

Niko Laasonen

# WEB CRAWLER

## Case: WorldSome Oy

Opinnäytetyö  
Tietojenkäsittely

Maaliskuu 2016




MAMK

University of Applied Sciences

# KUVAILULEHTI

 <b>MAMK</b> University of Applied Sciences	<b>Opinnäytetyön päivämäärä</b>  4.5.2016
<b>Tekijä(t)</b> Niko Laasonen	<b>Koulutusohjelma ja suuntautuminen</b> Tietojenkäsittelyn koulutusohjelma
<b>Nimeke</b> Web crawler – Case: WorldSome Oy	
<b>Tiivistelmä</b>  <p>Tässä opinnäytetyössä tutkin web crawlereiden toimintaa ja toteutin WorldSome Oy:lle tarkoituksiinsa sopivan web crawler -palvelun käyttöliittymineen. Työssä oli oleellista selvittää voidaanko vaatimuksien mukainen sovellus toteuttaa käytännöllisellä tavalla, jolloin asiakkaat saisivat kuvaillun mukaisen palvelun, jota on helppo käyttää. Palvelua on tarkoitus soveltaa hakemalla asiakkaiden yrityksistä ja tuotteista tietoa erilaisilta sivustoilta internetistä.</p> <p>Opinnäytetyössä käytiin läpi eri vaihtoehtoja työn toteuttamiseksi sekä lyhyesti web crawlereiden historiaa. Toteutuspohjaksi valikoidun OpenSearchServerin toimintaan perehdyttiin myös web crawler -lähtöisessä mielessä. Opinnäytetyön tuloksena on toimiva web crawler -palvelu ja käyttöliittymä, jota on mahdollista muokata moneen tarkoitukseen sopivaksi.</p>	
<b>Asiasanat (avainsanat)</b>  Web crawler, ohjelmointi, OpenSearchServer, tiedonlouhinta, avoin lähdekoodi	
<b>Sivumäärä</b>  37	<b>Kieli</b>  Suomi
<b>Huomautus (huomautukset liitteistä)</b>	
<b>Ohjaavan opettajan nimi</b> Arto Väättäinen	<b>Opinnäytetyön toimeksiantaja</b> WorldSome Oy

## DESCRIPTION

 <b>MAMK</b> University of Applied Sciences	<b>Date of the bachelor's thesis</b>  May 4, 2016
<b>Author(s)</b> Niko Laasonen	<b>Degree programme and option</b> Business Information Technology
<b>Name of the bachelor's thesis</b> Web crawler – Case: WorldSome Oy	
<b>Abstract</b>  <p>In this thesis I researched the operation of web crawlers and implemented a web crawler service with a user interface for WorldSome Oy's needs. It was essential in this bachelor's thesis to study whether it would be possible to implement the specified kind of service in a practical way, which at the same time would be user-friendly for the end users. The purpose of the service was to apply it for searching information about the customers' companies and products on the Internet.</p> <p>This thesis introduced different options for creating the service and a brief history of web crawlers. It included web crawler oriented familiarization of OpenSearchServer, the base of implementation for the service. This bachelor's thesis resulted in a finished web crawler service with a user interface, which could be customized to meet many needs.</p>	
<b>Subject headings, (keywords)</b>  Web crawler, programming, OpenSearchServer, data mining, open source	
<b>Pages</b> 37	<b>Language</b> Finnish
<b>Remarks, notes on appendices</b>	
<b>Tutor</b> Arto Väättäinen	<b>Bachelor's thesis assigned by</b> WorldSome Oy

# SISÄLTÖ

1	JOHDANTO .....	1
2	WEB CRAWLER .....	2
2.1	Toimintaperiaate .....	2
2.2	Historiaa ja big data .....	3
2.3	Vaihtoehdot.....	4
3	OPENSEARCHSERVER .....	5
3.1	Indeksi ja web crawler .....	6
3.2	Query ja renderer .....	12
4	KÄYTTÖLIITTYMÄ .....	17
4.1	Asiakkaan näkymä.....	18
4.2	Ylläpitäjän näkymä .....	25
5	PÄÄTÄNTÖ .....	34
	LÄHTEET .....	37

## 1 JOHDANTO

Tässä opinnäytetyössä tutustun aluksi web crawlereiden toimintaan ja niiden toteuttamiseen kartoittaakseni vaihtoehtoja palvelun toteuttamiseksi. Toimeksiantajana toimii WorldSome Oy, joka tarvitsee web crawler -palvelun asiakkailleen. WorldSome on yritys, joka tarjoaa sosiaalisen median sekä uutiskanavien sisältöjen analysointipalveluja. Näitä analyyskejä sitten tarjotaan asiakkaille eri tarkoituksiin, kuten julkisen brandikuvan tulkintaan. Toteutettavan palvelun olisi tarkoitus tarjota asiakkaille käyttöliittymä, jota kautta tutkia verkosta etsittävää tietoa ennalta määrätyiltä verkkosivustoilta omilla hakusanoillansa. Työssä on oleellista tutkia voidaanko vaatimuksien mukainen sovellus toteuttaa käytännöllisellä tavalla, jolloin asiakkaat saisivat kuvaillun mukaisen palvelun, jota on helppo käyttää.

Koska aihe ei ole entuudestaan tuttu, aluksi lähestyn työtä tutkimustyön avulla ja kartoitan mitä jo osaamiani taitoja pystyn työssä hyödyntämään sekä mitä mahdollisesti opiskelen sekä opettelen uusina asioina. Näin aluksi työ kuulostaa minusta juuri sopivalta itselleni, uusi ja haasteellinen työ, jonka tekeminen minua itseäni kiehtoo. Aihe ei ole sama mitä aluksi itse suunnittelin, mutta tämä toimeksiantajani kautta saamani vastine on minusta parempi siltä kannalta, että se tulee oikeaan käyttöön sekä tarpeeseen, jolloin minulla on myös mahdollista kokea suurempaa saavutuksen sekä onnistumisen tunnetta työni hedelmistä saatua projektin loppuun.

Työnteko tapahtuu itsenäisesti kahden kuukauden ajan etätyöskentelynä käsin, työtunteja on n. 7,5 per arkipäivä. Tämä tuntuu minusta sopivalta tavoitteiden sekä työn päämäärän saavuttamiseksi. Minulla on toimeksiantajan kanssa säännöllisin välein Skype -palavereita, joille on annettu etukäteen tavoitteet, joita saavuttaa niihin mennessä. Palavereissa käydään myös läpi tehdyt asiat ja seuraavan palaverin tavoitteet.

Toisessa luvussa käyn läpi web crawlereiden historiaa ja niiden käytännön toimintaperiaatteita. Tämä on oikeastaan työni ensimmäinen konkreettinen vaihe, koska joudun tekemään alustavan tutkimustyön myös tekniikoiden sekä vaihtoehtojen kartoittamiseksi.

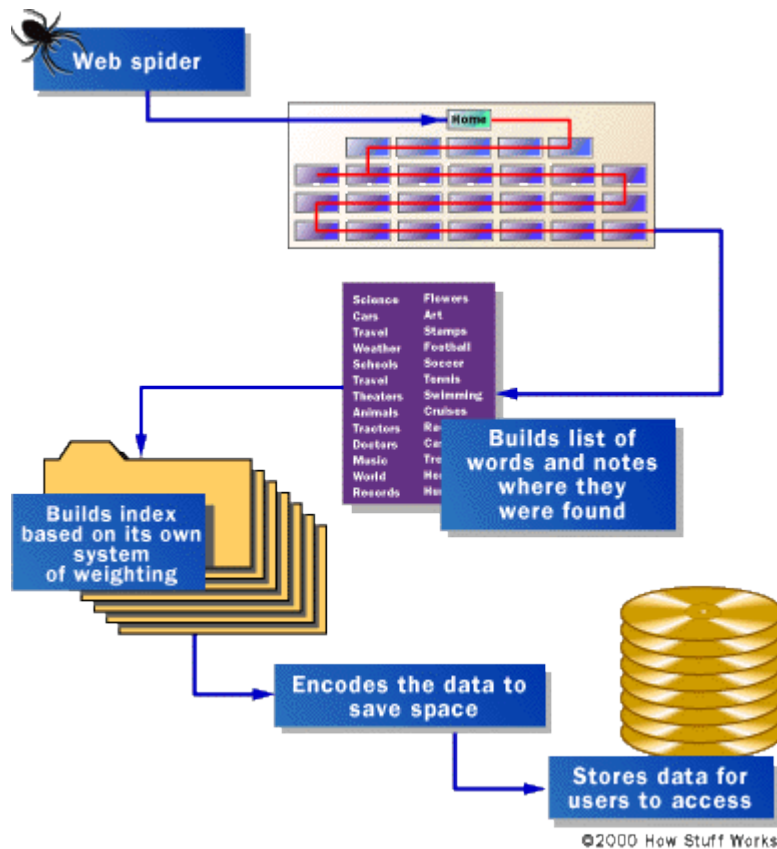
Kolmannessa luvussa alkaa käytännön toteutuksen kuvaus, sekä käydään tarkemmin läpi toimeksiantajan työn määrittely. Tässä luvussa ilmenee myös valitut tekniikat, sekä minkälaisista osista työ itseasiassa oikein loppujenlopuksi koostuu.

## **2 WEB CRAWLER**

Työn olennaisin ja myös suuri osa on alustava tutkimustyö, jota täytyy tehdä uuden aiheen kanssa. Tutkimustyön avulla pystytään syventävämmiin selvittämään kuinka voidaan saada valmis palvelu tuotettua. Sillä päästään myös päätöksiin käytännön toteutuksessa vaadittavien erilaisten tekniikoiden ja ratkaisujen kanssa.

### **2.1 Toimintaperiaate**

Web crawlerit ovat ohjelmistoja, jotka käytännössä lataavat mittavalla skaalalla verkkosivuja talteen. Näistä tallennetuista sisällöistä pääasiallisesti luodaan niin sanottuja indeksejä, joille käyttäjät voivat lähettää kyselykutsuja. Tämä on hakukoneiden pohjimmainen toimintaperiaate. Olston ja Marc kirjoittavat (2010), että hakukoneissa hyödyntämisen lisäksi web crawlereilla on kaksi muuta merkittävää käyttötarkoitusta. Niitä käytetään myös webarkistointiin, eli verkkosivustoja arkistoidaan kokonaisuudessaan vaikka myöhempää tarkastelua tai tutkimustyötä varten, mikäli ne syystä tai toisesta katoavat internetistä. Kolmas mahdollinen käyttötarkoitus web crawlereille on data mining, suomeksi tiedonlouhinta, jossa verkkosisällöistä saatua tietoa analysoidaan erilaisia tarkoituksia varten. Näitä voisivat olla esimerkiksi myyntitilastojen laskenta tai aluekohtaisen markkinointisegmentointi tiedon selvittäminen.



