POHJOIS-KARJALAN AMMATTIKORKEAKOULU
Viestinnän koulutusohjelma

Panu Karkiainen

AVAINSANAT JOOMLA!-SISÄLLÖNHALLINTAJÄRJESTELMÄSSÄ

Opinnäytetyö
Lokakuu 2011

Sisällys

POHJOIS-KARJALAN
AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Lokakuu 2011
Viestinnän koulutusohjelma

Länsikatu 15
80200 JOENSUU
p. (013) 260 6862 p. (013) 260 6906

Tekijä(t)
Panu Karkiainen

Nimeke
Avainsanat Joomla!-sisällönhallintajärjestelmässä

Toimeksiantaja

Tiivistelmä
Avainsanat ovat sanoja, joita liitetään erityyppisiin sisältöihin kuvaamaan kyseistä sisältöä. Avainsanat ovat tärkeä työkalu organisoitaessa sisältöjä verkkosivuilla. Ne tarjoavat mahdollisuuden luoda asiayhteyksiä eri sisältöjen välille ja liittää sisältöjä suurempiin kokonaisuuksiin. Ne soveltuvat erityisen hyvin ei-tekstuaalisten sisältöjen, kuten kuvien tai videoiden, organisointiin. Tehokas avainsanojen käyttö edellyttää kuitenkin suunnittelua ja kompromisseja.

Sisällön määrän kasvaessa verkkosivulla, tulevat erilaiset sisällönhallintajärjestelmät tarpeeseen. Sisällönhallintajärjestelmä mahdollistaa useiden eri käyttäjien tuottaa ja hallita sisältöjä. Ne tarjoavat valmiita työkaluja sisältöjen esittämiseen sekä muokkaamiseen ja organisointiin. Erityisesti avoimeen lähdekoodiin perustuvat sisällönhallintajärjestelmät, kuten Joomla!, ovat saaneet suosioita. Suosio perustuu niiden ilmaisuuteen sekä laajennettavuuteen ja helppokäytöisyyteen.

Tässä opinnäytetyössä keskitytään avainsanojen käyttöön Joomla!-sisällönhallintajärjestelmässä. Opinnäytetyössä käydään läpi aiheeseen liittyviä ongelmia sekä olemassa olevia ratkaisuja. Opinnäytetyössä esitetään yleiskäyttöinen malli, jolla avainsanoja voidaan liittää useisiin erityyppisiin sisältöihin. Tämä mallin toimintaperiaatteiden lisäksi kuvataan mallin kehitysprosessia toimivaksi ohjelmistolaajennukseksi, joka voidaan ottaa käyttöön Joomla!-sisällönhallintajärjestelässä. Kehitysprosessin kuvauksessa käydään läpi ongelmia, niihin löytyneitä ratkaisuja sekä tulevaisuuden suunnitelmia.

Kieli
suomi

Sivuja 57

Asiasanat
www-sivustot, avoin lähdekoodi, sisällönhallinta

NORTH KARELIA
UNIVERSITY OF APPLIED SCIENCES

THESIS
October 2011
Degree Programme in Communication
Länsikatu 15
FIN 80200 JOENSUU
FINLAND
Tel. 358-13-260 6862

Author(s)
Panu Karkiainen

Title
Avainsanat Joomla!-sisällönhallintajärjestelmässä

Commissioned by

Abstract

Keywords are words that are used to describe the content they are attached to. Keywords are an important tool when managing content on a web page. They can be used to create associations between pieces of content and they also can be used to associate content to more abstract concepts. They are especially useful when managing non-textual content suchs as images or videos. Still, their proper use requires planning and compromises.

The need for a content management system rises from the increasing amount of content that web pages contain. Content management systems allows multiple users to simultaneously create and manage content. Open source content management systems such as Joomla! have increased their popularity lately. This is largely because of their free use, extendability and ease of use.

This thesis focuses on the use of keyword on Joomla!. Problems that may arise are discussed and existing solutions are studied. In addition this thesis introduces a generic model for attaching keywords to content regardless of the type of the content. The theory of this model is described and a process in wich the theory is developed to an working software extension that can be installed to Joomal! content management system.

Language
Finnish

Pages 57

Keywords
www-site, open source, content management system

1 Johdanto

2010-luvulla avainsanojen käytöstä verkkosivun sisällön organisoinnissa on tullut yleinen käytäntö. Aikaisemmin avainsanoja oli käytetty lähinnä massiivisten tietomäärien hallintaan. Esimerkiksi YouTube-videopalvelussa avainsanat ovat olleet korvaamattoman tärkeä osa massiivisen yli sadan miljoonan videon kokoelman organisoinnissa. Nyt avainsanoja halutaan käyttää myös pienempien sisältömäärien hallintaan.

Samaan aikaan ovat avoimen lähdekoodin sisällönhallintajärjestelmät tulleet osaksi yhä useamman verkkosivun teknistä toteutusta. Nämä järjestelmät ovat ilmaisia käyttää, ne kehittyvät koko ajan ja niiden ympärillä toimii aktiivinen käyttäjäyhteisö. Järjestelmät helpottavat sisällöntuottamista verkkosivuille eikä jokaisen sisällöntuottajan tarvitse tietää verkkosivujen toteuttamiseen liittyviä teknisiä yksityiskohtia.

Avoimen lähdekoodin projekteissa käyttäjillä on mahdollisuus osallistua ohjelmiston kehitystyöhän omalla panoksellaan. Sisällönhallintajärjestelmissä tämä usein tarkoittaa itsenäisten laajennusten toteuttamista. Nämä laajennukset tarjoavat käyttäjille toiminnallisuuksia, joita järjestelmän perusversiosta puuttuu. Käyttäjäyhteisön kehittämät laajennukset ovat nykyään niin tärkeä osa sisällönhallintajärjestelmiä, etteivät järjestelmät edes yritä tarjota kaikkea toiminnallisuutta perusversiossaan.

Aktiivinen käyttäjäyhteisö on projektille mittava voimavara, mutta kehittäjien moninainen joukko aiheuttaa myös ongelmia. Koska kaikki sisällönhallintajärjestelmän laajennukset ovat omia itsenäisiä projektejaan, ei niiden toimintalogiikka aina noudata samaa kaavaa järjestelmän perusversion kanssa. Tämä ongelma tulee esiin avainsanojen käytön kohdalla. Avainsanojen liittäminen sisällönhallintajärjestelmän tekstisisältöihin onnistuu melko yksinkertaisesti. Jos sivustolle halutaan lisätä myös videoita, on syytä asentaa järjestelmään erillinen videoiden hallintaan erikoistunut laajennus. Tämä videosisältöjen hallintaan käytetty laajennus vaikeuttaa avainsanojen liittämistä sisältöihin. Sama tekniikka, jota käytettiin avainsanojen liittämiseen tekstisisältöihin ei osaakaan liittää avainsanoja videoihin, koska kyseinen laajennus ei ole osa järjestelmän perusversiota.

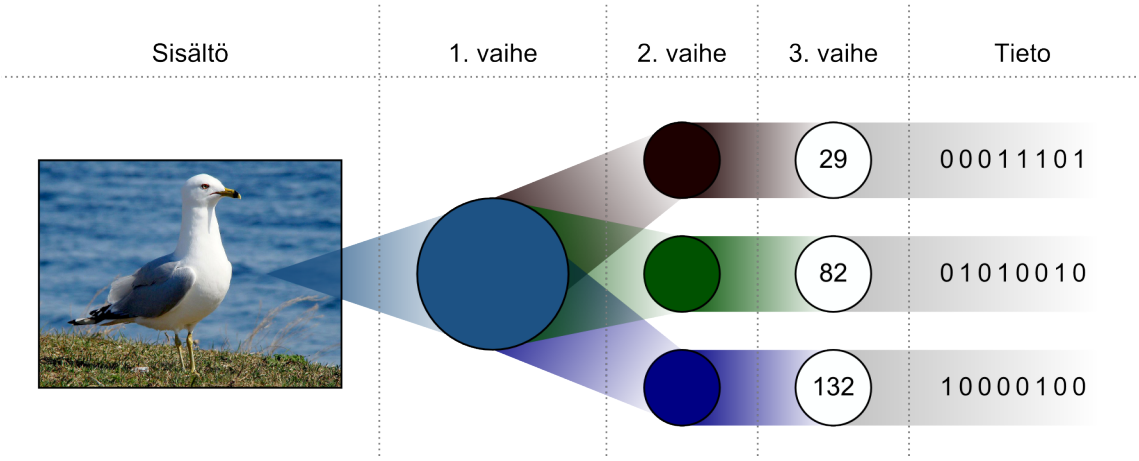
Ideaalitilanteessa avainsanoja pystyisi liittämään mihin tahansa sivustolla olevaan sisältöön riippumatta siitä, hallitaanko kyseistä sisältöä järjestelmän perusversion tarjoamilla työkaluilla vai jonkin erillisen laajennuksen kautta. Käytännössä asia ei kuitenkaan onnistu näin helposti ja tästä muodostuu ongelma. Tässä opinnäytetyössä perehdyn tähän ongelmaan Joomla-sisällönhallintajärjestelmässä. Käyn läpi syitä, jotka aiheuttavat tämän ongelman sekä esittelen oman ratkaisumallin. Kirjallisen osuuden lisäksi olen kehittänyt esittelemääni ratkaisumalliin perustuvan käytännön sovelluksen, jonka kehitystyöstä kerron luvussa neljä.

2 Tietoperusta

2.1 Avainsanat

2.1.1 Metatieto

Digitaalinen kuva on tietokoneelle tietoa. Se on tietoa pikseleistä, joista kuva muodostuu. Jokainen pikseli esitetään väriarvona. Väriarvo on taas tietoa siitä, kuinka kyseinen väri voidaan muodostaa pääväreistä. Varsinainen käsitys väristä syntyy vasta siinä vaiheessa, kun joku ihminen näkee sen. Samalla tavalla käsitys kuvan sisällöstä muodostuu vasta ihmisen mielessä. Ihmiselle kuva koostuu henkilöistä, esineistä, paikoista ja asioista. Kuva liittyy johonkin asiayhteyteen, mikä antaa sille merkityksen. Lisäksi kuva voi herättää näkijässään tunteita. Tietokoneelle kuva on informaatiota kun taas ihmiselle käsitys kuvasta on sisältöä. Kuvassa 1 on jaettu vaiheisiin sisällön muuntuminen tiedoksi. Ensimmäisessä vaiheessa kuva jaetaan pikseleihin, joista jokainen edustaa yhtä väriarvoa. Toisessa vaiheessa pikselin väriarvo jaetaan värikomponentteihin. Kolmannessa vaiheessa jokainen väriarvo esitetään numeerisesti. Numeerinen tieto voidaan sitten esittää binäärimuotoisena tietona, jota tietokone käsittelee.



Kuva 1. Sisältö ja tieto.

Ongelma muodostuu, kun haluamme tietokoneen etsivän tuhansien kuvien joukosta kuvan linnusta. Jos pystyisimme kertomaan tietokoneelle tarkalleen, mistä pikseleistä linnun kuva muodostuu, ei ongelmaa olisi. Lintulajeja on kuitenkin valtava määrä ja ne ovat kaikki eri näköisiä. Jokainen saman lajin edustajakin on ulkonäöltään yksilöllinen. Se miltä lintu lopulta näyttää kuvassa riippuu ympäristöstä, valaistuksesta sekä kuvakulmasta. Toisin sanoen on mahdotonta määritellä tietokoneelle tarkalleen, miltä lintu näyttää.

Metatieto tarjoaa osittaisen ratkaisun tähän ongelmaan. Yleisesti metatiedolla tarkoitetaan tietoa tiedosta, mutta tämä määritelmä on liian abstrakti ollakseen käytännöllinen (Morville & Rosenfeld 2006, 194). Konkreettisempi esimerkki metatiedosta on kuvaan liitetty avainsana. Kun kuvaan linnusta liitetään avainsana “lintu”, voi tietokone ratkaista aikaisemmin esitetyn hakuongelman. Tietokoneelle avainsana on samanlaista tietoa kuin itse kuvakin, eikä se pysty ymmärtämään sanan “lintu” merkitystä. Käyttäjä voi kuitenkin pyytää tietokonetta etsimään kuvien joukosta sellaisen kuvan, johon on liitetty avainsana “lintu”.

Metatieto voi olla kaikkea sellaista tietoa, joka voidaan liittää sisältöön helpottamaan sisällön organisointia ja hakua (Boiko 2005, 491). Esimerkiksi sisällön alkuperäinen tekijä tai luontipäivä ovat metatietoa. Myös monet tekniset tiedot, kuten käytetty kuvaformaatti, ovat metatietoa.

2.1.2 Avainsanat tiedon organisoinnin välineenä

Tallennettaessa sisältöjä on tyypillistä käyttää sisältöjen organisointiin hierarkkista rakennetta. Hierarkkisessa rakenteessa jokainen sisältö kuuluu johonkin ryhmään. Ryhmät voivat olla sisäkkäisiä ja näin muodostaa osa-kokonaisuussuhteita. Tästä esimerkki on tiedostojärjestelmässä olevien tiedostojen organisointi hakemistoihin ja alihakemistoihin. Käytännön sovelluksissa hierarkkiset rakenteet ovat usein liian rajoittuneita kuvaamaan kunnolla sisältöjen todellisia keskinäisiä suhteita.

Esimerkiksi lintukuvista koostuvan sisältökokoelman voi helposti organisoida hierarkkisesti kuvissa näkyvien lintujen perusteella, koska lintulajisto itsessään muodostaa hierarkkisen rakenteen. Ongelma syntyy, kun samassa kuvassa näkyy useampi eri lajiin kuuluva lintu. Tällöin kuvan tulisi kuulua yhtä aikaa useaan eri ryhmään hierarkiassa. Hierarkkisessa rakenteessa sisältö voi kuulua kuitenkin vain yhteen ryhmään. Ongelman voisi ratkaista kopioimalla kuvan jokaiseen ryhmään, johon se kuuluu. Tällöin käyttäjän näkökulmasta kuva näyttäisi kuuluvan yhtä aikaa useaan eri ryhmään. Sisällönhallinnan kannalta on kuitenkin järkevää välttää useiden samasta sisällöstä otettujen kopioiden ylläpitämistä. Useiden samasta sisällöstä otettujen kopioiden ylläpitäminen tuhlaa tallennustilaa ja jos sisältöön tulee muutoksia, on muutostyöt tehtävä jokaiseen sisällöstä otettuun kopioon.

Todellinen ongelma esimerkissä on se, etteivät kuvat kokonaisuutena muodosta yhtä isoa hierarkiaa. Jos kuvat halutaan ryhmitellä sen perusteella, mitä lintulajeja kuvassa näkyy, on luovuttava ajatuksesta pakottaa ryhmät osa-kokonaisuussuhteisiin. Yksi ryhmä pitää sisällään kuvia yhdestä lintulajista. Yksi kuva voi samaan aikaan kuulua useaan eri ryhmään. Ryhmistä voi tarpeen mukaan muodostaa hierarkkisia osakokonaisuuksia.

Tutkittaessa lisää äsken mainittua tapaa ryhmitellä kuvia huomataan, että jokainen ryhmä määrittää sen, mitä yhteistä siihen kuuluvilla kuvilla on. Toisin sanoen ryhmän voidaan sanoa määrittelevän jonkin ominaisuuden siihen kuuluville sisällöille. Koska tämä on juuri sama asia, johon avainsanoja käytetään, voidaan kaikki organisointiin käytetyt ryhmät korvata avainsanoilla. Tällä tavalla avainsanat muodostavat tehokkaan ja jousta-

van työkalun sisällön organisointiin. Avainsanat eivät ole pelkästään vaihtoehto hierarkkiselle rakenteelle. Kaikki hierarkkiset rakenteet voidaan esittää myös avainsanoja käyttämällä. Toisin sanoen avainsanoja käyttämällä voidaan tehdä kaikki se mitä hierarkkisilla rakenteillakin ilman hierarkkisten rakenteiden asettamia rajoituksia.

2.1.3 Avainsanojen muodostamat sanastot

Tekninen mahdollisuus käyttää avainsanoja sisällön organisoinnissa ei kuitenkaan riitä. Tehokas avainsanojen käyttö vaatii suunnittelua, ja sisältöön liittyvien avainsanojen määrittäminen vaatii sisällön tulkitsemista. Kumpikin näistä työvaiheista edellyttää ymmärrystä ympäröivästä maailmasta ja sisällön asiayhteydestä. Toisin sanoen ne ovat tehtäviä, joita tietokoneen on mahdoton suorittaa kunnolla.

Käytettäessä ihmiskäyttäjiä määrittämään sisältöön liittyviä avainsanoja syntyy uusia ongelmia. Eri ihmiset voivat tulkita saman sisällön eri tavoilla. Joku näkee kuvassa linnun, toinen osaa tunnistaa linnun haarapääskyksi. Ihmiselle on selvää, että haarapääsky on lintu, mutta tietokoneelle ei. Tässä tilanteessa avainsanoille on määriteltävä osa-kokonaisuussuhde, jossa haarapääskyt kuuluvat lintuihin. Toinen vaihtoehto on liittää sisältöön sekä avainsana ”lintu” että avainsana ”haarapääsky”. Sisällön kuvaamiseen valitut avainsanat muodostavat sanaston.

Boiko (2005) esittää yleisiä ohjeita sisällönhallintajärjestelmän sisältöä kuvaavien sanastojen luomiseen. Ensimmäisenä hän ehdottaa, että avainsanoista koostuvan sanaston tulisi sisältää kaikki oleelliset asiaan liittyvät termit (Boiko 2005, 660). Tämä vaatimus on sitä helpompi toteuttaa mitä lähemmin organisoitavat sisällöt liittyvät toisiinsa. Esimerkiksi organisoitaessa kuvastoa Suomen linnuista tulisi avainsanoista koostuvan sanaston sisältää ainakin kaikki kuvissa esiintyvät lintulajit. Tällaisessa tapauksessa olisi hyvä, jos käytettyjen avainsanojen abstraktiotaso olisi yhtenäinen. Toisin sanoen jokaisessa kuvassa esiintyvät linnut olisi tunnistettu ja lajeja vastaavat avainsanat olisi liitetty kyseiseen kuvaan. Tämä ei ole edellytys avainsanojen käytölle, mutta yhtenäiset metatiedot helpottavat tiedon hakua. Jos kuvaston teema kasvaa koskemaan esimerkiksi

koko Suomen luontoa, on tämän ajatus koko termistön kattamisesta huomattavasti hankalampaa.

Avainsanojen tulee kuvata sisältöön liittyviä konsepteja riippumatta siitä, onko kyseinen konsepti mainittu sisällössä tai tuleeko se jollain muulla tavalla ilmi (Boiko 2005, 660). Tekstuaalisessa sisällössä tämä tarkoittaa sitä, ettei sisältöön liittyvän avainsanan tarvitse sellaisenaan esiintyä tekstissä. Monessa tapauksessa on oikeastaan parempikin, että avainsana tuo sisältöön jotain sellaista uutta tietoa tai liittää sen johonkin asiayhteyteen, joka ei käy ilmi varsinaisesta sisällöstä. Esimerkiksi kuvasta on monesti vaikea nähdä kuvauspaikkaa tai -ajankohtaa.

