Opinnäytetyö (AMK)

Tietojenkäsittelyn koulutusohjelma

Sähköisen liiketoiminnan järjestelmät

2015

Aleksi Lehtisalo

VERKKOKAUPAN TUOTEHAUN TOTEUTTAMINEN APACHE SOLRILLA

- CASE VERKKOKAUPPA X

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietojenkäsittely | Sähköisen liiketoiminnan järjestelmät

Maaliskuu 2015 | 35

Päivi Killström

Aleksi Lehtisalo

VERKKOKAUPAN TUOTEHAUN TOTEUTTAMINEN APACHE SOLRILLA – CASE VERKKOKAUPPA X

Asiakkaille on tärkeää, että tuotteiden löytäminen verkkokaupasta on mahdollisimman helppoa ja nopeaa. Verkkokauppojen tuotehaku on ominaisuus, jolla pyritään helpottamaan tuotteiden löytämistä verkkokaupan laajoista valikoimista. Tästä syystä verkkokaupan tuotehausta tehdään mahdollisimman tarkka ja nopea asiakkaille.

Opinnäytetyössä toteutetaan yrityksen X verkkokaupalle uusi tuotehaku käyttäen Apache Solria. Vanhassa tuotehaussa on käytetty tekstihakua, joka käy läpi koko tuotetietokannan etsien kaikki mahdolliset tulokset. Tästä syystä se on hidas ja kankea sekä kuormittaa tuotetietokantaa. Uuden tuotehaun on tarkoitus korvata vanhan tuotehaun puutteet sekä lisätä uusia ominaisuuksia, kuten ennakoivan haun.

Uusi tuotehaku toteutettiin seuraavien ohjelmien avulla: Apache Tomcat, Apache Solr, Notepad++ sekä verkkoselain. Työ esittelee etenemisvaiheet seuraavassa järjestyksessä: vanhan tuotehaun puutteiden selvitys, hakukonemoottorin valinta sekä hakukonemoottorin käyttöönotto että testaaminen.

Uusi tuotehaku saatiin onnistuneesti toteutettua. Onnistumiseen vaikutti vanhan tuotehaun puutteiden selvittäminen sekä uuden tuotehaun systemaattinen ja perusteellinen testaaminen.

ASIASANAT:

Apache Solr, hakukone, hakumoottori, tuotehaku, haku

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Business Information Technology | e-Business Systems

March 2015 | 35

Päivi Killström

Aleksi Lehtisalo

USING APACHE SOLR TO IMPLEMENT PRODUCT SEARCH TO AN E-COMMERCE SITE - CASE E-COMMERCE SITE X

It is important to the customers to find the products from e-commerce sites as fast and easily as possible. Product search is a feature which is implemented to an e-commerce site to ease the search of products. For this reason it is important for e-commerce sites to have fast and accurate product search.

The thesis implements a new product search feature for the company X's e-commerce site using Apache Solr. The old product search is a text search which goes through the entire product database seeking for all the possible results. For this reason the old product search is slow, stiff and loads the product database. The new product search is intended to replace the old product search's shortcomings and add new features such as predictive search.

The new product search was implemented by the following programs: Apache Tomcat, Apache Solr, Notepad++ and a web browser. This work shows the steps of how the shortcomings of the old product search were solved, how the new search engine was chosen, as well as the deployment and testing of the search engine.

The new product search was successfully implemented. Solving the shortcomings of the old product search as well as long-term testing of the new product search contributed to the success of the search.

KEYWORDS:

Apache Solr, search engine, product search, search

SISÄLTÖ

KUVAT

SANASTO

Java	Ohjelmointikieli, joka perustuu olio-ohjelmointiin.
DAEMON	Tulee sanoista Disk And Execution MONitor. On Unixissa tai muissa moniajo-käyttöjärjestelmissä ohjelma, joka suoritetaan taustalla.
Indeksointi	Asiasanoitus on dokumenttien sisällönkuvailutapa.
Tietokanta	Tietovarasto, joka on kokoelma tietoja, joilla on yhteys toisiinsa.
API	Application Programming Interface eli ohjelmointirajapinta, määrittää miten eri ohjelmistokomponentit keskustelevat keskenään.
REST	REpresentational State Transfer on ohjelmointirajapinta, joka mahdollistaa ohjelmistokomponenttien keskustelun HTTP-protokollan kautta.
JSON	JavaScript Object Notation on kevyt tiedonsiirtomuoto.
XML	Extensible Markup Language on rakenteellinen kuvauskieli, joka jäsentää ja selkeyttää suuria tietomassoja.
CSV	Comma-Separated Values on tiedostomuoto, joka tallentaa yksinkertaiseen taulukkomuotoon dataa, joka erotetaan toisistaan pilkuilla ja rivinvaihdoilla.
URL	Uniform Resource Locator on verkkosivujen osoite.
HTML	HyperText Markup Language on merkintäkieli, jolla nettisivut luodaan.

CSS	Cascading Style Sheet on kuvauskieli, jolla voi muokata merkintäkielen tyyliä ja muotoja.
JavaScript	Ohjelmointikieli, jota käytetään web-ympäristössä dynaamisten ominaisuuksien luomiseen.
JQuery	Avoimeen lähdekoodiin perustuva JavaScript-kirjasto.
AJAX	Asynchronous JavaScript And XML on web-kehitystekniikka, jolla voidaan muokata dynaamisesti verkkosivustoa.

1 JOHDANTO

Yrityksellä X on verkkokauppa, jossa päivittäin käy yli 25 000 uniikkia kävijää. Kävijöitä seurataan tyypillisillä verkkosivujen seurantamenetelmillä, kuten esimerkiksi Google Analytics -palvelun avulla. Koska kyseessä on aktiivinen verkkokauppa, on tärkeätä, että asiakas löytää tuotteensa sivustolta mahdollisimman helposti ja nopeasti. Yritys on itse huomannut sekä saanut palautetta siitä, että sivustolta on hidasta ja kankeaa hakea tuotteita sen omalla tuotehaulla. Tuotehaku verkkosivustolla on tekstihaku, jolloin se käy koko tuotetietokannan läpi etsien kaikki mahdolliset tulokset. Ruuhka-aikoina verkkosivuston tuotehaulla haetaan tuotteita keskimäärin joka toinen sekunti, joka kuormittaa huomattavasti verkkosivuston tuotetietokantaa.

Verkkokauppa X on uudistamassa kokonaan verkkosivustonsa. Uudesta verkkosivustosta tehdään sellainen, joista tuotteet olisi mahdollisimman helppo etsiä käyttäen yleistä tuotehakua. Tätä varten sivustolle tehdään kokonaan oma uusi hakukonemoottori. Tärkeätä on saada tuotetietokannan kuormitusta pienemmäksi ruuhka-aikoina sekä tehdä tuotehausta mahdollisimman tarkka ja nopea. Tämän lisäksi olisi myös käytössä ennakoiva tuotehaku, jolloin tuotehaku ehdottaa käyttäjälle valmiiksi mahdollisia tuotteita, joita hän on hakemassa verkkosivustolta.

Tässä opinnäytetyössä esitellään hakukonemoottorin valitsemisen taustaa sekä miten päädyttiin Apache Solriin (lausutaan solar). Opinnäytetyössä käydään läpi tarvittavat ohjelmat ja vaiheet Apache Solrin asentamiseen ja asetuksien muokkaamiseen. Tuotehakujen eli kyselyiden testaaminen suoritettiin Apache Solrin hallintapaneelin kautta sekä omalla testisivustolla, jossa testattiin ennakoivaa tuotehakua.

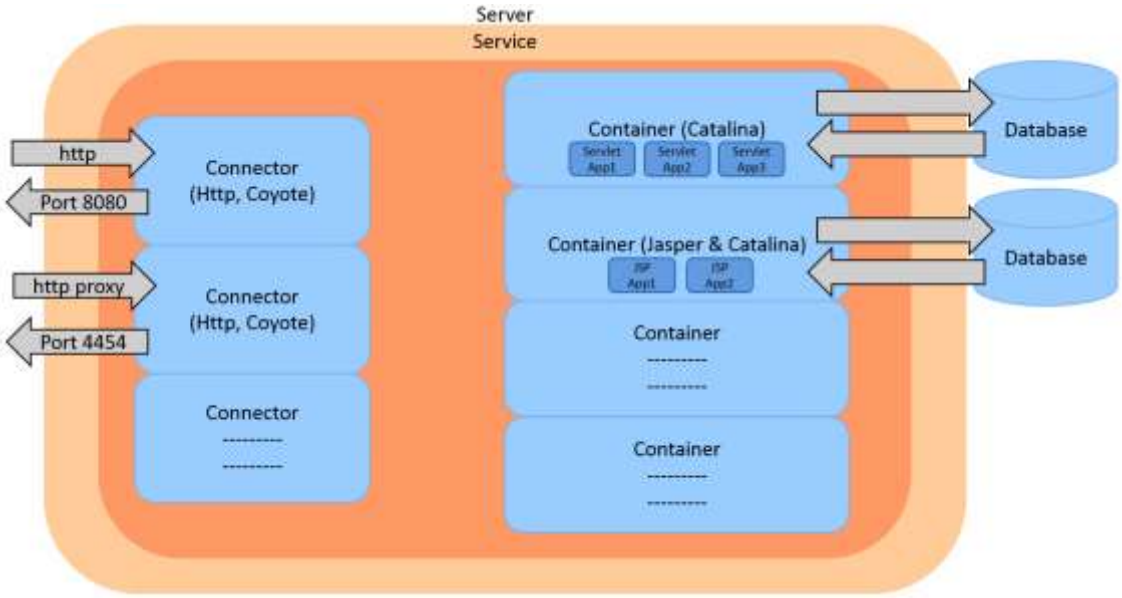
2 OHJELMAT

Lähes kaikki ohjelmat, joita käytän työssäni perustuvat avoimeen lähdekoodiin. Tämä tarkoittaa sitä, että ne ovat vapaasti käytettävissä ilman lisensointimaksuja. Apache Solrin pystyttämiseen ja muokkaamiseen tarvitaan kaksi ohjelmaa: Tomcat ja Notepad++. Tämän lisäksi Solrin hallintapanelin käyttöä varten tarvitaan verkkoselain, joka on vapaasti valittavissa.

