Lauri Kumpulainen

Sisällönhallintajärjestelmän moduulikehitys

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Mediatekniikan koulutusohjelma

Insinöörityö

12.5.2015

Metropolia

Tekijä Otsikko Sivumäärä Aika	Lauri Kumpulainen Drupal-moduulikehitys 34 sivua + 2 liitettä 12.5.2015
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Mediatekniikka
Suuntautumisvaihtoehto	Digitaalinen media
Ohjaajat	Drupal-arkkitehti Roni Kantis Lehtori Ilkka Kylmäniemi

Insinöörityön tarkoitus oli luoda käyttöliittymä, jonka avulla verkkoselaimen aloitussivulta päästäisiin siirtymään suoraan käyttäjän suosimille sivustoille ilman, että jouduttaisiin käyttämään hakukoneita. Useimmilla verkkosivustoilla on omanlaisensa hakuparametri, joka näkyy selaimen osoitekentässä haun tuloksen yhteydessä. Hakuparametrien tallentaminen verkkosivuston tietokantaan mahdollistaisi personoidun aloitussivun, jonka avulla käyttäjä voisi tehdä hakuja suoraan suosimiltaan verkkosivustoilta ilman, että tarvitsisi erikseen siirtyä halutulle sivustolle tekemään haku.

Tavoite oli valmistaa Drupal-moduuli, jonka avulla saadaan luotua dynaamisia linkkejä halutuille verkkosivustoille. Moduulin luoman sisältötyypin kautta tietokantaan tallennetaan kunkin halutun sivuston nimi ja hakuparametri. Moduulin luoman piensovelluksen avulla käyttäjän syöttämä hakusana yhdistetään tallennettuun hakuparametriin, ja näin syntyy dynaaminen linkki. Moduuli toteutettiin käyttämällä Drupalin sisäänrakennettuja hook-funktioita, joiden avulla päästään käsiksi sisällönhallintajärjestelmän rajapinnan eri osiin.

Drupal on avoimeen lähdekoodiin perustuva sisällönhallintajärjestelmä, jonka ympärille on muodostunut hyvin aktiivinen maailmanlaajuinen yhteisö. Drupalin toiminta perustuu moduuleihin, jotka tuovat järjestelmällä rakennettaviin verkkopalveluihin lisäominaisuuksia ja laajentavat Drupalin toiminnallisuutta. Moduulit ovat ladattavissa ilmaiseksi Drupal-yhteisön verkkosivuilta, eikä niiden käyttöä tai muokkaamista ole mitenkään rajoitettu. Räätälöity moduulikehitystyö tulee kysymykseen siinä vaiheessa, kun tiettyä ominaisuutta ei saada jo olemassa olevien moduulien avulla toteutettua.

Syntynyt Custom queries -moduuli auttoi ymmärtämään Drupalin rakennetta ja sisällönhallintajärjestelmän taustalla vaikuttavia toimintoja. Projektissa käytetyt hook-funktiot luovat perustan useimmille olemassa oleville moduuleille, ja niiden tunteminen on siksi tärkeää.

Moduuli on tarkoitus ottaa käyttöön erillisellä verkkosivustolla, johon kirjautunut käyttäjä voi tallentaa omia hakuparametreja. Asettamalla palvelun selaimen aloitussivuksi käyttäjä pääsee nopeammin käsiksi haluamaansa informaatioon, ja verkkosivuston kävijämäärän kasvaessa sivuston ylläpito voidaan rahoittaa mainostajilta saaduilla tuloilla.

Avainsanat	Drupal, moduulikehitys, PHP, HTML, JavaScript, jQuery

Author Title Number of Pages Date	Lauri Kumpulainen Content management system module development 34 pages + 2 appendices 12 May 2015
Degree	Bachelor of Engineering
Degree Programme	Media Technology
Specialisation option	Digital Media
Instructors	Roni Kantis, Drupal Architect Ilkka Kylmäniemi, Senior Lecturer

The aim of this final year project was to create a user interface, which would allow direct access to the favorite websites of the user, without the need of using a search engine. Most websites have a unique search parameter, which is shown in the address bar after a search has been done. If these search parameters could be saved into a database, they could be harnessed by a web browser start page. With this start page, a user could create custom queries directly to his/her favorite website. This would leave out the need to access the page just to make the query.