**KUVA 1. Web crawlereiden toimintaperiaate (Kurt 2000)**

Web crawlereiden toimintaperiaate yksinkertaistettuna alkaa siitä, että ohjelma käy läpi sille määritettyjä sivustoja tai muuta sisältöä ja ottaa tietoja niiltä talteen samalla merkiten talteen mistä tarkalleen nämä tiedot ovat löytyneet (kuva 1). Tämän jälkeen ohjelma rakentaa näistä kaikista tiedoista indeksin, jonka tiedoille annetaan painoarvoja oman crawler -kohtaisen luokittelun mukaisesti. Kaikki nämä tiedot myös yleensä enkoodaataan, eli muutetaan vähemmän tilaa vievään formaattiin, koska tietoa voi olla kuinka paljon tahansa. Nämä käsitellyt tiedot viimein tallennetaan tietokantaan tai muualle, josta käyttävät voivat päästä niihin käsiksi ja etsiä niistä vaikka jotain määrättyä tietoa.

## 2.2 Historiaa ja big data

Web crawlereiden ohjelmoimisen voidaan katsoa alkaneen niinkin aikaisin, kuin 1993. Silloin niitä käytettiin lähinnä ainoastaan tilastotietojen ja muun verkon informaation keräämiseen. Vuonna 1994 oli olemassa jo kuusi web crawler -ohjelmistoa, World Wide Web Wanderer, Jump Station, World Wide Web Worm, RBSE spider, WebCrawler sekä MOMspider (Mirtaheri ym. 2014). Vajaan vuoden

kehityksen sisällä tekniikka kuitenkin teki suuren loikan, indeksoitujen sivustojen lukumäärä nousi yli sadasta tuhannesta noin kahteen miljoonaan. Vielä tästä muutamia vuosia myöhemmin ensimmäiset kaupalliset crawlerit tulivat saatavaksi. Vuonna 1998 täysin uusi suuremman skaalan web crawler nimeltänsä Google luotiin ratkaisemaan juurikin internetin laajempaa tiedonlouhinnan tarvetta. Se oli ensimmäisiä web crawlereita, jotka osasivat vähentää levyjen lukuaikaa pakkaamalla ja indeksoimalla tietoja. Se myös hyödynsi algoritmia nimeltä PageRank, joka laskee mahdollisuuden, jolla käyttäjä vierailisi sivustolla ja järjestää hakutulokset tämän mukaan. Ensimmäinen versio Googlesta pystyi lataamaan peräti 100 verkkosivua sekunnissa. Lopun Googlen tarinasta useimmat tietävätkin nykyajan suurimpana hakukonejättinä. Näiden vuosien jälkeen on tapahtunut kyseisen teknologian saralla lukuisia kehitysaskelaita ja jopa jonkin asteen läpimurtoja. Olen varma, että jokainen internetiä käyttänyt on ollut siis web crawlereiden kanssa tekemisissä tietämättäänkin.

McKinsey Global Instituten mukaan (2011) niin kutsutusta “big datasta” tulee kovaa vauhtia yritysten kilpailukyvyn kulmakiviä. Big data käsittää isoja kokonaisuuksia informaatiota ja tietoja, joita käydään läpi sekä analysoidaan jonkinlaisen hyödyn aikaansaamiseksi. Tällainen tarkoituksena voisi esimerkiksi olla tarkemman segmentoinnin tavoittelu, analysoidessa tuotteisiin, palveluihin sekä asiakkaisiin liittyviä tietoja. Big datan olennainen hyöty tiedon ollessa sähköisessä muodossa tulee sen läpinäkyvyydestä ja nopeakäyttöisyydestä. Potentiaaliset hyödyt big datasta ylettyvät jokaiselle alalle, joilta tilastotietoja tai muunlaista informaatiota on vain kerättävissä. Juurikin web crawlerit ovat yksi tapa kerätä big dataa mitä erilaisimpia analysointia varten.

## 2.3 Vaihtoehdot

Web crawlereiden ohjelmoiminen tyhjästä on hyvinkin työlästä ja vie paljon aikaa, joten tyydyin tutustumaan crawlereiden toimintaan ja testaamaan yksinkertaista linkkien keräystä PHP:llä ohjelmallisesti (James 2010). Tämän eräänlaisen pikaperehdytyksen jälkeen tiedonlouhinnan maailmaan aloin kaavoittaa minkälaisia valmiita ratkaisuja minulla voisi olla käytössäni ja mitkä tai miten ne sopisivat käyttötarkoitukseeni. Web crawler -ohjelmistoja ja frameworkoja on olemassa maksullisia versioita, mutta myös varsin runsaasti avoimen lähdekoodin omaavia projekteja sekä valmiita kokonaisuuksia. Yksi joka minulla tuli heti alustavaa



tutkimustyötä tehdessäni vastaan oli PHP Crawler, joka nimensä mukaan toimii PHP:llä ja käyttää MySQL:ää. Vuonna 2006 valmiiksi ohjelmoitu PHP Crawler on edelleen yksi maailman suosituimmista PHP-pohjaisista web crawler -ohjelmistoista (Fedorkov 2011). Jatkoin kuitenkin vielä web crawlereiden maailmaan perehtymistä ja vaihtoehtojeni tutkimista. Kaksi muuta vastaani tullutta vaihtoehtoista avoimen lähdekoodin ohjelmistoa olivat Python-pohjainen Scrapy framework sekä Java-pohjainen Norconex HTTP Collector. Norconexin ohjelmiston tiputin pois listaltani, koska sen käyttöönotto olisi vaatinut Javan käyttämistä (Norconex 2013), mikä ei ole vahvin osaamisalueeni, enkä halunnut työlleni liikaa painoa ohjelmointikielen lisäopiskelulle. Katseeni kääntyi siis Scrapyyn, joka on hyvin laajalti käytetty eri kaupallistenkin yritysten toimesta heidän tiedonlouhinta ja hakukone tarpeisiinsa (Scrapy 2016). Kesken syventymistä Scrapyn toimintoihin löysin kuitenkin OpenSearchServerin, jolla oli näistä kaikista laajin tuki eri kielille ja joka tarjosi samaan aikaan palvelin toimintoa. OpenSearchServer perustuu Apache Luceneen, Javalla kirjoitettuun tekstihakukirjastoon, ja sen mukana tulee paikallisesti suoritettava Apache TomCat-palvelin (OpenSearchServer 2014), jolla OSS:ää voi ajaa ja testata. OpenSearchServerin laajojen mahdollisuuksien sekä ominaisuuksien takia valitsinkin juuri sen työni pohjalle varsinaista palvelun web crawl osuutta varten. Näin minulla oli nopea ja kätevä aloittaa suoraan työn tekemistä.

### **3 OPENSEARCHSERVER**

Olen perehtynyt aiheeseen ja valinnut käyttämäni ohjelmiston palvelun pohjalle. Seuraava looginen askel on alkaa kokoamaan siitä haluttua kokonaisuutta käyttöliittymineen ja toiminnallisuuksineen. OpenSearchServerin lisäksi työkaluina on käytettävissä Netbeans kehitysympäristö, sillä olen tottunut tekemään sillä ohjelmointityötä.

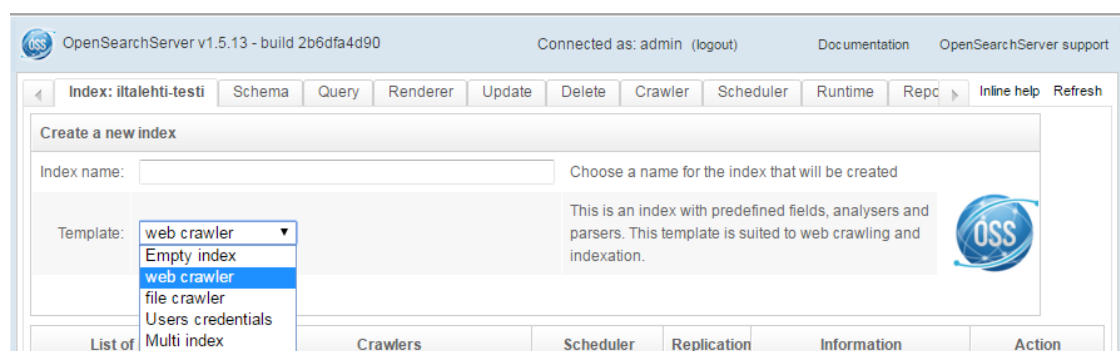
Toteutettavan palvelun määrittely on kokonaisuudessaan kolmiosainen. Ensimmäinen vaihe on manuaalisen haun toteutus, jonka avulla pystytään etsiä sisältöä erilaisilla hakusanoilla samaan aikaan monelta eri verkkosivustolta. Hakusanoja olisi yhdestä viiteen.

Toisessa vaiheessa haun pitäisi tämän lisäksi olla automaattisesti tietyin väliajoin päivittyvä, jolloin hakua ei tarvitsisi uusia aina manuaalisesti. Myös vanhemmat hakutulokset voisivat tippua pois hausta. Hakusanat syötetään itse, mutta verkkosivustot joilta haetaan tietoa määritellään ennalta koodissa.

Kolmannessa, eli viimeisessä, vaiheessa tähän automaattisesti päivittyvään hakuun lisättäisiin käyttöliittymä, jossa on kirjautumismahdollisuudet. Tällöin asiakkailta olisi yksilöllinen liittymä, joka avautuisi omilla tunnuksilla. Käyttäjien tai tilien luonti tehtäisiin etukäteen palvelun hallinnoijien kautta.

### 3.1 Indeksi ja web crawler

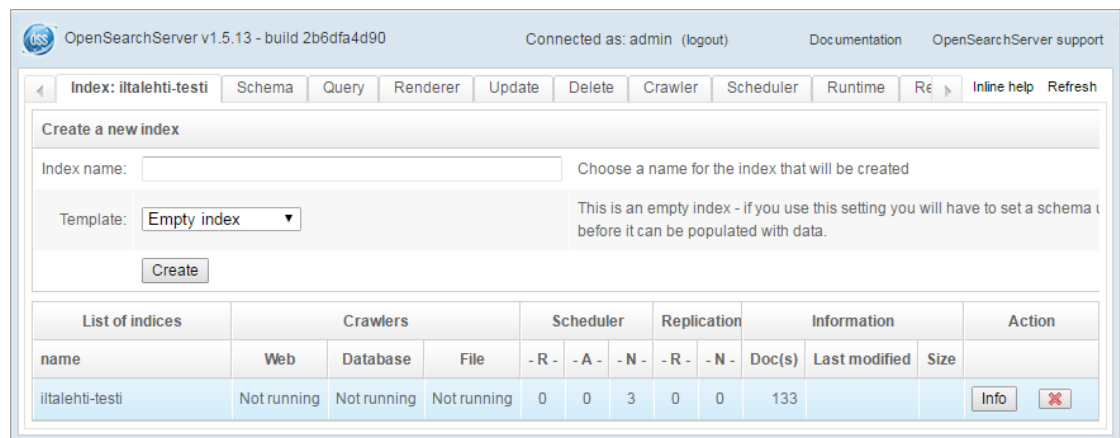
Asennettuani OpenSearchServerin aloitan käynnistämällä mukana tulevan TomCat-palvelimen, jolla OSS toimii. Nyt voin aloittaa itse OSS:n käytön. Heti etusivulla luon uuden indeksin palvelulleni, template valikossa on erilaisia pohjia crawlereille eri käyttötarkoituksia varten (kuva 2). ”Empty index” -valinta luo täysin tyhjän, omanlaiseksensa kustomoitavan indeksin ja ”file crawler” luo indeksin, joka soveltuu tiedostojen jäsentämiseen ja indeksointiin. ”Users credentials” sekä ”Multi index” ovat OSS:n sisäiseen käyttöön tarkoitettuja indeksipohjia, joita käytetään muiden indeksien käyttäjätietojen säilyttämiseen ja vastaavasti muiden indeksien tietojen kokoamiseen.