2.1 Apache Tomcat

Apache Tomcat on avoimeen lähdekoodiin perustuva Java-pohjainen verkkopalvelin, jota kehittää Apache Software Foundation (Apache Tomcat 2014). Tomcat implementoi useita Java EE (Enterprise Edition) spesifikaatioita kuten Java Servletiä ja JavaServer Pages, sekä mahdollistaa verkkopalvelin ympäristön, jossa Java-ohjelmointikieli ajetaan (Chopra 2007, 3–4). Java Servlet on Java-ohjelmointikielenluokka, joka laajentaa palvelimien ominaisuuksia. Se isännöi applikaatioita, jotka perustuvat pyyntö-vastaus – ohjelmointimalliin. JavaServer Pages (JSP) on teknologia, jolla voi kontrolloida verkkosivujen sisältöä tai ulkomuotoa pienen ohjelman (servlet) avulla (The Java EE 5 Tutorial 2010).

Tomcat koostuu kolmesta pääkomponentista: Catalina, Coyote ja Jasper. Catalina-komponentti toimii Tomcatin servlet -ohjelmistojen ajoympäristönä eli kehyksenä (servlet container). Coyote-komponentti toimii web-liittimenä (web connector), joka toimii http-protokollalla ja mahdollistaa Catalinan toimimaan verkkopalvelimena. Jasper-komponentti mahdollistaa JSP:n ajamisen Tomcatissä (Ellis 2004, 347).



Kuva 1. Tomcat -ohjelman rakenne.

Kuva 1 selventää Tomcat-ohjelman rakennetta. Tomcat toimii Windows-palveluna (Windows Service) tai Linux- / Unix-daemonina. Palvelu (service) odottaa oletuksena yhteyksiä portista 8080, mutta sen voi myös itse määrittää. Yksi yksittäinen Tomcat -instanssi voi tarjota useita palveluita, mutta tämä on harvinaista. Jokaisella palvelulla on ainakin yksi tai useampi liitin (connector) sekä yksi tai useampi kehys (container) (Ellis 2004, 347).

2.2 Apache Solr

Apache Solr on avoimeen lähdekoodiin perustava Java-pohjainen hakukone-moottori, joka toimii itsenäisenä palvelinohjelmistona. Hakukone on tiedonhaku-järjestelmä, joka etsii tallennettua tietoa tietokonejärjestelmistä (Oxford Dictionaries 2015). Apache Solr käyttää enterprise-hakua (Enterprise Search), joka on sisällön eli dokumenttien, kuvien tai tekstien yksilöimistä sekä näyttämistä käyttäjille (AIIM 2013). Apache Solr indeksoi eli asiasanoittaa sisällön tietokannoista tai dokumenttienhallintajärjestelmistä omaan indeksiin, joista käyttäjillä on mahdollista hakea sisältöä sanoilla tai fraaseilla.

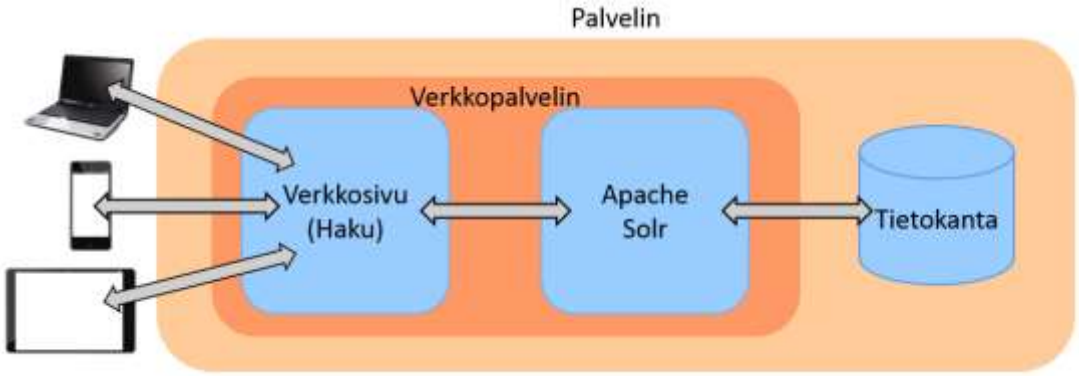
Apache Solrin ytimenä toimii Apache Lucene, joka on myös Apachen projekti. Apache Lucene on avoimeen lähdekoodiin perustuva korkean suorituskyvyn omaava hakukonemoottori ohjelmistokirjasto (Smiley 2009, 31). Se hyödyntää full-text search -tekniikkaa, joka tutkii kaikki sanat dokumentissa etsien hakusanoja vastaavia sanoja (Smiley 2009, 31).

Apache Solr muuntaa Apache Lucenen itsenäiseksi palvelimeksi (Smiley 2009, 32). Tämä mahdollistaa sen, että siihen voidaan ottaa yhteyttä http-protokollan GET- ja POST-pyyntöjen avulla. GET-pyyntö hakee määritellyt tiedot lähteestä ja POST-pyyntö antaa tiedot, jotka pitää tallentaa lähteeseen.

Yhteys Apache Solr palvelimeen otetaan REST (REpresentational State Transfer)-kaltaisen API:n (Application Programming Interface) avulla. REST on ohjelmistoarkkitehtuuri, joka antaa suuntaviivat skaalautuviin web-palveluihin ja API eli ohjelmointirajapinta määrittää, miten ohjelmistokomponentit keskustelevat keskenään. Sisältö indeksoidaan joko JSON-, XML-, CSV- tai binaarimuodossa http:n yli. Kyselyt tehdään HTTP GET-pyynnön avulla, josta tulokset tulevat JSON-, XML-, CSV- tai binaarimuodossa. (Apache Solr 2015)

Tärkeitä toimintoja, joita Apache Solr tuo Apache Luceneen, ovat:

- HTTP-protokollan yli toimivat indeksoinnit sekä kyselyt
- web-pohjainen hallintapaneeli ylläpitäjille
- skeeman eli haku sisällön ja palvelimen asetuksien määrittäminen XML-muodossa
- omien pyyntö käsittelijöiden (request handler) määrittäminen (Smiley 2009, 32)
- sanojen pilkkominen osiin (facets) (Apache Solr 2015)
- oikeinkirjoituksen tarkistaminen (spell checker) esimerkiksi automaattitäydennykseen (Apache Solr 2015).

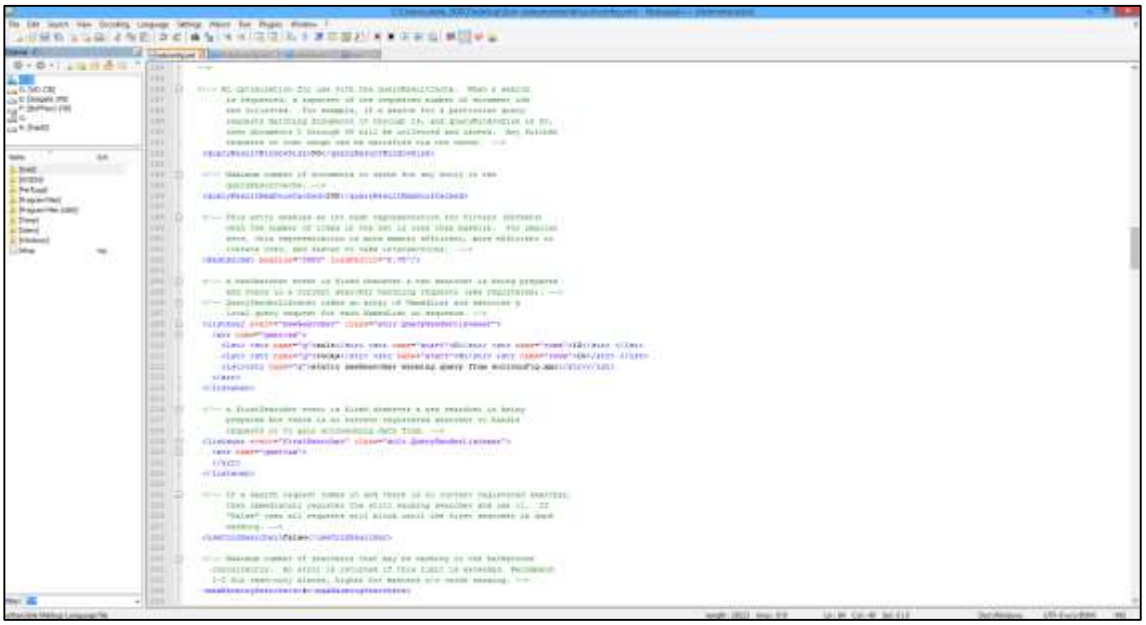


Kuva 2. Esimerkki Apache Solr-ohjelman toimintalogiikasta.

Kuvassa kaksi on esimerkkiä, miten Apache Solr voi toimia. Samalla palvelimella voi olla sekä verkkopalvelin, jossa verkkosivusto pyörii, sekä tietokanta mistä tiedot haetaan. Käyttäjä voi ottaa yhteyttä verkkosivuille tietokoneella, älypuhelimella tai tabletilla. Sivustolla on hakupalkki, josta voi etsiä esimerkiksi tuotteita. Käyttäjä syöttää hakusanan tai sanat hakupalkkiin, joita hän haluaa etsiä sivustolta. Hakupalkki lähettää pyynnön Apache Solrille hakusanoista, joita pitää hakea sen indeksistä. Apache Solr käy läpi indeksin ja etsii sopivimmat vaihtoehdot tuotteista, joihin sana tai sanat viittaavat. Apache Solr hakee tuotteet tietokannasta ja palauttaa tuotteet, jotka vastaavat hakusanoja, käyttäjälle.

