, že pro hledání výrazu „romantický víkend Vysočina“ na http://www.zbozi.cz se zobrazí pouze placená reklama a nabídka možnosti hledání pomocí fulltextového vyhledávače. Při hledání stejného výrazu na ostatních uvedených zbožových vyhledávačích http://www.monitor.cz (viz obrázek 2), http://www.srovnanicen.cz, http://www.hledejceny.cz a http://www.heureka.cz jsou výsledky hledání totožné a také nenajdeme shodu. Problém spočívá v tom, že zážitky nemají jednoznačný identifikátor jako je ISBN či čárový kód.



Obrázek 2 - Hledání na http://www.monitor.cz , zdroj: [vlastní]

Na základě tohoto porovnání byl stanoven jako cíl diplomové práce vytvořit komponentu pro volně dostupný redakční systém. Komponenta bude obsahovat pouze základní funkčnost nutnou pro správu e-shopů a import zboží. Vyhledávání v databázi bude zobrazovat čisté

fulltextové výsledky dle jejich relevance bez ovlivňování pořadí. Použití redakčního systému představuje výhodu, protože základní části řeší přímo redakční systém a není třeba je vytvářet. Jako příklad může být použita správa uživatelů, ochrana proti SQL injection a další. Vytvořená komponenta bude zohledňovat práci s řádově miliony záznamů a pro zobrazování výsledků bude používat čistě fulltextový přístup.

2 FUNKCE APLIKACE

2.1 Definice požadavků

Softwarový systém je založen na konečné množině požadavků. Popis jednotlivých požadavků je tak velmi důležitou částí vývoje systému a požadavek jako takový představuje základ všech systémů. Požadavky by měly být jediným vyjádřením toho co má systém dělat a jak se má chovat. Pro popis je nezbytné použít abstraktní popis, je popsáno, co bude systém dělat, bez popisu jak danou funkci bude systém zajišťovat. [2]

V literatuře [2] je uvedeno dělení požadavků:

- funkční požadavek určuje, jaké chování bude systém nabízet,
- nefunkční požadavek slouží k specifikaci vlastností a definici omezujících podmínek.

Pro popis požadavků pro daný systém nemá pevně definovaný postup. Specifikace je většinou napsána v přirozeném jazyce. Při popisu specifikací je vycházeno z otázky pomůže mi specifikace si uvědomit co má systém dělat a co dělat nemá?

2.2 Funkční požadavky

Na základě [22] zabývající se částí uživatelské komponenty pro zbožový vyhledávač http://onlinezbozi.cz byla s provozovatelem a tvůrcem webové aplikace definována základní funkcionalita. Aplikace pro vyhledávání zboží v katalogu e-shopů by měla umožnovat tuto základní funkcionalitu:

1. Správa uživatelů - tato funkčnost je součástí redakčních systémů a zahrnuje operace registrace, úpravy, aktivace/deaktivace a mazání. O uživateli budou uchovávány údaje: jméno, uživatelské jméno, email a heslo.
2. Správa subjektů – uživatelský účet v redakčním systému obsahuje pouze základní informace. Pro uchovávání informací o právnické osobě je potřeba zavést správu subjektů a uchovávat tyto informace: název, adresa, město, PSČ, země, IČ, DIČ a poznámku. Správa subjektů zahrnuje operace přidání, úpravy, smazání a za určitých podmínek aktivaci/deaktivaci subjektu. Uživatel může spravovat pod jedním uživatelským účtem více subjektů.
3. Správa kategorií – slouží pro správu kategorií, do kterých bude obchod zařazen. Obchod může být ve více kategoriích zároveň.

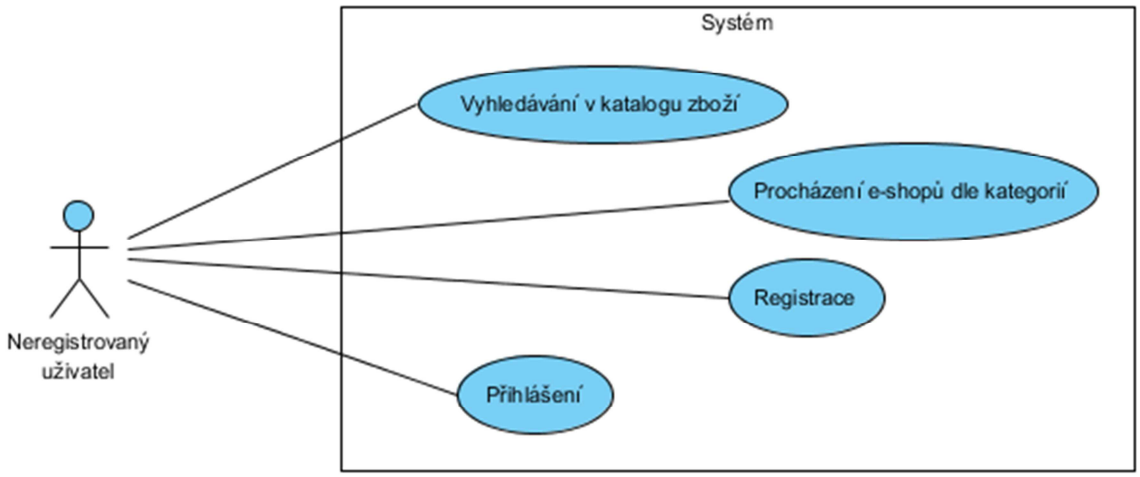
4. Správa e-shopů – e-shop je vázaný na subjekt, každý uvedený subjekt může mít více e-shopů. E-shop může být zařazený ve více kategoriích. O e-shopu bude potřeba uchovávat následující údaje: název, popis, URL, telefon, email, URL adresu XML souboru se zbožím a logo. S e-shopy se pojí operace přidání, úpravy, aktivace/deaktivace a mazání.
5. Správa zboží – zboží se do systému bude importovat automaticky pomocí XML souboru se zbožím. O zboží se budou ukládat pouze základní údaje: název, popis, cena s DPH, URL adresa obrázku produktu, URL adresa produktu v daném e-shopu, typ výrobku (nový, rozbaleno, bazar …) a doba vyřízení objednávky.
6. Vyhledávání - musí být schopné vyhledávat ve velké databázi produktů (obsahující řádově miliony záznamů) a používat fulltextové vyhledávání s možností nastavování typu shody a řazení výsledů.

Systém kromě popsané funkcionality musí mít definované role pro jednotlivé typy uživatelů. Na obrázku 3 jsou jednotlivé role pro uživatele uvedeny.

Obrázek 3 - Role v systému a jejich generalizace, zdroj: [vlastní]

První rolí je aktér „Neregistrovaný návštěvník“, to je běžný uživatel, který přijde na stránky podívat se a pravděpodobně bude hledat v katalogu jím poptávané zboží. Druhou rolí je aktér „Registrovaný uživatel“, to je uživatel (prodejce), který se registroval a může tak využívat další funkce stránek, tím se myslí především import zboží do aplikace. Poslední rolí je aktér „Administrátor“, tento aktér má kontrolu nad všemi položkami daného systému a celým redakčním systémem.

Funkčním požadavkem se formuluje to, co by měl systém dělat. Funkční požadavky budou definovány pro jednotlivé role. Každá role v systému má svá práva co může dělat. Na obrázku 4 je uveden přehled činností, pro neregistrovaného uživatele.



Obrázek 4 - Přehled činností pro aktéra "Neregistrovaný návštěvník", zdroj: [vlastní]

1. Hledání v katalogu zboží – neregistrovaný uživatel může vyhledávat v katalogu zboží. Při vyhledávání systém nabízí zvolit typ shody (všechna slova, některá slova, přesná shoda a použití booleovských operátorů), výchozí hodnota je shoda některých slov. Výsledky ve výchozím nastavení řadí dle relevance, uživatel musí mít možnost řazení změnit a výsledky si zobrazit seřazené dle ceny vzestupně i sestupně. Ve výsledcích se zobrazí zboží, které je označeno za aktivní. E-shop, do kterého dané zboží patří a subject do kterého patří daný e-shop musí být také aktivní.
2. Procházení e-shopů dle kategorií zobrazuje strukturu kategorií a seznam e-shopů, který je v dané kategorii uveden. Zobrazen je popis dané kategorie a základní informace o e-shopu tj. název, logo, www odkaz, kontaktní údaje. Zobrazeny budou pouze aktivní obchody (Aktivní obchod musí mít aktivní i subjekt).
3. Po registraci a přihlášení do systému je získán přístup k dalším službám internetové aplikace. Údaje potřebné k registraci jsou jméno, přihlašovací jméno, heslo a email. Po registraci systém uživatele automaticky přesměruje na zobrazení přehledu subjektů a následně bude systémem uživatel vyzván k vytvoření subjektu.
4. Do systému se budou moci přihlašovat libovolný počet uživatelů, k přihlášení je nutné zadat uživatelské jméno a heslo.

Na obrázku 5 je zobrazen přehled činností, které může provádět registrovaný uživatel po přihlášení.

Obrázek 5 - Přehled činností pro aktéra "Registrovaný uživatel", zdroj: [vlastní]

Pro každou z činností na obrázku 5 musí být uživatel přihlášen. Při každé činnosti je kontrolováno přihlášení. Pokud by uživatel spustil danou činnost bez přihlášení, nesmí být činnost povedena a uživatel musí být přesměrován s chybou zprávou na úvodní stránku.

1. Při přidání subjektu je k subjektu možné zadat jméno, ulici, číslo domu, město, PSČ, země, IČ, DIČ a poznámka. Povinné údaje jsou kontrolovány a bez jejich vyplnění není možné data uložit. Povinné údaje jsou název, adresa, číslo popisné, město, PSČ a IČ. IČ musí být v systému jedinečné. Systém k novému záznamu nastaví atribut user_id, který představuje id uživatele, který si subjekt přidal a atribut published na 1, aby byl nový subjekt aktivní.

2. Při změně údajů subjektu musí být při otevření zkontrolováno, jestli daný subjekt je přihlášeného uživatele. Pokud se záznam pokusí otevřít uživatel, kterému nepatří, nesmí být zobrazena data subjektu a uživatel musí být s chybnou zprávou přesměrován na úvodní stránku. Povinné údaje jsou uvedeny u přidání subjektu a bez jejich vyplnění není možné data uložit. Je možné měnit veškeré údaje.
3. Uživatel může smazat pouze subjekt, na který není napojen žádný e-shop.
4. Přehled subjektů zobrazuje všechny subjekty, u kterých je přihlášený uživatel uveden. Zobrazuje název, IČ, počet obchodů, stav subjektu a jeho id. Přehled subjektů umožňuje filtrovat podle názvu, IČ a řadit záznamy dle všech atributů.
5. Při přidání e-hopu je možné vybrat subjekt a zadat název, popis, URL, feed_url, email, telefon, zveřejnění a nahrát logo. Povinné údaje jsou kontrolovány a bez jejich vyplnění není možné data uložit. Povinné údaje jsou subjekt, název, popis, URL a email. Systém k novému záznamu nastaví atribut published na 1, aby byl nový subjekt aktivní.

   Při přidání e-shopu do systému musí být vybrány kategorie, do kterých e-shop patří. Bez vybrané kategorie nemůže být záznam uložen.
6. Při úpravě údajů o e-shopu je možné měnit všechny údaje a měnit kategorie, do kterých e-shop patří. Při otevření záznamu je kontrolováno, jestli daný subject je přihlášeného uživatele, pokud ne uživatel musí být s chybou zprávou přesměrován na úvodní stránku.
7. Při aktivaci e-shopu je zajištěno, že mohu být aktivovány e-shopy, které mají aktivní subjekt. Při deaktivaci e-shopu je zajištěno, aby se deaktivovalo všechno zboží, které má daný e-shop v nabídce.
8. Při smazání e-shopu systém smaže veškeré související údaje. Odstraní e-shop z kategorií, logů a smaže veškeré zboží, které je k obchodu přiřazeno.
9. Přehled e-shopů zobrazí všechny e-shopy, u kterých je uveden subject u kterého je uvedený přihlášený uživatel. Zobrazen je název, email, telefon, URL, stav vyplnění adresy zbožového XML (feed url), stav (aktivní/neaktivní). Přehled e-shopů umožňuje filtrovat podle názvu, emailu a telefonu.
10. Registrovaný uživatel může prohlížet záznamy o provedených importech zboží. V záznamu je uveden počet dobře importovaných položek, počet neimportovaných položek a stav importu, který obsahuje číslo chyby, pro zjištění k jaké chybě došlo.

Na obrázku 6 je zobrazen přehled činností, které může provádět administrátor po přihlášení.

Obrázek 6 - Přehled činností pro aktéra "Administrátor", zdroj: [vlastní]

Pro každou z činností na obrázku 6 musí být uživatel přihlášen. Při každé činnosti je kontrolováno přihlášení. Pokud by uživatel spustil danou činnost bez přihlášení, nesmí být činnost povedena a uživatel musí být přesměrován na stránku pro přihlášení. Administrátor má přístupné všechny záznamy od všech uživatelů.

1. Správa uživatelů – představuje veškeré operace, které umožňuje daný redakční systém. Typické operace jsou přidání/úprava/aktivace/deaktivace a smazání.

2. Přidání/úprava kategorie – povinný údaj u kategorie je pouze její název. Pokud není uvedeno jinak, kategorie bude nejvyšší úrovně a aktivní.
3. Aktivace/deaktivace kategorie slouží zneviditelnění kategorie a zviditelnění v uživatelské části aplikace.
4. Při smazání kategorií systém odstraní e-shopy z dané kategorie a případně i podkategorií. V druhém kroku systém smaže danou kategorii a její podkategorie.
5. Přehled kategorií administrátorovi zobrazí hierarchický strom kategorií. Je možné nastavovat pořadí a filtrovat zobrazení dle nejvyšší kategorie a hledat kategorie dle názvu. V přehledu je zobrazen název, nadřazená kategorie, počet s-ehopů v kategorii.
6. Při přidání a úpravu subjektu je k subjektu možné zvolit uživatele a zadat jméno, ulici, číslo domu, město, PSČ, země, IČ, DIČ a poznámka. Povinné údaje jsou kontrolovány a bez jejich vyplnění není možné data uložit. Povinné údaje jsou uživatel, název, adresa, číslo popisné, město, PSČ a IČ. IČ musí být v systému jedinečné. Systém k novému záznamu nastaví atribut published na 1, aby byl nový subjekt aktivní.
7. Aktivace/deaktivace subjektu systém povolí pouze administrátorovi. Při deaktivaci subjektu systém deaktivuje subjekt, e-shopy, kde je uveden daný subjekt a zboží, které e-shopy nabízejí. Při aktivaci je aktivován pouze subjekt.
8. Administrátor může smazat pouze subjekt, na který není napojen žádný e-shop.
9. Přehled subjektů zobrazuje veškeré subjekty, umožnuje hledání dle názvu, adresy, IČ, filtrovat dle aktivní/neaktivní a dle uživatele. K údajům uložených v databázi přidává počet e-shopů.
10. Při přidání a úpravě e-hopu je možné vybrat subjekt a zadat název, popis, URL, feed_url, email, telefon a zveřejnění. Povinné údaje jsou kontrolovány a bez jejich vyplnění není možné data uložit. Povinné údaje jsou subjekt, název, popis, URL a email. Systém k novému záznamu nastaví atribut published na 1, aby byl nový subjekt aktivní.

    Při uložení do DB musí být k e-shopu vybrané kategorie, do kterých e-shop patří. Bez vybrané kategorie nemůže být záznam uložen.
11. Při aktivaci e-shopu je zajištěno, že mohu být aktivovány e-shopy, které mají aktivní subjekt. Při deaktivaci e-shopu je zajištěno, aby se deaktivovalo všechno zboží, které má daný e-shop v nabídce.

12. Při smazání e-shopu systém smaže veškeré související údaje. Odstraní e-shop z kategorií, logů a smaže veškeré zboží, které je k obchodu přiřazeno.
13. Přehled logu – systém zobrazí informace o dění v systému, především o stavu importu zboží a počtu přesměrování do jednotlivých obchodů. Filtrovat lze dle subjektu, e-shopu a typu záznamu
14. Administrátor musí mít možnost smazat záznam logu.

2.3 Nefunkční požadavky

Nefunkční požadavky představují omezující podmínky pro daný systém a jeho implementaci.

1. Administrace aplikace bude uživatelsky přívětivá.
2. Uživatelská část bude uživatelsky přívětivá.
3. Aplikace bude podporovat jazykové mutace.
4. Uživatelská část bude obsahovat jazykový balíček s češtinou.
5. Aplikace je tvořena s ohledem na budoucí rozšíření.
6. Technologie je zvolena s ohledem na pokračování jiným programátorem.
7. Import zboží z XML souboru se zbožím pro aktivní e-shopy.
8. Obnova vyhledávajícího indexu.
9. Administrátor je notifikován při otevření právě editovaného záznamu.
10. Aplikace pro svůj provoz vyžaduje virtuální server . Běžný webový hosting není vhodný z důvodu výpočetní náročnosti, velikosti diskového prostoru potřebného pro uložení indexů pro fulltextové vyhledávání, použití jiného fulltextového vyhledávání než je obsažené v MySQL serveru a v poslední řadě je nutná rychlá konektivita pro stahování XML souborů se zbožím.

2.4 XML Feed

Aby bylo možné do systému importovat produkty z daného e-shopu, musí mít e-shop vytvořený XML soubor, pomocí kterého dojde k předání informací o aktuálně nabízeném zboží a jeho cenách z daného e-shopu do vyhledávače. XML soubor je z pravidla generován automaticky.

Takovému souboru se říká XML feed. Tyto XML soubory nemají zcela pevně daný formát. Ze služeb na vyhledávání zboží jsou nejvíce rozšířeny formáty, se kterými pracuje Heureka! a Zboží.cz. Struktura souboru těchto dvou vyhledávačů si je na první pohled dost podobná, ale stejná ani zaměnitelná není.

Způsob aktualizace pomocí XML feedu pro obě metody je stejný. Heureka! k specifikaci produktového zdroje přidává specifikaci zdroje obsahující detaily o dostupnosti jednotlivých položek (doba dodání, stav skladových zásob …). Data obsažená v produktovém zdroji jsou velmi obsáhlá a není nutné je aktualizovat tak často. Naopak data o dostupnosti dané položky jsou méně obsáhlá a vyžadují častější aktualizace. Takto lez snížit nároky na velikost přenesených dat a čas zpracování. Konkurenční Zboží.cz s takovýmto rozdělením datového zdroje nepočítá. [16], [17]

Pro definici struktury dokumentu XML definovaného konsorciem W3C se v literatuře [6] uvádí metoda definice typu dokumentu (DTD) a XML Schéma. Ve zdroji [12] jsou uvedené metody porovnány. Zápis jakéhokoli XML schémata zapsaného v jazyce DTD bude v jazyce XML Schéma výrazně delší a méně přehledný. XML Schéma má proti DTD tři výhody.

První výhodou je určení datového typu pro obsah elementů a atributů. XML bylo primárně určeno pro značkování dokumentů textové povahy, jako jsou např. knihy, kde v těchto dokumentech je převážně text. XML se masivně používá pro výměnu strukturovaných dat mezi informačními systémy. V dokumentech jako je faktura nebo v našem případě popis zboží není zdaleka vše text, obsahuje číselné a měnové údaje, EAN kód a další specifické údaje. Pro účinnou validaci takového dokumentu je zapotřebí omezit nejen strukturu jednotlivých elementů (toto DTD umí), ale i obsah jednotlivých elementů a atributů jejím datovým typem, což DTD neumí.

Druhou nevýhodou je, že pro zápis DTD se používá syntaxe podobná regulárním výrazům, kde definovat např. z jakých elementů se skládá faktura je velmi jednoduché, ale takovýto zápis používá syntaxi, která se nikde jinde v XML nepoužívá.

Třetí nevýhodou DTD proti XML Schéma je žádná podpora jmenných prostorů. Jmenné prostory jsou mechanismus, pomocí kterých je možné v jednom dokumentu XML kombinovat více sad značek.

Na základě těchto výhod přes uvedenou nevýhodu delšího zápisu bylo pro popis XML struktury vybráno XML Schéma. Pro ověření zdali je XML Schéma validní bylo použito profesionální vývojové prostřední Liquid XML Studio. Použitý software je možné použít v trial verzi, kdy program funguje 30 dní.