Boiko ehdottaa myös, että avainsanoista koostuvassa sanastossa tulisi ottaa huomioon yleisesti käytetyt synonyymit avainsanojen termeille (Boiko 2005, 660). Lintulajistossa esimerkiksi termi “pääskynen” on yleisnimi jollekin pääskylintuihin kuuluvalle linnulle. Tällöin haarapääskyn kuvaan voitaisiin liittää myös avainsana “pääskynen”. Mitä monimuotoisemmasta sisältökokoelmasta on kyse sitä hankalammaksi tulee toteuttaa tämä synonyymeihin liittyvä tavoite, varsinkin kun avainsanojen kohdalla termi synonyymi on hyvä nähdä kieliopillista määritelmää laajemmin. Käytännön sovelluksissa on usein tarve välttää esimerkiksi tilanteita, jossa tiedon löytyminen olisi kiinni pelkästään siitä, kirjoittiko käyttäjä avainsanan yksikössä vai monikossa. Tällaisessa tilanteessa avainsanan eri taivutusmuotojen voitaisiin katsoa olevan synonyymejä termin perusmuodolle, kuitenkin kohtuuden rajoissa.

2.1.4 Avainsanat tiedon haun välineenä

Hyvin toteutettu sisällön organisointi on itsessään jo tehokas tiedonhakuväline. Tästä huolimatta on tiedon haku varsinkin suurissa sisältökokonaisuuksissa välttämättömyys. Vaikka avainsanat ovat ennen kaikkea tiedon organisoinnin väline, on niistä merkittävää hyötyä myös esimerkiksi yksittäisten sisältöjen etsimisessä. Kuten aikaisemmin todettiin, ei-tekstuaalisen sisällön hakeminen vaatii aina jotakin metatietoa.

Tekstuaalisen sisällön kohdalla tilanne on helpompi. Jos haluamme etsiä lintua käsittelevän artikkelin tuhansien tekstiartikkelien joukosta, onnistuu tehtävä tietokoneelta melko hyvin. Erilaisia hakualgoritmeja tekstisisällön etsimiseen on kymmeniä. Jotkut algoritmit tuottavat useita aiheeseen enemmän tai vähemmän liittyviä osumia, toiset keskittyvät muutaman aiheeseen oleellisesti liittyvän osuman löytämiseen. (Morville & Rosenfeld 2006, 158.) Tietokone ei edelleenkään ymmärrä käytettyjen hakusanojen merkitystä tai sitä, minkälaisessa asiayhteydessä ne tekstissä ilmenevät. Siitä huolimatta tästä menetelmästä on merkittävää hyötyä varsinkin etsittäessä suurista sisältökokoelmista. Tämä ei tarkoita sitä, etteikö avainsanojen käytöstä voisi olla hyötyä myös haettaessa tekstuaalista sisältöä. Avainsanoja käyttämällä voidaan tekstiartikkelit liittää laajempiin asiayhteyksiin ilman, että kyseisiä konsepteja on erikseen mainittu tekstisisällössä. Esimerkiksi uutissivustolla uutinen jääkiekko-ottelun lopputuloksesta voidaan liittää yläkäsitteeseen “urheilu-uutiset”.

2.2 Verkkosivujen tekniikka

2.2.1 Verkkosivut ja -sivustot

Tässä opinnäytetyössä verkkosivustolla tarkoitetaan yksittäisistä verkkosivuista koostuvaa kokonaisuutta. Verkkosivusto voidaan monesti yksilöidä domain-nimen avulla. Verkkosivulla tarkoitetaan selaimessa näkyvää yksittäistä näkymää, joka voi koostua tekstistä, kuvista sekä muista mahdollisista sisällöistä kuten videoista. Monesti verkkosivu voidaan yksilöidä URL-osoitteen avulla. Teknisesti verkkosivu koostuu HTML-dokumentista sekä siihen liittyvistä, dokumentin ulkopuolisista, resursseista. Tällaisia resursseja voivat olla esimerkiksi kuvatiedostot, verkkosivun ulkoasun määrittelyyn käytetyt CSS-dokumentit sekä JavaScript-ohjelmat ja -kirjastot.

2.2.2 HTML-dokumentti ja selain

HTML-kieli on verkkosivujen kuvausta varten kehitetty kieli. Se on tekstuaalinen määritelmä sivuston sisällöstä sekä rakenteesta. Yksittäisen verkkosivun HTML-kielistä esi-

tystä kutsutaan HTML-dokumentiksi. CSS-kieli määrittelee joukon sääntöjä, joiden avulla HTML-dokumentin esitystä selaimessa voidaan hallita. Se on siis verkkosivujen ulkoasun määrittämiseen käytetty kieli. (W3C 2011.)

Selain on ohjelma, joka koostaa HTML-dokumentista sekä siihen liittyvistä resursseista käyttäjälle muodostuvan kokonaisuuden. Sivustot sijaitsevat yleensä erillisellä palvelinkoneella. Näin ollen selaimen tehtäviin kuuluu myös palvelinkoneen kanssa kommunikointi tietoliikenneprotokollia käyttäen. (W3C 2004.) Verkkosivujen selaaminen edellyttää jatkuvaa selaimen ja palvelimen välistä kommunikaatioita.

2.2.3 Asiakas ja palvelin

Internetissä selaimen ja palvelimen välinen kommunikaatio tapahtuu HTTP-tietoliikenneprotokolla käyttäen (The Internet Society 1999, 6-7). Tämä protokolla määrittää kommunikoinnin asiakkaan ja palvelimen välillä. Asiakkaan ei tässä protokollossa tarvitse olla selain. Se voi olla mikä tahansa sovellus, joka pystyy kommunikoimaan HTTP-protokollaa käyttäen. Protokolla koostuu pyynnöistä, joita asiakas lähettää palvelimelle, ja vastauksista, joita palvelin lähettää selaimelle vastauksena pyyntöön (The Internet Society 1999, 11). Esimerkiksi kun käyttäjä haluaa nähdä sivuston etusivun, lähettää selain palvelimelle pyynnön etusivun HTML-dokumentista. Palvelin käsittelee saapuneen pyynnön ja muodostaa pyyntöön vastauksen. Tässä esimerkissä vastaus muodostuisi etusivun HTML-dokumentista. Vastaus ei kuitenkaan sisällä HTML-dokumenttiin liittyviä resursseja, vaan jokainen noista ulkoisista resursseista on pyydettävä palvelimelta erillisille pyynnöllä. Pyynnöstä voidaan käyttää myös nimitystä sivupyyntö. Tämä termi on kuitenkin hieman harhaanjohtava, sillä palvelimelle on mahdollista tehdä pyyntöjä, joiden vastaus ei sisällä mitään sisältöä. Tällaisia pyyntöjä ovat muun muassa pyyntö tallentaa annetut tiedot palvelimelle. Vastaus pyyntöön voi olla myös niin kutsuttu uudelleenohjaus, jolloin palvelin kehottaa asiakasta lähettämään uuden pyynnön vastauksena annettuun URL-osoitteeseen.

2.2.4 Staattiset ja dynaamiset verkkosivut

Tavallisesti HTML-dokumentit on tallennettu palvelimille omiin tiedostoihinsa, jolloin palvelin lähettää selaimelle kyseisen tiedoston sisällön vastauksena sivupyyntöön. Tällaisista sivuista käytetään myös nimitystä staattinen sivu. Staattisessa sivussa palvelin lähettää vastauksena aina saman HTML-dokumentin niin kauan, kun alkuperäinen tiedosto palvelimella pysyy muuttumattomana. (Boiko 2005, 75.) Monissa tapauksissa tämä ei kuitenkaan riitä. Esimerkiksi hakukoneen hakutulossivu on aina riippuvainen hakusanasta, jonka käyttäjä syöttää hakukenttään. Olisi täysin kohtuutonta tallentaa kaikki mahdolliset eri hakutulokset omiin tiedostoihinsa palvelimelle. Dynaaminen sivu on sivu, jonka sisältö muodostetaan ohjelmallisesti (Boiko 2005, 75). Tällöin sivun sisältö voi vaihdella riippuen sivupyynnöstä tai jostain ulkoisesta tekijästä.

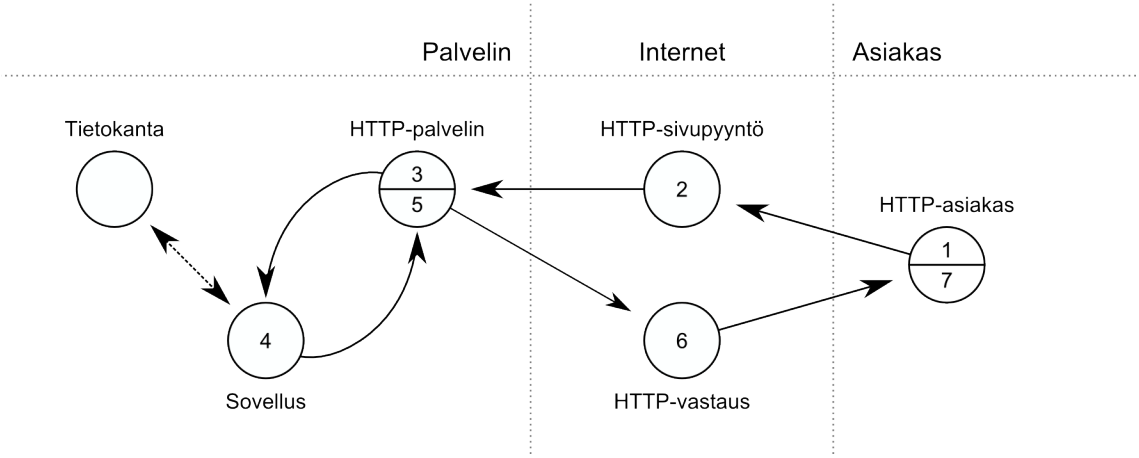
Hakukoneen hakutulossivu on esimerkki dynaamisesta sivusta, jonka sisältö on riippuvainen palvelimelle lähetetystä sivupyynnöstä. Sivupyynnössä palvelimelle tulee tieto hakusanoista, jotka käyttäjä on kirjoittanut hakukenttään. Sen sijaan, että palvelimella olisi valmiina tiedosto, joka sisältää lähetettävän vastauksen, suorittaa se erillisen sovelluksen, joka sitten muodostaa vastauksen selaimen lähettämään sivupyyntöön. Dynaaminen sivu tarkoittaa siis käytännössä palvelimella olevaa sovellusta, joka ottaa vastaan selaimelle tulleen sivupyynnön ja muodostaa esimerkiksi HTML-dokumentin. Näiden sovellusten tekeminen ei ole rajoittunut mihinkään tiettyyn ohjelmointikieleen tai ohjelmistoympäristöön. Dynaamisten sivujen tekniikka ei toki rajoitu pelkästään HTML-dokumentteihin, vaan mitä tahansa sisältöä jota palvelimelta pyydetään voidaan muodostaa dynaamisesti sovellusten avulla. Selaimen ei tarvitse tietää, millä tavalla sen vastauksena saama HTML-dokumentti tai vaikka kuvatiedosto on todellisuudessa muodostunut.

2.2.5 LAMP-ohjelmistoympäristö

Vaikka dynaamisten verkkosivujen tekeminen ei ole rajoittunut mihinkään tiettyyn ohjelmointikieleen tai ympäristöön, on varsinkin avoimen lähdekoodin ohjelmistoissa noussut suosioon niin kutsuttu LAMP-ohjelmistoympäristö. Tällä tarkoitetaan Linux-

pohjaista käyttöjärjestelmää, jossa käytössä on Apache HTTP -palvelinsovellus. Dynaamisten verkkosivujen ohjelmointiin käytetään PHP-ohjelmointikieltä ja sovellus tallentaa tietonsa MySQL-tietokantaan. Tällä yhdistelmällä on mahdollista luoda monenlaisia dynaamisia verkkosivustoja. Koska lisäksi yhdistelmää käytetään laajasti on siitä kehittynyt toimintavarma ja ongelmatilanteisiin on saatavilla runsaasti apua. Myös aihetta käsittelevää kirjallisuutta ja muuta opetusmateriaalia on runsaasti tarjolla.

Kuvassa 2 on esitetty kaavio dynaamisten verkkosivujen toiminnasta kokonaisuutena. Kaavio jakautuu kolmeen osaan. Oikeassa laidassa on asiakas, joka tarkoittaa palvelun käyttäjää. Tyypillisin HTTP-asiakasohjelmisto on internetselain. Tosin mikä tahansa ohjelmisto, joka lataa sisältöjä verkosta, voidaan luokitella HTTP-asiakkaaksi. Kuvan vasemmassa laidassa on palvelin. Palvelimella toimiva HTTP-palvelin on ohjelmisto, joka vastaa asiakkaan lähettämiin sivupyyntöihin. Nämä sivupyynnöt kulkevat tietoliikenneverkkoja pitkin. LAMP-ympäristöissä HTTP-palvelimena käytetään Apache HTTP Server -ohjelmistoa. Dynaamiset verkkosivut toimivat palvelimella erillisinä ohjelmina. LAMP-ympäristöissä käytetään dynaamisten verkkosivujen toteuttamiseen PHP-ohjelmointikieltä. Esimerkiksi Joomla-sisällönhallintajärjestelmä on PHP-ohjelmointikielellä kehitetty dynaaminen verkkosovellus. Nämä dynaamiset verkkosivustot tyypillisesti hakevat ja tallentavat tietoja tietokannasta. LAMP-ympäristöissä käytetään MySQL -tietokannanhallintajärjestelmää. Kun sovellus on saanut muodostettua vastauksen asiakkaan lähettämään sivupyyntöön, se välittää vastauksen HTTP-palvelimelle, joka taas lähettää vastauksen asiakkaalle tietoverkon yli.



Kuva 2. Asiakas-palvelinmalli.

2.2.6 Sisällönhallintajärjestelmät

Sisällönhallintajärjestelmä on moniselitteinen termi. Boiko esittää joukon määritelmiä sisällönhallintajärjestelmälle. Määritelmät riippuvat siitä, miltä kannalta sisällönhallintajärjestelmää katsotaan. Yleisellä tasolla Boiko määrittelee sisällönhallinnan prosessiksi, jossa hallitaan tuotettujen sisältöjen julkaisemista. Tällöin sisällönhallintajärjestelmä on ohjelmisto, joka tarjoaa työkaluja tähän prosessiin. (Boiko 2005, 65—66.)

Ensimmäinen Boikon määritelmistä määrittelee sisällönhallintajärjestelmän julkaisujen kysynnän ja tarjonnan yhdistävänä työkaluna. Informaation välittäminen on luonnollinen osa organisaation tai yrityksen toimintaa. Informaation välittämisellä pyritään saavuttamaan haluttu päämäärä, oli se sitten mikä tahansa. Yritykset hyödyntävät sisällönhallintajärjestelmiä esimerkiksi yritysviestinnän sekä markkinoinnin tarpeisiin. Samalla tavalla esimerkiksi poliittiset järjestöt voivat hyödyntää sisällönhallintajärjestelmiä oman agendansa levittämiseen. (Boiko 2005, 67—68.)

Toinen Boikon määritelmistä määrittelee sisällönhallintajärjestelmän sen mukaan, mitä sen pitäisi tehdä. Järjestelmää käyttävät käyttäjät, jotka jakautuvat useisiin eri rooleihin. Jotkut käyttäjät tuottavat järjestelmään uutta sisältöä. Joidenkin käyttäjien tehtävä on muokata ja hallinnoida olemassa olevaa sisältöä. Joillekin käyttäjille järjestelmä on palvelu, jota he käyttävät ja josta he etsivät itseään kiinnostavaa sisältöä. Sisältö, jota järjestelmässä hallinnoidaan, riippuu järjestelmän käyttötarkoituksesta. Se voi olla mitä tahansa digitaalisessa muodossa olevaa informaatiota. Tyypillisesti sisältö on tekstiä, kuvia tai videota. Myös se, mitä käyttäjät voivat sisällölle tehdä, riippuu järjestelmän käyttötarkoituksesta. Eri käyttäjärooleilla voi olla eri asteisia oikeuksia sisältöihin. Nämä kaikki yhdessä asettavat sisällönhallintajärjestelmälle vaatimukset, jotka sen tulisi toteuttaa ollakseen asianmukainen. (Boiko 2005, 69—70.)

Sisällönhallintajärjestelmään pohjautuvan verkkosivun toteutus ja ylläpitö voi vaatia useita eri alojen osaajia. Näitä henkilöitä ovat esimerkiksi suunnittelijat, jotka suunnittelevat sisällönhallintajärjestelmän rakenteen ja toiminnan toteuttajat, jotka vastaavat järjestelmän teknisestä toteuttamisesta, sisällön tuottajat jotka tuottavat järjestelmään var-

sinaisen sisällön sekä esimerkiksi graafikot, jotka vastaavat siitä, kuinka sisältö esitetään. (Boiko 2005, 70—71.)

Neljäntenä Boiko määrittelee sisällönhallinnan kolmivaiheiseksi prosessiksi. Tuotantovaiheessa sisältöä, joko tuotetaan itse tai se kerätään jostain muusta lähteestä. Tämä sisältö sitten lisätään järjestelmään joko sellaisenaan tai jollakin tavalla muokattuna. Järjestelmään lisättyä sisältöä voidaan hallita esimerkiksi määrittämällä se johonkin kategoriaan tai muulla tavalla suhteuttamalla se jo olemassa olevaan sisältöön. Lopulta sisältö julkaistaan siten, että se on loppukäyttäjien saatavilla. (Boiko 2005, 72—73.)

Viimeisenä Boiko tarkastelee sisällönhallintaa tietoteknisestä näkökulmasta. Digitaalisen sisällön sisällönhallinta vaatii ohjelmisto- ja laitteistoresursseja. Sisällönhallintajärjestelmän laitteistovaatimukset riippuvat sisällön määrästä sekä käyttöaktiivisuudesta. Vaadittu ohjelmistoympäristö riippuu taas sisällönhallintajärjestelmän teknisestä toteutustavasta. (Boiko 2005, 74—75.)

Sisällönhallintajärjestelmät jaetaan usein karkeasti kahteen alakategoriaan. ECM (enterprise content management system) on jonkin organisaation sisäiseen sisällönhallintaan tarkoitettu järjestelmä. WCM (web content management system) on järjestelmä, joka on tarkoitettu nimenomaan verkkosivujen sisällön hallintaan. Tässä opinnäytetyössä sisällönhallintajärjestelmällä tarkoitetaan nimenomaan verkkosivujen sisällön hallintaan tarkoitettua järjestelmää.