2.3 Notepad++

Notepad++ on avoimeen lähdekoodiin perustuva tekstieditori, joka tukee yli 50 eri ohjelmointikieltä. Ominaisuuksiin kuuluu muun muassa sanojen automaattinen täydennys, syntaksin eli ohjelmointikielien kieliasun korostaminen sekä funktioiden lista. Lisäksi ohjelmaan voi ladata lisäosia, joilla pystyy lisäämään ominaisuuksia esimerkiksi automaattitäydennyksen funktioille (Notepad++ 2015). Kuvassa 3 on auki Notepad++ käyttöliittymä. Siinä on käytössä explorer-lisäosa, jolla on mahdollista avata tiedostot suoraan Notepad++ sisältä.



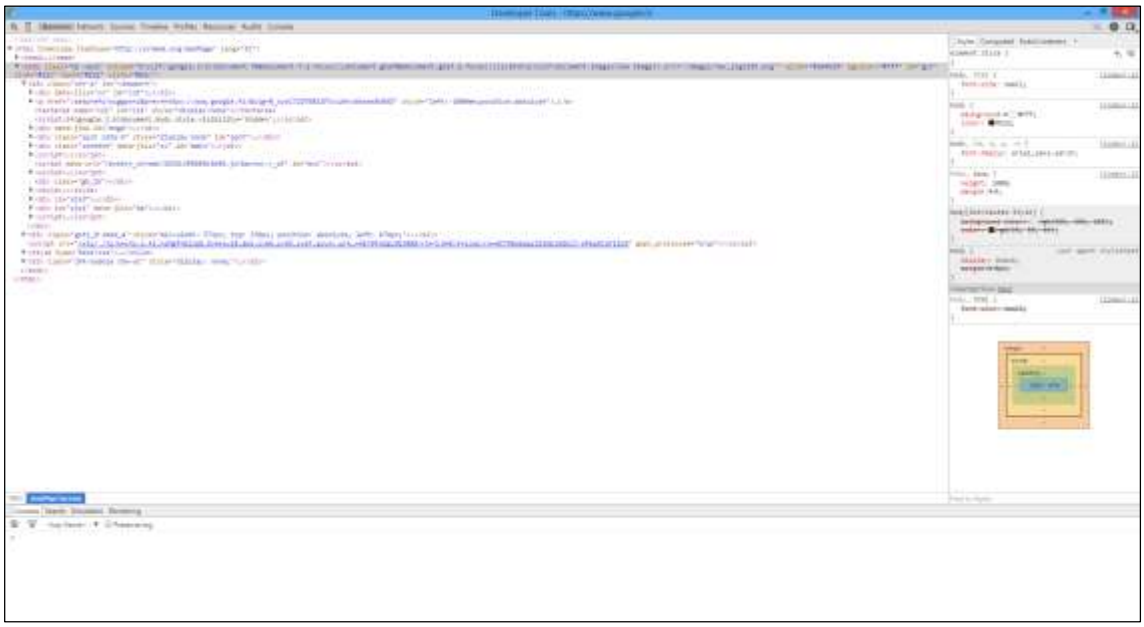
Kuva 3. Notepad++ -ohjelman käyttöliittymä.

2.4 Verkkoselain

Verkkoselain (lyhennetty selain) on tietokoneohjelma, jonka avulla voi selata www-sivuja. Selain yhdistää URL (Uniform Resource Locator) eli verkko-osoitteen avulla verkkosivuille, kuville, videoihin tai muuhun sisältöön. Tämän lisäksi verkkoselainta voidaan käyttää esimerkiksi yrityksen sisäverkossa ottamaan yhteyttä siinä oleviin palvelimiin (Webopedia 2015).

Suurimmat verkkoselaimet ovat Chrome, Firefox, Internet Explorer (IE), Safari ja Opera (W3Schools 2015). Vaikka selaimia on monia, löytyy jokaisesta standardituki verkkosivujen näyttämiseen. Tämän lisäksi jokaisessa on perusominaisuuksia, kuten kirjanmerkit, välilehdet sekä tuki lisäosiin.

Tässä opinnäytetyössä verkkoselaimena käytin Chromea. Vaikka muissa selaimissa on samat ominaisuudet, olin itse tottunut käyttämään Chromen Developer Tools-työkalua. Developer Tools -työkalun avulla voi muokata verkkosivujen ulkonäköä, tutkia verkkosivujen verkon käyttöä sekä testata JavaScript-koodia sivustoilla. Kuvassa 4 on Developer Tools käyttöliittymä avoinna, jossa voi muokata HTML-koodia.



Kuva 4. Developer Tools -työkalun käyttöliittymä.

3 CASE VERKKOKAUPPA X

Verkkokauppa X on uusimassa kokonaan verkkosivustonsa. Aikaisemmin käytössä olleen verkkosivun suurimpia ongelmia olivat sen tuotehaun hitaus ja kankeus sekä tuotetietokannan kuormitus. Pääsin vaikuttamaan uuden hakukonemoottorin valintaan sekä työstämään sen ominaisuuksia kuten nopeutta, tarkkuutta sekä ennakoivaa hakua.

3.1 Vanhan verkkokauppasivuston tuotehaku ja sen puutteet

Ennen kuin pääsin työstämään uutta ominaisuutta verkkokaupan uusille sivuille, oli tutustuttava vanhan järjestelmän hakukoneeseen sekä selvittää mitä ominaisuuksia halutaan uuteen hakukoneeseen. Näiden tietojen avulla voidaan tehdä suunnitelma, minkä avulla saadaan raamit uuden hakukoneen tekemiseen.

Ensimmäisenä haastattelin sivuston tekijää sekä yrityksen toimitusjohtajaa siitä, että mitä ominaisuuksia uuden hakukoneen pitää sisältää. Haastattelun tuloksena oli lista ominaisuuksista, jotka eivät tällä hetkellä ole käytössä nykyisillä sivuilla. Suurin vanhojen sivujen ongelma oli, että tuotehaku on epätarkka. Tuotehaku antoi myös tuloksia, jossa tuotteen ominaisuuksissa oli listattu kyseinen sana ja painotti sitä enemmän kuin esimerkiksi tuotteen nimikkeessä.

Edellisen haastattelun lisäksi selvitin sivuston ohjelmoitsijalta, miten nykyinen hakukone toimii sivustolla. Hän selvitti minulle sen, että kyseessä koko tuotetietokannan läpi menevä tekstihaku. Tämä tarkoittaa sitä, että kun asiakas käyttää sivuston hakukonetta tuotteen hakemiseen, haku kattaa koko tuotetietokannan. Tällöin se käy läpi jokaisen tuotteen tuotekoodin, nimikkeen sekä kuvauksen etsien kyseistä hakusanaa, jolla tuotetta on haettu. Näin ollen jokaisella haulla prosessi käydään uudestaan läpi. Tästä syystä tuotehaku on kankea ja hidas sekä kuormittaa tuotetietokantaa.

Haastattelujen jälkeen tein vielä itse konkreettisia testejä vanhan sivuston hakukoneella. Keskityin haastatteluissa ilmenneisiin ongelmiin, kuten tuotehaun

epätarkkuuteen ja tuotehaun hitauteen. Useiden testien jälkeen päädyin siihen tulokseen, että parhaiten vanhalla tuotehaulla löysi tuotteet tuotekoodilla, jolloin hakuaika oli erittäin lyhyt ja täsmällinen. Ensimmäiset ongelmat tulivat vastaan, kun hakuehtoina oli yleisiä sanoja. Näissä tapauksissa se yhdisti hakuehdot tuotteisiin, joiden nimikkeessä tai kuvauksissa mainitaan kyseinen hakuehto, listaten nämä hakutuloksiin. Tällöin haku oli hidas, koska se joutui etsimään haluamaani tuotetta muiden tuotteiden joukosta, jotka eivät välttämättä olleet yhteyksissä hakemaani tuotteeseen.

Omien testien, selvitystyön ja haastattelujen pohjalta oli hyvä lähteä tutkimaan, mitä erilaisia hakukoneita on mahdollista saada verkkosivuille, sekä mikä sopisi parhaiten kyseiselle yritykselle, jotta tuotehausta saadaan mahdollisimman tarkka ja nopea.

3.2 Uuden hakukoneen etsiminen

Toimitusjohtajan ja sivuston ohjelmoijan haastattelujen sekä omien testien tekeminen vanhan sivuston hakukoneella antoivat eväät uuden hakukoneen etsimiseen. Uuden hakukoneen kriteerit olivat selkeät: nopeus ja tarkkuus. Sivuston ohjelmoija oli selvitellyt varteen otettavia ohjelmia ennakkoon ja ehdotti muutamia minulle, joista voisin lähteä liikkeelle.

Kävin ensimmäisenä läpi niitä ohjelmia, mitä ohjelmoija oli itse aikaisemmin katsellut, ja jotka mahdollisesti sopivat uudelle sivustolle. Listalla oli kaksi ohjelmaa, joilla oli etsimäni ominaisuudet: painotus tarkkuuteen ja nopeuteen. Nämä varteen otettavat vaihtoehdot olivat Apache Solr ja ElasticSearch.

Tutkimukset aloitin Apache Solr-ohjelmasta, sillä sitä ohjelmoija ehdotti ensimmäisenä. Menin ohjelman kotisivuille ja tutkin, mikä kyseinen ohjelma on sekä mitä ominaisuuksia se sisältää. Sain selville, että Apache Solr on avoimeen lähdekoodiin perustuva itsenäinen palvelinohjelmisto, joka käyttää Apache Lucene-projektin hakukonemoottori ohjelmistokirjastoa. Ohjelma käyttää Apache-lisenssiä, joka sallii lähdekoodin avoimen että suljetun kehittämisen. Ominaisuuksien puolesta Apache Solr vaikutti erittäin hyvältä vaihtoehdolta, sillä siinä

on niitä ominaisuuksia, joita etsimme: hakuajan nopeus sekä tarkkuus. Nämä onnistuisivat datan monipuolisen käsittelemisen indeksoimisessa sekä laajojen ja joustavien kyselyjen (query) ehdoissa. Katsoin samalla myös mitkä sivustot käyttävät tällä hetkellä Apache Solr-ohjelmaa ja huomasin monien suurten yritysten käyttävän juuri sitä. Tästä ovat esimerkkinä videoiden ja tv-sarjojen suoratoistopalvelu Netflix, Facebookin omistama kuvienjakelupalvelu Instagram sekä yhdysvaltalainen internetissä toimiva huutokauppa eBay.