XML Schéma pro http://www.zbozi.cz bylo vytvořeno dle zdroje [16], kde je podrobný popis jednotlivých elementů, schéma je uvedeno v příloze C. XML Schéma pro http://www.heureka.cz, je uvedeno v příloze D. Bylo zpracováno dle zdroje[17], kde je i uveden detailní popis jednotlivých elementů. Oba ukázkové XML soubory a soubory s XML schématem prošli validací bez chyb.

V kapitole 2 bylo uvedeno, že aplikace bude o produktu uchovávat základní údaje a to název, popis, cena s DPH, URL adresa obrázku produktu, URL adresa produktu v daném e-shopu, stav produktu a dobu vyřízení objednávky.

Tabulka 2 - Použité elementy ve vlastním návrhu XML Feedu, zdroj: [vlastní]

Značka	Popis
SHOP	Kořenový element, který je použit pouze jednou.
SHOPITEM	Element, který obsahuje informace o konkrétním produktu.
PRODUCT	Název produktu obsahující k jednoznačné identifikaci.
DESCRIPTION	Popis výrobku.
URL	Odkaz na stránku s nabídkou daného výrobku.
IMGURL	Odkaz na obrázek výrobku.
ITEM_TYPE	Udává typ zboží tj. nové /bazarové.
DELIVERY_DATE	Datum možného odeslání k zákazníkovi
PRICE_VAT	Cena v Kč včetně DPH.
PRICE	Cena bez DPH
VAT	Hodnota DPH (desetinné číslo)

Dle tabulky 2 byl vytvořen ukázkový XML soubor s názvem com_fts.xml, který představuje strukturu položek pro import zboží do vyhledávače.

```
<?xml version="1.0" encoding="utf-8"?>

<SHOP xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="com_fts.xsd">

<SHOPITEM>
    <PRODUCT>HTC Desire</PRODUCT>
    <DESCRIPTION>HTC Desire oslní svým spracováním.</DESCRIPTION>
    <URL>http://obchod.cz/mobily/htcdesire</URL>
    <ITEM_TYPE>new</ITEM_TYPE>
    <IMGURL>http://obchod.cz/obrazky/mobily/htcdesire.jpg</IMGURL>
    <PRICE>5567</PRICE>
    <VAT>0,21</VAT>
    <PRICE_VAT>6736.07</PRICE_VAT>
    <DELIVERY_DATE>ihned</DELIVERY_DATE>
</SHOPITEM>
<SHOPITEM>
```

```
    <PRODUCT>HTC Desire</PRODUCT>
    <DESCRIPTION>HTC Desire oslní svým spracováním.</DESCRIPTION>
    <URL>http://obchod.cz/mobily/htcdesire</URL>
    <ITEM_TYPE>bazar</ITEM_TYPE>
    <IMGURL>http://obchod.cz/obrazky/mobily/htcdesire.jpg</IMGURL>
    <PRICE_VAT>7890</PRICE_VAT>
    <DELIVERY_DATE>4</DELIVERY_DATE>
</SHOPITEM>
</SHOP>
```

Pomocí XML Schématu byla definována struktura XML Feedu, který bude aplikací importován.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="SHOP" type="SHOP_TYPE"/>

  <xs:complexType name="SHOP_TYPE">
    <xs:sequence minOccurs="1"  maxOccurs="unbounded">
      <xs:element name="SHOPITEM" type="SHOPITEM_TYPE"></xs:element>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="SHOPITEM_TYPE">
    <xs:sequence>
      <xs:element name="PRODUCT"          type="xs:string" />
      <xs:element name="DESCRIPTION"      type="xs:string"/>
      <xs:element name="URL"              type="xs:anyURI" />
      <xs:element name="ITEM_TYPE"        type="ITEM__TYPE"      minOccurs="0"/>
      <xs:element name="IMGURL"           type="xs:anyURI"       minOccurs="0"/>
      <xs:element name="PRICE"            type="CZK_PRICE"       minOccurs="0" />
      <xs:element name="VAT"              type="VAT_TYPE"        minOccurs="0" />
      <xs:element name="PRICE_VAT"        type="CZK_PRICE" />
      <xs:element name="DELIVERY_DATE" type="DELIVERY__DATE" minOccurs="0“ />
    </xs:sequence>
  </xs:complexType>

 <xs:simpleType name="CZK_PRICE">
  <xs:restriction base="xs:decimal">
   <xs:minExclusive value="0" />
  </xs:restriction>
 </xs:simpleType>

  <xs:simpleType name="ITEM__TYPE">
  <xs:restriction base="xs:string">
   <xs:enumeration value="new" />
   <xs:enumeration value="bazar" />
  </xs:restriction>
 </xs:simpleType>

 <xs:simpleType name="DELIVERY__DATE">
  <xs:union>
```

```
    <xs:simpleType>
      <xs:restriction base="xs:date"/>
    </xs:simpleType>
    <xs:simpleType>
      <xs:restriction base="xs:int">
        <xs:minInclusive value="-1"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="ihned"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:union>
 </xs:simpleType>