Another purpose of the project was to build a custom Drupal module, which allows the creation of dynamic hyperlinks to external websites. The data concerning the websites' search queries is saved via a custom content type within the module. The module comes with a widget, which combines user input with the search queries stored in the database, and thus creates a dynamic external link. The module was crafted using hooks, built-in API functions in Drupal, which allow access to Drupal core.

Drupal is an open-source Content Management System, which is maintained by an active international community. The functionality of Drupal is based on modules, which extend the properties of the very system in numerous ways. Modules are downloadable free of charge via the Drupal community website, and there are no restrictions on the use and modification of these modules. Custom module development is needed when a certain type of functionality cannot be acquired with the existing modules.

The module development project helped to understand the structure and functions of Drupal. The hooks used in this project are commonly used in Drupal modules, and thus are a crucial part of the learning curve concerning Drupal content management system.

The module is meant to be a part of a web page, in which a registered user may store their preferred search queries. By setting the service as the start page in their web browser, users gain quick access to the information they seek. As the number of registered users increase, the site maintenance could be funded with advertisement income.

Keywords	Drupal, PHP, HTML, JavaScript, jQuery

Sisällys

1 Johdanto

Sisällöltään monipuolisissa verkkopalveluissa on omat hakuparametrit, jotka riippuvat sivuston rakenteesta ja haettavasta sisällöstä. Joissakin verkkoselaimissa, kuten Google Chromessa, voidaan hallita hakukoneita luomalla pikanäppäinkomentoja halutuille sivustoille. Esimerkiksi Youtube-sivuston hakuparametri saadaan poimittua selaimen osoiteriviltä haetun videon edeltävästä merkkijonosta. Parametri voidaan kopioida ja tallentaa selaimen hakukoneisiin näppäinkomennon y alle. Tällöin selaimen osoitekenttään kirjoitetun y-kirjaimen jälkeinen välilyönti laukaisee komennon, jonka kautta selain ymmärtää hakea syötetyn tekstin halutulta sivulta käyttäen samalla oikeanlaista hakuparametria.

Tämän insinöörityön tarkoitus on kehittää räätälöity moduuli, jolla lisätään Drupal-sivustolle toiminnallisuus dynaamisten hyperlinkkien muodossa. Moduulin avulla käyttäjä voi tallentaa sivuston tietokantaan edellisenkaltaisia omia hakukyselyjä, jotka sitten aktivoidaan sivulle luodun lomakkeen kautta. Tämän toiminnallisuuden tarve on peräisin ajatuksesta sivuuttaa hakukoneet tapauksessa, jossa käyttäjä tunnistaa yleisimmin käyttämänsä hakumenettelyt eri sivustoilla. Sivustot sisältävät useimmissa tapauksissa yksinkertaisen hakuparametrin, joita hyödyntämällä päästään siirtymään haluttuun palveluun ja hakutulokseen nopeammin.

Edelleen ajatusta hakukoneiden sivuuttamisesta pidemmälle viemällä voidaan moduulin avulla kehittää verkkosivusto, johon kirjautumalla käyttäjä voi luoda omia hakuparametreja usein vierailemiltaan sivuilta ja näin luoda oman näköisensä aloitussivun verkkoselaimeensa. Sivuston tallentaminen verkkoselaimen aloitussivuksi antaa käyttäjälle yksilöidyn näkymän selaimen avautuessa. Insinöörityössä ei keskitytä varsinaisen verkkopalvelun toteuttamiseen, jolloin tulisi ottaa huomioon muun muassa sivuston käyttäjistä aiheutuva palvelimen kuormitus sekä käyttäjien tallentamiin dynaamisiin linkkeihin liittyvät mahdolliset haittavaikutukset, kuten takaisin-linkkaus arveluttavaa materiaalia sisältävältä sivustolta. Sivuston on kuitenkin tarkoitus toimia käyttöesimerkkinä moduulin tarjoamista mahdollisuuksista.