Seuraavaksi tutustuin ElasticSearch-nimiseen hakukoneeseen. Se on Apache Solr -ohjelman suurin kilpailija. Se myös perustuu avoimeen lähdekoodiin sekä Apachen Lucene -projektin hakukonekirjastoon. ElasticSearch -ohjelmasta löytyy myös samat ominaisuudet kuin Apache Solr -ohjelmasta eli datan monipuolisen käsittelemisen indeksointi sekä laajat kyselymahdollisuudet. Sivustoja, jotka käyttävät ElasticSearch -ohjelmaa, ovat muun muassa Deezer, joka on musiikin suoratoisto palvelu, sekä The Guardian, joka on brittiläinen sanomalehti.

Käytyäni läpi molemmat ohjelmat, menin keskustelemaan sivuston ohjelmoijan kanssa. Kävimme yhdessä läpi molempien ohjelmien ominaisuuksia ja tukea. Tulimme lopulta siihen lopputulokseen, että otamme käyttöön Apachen Solr -ohjelman. Valintaperusteina olivat ohjelman vahva yhteisötuki sekä ohjelmistopaketin helppo ja suoraviivainen käyttöönotto Windows-käyttöjärjestelmässä.

3.3 Apache Solrin käyttöönotto ja asetuksien määrittäminen

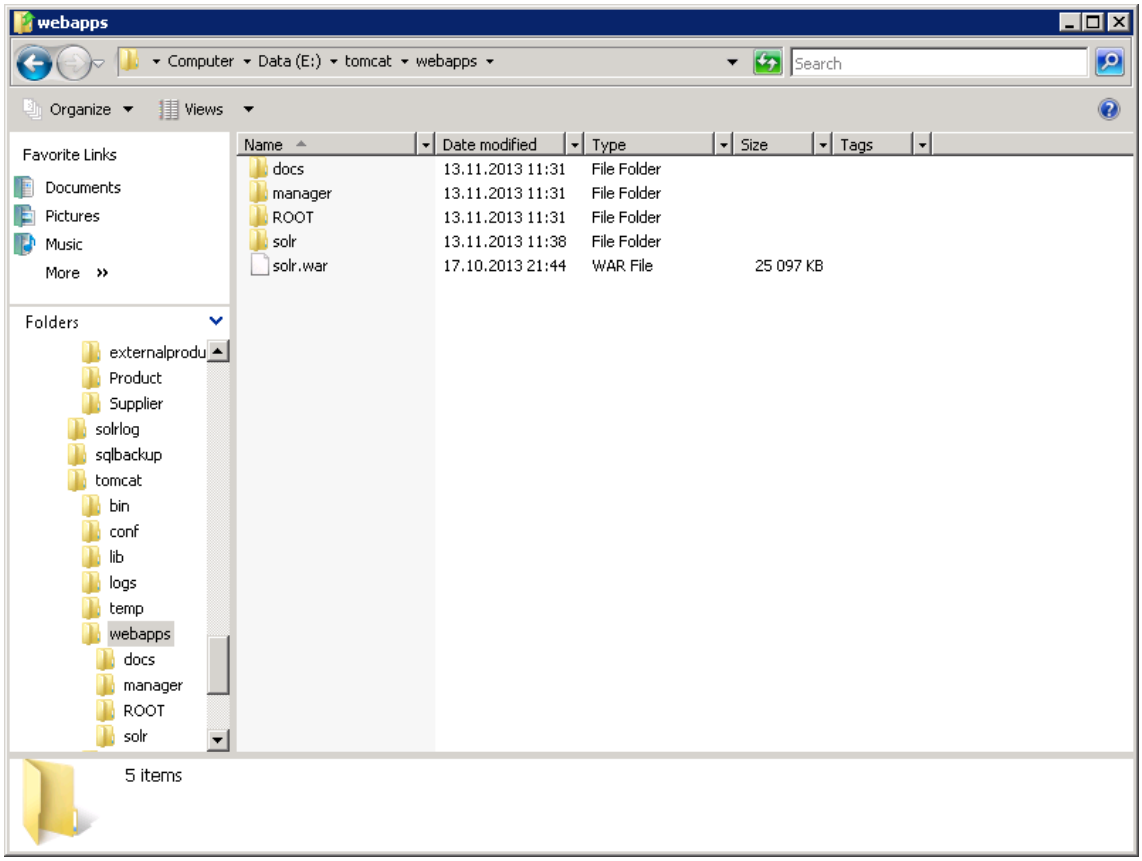
Apache Solr -ohjelman suurimpiin vahvuuksiin kuuluu sen laaja muokattavuus. Huonona puolena on se, että asetusten määrittäminen tapahtuu monessa eri tiedostossa, että Apache Solr saadaan toimimaan halutulla tavalla. Asetuksia joudutaan määrittämään kolmeen erilliseen xml-tiedostoon: db-data-config.xml, solrconfig.xml sekä schema.xml. Ennen kuin tiedostoja pääsee muokkaamaan, pitää Apache Solrin hallintapaneelin kautta luoda myös uusi ydin.

3.3.1 Apache Solrin käyttöönotto

Apache Solr-ohjelman käyttöönotto on tehty mahdollisimman yksinkertaiseksi. Ohjeet, jotka kattavat kaikki asennuksesta ominaisuuksiin, pystyy suoraan katsomaan Apache Solrin wiki-tietosivustosta tai lataamaan omalle koneelle pdf-muodossa. Asennuspaketti ladataan Apache Solr -ohjelman virallisilta sivuilta. Se sisältää ohjelman lisäksi esimerkki instanssin Apache Solr -ohjelmasta, jolla voi nopeasti kokeilla miten Apache Solr toimii.

Koska Solr on kirjoitettu Java-ohjelmointikielellä, se vaatii Java Servlet Container -kehyksen verkkopalvelimelle. Asennuspaketin mukana tulee kevyt Jetty-niminen Java servlet -palvelin. Emme kuitenkaan ottaneet kyseistä palvelinta käyttöön, sillä sivuston ohjelmoija halusi käyttää Apachen Tomcat-verkkopalvelinta. Perusteena oli se, että Tomcat on ollut kehittäjien ja tuotantojen käytössä vuosikausia, jolloin se on erittäin luotettava palvelinkäytössä.

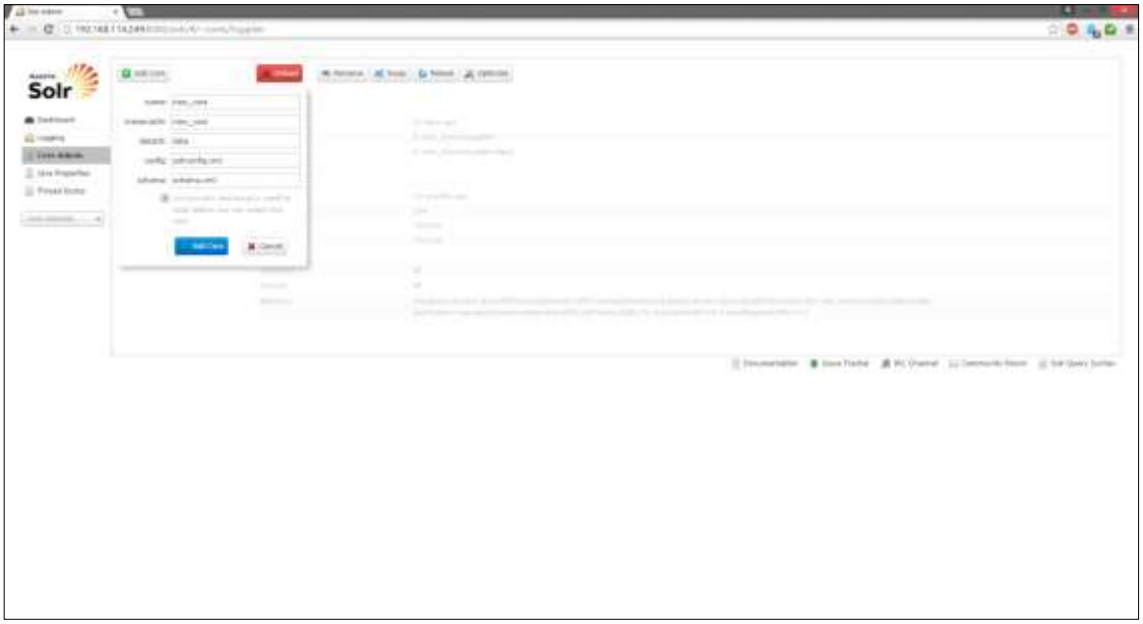
Apache Solr-asennus tapahtuu siten, että asennuspaketti puretaan haluttuun kansioon verkkopalvelimella, josta sitä myöhemmin kutsutaan. Apache Tomcatin webapps-kansioon lisätään tarvittavat Apache Solr -tiedostot, jonka jälkeen Apache Tomcat voidaan käynnistää omaksi palveluksi. Kuvassa 5 on kopioitu tarvittavat tiedostot Apache Tomcatin webapps-kansioon.



Kuva 5. Apache Solr -tiedostot lisätty Apache Tomcatin webapps-kansioon.

3.3.2 Ytimen luominen

Ennen tiedostojen muokkaamista, pitää luoda oma ydin (core) Apache Solriin. Ydin toimii Apache Solrin indeksinä. Uuden ytimen luominen tapahtuu hallintapaneelin kautta valitsemalla ytimien hallinnan (Core Admin). Sieltä voi lisätä uuden ytimen (Add core). Kuvassa 6 ollaan luomassa uutta ydintä hallintapaneelin kautta.