**KUVA 2. OpenSearchServer indeksin luonti**

Valmis luomani indeksi näkyy nyt listassa nimellä ”iltalehti-testi”, tarkoituksena on hakea sisältöä Iltalehden sivustolta ja suorittaa testauksen nimissä sille kyselyitä. Näin saadaan samalla demottua palvelun toteutuksen ensimmäistä vaihetta. Indeksien nimen vieressä näkyy erilaisten crawl prosessien tila. Verkkosivustojen, tietokantojen sekä

tiedostojen tietoja voi indeksoida erikseen näin halutessaan. Kuvassa kaikki prosessit ovat sillä hetkellä pysäytettynä (kuva 3).



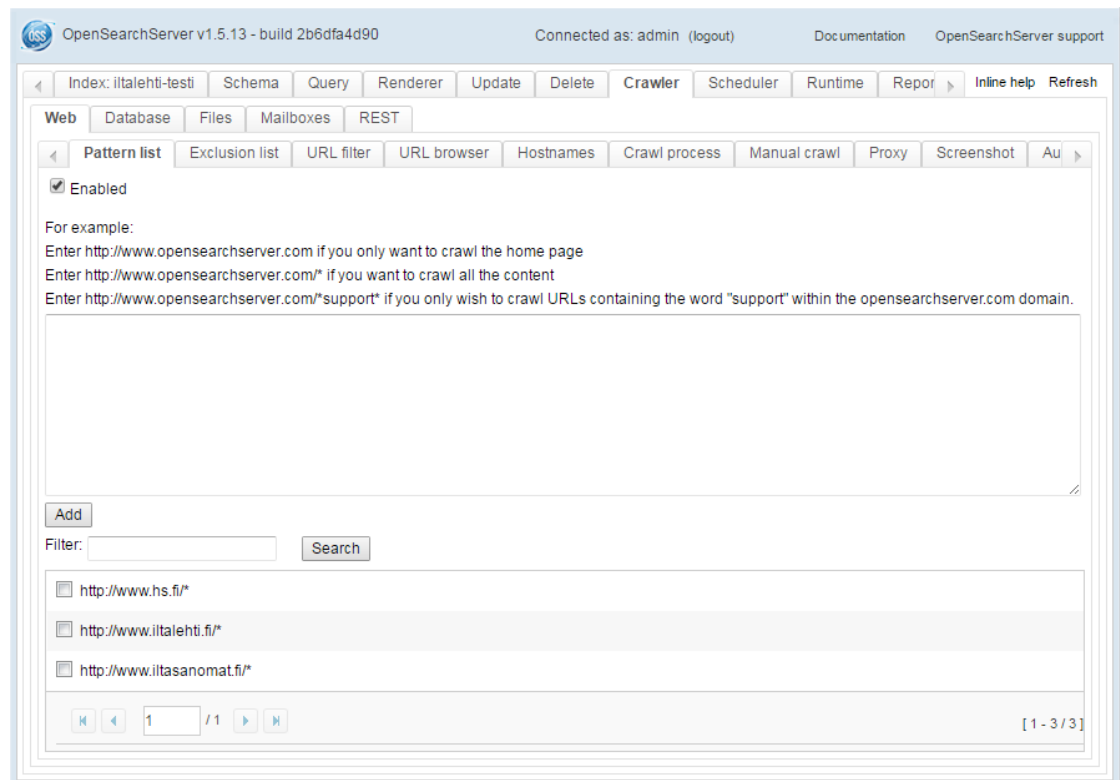
**KUVA 3. OpenSearchServer indeksilistaus ja toimintokatsaus**

Valittuani indeksini ”Schema” -välilehdellä näkyy kaikki tietueet, joita web crawler oletuksena hakee verkkosivuilta. Schema on indeksin rakenne ja määrittää mitkä kaikki tiedot web crawler sivustoja läpikäydessään merkitsee talteen. Näitä ovat mm. sivuston kieli, otsikko, tarkka osoite sekä host, eli sivuston niin kutsuttu pääsivu (kuva 4). Web crawler templatien tietueet sopivat täydellisesti tarvitsemaani käyttötarkoitukseen, joten minun ei tarvitse muuttaa niitä erikseen. ”Indexed” kohta kertoo indeksoidaanko tietue, eli voidaanko sitä etsiä myöhemmin kyselyllä. ”Stored” on lähes sama, mutta tieto säilytetään muokkaamattomana, toisin kuin indeksoidessa. ”TermVector” kohta määrittelee voiko tietueista tehdä snippettejä, eli eräänlaisia lyhennelmiä erilaisia käyttötarkoituksia varten. Kuten kuvasta näkyy, joihinkin tietueisiin on liitetty eräänlaisia analysointoreitoja, jotka yrittävät esimerkiksi poimia sanoja sekä tavuja erilaisista haetuista tiedoista. Näitä voi hyödyntää sitten eteenpäin esimerkiksi samankaltaisten tuloksien näyttämiseen hauissa. Joten esimerkiksi haku ”mies” voisi tuoda tuloksesi myös sisältöä, joissa on sanat ”miestä” tai ”mieheltä” ja niin edelleen.

List of existing fields and their settings					
Name	Indexed	Stored	TermVector	Analyzer	Copy Of
lang	yes	no	no		
title	yes	yes	positions_offsets	TextAnalyzer	
titleExact	yes	no	yes	StandardAnalyzer	[title]
titlePhonetic	yes	no	no	PhoneticAnalyzer	[title]
content	yes	compress	positions_offsets	TextAnalyzer	
contentExact	yes	no	yes	StandardAnalyzer	[content]
contentPhonetic	yes	no	no	PhoneticAnalyzer	[content]
url	yes	no	no		
urlSplit	yes	no	no	TextAnalyzer	[url]
urlExact	yes	no	no	StandardAnalyzer	[url]
urlPhonetic	yes	no	no	PhoneticAnalyzer	[url]
fileName	yes	yes	positions_offsets	TextAnalyzer	
fileNamePhonetic	yes	no	no	PhoneticAnalyzer	[fileName]
fileNameExact	yes	no	no	StandardAnalyzer	[fileName]
full	yes	no	no	TextAnalyzer	[title, content, fileNa
fullExact	yes	no	no	StandardAnalyzer	[title, content, fileNa
fullPhonetic	yes	no	no	PhoneticAnalyzer	[title, content, fileNa
autocomplete	yes	no	no	AutoCompletionAnalyzer	[title, fileName]
metaDescription	no	compress	no		
metaKeywords	no	compress	no		
host	yes	no	no		
subhost	yes	no	no		
contentBaseType	yes	no	no		
backlinkCount	yes	no	no	IntegerAnalyzer	

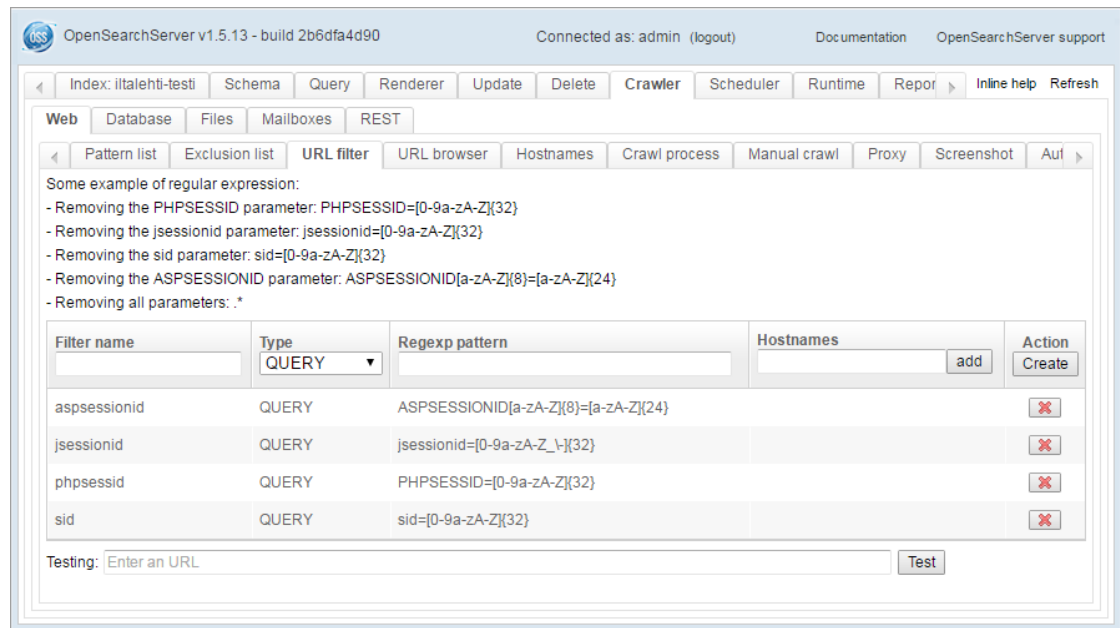
#### KUVA 4. OpenSearchServer web crawlerin tietueet

Crawler –välilehden alta löytyvällä Web- välilehdelle asetamme verkkosivustot, joilta sitten itse sisältöä haetaan, eli patternit (kuva 5). Iltalehden lisäksi olen jälkikäteen lisännyt listalleni myös Helsingin Sanomien sekä Iltasanomien verkkosivustot. Tähtimerkki osoitteiden perässä kertoo, että kaikki sivustolle kuuluvat linkit etsitään läpi etusivulta alkaen. Linkeille voi tehdä erilaisia määritteitä rajatakseen mitä sisältöä sieltä haetaan, vaikkapa Iltalehdeltä pelkät ulkomaiden uutiset voi määrittää erikseen käyttämällä hakusanaa ”ulkomaat” sijoitettuna tähtimerkkien väliin URL:n perässä.