Drupal on avoimeen lähdekoodiin perustuva sisällönhallintajärjestelmä, jota käytetään nykyaikaisten, dynaamisten verkkosovellusten toteuttamiseen. Lähtökohtaisesti Drupa-

lia voidaan käyttää kaikenkokoisten sivustojen pohjana, mutta varsinkin laajoissa kokonaisuuksissa Drupalin käyttäminen tulee kysymykseen siinä vaiheessa, kun sisällön hallinta ja sen esittäminen sivustolla halutaan toteuttaa helposti ja toimivaksi. Drupalin toiminta perustuu moduuleihin, jotka laajentavat sivuston toiminnallisuutta halutulla tavalla. [1, s. 1–2.]

Yleisesti ottaen yhden moduulin tarkoitus on tuoda Drupal-sovellukseen jokin tietty lisätoiminto, kuten esimerkiksi mahdollistaa kuvakentän lisääminen artikkelia luotaessa. Drupalin hallinnassa keskitytään suurimmalta osin juuri erilaisten moduulien hallinnan opetteluun ja niiden tarjoamien erilaisten ominaisuuksien kokeiluun. Samalla toiminnolla varustettuja moduuleita saattaa olla lukuisia erilaisia, ja on käytettävä aikaa kokeiltaessa mikä vaihtoehdoista on toimiva ja kehittäjän omia senhetkisiä tarpeita parhaiten vastaava. Drupalin parissa työskentelevät kehittäjät muodostavat yhteisön, jolla on suuri merkitys Drupalin toiminnassa. Yhteisön jäsenet luovat moduuleita ja teemoja vapaaseen käyttöön sekä järjestävät erilaisia tapahtumia, joissa jaetaan tietoa ja verkostoidutaan aktiivisesti. [1, s. 2–3.]

Verkkosivut ovat lähes poikkeuksetta ainutlaatuisia projekteja, jotka vaativat toiminnallisuuden ja sisällön kannalta yksilöityjä ratkaisuja. Asiakkaan tai sivuston kehittäjän omat vaatimukset sivuston toiminnalle vaativat hyvin usein räätälöityjä ominaisuuksia. Sisällönhallintajärjestelmät tarjoavat verkkosivujen kehittämiseen monipuolisen ympäristön, ja useimmissa tapauksissa ne sisältävät valmiita rakennuspalikoita, joiden avulla sivustolle saadaan tuotua erilaisia elementtejä hyvin helposti. [2, s. 1.]

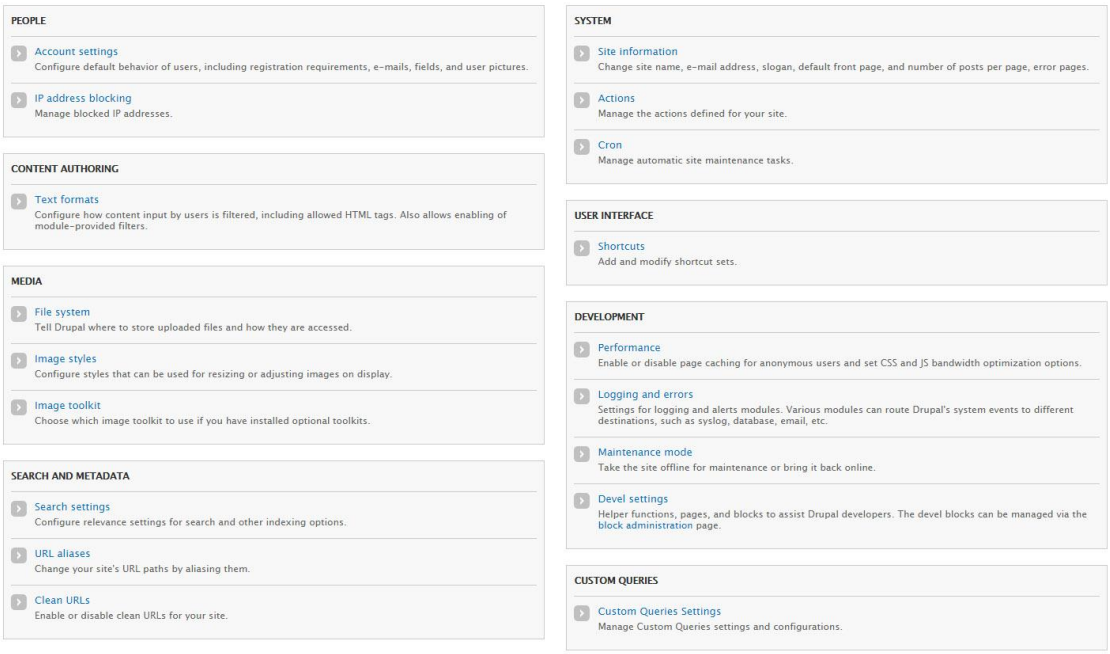
Drupal-sivuston ulkoasu rakennetaan erillisen teeman avulla, ja toiminnallisuus sekä lisäominaisuudet, toisin sanoen mainitut rakennuspalikat, ovat nimeltään moduuleita. Valmiiden, jo olemassa olevien moduulien ja teemojen käyttäminen on sallittua myös asiakastöissä käytetyn vapaan lähdekoodin ansiosta, ja jopa suotavaa työmäärän minimoimiseksi. Elementtien muokkaaminen projektikohtaisiin tarpeisiin sopivaksi onnistuu joissakin tapauksissa, kun taas toisinaan vaaditaan täysin uudentyyppistä toiminnallisuutta. Tämä tarkoittaa moduulikehitystyötä.