Kuva 6. Uuden ytimen luominen hallintapaneelin kautta.

3.3.3 Yhteyden ottaminen tuotetietokantaan

Hausta tulee nopea, kun tuotetietokanta indeksoidaan eli asiasanoitetaan omaan listaan. Yhteyden tuotetietokantaan muodostetaan määrittelemällä db-data-config.xml-tiedoston asetukset.

```
<?xml version="1.0" encoding="UTF-8" ?>
<dataConfig>
    <dataSource
        type="JdbcDataSource"
        driver="com.microsoft.sqlserver.jdbc.SQLServerDriver"
        url="jdbc:sqlserver://192.168.114.249;databaseName=test;selectMethod=cursor"
        user="test"
        password="test"
        responseBuffering="adaptive" />
    <document name="products" batchSize="500">
        <entity
            name="product"
            query="SELECT * FROM JW_JIBMS.dbo.JW_View_Product_Active"
            deltaQuery="SELECT * FROM JW_JIBMS.dbo.JW_View_Product_Active WHERE DateModified > '${dataimporter.last_index_time}'">
            <field column="VendorName" name="vendor_name"/>
            <field column="Code" name="code"/>
            <field column="Name" name="name"/>
            <field column="Description" name="description"/>
        </entity>
    </document>
</dataConfig>
```

Kuva 7. db-data-config.xml asetukset.

Kuvassa 7 on määritelty yhteys tuotetietokantaan. Datan lähteeseen (dataSource) määritämme oikeantyyppisen yhteyden. Koska käytössä on Microsoftin SQL Server tietokannan hallintajärjestelmä, pitää ajuriksi (driver) määritellä SQL

Server-ajurit. Myös URL eli tietokannan palvelimen osoite on määriteltävä, jotta yhteys saadaan tietokantaan. Samalla täytyy määritellä valmiiksi käyttäjä (user) ja salasana (password), joilla käyttöoikeus luodaan tietokantaan. Entity-elementti tekee SQL kyselyn (query) tuotetietokantaan ja hakee kaikki aktiiviset tuotteet. Tässä yhteydessä myös määritetään toinen kysely (deltaQuery). Se ajetaan silloin, kun indeksiin tehdään osittainen päivitys niihin tuotteisiin, joihin on tullut muutos. Myös tarvittavat kentät (field), jotka halutaan indeksoida, valitaan Entity-elementin sisällä. Tässä tapauksessa otamme käyttöön valmistajan (VendorName), tuotekoodin (Code), tuotteen nimike (Name) ja tuotteiden kuvaukset (description).

3.3.4 Skeeman asetukset

Skeemassa (schema) määritellään, miten Apache Solr tulee käsittelemään indeksoituvat tiedot. Jokaiselle tiedolle, jotka halutaan indeksoida, tehdään oma kenttä (field). Siinä määritellään kentän nimi (name), kentän tyyppi (type), kentän indeksointi (index) sekä kentän tallennus (stored). On myös mahdollista tehdä kopio valmiista kentästä, jolloin tehdään oma kopiokenttä (copyField). Siinä valitaan lähde (source), mistä kentästä tiedot kopioidaan ja kohde kenttä (dest), jonne tiedot kopioidaan.

Apache Solrissa voidaan tehdä kokonaan omat kenttätyypit (fieldType). Tämä on sen suurempia vahvuuksia, sillä siinä voidaan itse määritellä, miten tiedot indeksoidaan ja kysellään. Kenttätyyppi koostuu kahdesta analyzer-elementistä joiden tyypit ovat indeksi (index) ja kysely (query). Analyzer-elementti muodostuu yhdestä osittelijasta (tokenizer) ja yhdestä tai monesta suodattimesta (filter). Molemmilla on omat luokat (class) sekä omat lisäasetukset riippuen luokasta.

```
<!-- Valmistajan indeksointi ja query -->
<fieldType name="textVendor" class="solr.TextField" positionIncrementGap="100" >
  <analyzer type="index">
    <tokenizer class="solr.WhitespaceTokenizerFactory"/>
    <filter class="solr.LowerCaseFilterFactory"/>
  </analyzer>
    <analyzer type="query">
    <tokenizer class="solr.KeywordTokenizerFactory"/>
    <filter class="solr.LowerCaseFilterFactory"/>
  </analyzer>
</fieldType>
<!-- Valmistajan ENG -->
<!-- Pilkkoo sanoja pudottamalla perästä kirjaimen/numeron. Esim. asus => asus => asu, as, A jne. -->
<fieldType name="textVendor_eng" class="solr.TextField" positionIncrementGap="100" >
<analyzer type="index">
    <tokenizer class="solr.WhitespaceTokenizerFactory"/>
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.PatternReplaceFilterFactory" pattern="(.*)." replacement="$1" replace="first"/>
    <filter class="solr.EdgeNGramFilterFactory" maxGramSize="50" minGramSize="1"/>
</analyzer>
<analyzer type="query">
    <tokenizer class="solr.KeywordTokenizerFactory"/>
    <filter class="solr.LowerCaseFilterFactory"/>
  </analyzer>
</fieldType>
```

Kuva 8. Valmistajan nimen käyttämät kenttätyypit.

Kuva 8 on esimerkkinä miten valmistajan nimet indeksoidaan Apache Solriin. Käytössä on kaksi kenttätyyppiä textVendor sekä textVendor_eng. TextVendor-kenttätyyppi indeksoi ja kyselee koko valmistajan nimellä. Indeksoinnissa valmistaja ositetaan välilyöntien kohdalla, sillä käytössä on solr.WhitespaceTokenizerFactory-luokka. Tämän lisäksi valmistajan nimi muunnetaan pieniksi kirjaimiksi LowerCaseFilterFactory-suodattimen avulla.

Tämän lisäksi on luotu toinen kenttätyyppi (textVendor_eng), joka on ennakoivaa tuotehakua varten. Analyzer-elementit ovat molemmissa samat, mutta indeksointi vaiheessa lisätään kaksi suodatinta lisää. Ensimmäinen suodatin, joka käyttää PatternReplaceFilterFactory-luokkaa, tiputtaa valmistajien nimistä viimeisen kirjaimen pois. Seuraava suodatin, joka käyttää EdgeNGramFilerFactory-luokkaa, luo valmistajasta yhtä monta sanaa kuin siinä on kirjaimia. Esimerkiksi sanasta auto tulee a, au, aut ja auto. Tämän kenttätyypin avulla on mahdollista luoda tuki ennakoivaa tuotehakua varten.

```
<!-- VendorName -->
<field name="vendor_name" type="textVendor" indexed="true" stored="true" omitNorms="true"
termVectors="true" termPositions="true" termOffsets="true"/>
<!-- Valmistajan ENG tallennus -->
<field name="vendor_name_eng" type="textVendor_eng" indexed="true" omitNorms="true" stored="false"/>
<!-- Kopioi valmistajan ENG indeksin ja vie valmistajan ENG filterin läpi -->
<copyField source="vendor_name" dest="vendor_name_eng"/>
```

Kuva 9. Valmistajien kentät.

Kuvassa 9 on luotu valmistajille oma kenttä (vendor_name). Tämän lisäksi on luotu kenttä (vendor_name_eng) ennakoitua tuotehakua varten sekä kopiokenttä (copyField). Kopiokenttä kopioi indeksoidun vendor_name-kentän, jonka jälkeen se indeksoi valmistajat uudelleen vendor_name_eng-kenttätyypin mukaan.

3.3.5 Oman pyyntökäsittelijän luonti

Kun skeema on valmis, rakennetaan oma pyyntökäsittelijä (request handler). Tätä on mahdollista muokata solrconfig.xml-tiedostossa. Apache Solrissa on valmiina kolme kyselyjäsennintä (query parser): vakiokyselyjäsennin, DisMax sekä Extended DisMax (eDisMax). Vakiokyselyjäsennin on hyvin tarkka, mutta on herkkä syntaksi-virheille. DisMax käyttää yksinkertaistettua Apache Lucenen kyselyjäsennintä. eDisMax on paranneltu versio DisMaxista ja tukee kokonaan Apache Lucenen kyselyjäsennintä.

Otimme käyttöön eDisMax kyselyjäsentimen, sillä tarvitsimme sen ominaisuuksia ennakoivaa tuotehakua varten. Halusimme antaa omille kentille eri painotusarvot, sillä sitä kautta saimme hausta tarkemman. Voimme määritellä montako sanaa voi olla välissä hakusanoilla ja montako hakusanaa pitää tunnistaa, jolloin haku antaa relevantteja tuloksia.

```
<requestHandler name="/search" class="org.apache.solr.handler.component.SearchHandler">
  <lst name="defaults">
      <str name="df">name</str>
      <str name="defType">edismax</str>
      <str name="rows">10</str>
      <str name="fl">*,score</str>
      <str name="qf">
          code^15
          code_eng^4
          code_ng^2
          vendor_name^10
          vendor_name_eng^4
          name^4
          name_eng^2
          name_ng
          description_eng^0.1
      </str>
      <str name="pf">
          name^2
          name_eng^1
      </str>
      <str name="qs">5</str>
      <str name="mm">3</str>
  </lst>
```

Kuva 10. Pyyntökäsittelijän asetukset.