**KUVA 5. OSS web crawler pattern**

Exclusion listillä voidaan rajata tiettyjä osoitteita pois haettavien joukosta, esim. jos verkkokaupan tuotteita hakiessa ei haluta lainkaan linkkejä oheistuotteista, niin oheistuotteiden verkko-osoite lisätään kiellettyjen urlien joukkoon. ”URL filter” -välilehdellä annamme hienovaraisempia rajoituksia osoitteille, tässä noudatan OSS:n esimerkkiasetuksia ja poistan URL-tiedoista PHP, ASP ja JS sessiotiedot sekä SID-parametrit (kuva 6). Erilaisia sivustojen referenssitietoja voisi myös suodattaa näin pois indeksoitavista linkeistä. Tällä hetkellä tarkemmat säädöt kumpaankaan listaan eivät ole tarpeen.



**KUVA 6. OSS web crawler URL filter**

Seuraavaksi täytyy nimetä User-Agent, eli itse crawler elementti. Tämä tarkoittaa käytännössä millä nimellä ohjelmisto esittäytyy verkossa hakiessaan tietoa. Nimeän tässä vaiheessa crawlerini ”NikonTestiCrawler”:iksi ja asetan sille joitakin toiminnallisia rajoitteita. Samoja tietoja voi hakea uudestaan viiden minuutin välein ja crawler odottaa seitsemän sekuntia jokaisen onnistuneen haun jälkeen (kuva 7). Sivulla olevasta napista saa crawlerin suorittamaan toimintonsa joko kerran tai niin kauan kunnes se manuaalisesti otetaan pois päältä. Laitan crawlerini muutamaksi minuutiksi hakemaan indeksoitavaa sisältöä voidakseni demota myöhemmin palvelun ensimmäistä vaihetta.

OpenSearchServer v1.5.13 - build 2b6dfa4d90 Connected as: admin (logout) Documentation OpenSearchServer support

Index: iltalehti-testi Schema Query Renderer Update Delete Crawler Scheduler Runtime F Inline help Refresh

Web Database Files Mailboxes REST

Pattern list Exclusion list URL filter URL browser Hostnames Crawl process Manual crawl Proxy Screenshot

**Crawling parameters**

User-Agent: NikonTestCrawler	Number of URLs to crawl: 1000
Fetch interval between re-fetches: 5 minutes	Maximum number of URLs per host: 100
Number of simultaneous threads: 10	Delay between each successive access, in seconds: 7
Job run when each session ends: BuildAutocompletion	Indexation buffer: 1000
Propagate deletion: <input checked="" type="checkbox"/> propagate	

**Current status**

Not running RunForever Not running - Click to run

**KUVA 7. OSS web crawler -prosessi**

Pidettyäni crawleria jonkin aikaa päällä pelkästään Iltalehden sivustolla ja tämän jälkeen myös Helsingin Sanomien sekä Iltasanomien verkkosivuille, olen saanut jo hieman yli 3000 tulosta indeksiini (kuva 8). Jakauma ei ole tasan, HS.fi tuloksia on 632, iltasanomat.fi tuloksia 652 sekä iltalehti.fi tuloksia 2003. Tämä johtuu siitä, että aluksi testasin crawler -prosessia pelkästään yhdellä URL:illa.

OpenSearchServer v1.5.13 - build 2b6dfa4d90 Connected as: admin (logout) Documentation OpenSearchServer support

Index: iltalehti-testi Schema Query Renderer Update Delete Crawler Scheduler Runtime F Inline help Refresh

Web Database Files Mailboxes REST

Pattern list Exclusion list URL filter URL browser Hostnames Crawl process Manual crawl Proxy Screenshot

Hostname	URL Count
Search	0
www.hs.fi	632
www.iltasanomat.fi	652
www.iltalehti.fi	2003

**KUVA 8. OSS web crawler hostnames**

Tämän jälkeen luon indeksilleni sekä käyttäjä- että ylläpitäjätilit, jotta sen sisältämiin tietoihin voisi käydä käsiksi jollain tapaa. Yksinkertainen "user" -käyttäjä, jolla on oikeuksina vain indeksini luku on luotu, ja "admin" -käyttäjä samaa kaavaa, mutta luonnin yhteydessä klikattu "admin user" sekä "can monitor" -tävät (kuva 9). Käyttäjille annetut salasanat toimivat OpenSearchServerin backend palveluun

kirjautumiseen ja käyttäjille generoidut API-avaimet taas tarvitaan indeksien kaikkien toimintojen suorittamiseksi. Ennen kuin käyttäjiä on luotu, kaikki tiedot ovat avoimia ilman mitään tunnistautumista. Käyttäjien luominen on siis erittäin tärkeä ja välttämätön askel.

OpenSearchServer v1.5.13 - build 2b6dfa4d90 Connected as: admin (logout) Documentation OpenSearchServer support

Render Update Delete Crawler Scheduler Runtime Reports Replication Scripts Privileges Inline help Refresh

Edit the user user

User name:  Choose a name for the user that will be created

Password:  Choose a password for the user that will be created

Confirm password:  Retype the password

Admin user: ☐ Check the box if the user is admin

Can monitor: ☐ Check the box if the user is able to monitor

API Key:  Key to be used with API

Add privileges on following index:

Index	Role	
iltalehti-testi	Index: query the index	<input type="button" value="remove"/>

List of existing users - click on a user to edit it

admin
user

**KUVA 9. OSS web crawler oikeudet**

Nämä tehdyt askeleet kattavat web crawlerin perustoiminnan. Crawler poimii linkkejä halutuilta sivustoilta talteen valmiiksi määriteltyyn indeksiin, mutta tietojen tarkistelulle ei ole vielä mitään toiminnallisuutta tässä vaiheessa.

### 3.2 Query ja renderer

Tietoon käsiinpääsemiseksi täytyy indeksille rakentaa query, eli kysely. Kysely käy läpi indeksoidut tietueet hakusanan tai hakusanojen perusteella ja palauttaa ne tietueille määriteltyjen painoarvojen perusteella. Uuden kyselyn luominen käy nopeasti ”Query” -välilehdeltä (kuva 10). Query tyyppejä on kuusi erilaista. Field search, pattern search, spell check, more like this, named entity extraction ja document(s). Field search, eli tietue haku, sopii tarkoituksiimme parhaiten, eikä muita



hakuja tarvita. Tarvittaessa niitä voi kuitenkin soveltaa erilaisiin käyttötarkoituksiin, kuten niiden nimistäkin käy ilmi.

OpenSearchServer v1.5.13 - build 2b6dfa4d90

Connected as: admin (logout) Documentation OpenSearchServer support

Index: italehti-testi Schema **Query** Renderer Update Delete Crawler Scheduler Runtime F Inline help Refresh

Create a new query

Query name  Enter the name of the new query

Type **Search (field)** Select the type of the new query

New query... New copied query...

List of existing queries

Name	Type	Info	Action
search	Search (field)	[title - Term & phrase - 10.0/10.0, titleExact - Term & phrase - 10.0/10.0, titlePhonetic - Term & phrase - 10.0/10.0, content - Term & phrase - 1.0/1.0, contentExact - Term & phrase - 1.0/1.0, contentPhonetic - Term & phrase - 1.0/1.0, urlSplit - Term & phrase - 5.0/5.0, urlExact - Term & phrase - 5.0/5.0, urlPhonetic - Term & phrase - 5.0/5.0, fileName - Term & phrase - 10.0/10.0, fileNameExact - Term & phrase - 10.0/10.0, fileNamePhonetic - Term & phrase - 10.0/10.0, full - Term & phrase - 0.1/0.1, fullExact - Term & phrase - 0.1/0.1, fullPhonetic - Term & phrase - 0.1/0.1]	Edit Delete

**KUVA 10. OpenSearchServer kyselylista**

Kyselyn voi rajoittaa tietylle määrälle sivuja ja aloitus hakutuloksen offsetin voi asettaa myös etukäteen. Oletus hakuoperaattoriksi asetetaan "OR", jotta monen hakusanan haut olisivat mahdollisia. Vaihtoehtoinen operaattori on "AND", joka mahdollistaa vain "joko tai" haun. "OR" -operaattori taas palauttaa myös tulokset joissa kaikki hakusanat, tai jotkin niistä esiintyvät. Haun kielen voi erikseen määrittellä, jotta esimerkiksi samankaltaisia tuloksia voidaan tarjota tarkemmin (kuva 11).

OpenSearchServer v1.5.13 - build 2b6dfa4d90

Connected as: admin (logout) Documentation OpenSearchServer support

Index: italehti-testi Schema **Query** Renderer Update Delete Crawler Scheduler Runtime F Inline help Refresh

Search (field): search

Enter the query:

Empty query string returns all documents: ☒

Default operator: **OR**

Start offset:

Number of rows:

Phrase slop:

Language: **Finnish**

Search

**KUVA 11. OSS kysely**

Eri tietueille annetaan erilaiset painoarvot kyselyyn. Term boost koskee yksittäisen sanan haun yhteydessä painottamista, Phrase boost taas lauseen sisällä olevan hakutuloksen painottamista (kuva 12). Otsikon ja tiedostonimen tietueille on annettu suurimmat painoarvot, koska niiden perusteella käyttäjä todennäköisimmin etsii haluamaansa sisältöä indeksistä. Kaikilla tietueilla kuitenkin on edes jonkinlainen painoarvo, jotta ne otetaan huomioon hakutuloksissa.

OpenSearchServer v1.5.13 - build 2b6dfa4d90

Connected as: admin (logout) Documentation OpenSearchServer support

Index: iltalehti-testi Schema Query Renderer Update Delete Crawler Scheduler Runtime F Inline help Refresh

Search (field): search Cancel Save

Field	Mode	Term boost	Phrase Boost	Phrase Stop	Boolean group	Action
title	Term & phrase ▼	10	10			✕
titleExact	Term & phrase ▼	10	10			✕
titlePhonetic	Term & phrase ▼	10	10			✕
content	Term & phrase ▼	1	1			✕
contentExact	Term & phrase ▼	1	1			✕
contentPhonetic	Term & phrase ▼	1	1			✕
uriSplit	Term & phrase ▼	5	5			✕
uriExact	Term & phrase ▼	5	5			✕
uriPhonetic	Term & phrase ▼	5	5			✕
fileName	Term & phrase ▼	10	10			✕
fileNameExact	Term & phrase ▼	10	10			✕
fileNamePhonetic	Term & phrase ▼	10	10			✕
full	Term & phrase ▼	0,1	0,1			✕
fullExact	Term & phrase ▼	0,1	0,1			✕
fullPhonetic	Term & phrase ▼	0,1	0,1			✕

Search

**KUVA 12. OpenSearchServer kyselyn tietueet**

OpenSearchServerin kyselyllä on facetteja. Nämä facetit ovat eräänlaisia monen tiedonlähteen suodattimia, joilla voi lajitella haettua tietoa tarpeen mukaan. Kirjakokoelman kanssa mahdollinen facet voisi olla vaikkapa kirjankannen väri tai kirjallisuustyyppi. Omassa kyselyssäni käytän facetteina host sekä lang tietueita. Nämä ovat siis haetun tiedon kieli sekä mistä sivustolta tieto on löytynyt (kuva 13). Tiedot täytyy löytyä indeksoitavista tietueista valmiiksi, jotta facetit toimisivat.