Käytännössä sisällönhallintajärjestelmä on ohjelma, jonka avulla hallitaan verkkosivulla julkaistua informaatiota. Se tarjoaa sivuston ylläpitäjille mahdollisuuden muokata ja julkaista sisältöä. Sisällönhallintajärjestelmät tarjoavat aina myös niin sanotun julkisen puolen, joka näkyy sivuston vierailijoille verkkosivuna. Sivuston kävijät eivät usein edes tiedä olevansa tekemisissä sisällönhallintajärjestelmän kanssa. Sisällönhallintajärjestelmät eivät määritä sitä, miltä verkkosivu näyttää vaan sen, kuinka verkkosivulla olevaa sisältöä hallitaan ja millä tavoin se on käyttäjien käytettävissä.

2.2.7 Avoimen lähdekoodin sisällönhallintajärjestelmät

Avoin lähdekoodi on melko laveasti käytetty termi. Sillä kuitenkin tarkoitetaan ohjelmistoja, jotka on julkaistu tietyt erityisehdot täyttävällä lisenssillä. Tällaisia avoimen lähdekoodin lisenssejä on useita erilaisia. Open Source Initiative on luonut kymmenkohtaisen määritelmän avoimen lähdekoodin ohjelmistolisensseille. Tärkeimpänä määritelmänä listassa on mahdollisuus levittää ohjelmistoa veloituksetta. Ohjelmiston lähdekoodi tulee olla saatavilla ja ohjelmiston käyttäjillä on oikeus muokata lähdekoodia sekä hyödyntää sitä omissa ohjelmistoissaan. (Perens 1999, 171—189.)

Avoimen lähdekoodin lisensseistä suosituin on GNU General Public License (Lee 2007). Tällä lisenssillä on julkaistu useita sisällönhallintajärjestelmiä. Vuonna 2010 tehdyn kyselytutkimuksen mukaan näistä julkaisujärjestelmistä kolme erottui selkeästi edukseen muista. Järjestelmiä vertailtiin sekä käyttömäärien että brändien vahvuuksien perusteella. Tutkimuksen mukaan suosituin järjestelmä oli WordPress. Tätä seuraisivat Joomla! sekä Drupal. (Shreves 2010, 38.)

2.3 Joomla-sisällönhallintajärjestelmä

2.3.1 Yleistä

Päätin keskittyä tässä opinnäytetyössä Joomla-sisällönhallintajärjestelmään, koska minulla oli aikaisempaa kokemusta kyseisestä järjestelmästä. Lisäksi WordPressissä on sisään rakennettu mahdollisuus käyttää avainsanoja toisin kuin Joomla-järjestelmässa. Joomla-sisällönhallintajärjestelmän virallinen kirjoitusasu on Joomla!. Tässä opinnäytetyössä kieliopillisista syistä käytän kirjoitusasua Joomla.

Joomla-projekti sai alkunsa elokuussa 2005. Se pohjautui aikaisempaan Mambo-sisällönhallintajärjestelmään, ja ensimmäiset versiot Joomlasta olivat edeltäjänsä kaltaisia. Tammikuussa 2008 julkaistu Joomlan versio 1.5 oli ensimmäinen versio, jossa oli selkeitä rakenteellisia muutoksia Mamboon verrattuna. (Hutchinson 2010.) Puhuttaessa Joomlan versioista käytetään merkintätapaa 1.0, 1.5 tai 1.6. Nämä ovat järjestelmän his-

toriassa versioita, jotka eroavat toisistaan merkittävästi. Eroja löytyy niin sisäisistä rakenteista, järjestelmän käyttämistä konsepteista kuin käyttöliittymästäkin. Tämä opinnäytetyö perustuu Joomlan versioon 1.5. Opinnäytetyön teon aikana tammikuussa 2011 julkaistiin Joomlasta versio 1.6.

2.3.2 Joomla-sisällönhallintajärjestelmä käyttäjän näkökulmasta

Käyttäjän näkökulmasta Joomla on ohjelmisto, jonka voi ladata ilmaiseksi verkosta. Ohjelmisto asennetaan palvelimelle, minkä jälkeen järjestelmä on käyttövalmis. Järjestelmä tarjoaa käyttäjilleen kaksi käyttöliittymää. Ensimmäinen näistä on julkinen ja se on tarkoitettu sisällön esittämiseen. Toinen käyttöliittymä on tarkoitettu sisällön hallintaan ja se edellyttää käyttäjiltä tunnistautumista sisäänkirjautumalla.
Joomla tallentaa sisällön artikkeleina. Artikkeli on tyypillisesti tekstistä ja kuvista koostuva kokonaisuus. Sisällöntuottajat voivat kirjoittaa uusia artikkeleita ja muokata olemassa olevia. Artikkeleita voi tarvittaessa poistaa tai estää niitä näkymästä julkisesti. Artikkelin näkymisen voi myös rajata pelkästään jollekin tietylle käyttäjäryhmälle. Tarvittaessa artikkeleja voidaan ryhmitellä pääryhmiin ja alaryhmiin.

Sivuston yleinen rakenne määritellään valikoiden kautta. Valikko on kokoelma käyttäjälle tarjottavia linkkejä erilaisiin näkymiin. Näkymä voi olla esimerkiksi yksittäinen järjestelmään tallennettu artikkeli tai vaikka listaus jonkin ryhmän artikkeleista. Listausnäkymissä artikkeleista voidaan näyttää esimerkiksi vain otsikko, koko artikkeli tai vaikka pieni pätkä artikkelin alusta. Lisäksi artikkelilistauksiin on mahdollista määrittää, kuinka monta artikkelia näkyy kerralla. Jos kaikki listauksen artikkelit eivät mahdu kerralla näytettäväksi, voi artikkeleita selata erillisen sivutuksen avulla.

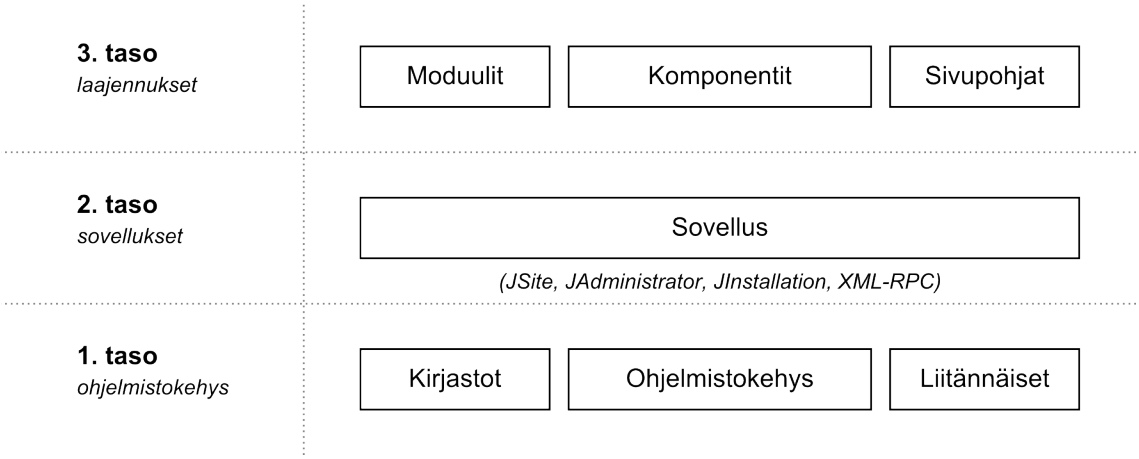
Joomlan perustoiminta keskittyy vahvasti artikkeleihin. Jos sivustolle halutaan lisätä esimerkiksi keskustelualue tai kuvagalleria, on otettava käyttöön erillinen laajennus. Avoimen lähdekoodin projekteihin kuuluu usein aktiivinen käyttäjäkunta, joka osallistuu projektin kehittämiseen. Joomlan tapauksessa se tarkoittaa esimerkiksi näiden laajennusten kehittämistä. Suurin osa laajennuksista toimii samoilla avoimen lähdekoodin periaatteilla kuin varsinainen sisällönhallintajärjestelmäkin. Joomlan virallisessa laajen-

nuksia tarjoavassa sivustossa on listattu yli 7 000 erillistä laajennusta (Joomla! Extensions Directory 2011).

Laajennukset asennetaan sivustolle erillisen laajennusten hallintanäkymän kautta. Vain tietyillä käyttäjäryhmillä on oikeus hallita järjestelmään asennettuja laajennuksia. Laajennuksia on moneen eri tarkoitukseen ja siksi ne toimivat eri tavoilla. Tyypillistä kuitenkin on, että laajennus lisää sivuston hallintakäyttöliittymään uuden hallintanäkymän itselleen. Tämän lisäksi laajennuksesta voi usein luoda uusia näkymiä sivuston puolelle.

Sivuston ulkoasu määritellään erillisten sivupohjien avulla. Myös nämä sivupohjat ovat eräänlaisia sivustolle asennettavia laajennuksia. Sivupohjien käytöllä pyritään erottamaan toisistaan varsinainen sisältö sekä se, miten sisältö esitetään. Sivupohjia ja sitä kautta sivuston ulkoasua voi vaihtaa ilman, että se vaikuttaa sivustolla olevaan sisältöön.
Vaikka Joomla-projekti käyttää englantia virallisena kielenään, on järjestelmä mahdollista saada toimimaan myös monella muulla kielellä. Sekä järjestelmästä itsestään että monista laajennuksista on olemassa erillisiä kielikäännöksiä. Nämä käännökset ovat yhteisön tuottamia ja niiden valikoima onkin siksi hyvin vaihteleva. Myös kielikäännösten laatu vaihtelee. Vaikka järjestelmän käyttämää kieltä voi vaihtaa melko helposti, on varsinaisen sisällön monikielisyyden hallinta huomattavasti hankalampaa. Jos järjestelmän käyttäjät koostuvat saman kielen osaajista, ei ongelmaa ole. Jos sisältöä on tarve tuottaa usealla eri kielillä, vaatii se tarkempaa suunnittelua. Tätäkin ongelmaa helpottamaan on olemassa erillisiä laajennuksia.

2.3.3 Joomla-sisällönhallintajärjestelmä ohjelmoijan näkökulmasta



Kuva 3. Joomlan arkkitehtuuri

Rakenteellisesti Joomla koostuu kolmesta eri kerroksesta. Alimman tason muodostaa Joomlan ohjelmistokehys. Tämä ei ole itsenäinen sovellus vaan pitää sisällään ohjelmistokirjastoja, joita ylemmät rakenteen kerrokset hyödyntävät. Näistä ohjelmistokirjastoista löytyy työkaluja esimerkiksi tietokannan kanssa kommunikointiin tai kielikäännösten hallintaan. (Joomla! Documentation 2011a.)

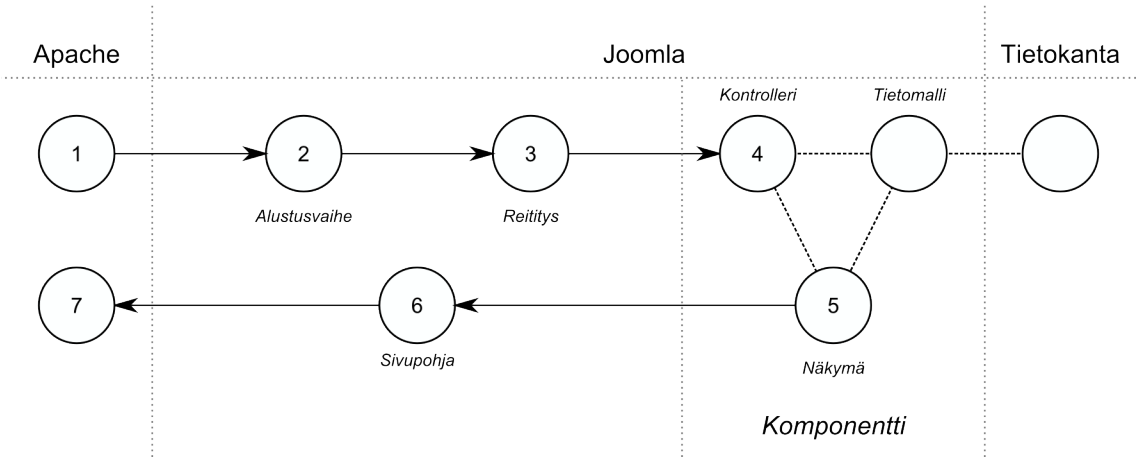
Rakenteen toisen tason muodostaa ohjelmistokehystä hyödyntävä sovellus. Tältä kannalta katsottuna Joomla koostuu neljästä erillisestä sovelluksesta. Käyttäjille tarkoitetut julkinen näkymä sekä hallintanäkymä ovat molemmat erillisiä sovelluksia. Tämän lisäksi Joomlan asennusohjelma toimii omana erillisenä sovelluksena sekä järjestelmän etäkäytön mahdollistama sovellus omanaan. Järjestelmän etäkäyttösovellus on ainut, jolla ei ole käyttäjille näkyvää käyttöliittymää. Se on tarkoitettu sitä varten että ulkopuoliset ohjelmat voivat tarvittaessa kommunikoida järjestelmän kanssa. (Joomla! Documentation 2011a.)

Ylimmän tason muodostavat moduulit, komponentit sekä sivupohjat. Sivupohjat määrittävät pitkälle sen, miltä sovellus näyttää käyttäjälle. Moduulit liittyvät vahvasti sivupohjiin, ja niitä käytetään hallitsemaan, mitä sisältöä näkyy sivuston eri näkymissä. Komponentit vastaavat käyttäjille tarjottavista toiminnoista sekä erilaisten näkymien muodosta-

misesta. Ne ovat ehkä merkittävin eri laajennustyypeistä. (Joomla! Documentation, 2011a.)

2.3.4 Joomlan yleinen toimintamalli

Joomlan tehtävä palvelimella on muodostaa vastaus selaimelta tulleeseen sivupyyntöön. Erillinen palvelinsovellus vastaa sivupyynnön vastaanottamisesta ja sen välittämisestä sisällönhallintajärjestelmälle. Sisällönhallintajärjestelmä käsittelee sivupyynnön ja muodostaa sivupyyntöä vastaavan vastauksen, jonka palvelinsovellus sitten välittää takaisin selaimelle. Tyypillistä dynaamisille verkkosovelluksille ja siten myös Joomlalle on, etteivät ne ole jatkuvasti käynnissä olevia prosesseja. Sen sijaan sovellus käynnistetään erikseen jokaista sivupyyntöä varten ja sovellus lopetetaan, kun vastaus sivupyyntöön on muodostettu.



Kuva 4. Joomlan toimintakaavio

Kuva 4 kuvaa Joomla-järjestelmän toimintaa. Suoritus alkaa siitä, kun HTTP-palvelin välittää asiakkaalta tulleen sivupyynnön sille. Tämän jälkeen Joomla suorittaa alustusvaiheen. Tässä vaiheessa suoritetaan erilaisia toimintoja, jotka varsinaisen sovelluksen suorittaminen edellyttää. Tällaisia ovat muun muassa tietokantayhteyksien luominen.

Alustusvaiheen jälkeen suoritus siirretään komponentille, jolle sivupyyntö on tarkoitettu. Tätä vaihetta kutsutaan nimellä reititys. Käytännössä aina kun selain lähettää palveli-

melle sivupyyntöjä, jotka kohdistuvat Joomla-järjestelmään, määritellään sivupyyntöön komponentin nimi, jolle sivupyyntö on tarkoitettu. Toisin kuin kuvassa 4, Joomla-järjestelmässä on aina useita, joskus jopa kymmeniä eri komponentteja. Reititysvaiheen tärkein tehtävä onkin ohjata ohjelman suoritus juuri oikealle komponentille.

Kaikki Joomlan toiminta perustuu komponentteihin. Komponentilla on aina jokin vastuualue ja se vastaa kyseiseen aiheeseen liittyvistä sivupyynnöistä. Esimerkiksi kun käyttäjä haluaa nähdä yhden artikkelin sivulla, kohdennetaan sivupyyntö komponentille com_content, joka vastaa artikkeleista. Sivupyyntöön liitetään lisäksi tieto siitä, mikä artikkeli halutaan nähdä. Tämän jälkeen kyseinen komponentti noutaa halutun artikkelin tietokannasta ja lähettää sen takaisin selaimelle. Sama komponentti osaa muodostaa myös kaikki erilaiset artikkelilistaukset. (LeBlanc 2008, 8.)

Sisäisesti komponentti jakaantuu kolmeen eri tyyppiseen elementteiin. Kontrolleri ohjaa koko komponentin toimintaa ja tekee päätökset, kuinka kuhunkin sivupyyntöön reagoidaan. Tietomalli vastaa komponentin käsittelemien sisältöjen ja tietojen hakemisesta ja tallentamisesta tietokantaan. Näkymä vastaa taas vastauksena palautettavan HTML-dokumentin muodostamisesta.

Komponentin suorituksen tuloksena syntyy usein HTML-koodia. Tämä HTML-koodi upotetaan sivupohjassa sille varattuun paikkaan. Valmis lopputulos palautetaan sitten vastauksena takaisin HTTP-palvelimelle, joka välittää HTML-dokumentin edelleen asiakkaalle.

2.3.5 Joomla-komponentit

Joomlan modulaarisen rakenteen keskeisimpiä osia ovat komponentit. Yksi komponentti vastaa sisällöllisesti yhteen liittyvistä toiminnoista. Tällaisia ovat esimerkiksi käyttäjiin ja käyttäjienhallintaan liittyvät toiminnot. Näistä toiminnoista vastaan Joomlassa oma käyttäjäkomponentti. Kaikki komponentit tunnistetaan Joomlassa teknisen nimen perusteella. Tämä tekninen nimi alkaa aina kolmikirjaimisella lyhenteellä ”com”, jota seuraa alaviiva. Tämän jälkeen seuraa komponentin nimi. Komponenttien nimet koostuvat

usein yhdestä tai kahdesta sanasta eivätkä ne sisällä välilyöntejä tai mitään erikoismerkkejä. Esimerkiksi Joomlan käyttäjäkomponentti on tekniseltä nimeltään com_user.

Käyttäjälle komponentit näyttäytyvät usein kaksiosaisena. Komponentista on olemassa sekä julkinen puoli, joka näkyy sivustolla sivuston vierailijoille sekä hallintakäyttöliittymä, joka näkyy sivuston ylläpitäjille Joomlan hallinnassa. Teknisesti on kyse kahdesta eri komponentista, jotka käsittelevät samoja tietoja. Kaikki komponentit eivät tarjoa minkäänlaista hallintaliittymää ja toisaalta on myös komponentteja, jotka eivät näy millään tavalla järjestelmän julkisella puolella.