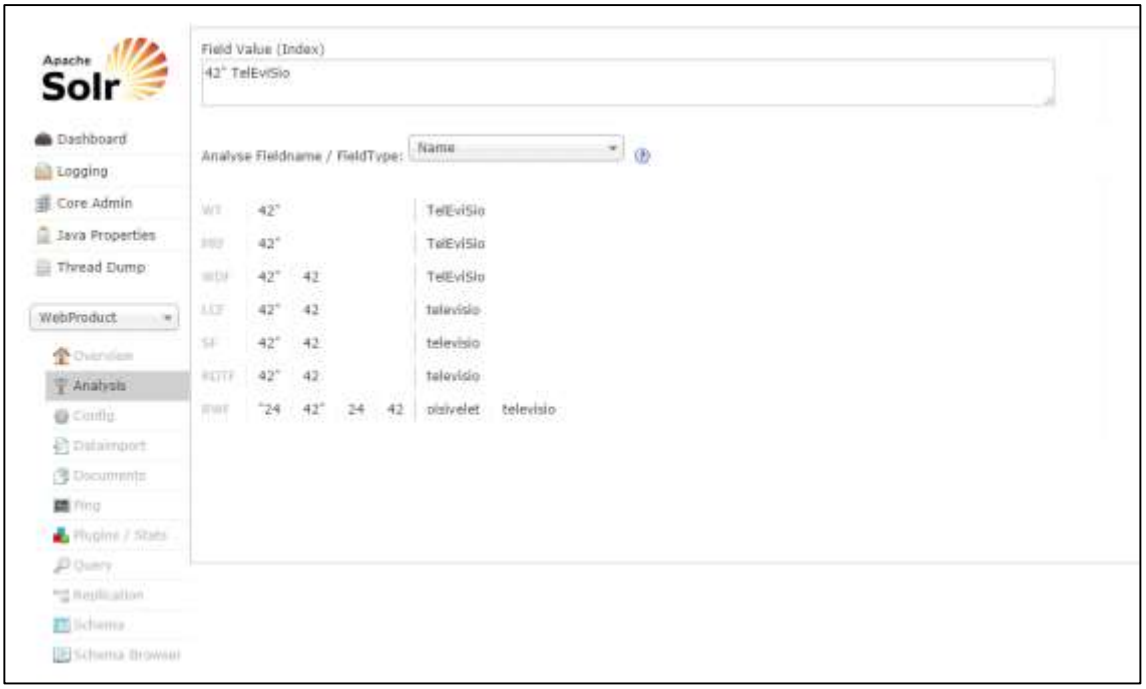
Kuvassa 10 on määritelty valmiiksi omat pyyntökäsittelijän asetukset. Jotta ennakoiva tuotehaku voisi toimia kunnolla, pitää kentille asettaa omat arvot. Pyyntökäsittelijälle annetaan oma nimi, jolla sitä kutsutaan. Tässä tapauksessa annettiin helppo englanninkielinen nimi /search. Oletusarvokenttänä (df) toimii tuotteen nimi, jos kyselykenttiä (qf) ei ole määritetty. Kyselykenttiin määritellään kaikki kentät, joista kysely haetaan. Käytössä on tuotekoodi, valmistajan nimi sekä tuotteen nimi ja niiden ositetut osat. Myös tuotekuvauksen ositetuista sanoista haetaan tuotetta. Jokaiselle kentälle on myös annettu omat arvot, joilla kenttää painotetaan. Kuten kuvasta näkyy, tuotekoodia painotetaan eniten, jonka jälkeen valmistajaa ja nimeä. Samalla määritetään lisäarvot (pf) nimi-kentille, jos hakusanat ovat sekä tuotekuvauksessa että nimissä. Hakusanat saavat olla vähintään 5 sanan päässä (qs) ja minimissään 3 hakusanaa (mm) pitää olla kyselyssä, jotta se antaa relevantin tuloksen.

3.4 Hakujen testaaminen

Kun asetukset olivat kunnossa, testattiin itse hakua. Hallintapaneelin kautta pystyy analysoimaan, miten omat kenttätyypit indeksoivat tiedot. Myös hallintapaneelin kautta voi kokeilla hakuja kysely-työkalun (Query) avulla. Tämän lisäksi tehtiin oma testisivusto, jossa testattiin ennakoivaa hakua JavaScript-ohjelmointikieltä hyödyntäen.

3.4.1 Analysointi-työkalun käyttö

Omien kenttätyyppien testaaminen on tehty helpoksi Apache Solrissa. Hallintapaneelista löytyy analysointi (analysis) -työkalu, joka näyttää miten ositin ja eri suodattimet käyttäytyvät. Työkalua käytettiin paljon, kun testattiin esimerkiksi tuotteen nimen pilkkomista osiksi tai miten eri suodattimia kannattaa käyttää.



Kuva 11. Analysointi-työkalu käytössä tuotteen nimi-kentässä.

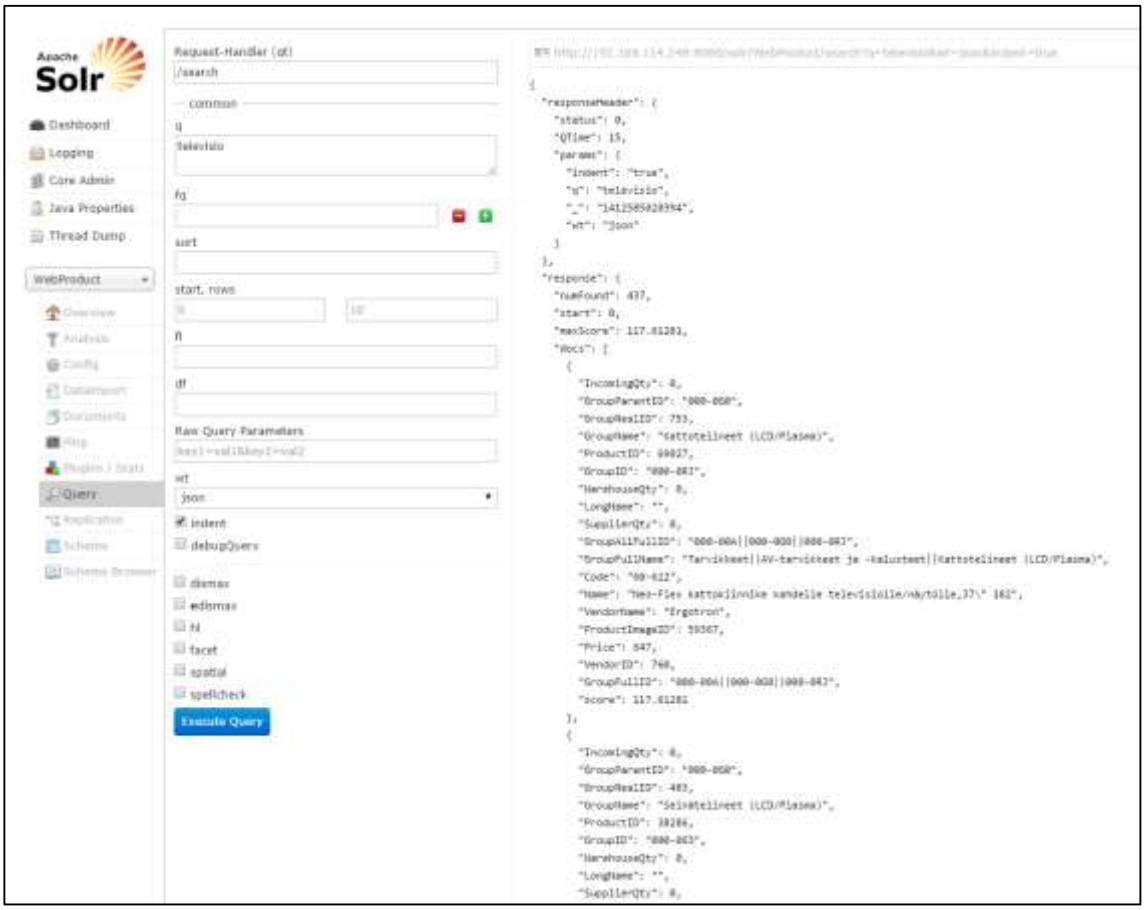
Kuvassa 11 kokeiltiin, miten tuotteen nimi-kenttä indeksoi tuotteen nimen. Esimerkkinä toimi 42” TelEviSio. Televisio-sana tarkoituksella kirjoitettiin isoilla se-

kä pienillä kirjaimilla, sillä haluttiin nähdä miten omat suodattimet toimivat. Kuvassa näkee miten Apache Solr käsittelee järjestyksessä analyzer-elementin osittajan ja suodattimet.

Nimi-kentässä on käytössä välilyönnin osittaja (WT), jolloin hakusanojen välilyönnin kohdalla Apache Solr tekee niistä omat osat. Osittajan jälkeen sanat siirtyvät suodattimiin. Ensimmäisenä suodattimena on merkkien korvaaja (PRF), joka poistaa hakusanoista ylimääräiset välimerkit. Seuraavan suodattimen (WDF) avulla pilkotaan osista numerot ja sanat erikseen, jos ne ovat yhdessä tai käytössä on välimerkkejä. Tässä tapauksessa 42”-osasta tulee kaksi erillistä osaa: 42” ja 42. Tämän jälkeen kaikki osat muunnetaan pieniksi kirjaimiksi (LCF), jolloin TelEviSio sanasta tulee televisio. Seuraava suodatin (SF) tarkistaa, onko osilla synonyymejä, jotka olemme määrittäneet omaan tekstitiedostoon. Toiseksi viimeinen suodatin (RDTF) tarkistaa, onko osista tullut kaksoiskappaleita ja tarvittaessa poistaa ne. Viimeinen suodatin (RWF) kääntää osat päinvastaiseen järjestykseen ja säilyttää alkuperäisen osat.

3.4.2 Kysely-työkalun käyttö

Kun indeksoituvat kentät saatiin toimimaan, päästiin kokeilemaan, miten kyselyt toimivat. Apache Solrissa on hallintapaneelissa oma työkalu tätä varten. Kysely (Query) -työkalun avulla voidaan kokeilla omien pyyntökäsittelijöiden toimivuutta. Oletuksena Apache Solr käyttää vakiokyselyjäsennintä, jota voidaan muuttaa. Työkalun avulla on helppo kokeilla, miten kenttien arvojen painotukset kannattaa tehdä ilman, että joutuu jatkuvasti muokkaamaan suoraan solrconfig.xml-tiedostoa.