Field	Multivalued	Post collapsing	Minimal count	Limit	Order By	Action
host	no	no	1			remove
lang	no	no	1			remove
title						add facet

**KUVA 13. OpenSearchServer kysely faceted fields**

Seuraavalla välilehdellä määritellään snippet fieldit. Nämä sisältävät lyhennettyjä versioita tietueiden sisällöstä, jotka täsmäävät hakusanon kanssa. Valitsemani snippetit ovat title, fileName sekä content. Eli linkin otsikko, mahdollinen tiedostonimi ja itse sisältö. Snippettien pituus on maksimissaan 200 merkkiä (kuva14).

Field	Fragmenter	Tag	Separator	Max snippet siz	Max snippet nur	Time limit	Action
title	NoFragmente	b	...	200	1	0	Remove
fileName	NoFragmente	b	...	200	1	0	Remove
content	SentenceFrag	b	...	200	1	0	Remove
metaDescription							add snippet

**KUVA 14. OpenSearchServer kysely snippet fields**

Indeksin luonnin ja web crawler prosessin ajon jälkeen se on saanut sisältöä itselleen. Tämän jälkeen on aika luoda sivusto kyselylleni, jotta indeksoituja tietoja voidaan tarkastella ja hakea. Tässä pääsee alkuun OpenSearchServerin renderereillä. ”Renderer” –välilehdeltä päästään luomaan tällainen (kuva 15).

Name	Request	Action
default	search	View Edit Delete
renderer	search	View Edit Delete

## KUVA 15. OpenSearchServer renderer listaus

Muokkaamme default rendererin asetuksia sopivammaksi, käyttäjälle näkyvät tulostenhaun tekstit kirjoitetaan suomeksi ja kyselyksi asetetaan ”search”. Käyttöön otetaan oletusasetuksilla toimiva autocomplete scripti, joka indeksin tiedoista kokoaa mahdollisia hakutuloksia, joita käyttäjä voisi hakea (kuva 16).

## KUVA 16. OpenSearchServer renderer perusasetukset

Rendererin käyttämät tietueet ovat kyselystä palautetut snippetit titlestä sekä contentista ja itse URL (kuva 17). Nämä tiedot esitetään rendererissä kyselyn antamien tulosten kanssa käyttäjälle, jotta linkkien sisältämää tietoa voidaan silmäillä etukäteen.

Field / Snippet join	Replace previous	URL Field	URL Decode	URL Pattern	URL Replace	Css Class	Widget type
FIELD	<input type="checkbox"/>	url	<input type="checkbox"/>				TEXT
SNIPPET : title	false	url	false				TEXT
SNIPPET : content	false		false				TEXT
FIELD : url	false		false				TEXT
FIELD : url	false		false				TOOLS

## KUVA 17. Renderer fields

Kaikki tekemämme muutokset sekä asetukset antavat seuraavanlaisen lopputuloksen (kuva 18). Nämä rendererin asetukset antavat käyttäjälle tarpeeksi vuorovaikutusmahdollisuuksia tiedonhaussa ja jäsentämisessä, ylläpitäen samalla tavoiteltua käyttäjäystävällisyyttä ja helppokäyttöisyyttä.

Lang

fi (132)

Host

[www.hs.fi \(6\)](#)  
[www.iltasanomat.fi \(14\)](#)  
[www.iltalehti.fi \(112\)](#)

Hae sisältöä lehtien sivustoilta

nainen mies auto

132 tulosta löytynyt (0.145s)

[Noora Rädyn ura naisten maajoukkueessa koki kovan takaiskun](#)  
Kolmea Soldiers of Odinin takkeihin pukeutunutta **miestä** epäillään pahoinpitelystä Imatralla - Kotimaan uutiset - 11:33...Saunaseura siunaa "Siltsumittarin" - Viihde - 11:32...Uudet sähköpyörät...  
[http://www.iltalehti.fi/jaakiekko/2016030921240753\\_jk.shtml](http://www.iltalehti.fi/jaakiekko/2016030921240753_jk.shtml)

[Yön NHL-tulokset](#)  
Kolmea Soldiers of Odinin takkeihin pukeutunutta **miestä** epäillään pahoinpitelystä Imatralla - Kotimaan uutiset - 11:33...Saunaseura siunaa "Siltsumittarin" - Viihde - 11:32...Uudet sähköpyörät...  
[http://www.iltalehti.fi/nhl/2016030921240548\\_nh.shtml](http://www.iltalehti.fi/nhl/2016030921240548_nh.shtml)

[Käännä Adelen kuvaa - saatat yllätyä](#)  
**Mies** ei saanut kunnalta lupaa rakentaa autotallia - keksi luovan ratkaisun...Kuukauden luetuimmat...1. ...Käännä Adelen kuvaa - saatat yllätyä...2. ...Koira luuli olevansa ovela - löydätkö...  
[http://www.iltalehti.fi/fiifi/201603030154501\\_fd.shtml](http://www.iltalehti.fi/fiifi/201603030154501_fd.shtml)

[Vuosisadan lautapeliottelu - Googlen tekoäly voitti ensimmäisen pelin go-suurmestaria vastaan](#)  
Malmin **nainen** vaatii syytteiden hylkäämistä: "Mitään näyttöä syyllisyydestä ei ole" - Kotimaan uutiset - 12:07...Pingviini ui vuosittain 8000 kilometriä tavatakseen pelastajansa - Fiidi.fi - 12:00  
[http://www.iltalehti.fi/digi/2016030921241065\\_du.shtml](http://www.iltalehti.fi/digi/2016030921241065_du.shtml)

## KUVA 18. Renderer ajaa queryn

Tässä vaiheessa työn ensimmäinen osa on saavutettu, indeksistä voidaan manuaalisesti hakea eri hakusanoilla helposti tietoa. Seuraava askel on käyttöliittymän luominen, jotta haku saataisiin automaattiseksi sekä kirjautumisen toteutus käyttäjille.

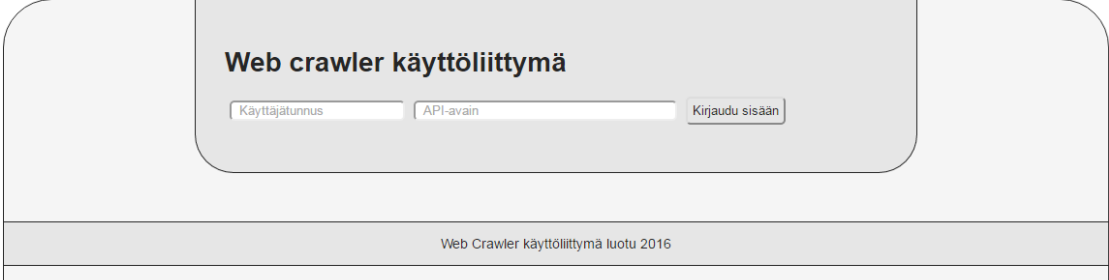
## 4 KÄYTTÖLIITTYMÄ

Käyttöliittymää lähdin toteuttamaan Netbeans kehitysympäristössä HTML:llä käyttäen apuna CSS-tyylitiedostoa ja jQuery-skriptejä. Niin kutsuttu admin-näkymä käyttää lisäksi hyväkseen PHP:llä ohjelmoitua XML-muokkausskriptiä. Suunnittelin

käyttöliittymän harmaasävyiseksi ja simppeliksi, jotta sitä voitaisiin helposti kehittää eteenpäin ilman minkäänlaisia ongelmia.

#### 4.1 Asiakkaan näkymä

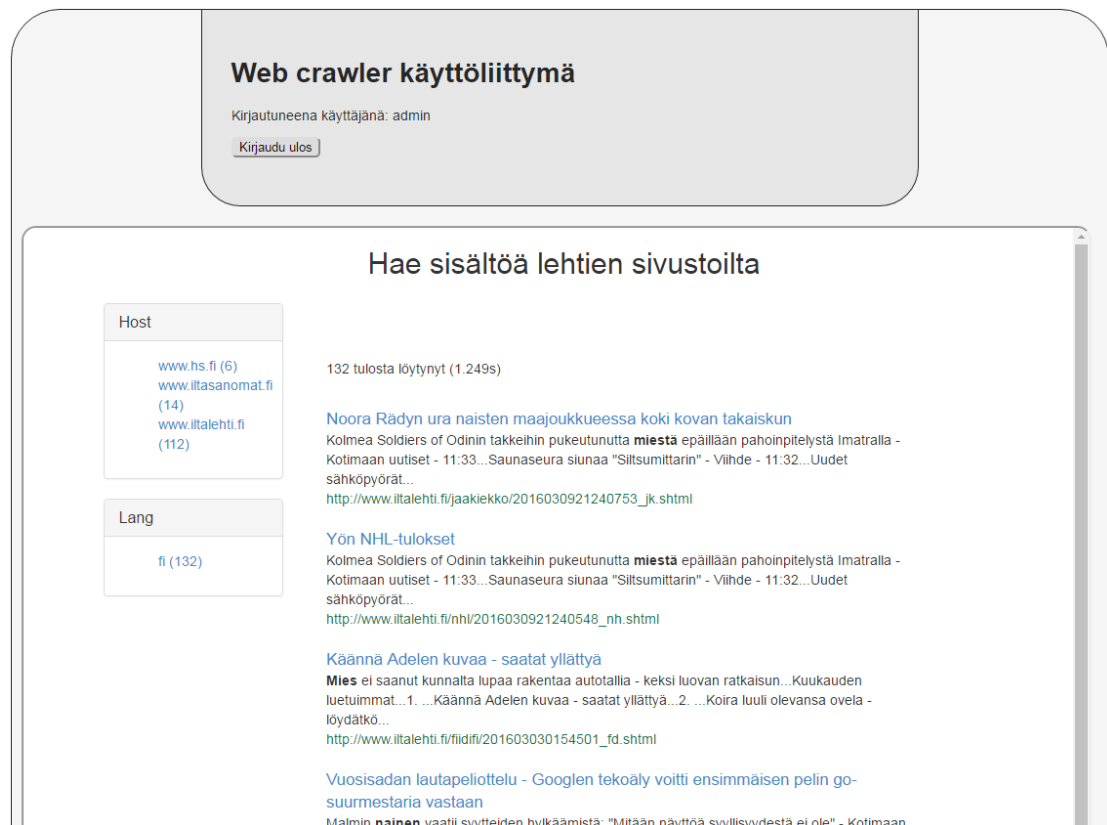
Käyttäjän on nopea päästä palveluun käsiksi. Palvelun ylläpitäjä luo käyttäjälle etukäteen tunnukset, jonka API-avaimella tämä sitten pääsee kirjautumaan palveluun tämän käyttöliittymästä (kuva 19).