Komponentin tarkoitusta ei ole määritelty. Esimerkiksi Joomlan käyttäjäkomponentti vastaa muun muassa uusien käyttäjien rekisteröinnistä sekä rekisteröityneiden käyttäjien sisäänkirjautumisista. Lisäksi komponentti tarjoaa käyttäjille mahdollisuuden palauttaa unohtunut salasana. Komponentin hallintakäyttöliittymän kautta sivuston ylläpitäjät voivat luoda uusia käyttäjiä tai muuttaa olemassa oleviin käyttäjiin liittyviä asetuksia. Tällaisia asetuksia ovat muun muassa käyttäjän käyttöoikeustaso tai vaikka kieli, jota käyttäjä haluaa käyttää. Yleisellä tasolla voi sanoa, että komponentti vastaa jonkin sisältötyypin hallinnoimisesta sekä siihen liittyvistä toiminnoista.

Joomlan hakemistorakenne heijastaa järjestelmän arkkitehtuuria. Juurihakemistossa olevat tiedostot liittyvät pääsääntöisesti sivustosovellukseen tai ovat yhteisiä koko järjestelmälle. Asennukseen käytettävän asennussovelluksen tiedostot sijaitsevat installation-hakemistossa. Hallintakäyttöliittymäsovellus sijaitsee administrator-hakemistossa. Hallintasovellus muistuttaa rakenteeltaan sivustosovellusta. Kummassakin tapauksessa kaikki komponentit on tallennettu components-hakemistoon. Tämän hakemiston alla tiedostot on jaettu komponenttien teknisten nimien mukaan. Jokaisen komponentin hakemistosta tulee löytyä tiedosto, joka on sama kuin komponentin tekninen nimi ilman alkuliitettä. Tästä tiedostosta alkaa komponentin suoritus. Hakemistorakenteen ja tiedostojen osalta nämä ovat Joomlan asettamat vähimmäisvaatimukset komponentille.

Komponentista tulee olla merkintä myös tietokannan components-taulussa. Tämä kyseinen tietokantataulu on rakenteeltaan sekä käyttötarkoitukseltaan sekava, enkä itse ainakaan ole pystynyt tyhjentävästi selvittämään taulun kaikkien sarakkeiden käyttötarkoi-

tusta. Tietokantataulua käytetään samaan aikaan pitämään kirjaa järjestelmässä olevista komponenteista sekä muodostamaan valikkorakenne Joomlan hallintasovellukseen. Lisäksi tauluun tallennetaan komponenttikohtaisia asetuksia.

Kun komponentti on luotu oikein sekä hakemistorakenteeseen että tietokantaan on komponentti toimintavalmis. Jokaisella sivupyynnöllä, joka Joomlalle lähetetään, määritellään komponentti, joka käsittelee kyseisen sivupyynnön. Kaikkein yleisintä on määritellä komponentti niin kutsutulla GET-parametrilla. Tällöin komponentin nimi näkyy suoraan sivun osoitteessa. Tästä onkin muodostunut keino, jolla helposti tunnistaa Joomla-sivuston. Kaaviossa 1 kysymysmerkki tiedoston nimen jälkeen tarkoittaa GET-parametrilistan alkua. Parametrilista koostuu parametrin nimestä, sitä seuraavasta yhtäläisyysmerkistä sekä parametrin arvosta. Jos parametrejä on useampi, ne erotellaan toisistaan &-merkillä. Option-parametri Joomlassa määrittää komponentin, jolle sivupyyntö kohdistuu.

http://www.domain.com/index.php?option=com_adam
Kaavio 1.

Kaavion 1 sivupyyntö ohjaisi suorituksen Joomlassa määritellylle komponentille. Suorituksen ohjaaminen tarkoittaa käytännössä sitä, että järjestelmä avaa komponentin hakemistosta komponentin nimeä vastaavan tiedoston ja suorittaa sen. Vähääkään isommat komponentit jakautuvat useisiin eri tiedostoihin ja näin ollen ensimmäisenä suoritettava tiedosto lähinnä ohjaa suorituksen komponentin sisällä oikeaan paikkaan.

Joomlan ohjelmointirajapinnat tarjoavat kehittäjille mahdollisuuden hyödyntää komponenteissa niin kutsuttua MVC-mallia. Tässä komponentin toiminta jaetaan kolmeen eri vastuualueeseen. Kontrolleri (controller) ottaa sivupyynnön vastaan ja päättää tarkemmin kuinka siihen reagoidaan. Tietomalli (model) vastaa ulkoisten tietolähteiden kanssa kommunikoinnista. Joomlan tapauksessa tämä käytännössä tarkoittaa tietojen hakemista tai tallentamista tietokantaan. Näkymä (view) vastaa käyttäjälle näytettävän HTML-dokumentin muodostamisesta. Malli on melko yleisesti käytössä erityisesti verkkosovelluksia ohjelmoitaessa. MVC-malli on melko abstrakti eikä sen soveltaminen käytäntöön

ole täysin yksiselitteistä. Monissa tilanteissa se tarjoaa kuitenkin hyödyllisen tavan lähestyä ohjelmiston kehitystä.

Oman kehitystyöni aikana tutkin kuinka muut komponentit hyödyntävät MVC-mallia käytännössä ja yritin löytää parhailta vaikuttavia ratkaisuja. Lopulta päädyin ratkaisuun, jossa komponentti jaetaan eri näkymiin. Parhaiten tämä konkretisoituu komponentin hallintaliittymässä. Jokainen hallitaliittymän näkymä on yksi komponentin osio. Jokaisella komponentin osiolla on oma kontrolleri, tietomalli sekä näkymä. Tässä mallissa käytettävä osio on aina määriteltävä sivupyynnön yhteydessä. Kaaviossa 2 käytettävä osio määritellään antamalla halutun osion kontrollerin nimi sivupyynnön yhteydessä. Komponentin päätiedoston tehtäväksi jää tällöin ladata oikea kontrolleri controller-parametrin perusteella.

http://www.domain.com/index.php?option=com_adam&controller=entities
Kaavio 2.

Pelkkä oikean kontrollerin lataaminen ei kuitenkaan vielä riitä. Kontrollerille on määriteltävä tehtävä, joka sen tulisi toteuttaa. Suoritettava tehtävä määritellään task-parametrilla. Tyypillisin tehtävä on jonkin osion näyttäminen. Tämä on niin tyypillinen, että sitä käytetään oletustehtävänä, jos mitään tehtävää ei ole erikseen sivupyynnössä määritelty. Kaaviossa 3 on on sivupyyntö, joka suorittaisi synkronisointitoimenpiteen.

http://www.omadomain.com/index.php?
option=com_adam&controller=entities&task=sync
Kaavio 3.

Jos tehtävän suoritus vaatii vielä lisätietoja, ne ilmoitetaan samalla tavalla erillisinä parametreinään. Esimerkiksi sisäänkirjautumisessa käyttäjäkomponentti saa task-parametrikseen komennon ”login” ja tämän lisäksi käyttäjätunnuksen ja salasanan, jotka käyttäjä on kirjoittanut sisäänkirjautumislomakkeeseen. Kun tehtävä on delegoitu kontrollerille, riippuu jatkotoiminta aina tehtävästä. Jos tehtävä on muodostaa listaus kaikista käyttäjistä, välittää kontrolleri suorituksen edelleen näkymälle. Näkymä taas pyytää tietomallia hakemaan tietokannasta listan käyttäjistä. Tämän jälkeen näkymä muotoilee

käyttäjälistan HTML-dokumentiksi. Raja tietomallin ja näkymän välillä on kaikkein häilyvin. Sinällään ei ole mitään teknistä estettä sille, etteikö edellisessä esimerkissä käyttäjälistaa olisi voinut hakea tietokannasta suoraan näkymässä. Kysymys on lähinnä ohjelman jakamisesta loogisiin osiin ja eri vastuualueisiin.

Kaikki tehtävät eivät tuota näkyvää tulosta. Esimerkiksi sivustolle sisäänkirjautumisen onnistuessa ei muodosteta erillistä HTML-dokumenttia, joka ilmoittaisi käyttäjän onnistuneesta kirjautumisesta sisään. Tällaisia tilanteita varten sivupyyntöön ei ole pakko vastata aina HTML-dokumentilla. Vaihtoehtona on vastata sivupyyntöön uudelleenohjauksella. Tällöin selain saa ilmoituksen osoitteesta, johon sen olisi syytä tehdä uusi sivupyyntö. Nämä uudelleenohjaukset tapahtuvat aina automaattisesti ja käyttäjä voi huomata ne ainoastaan osoiterivin sisällön muuttumisesta. Onnistuneen sisäänkirjautumisen jälkeen käyttäjä usein ohjataankin takaisin sivuston etusivulle. Suoritettava tehtävä lopulta määritteeä vastataanko sivupyyntöön muodostetulla HTML-dokumentilla vai uudelleenohjauksella. Kontrollerin tehtävä on kertoa Joomlalle, kuinka kyseiseen tehtävään vastataan.

2.4 Tietokannat

Tietokanta voidaan määritellä loogisesti yhteenkuuluvien, tallennettujen tietojen joukoksi, jota voi helposti käsitellä tietokantakielellä (Hovi, Huotari & Lahdenmäki 2003, 4). Tässä opinnäytetyössä tietokanta-termillä viitataan nimenomaan relaatiotietokantaan, joka on yksi tyypillisimmistä tietokantojen tietomalleista (Hovi, Huotari & Lahdenmäki 2003, 6). Joomla tallentaa tietokantaan kaiken sisällönhallintajärjestelmän kautta hallittavissa olevan tiedon. Käytännössä tämä tarkoittaa artikkeleita, sisällönhallintajärjestelmään rekisteröityjä käyttäjiä, käyttäjien käyttöoikeuksia, sivuston valikoiden rakennetta sekä esimerkiksi tietoa käytössä olevista laajennuksista. Tämän lisäksi Joomlan laajennukset voivat tallentaa omia tietojaan ja sisältöjään tietokantaan. Binäärimuotoisia sisältöjä kuten kuva- tai videotiedostoja ei yleensä tallenneta tietokantaan. Tällaisista sisällöistä voidaan tietokantaan tallenntaa sisältöön liittyviä metatietoja sekä tiedostopolku tai osoite, josta varsinaisen sisältö löytyy.

Tietokanta koostuu tauluista, joista kukin kuvaa yhtä asiakokonaisuutta. Taulun sarakkeet määrittävät taulun rakenteen ja taulun rivit sen sisällön. (Hovi, Huotari & Lahdenmäki 2003, 8.) Esimerkiksi Joomlassa kaikki artikkelit on tallennettu yhteen tauluun. Jokaisesta artikkelista tallennetaan artikkelin nimi, kirjoittaja, sisältö sekä monta muuta artikkeliin liittyvää tietoa. Jokainen taulun rivi kuvaa yhtä järjestelmään tallennettua artikkelia. Kuvassa 5 on esimerkki siitä, kuinka tietoja voidaan tallentaa tietokantatauluihin.

id	title	title
1	Welcome to Joomla!	With a fully documented library of developer resources ...
2	How do I localise Joo ...	Translation Teams for Joomla! 1.5 may have also released ...
3	What is the FTP layer ...	The FTP Layer allows file operations (such as installing ...

Kuva 5. Esimerkki tietokantataulusta

Jokaiseen tietokannan tauluun määritellään perusavain. Tämä on sarake, jonka arvo yksilöi jokaisen taulun rivin. Toisin sanoen taulussa ei saa olla kahta sellaista riviä, joiden perusavain olisi sama. (Hovi, Huotari & Lahdenmäki 2003, 9.) Joomlassa yleinen käytäntö on määrittää jokaiseen tauluun sarake nimeltä “id”, jota käytetään taulun perusavaimena. Ensimmäinen tauluun tallennettu rivi saa id-arvokseen 1 ja tätä seuraavat rivit aina edellistä yhtä suuremman arvon.

3 Avainsanat Joomlassa

3.1 Sisältöön liitetyt ja sisältöön linkitetyt avainsanat

Liitetyt avainsanat

id	otsikko	avainsanat
1	harmaalokki	lintu, lokki
2	lapintiira	lintu, tiira
3	rauduskoivu	kasvi, puu, koivu
4	mänty	kasvi, puu, mänty

Linkitetyt avainsanat

id	otsikko
1	harmaalokki
2	lapintiira
3	rauduskoivu
4	mänty

otsikko	avainsana
1	1
1	2
2	1
2	3
3	4
3	5
3	6
4	4
4	5
4	7

id	avainsana
1	lintu
2	lokki
3	tiira
4	kasvi
5	puu
6	koivu
7	mänty

Kuva 6. Sisältöön liitetyt ja linkitetyt avainsanat

Avainsanoja voidaan yhdistää sisältöihin tietokannassa kahdella eri tavalla. Yksinkertaisempi tapa on tallentaa sisältöön liittyvät avainsanat samaan tauluun varsinaisen sisällön kanssa. Tällöin avainsanoille luodaan tauluun oma sarake, johon avainsanat tallennetaan tekstimuodossa. Tätä tapaa kutsun sisältöön liitetyiksi avainsanoiksi.

Yhteen sisältöön voi liittyä useampi eri avainsana ja yksi avainsana voi liittyä useampaan eri sisältöön. Tällöin avainsanojen ja sisältöjen välillä on niin sanottu moni moneen -yhteys. Tietokannassa tällaista asioiden välistä yhteyttä voidaan mallintaa käyttämällä kolmea eri taulua. Ensimmäiseen tauluun tallennetaan varsinaiset sisällöt, toiseen avainsanat ja kolmanteen tallennetaan sisältöjen ja avainsanojen yhteydet. (Hovi, Huotari, Lahdenmäki 2003, 44.) Tämä monimutkaistaa tietokannan rakennetta, mutta helpottaa avainsanojen hallintaa. Jos jotain avainsanaa halutaan muuttaa, tarvitsee muutos tehdä vain siihen tauluun, johon avainsanat on tallennettu. Myös erilaisten tilastojen, kuten avainsanaan liitettyjen sisältöjen lukumäärän selvittäminen, onnistuu kätevästi suoraan yhteystaulusta. Tällä tavalla toteutettuja avainsanoja kutsun sisältöön linkitetyiksi avainsanoiksi.

3.2 Avainsanat Joomlassa

Joomlan versiossa 1.5 ei ole suoraa mahdollisuutta sisällön hallintaan avainsanojen avulla. Toisin sanoen ilman erillisiä laajennuksia ei avainsanoja pysty liittämään sisältöön kummallakaan mainitulla tavalla. Poikkeuksena tähän ovat HTML-dokumentin metatietoihin tallennettavat avainsanat.

Koska Joomla on selaimessa toimiva sisällönhallintajärjestelmä, perustuvat kaikki Joomlan näkymät HTML-dokumentteihin. Yksinkertaisemmin sanottuna Joomlan käyttöliittymä on toteutettu verkkosivuina. HTML-dokumenttiin on mahdollista liittää avainsanoja. Nämä avainsanat määritellään HTML-koodissa dokumenttiin liittyvinä metatietoina. Hakukoneet voivat käyttää näitä avainsanoja hakutulosten muodostamisessa. (W3C 1999.) Joomlassa artikkeliin voi liittää avainsanoja, jotka Joomla liittää näkymän HTML-dokumenttiin. Koska näitä avainsanoja ei ole tarkoitettu sivuston sisäiseen sisällön hallintaan, en käsittele niitä tätä tarkemmin.

Jotkin laajennukset tarjoavat mahdollisuuden hallita kyseiseen laajennukseen liittyviä sisältöjä avainsanojen avulla. Esimerkiksi videoiden hallintaan tarkoitetussa hwdVideoShare-komponentissa on mahdollisuus liittää avainsanoja videoihin. Tällaisten laajennuskohtaisten avainsanojen käyttö rajoittuu aina pelkästään kyseisen laajennuksen sisältöihin.

Joomlaan on olemassa useita laajennuksia, jotka ovat erikoistuneet avainsanojen linkittämiseen artikkeleihin. Aivan kuten laajennuskohtaisissa avainsanoissa, voi näitä avainsanoja linkittää pelkästään artikkeleihin. Jos sivuston sisältö koostuu pääsääntöisesti artikkeleista, voi tällainen avainsanalaajennus olla riittävä kyseisen sivuston tarpeisiin. Ongelma muodostuu silloin, kun halutaan linkittää avainsanoja useisiin eri tyyppisiin sisältöihin sivulla. Käytännössä tilanteen voi ratkaista käyttämällä yhtä aikaa useita avainsanojen liittämis- ja linkittämistekniikoita, esimerkiksi käyttämällä artikkelien avainsanoihin erillistä avainsanalaajennusta ja sen lisäksi laajennusten mahdollisia omia avainsanatoiminnallisuuksia. Tällöin avainsanat eivät kuitenkaan ole keskenään yhteensopivia, mikä voi aiheuttaa ongelmia esimerkiksi haettaessa tietoa avainsanojen avulla.

3.3 Avainsanojen yhdistäminen erityyppisiin sisältöihin

Avainsanoja voidaan yhdistää sisältöön liitämällä tai linkittämällä. Liitettäessä avainsanoja on sisältöä kuvaavassa tietokantataulussa oltava sarake, johon liitetyt avainsanat tallennetaan. Koska iso osa Joomlan laajennuksista ei ole ottanut huomioon avainsanojen käyttömahdollisuutta, on tarvittava avainsanasarake melko harvinainen. Tämän takia etsittäessä ratkaisua avainsanojen yhdistämiseen eri sisältötyyppeihin on avainsanojen liittäminen käytännössä mahdoton ratkaisu. Toimiakseen se vaatisi laajennusten kehittäjiltä yhtenäistä sopimusta avainsanasarakkeen käytöstä sekä muutoksia tuhansiin jo olemassa oleviin laajennuksiin. Tämän takia keskityn avainsanojen linkittämiseen.

Avainsanojen linkittäminen yhteen sisältötyyppiin on melko yksinkertaista. Jokainen sisältö on yksilöity perusavaimella. Avainsanojen ja sisältöjen välisessä suhdetaulussa on ilmoitettu jokainen linkitys kahden perusavaimen parina. Toinen perusavaimista määrittää linkityksen kohteena olevan sisällön ja toinen määrittää avainsanan, joka kyseiseen sisältöön halutaan linkittää. Ongelma kuitenkin syntyy, kun suhdetaulussa halutaan viitata sisältöihin useasta eri taulusta. Sisällön yksilöivä perusavain ei kuitenkaan itsessään kerro, mihin sisältötauluun se viittaa. Esimerkiksi Joomlassa paljon käytetyt id-numerot yksilöivät rivin taulun sisällä mutta eivät kerro, minkä taulun rivistä on kysymys. Toinen ongelma, joka seuraa samantyyppisistä perusavaimista, ovat päällekkäiset arvot. Esimerkiksi jos taulun ensimmäinen rivi saa aina id-arvokseen yksi, tarkoittaa se sitä, että kaikki sisältötaulut, joissa on vähintään yksi rivi, sisältävät rivin, jonka perusavaimen arvo on yksi. Jos avainsanojen suhdetaulussa avainsana on linkitetty id-numeroon yksi, on täysin mahdotonta sanoa, onko avainsana liitetty jokaisen sisältötaulun ensimmäiseen riviin vai vain osaan tai ainoastaan yhteen riviin. Toisin sanoen, pelkkä taulun perusavain ei riitä yksilöimään sisältöjä, kun avainsanoja halutaan linkittää sisältöihin useista eri tauluista.