Kuva 12. Oman pyyntökäsittelijän testaaminen.

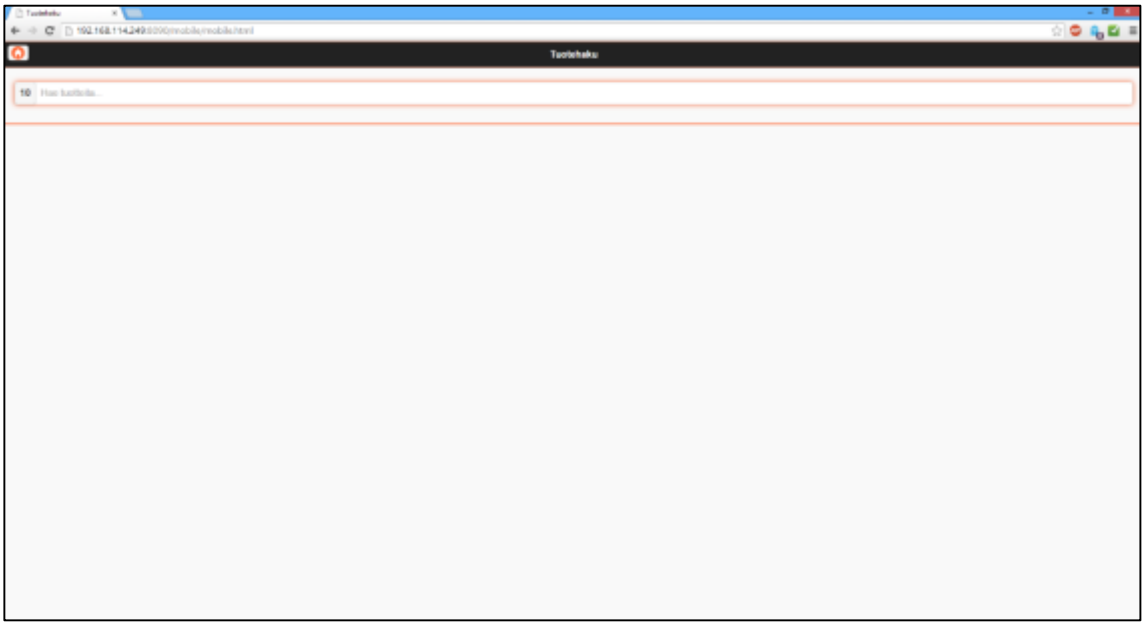
Kuvassa 12 testataan oman pyyntökäsittelijän toimintaa. Solrconfig.xml-tiedostossa on määritelty, mitä ja miten kyselyn tulee toimia. Tässä tapauksessa haettiin televisio-sanalla tuotteita tuotetietokannasta. Tulokset tulevat JSON eli JavaScript Object Notation muodossa. Tuloksista huomaa, että televisio-hakusanalla on löytynyt 437 mahdollista tuotetta hakusanalla (q) ja tuotteiden hakemiseen meni 15 millisekuntia (QTime). Ensimmäinen tuote, joka on saanut 117.61 pistettä (score), on kattokiinnike kahdelle televisiolle. Tuotteen nimestä löytyy televisio-sana, jolloin tuote saa hyvät pisteet.

3.4.3 Oman testisivuston tekeminen

Kun yksittäisten sanojen ja fraasien hakeminen saatiin valmiiksi, alkoi oman testisivuston luonti. Se on kirjoitettu HTML5-ohjelmointikielellä ja käyttää CSS3-

tyyliohjeita. Sivuston tarkoituksena on kokeilla, miten ennakoiva haku toimisi uusilla sivuilla. Tätä varten piti sivustolla käyttää JQuery Mobile-ohjelmistokirjastoa, joka laajentaa JavaScript-ohjelmointikielen ominaisuuksia.

Ennakoivaa hakua varten tarvittiin JQuery Mobile-ohjelmointikirjastoa. Tässä tapauksessa versio oli 1.4.1, jonka kirjastossa on tuki ennakoivaa syöttöä varten. Se myös sisältää valmiiksi oman käyttöliittymätyylin, jota käytettiin testisivustolla. Kuvassa 13 on esillä testisivuston käyttöliittymä.



Kuva 13. Testisivuston käyttöliittymä.

Kun sivuston pohja saatiin valmiiksi, kirjoitettiin tarvittava JavaScript-koodi, jolla ennakoiva haku onnistuisi. Samalla tehtiin myös oma JavaScript-tiedosto, johon tuleva koodi kirjoitettaisiin. Sivuston HTML-koodiin lisättiin linkki tiedostoon, jotta sivusto käyttäisi JavaScript-tiedostoa.

```
<!DOCTYPE html>
<html>
<head>
    <title>Tuotehaku</title>
    <meta charset="utf-8">
    <link rel="stylesheet" href="http://code.jquery.com/mobile/1.4.1/jquery.mobile-1.4.1.min.css" />
    <link rel="stylesheet" type="text/css" href="jquery.mobile.icons.min.css"/>
</head>
<body>

<div data-role="page">

    <div data-role="header">
        <a class="clearItems" data-role="button" data-icon="home"></a>
        <h1>Tuotehaku</h1>
    </div>

    <div data-role="content">
        <div data-role="listview" data-icon="info" id="autocomplete"
        data-inset="true" data-filter="true" data-filter-placeholder="Hae tuotteita...">
        </div>
    </div>

    <div data-role="footer">
    </div>

</div>
<script src="http://code.jquery.com/jquery-1.9.1.min.js"></script>
<script src="http://code.jquery.com/mobile/1.4.1/jquery.mobile-1.4.1.min.js"></script>
<script src="/js/jquery_mobile.js"></script>
</body>
</html>
```

Kuva 14. Testisivuston HTML-koodi.

Koska käytössä on JQuery Mobile, käyttö vaatii sen antamia attribuutteja. Niiden avulla saadaan sivustolla toimimaan sen luomat dynaamiset sivut. Esimerkkinä on kuva 14, jossa on testisivustoni HTML-koodi. JQuery Mobilessa pitää määritellä omat data roolit (data-role) kaikille HTML-merkeille. Ennakoivaa hakua varten on käytettävä data-roolina luettelonäkymää (listview). Tämän lisäksi siihen on annettu data-suodatin (data-filter) arvo todeksi, jolloin se suodattaa näkyvän tiedon pois, jos sitä ei ole listalla. Myös oletusarvo (data-filter-placeholder) voidaan antaa tekstikentälle. Listanäkymän tunnisteena (id) on autocomplete, jolla sitä tullaan kutsumaan JavaScript-koodissa.

```
$( "#autocomplete" ).on( "filterablebeforefilter", function ( e, data ) {
    var $ul = $( this ),
        $input = $( data.input ),
        value = $input.val(),
        html = "";
    if ( value && value.length > 2 ) {
        $.ajax({
            type:'GET',
            url:'/solr/WebProduct/search',
            dataType:'json',
            data: {q:value, wt:"json", indent:"true"},
        })
        .then( function ( json ) {
            for(var i = 0; i < json.response.docs.length; i++) {
                html += "<li><b>"+json.response.docs.vendor_name+"</b>" + " " + json.response.docs.name + "</li>"
            };
            $ul.html( html );
            $ul.listview( "refresh" );
            $ul.trigger( "updatelayout");
        });
    }
});
```

Kuva 15. JavaScript-koodi, joka toteuttaa ennakoivan haun.

Kuvassa 15 on JavaScript-koodi, jolla on toteutettu ennakoiva haku. JQuery Mobile hyödyntää luettelonäkymän data-roolia HTML-koodissa tehdessään ennakoivan haun. JavaScriptissä tehtiin on-funktio autocomplete-valitsimeen. Se käyttää toimintoa (event), joka tarkistaa onko mahdollista suodattaa lapsielementtejä, jotka tässä tapauksessa ovat hakutuloksia. Samalla luodaan myös tarvittavat muuttujat, joita tarvitaan koodia varten. $ul-muuttuja määrittää sillä hetkellä valitun valitsijan, joka tässä tapauksessa on autocomplete. Seuraava muuttuja $input ottaa talteen painetut näppäimet ja tallentaa ne value-muuttujaan. Viimeinen muuttuja html on aluksi tyhjä, koska myöhemmin sinne tallennetaan tulevan tuotteen tiedot.

Kun muuttujat on määritelty, tehdään ehtolause (if). Ehtolause tarkistaa sen, että value-muuttujassa on tallennettu tietoja ja se on vähintään kaksi merkkiä pitkä. Tämän jälkeen se tekee AJAX-kutsun Apache Solrin palveluun. Tyypiksi (type) määritetään hae (GET), jolla haetaan tiedot. Osoitteeksi (url) määritetään Apache Solr. Datatyypiksi (dataType) määritetään JSON ja data-kohtaan määritetään kysely (q) hakumuoto (wt), sekä sallitaanko hakeminen sisennyksistä (indent).

Ehdon toteutuessa (then) tehdään uusi funktio, jossa on oma muuttuja json. Funktion sisälle tehdään for-silmukka, joka alkaa nollasta ja jatkuu kunnes kymmenes tuote on listattu. Tämä johtuu siitä, että olemme määritelleet pyyntö-

käsittelijään maksimirivien määräksi kymmenen. For-silmukka täyttää html-muuttujan kymmenen kertaa eri tuotteilla. Tämä johtuu siitä, että muuttujana toimii i, joka määrittää minkä tuotteiden nimet ja valmistajat listataan.