The image shows a login interface for a web crawler. It features a light gray background with a central white rounded rectangle containing the title "Web crawler käyttöliittymä". Below the title are two input fields: "Käyttäjätunnus" and "API-avain", followed by a "Kirjaudu sisään" button. At the bottom of the interface, a footer bar contains the text "Web Crawler käyttöliittymä luotu 2016".

**KUVA 19. Käyttöliittymän etusivu**

Kirjautumisen onnistuttua käyttöliittymä hakee sisällensä iframe-elementtiin sisällön, joka toimii OSS:n rendererin pohjalta (kuva 20). Hakupalkki on piilotettu CSS:ssä ja hakusanat annetaan rendererille ohjelmallisesti kirjautumisen yhteydessä. Samalla kirjautumisessa käytetty API-avain tunnistautuu OSS:n palvelimella ja hyväksyy indeksin kyselykutsun käyttöliittymälle. Iframen sisällä oleva tieto on siis edelleen yhtä helppo käyttäjälleen navigoida kuin OSS:n rendererin aikaisemmin luoma sisältö. Tiedot iframe-elementin sisällä päivittyvät tunnin välein, kunnes käyttäjä kirjautuu ulos. Näin saadaan käyttäjälle mahdollisimman uutta tietoa, mikäli web crawler on sitä verkosta kerennyt löytää ja indeksoida.



**KUVA 20. Käyttöliittymä kirjautuneena**

HTML-koodi on hyvinkin yksinkertainen. Se sisältää perus verkkosivurakenteen, johon on linkitetty luomani tyyli- sekä skriptitiedostot ja jQueryn tuotantoversio skripti. Div elementit muodostavat sisältö-, otsikko-, painike-, data- ja alatunnistelaatikot, joiden avulla sivuston sisältöä on helppo hallita ja muokata vastaisuudessa (kuva 21). Painike- ja data-laatikoiden sisältöä muokataan ohjelmallisesti jQuery-skriptin kautta.

```

<!DOCTYPE html>
<html>
  <head>
    <title>Web crawler käyttöliittymä</title>
    <meta charset="UTF-8"/>
    <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
    <link rel="stylesheet" href="tyylit.css"/>
    <script src="js/jquery-1.12.2.min.js"></script>
    <script src="js/skriptit.js"></script>
  </head>
  <body>
    <div id="contentbox">
      <div id="headerbox">
        <h1>Web crawler käyttöliittymä</h1>

        <div id="box1">
          <input id="login" placeholder="Käyttäjätunnus"/>
          <input id="key" size="33" placeholder="API-avain"/>
          <input type="button" id="nappi01" value="Kirjaudu sisään"/>
        </div>

        <div id="box2"></div>

      </div>

      <div id="data">
      </div>

      <div id="footerbox"><p>Web Crawler käyttöliittymä luotu 2016</p></div>
    </div>
  </body>
</html>

```

## KUVA 21. HTML-koodi

CSS-tyylitiedostossa on määritetty käyttöliittymän käyttämä fontti, sekä elementtien harmaansävyt, joita se oletuksena tässä ensimmäisessä versiossa käyttää (kuva 22). Sisältöelementeissä on käytetty nurkkien pyöristystä, jotta käyttöliittymän ulkoasu ei olisi yksinkertaisuudessaan liian karu, vaan pyrkisi viestimään käyttäjälleen samalla käytettävyyden virtaviivaisuutta ja helppoutta.



```
#contentbox {  
    position: relative;  
    border-radius: 50px 50px 5px 5px;  
    padding: 0px 0px 20px 0px;  
    border: solid 1px;  
    background-color: #F5F5F5;  
    font-family: arial,sans-serif;  
    font-size: 14px;  
    color: #262626;  
    overflow: auto;  
}  
  
#headerbox {  
    border:solid 1px;  
    border-top: 0;  
    background-color: #e6e6e6;  
    border-radius: 0px 0px 40px 40px;  
    margin: auto;  
    width: 60%;  
    padding: 30px;  
}  
  
#footerbox{  
    color: #262626;  
    background-color: #e6e6e6;  
    border:solid 1px;  
    border-left: 0;  
    border-right: 0;  
    margin: auto;  
    text-align: center;  
    clear: both;  
}
```

## KUVA 22. Ensimmäinen osa CSS:stä

Tyylitiedostoissa ei myöskään ole liiallisia hienouksia, jotta sitä voitaisiin tehokkaasti kehittää eteenpäin tarpeen näin tulevaisuudessa vaatiessa, mikä on todennäköistä. Suurin osa tyylien asetuksista on elementtien reunuksiin, marginaaleihin sekä toppauksiin perustuvia asetuksia, vaikka seassa on myöskin elementtien jäsentämiseen tarkoitettuja kohtia (kuva 23).

```

#data {
    position: relative;
    clear: both;
    overflow: auto;
    margin: auto;
    margin-top: 10px;
    margin-bottom: 20px;
    padding: 10px 10px 10px 10px;
}

input {
    border-radius: 5px 5px 5px 5px;
}

iframe {
    border-radius: 15px 15px 15px 15px;
    height: 1000px;
    display: inline-block;
    margin: 0;
    padding: 0;
    vertical-align: top;
}

#login,#key {
    padding-left: 8px;
    margin-left: 5px;
}

#nappi01 {
    margin-left: 5px;
    padding: 5px;
    color: #262626;
    background-color: #e6e6e6;
}

```

### KUVA 23. Toinen osa CSS:stä

Sisäänkirjautumisprosessissa alustetaan heti alkuun varmuudeksi datalaatikko-elementti ja poimitaan talteen käyttäjän ilmoittamat käyttäjätunnus sekä API-avain tiedot. Näitä muuttujia käytetään sitten hyväksi data-elementtiin iframea lisättäessä ja ne säilyvät globaaleissa muuttujissa muita funktioita varten. Skripti tarkistaa onko API-avain tyhjä ja käyttäjätunnuksen olemassaolon käymällä läpi XML-tiedoston, johon on muutamia käyttäjiä tehty. Mikäli tiedot ovat virheellisiä, siitä ilmoitetaan käyttäjälle. Muussa tapauksessa käyttäjäkohtaiset hakusanat otetaan talteen (kuva 24).

```

$(function () {
    var login = "";
    var key = "";

    $("#nappi01").click(function () {
        $("#data").empty();

        login = $("#login").val();
        key = $("#key").val();

        if (key !== "") {
            var tarkistus = 0;
            var query = "";

            $.ajax({
                type: "GET",
                url: "kayttajat.xml",
                dataType: "xml",
                success: function (xml) {
                    $(xml).find('kayttaja').each(function () {
                        var nimi = $(this).find('nimi').text();
                        var hakusanat = $(this).find('hakusanat').text();
                        if (login === nimi) {
                            tarkistus = 1;
                            query = hakusanat;
                        } // tarkistus if
                    });
                }
            });
        }
    });
});

```

## KUVA 24. Kirjautumisen jQuery-koodia

Skripti luo myös uloskirjautumisnapin käyttäjälle ja tyhjentää kirjautumistietojen elementin. Tämän jälkeen ajastinfunktio alkaa pyörimään taustalla ja tunnin välein alustaa uudelleen data-elementin ja poimii indeksin uusimmat tiedot taas käyttäjän ilmoittamilla tunnustautumistiedoilla (kuva 25). Uloskirjautumisnapin vieressä ilmoitetaan kirjautuneen käyttäjän nimi ja annetaan painike, josta voi laittaa päivityksen pois käytöstä.

```

}); //XML -each

if (tarkistus === 1) {
    $("#data").append("<iframe id='iframe' "+
        "src='http://localhost:9090/renderer?query="+query+"&use=iltalehti-testi&login=user&key="+key+"&name=default' "+
        "scrolling='auto' width='99%'><p>Selaimesi ei tue iframe -elementtejä.</p></iframe>");

    $("#box2").append("<p>Kirjautuneena käyttäjänä: "+login+"</p> <input type='button' id='nappi02' value='Kirjaudu ulos'/>");
    $("#box2").append("<input type='button' id='nappi03' value='Ota päivitys pois käytöstä'/>");

    $("#login").val("");
    $("#key").val("");
    $("#box1").empty();

    setInterval(function () {
        $("#data").empty();
        $("#data").append("<iframe id='iframe' "+
            "src='http://localhost:9090/renderer?query=nainen+mies+auto&use=iltalehti-testi&login=user&key="+key+"&name=default' "+
            "scrolling='auto' width='99%'><p>Selaimesi ei tue iframe -elementtejä.</p></iframe>");
    }, 3600000); // interval
} else {
    alert("Virheellinen käyttäjätunnus");
}

} // success

}); //XML -ajax

```

## KUVA 25. Lisää kirjautumisen jQuery-koodia

Uloskirjautuessa ja päivitystä vaihtaessa hyödynnetään delegate-funktiota dynaamisesti luodulle uloskirjautumisnapille, jotta ajastinfunktio saadaan nollattua. Uloskirjautumisskripti alustaa uloskirjautumis- sekä data-elementin ja ilmoittaa käyttäjälle uloskirjautumisesta. Käyttöliittymän etusivu ladataan uudestaan ja varmuudenvuoksi kirjautumisen tunnistautumistiedot alustetaan kertaalleen (kuva 26).

```

    } else { // null tarkistus

        alert("Syötä kirjautumistiedot");

    } // null tarkistus

}); // kirjautuminen

$("#box2").delegate("#nappi02", "click", function() {

    clearInterval();

    login = "";
    key = "";

    $("#box2").empty();
    $("#data").empty();
    alert("Sulje selain varmistaaksesi uloskirjautumisen");
    location.reload();
    $("#login").val("");
    $("#key").val("");

}); // uloskirjautuminen

$("#box2").delegate("#nappi03", "click", function() {

    clearInterval();
    $("#nappi03").remove();
    $("#box2").append("<input type='button' id='nappi04' value='Päivitä tunnin välein'/>");

}); // päivitys

```

## KUVA 26. Uloskirjautumisen ja päivityksen jQuery-koodia

Päivitystä laittaessa päälle tai pois sitä varten luodaan ja poistetaan eri painikkeita, jotka käyttävät sitten skriptin alkuosan tapaa tuoda sisältö data-elementtiin (kuva 27).

```

$("#box2").delegate("#nappi04", "click", function() {

    setInterval(function() {

        $("#data").empty();
        $("#data").append("<iframe id='iframe' "+
            "src='http://localhost:9090/renderer?query=nainen+mies+auto&use=iltalehti-testi&login=user&key="+key+"&name=default' "+
            "scrolling='auto' width='99%'><p>Selaimesi ei tue iframe -elementtejä.</p></iframe>");

    }, 3600000); // interval
    $("#nappi04").remove();
    $("#box2").append("<input type='button' id='nappi03' value='Ota päivitys pois käytöstä'/>");

}); // päivitys

}); // document ready

```

## KUVA 27. Päivityksen loput jQuery-koodit

Käyttäjien tarkista varten on yksinkertainen XML-tiedosto, josta löytyy käyttäjäkohtaiset nimi- sekä hakusanatietueet (kuva 28). XML-tiedoston muokkaukselle luodaan myös erillinen ylläpitäjän näkymä, josta tietoja voi poistaa, lisätä tai muokata näin halutessaan.

```
<?xml version="1.0" encoding="utf-8" ?>
<kayttajat>
  <kayttaja xml:id="kayttaja_14116330665982">
    <nimi>kayttaja</nimi>
    <hakusanat>mies nainen auto</hakusanat>
  </kayttaja>
  <kayttaja xml:id="kayttaja_14116330712509">
    <nimi>autokayttaja</nimi>
    <hakusanat>auto</hakusanat>
  </kayttaja>
  <kayttaja xml:id="kayttaja_14116330788053">
    <nimi>mieskayttaja</nimi>
    <hakusanat>mies</hakusanat>
  </kayttaja>
  <kayttaja xml:id="kayttaja_14116330872456">
    <nimi>naiskayttaja</nimi>
    <hakusanat>nainen</hakusanat>
  </kayttaja>
</kayttajat>
```

## KUVA 28. XML-tiedosto

Palvelun määrittelyn toisen sekä kolmannen vaiheen vaatimukset on tässä vaiheessa siis toteutettu, eli automaattinen tietojen haku toimii, eikä hakusanoja aseteta enää manuaalisesti. Myös sisäänkirjautuminen käyttäjälle tietojen näyttämiseksi toimii, vaikka palvelussa olisi vielä toki kehitettävää.