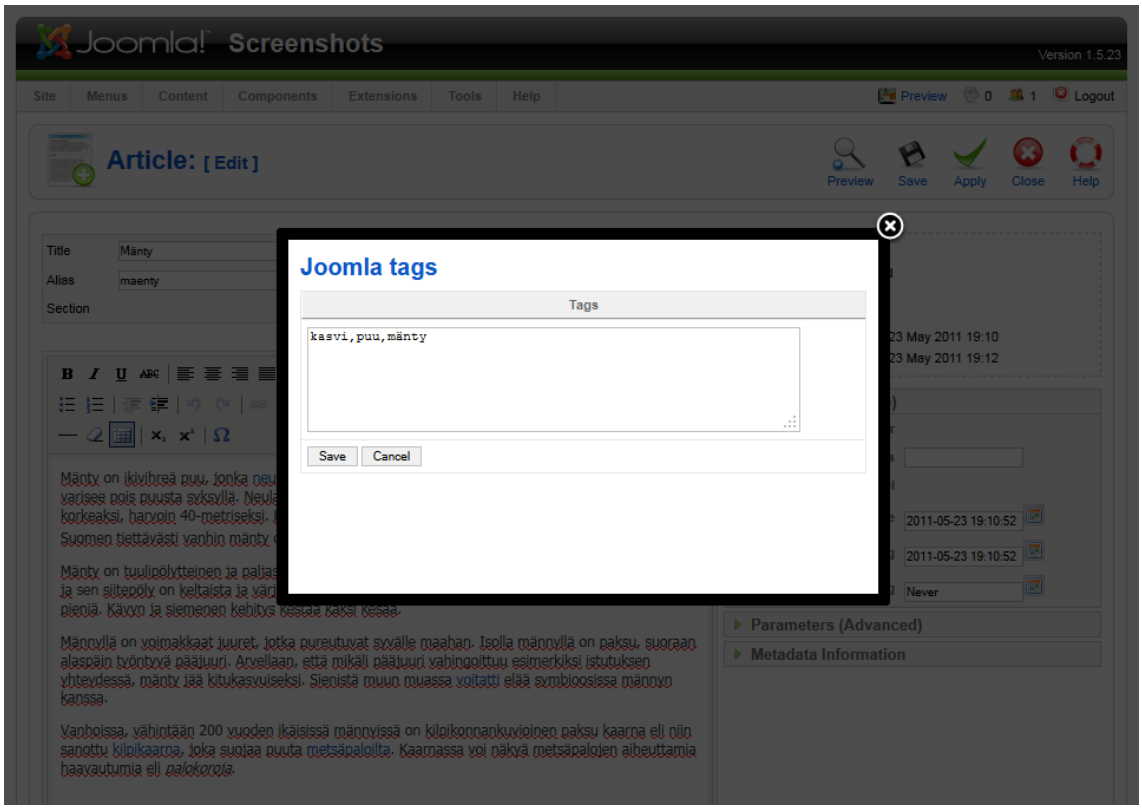
3.4 Avainsanoihin erikoistuneet Joomla-laajennukset

Joomlaan löytyy useita eri tyyppisiä avainsanalaajennuksia. Avainsanalaajennukset voidaan jakaa karkeasti kolmeen eri ryhmään. Ensimmäiseen ryhmään kuuluvat laajennuk-

set, jotka linkittävät avainsanoja artikkeleihin. Toiseen ryhmään kuuluvat laajennukset, jotka lisäävät johonkin muuhun laajennukseen avainsanojen käyttömahdollisuuden. Kolmanteen ryhmään kuuluvat sekalaiset avainsanalaajennukset. Tällaisia ovat esimerkiksi laajennukset, jotka automaattisesti etsivät artikkelista eniten käytettyjä sanoja ja muodostavat näiden pohjalta erilaisia avainsananäkymiä.

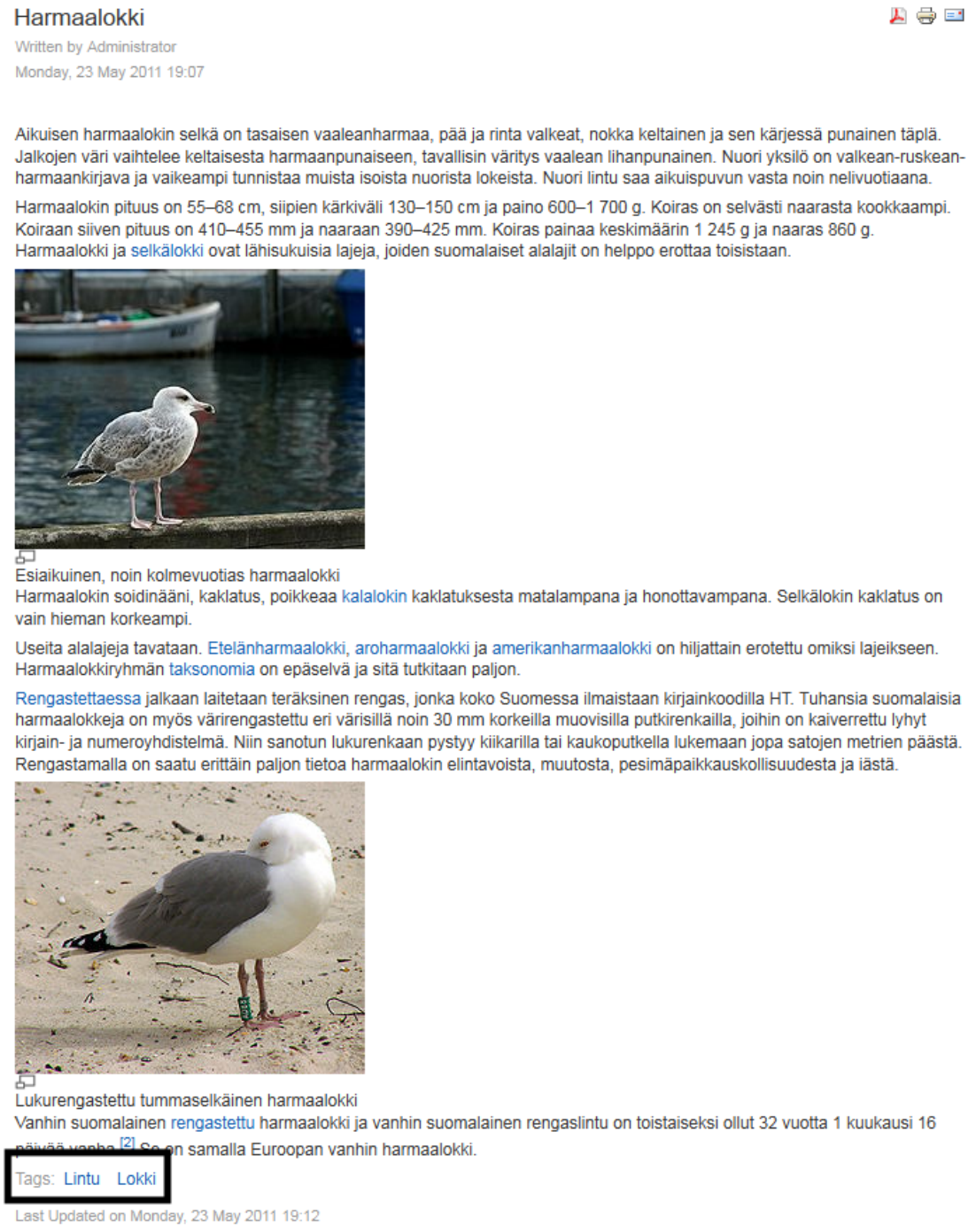
Perehdyin erityisesti Tags for Joomla -nimiseen laajennukseen. Kyseinen laajennus on yksi käytetyimmistä avainsanoihin liittyvistä laajennuksista. Se koostuu varsinaisesta komponentista, muutamasta moduulista ja muutamasta liitännäisestä. Moduulien tarkoitus on tarjota sivustolle erilaisia avainsanoihin liittyviä käyttöliittymiä. Liitännäiset integroivat laajennuksen paremmin sisällönhallintajärjestelmään.

Eräs laajennuksen mukana tulevista liitännäisistä lisää Joomlan omaan artikkelin muokkausnäkymään painikkeen, jota kautta kyseiseen artikkeliin voi liittää avainsanoja. Kuvasta 7 näkyy, kuinka artikkelin muokkausnäkymässä olevaa painiketta klikkaamalla avautuu uusi ikkuna, johon voi kirjoittaa haluamansa avainsanat pilkulla erotettuna toisistaan. Tämän jälkeen kirjoitetut avainsanat tallentuvat tietokantaan.



Kuva 7. Artikkeliin liittyvät avainsanat muokkausnäkymässä

Toinen laajennuksen mukana tulevista liitännäisistä listaa artikkeliin liitetyt avainsanat, kun artikkeli näytetään sivuston puolella. Jokainen avainsana toimii linkkinä erilliseen näkymään, jossa listautuvat kaikki artikkelit, joihin kyseinen avainsana on liitetty. Liitännäinen tulostaa avainsanalistan artikkelin loppuun. Ulkopuolisen on mahdotonta hahmottaa näkyvästä kokonaisuudesta, että avainsanalista on peräisin jostakin erillisestä laajennuksesta. Tämä on esimerkki siitä, kuinka laajennukset integroivat omia käyttöliittymiään olemassa oleviin. Tällaiset liitännäiset, jotka liittävät omia sisältöjään artikkelinäkymiin ovat yksi liitännäisten alalaji. Kuvassa 8 näkyy tyypillinen artikkelinäkymä sellaisena kuin sivustolla vierailija sen näkee. Kuvasta on korostettu liitännäisen muodostama sisältö ääriviivalla.



Harmaalokki

Written by Administrator

Monday, 23 May 2011 19:07

Aikuisen harmaalokin selkä on tasaisen vaaleanharmaa, pää ja rinta valkeat, nokka keltainen ja sen kärjessä punainen täplä. Jalkojen väri vaihtelee keltaisesta harmaanpunaiseen, tavallisin väritys vaalean lihanpunainen. Nuori yksilö on valkean-ruskean-harmaankirjava ja vaikeampi tunnistaa muista isoista nuorista lokeista. Nuori lintu saa aikuispuvun vasta noin nelivuotiaana.

Harmaalokin pituus on 55–68 cm, siipien kärkiväli 130–150 cm ja paino 600–1 700 g. Koiras on selvästi naarasta kookkaampi. Koiraan siiven pituus on 410–455 mm ja naaraan 390–425 mm. Koiras painaa keskimäärin 1 245 g ja naaras 860 g. Harmaalokki ja selkälokki ovat lähisukuisia lajeja, joiden suomalaiset alalajit on helppo erottaa toisistaan.

Esiaikuinen, noin kolmevuotias harmaalokki

Harmaalokin soidinääni, kaklatus, poikkeaa kalalokin kaklatuksesta matalampana ja honottavampana. Selkälokin kaklatus on vain hieman korkeampi.

Useita alalajeja tavataan. Etelänharmaalokki, aroharmaalokki ja amerikanharmaalokki on hiljattain erotettu omiksi lajeikseen. Harmaalokkiryhmän taksonomia on epäselvä ja sitä tutkitaan paljon.

Rengastettaessa jalkaan laitetaan teräksinen rengas, jonka koko Suomessa ilmaistaan kirjainkoodilla HT. Tuhansia suomalaisia harmaalokkeja on myös värirengastettu eri värisillä noin 30 mm korkeilla muovisilla putkirenkailla, joihin on kaiverrettu lyhyt kirjain- ja numeroyhdistelmä. Niin sanotun lukurenkaan pystyy kiikarilla tai kaukoputkella lukemaan jopa satojen metrien päästä. Rengastamalla on saatu erittäin paljon tietoa harmaalokin elintavoista, muutosta, pesimäpaikkauskollisuudesta ja iästä.

Lukurengastettu tummaselkäinen harmaalokki

Vanhin suomalainen rengastettu harmaalokki ja vanhin suomalainen rengaslintu on toistaiseksi ollut 32 vuotta 1 kuukausi 16 [2] on samalla Euroopan vanhin harmaalokki.

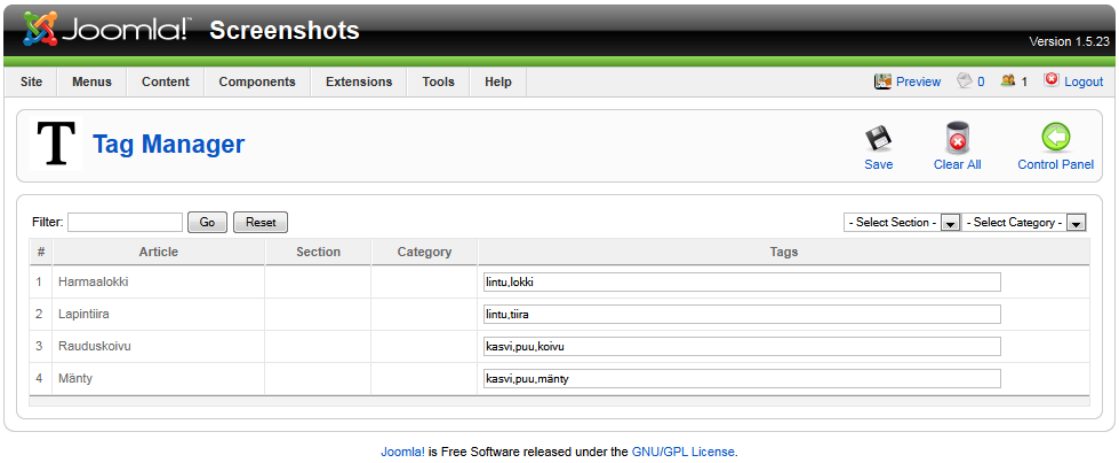
Tags: Lintu Lokki

Last Updated on Monday, 23 May 2011 19:12

Kuva 8. Avainsanat artikkelinäkymässä

Laajennuksen mukana tulee myös neljä erillistä moduulia. Nämä moduulit tarjoavat erilaisia listauksia avainsanoista. Avainsanoja voi listata moduuleissa uutuuden mukaan, yleisyyden mukaan tai satunnaisessa järjestyksessä.

Komponentin hallintapuolen käyttöliittymä tarjoaa kaksi keskeistä näkymää. Ensimmäisessä näistä näkyy listaus kaikista avainsanoista, joita on käytetty. Näkymässä näkyy myös, kuinka monta kertaa avainsanalla olevia artikkeleita on haettu ja kuinka moneen artikkeliin kyseinen avainsana liittyy. Toinen näkymistä listaa kaikki Joomlan artikkelit sekä niihin liitetyt avainsanat. Avainsanat on listattu omaan sarakkeeseen ja niitä voi tarvittaessa lisätä tai poistaa suoraan tekstikentästä. Näiden kahden näkymän lisäksi komponentin hallinnasta löytyy myös komponentin käyttöön liittyviä asetuksia sekä mahdollisuus muokata avainsanalistauksissa käytettyjä CSS-tyylimäärityksiä.



Kuva 9. Avainsanojen hallintanäkymä

3.5 Erityyppisten sisältöjen hakeminen

Vaikka Joomlan hakutoiminto ei suoranaisesti liity avainsanoihin, on sen toimintaperiaate tämän opinnäytetyön kannalta merkittävä. Jotta avainsanoja pystyisi liittämään useisiin eri sisältötyyppeihin, tulisi löytää malli siitä, kuinka eri laajennusten kanssa kommunikointi onnistuisi eroavista teknisistä toteutustavoista huolimatta.

Joomlan oma hakutoiminto toimii juuri tällä tavalla. Hakutoiminto on toteutettu omana komponenttinaan. Kun käyttäjää syöttää hakukomponentille hakusanan, komponentti delegoi hakutoiminnon erillisille liitännäisille, jotka ovat kaikki vastuussa jonkun sisältötyypin etsimisestä. Laajennukset voivat sisällyttää omia sisältöjään Joomlan omaan

hakutoimintoon liitännäisellä, joka osaa etsiä sisältöjä kyseisestä komponentista. Tämän menetelmän etu on se, että uusia hakutoimintoa pystyy laajentamaan muuttamatta itse hakukomponenttia. Jokainen hakuliitännäinen on täysin itsenäinen ja niitä voi lisätä tai ottaa pois käytöstä tarpeen mukaan. On tyypillistä, että komponenttien mukana tulee erillinen hakuliitännäinen.

Ensimmäisessä vaiheessa käyttäjä syöttää hakutermin. Tässä esimerkissä hakutermi on “lintu”. Hakutermi syötetään sivustolla olevaan hakukenttään, josta se ohjautuu varsinaiselle hakukomponentille. Hakukomponentti on yksi Joomlan perusasennuksen mukana tulevista komponenteista.

Toisessa vaiheessa hakukomponentti hakee Joomlasta listauksen asennetuista ja käytössä olevista hakuliitännäisistä. Annettu hakutermi välitetään jokaiselle hakuliitännäiselle.

Kolmannessa vaiheessa hakuliitännäinen etsii tietokannasta komponentin sisältöjä, jotka vastaavat hakutermiä. Tämä vaihe tapahtuu kaikissa hakuliitännäisissä hieman eri tavalla. Jos komponentti on tarkoitettu esimerkiksi videokokoelman organisointiin, voi hakuliitännäinen etsiä kokoelmasta videoita, joiden otsikossa annettu hakutermi esiintyy. Lisäksi hakuliitännäinen voi etsiä hakutermiä esimerkiksi mahdollisista tekstikuvauksista, joita videoon on liitetty.

Neljännessä vaiheessa löydetyt videot kootaan listaksi. Listassa ilmoitetaan jokaisesta löydetystä sisällöstä, nimi kuvaus, päivämäärä, sisällön tyyppi sekä linkki varsinaiseen sisältöön (LeBlanc 2008, 201). Sisällöstä riippumatta kaikki hakuliitännäiset palauttavat löydetyt tulokset samassa formaatissa. Tämä helpottaa hakukomponentin viimeistä vaihetta.

Viidennessä vaiheessa hakukomponentti koostaa kaikkien hakuliitännäisten palauttamat hakutulokset yhdeksi kokonaiseksi listaksi, jonka hakukomponentti näyttää käyttäjälle. Hakutulokset on listausnäkymässä aina ryhmitelty sisältötyyppien mukaan. Toisin sanoen hakutulokset on ryhmitelty sen mukaan, mikä hakuliitännäinen on kyseiset hakutulokset tuottanut.