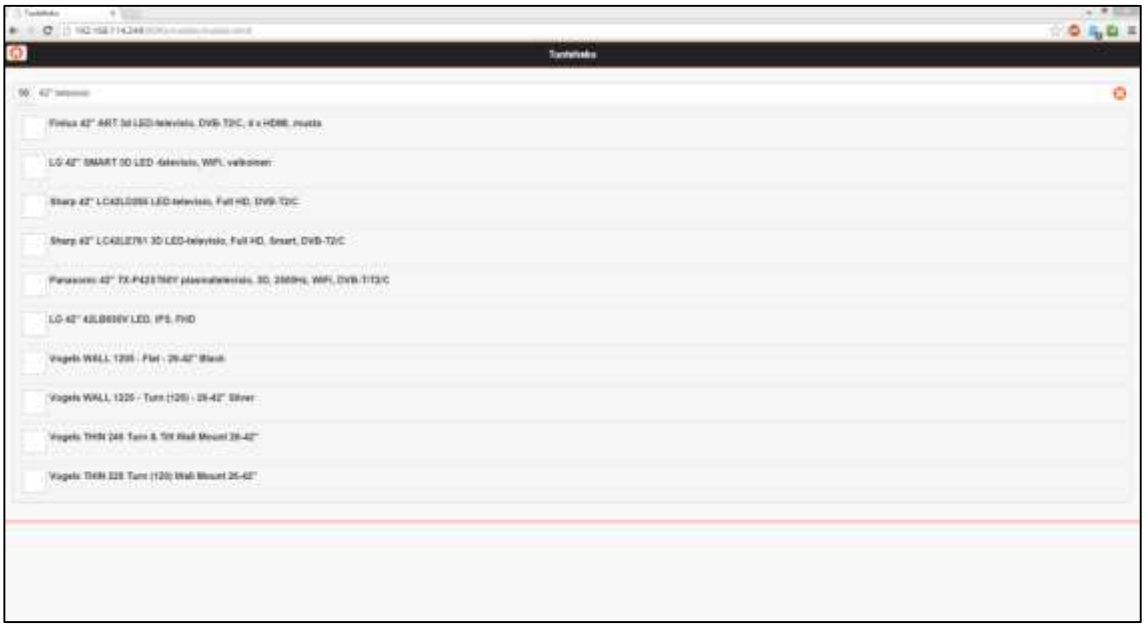
For-silmukan jälkeen täytetään luettelonäkymä saaduista tuotteista. Seuraavaksi sivuston näkymä päivitetään dynaamisesti. Tämän hoitaa luettelonäkymän listauksen päivittäminen ($ul.listview(”refresh”)), se myös laukaisee sivuston päivityksen ($ul.trigger(”updatelayout”)), joka piilottaa ei-relevantit tulokset listauksesta.

3.4.4 Ennakoivan haun testaaminen

Kun JavaScript-koodi saatiin toimimaan, aloitettiin ennakoivan haun testaaminen. Hakemisen testaamiseen käytettiin yleisiä sanoja, jotka olivat ongelmana vanhassa hakukoneessa. Samalla myös kokeiltiin miten tuotekoodeja sekä valmistajan ja tuotteen nimikkeen sekoituksia oli mahdollista löytää uudella hakukoneella.

Kentille, joista tiedot haetaan, on ensimmäisenä annettava omat painotukset. Tiedettiin, että tuotekoodi oli tärkein hakuehto, jolla voi hakea. Tästä syystä annettiin code-kentälle alustavasti arvo 100 ja sen osakentille code_eng 75 ja code_ng 50. Seuraavaksi tärkeintä oli löytää valmistajan nimellä tuotteet, joten vendor_name-kentän arvoksi asetettiin 100 ja vendor_eng-kentälle 70. Tuotteen nimikkeeseen annettiin pienempiä arvoja. Name-kentälle asetettiin arvoksi 60 ja sen osille name_eng 40 ja name_eg 30. Myös haluttiin, että tuotetekstistä olisi mahdollista hakea erikoisominaisuuksilla, joten description_eng-kentälle laitettiin arvoksi 20.

Eniten aikaa testaamisessa meni oikeiden painotusten löytämiseen. Ennakoiva haku tuotti hyvin paljon vaikeuksia, sillä oikeita arvoja oli hankala määrittää. Syynä tähän oli se, että sanojen osituskentille annettiin liian korkeat arvot, jolloin ne saivat suuremmat pisteet. Lopulta päädyttiin tekemään yksinkertainen taulukko, johon laskettiin mitkä voisivat olla sopivimmat arvot. Arvot voi katsoa kuvasta 10.



Kuva 16. Testisivuston haun testaaminen.

Kuvassa 16 on haettu 42” televisio-sanalla tuotteita ja näkyvillä on tulokset. Tuloksista voi huomata, että haku toimii hyvin, sillä ensimmäiset kuusi hakutulosta ovat 42-tuumaisia TV-vastaanottimia. Seuraavat neljä hakutulosta ovat tuotteita television seinäkiinnitykseen. Haku toimi oikein, sillä testihetkellä oli tuotetietokannassa kuusi 42”-tuumaista televisiota, ja tämän jälkeen haku etsi muut relevantit tuotteet kyseiselle hakusanalle.

Lopulta päädyttiin kuvassa 10 näkyviin arvoihin, koska ne antoivat parhaimmat tulokset. Kun testaukset oli saatu päätökseen, lisättiin uusi sivusto käyttämään Apache Solrin ajamaa uutta hakumoottoria.

4 YHTEENVETO

Käytämme joka päivä erilaisia hakukoneita etsien uutisia, kuvia tai videoita verkosta. Vaikka tässä opinnäytetyössä keskitytään tuotteiden hakemiseen, on hakukoneen perustoiminto sama kaikilla: tallennettujen tietojen etsimistä tietokonejärjestelmistä. Minulla ei ollut aikaisempaa kokemusta suuremmasta ohjelmointityöstä eikä varsinkaan hakukoneisiin liittyvästä ohjelmoinnista. Omat aiemmat kokemukseni perustuivat esimerkiksi videoiden etsimiseen YouTube-videopalvelusta.

Työn ensimmäinen ja helpoin osuus oli kerätä tiedot vanhan tuotehaun ongelmista toimitusjohtajan ja sivuston ohjelmoijan haastatteluista sekä omat testaukset vanhalla hakukoneella. Näiden tietojen avulla sekä omien kokemuksien pohjalta muodostin tavoitteet ja raamit miten tulevan hakukoneen piti toimia ja mitä sen piti sisältää. Työlle ei annettu tiukkaa aikarajaa vaan tärkeintä oli saada toimiva lopputulos.

Vaikeinta koko työssä oli Apache Solrin asetuksien määrittäminen. Vaikka itselläni ei ollut aiempaa kokemusta Java-ohjelmien pyörittämisestä verkkopalvelimella, oli sen käyttöönotosta tehty hyvin yksinkertainen. Eniten päänvaivaa tuotti osittajien luominen ennakoivaa tuotehakua varten. Suurin ongelma oli se, että valmiit suodattimet oli tarkoitettu vakiopituisten sanojen luomiseen. Kuvassa 9 on selitetty, miten lopulta saimme sanojen pilkkomisen toimimaan oikein.

Pääsin hyödyntämään omia ohjelmointitaitojani, kun tein omaa testisivustoa. Itse osasin jo ennestään sekä html- että css-kieltä. Sen sijaan jouduin opettelemaan JavaScriptiä ja JQuery-kirjaston käyttöä ennakoivaa hakua varten. JQueryn sivustolla oli erinomaiset esimerkit, jonka avulla sain tehdyksi ennakoivan haun testisivustolle.

Testaaminen oli myös merkittävä osa työtäni. Valitettavasti en voi selittää ja kuvata enempää testauksia opinnäytetyössäni, sillä en saanut lupaa esitellä tuloksia. Testaaminen toteutettiin niillä sanoilla ja fraaseilla, jolla oli ongelmia

vanhalla hakukoneella. Ennakoivaa hakua testailtiin enemmän yleisempiä tuotteita hakemalla.

Pääsimme toivottuun lopputulokseen Apache Solrilla. Tuotehausta saatiin sekä tarkka että nopea ja samalla palvelimen kuormitusta pienennettiin huomattavasti. Ennakoiva haku, joka oli uusi ominaisuus tuotehaulle, toteutettiin onnistuneesti ja siitä saatiin Googlemainen haku.

Hakukoneen kehittäminen jatkuu edelleen kyseisessä yrityksessä. Vaikka haku toimii hyvin, tarkkuutta lisätään sen mukaan, kun saadaan palautetta käyttäjiltä. Hakuun on tarkoitus vielä lisätä ennakoivan haun ominaisuuksia, kuten se että se listaa varastossa olevat tuotteet suuremmalle painoarvolle sekä nostaa niitä tuotteita enemmän, jotka ovat saaneet hyvät tuotearvostelut. Näiden ominaisuuksien lisäksi verkkosivustolle on tarkoitus saada tarkennettu tuotehaku, joka hyödyntäisi nykyistä järjestelmää.

LÄHTEET

AIIM 2014, hakupäivä 15.12.2014, http://www.aiim.org/What-is-Enterprise-Search

Apache Solr 2015, hakupäivä 1.2.2015, http://lucene.apache.org/solr/features.html

Apache Tomcat 2014, hakupäivä 15.12.2014, http://tomcat.apache.org/

Chopra Vivek, Sing Li, Genender Jeff. 2007, Professional Apache Tomcat 6 Indianapolis: Wiley Publishing, Inc.

Ellis, Graham J. 2004, Tomcat Overview, http://www.wellho.net/downloads/A651.pdf

Notepad ++ 2015, hakupäivä 1.2.2015, http://notepad-plus-plus.org/

Oxford Dictionaries 2015, hakupäivä 28.1.2015, http://www.oxforddictionaries.com/definition/english/search-engine

Smiley David, Pugh Eric. 2009, Solr 1.4 Enterprise Search Server. Birmingham: Packt Publishing Ltd.

The Java EE 5 Tutorial 2010, hakupäivä 10.1.2015, http://docs.oracle.com/javaee/5/tutorial/doc/bnafd.html

Webopedia 2015, hakupäivä 7.2.2015, http://www.webopedia.com/TERM/B/browser.html

W3Schools 2015, Browser Statics January 2015, hakupäivä 7.2.2015, http://www.w3schools.com/browsers/browsers_stats.as