 <xs:simpleType name="VAT_TYPE">
  <xs:restriction base="xs:string">
    <xs:enumeration value="0,21" />
    <xs:enumeration value="21" />
    <xs:enumeration value="0,15" />
    <xs:enumeration value="15" />
  </xs:restriction>
 </xs:simpleType>
</xs:schema>
```

Při definici XML Schémata byl zjištěn nedostatek tohoto jazyka. Pomocí tohoto jazyka nelze zapsat tvrzení o přítomnosti nebo absenci určitých elementů, atributů v dokumentu. Konkrétně se jedná o problém s elementy PRICE, VAT a PRICE_VAT. Při uvedení elementů PRICE a VAT již není nezbytné uvádět element PRICE_VAT a opačně.

3 NÁVRH DATABÁZE

Pro správnou funkčnost aplikace je velmi důležité, aby data uložená v databázi byla správná a konzistentní. V této kapitole je popsán základní návrh databáze. Při návrhu bylo postupováno v souladu s [18], kde je návrh databáze podrobně popsán.

Pro potvrzení implementační vize byl v prvním kroku popsán obsah systému. Byly identifikovány jednotlivé entity, jejich atributy, vztahy a vzájemné omezení. V druhém kroku je provedena transformace entit a vztahů. Následně je nezbytné odstranit anomálie pomocí procesu zvaného normalizace dat. Třetí krok představuje způsob fyzického uložení dat a jejich strukturu v daném databázovém systému [18]. Pro návrh byl použit program Microtool case 4/0, který je dostupný na adrese http://www.microtool.de.

3.1 Identifikace entit, atributů, omezení a vztahů

Dle popisu funkcí aplikace (viz kapitola 2), byly určeny tyto entity: users, subjects, e-shops, products, logs a e-shop categories. Entita users představuje uživatele a atributy potřebné pro vytvoření účtu. V tabulce 3 jsou popsány jednotlivé atributy.

Tabulka 3 - Přehled atributů entity users, zdroj: [vlastní]

Název atributu	Datový typ	Popis	Ukázka hodnoty
Name	varchar (255)	jméno uživatele	Ondřej Zápotočný
Username	varchar (150)	přihlašovací jméno	ondra
Email	varchar (100)	email	ondra@seznam.cz
Password	varchar (100)	hash hesla	6f51af998a4b989343e43e43e …

Entita subjects představuje právnické osoby a právnické osoby představují v systému provozovatele e-shopů. V tabulce 4 jsou popsány jednotlivé atributy.

Tabulka 4 - Přehled atributů entity subjects, zdroj: [vlastní]

Název atributu	Datový typ	Popis	Ukázka hodnoty
Name	varchar (100)	jméno subjektu	KomTeSa spol. s r.o.
Address	varchar (100)	fakturační adresa	Jihlavská 127, 58254 Havl. Brod
Country	varchar (40)	země	Czech Republic
Ic	integer (8)	identifikační číslo subjektu	45535663
Dic	varchar (10)	daňové identifikační číslo	CZ45535663
Note	text	poznámka pro daný subjekt	Dodavatel komplexních IT služeb

Entita e-shops představuje internetové obchody, které jsou v systému registrovány. Obchod může využívat služeb vyhledávače a nabízet v něm své produkty. V tabulce 5 jsou popsány jednotlivé atributy.

Tabulka 5 - Přehled atributů entity e-shops, zdroj: [vlastní]

Název atributu	Datový typ	Popis	Ukázka hodnoty
Title	varchar (100)	název eshopu	HEUREKA.CZ
Description	text	popis eshopu	Nákupní rádce, který radí …
Url	varchar (100)	www adresa	http://www.heureka.cz
Feed_url	varchar (100)	www adresa XML obsahující zboží	http://www.heureka.cz/export.php
Email	varchar (40)	kontaktní email	info@heureka.cz
Telephone	varchar (13)	kontaktní telefon	+420569234789
Img	bool	nahrané logo	1

Entita e-shop categories představuje kategorie, do kterých je e-shop zařazen. V tabulce 6 jsou popsány jednotlivé atributy.

Tabulka 6 - Přehled atributů entity e-shop categories, zdroj: [vlastní]

Název atributu	Datový typ	Popis	Ukázka hodnoty
Title	varchar (40)	Název kategorie	Výpočetní technika
Description	text	Popis kategorie	Obchod nabízející výpočetní techniku od výrobců …

Entita log představuje informace o provedených importech zboží do vyhledávače. Její atributy jsou popsány v tabulce 7.

Tabulka 7 - Přehled atributů entity logs, zdroj: [vlastní]

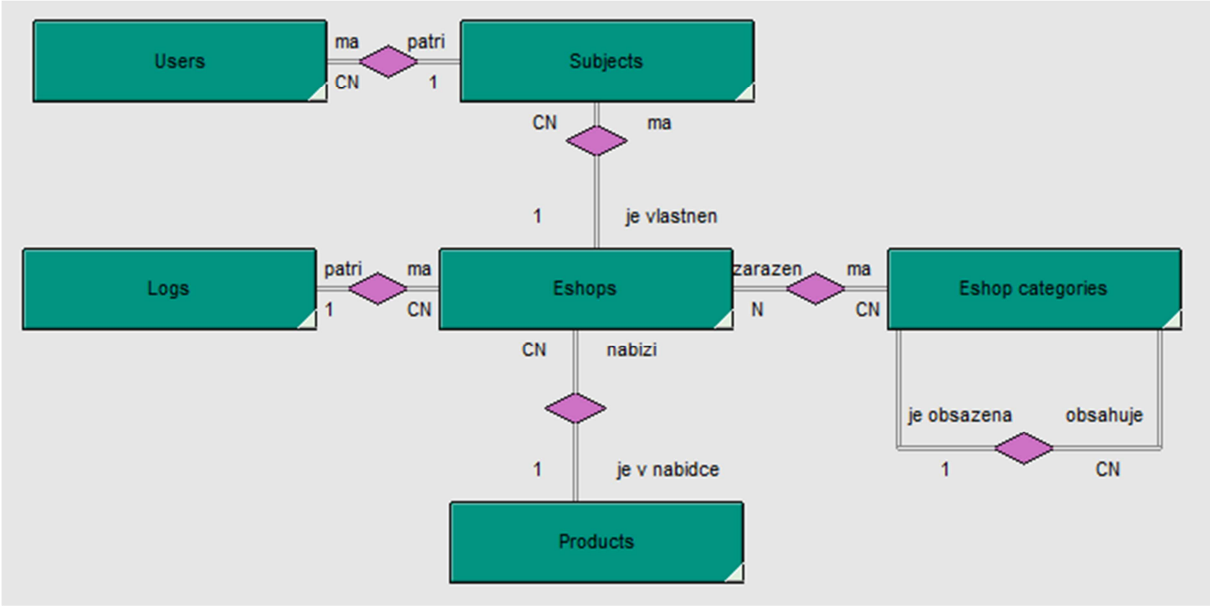
Název atributu	Datový typ	Popis	Ukázka hodnoty
Type	varchar (40)	Typ záznamu	import
Time	datatime	Čas události	2013-04-05 11:04:10
Good	integer (11)	Počet správně importovaných položek	707
Error	integer (11)	Počet položek, které nebyly importovány	0
State	integer (11)	Stav záznamu	1

Poslední entitou je entita products, která představuje nabízené produkty, které jsou v nabídce daného e-shopu. Pokud je položka nabízena více e-shopy, nebude seskupována ale ukládána každá samostatně. Jednotlivé atributy jsou popsány v tabulce 8.

Tabulka 8 - Přehled atributů entity products, zdroj: [vlastní]

Název atributu	Datový typ	Popis	Ukázka hodnoty
Product	varchar (100)	Název produktu	Samsung Galaxy S III Mini
Description	text	Popis produktu	Mobilní telefon 4.0" 480x800 …
URL	varchar (200)	Odkaz na daný produkt do e-shopu	http://www.obchod.cz/mobil
IMGURL	varchar (200)	Odkaz na obrázek daného produktu	http://www.obchod.cz/mobil.jpg
Price_vat	decimal (9,2)	Cena produktu s DPH	1342
Item_type	varchar (60)	Pro rozlišení nového a bazarového zboží	new
Delivery_date	varchar (60)	Datum dodání	ihned

Integrita databáze zjednodušeně znamená, že musí být zajištěny správná data v databázi [18]. Integritní omezení vztahů mezi entitami lze definovat pomocí kardinality a parciality. Kardinalita vztahu mezi entitami vyjadřuje četnost tj. minimální a maximální počet výskytu entity v určitém vztahu. Nabývá hodnot 1 pro výskyt maximálně jednou nebo N pro více výskytů. Parcialita představuje volitelnost členství ve vztahu. Členství může být povinné, kdy musí vzniknout vztah nebo je členství volitelné, kdy vztah vzniknout nemusí. Písmeno C vyjadřuje volitelnost vztahu, absence písmena C vyjadřuje povinnost vztahu. Existuje několik notací zápisu kardinality a parciality, použitá notace byla převzata z [19]. Na obrázku X je znázorněn model vztahů mezi entitami, který je doplněn o kardinalitu a parcialitu.



Obrázek 7 - Model vztahů mezi entitami, zdroj: [vlastní]

Popis jednotlivých vztahů:

- Users-Subjects – Jeden uživatel nemusí nebo může vlastnit více subjektů. Subjekt musí být vlastněn právě jedním uživatelem. Vztah zajišťuje, aby měl každý subjekt svého jednoznačného vlastníka.
- Subjects-Eshops – Jeden subjekt nemusí nebo může vlastnit více e-shopů. E-shop musí být vlastněn právě jedním subjektem. Vztah zajišťuje, aby každý e-shop měl svého jednoznačného provozovatele.
- Eshops-Eshop categories – Každý e-shop musí být zařazený do jedné nebo více kategorií. V každé kategorii může být žádný nebo N eshopů, nebo-li kategorie může být prázdná. Vztah zajišťuje, aby každý e-shop mohl být zařazen do kategorií dle svého nabízeného sortimentu.
- Eshop categories- Eshop categories – Kategorie může obsahovat podkategorie, ale každá kategorie musí mít nadřazenou kategorii pouze jednu. Tento vztah slouží pro vytvoření stromu kategorií.
- Eshops-Products – Představuje produkty, které jsou v nabídce daného e-shopu. Jeden e-shop nemusí mít vložen žádný produkt nebo jich může mít v systému importovaný libovolný počet. Každý produkt musí mít svého jednoznačného prodejce.

Integritní omezení jednotlivých atributů zachycuje vlastnosti a požadavky na atributy. Omezení u entity subject představuje jedinečnost IČ a u entity ehops představuje jedinečnost www adresy a adresy XML feedu.

3.2 Transformace entit a normalizace dat

Pomocí pravidel transformace vznikají první návrhy relací. Relace za určitých podmínek vznikají z entit a ze vztahů mezi entitami. Vztahy mezi entitami v ER schématu jsou binární (mezi dvěma entitami) nebo n-ární (mezi více než dvěma entitami). V modelu vztahů mezi entitami jsou pouze dva typy vztahů 1:N a N:M, pokaždé je jeden entitní typ nepovinný. Pro transformaci použijeme dvě pravidla, která jsou následně popsána.

Pro binární vztah 1:N a povinné členství pro jeden a nepovinné členství pro druhý entitní typ definujeme dvě schémata relací. Pro vyjádření závislosti druhého na prvním entitním typu přidáme atribut identifikačního klíče prvního entitního typu k druhému entitnímu typu. Např. entity users, subjects a jejich vztah transformujeme do dvou schémat relací a primární klíč id relace users přidáme jako atribut user_id k relaci subjects. [18]

Pro binární vztah N:M bez ohledu na povinnost členství entitního typu ve vztahu definujeme 3 schémata pro každý entitní typ a pro jejich vztah. Ve vztahovém schématu je obsažen primární klíč prvního i druhého entitního typu. [18]

Tabulka 9 - Výsledek transformace, zdroj: [vlastní]

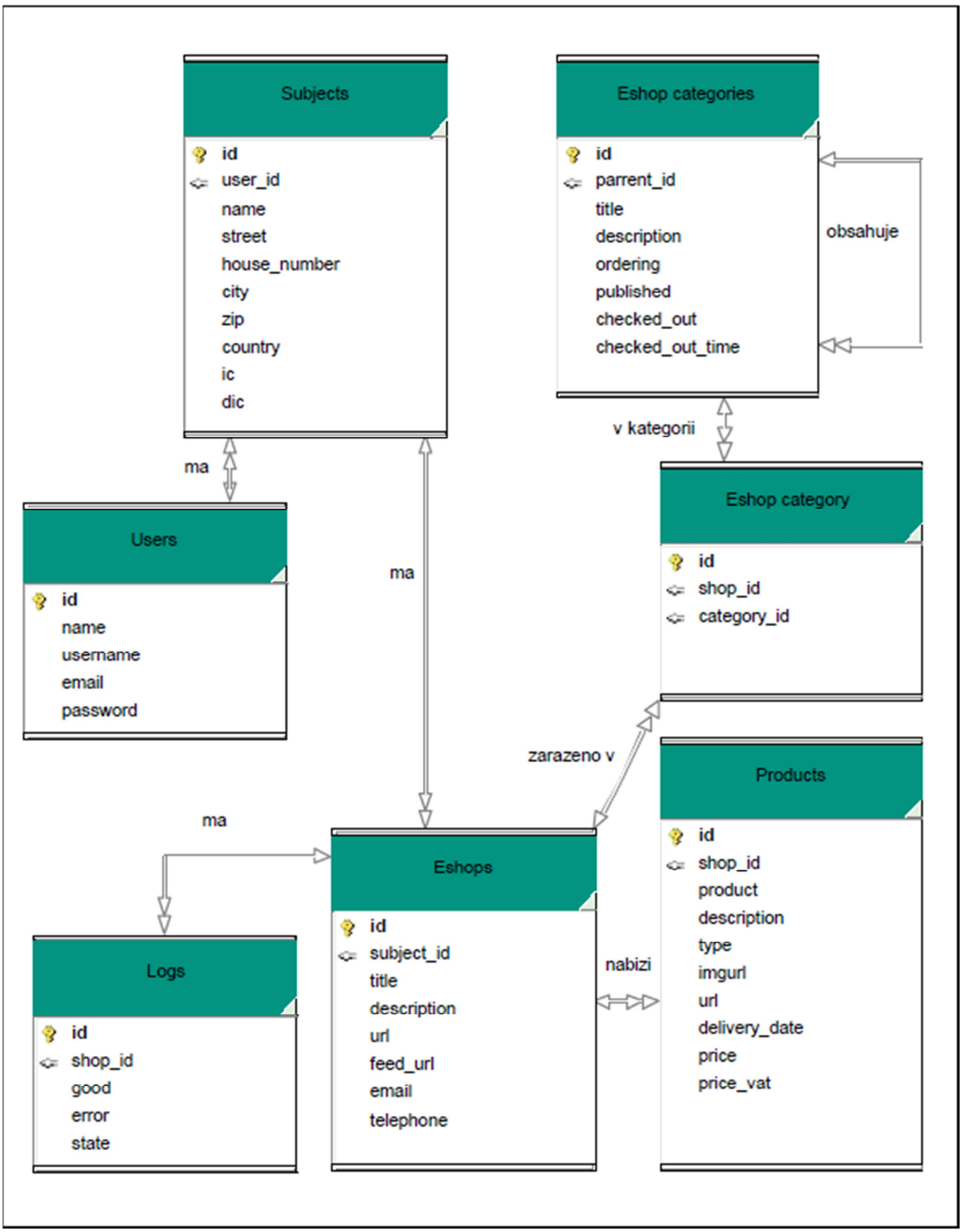
Název tabulky	Atributy
Users	id, name, username, email, password
Subjects	id, user_id, name, address, ic, dic, note
Eshops	id, subject_id, title, description, url, feed_url, email, telephone, img
Logs	id, shop_id, good, error, state, time, type
Products	id, shop_id, product, description, url, imgurl, price_vat, item_type, delivery_date
Eshop category	id, shop_id, category_id
Eshop categories	id, parrent_id, title, description,

Dalším krokem je normalizace dat, která se stará o odstranění anomálií vzniklých v prvotním návrhu datového modelu získaného aplikací pravidel transformace. Cílem je postupná dekompozice datového modelu rozdělením atributů do většího počtu relací, které již nevykazují dané nedostatky.

Normálních forem je 5. Bylo rozhodnuto, že návrh bude splňovat první 3 normální formy. [18]

- Tabulky jsou v první normální formě, jestliže lze do každého pole dosadit pouze jednoduchý datový typ. Pro splnění této podmínky je nutné rozdělit atribut adresa na více atributů. Atribut Adresa rozdělíme na atributy ulice, číslo popisné, obec a PSČ. Podmínka je splněna a žádný z atributů již není dále dělitelný.
- Druhá normální forma se vyznačuje tím, že každý neklíčový atribut je plně závislý na atributech primárního klíče. Podmínka je splněna.
- Třetí normální forma zajišťuje, že hodnoty atributů nejsou funkčně závislé na hodnotách jiných atributů, jinak řečeno atribut nezávisí na jiném atributu závisícím na primárním klíči.

Všechny tři výše popsané normální formy uvedený model splňuje a výsledný návrh databáze je uveden na obrázku 8.



Obrázek 8 - Relační model dat po transformaci a normalizaci, zdroj: [vlastní]

3.3 Implementace

Implementace se zabývá popisem relačního modelu dat v konkrétním databázovém systému, v našem případě v MySQL. Model databáze je dále potřeba upravit pro lepší integraci s použitým redakčním systémem Joomla!. Redakční systém pro správu záznamu využívá několik dalších atributů, které se můžou do tabulky přidat pro jednodušší práci

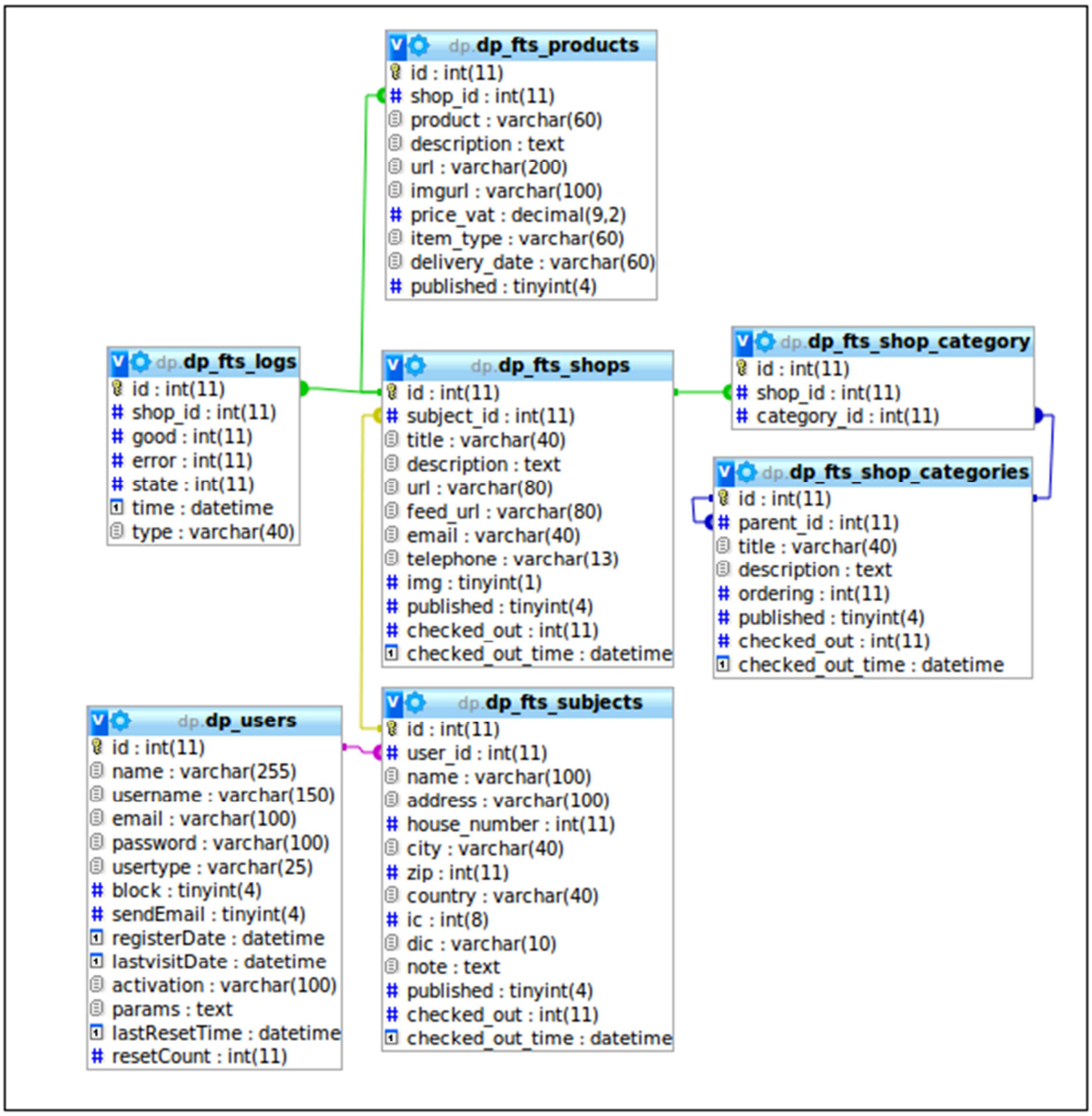
se záznamy. Ve zdroji [9] jsou uvedeny základní atributy pro Joomla! 1.5, v současné verzi Joomla! 2.5 jsou tyto atributy dále využívány. Popis jednotlivých atributů, které budou použity, je uveden v tabulce 10.

Tabulka 10 - Atributy používané v redakčním systému Joomla!, zdroj: [9]

Název atributu	Datový typ	Popis
Published	tinyint(4)	Slouží pro aktivaci a deaktivaci záznamu. Nabývá hodnot 0/1
Ordering	integer (11)	Slouží pro uložení uživatelsky definovaného pořadí záznamů
Checked_out	integer (11)	Slouží pro uložení id uživatele a času otevření záznamu pro editaci k následnému uzamčení řádku před ostatními uživateli. Zajišťuje tak, aby více uživatelů nemohlo otevřít záznam pro editaci.
Checked_out_time	datatime	

Do jednotlivých tabulek, byly dle potřeby využitelnosti, doplněny atributy a tabulky 10. Na obrázku 9 je uvedený výsledný návrh relační databáze pro potřeby vlastní komponenty. Návrh byl vytvořen v programu phpmyadmin. O správu uživatelů se stará přímo redakční systém Joomla!, proto tato tabulka users obsahuje další atributu, které nejsou popsány a uvedeny v postupu návrhu.

Při implementaci je nezbytné zajistit správnost a konzistentnost uložených dat neboli integritní omezení definovaná v předešlých krocích. Integritní omezení můžeme zajistit buď při vkládání dat do databáze, nebo naprogramovat v aplikaci, která s databází pracuje. Aplikace bude využívat druhý způsob a ve svém kódu bude hlídat a zajišťovat integritu dat.



Obrázek 9 - Návrh implementace relační databáze, zdroj: [vlastní]

4 PŘÍPRAVA K IMPLEMENTACI FULLTEXTOVÉHO VYHLEDÁVAČE

4.1 Server

Pro účel vývoje a testování byl pomocí program VirtualBox vytvořen virtuální LAMP (Linux, Apache, MySQL, PHP) server doplněný o další potřebné balíčky. Byly použity tyto technologie:

- VirtualBox 4.2.6
- Ubuntu 12.04 32b LTS
- Apache 2.2.22
- MySQL 5.5.29
- PHP 5.3.10
- ProFTPd 1.3.4a
- Netbeans 7.3

Konfigurace virtuálního počítače byla zvolena následující: 2 logická jádra procesoru Intel Core i5-2430M, 3GB RAM a 20GB HDD. Zvolen byl linuxový operační systém Ubuntu 12.04 32b z důvodů: minulých zkušeností, velké uživatelské základy a prodloužené podpory od vydavatele do dubna 2015. Verze Apache, MySQL a PHP byla instalována nejnovější dostupná v repozitářích pro daný systém. Další potřebná komponenta pro webový server je FTP server. Z velmi kladných minulých zkušeností byl použit ProFTPd.

Vývojové prostřední pro psaní PHP aplikace bylo zvoleno NetBeans z důvodu zkušeností z minulých projektů. Instalátor poslední stabilní verze byl stažen z http://netbeans.org/.

V příloze A je uveden podrobný popis instalace virtuálního serveru pro vývoj webové aplikace. Virtuální počítač byl zvolen z důvodu usnadnění práce budoucím programátorům. Dotyčný si nainstaluje Virtualbox z https://www.virtualbox.org, provede import virtuálního počítače a hned má přístup do celého vývojového prostředí, kde je daná aplikace plně funkční. Přístup do virtuálního počítače je:

Uživatel: ondrazap

Heslo: server2003

4.2 Redakční systém

Redakční systém byl vybrán Joomla!. Jedná se o typického představitele těchto systémů s velkou uživatelskou základnou.

Joomla! je redakční systém, obsahující stejnojmenný PHP Framework, který je distribuován pod licencí GNU GPL. Systém je rozdělen do dvou částí, na front-end a back-end. Front-end je část pro uživatele, kde je zobrazován obsah a back-end je část pro administrátora, kde se provádí veškeré nastavení a správa systému. [14]

Instalace redakčního systému Joomla! je velmi jednoduchá, proto nebude v této práci detailněji popsaná. Návod na instalaci redakčního systému v češtině je uveden např. zde http://www.website21.cz/joomla-2-5-instalace. Instalační baliček je k dispozici ke stažení na stránkách projektu http://www.joomla.org.

4.2.1 Možnosti rozšíření

Ve zdroji [7] jsou popsány jednotlivé možnosti rozšíření redakčního systému Joomla! spolu s návody jak dané rozšíření vytvořit.

Nejzákladnějším rozšířením jsou jazyky. Soubor s jazykovou mutací obsahuje dvojice klíč a hodnota. Tyto dvojice poskytnout překlad textových řetězců, které jsou použity ve zdrojovém kódu pomocí třídy JText. Jazykové balíčky se nastavují zvlášť na front-endu a back-endu.

Šablona představuje design webové stránky. Pomocí šablon můžeme měnit vzhled a definovat pozice pro zobrazování modulů a komponent.

Pluginy neboli zásuvné moduly obsluhují událostí. Naslouchají a čekají na určitou událost, při spuštění události registrované moduly pro tuto konkrétní událost se spustí. Např. pomocí pluginu můžeme definovat akci automatického přihlášení uživatele po registraci.

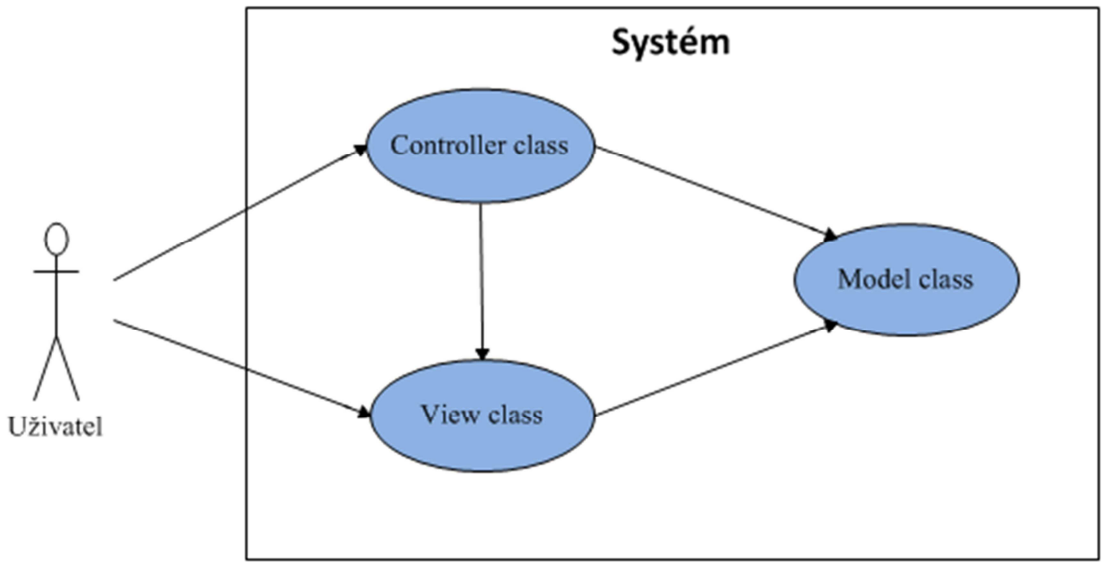
Modul je znám jako „krabice“, které jsou uspořádány kolem hlavního obsahu stránky. Moduly jsou určeny pro menu, přihlášení, zobrazování naposledy přidaných článků atd. Lze definovat, za jakých podmínek se mají zobrazit a kde se mají zobrazit na stránce. Např. v závislosti na tom jakou položku menu uživatel prohlíží, můžeme skrýt modul pro přihlášení.

Komponenta je největší a nejrozšířenější typ rozšíření pro redakční systém Joomla!. Většina komponent bývá rozdělena na dvě části a to front-end pro uživatele a back-end pro správce. Např. komponenta com_content se stará o zobrazení článků, které uživatelé

mohou prohlížet na front-endu a správce může upravovat obsah v back-endu. Komponenta využívá architekturu MVC pro psaní rozšíření.

4.2.2 Návrhový vzor MVC

Joomla! Framework využívá architekturu MVC, která dělí aplikaci do 3 logických celků tak, aby je šlo samostatně upravovat s co nejmenším dopadem na ostatní části. Jednotlivé části se nazývají model, view a controller a jsou zobrazeny na obrázku 10. Model obsahuje funkce pro práci s daty aplikace, view se stará o zobrazování uživatelského rozhraní a controller se stará o tok informací v aplikaci.



Obrázek 10 - Architektura MVC, zdroj: [7]

Při zpracování dat od uživatele získaných přes GET nebo POST je řízení předané controlleru. Controller stanoví, které modely se použijí pro splnění požadavku a které view se použije pro vrácení výsledků zpátky uživateli.

Model je součást komponenty, která zapouzdřuje práci s daty. Poskytuje funkce pro správu a úpravu dat. Např. aplikace využívá pro ukládání informací textový soubor a v nové verzi má využívat redakční databázi, veškeré změny se provedou v modelu bez nutnosti změn ostatních celků.

View využívá model, který mu je předán z controlleru pro získání dat. Data jsou následně upravena a prezentována uživateli. View data neupravuje, pouze zobrazuje data získaná z modelu v šabloně.

MVC architektura funguje tak, že uživatel otevře odkaz na komponentu bez odeslání dalších proměnných jako je úloha (task). Načte se výchozí controller, který zavolá výchozí

view nebo zavolá konkrétní úlohu. Při načtení view si načte data z modelu a zobrazí je uživateli.

4.3 Databáze pro test vyhledávání

Při hledání obsahu pro test fulltextového vyhledávání jsem narazil na velký problém a to, kde získat potřebná nejlépe reálná data pro testování vyhledávání. První možnost jsem zvolil oslovit cca 80 internetových obchodů emailem a požádat provozovatele o poskytnutí XML souboru se zbožím pro účel diplomové práce. Bohužel žádný z dotázaných provozovatelů nereagoval a zjistil jsem, že tudy cesta k získání dat nepovede.

Druhou možností jak připravit databázi pro vyhledávání bylo najít adresy XML souborů na internetu. Pomocí internetového vyhledávače http://www.google.cz byli nalezeny seznamy adres XML souborů se zbožím. Dané seznamy byly dány dohromady a výsledný seznam čítal 5718 adres XML souborů se zbožím různých internetových obchodů.

Ze seznamu byly namátkou otestované některé adresy XML souborů. Bylo zjištěno, že většina odkazů na XML soubory je chybných a k dnešnímu dni již nefunkčních. Pro projití tolika odkazů byla vytvořena komponenta (com_fts) pro redakční systém Joomla! zajišťující import zboží. Tato komponenta prošla celý seznam e-shopů s odkazy na XML soubory se zbožím a ke každému záznamu vytvořila log s informacemi o provedeném importu. Klíčová funkce této komponenty, která automaticky kontrolovala celý seznam odkazů na XML je níže uvedena. Cílem funkce bylo, každé adrese přiřadit její stav (1- v pořádku, -1 chyba při ukládání do databáze, -2 XML soubor má špatnou strukturu a -3 pro nefunkční adresu XML souboru). Tato funkcionalita je obsažena v modelu import.php. [5]

```
public function import(){
    //delete all products
    $this->deleteproducts();

    //get active shops
    $shops = $this->getActiveShops();

    //list all shops
    foreach ($shops as $shop){

        //(1)good,(-1)store error,(-2)bad structure,(-3)bad link
        $log=$this->initlog($shop);

        if(fopen($shop->feed_url, false)){
            //download data from xml
            $xml = simplexml_load_file($shop->feed_url);
            $temp = array();
```

```
            if (count($xml->SHOPITEM)){
                //products
                foreach($xml->SHOPITEM as $items){

                //get object from simpleXMLObject and add atributes
                $row            = $this->toObject($items);
                $row->shop_id   = $shop->id;
                $row->published   = 1;