Olemassa olevat avainsanalaajennukset eivät voi muuttaa kokonaan hakukomponentin toiminnallisuutta. Sen sijaan avainsanalaajennukset voivat tarjota oman hakuliitännäisen. Kyseinen hakuliitännäinen voi esimerkiksi etsiä hakutermiä vastaavia avainsanoja ja listata hakutuloslistaan ne artikkelit, joihin kyseiset avainsanat on liitetty. Tämä menetelmä on hieman monimutkainen eikä se ole lopputuloksen kannalta erityisen toimiva. Tyypillisesti Joomlan hakutoiminto etsii erikseen artikkeleita, ja jos tämän lisäksi käytössä on vielä avainsanojen kautta artikkeleita etsivä hakuliitännäinen, voivat samat artikkelit ilmaantua lopulliseen hakutuloslistaukseen useita kertoja. Tätä varten esimerkiksi Tags for Joomla -laajennus on toteuttanut kokonaan oman hakutoiminnon, jossa artikkeleita voi hakea avainsanojen perusteella. Tässä näkymässä ei kuitenkaan voi hakea muiden komponenttien tarjoamia sisältöjä.

4 Kehitystyö

4.1 Avainsanojen linkittäminen useisiin eri sisältötyyppeihin

Talvella 2010 aloin miettiä ratkaisua avainsanojen linkittämiseen eri sisältötyyppeihin. Koska pelkkä sisältötaulun perusavain ei riitä sisältöjen yksilöimiseen tietokannassa, päädyin aluksi malliin, jossa avainsanojen suhdetauluun tallennettaisiin aina sisällön perusavaimen lisäksi myös taulun nimi, johon sisältö kuuluu. Malli tuntui toimivalta, mutta hieman epäkäytännölliseltä. Sisällön yksilöiminen olisi vaatinut aina tietoa sekä perusavaimesta että taulusta, johon avain kuului. Halusin kehittää mallin, jossa sisältöjä pystyisi yksilöimään yhdellä muuttujalla riippumatta taulusta, johon se oli tallennettu.

4.2 Id-avaruuden osiointi

Tietokannassa rivin id-arvo tallennetaan tyypillisesti 32 bitin tarkkuudella. Tiedon tallentamiseen käytettävissä olevien bittien lukumäärä määrittää sen, kuinka suuria lukuja kyseiseen muuttujaan voidaan tallentaa. Tähän vaikuttaa myös se, halutaanko luvut tallentaa etumerkin kanssa vai ilman. Jos luvut tallennetaan etumerkittöminä, voi muuttuja esittää ainostaan ei-negatiivisia lukuja. Etumerkillisiin muuttujiin voidaan tallentaa

myös negatiivisia lukuja, mutta suurin mahdollinen tallennettava arvo on puolet pienempi, koska yksi luvun esittämiseen käytettävistä biteistä varataan luvun etumerkin esittämiseen.

Tietokannassa tämä tarkoittaa sitä, että suurin mahdollinen id-arvo on 2 147 483 647 (MySQL 2011). Käytännössä tietokantaan id-arvot tallennetaan etumerkittöminä, jolloin suurin mahdollinen id-arvo on kaksinkertainen. PHP-ohjelmointikielessä kaikki muuttujat ovat kuitenkin aina etumerkillisiä, joten vaikka suurempia arvoja voitaisiinkin tallentaa tietokantaan, ei niiden käsittely PHP-ohjelmissa onnistu. Joomla-sivustoissa on kuitenkin äärimmäisen harvinaista, että id-arvot kasvaisivat lähelle maksimiarvoaan. Esimerkiksi vilkkaalla keskustelualueella voivat viestimäärät kasvaa kymmeniin tuhansiin, mutta tuolloinkin yli 99 % mahdollisista id-arvoista jää käyttämättä. Tästä sain idean, jolla voisin ratkaista id-arvoihin liittyvät ongelmat.

Päätin kutsua kaikkien mahdollisten esitettävissä olevien id-arvojen joukkoa id-avaruudeksi. Tämän jälkeen jaoin koko id-avaruuden pienempiin osiin. Jako tapahtui siten, että jokainen tietokannan taulu, johon haluttiin viitata, sai oman osionsa. Tämän jälkeen id-arvot muunnettiin siten, että ne sijaitsisivat taululleen varatussa osiossa.

Id-avaruuden osioiminen perustuu kahteen tekijään. Koska jokainen tietokannan taulu, johon halutaan viitata, vaatii itselleen yhden osion, määrittää id-avaruuden osioiden lukumäärä sen, kuinka moneen eri tietokannan tauluun voidaan viitata. Yksittäisen osion koko taas määrittää sen, kuinka monta riviä yhdessä taulussa voi enintään olla. Id-avaruuden koon ollessa vakio ovat osioiden lukumäärä ja yhden osion koko toisiinsa nähden kääntäen verrannollisia. Toisin sanoen lisättäessä osioiden lukumäärää on yksittäisen osion kokoa pienennettävä samassa suhteessa. Mietittyäni osioiden lukumäärän ja yksittäisen osion koon keskinäisiä suhteita päädyin käyttämään molempiin kompromissin omaisesti lukua 45 000, joka on maksimiarvon neliöjuuren likiarvo. Tämä tarkoittaa että id-avaruus jaettaisiin 45 000 osioon, joista jokaiseen mahtuisi 45 000 id-arvoa. Uskon tämän olevan enemmän kuin riittävä käytännön sovelluksiin.

4.3 Paikallinen ja globaali id-arvo

Pelkkä id-avaruuden osioiminen ei vielä ratkaissut kaikkia ongelmia. Osioinnin lisäksi piti kehittää menetelmä, jolla olemassa olevat id-arvot muunnettaisiin uuden järjestelmän mukaisiksi ja takaisin alkuperäisiin arvoihinsa. Tässä vaiheessa otin käyttöön kaksi uutta termiä. Paikallinen id-arvo tarkoittaa esimerkiksi artikkelin id-arvoa sellaisena kuin se on kyseiselle artikkelille annettu. Globaali id-arvo tarkoittaa paikallisesta id -arvosta muodostettua, muunnettua id-arvoa, joka tallennetaan avainsanojen suhdetauluun. Tämän globaalin id-arvon hyöty on se, että siitä voidaan selvittää sekä sisällön paikallinen id-arvo sekä se, mihin id-avaruuden osioon se kuuluu ja sitä kautta mihin tauluun kyseinen id-arvo kuuluu.

Muunnos paikallisesta id-arvosta globaaliin id-arvoon on yksinkertainen. Ensin on päätettävä, mikä tietokannan taulu saa minkäkin osion id-avaruudesta. Koska eri Joomla-sivustoilla on aina käytössä erilainen kokoonpano asennettuja laajennuksia, olisi tämä jako tehtävä aina sivustokohtaisesti esimerkiksi asennuksen yhteydessä. Jokaisesta osioista on tiedettävä id-arvo, josta osio alkaa, osion koko sekä se, mille taululle kyseinen osio kuuluu. Osioiden jako tallennetaan tietokantaan. Alla oleva PHP-funktio suorittaa muunnoksen paikallisesta id-arvosta globaaliin id-arvoon. Funktion ensimmäinen parametri on muunnettava paikallinen id-arvo ja toinen parametri on osion ensimmäinen id-arvo:

```
function convertToGlobalId($local_id, $base_id) {
        return $base_id + $local_id;
}
```

Muunnos globaalista id-arvosta paikalliseen tapahtuu jakojäännöksen avulla. Funktio palauttaa globaalia id-arvoa vastaavan paikallisen id-arvon. Funktion ensimmäinen parametri on muunnettava globaali id-arvo ja toinen parametri on yksittäisen osion koko.

```
function convertToLocalId($global_id, $size) {
        return $global_id % $size;
}
```

Globaalin id-arvon muuntaminen paikalliseksi ei vielä riitä. Tämän lisäksi täytyy pystyä selvittämään, mihin tauluun kyseinen id-arvo kuuluu. Funktio palauttaa sen osion järjestysnumeron, johon kyseinen globaali id-arvo kuuluu. PHP-standardikirjaston floor-funktio pyöristää jakolaskun tuloksen alaspäin seuraavaan kokonaislukuun:

```
function convertToSection($global_id, $size) {
        return floor($global_id / $size);
}
```

Osion järjestysnumero on vielä muunnettava vastaamaan varsinaista taulun nimeä. Tämän muunnoksen voi tehdä yksinkertaisesti esimerkiksi hyödyntämällä assosiatiivisia taulukoita, joissa järjestysnumerot on yhdistetty niitä vastaavien taulujen nimiin:

```
$sections_array = array(0 => 'jos_content', 1 => 'jos_users', 2 => 'jos_hwdvidsvideos');
$section = convertToSection($global_id, $size);

$table_name = $sections_array[$section];
```

Yllä esitetyt funktiot toimivat, kun kaikki osiot ovat samankokoisia ja sijaitsevat siten, ettei kahden vierekkäisen osion välissä ole sellaisia id-arvoja, jotka eivät kuulu kumpaakaan osioon. Varsinaisessa toteutuksessa käytin vielä kehitetympiä versioita funktioista, joissa eri osiot voivat olla eri kokoisia ja sijaita vapaasti id-avaruudessa edellyttäen, että jokainen id-arvo kuuluu enintään yhteen osioon.

Kun paikalliset id-arvot voidaan muuntaa globaaleiksi, vältytään päällekkäisten id-arvojen ongelmalta. Tämän jälkeen voidaan avainsanojen linkitykseen käytetty suhdetaulu muuttaa sellaiseksi, että siinä sisältöihin viitataan ainoastaan globaaleilla id-arvoilla. Tämän ansioista voidaan avainsanoja linkittää useaan eri sisältötyyppiin. On vain pidettävä huolta, että aina käsiteltäessä avainsanojen linkityksiä tehdään tarvittavat muunnokset.

Tässä esitelty malli toimii yleisellä tasolla. Käytännön sovelluksessa on osion perus id-arvon lisäksi tallennettava myös tietokantataulun nimi, jolle osio kuuluu. Lisäksi on tallennettava sen kentän nimi, joka kyseisessä taulussa pitää sisällään id-arvon.

4.4 Avainsanalaajennuksen rakenne

Vaikka olin ratkaissut teoriatasolla keskeisimmän avainsanojen linkitykseen liittyvän ongelman, olivat avainsanojen linkitykset vain tietokannassa olevaa informaatiota. Tekniikka ei vielä sellaisenaan ollut hyödyllinen, vaan sen ympärille tuli kehittää Joomla-laajennus, joka tarjoaisi käyttöliittymiä ja työkaluja avainsanojen käyttöön. Aloin pohtia avainsanalaajennuksen yleistä rakennetta. Rakenteen tuli olla ennen kaikkea joustava ja helposti laajennettava.

Aloin kehittää laajennukselle mallia, jonka keskiössä on avainsanakomponentti. Tämä komponentti vastaisi kaikkein keskeisimmistä avainsanojen käyttöön liittyvistä toiminnoista. Komponentin lisäksi ajattelin toteuttaa joukon moduuleita ja liitännäisiä, joilla avainsanatoimintoja pystyisi integroimaan sivuston muuhun käyttöliittymään. Vaikka olin kehittänyt mallin, jolla eri tauluissa olevia id-arvoja pystyi käsittelemään tietokannassa, oli laajennukseen kehitettävä systeemi, jolla toimintalogiikkaa pystyisi hienosäätämään jokaiselle komponentille erikseen. Esimerkiksi id-arvojen esittämiseen käytetyn kentän nimi vaihtelee. Monet taulut käyttävät ”id”-nimeä, mutta koska mitään erityistä velvoitetta tähän ei ole, eivät poikkeukset ole harvinaisia. Olemassa olevia variaatioita ovat esimerkiksi ”video_id” tai ”itemId”. Tämän lisäksi monet muut avainsanoihin liittyvät toiminnallisuudet vaativat komponenttikohtaisia asetuksia.

Päädyin hyödyntämään Joomlan hakutoiminnon käyttämää mallia, jossa komponenttikohtaiset toiminnallisuudet on eriytetty omiksi liitännäisikseen. Koska termi ”liitännäinen” oli jo Joomla-ympäristössä käytössä, oli tilalle keksittävä jokin toinen termi. Päädyin käyttämään termiä ”silta” kuvaamaan ohjelmistoa, joka mahdollistaa sen, että avainsanakomponentin linkittää avainsanoja johonkin toiseen komponenttiin. Tämä siis tarkoitti sitä, että jokaiselle komponentille, jonka sisältöihin avainsanoja halutaan linkittää oli oltava olemassa erillinen siltalaajennus.

Riippuvuus siltalaajennuksista tekee avainsanalaajennuksesta käyttäjälle asteen verran monimutkaisempaa, mutta uskon, että järjestelmä tällaisenakin tuo huomattavan parannuksen nykyiseen tilanteeseen. Asiaa helpottaakseni otin tavoitteekseni kehittää laajennuksesta teknisesti sellaisen, että uusien siltalaajennusten tekeminen olisi mahdollisimman vaivatonta.

4.5 Ensimmäinen versio

4.5.1 Ensimmäisen version lähtökohta ja tavoitteet

Projektin ensimmäinen versio syntyi alkuvuodesta 2010. Tuolloin projektilla oli aika lailla eri tavoitteet kuin myöhemmässä vaiheessa. Tuolloin tarkoitus oli kehittää ja tutkia avainsanojen liittämistä videoihin Pixoff-projektissa. Pixoff on suomalaisille harrastelijaelokuvantekijöille tarkoitettu verkkoyhteisö. Olin toiminut kyseisen sivuston kehityksessä ja tutustunut sitä kautta Joomla-sisällönhallintajärjestelmään teknisenä alustana. Aluksi lähdin kehittämään Joomla-laajennusta varta vasten Pixoff-sivuston tarpeisiin. Tästä syystä laajennuksen ensimmäinen versio kulki nimellä “Pixoff Administration”. Vaikka olin tuolloin jo kaavaillut avainsanojen käyttöä paljon laajemminkin, halusin lähteä aluksi toteuttamaan yksinkertaisempaa mallia. Tuolloin en ollut vielä täysin vakuuttunut siitä, kuinka hyvin kehittämäni teoriat toimivat käytännön tasolla. Tämän lisäksi Joomla ohjelmointiympäristönä oli minulle vielä vieras. Tarvitsin käytännön kokemusta ohjelmoinnista Joomla-ympäristössä. Tämän vuoksi päätin asettaa ensimmäiselle version kehittämiselle kaksi tärkeää tavoitetta. Ensimmäinen näistä oli oppia ohjelmoimaan Joomlan komponentille hallintakäyttöliittymä. Toinen tavoitteista oli toteuttaa sovellus, jolla avainsanoja pystyi linkittämään ulkopuolisessa komponentissa olevaan sisältöön.

Pixoff-sivusto hyödynsi videoiden hallinnassa hwdVideoShare-nimistä komponenttia. Olin jonkin verran perehtynyt kyseisen komponentin yleisiin toimintaperiaatteisiin. Komponentti tallentaa varsinaiset videon erillisiin tiedostoihin ja videoon liittyvät metatiedot tietokantaan. Metatiedoista löytyvät muun muassa videon nimi, kuvaus ja käyttä-

jä, joka on videon lisännyt sivustolle. Kyseinen komponentti mahdollisti myös avainsanojen liittämisen videoihin. Käsitykseni mukaan avainsanojen liittäminen sisältöön oli epäkäytännöllistä ja tuotti ongelmia pitkällä tähtäimellä. Halusin keskittyä nimenomaan avainsanojen linkitykseen. Asetin reunaehdoksi kehitystyölle sen, että varsinaiseen hwdVideoShare-komponenttiin ei tulisi tehdä minkäänlaisia muutoksia tai lisäyksiä. Itse tehdyt muutokset komponenttiin ovat kyllä mahdollisia ja sallittuja mutta aiheuttavat yhteensopivuusongelmia, jos komponentin alkuperäinen kehittäjä julkaisee komponentista päivitetyn version.

4.5.2 Ensimmäisen version kehitys ja lopputulos

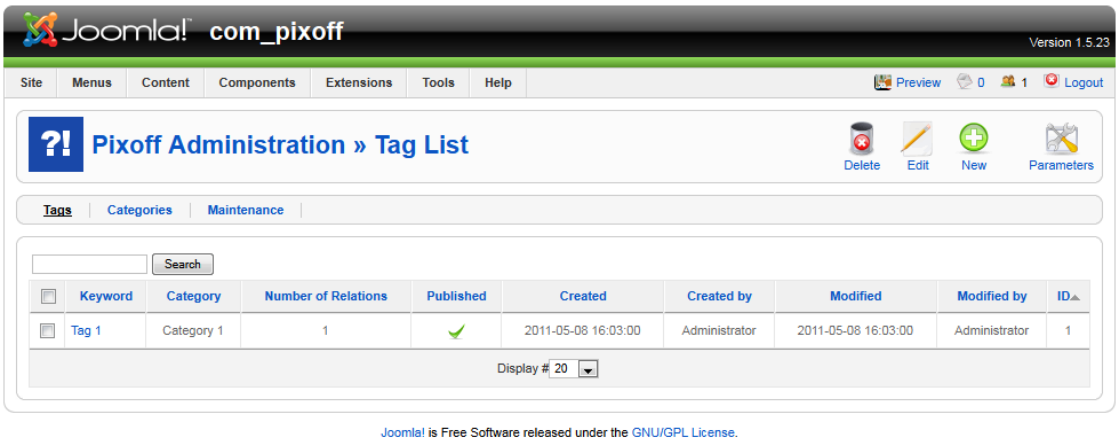
Ensimmäisen version kehitys pohjautui pitkälti internetistä löytyneisiin oppaisiin ja erinäisiin keskustelualueilla käytyihin keskusteluihin. Joomlan ohjelmointirajapintoja koskeva dokumentaatio oli melko hajanaista ja vaihtelevan laatuista. Tästä syystä muiden avoimen lähdekoodin Joomla-laajennusten tutkiminen osoittautui hedelmälliseksi.

Helmikussa 2010 olin saanut kehitettyä komponentin, jonka avulla pystyin luomaan ja muokkaamaan avainsanoja. Lisäksi avainsanoille pystyi määrittämään luokan, johon se kuului. Tämän luokan tarkoitus oli ryhmitellä avainsanoja eri käyttötarkoitusten mukaan. Olin ajatellut, että avainsanojen jakaminen luokkiin olisi yksi laajennuksen kantavista ominaisuuksista. Myöhemmissä kehitysversioissa ne kuitenkin jäivät pois, koska ne eivät olleet välttämättömiä avainsanojen käytön kannalta vaan ennemminkin ylimääräinen toiminnallisuus.

Olin pyrkinyt integroimaan komponentin Joomlaan niin hyvin kuin mahdollista. Pyrin kehitystyössä hyödyntämään Joomlan ohjelmointirajapintojen tarjoamia mahdollisuuksia aina kun mahdollista. Erityisesti käyttöliittymän kohdalla pyrin siihen, että se olisi mahdollisimman yhtenäinen muiden Joomla-komponenttien hallintakäyttöliittymien kanssa. Käyttöliittymien toteutukseen ohjelmointirajapinnoista löytyikin paljon valmiita työkaluja. Tosin useasti näiden työkalujen löytäminen oli hankalaa.