### 4.2 Ylläpitäjän näkymä

Ylläpitäjän näkymässä listataan kaikkien käyttäjien nimet sekä hakusanat taulukkomaisesti. Näille käyttäjille annetaan myös muokkaus- sekä poistomahdollisuudet ja uuden käyttäjän lisääminen onnistuu taulukon yllä sijaitsevasta linkistä (kuva 29). Tämä aukeaa myöskin iframe-elementin kautta.



**KUVA 29. Ylläpitäjän näkymä**

Ylläpitäjän XML-muokkausskripti toteutettiin PHP:llä. PHP-tiedoston alku käsittelee peruuta-painikkeiden toimintoa, sekä uuden käyttäjätietueen lisäystä XML-tiedostoon (kuva 30). Skriptin toiminta perustuu hidden-kentissä kuljetettuihin muuttujatietoihin, joita ovat käyttäjä ID sekä "do"-parametri. Tällä parametrillä välitetään mitä toimintoa PHP-skripti suorittaa, näitä voi olla lisääminen "add", poistaminen "del" sekä päivittäminen "upd". Kaksi muuta parametria ovat "cancel" ja "go", jotka nimensä mukaisesti, joko peruuttavat tai suorittavat toimintoja loppuun.

```
<?php

if (isset($_REQUEST["cancel"])) {
    header("Location: kayttajahallinta.php");
}

$tiedostonimi = "
http://localhost:9090/kayttoliittymatesti/public_html/kayttajat.xml";

if (isset($_REQUEST["go"])) {
    if ($_REQUEST["do"] == "add") {
        if ($_REQUEST["nimi"]) {
            $nimi = $_REQUEST["nimi"];
        } else {
            $nimi = "(nimenä käyttäjä)";
        }

        if ($_REQUEST["hakusanat"]) {
            $hakusanat = $_REQUEST["hakusanat"];
        } else {
            $hakusanat = "";
        }

        $dom = new DOMDocument("1.0");

        $dom->encoding = "utf-8";

        $dom->preserveWhiteSpace = false;
```

**KUVA 30. XML-tietueen lisäämistä**

Käyttäjätietueiden yksilöimiseksi niille annetaan käyttäjä ID, joka määritetään käyttämällä hyväksi eräänlaista timestamp -funktia. Tähän lisätään sitten vielä

satunnainen luku perään, jotta samanlaista ID:tä ei varmasti esiinny (kuva 31). Käyttäjää päivittäessä niiden tiedot haetaan juurikin tätä ID-attribuuttia käyttäen.

```
$dom->formatOutput = true;

$dom->load($tiedostonimi);

$juuri = $dom->getElementsByTagName("kayttaja")->item(0);

$kayttajaelementti = new DOMElement("kayttaja");

$juuri->appendChild($kayttajaelementti);

$kayttajaelementti->appendChild(new DOMElement("nimi", $nimi));

$kayttajaelementti->appendChild(new DOMElement("hakusanat", $hakusanat
));

$uusi_kayttajaid = "kayttaja_".time().rand(1000, 9999);

$kayttajaelementti->setAttribute("xml:id", $uusi_kayttajaid);

$dom->save($tiedostonimi);
}

if ($_REQUEST["do"] == "upd") {

    $dom = new DOMDocument("1.0");

    $dom->load($tiedostonimi);

    $kayttaja_id = $_REQUEST["kayttaja_id"];
```

### KUVA 31. XML-tietueen lisäämistä ja päivittämistä

Päivittäessä käyttäjätietue korvataan uudella tietueella, jolle ylläpitäjä on määrittänyt nimen sekä hakusanat. PHP-skriptissä nähtyä kaavaa mukaillen voidaan toteuttaa myös tietueiden poistaminen (kuva 32). Käyttäjätiedot haetaan tässäkin tapauksessa käyttäjän ID:n perusteella.

```

    $nimi = $_REQUEST["nimi"];

    $hakusanat = $_REQUEST["hakusanat"];

    $juuri = $dom->getElementsByTagName("kayttaja")->item(0);

    $kayttaja = $dom->getElementById($kayttaja_id);

    $kayttajaelementti = new DOMElement("kayttaja");

    $juuri->replaceChild($kayttajaelementti, $kayttaja);

    $kayttajaelementti->appendChild(new DOMElement("nimi", $nimi));

    $kayttajaelementti->appendChild(new DOMElement("hakusanat", $hakusanat));

    $kayttajaelementti->setAttribute("xml:id", $kayttaja_id);

    $dom->save($tiedostonimi);
}

if ($_REQUEST["do"] == "del") {

    $dom = new DOMDocument("1.0");

    $dom->load($tiedostonimi);

    $kayttaja_id = $_REQUEST["kayttaja_id"];

    $kayttaja = $dom->getElementById($kayttaja_id);

```

**KUVA 32. XML-tietueen päivittämistä sekä poistamista**

Ohjelmakoodissa poistaessa tietueita XML-tiedosto tallennetaan erilleen alkuperäisestä (kuva 33). Tämä johtuu koodia työstäessä suoritettua testauksesta, jolloin en halunnut tietueiden häviävän käsiteltävästä tiedostosta. Muuttamalla tallenuspolun takaisin ”\$tiedostonimi”-muuttujaksi muutokset tapahtuvat jälleen alkuperäiseen tiedostoon.



```

$juuri = $dom->getElementsByTagName("kayttajat")->item(0);

$juuri->removeChild($kayttaja);

$dom->save("
http://localhost:9090/kayttoliittymatesti/public_html/kayttajat2.xml");

}

header("Location: kayttajahallinta.php");
} else {

$dom = new DOMDocument("1.0");

$dom->load($tiedostonimi);

$kayttajat = $dom->getElementsByTagName("kayttaja");

}

?>

<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <link rel="stylesheet" type="text/css" href="tyylit.css">
    <title>Käyttäjärekisteri</title>
  </head>
  <body>

```

**KUVA 33. XML-tiedoston poistamista sekä HTML-pohjaa**

Uuden käyttäjän lisäykselle on oma näkymänsä, jossa on nimelle ja hakusanoille omat syöttölaatikkonsa (kuva 34). Tämän lomakkeen mukana kulkee piilotettuna hidden-laatikossa "add"-parametri, jotta Tallenna-painiketta klikatessa ohjelma osaa siirtyä oikeaan kohtaan.

```

<div id="container">

<?php
if (isset($_REQUEST["do"])) {
?>

    <?php if ($_REQUEST["do"] == "add") { ?>

        <h2>Lisää uusi käyttäjä</h2>

        <form method="get" action="kayttajahallinta.php">

            <fieldset>
                <legend>Käyttäjätiedot</legend>

                <label for="nimi">Käyttäjänimi:</label>
                <input type="text" id="nimi" name="nimi" value="" size="20"
                ><br/>
                <label for="hakusanat">Hakusanat:</label>
                <input type="text" id="hakusanat" name="hakusanat" value=""
                size="20">
                <input type="hidden" name="do" value="add">
            </fieldset>
            <input type="submit" name="go" value="Tallenna">
            <input type="submit" name="cancel" value="Peruuta">

        </form>

    <?php }

```

**KUVA 34. Käyttäjälisäyksen form**

Käyttäjän muokkaaminen on ulkonäöllisesti periaatteessa sama, kuin käyttäjänlisäys. Käyttäjän tiedot haetaan ensin tiedostosta valmiiksi ID:tä käyttäen. Ainut ero on, että syöttötietoihin on lisätty placeholderiksi käyttäjän nykyiset tiedot. Näin voidaan silmäyksellä nähdä mitä tietoja tältä käyttäjältä ollaan muokkaamassa (kuva 35).

```

if ($_REQUEST["do"] == "update") {

    $dom = new DOMDocument("1.0");

    $dom->load($tiedostonimi);

    $kayttaja_id = $_REQUEST["kayttaja_id"];

    $kayttaja = $dom->getElementById($kayttaja_id);

    $nimi = $kayttaja->getElementsByTagName("nimi")->item(0)->
    textContent;

    $hakusanat = $kayttaja->getElementsByTagName("hakusanat")->item(
    0)->textContent;

    ?>

    <h2>Muokkaa käyttäjän tietoja</h2>

    <form method="get" action="kayttajahallinta.php">

        <fieldset>
        <legend>Käyttäjätiedot</legend>

        <label for="nimi">Käyttäjän nimi:</label>
        <input type="text" id="nimi" name="nimi" placeholder="<?php
        echo $nimi; ?>" size="20"><br/>
        <label for="hakusanat">Hakusanat:</label>
        <input type="text" id="hakusanat" name="hakusanat"
        placeholder="<?php echo $hakusanat; ?>" size="20">

```

### KUVA 35. Käyttäjämuokkauksen form

Käyttäjää poistaessa ohjelma näyttää käyttäjätietueen nimen ja kysyy varmistuksen poistamiselle. Tämä estää vahingossa tapahtuneet poistonapin klikkaukset (kuva 36). Tiedostosta haetaan käyttäjä ID:tä vastaava tietue valmiiksi poistotoiminnalle.

```

<input type="hidden" name="kayttaja_id" value="<?php echo
$kayttaja_id; ?>">
<input type="hidden" name="do" value="upd">
</fieldset>
<input type="submit" name="go" value="Tallenna">
<input type="submit" name="cancel" value="Peruuta">

</form>

<?php }

if ($_REQUEST["do"] == "del") {

    $dom = new DOMDocument("1.0");

    $dom->load($tiedostonimi);

    $kayttaja_id = $_REQUEST["kayttaja_id"];

    $kayttaja = $dom->getElementById($kayttaja_id);

    ?>

<h2>Poista käyttäjän tiedot</h2>

<form method="get" action="kayttajahallinta.php">

    <p>
        Haluatko varmasti poistaa käyttäjän: <?php echo $_REQUEST[
        "nimi"] ?>?