                //count price_vat
                if(!$row->price_vat){
                    if ($row->vat>1) $row->vat=$row->vat/100;
                    $row->price_vat = $row->price * $row->vat;
                }
                //it is ok ?
                If($row->product<>"" && $row->description<>"" && $row->url<>"" &&
                $row->price_vat>0  ){
                    //product ok
                    $log->good++;
                } else {
                    //product error
                    $log->error++;
                }
                if (count($temp)==4000){
                    $log->state=$this->store($temp);
                    $temp = array();
                }
            }
            //store products
            $log->state=$this->store($temp);
          }else{
            //xml wrong structure
            $log->state=-2;
          }
        }else {
          //bad link
          $log->state=-3;
        }
        //store log
        $this->storelog($log);
      }
      return;
    }
```

Při řešení problému importu dat do systému byl zjištěn nedostatek redakčního systému Joomla!. Nedostatek spočívá v práci s databází při potřebě ukládat více řádků do databáze současně. Tento problém je označován jako architektura N+1 a v tomto případě spočívá v tom, že pro jeden e-shop systém načítá jeho produkty a následně by každý produkt ukládal samostatně do databáze. Při takovém to uložení se musí vždy načíst třída pro práci s dotazy a následně spustit dotaz. Problém je řešen ukládáním jednotlivých produktů do pomocného

pole, ze kterého je následně vytvořen sql dotaz pro vložení dat do databáze. Systém vkládá vždy 4000 položek najednou a při dojití nakonec XML souboru se zbožím vloží zbylé produkty do databáze. Není možné vložit všechny produkty do databáze najednou, protože u některých obchodů byl sql dotaz větší než 16 MB, což představuje omezení velikosti spustitelného dotazu proměnnou max_allowed_packet. Pro běh importu je potřeba více operační paměti než standardně nastavených 128 MB. V souboru s konfigurací php.ini byla upravena hodnota max memory size na 512 MB.

V aplikaci byl založen účet robot s heslem robot. Pod tímto účtem byl také vytvořen subjekt a následně automaticky importovány veškeré obchody s automaticky generovanými údaji s odkazem na XML soubor se zbožím. Import se spouští pomocí www adresy localhost/dp/index.php?option=com_fts&controller=import&task=import. Daným odkazem říkáme redakčnímu systému, aby načetl controller import a v něm spustil úlohu import. Daná úloha načte všechny e-shopy, které mají příznak published nastaven na 1. Vzhledem k použitému internetovému připojení s rychlostí downloadu 8 Mbps byl import časově velmi náročný cca 6 hodiny. Po importu tabulka dp_fts_products zabírala 3171,9 MB a obsahovala 7 918 064 záznamů z 1486 internetových obchodů. Internetové obchody, se špatnou adresou XML souboru se zbožím, byly smazány.

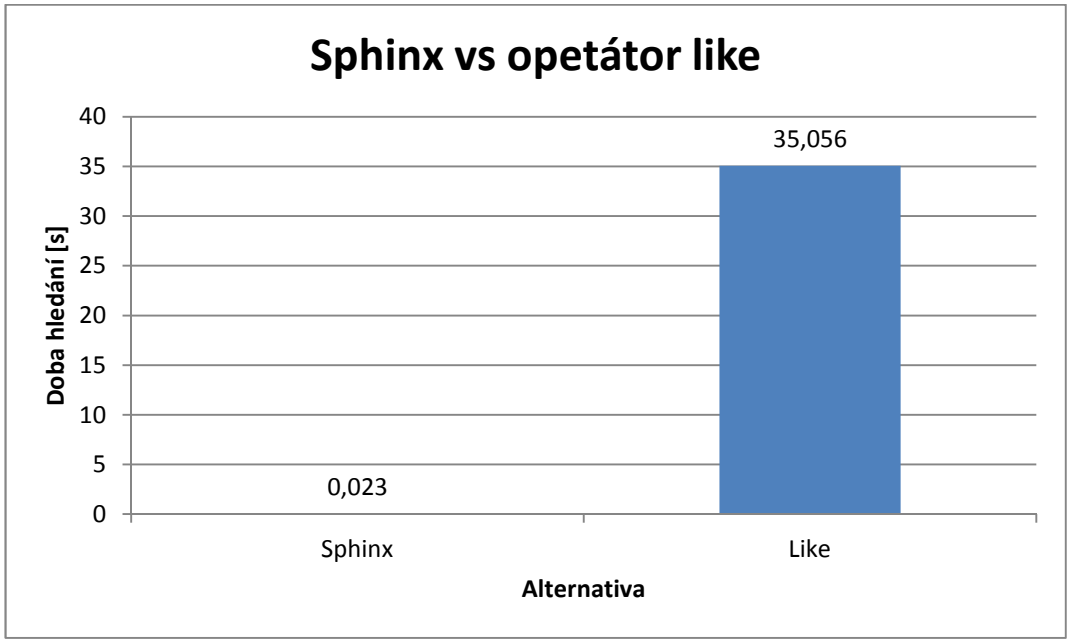
Databáze importovaných produktů obsahuje řádově miliony záznamů. Toto množství představuje dostatečný vzorek na projevení rozdílů různých fulltextových vyhledávačů.

Při kontrole záznamu logů pro jednotlivé e-shopy, bylo zjištěno, že v některých případech se část nabízeného zboží do systému nepřenese, protože neobsahuje jeden z povinných údajů.

5 FULLTEXTOVÉ VYHLEDÁVÁNÍ

Vyhledávání je v této aplikaci hlavní funkčnost, proto je jí věnována celá následující kapitola. Fulltextové vyhledávání je vyhledávání v textu pomocí klíčového slova či slov. Vyhledávání probíhá tak, že klíčové slovo je porovnáváno se slovy, která jsou v prohledávaném textu. Cílem této kapitoly je vybrat fulltextový vyhledávač pro vyhledávání v milionech záznamů, vyhledávání musí být použitelné z programovacího jazyka PHP a nesmí podléhat licenčním poplatkům za použití. [4], [11]

V MySQL databázi pro vyhledávání je možné použít také operátor like. Tento operátor hledá pouze přesnou shodu hledaného výrazu v textovém řetězci.



Obrázek 11 - Sphinx vs opetátor like, zdroj: [vlastní]

Na obrázku 11 jsou zobrazeny časy hledání výrazu „Nokia Asha“ v necelých osmi miliónech záznamů. Na první pohled je zřejmé, že operátor like je pro velké databáze velmi pomalý. Níže jsou uvedeny vybrané fulltextové vyhledávače, které jsou vhodné pro vyhledávání ve velkých databázích.

První alternativou je použití MySQL fulltext search, toto řešení bylo zvoleno z důvodů jednoduchosti použití, protože je obsaženo přímo v databázovém serveru. Druhou alternativou byl zvolen projekt Apache Solr, který využívá knihovnu Lucene a je použit například ve fulltextovém vyhledávači v internetu http://www.search.com. Poslední alternativou byl zvolen fulltextový vyhledávač Sphinx, protože jej využívá vyhledávač zboží http://www.zbozi.cz.

5.1 Alternativy vyhledávání

5.1.1 MySQL

MySQL je šířeno pod bezplatnou licencí GPL, nebo pod komerční licencí. Pro fulltextové vyhledávání se v MySQL používá fulltextový index (od verze 3.23.23). Tento index slouží pro prohledávání sloupců s datovým typem char, varchar a text. MySQL standardně vynechává všechna slovo, která mají méně než čtyři znaky. [4]

Důležité parametry pro úpravu chování jsou Ft_min_word_len, Ft_max_word_len a Ft_stopword_file.

Pro vyhledávání přirozeným jazykem v tabulce, kde se realizuje fulltextové vyhledávání, musí se vytvořit fulltextový index příkaz fulltext (název sloupce). Dále pro vyhledávání se používají dvě speciální funkce a to MATCH() a AGAINST(). Relevanci výsledků definujeme použitím funkce MATCH() v klauzuli WHERE SQL dotazu. Pak dotaz vrací seznam vážených hodnocení nalezených řádků. Čím vyšší skóre, tím vyšší relevance. [4]

Select description from jos_title MATCH(description) AGAINST (‘Apache2‘);

Kromě slov, která jsou definována jako příliš často se vyskytující a nemohou být tak brána jako relevantní se také ignorují slova, která se vyskytují ve více než 50% záznamů. To může být velký problém. Například, pokud uživatelé uvedou jedno slovo ve většině záznamů a podle tohoto slova hledáme, pak fulltextové vyhledávání nenajde žádný záznam. Vyhledávání přirozeným jazykem neumožňuje také použít operátory pro upřesnění výsledků. [4]

Druhý způsob hledání v MySQL je boolovské fulltextové vyhledávání nabízí jemnější kontrolu nad vyhledáváním. Umožňuje pomocí operátorů definovat, jaká slova mají a nemají být obsažena ve výsledcích, jestli mají být přítomna všechny klíčová slova, nějaké nebo žádné nemá být přítomno. Pro úpravu výsledků můžeme použít operátory. Níže je uveden sql dotaz, který vrátí řádky obsahující slovo Web a neobsahující slovo Joomla. [4]

Select description from jos_title where match(popis) against (‘+Web -Joomla‘ in boolean mode);

Pro správu MySQL lze použít terminálové připojení nebo pro snazší správu je k dispozici velké množství aplikací jako je např. PhpMyAdmin.

5.1.2 Apache Solr

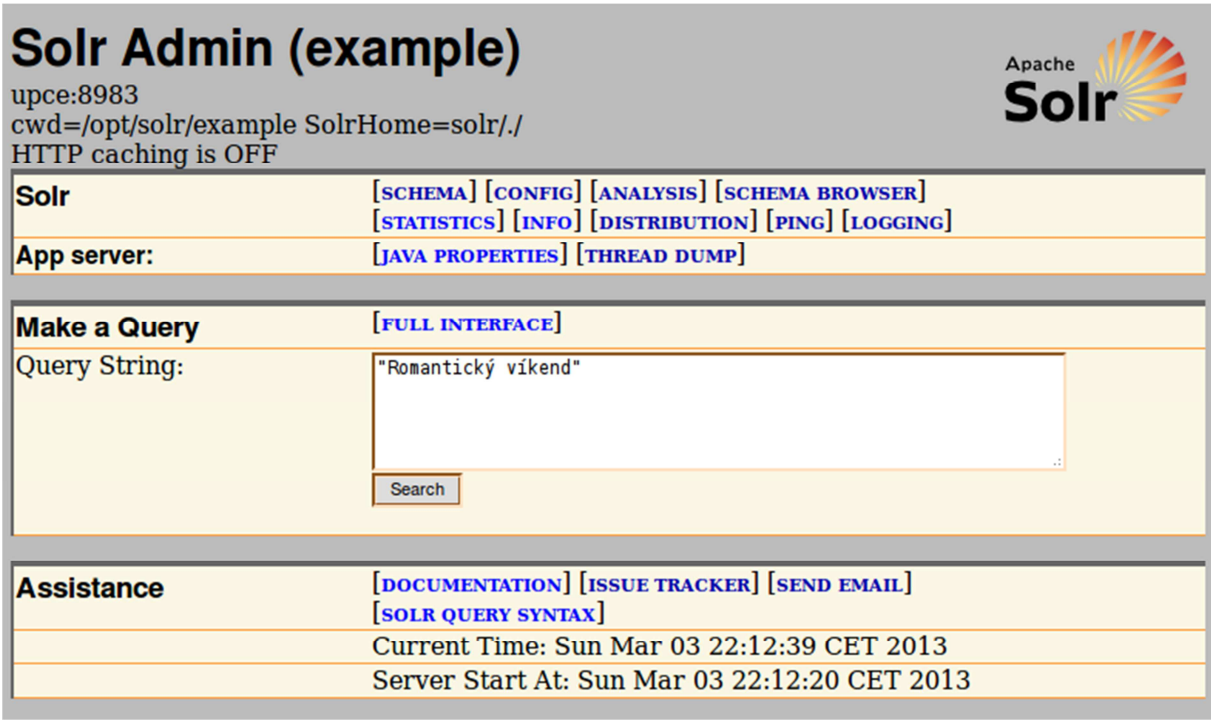
Apache Solr je webová aplikace, která k indexování a k fulltextovému vyhledávání využívá knihovnu Lucene. Tato webová aplikace využívá licenci Apache License 2.0. Apache Solr i Lucene jsou psané v programovacím jazyce Java a díky tomu jsou použitelné na různých platformách. V roce 2010 byl jejich vývoj spojen. Knihovna Lucene poskytuje pouze funkce pro vyhledávání a není možné jí použít samostatně. Zajímavostí tohoto řešení je použití pro indexování dokumentů, např. PDF. O kvalitách tohoto projektu svědčí jeho použití agenturou NASA ve svých projektech jako je např. NEBULA cloud computing platform nebo NASA Planetary Data System. [1]

Primárně je Lucene určené pro programy psané v Javě. Pro ostatní programovací jazyky existují porty (C#, C++,PHP, Delphi…). Bohužel, porty vždy zaostávají za hlavní vývojovou větví.

V současné době jsou k dispozici dvě verze 4.1 a 3.6. Novější verze trpí v současné době problémy s ovladačem k MySQL databázi, z tohoto důvodu byla použita verze starší. Pro zprovoznění Apache Solr bylo potřeba stáhnout z http://lucene.apache.org/solr/ balíček s archívem, ten rozbalit do adresáře. Pro běh Apache Solr je potřeba mít nainstalovanou javu.

Archív s Apache Solr 3.6 byl rozbalen do adresáře /opt. Pro konfiguraci se používají 2 základní soubory(schema.xml, solrconfig.xml) a jeden soubor obsahující definici tabulky s daty v MySQL (data-config.xml). Konfigurační soubory jsou uložené v /opt/solr/example/solr/conf.

Apache Solr je webová aplikace, k její správě se využívá webové rozhraní, ze kterého se spouští jednotlivé dotazy (viz obrázek 12).



Obrázek 12 - Rozhraní Apache Solr, zdroj: [vlastní]

5.1.3 Sphinx

Sphinx je fulltextový vyhledávací nástroj, veřejně šířeny pod GPL verze 2. Jedná se o samostatný softwarový balíček napsaný v C++ poskytující rychlé fulltextové vyhledávání klientským aplikacím. Sphinx běží jako démon (searchd) na pozadí, se kterým se komunikuje přes TCP socket pomocí vlastního protokolu. Dále je obsažena utilita search, která se dá využít v terminálu. Sphinx nabízí API pro několik programovacích jazyků včetně PHP a je též podporována většina operačních systémů. [13]

Pro vyhledávání je potřeba prvně vytvořit index pro vyhledávání. K tomu slouží program indexer, který se připojí k databázi a pomocí SQL dotazu získá potřebná data, která jsou následně indexována a uložena. Výsledky vyhledávání dostaneme stejně jako z relační databáze, to velmi usnadňuje další zpracování výsledků. [13]

Instalace byla provedena stažením instalačního balíčku poslední stabilní verze z webové stránky http://www.sphinxsearch.com. Jde o verzi 2.0.6, následně byl balíček z terminálu nainstalován.

```
wget http://sphinxsearch.com/files/sphinxsearch_2.0.6-release-0ubuntu11~lucid_i386.deb
dpkg -i sphinxsearch_2.0.6-release-0ubuntu11~lucid_i386.deb.1
```

Po dokončení instalace byla provedena základní konfigurace Sphinx nezbytná pro napojení na tabulku s produkty. Konfigurace se provádí v konfiguračním souboru /etc/sphinxsearch/sphinx.conf. V základní konfiguraci bylo změněno následující.

```
##připojení k mysql
sql_host        = localhost
sql_user        = root
sql_pass        = root
sql_db          = dp
sql_port        = 3306