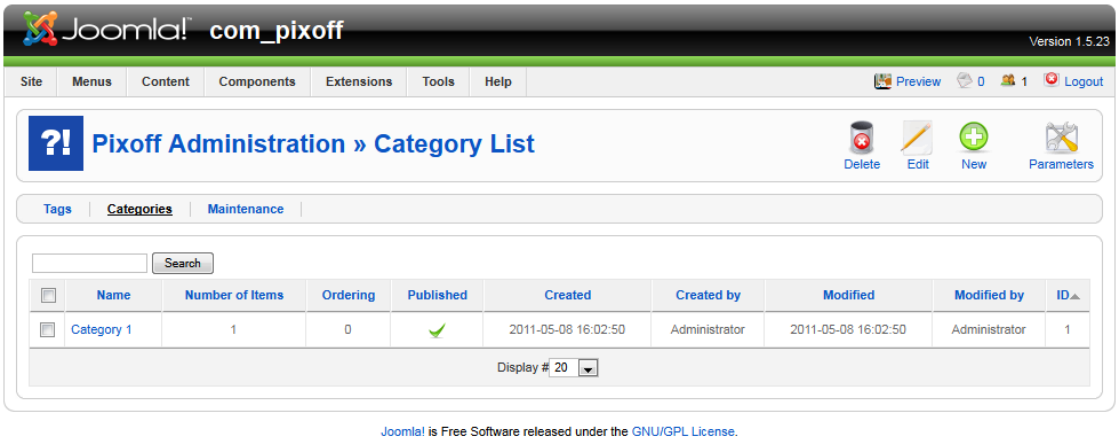
Ensimmäisen version käyttöliittymä jakaantui pääasiassa kolmeen näkymään. Jokainen hallintanäkymä noudatti yleistä linjaa Joomla-komponenttien hallintanäkymissä. Varsinainen hallittava sisältö näkyi sivulla samaan tapaan kuin se oli tallennettu tietokantaan. Sisältöalue jakaantuu sarakkeisiin ja riveihin. Jokainen rivi kuvaa yhtä hallittavaa sisältöä ja jokainen sarake yhtä kyseisen sisällön ominaisuutta. Kulloinkin näkyvissä olevia rivejä voi hallita tarpeen mukaan. Vasemmassa yläkulmassa olevalla search-kentällä voi suorittaa sanahaun, jolloin sisältöalueella näkyvät ainoastaan ne rivit, jotka vastaavat suoritettua sanahakua. Klikkaamalla sarakkeen nimeä voi järjestää näkyvät rivit kyseisen sarakkeen mukaan joko nousevaan tai laskevaan järjestykseen.

Tags-näkymässä näkyi lista kaikista avainsanoista. Jokaisesta avainsanasta näkyi ryhmä, johon kyseinen avainsana kuului sekä se, kuinka moneen videoon kyseinen avainsana oli linkitetty. Lisäksi jokaiselle avainsanalle oli varattu mahdollisuus ottaa se tarvittaessa pois käytöstä. Tämä toiminnallisuus ei kuitenkaan ollut käytännössä toiminnassa. Ensimmäisessä versiossa tallensin avainsanoista myös käyttäjän, joka avainsanan oli luonut ja luontiajankohdan. Tämä toiminnallisuus jäi kuitenkin pois myöhemmissä versioissa tarpeettomana. Uusia avainsanoja pystyi luomaan ylärivin New-painikkeella.



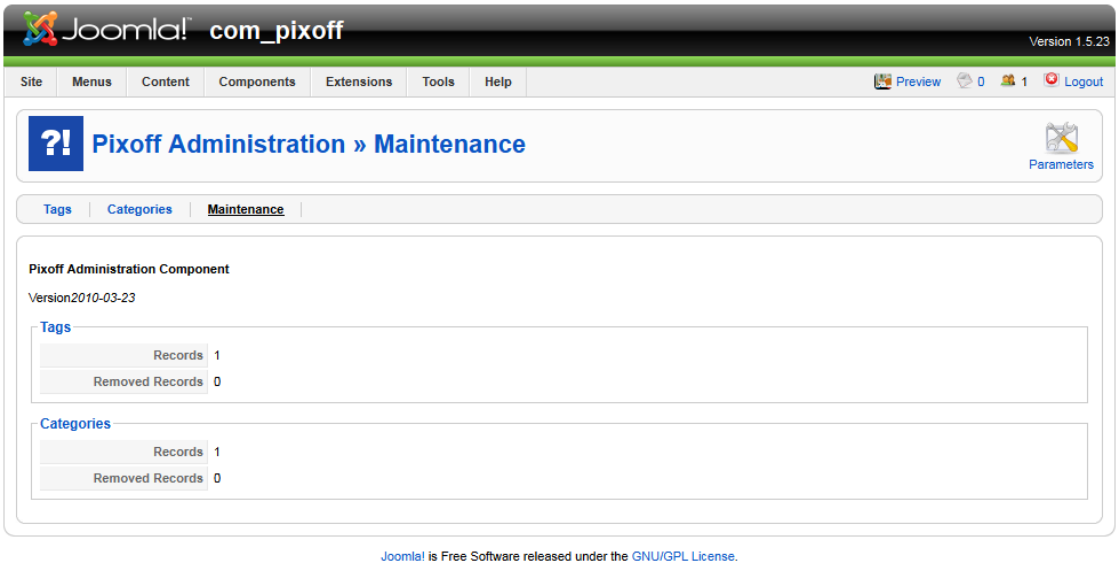
Kuva 10. Avainsanojen hallintanäkymä

Categories-näkymässä näkyivät avainsanojen ryhmät. Jokaisesta ryhmästä näkyi ryhmän nimi sekä kuinka monta avainsanaa kyseisessä ryhmässä on. Myös ryhmille oli käyttöliittymässä varattu mahdollisuus ottaa ryhmä pois käytöstä, mutta tämäkään ei toiminto ei ollut toiminnassa. Lisäksi ryhmistä näkyi kuka käyttäjä ne oli luonut ja milloin. Samoin kuin avainsanoja, pystyi uusia ryhmiä luomaan New-painikkeella.



Kuva 11. Avainsanojen ryhmien hallintanäkymä

Kolmas näkymä tarjosi yleistä tietoa komponentista. Siitä näkyivät komponentin versio sekä avainsanojen ja ryhmien lukumäärät. Olin ajatellut, että tähän näkymään voisi koota myös muuta tilastotietoa, kuten listan eniten käytetyistä avainsanoista.



Kuva 12. Komponentin yleishallintanäkymä

Ensimmäisessä versiossa olin keskittynyt kehittämään pelkästään komponentin hallintakäyttöliittymää. Hallintakäyttöliittymässä käyttäjälle on annettava huomattavasti enemmän toimintoja kuin varsinaisella sivustolla näkyvässä käyttöliittymässä. Tästä syystä olin arvioinut, että hallintakäyttöliittymän kehittäminen olisi haasteellisempaa ja työ-

määrältään suurempaa. Siksi se tuntui luontevalta kohdalta aloittaa kehitystyö. Toinen merkittävä syy siihen, miksi halusin kehittää hallintakäyttöliittymän ensin oli se, että testatakseni sivuston puolen käyttöliittymiä tarvitsisin testausympäristööni videoita sekä niihin liittyviä avainsanoja. Tämän testimateriaalin toteuttaminen ja hallinnointi on huomattavasti helpompaa, jos käytössä on edes kehitysvaiheessa oleva hallintakäyttöliittymä komponenttiin.

Komponentin ensimmäisen version kehitys päättyi huhtikuussa 2010. Vaikka komponentti ei tuolloin ollut vielä valmis, olin saavuttanut sen kehitystyölle asettamani tavoitteet. Olin oppinut huomattavan paljon Joomlasta ohjelmointiympäristönä ja saanut kehitettyä halintakäyttöliittymän avainsanakomponentille.

4.6 Toinen versio

4.6.1 Toisen version lähtökohta ja tavoitteet

Kesän 2010 aikana hahmottelin avainsanalaajennuksen jatkokehitystä. Elokuussa 2010 aloitin toisen version kehittämisen. Toinen versio oli ensimmäistä kunnianhimoisempi. Päätin luopua kytköksistä Pixoff-projektiin ja yksittäisen sivuston tarpeiden tyydyttämisen sijaan halusin kehittää paljon yleispätevämmän ratkaisun avainsanojen käyttöön. Luovuin Pixoff-termin käyttämisestä komponentin nimessä ja annoin projektille työnimen “Adam”. Uusi versio komponentista pohjautui edellisenä keväänä kehittämääni ohjelmistoon. Tavoitteenani oli kuitenkin toteuttaa edelliseen versioon merkittäviä muutoksia. Näistä merkittävin oli siltalaajennusten toteuttaminen käytännössä. Käyttöliittymien suhteen en asettanut erityisiä tavoitteita. Nykyiset hallintakäyttöliittymät olivat riittävät komponentin toiminnan testaamiseen käytännössä.

Siltalaajennukset tuli toteuttaa siten, että ne olisivat kaikki itsenäisiä ja toisistaan riippumattomia. Käyttäjän tulisi pystyä kytkemään komponenttiin siltalaajennuksia vapaasti ja tarvittaessa ottamaan pois käytöstä. Tarvittaessa siltalaajennuksia piti pystyä myös päivittämään. Käytännössä tämä tarkoitti sitä, että komponentti piti ohjelmoida siten, ettei käytettävissä olevien siltalaajennusten lukumäärää tiedetty etukäteen. Vastaavanlaisia

laajennuksia on monissa peruskäyttäjillekin tutuissa ohjelmistoissa. Esimerkiksi internetselainten liitännäiset tai vaikka käyttöjärjestelmään asennettavat laitteistoajurit noudattavat tätä samaa periaatetta. Onnekseni PHP-kieli on luonteeltaan sellainen, että tällaisten vaihtuvien ohjelmaosien toteuttaminen onnistuu kohtalaisen helposti.

Siltalaajennusten lisäksi otin ohjelmistossa käyttöön termin ”entiteetti” kuvaamaan linkityksen kohteena olevia sisältöobjekteja. Mielestäni termin abstrakti luonne korostaa sitä, ettei avainsanojen linkityksen kohteella ole varsinaisesti merkitystä. Riittää, että linkityksen kohde voidaan yksilöidä tietokannassa. Ohjelman toiminnan kannalta on yhdentekevää, onko linkityksen kohde kuva, video, käyttäjä, lokimerkintä tai vaikkapa toinen avainsana.

Toista versiota kehittäessäni aloin nimetä eri kehitysasteita. Aloin nimetä kehitysasteita yleisen käytännön mukaan alpha- ja beta-versioihin. Ensimmäinen alpha-versioista valmistui toukokuussa 2010 ja sitä seuraavat noin kuukauden välein.

4.6.2 Entiteetit ja siltalaajennukset

Vaikka olin suunnitellut, kuinka avainsanakomponentti toimisi yleisellä tasolla, oli käytännön toteutus silti haastavaa. Siltalaajennukset ja entiteetit muodostivat koko komponentin toiminnan ytimen. Ensimmäisessä versiossa halusin saada toimimaan jotakin yksinkertaista, joka kuitenkin hyödyntäisi siltalaajennusten periaatetta. Päätin lähteä toteuttamaan entiteettilistaa, joka koostuisi kolmen eri komponentin sisällöistä. Ensimmäiseksi komponentiksi valitsin hwdVideoShare-komponentin, jota olin jo aikaisemmin hyödyntänyt kehitystyössä. Kaksi muuta olivat Joomlan omia komponentteja. Toinen niistä hallinnoi artikkeleita ja toinen rekisteröityneitä käyttäjiä. Listassa oli tarkoitus näkyä jokaisen sisältöobjektin nimi.

Oliopohjaisen ohjelmoinnin menetelmiä noudattaen määrittelin ensin kaikille siltalaajennuksille yhteisen rajapinnan. Tämä rajapinta käytännössä määritteli sen, kuinka komponentti kommunikoi siltalaajennuksen kanssa. Jokainen siltalaajennus tallennettiin erillisiin tiedostoihin ja keskitettiin yhteen hakemistoon. Lähdekoodin lisäksi jokaiselle sil-

talaajennukselle tallennettiin tietokantaan tieto siitä, mikä id-avaruuden osio sille oli varattu. Tämä mahdollisti sen, että id-avaruuden osiojako oli aina sivustokohtainen. Avainsanakomponentti huolehti tietokantaan tallennettujen tietojen välittämisestä siltalaajennukselle.

Entiteettilistan muodostaminen koostui kolmesta eri vaiheesta. Ensimmäisessä vaiheessa komponentti latasi ja otti käyttöön kaikki siltalaajennukset. Toisessa vaiheessa komponentti pyysi jokaista siltalaajennusta muodostamaan sen sisältöobjekteja vastaavat entiteetit. Kolmannessa vaiheessa kaikki siltalaajennusten toimittamat entiteettilistat liitettiin yhteen ja lista näytettiin käyttäjälle.

Ensimmäisessä versiossa entiteetti oli ohjelman suorituksen aikana muodostuva esitys varsinaisesta sisältöobjektista. Entiteettiin tallennettiin varsinaista sisältöobjektia kuvaava globaali id-arvo. Koska globaalit id-arvot eivät sisällä ihmiskäyttäjälle mielekästä informaatiota, päätin tallentaa jokaiseen entiteettiin tekstimuotoisen kuvauksen sisältöobjektista, jota se edustaa. Se mitä kuvaus käytännössä sisältää, on kunkin siltalaajennuksen vastuulla. Esimerkiksi artikkeleista kuvauskenttään tallennetaan artikkelin nimi.

4.6.3 Toinen alpha-versio

Ensimmäisessä versiossa olin saanut muodostettua entiteettilistan kolmen eri komponentin sisällöistä ja tulostettua tuon listan yhdessä komponentin hallintanäkymistä. Ongelma kuitenkin oli, että ohjelmalogiikka, jolla entiteettilista muodostettiin, oli sidoksissa käyttöliittymäkoodiin. Toisin sanoen jos halusin saada saman listan tai samankaltaisen listan näkymään jossakin muualla, olisi minun kirjoitettava uudelleen sama ohjelmakoodi, joka latasi siltalaajennukset ja haki niiltä listan entiteeteistä. Tällainen koodin uudelleenkirjoittaminen olisi työlästä ja epäkäytännöllistä.

Aloin kehittää seuraavaa alpha-versiota, jossa merkittävin muutos oli se, kuinka saisin entiteetteihin ja siltalaajennuksiin liittyvän logiikan erotettua käyttöliittymistä. Tavoitteena oli kehittää kaikille käyttöliittymille yhteinen ohjelmistokirjasto, jonka avulla pääsisi käsiksi siltalaajennuksiin ja entiteetteihin. Tämän kirjaston kehitys tarkoitti käytän-

nössä olemassa olevan koodin uudelleenjärjestelyä ja joidenkin ohjelman osien uudelleenkirjoittamista. Tästä syystä toiseen alpha-versioon ei tullut varsinaisia uusia ominaisuuksia. Koska tässä vaiheessa ohjelman kehitystä erilaiset virhetilanteet olivat tyypillisiä, päätin alkaa käyttää lokitiedostoa. Tarkoitus oli, että aina kun ohjelmassa tunnistettaisiin virhetilanne tai jollakin muulla tavalla normaalista poikkeava tilanne, siitä tulisi merkintä lokitiedostoon. Lokitiedoston käyttöön oli Joomlan ohjelmointikirjastoissa valmis pohja, jota hyödynsin.

4.6.4 Kolmas alpha-versio

Kolmannessa versiossa kehitin ensimmäisen sivustolla näkyvän käyttöliittymän avainsanoille. Käyttöliittymä oli sivustolle sijoitettava moduuli, joka näytti sillä hetkellä näkyvään sisältöobjektin liittyvät avainsanat. Moduuli tunnisti aktiivisena olevan komponentin ja pyysi ohjelmistokirjastolta komponenttia vastaavan siltalaajennuksen. Tämän jälkeen siltalaajennus selvitti, oliko kyseisen näkymän yhteydessä mahdollista näyttää avainsanoja. Näkymän tuli olla sellainen, että sisältöobjektin pystyi yksilöimään. Esimerkiksi yksittäistä artikkelia luettaessa siltalaajennus pystyi yksilöimään artikkelin, mutta jos näkymässä oli listaus ryhmän artikkeleista, ei yksittäisen artikkelin yksilöiminen onnistunut. Jos sisältöobjekti pystyttiin yksilöimään, moduuli käytti ohjelmistokirjastoa ladatakseen siihen linkitetyt avainsanat ja tulosti ne listana käyttäjälle.

Aluksi moduuli vain näytti käyttäjälle tietoa. Olin kuitenkin suunnitellut moduulista interaktiivista. Halusin, että moduulissa näkyviä avainsanoja klikkaamalla moduulin sisältö vaihtuisi ja moduulissa näkyisi esimerkiksi, mitä muita sisältöobjekteja oli linkitetty kyseiseen avainsanaan. Verkkosivujen toiminta perustuu sivupyyntöihin ja niihin saataviin vastauksiin. Tyypillisesti käyttäjän klikatessa linkkiä selain lähettää sivupyynnön ja korvaa sillä hetkellä näkyvän sivun sivupyynnön vastauksena saatavalla sivulla. Tämä aiheuttaa ongelmia tavassa, jolla halusin moduulin toimivan esimerkiksi tilanteessa, jossa käyttäjä katsoo sivulla näkyvää videota. Jos käyttäjä klikkaisi moduulissa näkyvää linkkiä, aiheuttaisi se uuden sivupyynnön ja vaikka sivupyynnön vastauksena olisi edelleen se sama sivu, jolla käyttäjä äsken oli, olisi videotiedoston toisto katkennut ja käyttäjä joutuisi aloittamaan videon katsomisen alusta.

Ajax-tekniikalla selain saadaan lähettämään sivupyyntö palvelimelle ilman, että vastauksena tuleva sivupyyntö korvaa sillä hetkellä näkyvän sivun. Näin voidaan vastauksena saava sivupyyntö näyttää esimerkiksi jossakin tietyssä kohdassa verkkosivua. Ajax-tekniikkaa käytetään nykyään kasvavasti. Muun muassa Facebook ja Googlen sähköpostipalvelu Gmail hyödyntävät käyttöliittymissään Ajax-tekniikkaa huomattavasti.

Tiesin, että saadakseni moduulin toimimaan haluamallani tavalla, oli minunkin toteutettava se hyödyntäen Ajax-tekniikkaa. Ongelmana Ajax-tekniikassa on, että se monimutkaistaa ohjelman toimintaperiaattetta. Moduuli ei muodostakaan itse sitä HTML-koodia, joka kyseisesä moduulissa näkyy, vaan se muodostaa koodin, jolla näytettävä HTML-koodi haetaan jostain muualta. Tässä tilanteessa kun moduulissa ei näkynyt muuta kuin kyseiseen sisältöobjektiin liittyvät avainsanat, voi Ajax-tekniikan käyttö tuntua kohtuuttoman monimutkaiselta. Halusin kuitenkin jo tässä vaiheessa varautua myöhempiä kehitysversioita varten.