```

**KUVA 36. Käyttäjän muokkausta sekä käyttäjän poistoa**

Alkunäkymän taulukkorakenne alkaa neljällä header -solulla. Näistä kaksi käytetään ostikoiksi käyttäjien tiedoille ja kaksi on tyhjiä (kuva 37). Tyhjien solujen alle tulee tietoja taulukkoon haettaessa muokaus- sekä poistopainikkeet. Taulukkorakennetta muodostaessa sen yläpuolelle lisätään samalla linkki uuden käyttäjän luomiselle.

```

        </p><br/>

        <input type="hidden" name="kayttaja_id" value="<?php echo
        $_REQUEST["kayttaja_id"] ?>">
        <input type="hidden" name="do" value="del">
        <input type="submit" name="go" value="Poista">
        <input type="submit" name="cancel" value="Peruuta">

    </form>

    <?php }

} else {
?>

<a href="kayttajahallinta.php?do=add">Lisää uusi käyttäjä</a>

<table>
    <tr>
        <th>Käyttäjän nimi</th>
        <th>Hakusanat</th>
        <th>&nbsp;</th>
        <th>&nbsp;</th>
    </tr>

    <?php

    foreach ($kayttajat as $kayttaja) {

        $nimi = $kayttaja->getElementsByTagName("nimi")->item(0)->
        textContent;
    }

```

### KUVA 37. Alkunäkymän muodostusta

Käyttäjätiedot käydään läpi foreach -silmukassa, jolloin ne voidaan sijoittaa taulukkosoluihin. Käyttäjätietueille luodaan yksilölliset muokkaus- sekä poistopainikkeet, jotka toimivat käyttäjä ID:n kautta (kuva 38). Näin ohjelma tietää aina mille käyttäjälle ylläpitäjä on halunnut tehdä muutoksia, tai vastaavasti minkä käyttäjän tämä on halunnut poistaa.

```

        $hakusanat = $kayttaja->getElementsByTagName("hakusanat")->item(
0)->textContent;

        $kayttaja_id = $kayttaja->getAttribute("xml:id");

    ?>

    <tr>
        <td><?php echo $nimi;?></td>
        <td><?php echo $hakusanat;?></td>
        <td><a href="kayttajahallinta.php?do=upd&kayttaja_id=<?php echo
        $kayttaja_id; ?>">Muokkaa</a></td>
        <td><a href="kayttajahallinta.php?do=del&kayttaja_id=<?php echo
        $kayttaja_id; ?>&nimi=<?php echo $nimi; ?>">Poista</a></td>
    </tr>

    <?php
} //foreach
?>

</table>
<?php
} //if
?>

</div>

</body>

```

### KUVA 38. Käyttäjätietojen taulukkorakenne

Tämä ylläpitäjän PHP-skripti on yleispätevä XML-formaatissa esitettyjen käyttäjätietojen muokkaus- ja päivitystyökalu. Sen saa helposti muokattua erilaiseksi niin ulkoasullisesti, että toiminnallisesti, tarpeen näin vaatiessa. Sellaisenaankin sillä voisi hallinnoida tarpeeksi kattavasti vastaavanlaisessa formaatissa tallennettuja käyttäjätietoja.

## 5 PÄÄTÄNTÖ

Opinnäytetyö täytti sille asettamani henkilökohtaiset tavoitteet. Sain kehitettyä OpenSearchServeriä hyödyntäen verkkopalvelun, joka on kaikille mahdollisille käyttäjille helppo ja nopea ottaa käyttöön. Palvelulle suuntautuva kaikki tuleva jatkokehitys ja käyttöönotto käy vaivattomasti sen joustavuuden takia. Myös toimeksiantajan määrittelemät tavoitteet on saavutettu. Myös tavoitteiden lisäksi tein pienenä ylimääräisenä loppusilauksena ylläpitäjälle oman näkymän. Valmiin palvelun käyttötarkoituksena olisi toimeksiantajan yritysasiakkaiden pystyä määritellyiltä

sivustoilta etsimään yrityksestään ja esimerkiksi tuotteistaan tietoja vaikka bränditutkimuksen merkeissä. Jatkokehitettävää palveluun on ainakin sen kokonaisvaltainen visuaalisen ulkoasun parantelu, sekä ylläpitäjän käyttöliittymän parannus. Tällä hetkellä ylläpitäjän näkymään kirjautuessa ei tarvitse salasanaa, vaan kirjautumiseen riittää ylläpitäjän tunnuksen tietäminen. Sellaisenaan PHP-skripti toimii myös erillisellä palvelimella. Työtä oli mukava tehdä ja siitä saisi hyvin laajankin kokonaisuuden luotua ja kirjoitettua, mikäli aikaa olisi enemmän.

Tämä kyseinen aihe ei ollut minulle lainkaan entuudestaan tuttu, mutta otin sen heti vastaan mielenkiinnolla sekä innostuneella asenteella. Minusta tuntui, että tämä on hyvinkin kiinnostavan ja työelämälähtöisen työn alku. Tyydyin opinnäytetyöni aiheen rajauksessa aluksi täyttämään työn toimeksiantajan määrittelemät tavoitteet pysyäkseni aikataulussa, mutta lisäsin siihen myös ylläpitäjän näkymän lopuksi. Alussa työ vaikutti hirveän isolta, vaikkakin sitäkin kiinnostavammalta. En oikein tiennyt mistä voisin aloittaa, mutta tiesin, että täysin tyhjästä palvelun ohjelmoiminen ei ainakaan olisi kannattavaa. Kokeilin silti yksinkertaisen web crawler mekaniikan testausta PHP-pohjaisella web servicellä, joka haki haluamiltani sivustoilta kuvatiedostoja. Tämän jälkeen avoimen lähdekoodin web crawler ohjelmistoihin tutustuessani sain selvän kuvan mielessäni, että aloittaisin alustavalintani mukaan sen vaatimista perusasioista palvelun kokoamista työstäessäni siitä valmista ratkaisua.

Muutamia ongelmia joihin kohtasin työtä tehdessä oli millä tekniikoilla sen tekemistä lähestyisin ja miten näitä hyödyntäisin. OSS:n valinta palvelun pohjaksi auttoi tässä huomattavasti, tarjotessaan valmiin palvelinympäristön jossa testata palvelua. OSS:llä on oma RESTful JSON-rajapinta jonka pohjalta aloin ensiksi työtäni tekemään. Aluksi tämän kanssa oli ongelmia saada erillinen palvelu kommunikoimaan OSS:n palvelimen kanssa, mutta se ratkesi siirtämällä palvelu samalle palvelimelle, jolla OSS toimii. Rajapinnan avulla työn toteutus oli hitaampaa ja työläämpää. Työpäivät kuluivat lähinnä erilaisiin kokeiluihin, joiden kautta havaitsin tarkemmin mitkä tavat eivät toimi palvelun toteuttamisessa. Päädyinkin lopulta hyödyntämään OpenSearchServerin rendererer -toimintoa iframe elementin kautta hyväkseni. Tällä tavalla tunnistautumisen ympärille jää myös tarpeeksi joustavuutta erilaisten tarkistusten ja parannusten kannalta. Ylläpitäjän näkymää ohjelmoidessani törmäsin myös ongelmaan PHP:n kannalta, sillä TomCat-palvelin ei voinut ajaa sitä.

Päädyinkin ratkaisemaan tämän ajamalla PHP-skriptin erilleen WAMP-palvelimen kautta, sillä se oli sillä hetkellä nopein ratkaisu toteuttaa.

OpenSearchServer tarjoaa hyvin runsaasti mahdollisuuksia tiedonlouhinnan ja data-analyysin parissa, ja työni on aikalailla pintaraapaisua sen maailmaan. Kaiken kaikkiaan olen tyytyväinen opinnäytetyön tarjoamaan tilaisuuteen oppia web crawlereiden maailmasta ja sitä ympäröivästä bisneksestä. Myös SCRUM tyylinen työelämälähtöinen työskentely oli mukavaa vaihtelua. Selvät tavoitteet ja deadlinet motivoivat saamaan työtä kokoajan eteenpäin tasaisella tahdilla. Tutkimustyö ja kaikki tekemäni kokeilu eivät valitettavasti näy lopputuloksen niin kutsutussa hienoudessa, vaan lähinnä siinä, että pystyin käyttämään hyväkseni tekniikoiden tarjoamia yksinkertaisia lähestymistapoja.



## LÄHTEET

Bruce, James 2010. How To Build A Basic Web Crawler To Pull Information From A Website. WWW-dokumentti. <http://www.makeuseof.com/tag/build-basic-web-crawler-pull-information-website/>. Päivitetty 10.12.2010. Luettu 14.3.2016.

Fedorkov, Vlad 2011. PHP Crawler. WWW-dokumentti. <http://astellar.com/php-crawler/>. Päivitetty 13.12.2011. Luettu 23.3.2016.

Franklin, Kurt 2000. How Internet Search Engines Work. WWW-dokumentti. <http://computer.howstuffworks.com/internet/basics/search-engine.htm>. Päivitetty: 27.9.2000. Luettu 14.3.2016.

McKinsey Global Institute 2011. Big data: The next frontier for innovation, competition, and productivity. WWW-dokumentti. <http://www.mckinsey.com/business-functions/business-technology/our-insights/big-data-the-next-frontier-for-innovation>. Ei päivitystietoa. Luettu 13.4.2016.

Mirtaheri, Seyd, Dinçtürk, Mustafa, Salman, Hooshmand, Bochmann, Gregor, Guy-Vincent, Jourdan & Onut, Iosif 2014. A Brief History of Web Crawlers. PDF-dokumentti. <http://arxiv.org/pdf/1405.0749v1.pdf>. Ei päivitystietoa. Luettu 14.3.2016.

Norconex 2013. Getting Started. WWW-dokumentti. <http://www.norconex.com/collectors/collector-http/getting-started>. Ei päivitystietoa. Luettu 23.3.2016.

NT, Baiju 2015. Top 50 open source web crawlers for data mining. WWW-dokumentti. <http://bigdata-madesimple.com/top-50-open-source-web-crawlers-for-data-mining/> Päivitetty 23.1.2015. Luettu 14.3.2016.

Olston, Christopher & Najork, Marc 2010. Web Crawling. Massachusetts: now Publishers Inc.

OpenSearchServer 2014. Discovering the main concepts. WWW-dokumentti. <http://www.opensearchserver.com/documentation/tutorials/functionalities.md>. Ei päivitystietoa. Luettu 23.3.2016.

Scrapy 2016. Companies that are using Scrapy. WWW-dokumentti. <http://scrapy.org/companies/>. Ei päivitystietoa. Luettu 23.3.2016.