##dotaz
sql_query_pre   = SET NAMES utf8
charset_type    = utf-8
sql_query       = SELECT id,product,description FROM dp_fulltext
sql_query_info  = SELECT * FROM dp_fulltext WHERE id=$id
charset_tablet  = 0..9, A..Z->a..z, _, a..z, U+410..U+42F->U+430..U+44F, U+430..U+44F, U+C0..U+D6-
                  >U+E0..U+F6, U+D8..U+DE->U+F8..U+FE, U+178->U+FF, U+FF, U+100..U+177/2,
                  U+179..U+17E/2
min_word_len    = 3
```

Jak je dle výše uvedené tabulky patrné, první je potřeba nastavit připojení k MySQL serveru a následně nastavit dotaz pro vyhledávání. Velmi důležitá je proměnná charset_table, protože Sphinx ve výchozím nastavení indexuje jen anglické a ruské znaky, proto byli přidány znaky latinky a nastaveno kódování na utf-8..

Pro prvotní test funkčnosti vyhledávání přímo z terminálu je potřeba prvně vytvořit vyhledávácí index příkazem „indexer --rotate --all a následně pomocí příkazu „search technologie“ otestovat vyhledávání.

5.2 Kritéria rozhodování

Kritéria pro porovnání fulltextového vyhledávání byly převzaty z [21].

- čas vytvoření indexu pro vyhledávání (K1),
- velikost indexu pro vyhledávání (K2),
- čas zpracování hledání fráze (K3),
- čas zpracování dotazu s booleovským operátorem (K4),
- čas zpracování dotazu (K5).

Saaty doporučuje pro vyjádření velikosti preference mezi kritérii použít bodovou stupnici (tabulka 11). Stanovme rozpětí 1 až 9, které odpovídá tomu, že nejvýznamnější kritérium je 9x významnější než nejméně významné kritérium [3].

Tabulka 11 - Saatym doporučená bodová stupnice, zdroj: [3]

Body	Významnost
1	jsou stejně důležitá
3	první kritérium je slabě významnější než druhé
5	první kritérium je dosti významnější než druhé
7	první kritérium je dosti významnější než druhé
9	první kritérium je absolutně významnější než druhé

Po určení kritérií je nutno určit pořadí jejich významnosti mezi sebou. Pořadí kritérií, které bylo zvoleno, je uvedené v tabulce 12. [3]

Tabulka 12 - Významnost jednotlivých kritérii, zdroj: [vlastní]

Kritérium	Významnost
K5	nejvýznamnější
K4	více významné
K3	významné
K1	nevýznamné
K2	nejvíce nevýznamné

Hodnoty kritérií pro Apache Solr byly zjištěny z webového rozhraní, kde po zadání dotazu bylo zobrazeno XML s odpovědí na daný dotaz. Hodnota kritéria 2 byla zjištěna jako velikost složky, obsahující soubory s indexem.

Tabulka 13 - Hodnoty kritérii pro Apache Solr, zdroj: [vlastní]

Kritérium	Zjištění hodnoty kritéria	Hodnota
K1	Otevření adresy pro spuštění importu http://localhost:8983/solr/dataimport?command=full-import a následně průběh zobrazen na adrese http://localhost:8983/solr/dataimport	27 m 25, 992 s
K2	Velikost složky /opt/solr/example/index	2633795 KB
K3	“Romantický víkend“	76 ms
K4	+romantický +víkend -vysočina	25 ms
K5	romantický víkend vysočina	65 ms

Hodnoty kritérií pro MySQL byly zjištěny po přihlášení z konzole jako odpovědi na SQL dotazy. Kromě kritéria 2, kde velikost indexu byla určena z uloženého souboru obsahující fulltextový index na disku. Sql dotazy jsou uvedeny v tabulce 14.

Tabulka 14 - Hodnoty kritérii pro MySQL, zdroj: [vlastní]

Kritérium	SQL dotaz	Hodnota
K1	ALTER TABLE `dp`.`dp_fts_products` ADD FULLTEXT `products` (`product`,`description`);	36 m 31,96 s
K2	Velikost souboru var/lib/mysql/dp_fts_products.myi	1750 MB
K3	SELECT product, description, (MATCH(product,description) AGAINST ('romantický víkend')) AS search_priority FROM dp_fts_products ORDER BY `search_priority` DESC	27,3554 s
K4	SELECT product FROM `dp`.`dp_fts_products` WHERE MATCH (product, description) AGAINST ('+romantický +víkend -vysočina' IN BOOLEAN MODE)	0,01 s
K5	SELECT product, MATCH (product, description) AGAINST ('romantický víkend vysočina') AS Relevance FROM `dp`.`dp_fts_products`WHERE MATCH (product, description) AGAINST ('romantický víkend vysočina' IN NATURAL LANGUAGE MODE) ORDER BY Relevance DESC	0,0964 s

Hodnoty kritérií pro Sphinx byly zjištěny z terminálu a PHP aplikace. Hodnota pro K1 a K2 byla zjištěna jako odpověď z terminálu. Pro ostatní kritéria byla zjištěna hodnota z PHP aplikace, kde se měnil hledaný výraz a způsob shody.

Tabulka 15 - Hodnoty kritérii pro Sphinx, zdroj: [vlastní]

Kritérium	Zjištění hodnoty kritéria	Hodnota
K1	indexer --all	348,664 s
K2		2013,6 MB
K3	$search = 'Romantický víkend Vysočina'; $sp->SetMatchMode(SPH_MATCH_PHASE);	0,001 s
K4	$search = '+romantický +víkend -vysočina'; $sp->SetMatchMode(SPH_MATCH_BOOLEAN);	0,004 s
K5	$search = 'Romantický víkend Vysočina'; $sp->SetMatchMode(SPH_MATCH_ANY);	0,003 s

U všech alternativ při opakovaném měření hodnot kritéria K1, se hodnota kritéria mírně měnili v řádu desítek milisekund. Hodnota kritéria K2 byla vždy stejná. Hodnoty kritérií K3, K4 a K5 vrátily při opakovaném spuštění vždy nulový čas trvání zpracování. Z tohoto důvodu byl u kritérií K3-K5 zaznamenán první čas zpracování výsledku dotazu. V tabulce 16 jsou uvedeny hodnoty kritérií pro všechny alternativy.

Tabulka 16 - Alternativy řešení, zdroj: [vlastní]

Kritéria	MySQL	Sphinx	Apache Solr
K1 - čas vytvoření indexu pro vyhledávání (s)	2191,960	348,664	1645,992
K2 - velikost indexu pro vyhledávání (MB)	1750,000	2013,600	2572,060
K3 - čas zpracování hledání fráze (ms)	27355,000	1,000	76,000
K4 - čas zpracování SQL dotazu s booleovským operátorem (ms)	10,000	4,000	25,000
K5 - čas zpracování dotazu (ms)	96,400	3,000	65,000

5.3 Řešení vícekriteriálního problému

V kapitole 5.1 a 5.2 je popsán rozhodovací problém se třemi alternativami a pěti kritérii. Při řešení rozhodovacího problému jsou jednotlivé alternativy ohodnoceny na základě daných kritérií. Cílem rozhodování je vybrat alternativu, která je podle daných kritérií ohodnocena nejlépe. Pro výpočet vah jednotlivých kritérií existuje několik metod. [10]

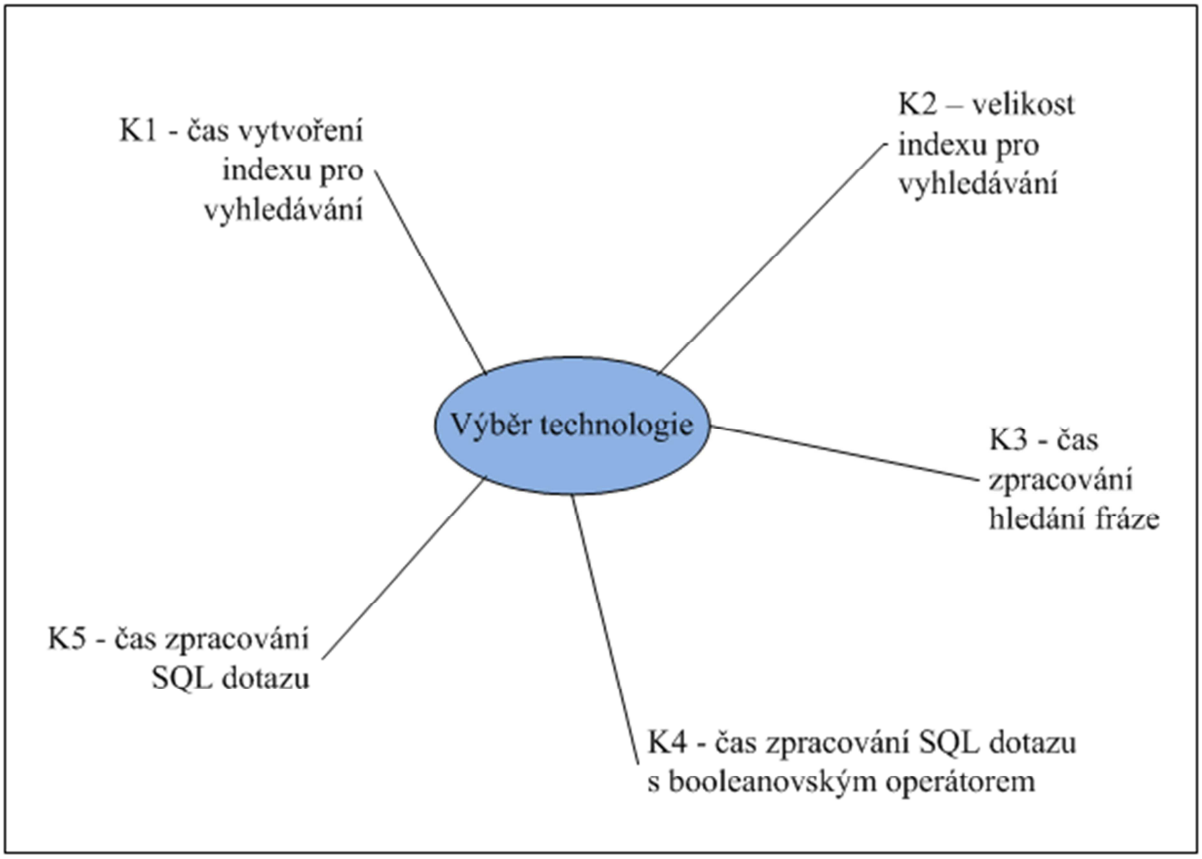
Pro vyjádření preferencí se využívá tzv. Fullerův trojúhelník, proto je metoda někdy označována jako Fullerova metoda párového porovnání. V nejjednodušší modifikaci metody párového porovnání se pro každé kritérium zjišťuje počet jeho preferencí vzhledem ke všem ostatním kritériím souboru. V trojúhelníkové matici rozhodovatel u každé dvojice kritérií zjišťuje, zda preferuje kritérium uvedené v řádku před kritériem uvedeným ve sloupci. V případě preference označí hodnotu 1, v opačném případě 0. Pro každé kritérium je následně stanoven počet jeho preferencí, který je roven součtu jednotek v řádku uvažovaného kritéria zvětšenému o počet nul ve sloupci tohoto kritéria. Na základě počtu preferencí jednotlivých kritérií se stanoví váhy. Metoda má dvě nevýhody. První nevýhoda spočívá v tom, že pokud počet preferencí daného kritéria je roven nule, bude jeho váha rovna nule i přesto, že dané kritérium nemusí být bezvýznamné. Druhá nevýhoda spočívá v tom, že můžeme pouze určit směr preference a ne velikost preference jednotlivých kritérií. [10]

Saatyho metodu stanovení vah kritérií lze rozdělit do dvou kroků. První krok je analogický metodě párového srovnávání, kdy se opět zjišťují preferenční vztahy dvojic kritérií uspořádaných v tabulce, v jejíchž řádcích i sloupcích jsou zapsána kritéria ve stejném pořadí. Na rozdíl od metody párového srovnávání se však kromě směru preference dvojic kritérií určuje také velikost této preference, která se vyjadřuje určitým počtem bodů ze zvolené bodové stupnice. Výsledkem tohoto kroku je získání matice velikostí preferencí. Hrubé odhady vah kritérií získáme sečtením prvků v každém řádku Saatyho matice a vydělíme

je součtem všech prvků této matice. Stanovené podíly pro jednotlivé řádky představují pak odhady vah odpovídajících kritériím. [10]

Metoda AHP bere v úvahu všechny prvky, které ovlivňují výsledek analýzy, vazby a intenzitu, s jakou na sebe vzájemně působí. Prvním krokem je definice rozhodovacího problému pomocí hierarchické struktury. V druhém kroku je definována Saatyho matice párových porovnávání pro všechna kritéria. Ve třetím kroku je vypočítáno největší vlastní číslo λmax, vektor vlastních čísel σ, výpočet výsledného váhového vektoru w a stanovení konzistenčního indexu CI. Ve čtvrtém kroku se vypočítá vektor normovaný maximalizačních a minimalizačních kritérií G. Pátým krokem je opakování kroků 3 a 4, kdy je spočítána hodnota v_{ij} jako část matice vlastních vektorů V. Posledním krokem je výpočet nejlepší alternativy. [10]

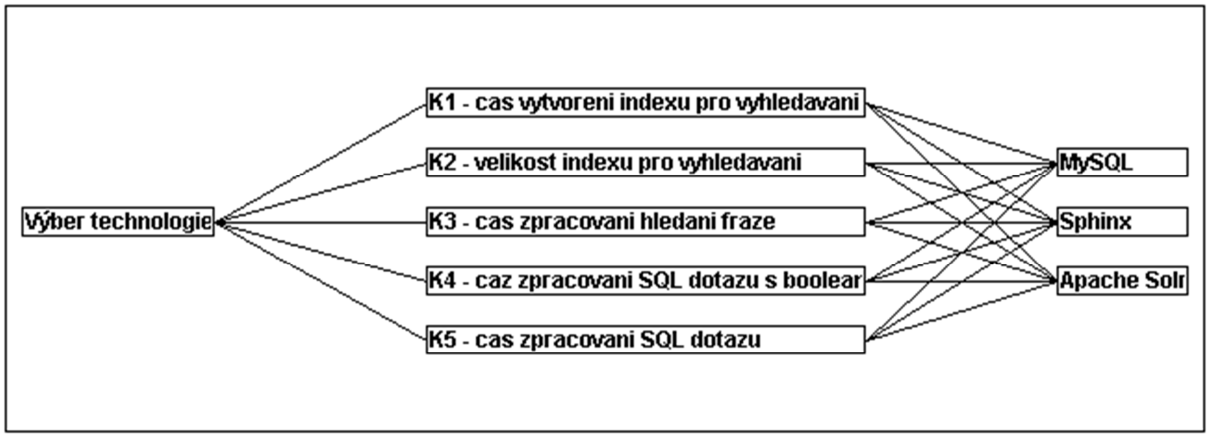
Pro řešení daného problému byl zvolen software Criterium Decision Plus a metoda AHP (Analytický hierarchický proces) [10]. Jako první byl vytvořen brainstormingový model daného rozhodovacího procesu (viz obrázek 13).



Obrázek 13 - Brainstormingový model rozhodovacího procesu, zdroj: [vlastní]

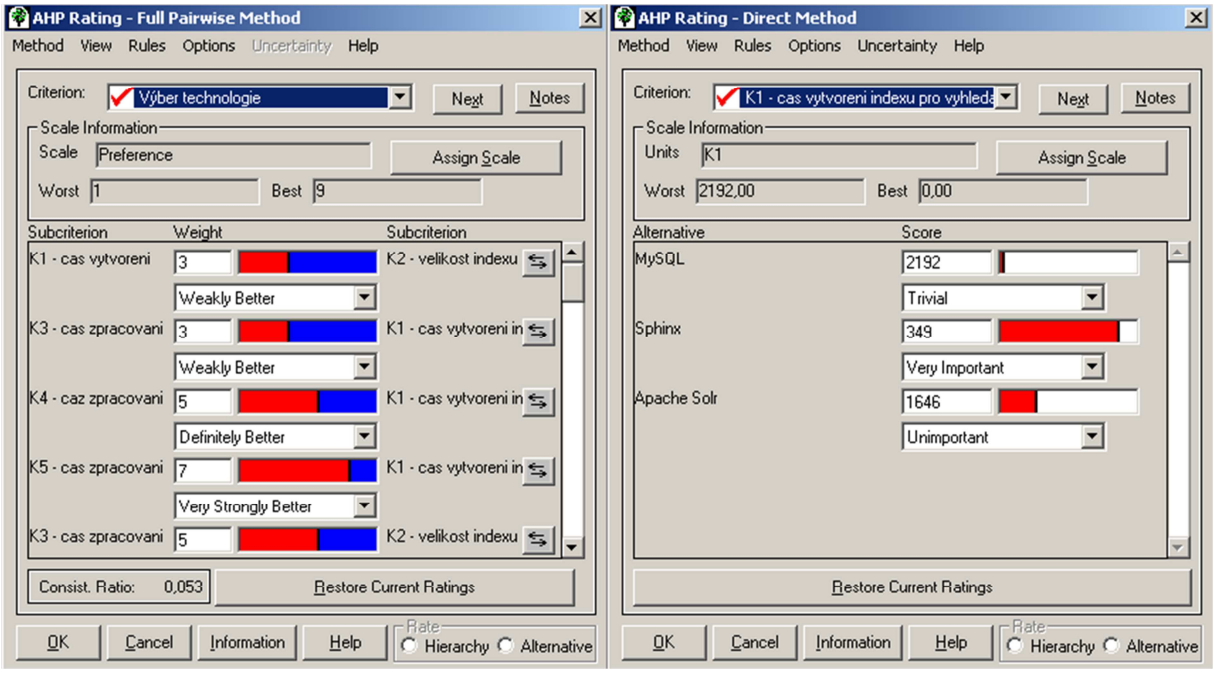
Brainstormingový model byl následně převeden do modelu hierarchické struktury (viz obrázek 14), na tomto obrázku vidíme vlevo cíl rozhodování, uprostřed kritéria a vpravo

jsou alternativy.



Obrázek 14 - Model hierarchické struktury, zdroj: [vlastní]

Dále byli do zvoleného sw zapsány vzájemné váhy kritérií a též i váhy pro jednotlivé alternativy daného kritéria (viz obrázek 15).



Obrázek 15 - Nastavení vah, zdroj: [vlastní]

Na obrázku 15 v pravém dialogovém okně je uveden konzistenční poměr Saatyho matice 0,053. Tento index nám říká, že Saatyho matice je sestavena správně. V levé části obrázku 15 je vidět způsob zadávání hodnot pro jednotlivá kritéria, zde konkrétně pro kritérium 1. Pro každé kritérium byla vytvořena samostatná stupnice s nejlepší hodnotou 0 a s nejhorší hodnotou rovnou maximální daného kritéria.

5.4 Ověření řešení

Pro ověření řešení je spočítán index konzistence v sw Matlab. Dle velikosti preferencí je sestavena Saatyho matice označená *S*, která je definována pomocí kvantitativních párových srovnání s_{ij} [10].

Vzorec pro výpočet prvků matice S:

$$s_{ij} \approx \frac{v_i}{v_j}, \text{pro } i,j = 1,2,\dots,m. \tag{5.1}$$

s_{ij} – prvek matice

v_i – váha *i*-tého (řádkového) kritéria

v_j – váha *j*-tého (sloupcového) kritéria

m – počet kritérií

Pokud je kritérium v řádku významnější než kritérium ve sloupci, zapíše se do příslušného pole počet bodů (preference kritéria v řádku vzhledem ke kritériu ve sloupci). Naopak, pokud je kritérium ve sloupci významnější než kritérium v řádku, zapíše se převrácená hodnota počtu bodů V tabulce 17 je uvedena Saatyho matice pro daný rozhodovací problém. [3]

Pro prvky Saatyho matice platí následující vzorce [10]:

$$S_{ii} = 1, \text{pro } i = j \in \{1,2,\dots,m\} \tag{5.2}$$

$$S_{ij} = \frac{1}{s_{ji}} \text{ pro } i \neq j \in \{1,2,\dots,m\} \tag{5.3}$$

m – počet kritérií

i – index řádkového kritéria

j – index sloupcového kritéria

s_{ij} – prvek matice

Tabulka 17 - Saatyho matice, zdroj: [vlastní]

Kritérium	K1	K2	K3	K4	K5
K1	1	3	1/3	1/5	1/7
K2	1/3	1	1/5	1/7	1/9
K3	3	5	1	1/3	1/5
K4	5	7	3	1	1/3
K5	7	9	5	3	1

Nejdříve byly hodnoty matice zadány do programu Matlab. Následně pomocí funkce EIG byla získána matice vlastních vektorů a matice vlastních čísel. Matice vlastních čísel má čísla pouze na diagonále a největší číslo představuje λ_{max} (největší vlastní číslo Saatyho matice), pomocí kterého je spočítaná hodnota dle vzorce (5.4) hodnota konzistenčního indexu *CI*. Následně je spočítána dle vzorce (5.5) hodnota konzistenčního poměru *CR*.

Index konzistence *CI* byl spočítán dle vzorce:

$$CI = \frac{\lambda_{max} - m}{m - 1} \tag{5.4}$$

CI – index konzistence

m – počet kritérií

λ_{max} – největší vlastní číslo matice

Konzistenční poměr *CR* byl spočítán dle vzorce:

$$CR = \frac{CI}{RI} \tag{5.5}$$

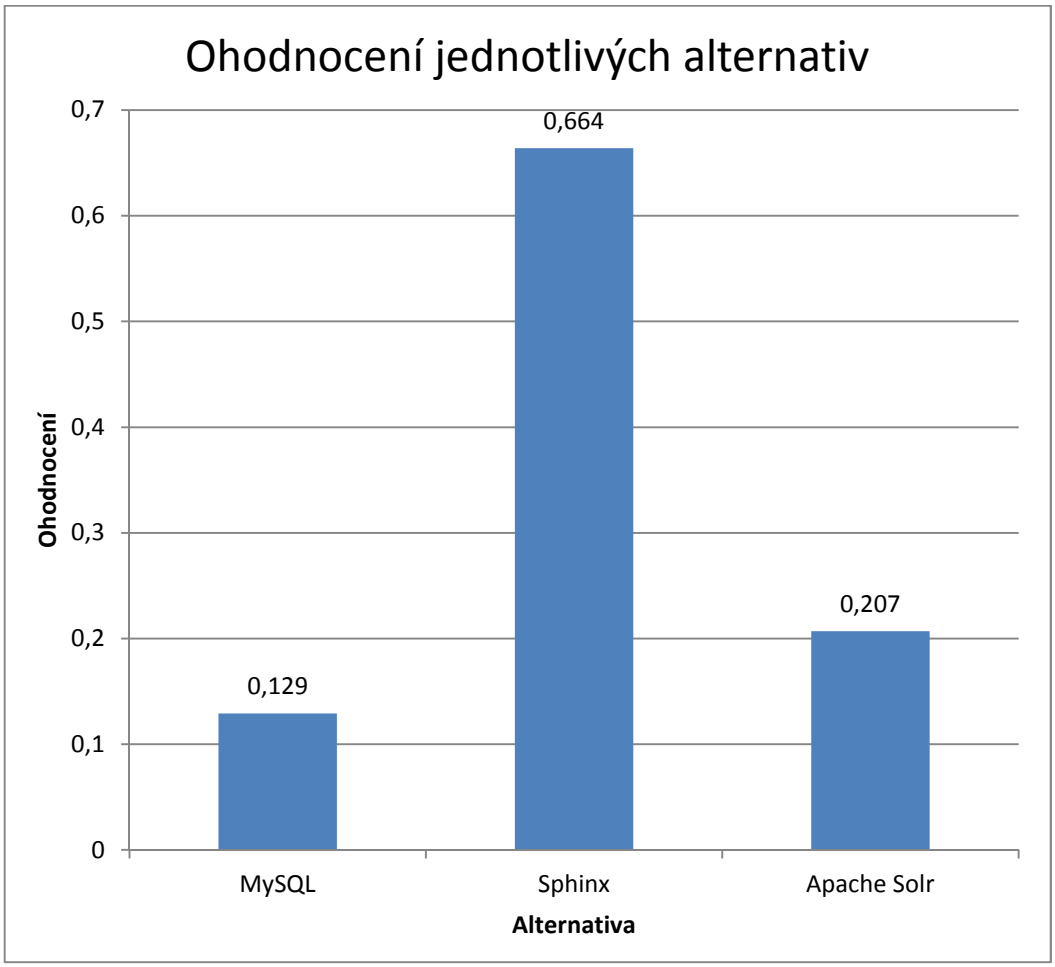
CI – konzistenční index

RI – hodnota náhodného konzistenčního indexu udávané v tabulkách

CR – konzistenční poměr

Konzistenční poměr *CR* dle sw Matlab je 0,0530. Přesný postup celého výpočtu v sw Matlab je uveden v příloze B.

5.5 Závěr výběru fulltextového vyhledávače



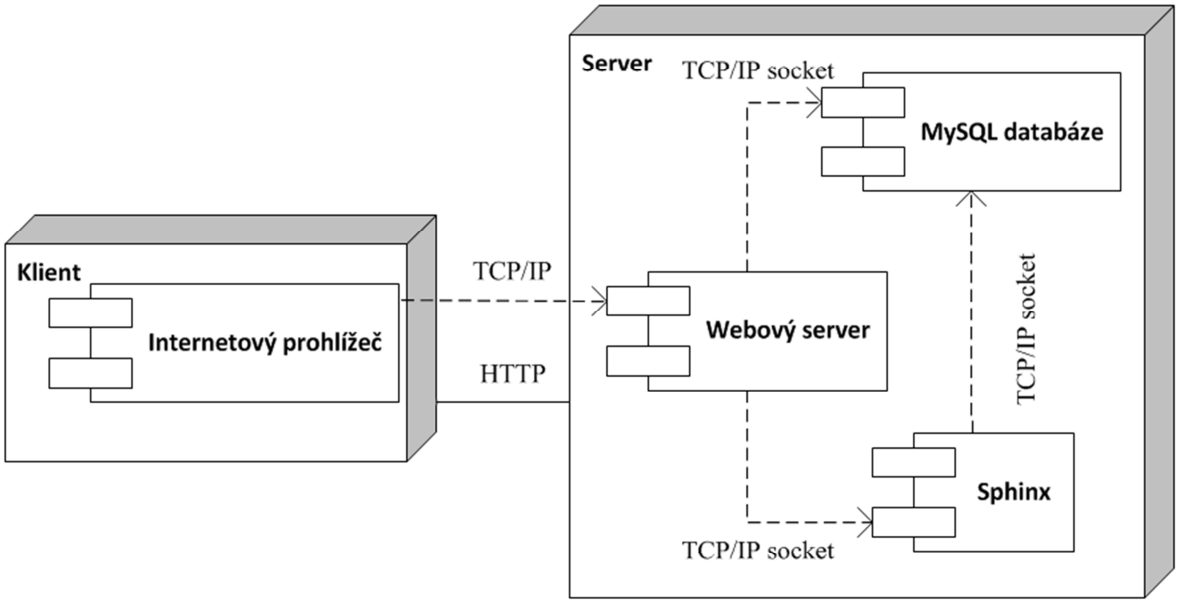
Obrázek 16 - Ohodnocení alternativ, zdroj: [vlastní]

Na obrázku 16 je zobrazeno ohodnocení jednotlivých alternativ. Nejvyšší ohodnocení dosáhl fulltextový vyhledávač Sphinx, z těchto alternativ je to nejlépe hodnocená alternativa.

Konzistenční poměr *CR* vyjadřuje, do jaké míry jsou subjektivní párová ohodnocení Saatyho matice konzistentní, neboli do jaké míry si uživatel v hodnocení protiřečil či nikoliv. Dle programů Criterium Decision Plus a Matlab je v obou případech konzistenční poměr *CR* roven 0,053. Saatyho matice byla sestavena správně.

6 IMPLEMENTACE FULLTEXTOVÉHO VYHLEDÁVAČE

Na základě závěru kapitoly 5 byl pro implementaci vybrán fulltextový vyhledávač Sphinx. Dle [2] pro zobrazení způsobu, jakým bude daný sw nasazen na fyzickém hardware se používá diagram nasazení.



Obrázek 17 - Diagram nasazení, zdroj: [vlastní]

Z obrázku 17 je patrné, že webový server Apache, databáze MysQL a vyhledávač Sphinx jsou nasazeni na jednom serveru a komunikace mezi nimi probíhá pomocí TCP/IP socketu. Komunikace mezi klientem, kde je spuštěný internetový prohlížeč a serverem probíhá pomocí protokolu HTTP.

6.1 Konfigurace Sphinx

Instalace a základní konfigurace byla popsána v kapitole 5. Tato konfigurace byla následně upravena. První změna se týká nastavení zdrojů pro indexování. Zdroj byl rozdělen na dva. První zdroj indexuje název produktu (product). Ukázka nejdůležitější části prvního zdroje je následně uvedena.