4.6.5 Neljäs alpha-versio

Neljännen alpha-version merkittävimpiin muutoksiin kuului tietoturvaan liittyvien kysymyksiin puuttuminen. Tietoturvaan liittyvät kysymykset sinällään ovat aivan liian laaja aihe tässä opinnäytetyössä käsiteltäväksi. Verkkosovelluksissa tyypillisin tietoturvaan liittyvä ongelma on käyttäjältä tulevien syötteiden puutteellinen tai kokonaan puuttuva tarkastus. Tämän seurauksena haavoittuvalle verkkosovellukselle on mahdollista antaa sellaisia syötteitä, jotka aiheuttavat joko virhetilanteita sovelluksen toiminnassa tai pahemmassa tapauksessa saavat sovelluksen toimimaan epätoivotulla tavalla. Hyökkääjä voi esimerkiksi saada ohjelmiston tulostamaan näytölle kaikkien järjestelmän käyttäjien käyttäjätunnukset ja salasanat.

Tietoturvakysymykset koskevat kaikkia verkkosovelluksia, myös Joomlaa ja sen laajennuksia. Verkkosovellusten ohjelmoijan tulee olla erityisen hulellinen aina, kun ohjelmassa käsitellään tietoa, joka on alun perin peräisin käyttäjältä. Syötteiden tarkistus käytännössä tarkoittaa esimerkiksi sitä, että jokaiselle käyttäjältä tulevalle syötteelle

määritellään, minkä tyyppistä tietoa se on ja vielä parempi, jos kyseiselle tiedolle voidaan määrittää myös sallittujen arvojen joukko. Tämän jälkeen aina, kun ohjelmassa käsitellään syötettä, joka on peräisin käyttäjältä, tarkistetaan aluksi, onko syöte oikean tyypistä ja onko sen arvo sallituissa rajoissa. Jos syöte ei täytä sille määriteltyjä kriteerejä on ohjelman hylättävä se. Tilanteesta riippuen ohjelma voi korvata hylätyn arvon jollakin oletusarvolla tai jos se ei ole mahdollista, lopettaa ohjelman suoritus siihen.

Toisessa alpha-versiossa olin toteuttanut syötteiden tarkistukset käyttöliittymien yhteyteen. Tämä toimintatapa tarkoitti sitä, että syötteiden tarkistuksia tulisi varsin moneen eri kohtaan ohjelmistoa. Samalla kun olin siirtynyt käyttämään sisäistä ohjelmistokirjastoa siltalaajennusten ja entiteettien hallintaan, aloin miettiä, voisinko toteuttaa syötteiden tarkistukset samalla tavalla keskitetysti. Tämän seurauksena päätin siirtyä suoraan seuraavaan alpha-versioon, jossa toteuttaisin syötteiden tarkistuksen keskitetysti ohjelmistokirjastossa.

Neljännen version merkittävimpiä muutoksia oli keskitetty syötteen tarkistus. Syötteen tarkistus aiheutti kuitenkin ongelmia. Monesti järjestelmä hylkäsi kelvollisia syötteitä ja tällä tavalla haittasi komponentin toimintaa. Uskoin kuitenkin vahvasti siihen, että pidemmällä tähtäimellä keskitetty syötteen tarkistus osoittautuisi sekä luotettavammaksi että tehokkaammaksi tavaksi kuin syötteiden tarkistaminen erikseen jokaista käyttökertaa varten.

Lisäksi neljännessä versiossa aloin kehittellä mahdollisuutta lisätä aktiiviseen sisältöobjektiin avainsanoja edellisessä versiossa kehittämäni moduulin kautta. Tämä olisi moduulille ensimmäinen käyttötarkoitus, jossa Ajax-tekniikasta olisi käytännön hyötyä. Tarkoitus oli lisätä moduuliin tekstikenttä ja linkki. Käyttäjä voisi kirjoittaa tekstikenttään haluamansa avainsanan ja linkkiä klikkaamalla moduuli lähettäisi komponentille sivupyynnön, jonka seurauksena komponentti lisäisi kyseisen avainsanan moduuliin.

Ajax-tekniikan käyttö mahdollisti tiettyjä käytettävyyttä parantavia toimintoja mutta samalla teki ohjelman toiminnasta merkittävästi monimutkaisempaa. Nyt sivuston toimintaa ei ohjelmoitu pelkästään palvelimella ja valmista HTML-dokumenttia toimitettu käyttäjälle. HTML-dokumentin lisäksi käytössä oli pieniä JavaScript-ohjelmointikielisiä

sovelluksia, jotka suoritettiin selaimessa. Nämä JavaScript-sovellukset vastasivat Ajax-kyselyiden lähettämisestä selaimelle ja vastauksen saapuessa sen sijoittamisesta oikeaan kohtaan verkkosivulla. Kokonaisuutena tähän asti kehittämäni avainsanalaajennus oli melko hajanainen. Kokonaisuuden hallinta tuotti itsellenikin ongelmia.

4.6.6 Neljännen alpha-version jälkeen

Neljännen version jälkeen ohjelmiston kehityksessä oli taas tauko. Seuraava versio syntyi huhtikuussa 2010. Olin tässä vaiheessa päättänyt siirtyä käyttämään ohjelmiston versiohallintaa. Se on erillinen ohjelmisto, joka pitää kirjaa ohjelmistoon tehdyistä muutoksista. Tämän seurauksena olin luopunut aikaisemmin käyttämistäni nimetyistä versioista. Versionhallintaohjelmisto antoi ohjelmistolle juoksevan versionumeron, joka kasvoi aina, kun versionhallintaan tallennettiin uusi versio ohjelmistosta.

Uudessa versiossa tuli myös rakenteellisia muutoksia. Ensimmäisestä alpha-versiosta asti entiteetit olivat olemassa vain ohjelmiston ajon aikana ja siltalaajennukset loivat niitä tarpeen mukaan. Tämä aiheutti ohjelmoinnin kannalta muutamia ongelmia. Koska entiteetit koottiin yhteen eri siltalaajennuksilta, oli esimerkiksi niiden järjestäminen haluttuun järjestykseen tehtävä ohjelmallisesti. Toisin oli esimerkiksi avainsanojen kanssa, jotka sijaitsivat kaikki samassa taulussa tietokannassa. Jos avainsanat halusi tulostaa tietyssä järjestyksessä, onnistui se kätevästi kertomalla haluttu järjestys tietokannalle, joka huolehti tuloslistan haluttuun järjestykseen. Tätä mahdollisuutta ei kuitenkaan ollut entiteettien kohdalla. Jos entiteettejä olisi todella paljon, voisi niiden järjesteleminen PHP-sovelluksessa olla sekä laskennallisesti raskasta sekä vaatia paljon muistia. Tietokantasovellus oli sitä paitsi optimoitu juuri tuollaisia tehtäviä varten.

Aloin pohtia entiteettien tallentamista tietokantaan. Toisin sanoen entiteetit eivät enää olisi olemassa ainoastaan ohjelman suorituksen aikana, vaan tiedot entiteeteistä olisi tallennettu tietokantaan. Tämä mahdollistaisi esimerkiksi entiteettien lajittelun haluttuun järjestykseen suoraan tietokannassa. Entiteettien tallentaminen tietokantaan toi kuitenkin mukanaan uuden ongelman. Siltalaajennusten olisi nyt pidettävä huolta, että tietokantaan tallennetut entiteettilistat olivat ajan tasalla. Jos käyttäjä esimerkiksi loi uuden

artikkelin, oli siltalaajennuksen huomattava uuden artikkelin ilmestyminen järjestelmään ja lisättävä uutta artikkelia vastaava entiteetti tietokantaan. Päivityksistä oli pidettävä huolta myös silloin, kun järjestelmästä poistettiin sisältöobjekteja. Nämä eivät kuitenkaan olleet ylitsepääsemättömiä ongelmia. Lopputuloksena komponentin sisäinen ohjelmistokirjasto keveni jonkin verran ja muuttui yksinkertaisemmaksi.

Myös syötteiden tarkistus ohjelmistokirjastossa koki muutoksia. Vanha mekanismi, joka koostui yhdestä pitkästä funktiosta, jakautui useampaan pienempään funktioon. Lisäksi syötteiden tarkistus muuttui automaattisemmaksi. Koodiin tarvitsi nyt enää määritellä, minkä tyyppistä sisältöä mikäkin syöte sai sisältää ja mekanismi hoiti loput. Aikaisemmin tarkistukset jokaiselle syötteelle oli kirjoitettava erikseen.

Myös hallintapuolen käyttöliittymät kokivat muutoksia. Hallintanäkymiin tuli mahdollisuus muun muassa rajata näkyviä sisältöjä entistä monipuolisemmin. Lisäksi näkymien ulkoasuun tuli joitakin visuaalisia ja käytettävyyteen liittyviä parannuksia.

5 Pohdinta

5.1 Yhteenveto

Tätä kirjoittaessani toukokuussa 2011 oli kulunut yli vuosi siitä, kun aloitin suunnittelemaan avainsanalaajennusta. Kuluneen vuoden aikana projekti on edennyt välillä hyvinkin intensiivisesti ja välillä seissyt kuukausia paikallaan. Työtä ei voi vielä kutsua valmiiksi, mutta toisaalta olen saanut toteutettua visiot, joista projekti lähti alun perin liikkeelle. Tähän mennessä hankalin aihe on ollut kaikkein keskeisimpien toiminnallisuuksien hiominen. Toisaalta kaikkein työläintä on käyttöliittymien toteuttaminen.

5.2 Julkaiseminen

Tarkoitus on kehittää avainsanalaajennuksesta julkaisukelpoinen. Laajennus aiotaan julkaista GPL-lisenssillä. Tähän olen päätynyt muutamasta syystä. Ensimmäinen ja kaik-

kein painavin syy on, että Joomlan virallinen jakelukanava laajennuksille ei enää ota vastaan muuta kuin sellaisia laajennuksia, jotka on julkaistu GPL-lisenssillä. Koska Joomlan virallinen jakelukanava on ainut yleisesti käytetty jakelukanava, on kyseisen kanavan ehtoja noudatettava. Toinen syy GPL-lisenssin käyttöön on ideologinen. Joomla on julkaistu kyseisellä lisenssillä samoin kaikki Joomlan laajennukset, joita olen tutkinut ja joista olen oppinut. Myös kaikki käyttämäni palvelinohjelmistot ovat GPL-lisenoituja puhumattakaan ohjelmointiin ja versionhallintaan käyttämistä ohjelmistoista. Myös tämä opinnäytetyö on kirjoitettu avoimen lähdekoodin ohjelmistolla. Tästä syystä GPL-lisenssin käyttö tuntuu luonnolliselta vaihtoehdolta.

5.3 Jatkokehitys

Laajennuksen perustoiminnallisuus on osoittautunut toimivaksi. Jatkokehitystä tarvitaan, jotta laajennus olisi hyödyllisempi työkalu. Kriittisin jatkokehityksen tarve on käyttöliittymissä.

Käyttöliittymiä kehitettäessä on aina muistettava, että ne toimivat osana jotain isompaa kokonaisuutta. Tässä tapauksessa laajennuksen käyttöliittymä voi esimerkiksi tarkoittaa artikkelinäkymän lopussa olevaa avainsanalistaa. Toisin sanoen käyttöliittymä ei saa olla itsetarkoitus, vaan sen on mahdollisimman hyvin istuttava ympäristöönsä. Tämän tavoitteen saavuttaminen onnistuu ehkä helpoiten pyrkimällä yksinkertaisuuteen eli toisin sanoen pyrkimällä näyttämään käyttöliittymissä vain se mikä on todella oleellista. Joomlassa hyvä puoli on se, että laajennuksiin voidaan määritellä käyttöliittymiä koskevia asetuksia, jolloin sivuston ylläpitäjä voi hienosäätää käyttöliittymiä paremmin oman sivustonsa tarpeisiin.

Toinen tärkeä jatkokehitysala on siltalaajennusten kehittäminen. Mitä useampaan komponenttiin on olemassa siltalaajennus sitä hyödyllisempi koko kehittämäni avainsanalaajennus on. Tässä tosin alussa kannattaa keskittyä nimenomaan suosittujen komponenttien tukemiseen.

5.4 Levitys

Vaikka avainsanalaajennuksen kehitys on lähtenyt henkilökohtaisista motiiveista, olen tekoprosessin aikana vakuuttunut kasvavasti siitä, että toimivaa laajennusta on pyrittävä levittämään mahdollisimman laajasti. Käytännössä tämä tarkoittaa laajennuksen kehittämistä siihen kuntoon, että se voidaan julkaista Joomlan virallisella laajennuksia tarjoavalla sivustolla. Ennen tätä vaihetta koko laajennus on tosin saatava sellaiseen kuntoon, että se selvästi erottuu edukseen muista avainsanoihin liittyvistä laajennuksista.

Levitykseen ja laajennuksen jatkokehitykseen liittyy vielä avoimia kysymyksiä. Kehitystyön aikana Joomlan perusversio on siirtynyt seuraavaan versioon 1.6. Avainsanalaajennukseni on kuitenkin kehitetty versiolle 1.5. Laajennuksen siirtäminen versiolle 1.6 ei periaatteessa ole kohtuuton urakka mutta herättää kysymyksen siitä, kuinka työlästä on ylläpitää samaa ohjelmaa kahdelle eri sisällönhallintajärjestelmän versiolle.

Lisäksi olen pohtinut levityksen ainakin osittaista kaupallistamista. Avoimen lähdekoodin ohjelmistoissa ohjelmistojen tai ohjelmistolisenssien myyminen ei onnistu yhtä yksinkertaisesti kuin suljetummassa ohjelmistokehityksessä. Avoimen lähdekoodin ohjelmistosta itsestään ei voi periä minkäänlaisia käyttömaksuja. Monet muut Joomla-laajennusten kehittäjät ovatkin rakentaneet sinällään ilmaisen laajennuksen ympärille maksullisia sisältöjä, jotka eivät kuulu avoimen lähdekoodin lisenssin piiriin. Esimerkiksi maksulliset dokumentaatiot tai maksulliset tukipalvelut ovat melko yleisiä. Olen pohtinut myös avoimen lähdekoodin lisenssistä luopumista kaupallisen toiminnan edistämiseksi. Tämän vaihtoehdon keskeisin ongelma on, ettei Joomlan virallinen listaussivusto hyväksy enää muita kuin avoimen lähdekoodin laajennuksia. Toisin sanoen luopumalla avoimen lähdekoodin lisenssistä laajennukseni menettäisi käytännössä monopoliasemassa olevan levityskanavan.

5.5 Visiot

Konkreettisen ohjelmointityön rinnalla olen kehitellyt myös abstrakteja visioita siihen, mihin avainsanoilla pyritään ja miksi niitä pitäisi käyttää. Olen tässä työssä käsitellyt jo

sitä, kuinka avainsanoja voidaan käyttää nykyisiä menetelmiä tehokkaampaan tiedon organisointiin. Avainsanat kuitenkin liittyvät paljon laajempaan käsitykseen metatiedosta ja siitä, kuinka kuilua sisällön ja tiedon välillä voidaan kaventaa. Nämä käsitteet ovat avainasemassa semanttisen webin kehitystyössä. Semanttinen web itsessään on liian laaja aihe tässä käsiteltäväksi, mutta itse näen siinä olevan kysymys nimenomaan tiedon rikastamisella metatiedolla ja erinäisten sovellusten kehittämisellä.

Tässä työssä käsiteltyjen sinällään monimutkaisilta kuulostavien teknologioiden tarkoitus on yhä digitalisoituvammassa maailmassa rakentaa tietojärjestelmiä, jotka paremmin mallintaisivat sitä todellisuutta, jossa elämme. Tämä vaatii ymmärrystä teknologiasta mutta samalla ymmärrystä ihmisistä sekä siitä, kuinka ihmiset hahmottavat ympäröivää maailmaa. Vaaditaan pohtimista siitä, kuinka me ihmiset luokittelemme ja määrittelemme asioita, joita koemme ja näemme. Kehitystyön aikana huomasin, että suurimmat avainsanoihin liittyvät ongelmat eivät ole teknologiassa vaan siinä, kuinka avainsanoja käytetään. Varsinkin laajoissa sisältökokonaisuuksissa niin kutsutuilla avainsanojen käytön pelisäännöillä on merkitystä sen suhteen, kuinka tehokkaasti avainsanoja hyödyntävät teknologiat toimivat.

Lähteet

Boiko, B. 2005. Content Management Bible. Indianapolis, IN: Wiley Publishing.

Hovi, A., Huotari, J. & Lahdenmäki, T. 2003. Tietokantojen suunnittelu & indeksointi. Jyväskylä: Docendo Finland.

Hutchinson, J. 2010. The history of Joomla!: An in-depth chat with CMS core developer, Andrew Eddie. IDG Communications. http://www.computerworld.com.au/article/354247/history_joomla_an_in-depth_chat_cms_core_developer_andrew_eddie/. 13.9.2011.

Lee, M. 2007. FSF releases the GNU General Public License, version 3. Free Software Foundation. http://www.fsf.org/news/gplv3_launched. 13.9.2011.

Morville, P. & Rosenfeld, L. 2006. Information Architecture for the World Wide Web. Sebastopol, CA: O'Reilly Media.

Perens, B. 1999. Open Sources: Voices of the Open Source Revolution. Sebastopol, CA: O'Reilly Media, Inc.

Shreves, R. 2010. Open Source CMS Market Share Report. Kerobekan: Water & Stone.

The Internet Society. 1999. Hypertext Transfer Protocol. Fremont, CA: Internet Engineering Task Force.

Joomla. 2011. Joomla! Extensions Directory. Open Source Matters. http://extensions.joomla.org/. 13.9.2011.

Joomla! Documentation. 2011a. Framework/1.5. Open Source Matters, Inc. http://docs.joomla.org/Framework/1.5. 13.9.2011.

LeBlanc J. L. 2008. Learning Joomla! 1.5 Extension Development. Birmingham: Pact Publishing.

MySQL. 2011. MySQL 5.0 Reference Manual :: 10.2 Numeric Types. Oracle Corporation. http://dev.mysql.com/doc/refman/5.0/en/numeric-types.html. 13.9.2011.

W3C. 2011. HTML & CSS. World Wide Web Consortium. http://www.w3.org/standards/webdesign/htmlcss. 13.9.2011.

W3C. 2004. Architecture of the World Wide Web, Volume One. World Wide Web Consortium. http://www.w3.org/TR/webarch/. 13.9.2011.

W3C. 1999. The global structure of an HTML document. World Wide Web Consortium. http://www.w3.org/TR/html40/struct/global.html. 13.9.2011.