```
sql_query_pre    = SET NAMES utf8

sql_query        = SELECT id,product,price_vat FROM dp_fts_products

sql_field_string = product
sql_attr_uint    = price_vat
```

Druhý zdroj se liší v dotazu, který se posílá do databáze, kde je získáván sloupec obsahující popis produktu (description). Tyto dva zdroje jsou následně indexovány.

Index je nezbytný pro rychlé vyhledávání ve velkém množství dat. Data jsou předpracovány a uloženy do speciální datové struktury zvané index. Index obsahuje většinou sufixové a prefixové vyhledávací stromy a hashovací tabulky. S rostoucím objemem dat pro indexaci se zvětšují nároky na úložný prostor pro samotné soubory s indexy. První index indexuje první zdroj, takže indexuje jméno produktu. Druhý index indexuje popis produktu. Zde je uvedena ukázka konfigurace druhého indexu. Indexy se liší použitým zdrojem.

```
index product_2
{
    source          = src2
    path            = /var/lib/sphinxsearch/data/product_2
    docinfo         = extern
    mlock           = 0
    morphology      = stem_cz
    stopwords       = /var/lib/sphinxsearch/data/stopwords_cz.txt
    min_word_len    = 3
    charset_type    = utf-8

    # charset definition and case folding rules "table"
    charset_table = 0..9, A..Z->a..z, _, a..z, U+410..U+42F->U+430..U+44F, U+430..U+44F,\
    U+C0..U+D6->U+E0..U+F6, U+D8..U+DE->U+F8..U+FE, U+178->U+FF, U+FF,\
    U+100..U+177/2, U+179..U+17E/2,U+DD->y, U+FD->y, \
    U+C0->a, U+C1->a, U+C2->a, U+C3->a, U+C4->a, U+C5->a, \
    U+E0->a, U+E1->a, U+E2->a, U+E3->a, U+E4->a, U+E5->a, \
    U+C8->e, U+C9->e, U+CA->e, U+CB->e, \
    U+E8->e, U+E9->e, U+EA->e, U+EB->e, \
    U+CC->i, U+CD->i, U+CE->i, U+CF->i, \
    U+EC->i, U+ED->i, U+EE->i, U+EF->i, \
    U+D2->o, U+D3->o, U+D4->o, U+D5->o, U+D6->o, \
    U+F2->o, U+F3->o, U+F4->o, U+F5->o, U+F6->o, \
    U+D9->u, U+DA->u, U+DB->u, U+DC->u, \
    U+F9->u, U+FA->u, U+FB->u, U+FC->u

    html_strip      = 1

}
```

Indexy pro podporu českého jazyka využívají streaming. Tato funkce pro skloňované, odvozené nebo časované slovo vrací jeho kořen. Cílem je odstranit morfologické koncovky a předpony. Před vlastním vyhledáváním se takto ošetří hledané slovo a teprve pak se testuje shoda. Problémem jsou ale slova, která při skloňování mění svůj tvar. Pro použití streamingu byla doplněna tabulka charset table.

Konfigurace obou indexů byla doplněna o seznam slov stopwords, ve kterém jsou uvedena slova, která se nemají indexovat (jde o slova bez vlastního obsahu, např. spojky či předložky). Sphinx nenabízí na svých stránkách seznam českých slov, seznam stopwords byl použit z projektu Apache Solr.

Vyhledávající index při hledání slova ohodnotí výsledek číselnou hodnotu, které se říká relevance. Dva zdroje a 2 indexy byly použity z toho důvodu, aby bylo možné nastavovat jejich váhy zvlášť pro název produktu a zvlášť pro jeho popis.

Soubor s konfigurací je uložený v /etc/var/sphinxsearch/gedit.conf. Indexování databáze produktu se spouští příkazem indexer --config /etc/sphinxsearch/sphinx.conf --rotate –all. Pro test funkčnosti vyhledávání je možné otestovat hledání z terminálu. Vyhledávání má mnoho parametrů, které lze zjistit spuštěním příkazu search.

6.2 Použití Sphinx z PHP

Pro použití vyhledávání z PHP byl vytvořen model search v komponentě com_fts. Konfigurace byla provedena dle zdroje [13]. Pro vlastní vyhledávání je nutné prvně vložit PHP třídu zajišťující komunikaci s vyhledávacím démonem searchd. Druhý potřebný údaj je číslo portu, na kterém naslouchá vyhledávací démon searchd, port je uveden v konfiguračním souboru sphinx.conf. PHP kód pro připojení vypadá následovně.

```
// insert sphinx API class
require('/usr/shere/sphinxsearch/api/sphinxapi.php');

// connect to sphinx
$sp = new SphinxClient();
 $sp->SetServer('localhost', 9312);

//set match mode
$sp->SetMatchMode(SPH_MATCH_ALL);
```

V druhém kroku je nastaven typ shody pro vyhledávání. Typ shody se nastavují pomocí funkce SetMatchMode(). Výchozí režim shody je v aplikaci nastaven na SPH_MATCH_EXTENDED2. V tabulce 18 jsou popsány použité metody shody.

Tabulka 18 - Sphinx - možnosti režimu shody, zdroj: [14]

Metoda shody	Popis
SPH_MATCH_ALL	Obsahuje všechny dotazované slova
SPH_MATCH_ANY	Obsahuje některá dotazovaná slova
SPH_MATCH_EXTENDED2	Odpovídá vnitřnímu dotazovacímu jazyku Sphinx

S metodou shody SPH_MATCH_EXTENDED2 je možné zvolit metodu výpočtu relevance. Sphinx nabízí několik metod výpočtu, výpočty pracují se dvěma faktory. První faktor LCS představuje blízkost slov (významově) a druhý faktor BM25 představuje frekvenci nalezených slov. Byla zvolena doporučená metoda výpočtu relevance SPH_RANK_PROXIMITY_BM25, která využívá oba zmíněné faktory. Ostatní metody výpočtu jsou popsány v dokumentaci. [13]

Řazení výsledků je nastaveno funkcí setSortMode. Výchozí řazení je určeno řazení výsledků dle relevance. V aplikaci je také implementováno řazení dle ceny vzestupně a sestupně. Např. řazení dle ceny vzestupně se nastaví takto: SetSortMode(SPH_SORT_EXTENDED,"price_vat DESC").

Poslední omezující parametr, který je vhodné nastavit je omezení počtu vyhledaných záznamů na stránku. K tomuto slouží funkce SetLimits() s parametry, říkající od jakého čísla po jaké číslo má vrátit záznamy. Uživatel musí mít možnost zobrazit si všechny vyhledané relevantní výsledky. K řízení stránkování je využita funkce getPagination(), její parametry jsou celkový počet nalezených záznamů, od jakého záznamu má zobrazit výsledky a kolik záznamů má být zobrazeno na stránku. Tato funkcionalita je obsažena přímo v redakčním systému Joomla! a její parametry jsou předávány funkci SetLimits().

Hledání je spuštěno pomocí funkce Query($search, 'product_1 product_2'), kde první parametr představuje zadaný hledaný výraz. Druhý parametr obsahuje vyjmenované indexy, ve kterých je hledáno. Pro metody shody SPH_MATCH_ALL a SPH_MATCH_ANY jsou jednotlivým indexům přiřazeny váhy a omezena dolní hranice výsledné relevance pro vrácení záznamů.

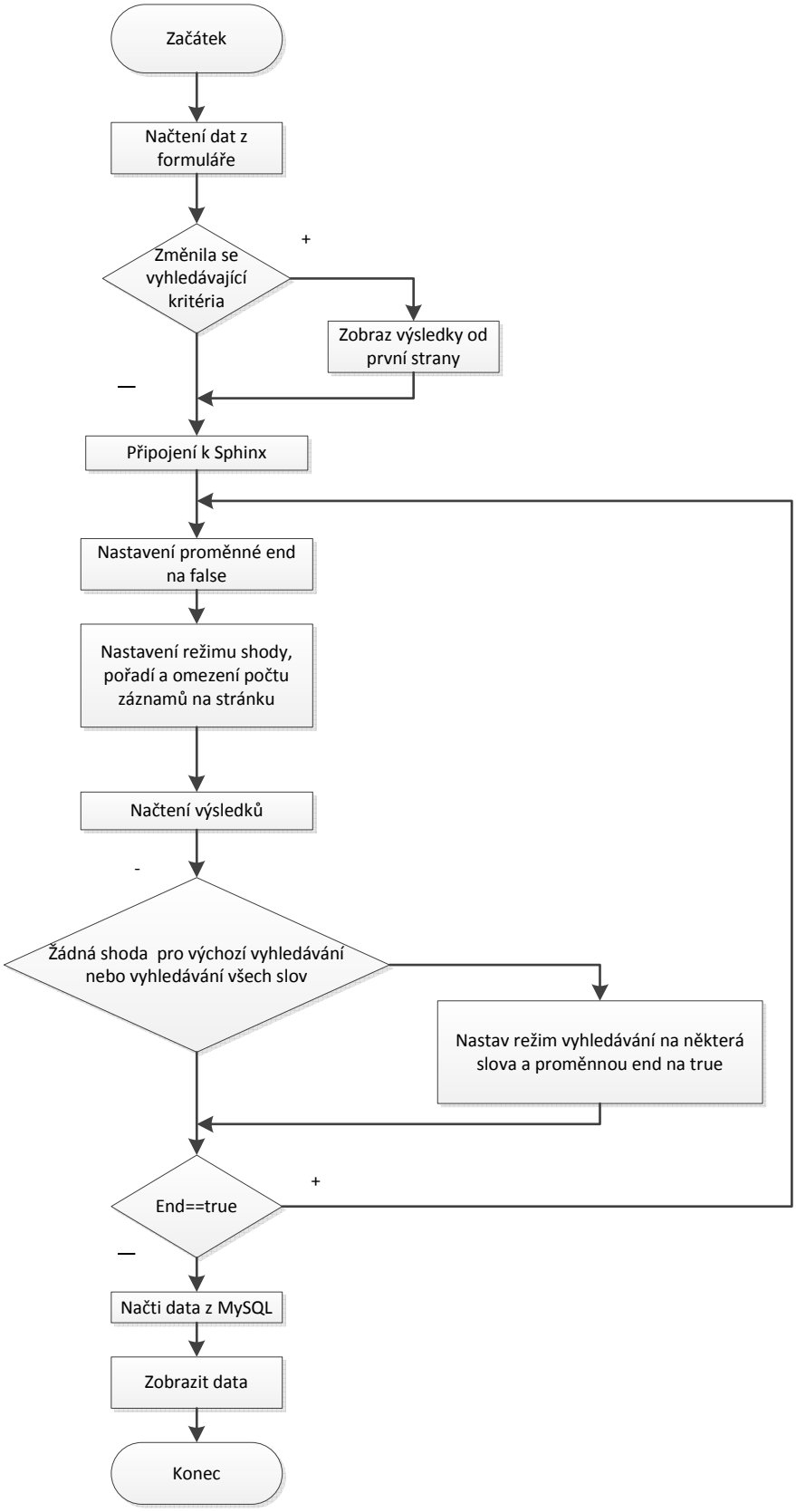
Výsledkem funkce Query() je pole obsahující pouze ID záznamů jednotlivých položek z databáze. Sphinx nabízí také možnost vyhledávání pomocí real time indexu, kdy je vyhledávající index aktualizován při každé změně MySQL databázi. Toto vyhledávání je v dokumentaci doporučováno pro tabulky nepřesahující půl milionu záznamu. Pro větší tabulky již jeho rychlost vyhledávání velmi klesá. Celé řádky z databáze nelze v našem případě získat přímo (pracujeme s milióny záznamů), k získaným ID záznamů získáme data pomocí tohoto kódu.[5]

```
//get ids
$idlist = array();
foreach ( $data["matches"] as $row ) {
        $idlist[] = $row["id"];
}
$ids = implode(", ", $idlist);
```

```
//connect to db
$db       = $this->getDbo();
$query   = $db->getQuery(true);

// Select the required fields from the table.
$query->select(
                        $this->getState(
                                'list.select',
                                'a.id,a.product,'.
                                'a.url,a.imgurl,a.delivery_date,a.price_vat,'.
                                'a.published'
                        )
            );
//from
$query->from($db->quoteName('#__fts_products').' AS a');

//where
$query->where('a.id IN ('.$ids.') AND a.published=1');

//order by
$orderCol="";
$orderDirn='FIELD(a.id,'.$ids.')';
$query->order($db->escape($orderCol.' '.$orderDirn));

return $query;
```

Pro získání záznamu z MySQL databáze ve stejném pořadí, v jakém je seřadil Sphinx je použito ORDER BY FIELD. Uvedená funkce vrátí vytvořený dotaz, který je zpracován funkcí getItems().

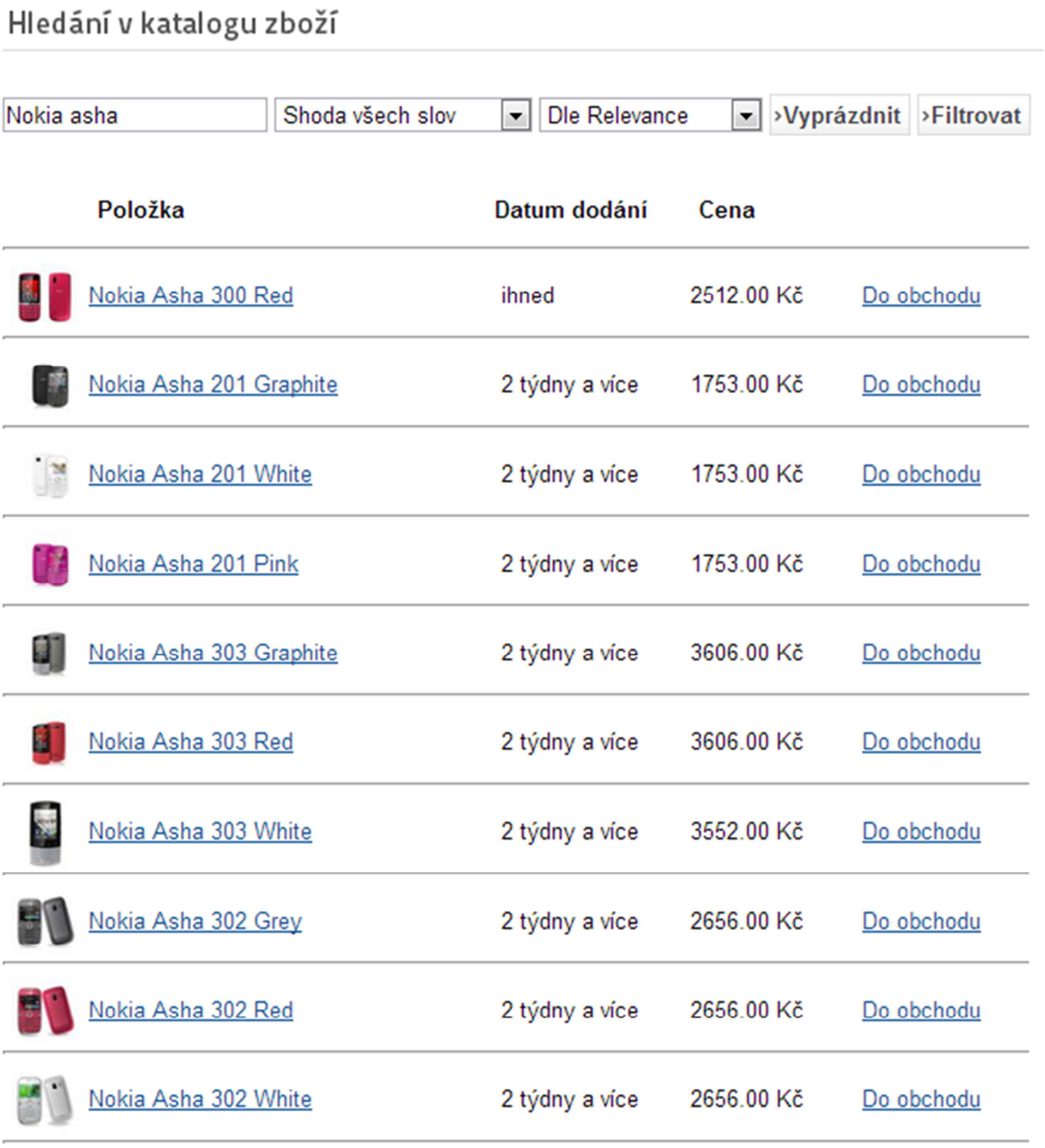
Na obrázku 18 je uveden použitý algoritmus fulltextového vyhledávání. Data z formuláře pro vyhledávání jsou načtena. Pokud se od minulého hledání změnily, dojde k přepnutí zobrazování výsledků na první stránku. Pokud je použit výchozí model shody SPH_MATCH_EXTENDED2 nebo je zvolena shoda všech slov SPH_MATCH_ALL a nebude nalezená žádná shoda. Systém přepne vyhledávání na SPH_MATCH_ANY a hledá se shoda některých slov. V příloze E je uveden vlastní PHP kód vyhledávání sestavený na základě algoritmu (viz obrázek 18).



Obrázek 18 - Algoritmus fulltextového vyhledávání, zdroj: [vlastní]

6.3 Otestování fulltextového vyhledávače

Pro testování fulltextového vyhledávače bylo vytvořeno view search v komponentě com_fts, které zobrazuje výsledky vyhledávání, viz obrázek 18.



Obrázek 19 - Aplikace pro testování fulltextového vyhledávání, zdroj: [vlastní]

V tomto view je zobrazen filtr pro nastavení parametrů fulltextového vyhledávání a zadání hledaného řetězce. Popis parametrů fulltextového vyhledávaní je uveden v kapitole 6.2. Ve výpisu je zobrazen obrázek produktu, název, datum dodání, cena a odkaz na daný produkt přímo do e-shopu na jeho stránku s detailem.

Ve výpisu jsou zahrnuty pouze produkty, které mají nastaven atribut published na 1. Tato podmínka je zahrnuta z toho důvodu, aby administrátor mohl vypnout zobrazování nabídek e-shopů, které porušují obchodní podmínky nebo se jinak provinili.

Aktualizace probíhá pomocí importu zboží z adresy obsahující odkaz na XML soubor se zbožím a následně reindexací vyhledávajících indexů. Rychlost importu je velmi závislá na rychlosti připojení k internetu. Testovací databáze se stahovala několik hodin na neagregovaném připojení s rychlostí stahování 8 Mbps.

Při testu aplikace a náhodném hledání byly ve výsledkách zjištěny nesrovnalosti. Na první pohled se zdálo, že systém špatně zpracovává cenu produktu, při důkladném prozkoumání problému, byla zjištěna chyba v XML souboru (viz obrázek 19).

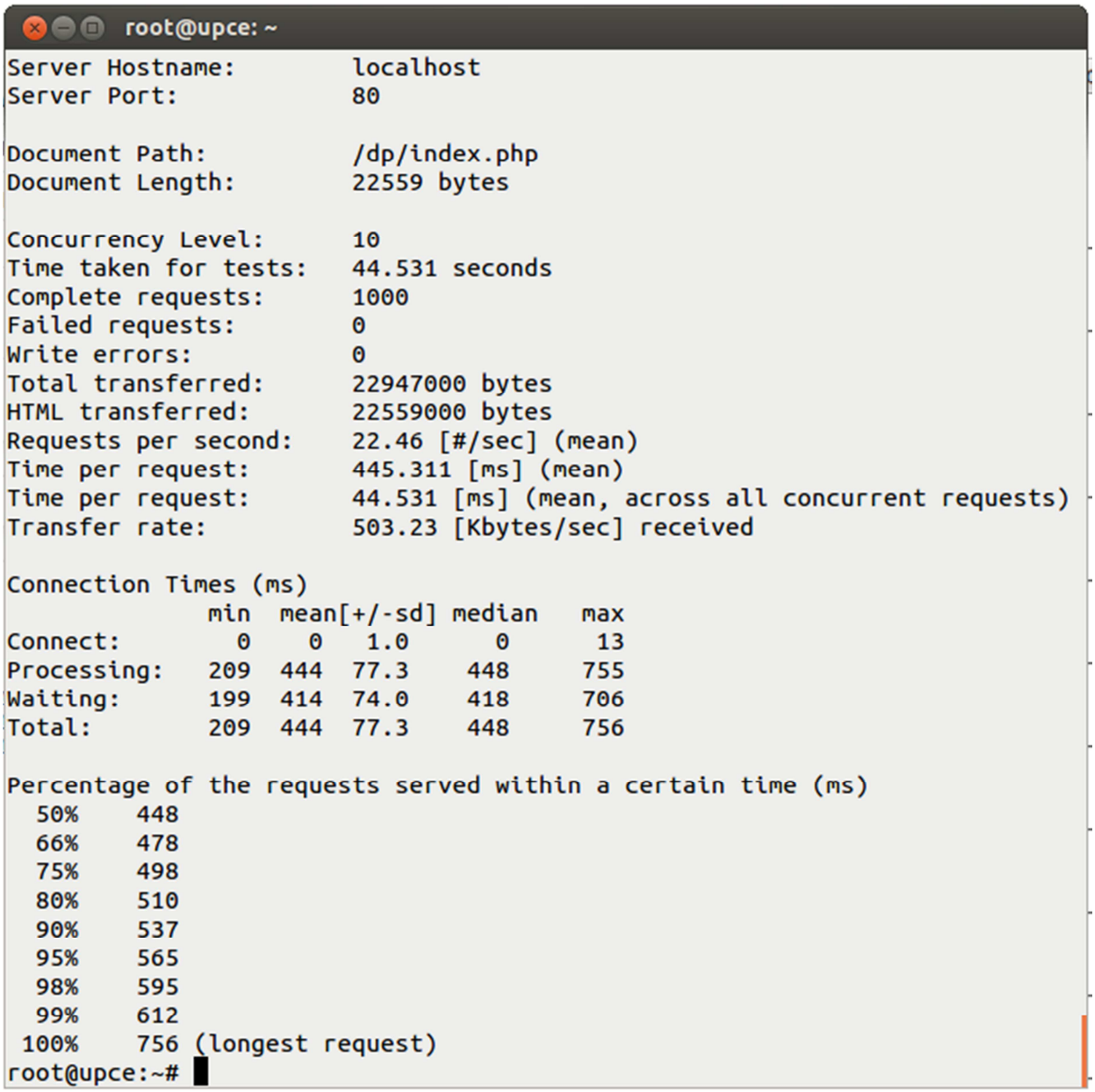
```
▼<SHOPITEM>
  <MANUFACTURER>Philips</MANUFACTURER>
 ▼<PRODUCT>
    Holící frézka Philips HQ 5/50 pro strojky Philishave
  </PRODUCT>
 ▼<DESCRIPTION>
    Holící frézka Reflex action pro strojky:<br>HQ5401, HQ5421, HQ5425,
    HQ5461, HQ5465, HQ5601, HQ5699, HQ5801, HQ5806, HQ5811, HQ5814,
    HQ5815, HQ5816, HQ5821, HQ5825, HQ5826, HQ5830, HQ5841, HQ5845,
    HQ5846, HQ5848, HQ5849, HQ5850, HQ5851, HQ5853, HQ5854, HQ5
  </DESCRIPTION>
  <DUES>0.0</DUES>
  <AVAILABILITY>48</AVAILABILITY>
  <SHOP_DEPOTS>Veletrzni</SHOP_DEPOTS>
 ▼<URL>
    http://www.holici-strojky.com/?page=shop/flypage&product_id=371
  </URL>
  <PRICE VAT>86900</PRICE VAT>
  <DELIVERY>6</DELIVERY>
  <CATEGORY>Philips náhradní díly</CATEGORY>
 ▼<IMGURL>
    http://www.holici-
    strojky.com/shop_image/product/4206f2a3aaff996a0a552f6f5c1dac68.jpg
  </IMGURL>
</SHOPITEM>
```

Obrázek 20 - Chybná hodnota v XML, zdroj: [vlastní]

V daném e-shopu je cena uvedena správně. Takovýchto chyb je v testovacích datech mnoho. Chyby jsou způsobeny tím, že jsou importovány XML soubory z již neaktivních e-shopů.

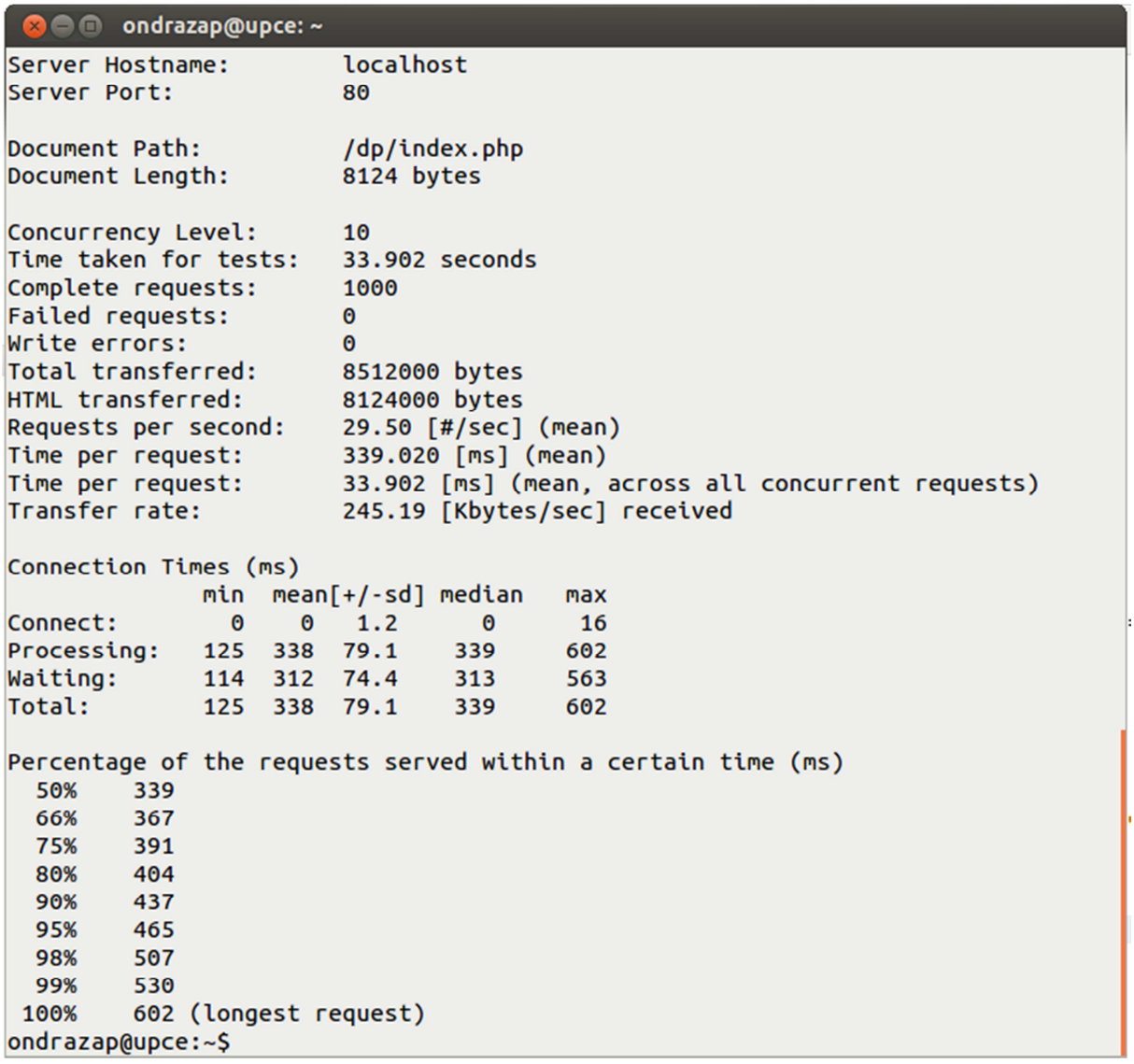
6.4 Testování výkonu webového serveru

Výkon webového serveru byl změřen pomocí nástroje Apache http server benchmarking tool, zkráceně AB. Nástroj je součástí webového serveru Apache, zasílá požadavky na danou stránku a tím simuluje její zátěž. Nástroj se spouští v terminálu, pomocí samotného příkazu ab je zobrazena nápověda. Pro první test byla zvolena stránka s výpisem hledaných produktů, výsledek je zobrazen na obrázku 21. Hledaný řetězec není možné zadat přímo do www adresy, proto byl zadán přímo do funkce zajišťující hledání. Hledáno bylo slovo „Nokia“. Test byl spuštěn s parametry –c 10 –n 1000. Parametry představují 10 paralelních spojení a 1000 představuje celkový počet zaslaných požadavků.



Obrázek 21 - Výsledky testu domovské stránky s vyhledáváním produktu, zdroj: [vlastní]

Druhý test byl spuštěn pro stránku výpisů subjektů. Test vyžadoval dvě úpravy komponenty. První úprava spočívá v odstranění podmínek pro zobrazení přehledu subjektu bez nutnosti přihlášení. Druhá úprava spočívala v nastavení přehledu subjektů jako výchozí stránky. Nástroj pro test výkonu neumožnil zadat adresu se dvěma parametry (parametr pro komponentu a pro view). Test byl spuštěn se stejnými parametry jako předešlý test, výsledky jsou zobrazeny na obrázku 22.



Obrázek 22 - Výsledky testu přehledu subjektů, zdroj: [vlastní]

V prvním testu bylo vyřízeno 22 požadavků za sekundu a ve druhém 29 požadavků za sekundu. Webová aplikace běží na virtuálním serveru, který je spuštěn na obyčejném notebooku, tento výsledek je odpovídající. 10 paralelních spojení je dostatečné pro desítky uživatelů, protože jednotlivé požadavky nejsou odesílány bezprostředně za sebou.

Samotný výkon aplikace není možné jednoznačně změřit. SQL dotazy i při produkčním množství dat trvají v řádu ms a v SQL dotazech není použita N+1 architektura.

7 POUŽITÍ REALIZOVANÉHO FULLTEXTOVÉHO VYHLEDÁVAČE

7.1 Instalace

Vytvořená aplikace má několik závislostí. Pro běh je nezbytné nainstalovat tyto balíčky Apache, PHP, MySQL, Joomla! a Sphinx. Po instalaci nezbytného software by následovala konfigurace a instalace samotné aplikace. Toto by bylo velmi časově náročné. Z tohoto důvodu byla aplikace odevzdána jako virtuální počítač, kde je aplikace funkční. Pro spuštění samotného virtuálního počítače je potřebné nainstalovat virtualizační program VirtualBox. Instalační soubor je na přiloženém DVD, doporučuji vždy používat aktuální verzi programu z důvodu větší stability a rychlosti. Aktuální verze je k dispozici ke stažení na stránkách projektu https://www.virtualbox.org. Virtuální počítač se do programu VirtualBox naimportuje spolu s nastavením. Virtuální počítač má přidělené 2 výpočetní jádra a 3 GB operační paměti.

Vytvořenou aplikaci lze také nahrát do již fungujícího redakčního systému Joomla!. Postup instalace je uveden na přiloženém DVD.

7.2 Omezení

Z důvodu omezení volného místa na DVD je v aplikaci funkční pouze fulltextový vyhledávač Sphinx. Ostatní fulltextové vyhledávače MySQL Fulltext a Apache Solr byly proto odstraněny. V databázi aplikace je cca sto internetových obchodů a cca milión importovaných produktů.

7.3 Spuštění

Pří spuštění virtuálního počítače je nutné zadat uživatelské jméno (ondrazap) a heslo (server2003). V aplikaci je necelých 400 e-shopů a necelé dva milióny produktů. Data jsou zaindexována. Po přihlášení a spuštění webového prohlížeče je front-end aplikace dostupný na adrese http://localhost/dp a back-end je dostupný na adrese http://localhost/dp/administrator. Součástí virtuálního počítače je i vývojové prostření, uživatel může aplikaci jednoduše upravovat.

V aplikaci je založen účet admin s heslem server2003.

ZÁVĚR

Cílem této práce bylo popsat a porovnat možnosti vyhledávání zboží v nabídkách e-shopů. Dalším cílem bylo identifikovat nedostatky ve stávajícím řešení, navrhnout řešení daného nedostatku a implementace v redakčním systému Joomla!.

V části zabývající se porovnáním vyhledávačů zboží je definován problém. Problém se týká hledání zboží, které není možné jednoznačně identifikovat. Jedná se vyhledávání zážitků jako je např. pobyt v lázních nebo romantický víkend. Takovéto zboží nemá čárový kód ani ISBN pro jednoznačnou identifikaci. Výsledky při hledání by měli zůstat čistě fulltextové a v tomto případě se to nedaří. Fulltextové hledání je následně uživateli nabídnuto.

Kapitola 5 se zabývá popisem jednotlivých možností fulltextových vyhledávání. Jsou popsány 3 alternativy a definováno 5 kritérií pro porovnání. Testy jsou provedeny na vzorku dat obsahující necelých 8 milionů záznamů s produkty, což představuje dostatečný vzorek k projevu rozdílu ve výkonu jednotlivých zvolených alternativ. Dané alternativy jsou metodou AHP porovnány a nejvyšší ohodnocení získal fulltextový vyhledávač Sphinx.

Pro implementaci fulltextového vyhledávače Sphinx v prostřední redakčního systému Joomla! byly popsány funkční a nefunkční požadavky na systém a proveden návrh databáze. Následně byla vytvořena komponenta pro redakční systém com_fts, která splňuje dané požadavky. V komponentě com_fts je implementován fulltextový vyhledávač Sphinx pro vyhledávání v databázi produktu. Implementace je popsána v kapitole 6.

Na přiloženém DVD jsou zdrojové kódy aplikace a vyexportovaný virtuální počítač, který obsahuje funkční vyhledávač zboží, který je možné použít jako základ a dále rozšířit o další funkcionalitu. Lze tedy konstatovat, že cíle práce uvedené v zadávacím listě byly splněny.

Vytvořená komponenta řeší pouze základní funkce vyhledávače zboží. Vyhledávače zboží jsou dnes provozovány s vidinou zisku. Proto v budoucím vývoji je nutné vytvořit obchodní model a definovat služby, které zajistí rentabilitu provozu aplikace. Mezi takové služby může patřit přednostní výpis v kategoriích e-shopů, zobrazování cílené reklamy nebo cena za proklik z katalogu zboží přímo do e-hopu, které zboží nabízí.

POUŽITÁ LITERATURA

[1] Apache Solr. APACHE SOFTWARE FOUNDATION. *Apache Lucene - Apache Solr* [online]. 2013 [cit. 2013-03-04]. Dostupné z: http://lucene.apache.org/solr/

[2] ARLOW, Jim a Ila NEUSTADT. *UML a unifikovaný proces vývoje aplikací: průvodce analýzou a návrhem objektově orientovaného softwaru*. Vyd. 1. Brno: Computer Press, 2003, xviii, 387 s. ISBN 80-7226-947-X.

[3] FOTR, Jiří, Lenka ŠVECOVÁ, Jiří DĚDINA, Helena HRŮZOVÁ a Jiří RICHTER. *Manažerské rozhodování: postupy, metody a nástroje*. Vyd. 1. Praha: Ekopress, 2006, 409 s. ISBN 80-869-2915-9.

[4] GILMORE, Jason W. *Velká kniha PHP 5 MySQL: kompendium znalostí pro začátečníky i profesionály*. Vyd. 1. Brno: Zoner Press, 2005, 711 s. ISBN 80-868-1520-X.

[5] HUB, Miloslav. *Technologie internetu - PHP 5: distanční opora*. Vyd. 1. Pardubice: Univerzita Pardubice, 2009, 88 s. ISBN 978-80-7395-163-4.

[6] HUB, Miloslav, Renáta MYŠKOVÁ a Pavel JIRAVA. *Technologie internetu - XML: distanční opora*. Vyd. 1. Pardubice: Univerzita Pardubice, 2010, 84 s. ISBN 978-80-7395-306-5.

[7] Joomla! Official Documentation. OPEN SOURCE MATTERS. *Joomla! Documentation* [online]. 2013 [cit. 2013-04-29]. Dostupné z: http://docs.joomla.org/

[8] KADLEC, Václav. Agilní programování: metodiky efektivního vývoje softwaru. 1. vyd. Brno: Computer Press, 2004, 278 s. ISBN 80-251-0342-0.

[9] KENNARD, James. *Mastering Joomla! 1.5 : Extension and Framework Development*. Birmigham: Packt publishing, 2007. 2007th edition. ISBN 978-1-84719-282-0.

[10] 22x22 KŘUPKA, Jiří, Miloslava KAŠPAROVÁ a Renáta MÁCHOVÁ. *Rozhodovací procesy* [online]. 2012. vyd. Univerzita Pardubice Fakulta ekonomicko-správní. ISBN 978-80-7395-478-9. Dostupné z: http://rozhodovaciprocesy.cz.

[11] LACKO, Ľuboslav. PHP 5 a MySQL 5: hotová řešení. Vyd. 1. Překlad Ondřej Gibl. Brno: Computer Press, 2007, 320 s. ISBN 978-80-251-1695-1.

[12] MLÝNKOVÁ, Irena a Jaroslav POKORNÝ. *XML technologie: principy a aplikace v praxi. 1*.vyd. Praha: Grada, 2008, 267 s. Průvodce (Grada). ISBN 978-80-247-2725-7.

[13] PHP API Documentation. SPHINX TECHNOLOGIES INC. *Sphinx Wiki* [online]. 2013 [cit. 2013-04-15]. Dostupné z: http://sphinxsearch.com/wiki/doku.php

[14] RAHMEL, Dan. Joomla!: podrobný průvodce tvorbou a správou webů. Vyd. 1. Překlad Ondřej Gibl. Brno: Computer Press, 2010, 382 s. ISBN 978-80-251-2714-8.

[15] Služby pro obchody. MITON MEDIA, a.s. *Služby obchodům - Heureka.cz* [online]. 2013 [cit. 2013-01-06]. Dostupné z: http://sluzby.heureka.cz

[16] Specifikace XML pro internetové obchody. SEZNAM.CZ A.S. *Seznam Nápověda* [online]. 2013 [cit. 2013-04-29]. Dostupné z: http://napoveda.seznam.cz/cz/specifikace-xml.html

[17] Specifikace XML souboru. MITON MEDIA, a.s. *Služby obchodům - Heureka.cz* [online]. 2013 [cit. 2013-01-06]. Dostupné z: http://sluzby.heureka.cz/napoveda/xml-feed

[18] ŠIMONOVÁ, Stanislava a Jan PANUŠ. *Databázové systémy I: pro kombinovanou formu studia*. Vyd. 1. Pardubice: Univerzita Pardubice, 2007, 106 s. ISBN 978-80-7194-988-6.

[19] ŠIMONOVÁ, Stanislava, Renáta MYŠKOVÁ a Pavel JIRAVA. *Projektování informačních systémů - UML, procesní řízení: pro kombinovanou formu studia*. Vyd. 1. Pardubice: Univerzita Pardubice, 2006, 114 s. ISBN 80-719-4895-0.

[20] Všeobecné obchodní podmínky. SEZNAM.CZ, a.s. *Seznam Nápověda* [online]. 2013 [cit. 2013-04-29]. Dostupné z: http://napoveda.seznam.cz/cz/zbozi/napoveda-pro-inzertni-servery/vseobecne-obchodni-podminky

[21] ZAITSEV, Petr a Vadim TKACHENKO. High Performance Full Text Search for Database Content. *MySQL Performance Optimization* [online]. 2013 [cit. 2013-04-29]. Dostupné z: http://www.mysqlperformanceblog.com/files/presentations/EuroOSCON2006-High-Performance-FullText-Search.pdf

[22] ZÁPOTOČNÝ, Ondřej, *Softwarová komponenta pro komerční portál onlinezbozi.cz*. Jihlava, 2010. Bakalářská práce. Vysoká škola polytechnická Jihlava.

SEZNAM PŘÍLOH

Příloha A – příprava linuxového serveru

1) Instalace virtuálního počítače

   Uživatelské jméno: ondrazap, Heslo: server2003

2) Aktualizace systému:

   sudo apt-get update; sudo apt-get upgrade

3) Instalace doplňků pro virtuální počítač
4) Instalace lamp

   sudo apt-get install lamp-server^ - heslo pro uživatele root bylo zvoleno root

5) Instalace phpmyadmin

   sudo apt-get install phpmyadmin

6) Test přihlášení k databázi http://localhost/phpmyadmin/ a vytvoření databáze dp
7) Úprava nastavení PHP, byla provedena v souboru /etc/php5/apache2/php.ini

   upload_max_filesize = 100 MB

   max_execution_time = 800 ms

   post_max_size = 100 MB

   memory_limit= 512 MB

8) Instalace ftp serveru

   Sudo apt-get install proftpd

   Gedit /etc/proftpd/proftpd.conf a změníme parametr DefaultRoot na /var/www/

9) Instalace vývojového prostřední NetBeans z http://www.netbeans.org, pro toto prostředí je nezbytné nainstalovat prvně balíček openjdk-7-jre.

Příloha B – výpočet indexu konzistence v SW Matlab

```
>> matice =  [1      3      1/3    1/5    1/7
              1/3    1      1/5    1/7    1/9
              3      5      1      1/3    1/5
              5      7      3      1      1/3
              7      9      5      3      1
              ]
matice =
   1.0000   3.0000   0.3333   0.2000   0.1429
   0.3333   1.0000   0.2000   0.1429   0.1111
   3.0000   5.0000   1.0000   0.3333   0.2000
   5.0000   7.0000   3.0000   1.0000   0.3333
   7.0000   9.0000   5.0000   3.0000   1.0000

>> [V,D] = eig(matice)
V =
  0.1067          0.0909 + 0.0371i   0.0909 - 0.0371i   0.0800 + 0.1312i   0.0800 - 0.1312i
  0.0561         -0.0073 + 0.0610i  -0.0073 - 0.0610i  -0.0539 - 0.0353i  -0.0539 + 0.0353i
  0.2170          0.1454 - 0.1379i   0.1454 + 0.1379i   0.0384 - 0.2695i   0.0384 + 0.2695i
  0.4401         -0.1434 - 0.3661i  -0.1434 + 0.3661i  -0.3976 + 0.2926i  -0.3976 - 0.2926i
  0.8630         -0.8898            -0.8898            0.8089             0.8089
D =
  5.2375             0                  0                  0                  0
       0        0.0258 + 1.1004i       0                  0                  0
       0             0             0.0258 - 1.1004i       0                  0
       0             0                  0             -0.1446 + 0.1623i       0
       0             0                  0                  0             -0.1446 - 0.1623i

>> maxlambda=D(1,1)
maxlambda =
   5.2375

>> CI=(maxlambda -5)/(5-1)
CI =
   0.0594

>> CR=CI/1.12
CR =
   0.0530
```

Příloha C – XML schéma pro Zboží.cz

Ukázkový soubor zbozi.xml se zbožím pro import. Data byla převzata z nápovědy služby Zboží.cz.

```
<?xml version="1.0" encoding="utf-8"?>

<SHOP xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="zbozi.xsd">

<SHOPITEM>
   <PRODUCT>Podložka pod myš</PRODUCT>
      <DESCRIPTION>Fosforeskující okraj, nevyžaduje baterie.</DESCRIPTION>
      <URL>http://obchod.cz/podlozky-pod-mys/fosfor</URL>
      <IMGURL>http://obchod.cz/obrazky/podlozky-pod-mys/fosfor.jpg</IMGURL>
      <PRICE_VAT>681</PRICE_VAT>
      <DELIVERY_DATE>0</DELIVERY_DATE>
      <ITEM_TYPE>new</ITEM_TYPE>
      <EAN>1234567890123</EAN>
                  <VARIANT>
                  <PRODUCTNAMEEXT>Podložka pod myš velká</PRODUCTNAMEEXT>
                  <PRICE_VAT>856</PRICE_VAT>
                  <EAN>1234567890123</EAN>
                  <PRODUCTNO>123456789</PRODUCTNO>
                  </VARIANT>
</SHOPITEM>
<SHOPITEM>
      <PRODUCT>Světélkující podložka pod myš</PRODUCT>
      <PRODUCTNAMEEXT>EXT</PRODUCTNAMEEXT>
      <DESCRIPTION>Fosforeskující okraj, nevyžaduje baterie.</DESCRIPTION>
      <URL>http://obchod.cz/podlozky-pod-mys/fosfor</URL>
      <IMGURL>http://obchod.cz/obrazky/podlozky-pod-mys/fosfor.jpg</IMGURL>
      <PRICE>620</PRICE>
      <PRICE_VAT>756</PRICE_VAT>
      <DELIVERY_DATE>2011-11-11</DELIVERY_DATE>
      <ITEM_TYPE>new</ITEM_TYPE>
   </SHOPITEM>
</SHOP>
```

Soubor zbozi.xsd obsahující schéma XML dokumentu použitého pro import zboží

```
<?xml version="1.0" encoding="utf-8" ?>

<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="SHOP" type="SHOP_TYPE" />

 <xs:complexType name="SHOP_TYPE">
  <xs:sequence minOccurs="1" maxOccurs="unbounded">
   <xs:element name="SHOPITEM" type="SHOPITEM_TYPE" />
  </xs:sequence>
 </xs:complexType>
```

```
<xs:complexType name="SHOPITEM_TYPE">
 <xs:sequence>
  <xs:element name="PRODUCT"              type="xs:string" />
  <xs:element name="PRODUCTNAME"          type="xs:string"              minOccurs="0" />
  <xs:element name="PRODUCTNAMEEXT" type="xs:string"              minOccurs="0"/>
  <xs:element name="DESCRIPTION"          type="xs:string" />
  <xs:element name="URL"                  type="xs:anyURI" />
  <xs:element name="IMGURL"               type="xs:anyURI"              minOccurs="0"/>
  <xs:element name="PRICE"                type="CZK_PRICE"              minOccurs="0" />
  <xs:element name="VAT"                  type="VAT_TYPE"               minOccurs="0" />
  <xs:element name="PRICE_VAT"            type="CZK_PRICE" />
  <xs:element name="MAX_CPC"              type="CZK_PRICE"              minOccurs="0" />
  <xs:element name="DUES"                 type="CZK_PRICE"              minOccurs="0" />
  <xs:element name="DELIVERY_DATE"    type="DELIVERY__DATE" minOccurs="0" />
  <xs:element name="SHOP_DEPOTS"          type="xs:string"              minOccurs="0" />
  <xs:element name="ITEM_TYPE"            type="ITEM__TYPE"             minOccurs="0" />
  <xs:element name="UNFEATURED"           type="xs:boolean"             minOccurs="0" />
  <xs:element name="EXTRA_MESSAGE"    type="EXTRA__MESSAGE" minOccurs="0" />
  <xs:element name="FIRMY_CZ"             type="xs:boolean"             minOccurs="0" />
  <xs:element name="MANUFACTURER"     type="xs:string"              minOccurs="0" />
  <xs:element name="CATEGORY"             type="xs:string"              minOccurs="0"
    maxOccurs="unbounded" />
  <xs:element name="EAN"                  type="EAN_TYPE"               minOccurs="0" />
  <xs:element name="PRODUCTNO"            type="xs:string"              minOccurs="0" />
  <xs:element name="VARIANT"              minOccurs="0"                 maxOccurs="unbounded" >
   <xs:complexType>
    <xs:sequence>
    <xs:element name="PRODUCT"              type="xs:string"              minOccurs="0" />
    <xs:element name="PRODUCTNAME"      type="xs:string"              minOccurs="0" />
    <xs:element name="PRODUCTNAMEEXT" type="xs:string"              minOccurs="0"/>
    <xs:element name="DESCRIPTION"          type="xs:string"              minOccurs="0" />
    <xs:element name="URL"                  type="xs:anyURI"              minOccurs="0" />
    <xs:element name="IMGURL"               type="xs:anyURI"              minOccurs="0"/>
    <xs:element name="PRICE"                type="CZK_PRICE"              minOccurs="0" />
    <xs:element name="VAT"                  type="VAT_TYPE"               minOccurs="0" />
    <xs:element name="PRICE_VAT"             type="CZK_PRICE"             minOccurs="0" />
    <xs:element name="MAX_CPC"              type="CZK_PRICE"              minOccurs="0" />
    <xs:element name="DUES"                 type="CZK_PRICE"              minOccurs="0" />
    <xs:element name="DELIVERY_DATE"  type=" DELIVERY__DATE" minOccurs="0" />
    <xs:element name="SHOP_DEPOTS"          type="xs:string"              minOccurs="0" />
    <xs:element name="ITEM_TYPE"            type="ITEM__TYPE"             minOccurs="0" />
    <xs:element name="UNFEATURED"           type="xs:boolean"             minOccurs="0" />
    <xs:element name="EXTRA_MESSAGE" type="EXTRA__MESSAGE" minOccurs="0" />
    <xs:element name="FIRMY_CZ"             type="xs:boolean"             minOccurs="0" />
    <xs:element name="MANUFACTURER"  type="xs:string"              minOccurs="0" />
    <xs:element name="CATEGORY"             type="xs:string"              minOccurs="0"
      maxOccurs="unbounded" />
    <xs:element name="EAN"                  type="EAN_TYPE" minOccurs="0" />
    <xs:element name="PRODUCTNO"            type="xs:string" minOccurs="0" />
    </xs:sequence>
   </xs:complexType>
  </xs:element>
 </xs:sequence>
</xs:complexType>
```

```
 <xs:simpleType name="CZK_PRICE">
  <xs:restriction base="xs:decimal">
   <xs:minExclusive value="0" />
  </xs:restriction>
 </xs:simpleType>
 <xs:simpleType name="VAT_TYPE">
  <xs:restriction base="xs:string">
   <xs:enumeration value="0,21" />
   <xs:enumeration value="21" />
   <xs:enumeration value="0,15" />
   <xs:enumeration value="15" />
  </xs:restriction>
 </xs:simpleType>
<xs:simpleType name=" DELIVERY__DATE">
  <xs:union>
   <xs:simpleType>
    <xs:restriction base="xs:date"/>
   </xs:simpleType>
   <xs:simpleType>
    <xs:restriction base="xs:int">
     <xs:minInclusive value="-1"/>
    </xs:restriction>
   </xs:simpleType>
   <xs:simpleType>
    <xs:restriction base="xs:string">
     <xs:enumeration value="ihned"/>
    </xs:restriction>
   </xs:simpleType>
  </xs:union>
 </xs:simpleType>
<xs:simpleType name="ITEM__TYPE">
  <xs:restriction base="xs:string">
   <xs:enumeration value="new" />
   <xs:enumeration value="bazar" />
  </xs:restriction>
 </xs:simpleType>
 <xs:simpleType name="EXTRA__MESSAGE">
  <xs:restriction base="xs:string">
   <xs:enumeration value="extended_warranty" />
   <xs:enumeration value="free_accessories" />
   <xs:enumeration value="free_case" />
   <xs:enumeration value="free_delivery" />
   <xs:enumeration value="free_gift" />
   <xs:enumeration value="free_installation" />
   <xs:enumeration value="free_store_pickup" />
  </xs:restriction>
 </xs:simpleType>
 <xs:simpleType name="EAN_TYPE">
  <xs:restriction base="xs:integer">
   <xs:pattern value="[0-9]{13}" />
  </xs:restriction>
 </xs:simpleType>

</xs:schema>
```

Příloha D – XML schéma pro Heureka!

Ukázkový soubor heuraka.xml se zbožím pro import. Data byla převzata z nápovědy služby Heureka!.

```
<?xml version="1.0" encoding="utf-8"?>

<SHOP xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="heureka.xsd">

 <SHOPITEM>
        <ITEM_ID>AB123</ITEM_ID>
        <PRODUCTNAME>Nokia 5800 XpressMusic</PRODUCTNAME>
        <PRODUCT>Nokia 5800 XpressMusic + pouzdro zdarma</PRODUCT>
        <DESCRIPTION>Klasický s plným dotykovým uživatelským rozhraním</DESCRIPTION>
        <URL>http://obchod.cz/mobily/nokia-5800-xpressmusic</URL>
        <IMGURL>http://obchod.cz/mobily/nokia-5800-xpressmusic/obrazek.jpg</IMGURL>
        <IMGURL_ALTERNATIVE>http://obchod.cz/mobily/nokia-5800.jpg</IMGURL_ALTERNATIVE>
        <VIDEO_URL>http://www.youtube.com/watch?v=KjR759oWF7w</VIDEO_URL>
        <PRICE_VAT>6000</PRICE_VAT>
        <HEUREKA_CPC>5.8</HEUREKA_CPC>
        <MANUFACTURER>NOKIA</MANUFACTURER>
        <CATEGORYTEXT>Elektronika | Mobilní telefony</CATEGORYTEXT>
        <EAN>6417182041488</EAN>
        <PRODUCTNO>RM-559394</PRODUCTNO>
        <PARAM>
                <PARAM_NAME>určení</PARAM_NAME>
                <VAL>dámské</VAL>
        </PARAM>
        <PARAM>
                <PARAM_NAME>objem</PARAM_NAME>
                <VAL>100ml</VAL>
        </PARAM>
        <PARAM>
                <PARAM_NAME>velikost</PARAM_NAME>
                <VAL>S</VAL>
        </PARAM>
        <PARAM>
                <PARAM_NAME>barva</PARAM_NAME>
                <VAL>zelená</VAL>
        </PARAM>
        <DELIVERY_DATE>2</DELIVERY_DATE>
        <DELIVERY>
                <DELIVERY_ID>CESKA_POSTA</DELIVERY_ID>
                <DELIVERY_PRICE>120</DELIVERY_PRICE>
        </DELIVERY>
        <ITEMGROUP_ID>EF789</ITEMGROUP_ID>
        <ACCESSORY>CD456</ACCESSORY>
</SHOPITEM>

</SHOP>
```

Soubor heureka.xsd obsahující schéma XML dokumentu použitého pro import zboží.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="SHOP" type="SHOP_TYPE"/>

  <xs:complexType name="SHOP_TYPE">
    <xs:sequence minOccurs="1"  maxOccurs="unbounded">
      <xs:element name="SHOPITEM" type="SHOPITEM_TYPE"></xs:element>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="SHOPITEM_TYPE">
    <xs:sequence>
      <xs:element name="ITEM_ID"                type="ITEM_ID_TYPE"/>
      <xs:element name="PRODUCTNAME"            type="xs:string" />
      <xs:element name="PRODUCT"                type="xs:string"          minOccurs="0"/>
      <xs:element name="DESCRIPTION"            type="xs:string"/>
      <xs:element name="URL"                    type="xs:anyURI" />
      <xs:element name="IMGURL"                 type="xs:anyURI"          minOccurs="0"/>
      <xs:element name="IMGURL_ALTERNATIVE"     type="xs:anyURI"          minOccurs="0"
       maxOccurs="unbounded"/>
      <xs:element name="VIDEO_URL"              type="xs:anyURI"          minOccurs="0"
       maxOccurs="unbounded" />
      <xs:element name="PRICE_VAT"              type="CZK_PRICE"/>
      <xs:element name="HEUREKA_CPC"            type="CZK_PRICE" />
      <xs:element name="ITEM_TYPE"              type="ITEM__TYPE"         minOccurs="0"/>
      <xs:element name="MANUFACTURER"           type="xs:string"          minOccurs="0"/>
      <xs:element name="CATEGORYTEXT"           type="xs:string"          minOccurs="0"
       maxOccurs="unbounded"/>
      <xs:element name="EAN"                    type="EAN_TYPE"           minOccurs="0"/>
      <xs:element name="PRODUCTNO"              type="xs:string"          minOccurs="0"/>
      <xs:element name="PARAM"                  type="PARAM_TYPE"         minOccurs="0"
       maxOccurs="unbounded" />
      <xs:element name="DELIVERY_DATE"          type="DATE_TYPE"          minOccurs="0"/>
      <xs:element name="DELIVERY"               type="DELIVERY_TYPE"      minOccurs="0"
       maxOccurs="unbounded"/>
      <xs:element name="ITEMGROUP_ID"           type="ITEM_ID_TYPE"       minOccurs="0"/>
      <xs:element name="ACCESSORY"              type="ITEM_ID_TYPE"       minOccurs="0"
       maxOccurs="unbounded" />
      </xs:sequence>
  </xs:complexType>

  <xs:simpleType name="ITEM_ID_TYPE">
    <xs:restriction base="xs:string">
      <xs:pattern value="[0-9a-zA-Z_\-]{1,36}" />
    </xs:restriction>
  </xs:simpleType>

 <xs:simpleType name="CZK_PRICE">
  <xs:restriction base="xs:decimal">
    <xs:minExclusive value="0" />
  </xs:restriction>
 </xs:simpleType>
```

```
 <xs:simpleType name="ITEM__TYPE">
  <xs:restriction base="xs:string">
   <xs:enumeration value="bazar" />
  </xs:restriction>
 </xs:simpleType>
 <xs:complexType name="PARAM_TYPE">
  <xs:sequence>
   <xs:element name="PARAM_NAME"          type="xs:string"/>
   <xs:element name="VAL"                 type="xs:string" />
  </xs:sequence>
 </xs:complexType>

 <xs:simpleType name=" DATE_TYPE">
  <xs:union>
   <xs:simpleType>
    <xs:restriction base="xs:date"/>
   </xs:simpleType>
   <xs:simpleType>
    <xs:restriction base="xs:int">
     <xs:minInclusive value="-1"/>
    </xs:restriction>
   </xs:simpleType>
   <xs:simpleType>
    <xs:restriction base="xs:string">
     <xs:enumeration value="ihned"/>
    </xs:restriction>
   </xs:simpleType>
  </xs:union>
 </xs:simpleType>

 <xs:complexType name="DELIVERY_TYPE">
  <xs:sequence>
   <xs:element name="DELIVERY_ID"          type="xs:string" />
   <xs:element name="DELIVERY_PRICE"       type="xs:decimal" />
   <xs:element name="DELIVERY_PRICE_COD" type="xs:decimal" minOccurs="0" />
  </xs:sequence>
 </xs:complexType>

 <xs:simpleType name="EAN_TYPE">
  <xs:restriction base="xs:integer">
   <xs:pattern value="[0-9]{13}" />
  </xs:restriction>
 </xs:simpleType>

</xs:schema>
```

Příloha E – PHP kód vyhledávání

```
$search = $this->getState('filter.search');
$match  = $this->getState('filter.match');
$sort   = $this->getState('filter.sort');

if($this->sendform==1){
   $this->limitstart=0;
}
//get instance sphinx
$sp = new SphinxClient();
$sp->SetServer('localhost', 9312);

do {
    $end    = false;
   switch ($match) {
       case 0:
          $sp->SetMatchMode(SPH_MATCH_EXTENDED2);
          $sp->SetRankingMode(SPH_RANK_PROXIMITY_BM25);
        break;
      case 1:
          $sp->SetMatchMode(SPH_MATCH_ANY);
          $sp->SetIndexWeights(array('product_1' => 10, 'product_2' => 5));
          $sp->SetFilterRange('@weight', 10, 999999, false);
          break;
      case 2:
          $sp->SetMatchMode(SPH_MATCH_ALL);
          $sp->SetIndexWeights(array('product_1' => 10, 'product_2' => 5));
          $sp->SetFilterRange('@weight', 10, 999999, false);
          break;
   }
   switch ($sort) {
      case 0:
          $sp->SetSortMode(SPH_SORT_RELEVANCE);
          break;
      case 1:
          $sp->SetSortMode(SPH_SORT_EXTENDED,"price_vat DESC");
          break;
      case 2:
          $sp->SetSortMode(SPH_SORT_EXTENDED,"price_vat ASC");
          break;
   }
   $sp->SetArrayResult(true);
   $sp->SetLimits($this->limitstart,$this->limitstart+$this->limit);
   $data = $sp->Query($search, 'product_1 product_2');

   if (($data["total"]==0) && (($match==0)||($match==2))){
      $match=1;
      $this->setState('filter.match', $match);
      $end=true;
   }
 }while($end);
$this->total = $data["total"];
```