2 Moduulikehityksen taustat

Sisällönhallintajärjestelmät (engl. Content Management System, myöh. CMS) ovat syntyneet staattisten verkkosivustojen ylläpidon helpottamiseksi. Aiemmin suurten verkkosivustojen ylläpitäminen oli hankalaa, sillä kaikki muutokset oli tehtävä suoraan HTML-koodiin, josta ylläpitäjällä oli oltava aiempaa kokemusta ja paljon osaamista. Sittemmin yleistyneet CMS-alustat, kuten WordPress, Hippo, Contentful ja Drupal ovat sen sijaan lähes kenen tahansa ylläpidettävissä, ja osaaminen on keskittynyt verkkopalveluiden ja -sivustojen rakentamiseen ja kehitystyöhön. [3, s. 9–10.]

Drupal on myös järjestelmänä enemmän kuin sisällönhallintajärjestelmä. Sen avulla voidaan kyllä rakentaa verkkosovelluksia pienistä sivustoista suuriin kokonaisuuksiin, mutta sen todellisia mahdollisuuksia kuvaa parhaiten luokittelu Content Management Framework, CMF. Toisin sanoen, Drupalilla voidaan periaatteessa rakentaa räätälöityjä sisällönhallintajärjestelmiä. Verkkopalvelun rakentaminen Drupal-järjestelmällä on ennen kaikkea pitkäkatseista, sillä sivustoa voidaan uusia komponentteja lisäämällä laajentaa lähes rajattomasti. [4, s. 8.]

Drupalin räätälöitävyys on yksi sen vahvuuksista verrattuna muihin sisällönhallintajärjestelmiin. Projektikohtainen muokkaaminen vaatii verkkosovelluksen koosta riippumatta aikaa, ja suuret kehitysprojektit vievät kokonaisia vuosia. Yritykset myyvät asiakkailleen osaamista rakentamalla verkkosovelluksia, jotka asiakas omistaa valmistumisen jälkeen. Kuten kuva 1 osoittaa, Drupalin käyttöliittymä on loppuasiakkaan kannalta hyvin monimutkainen. Vaikka käyttöliittymää muutettaisiinkin projektiin sopivaksi ja yksinkertaisempaan muotoon, palveluntarjoajan on useissa tapauksissa tarjottava toimivan tuotteen lisäksi myös koulutusta myöhempää sisällönhallintaa varten. Verkkopalvelun julkistamisen jälkeen on lisäksi mahdollista sopia ylläpitoon ja jatkokehitykseen liittyvistä seikoista ja tällä tapaa jatkaa asiakassuhdetta, vaikka itse projekti tulisikin valmiiksi. [3, s. 11, 13.]



Kuva 1. Drupalin konfiguraatiosivun käyttöliittymä.

2.1 Drupalin kehitysmenetelmistä

Drupal-sivuston rakentaminen asiakkaan vaatimusten mukaan voi olla pitkä prosessi, ja ajallisesti tämä voi tarkoittaa vuosien tiivistä yhteistyötä toimeksiantajan ja palveluntarjoajan välillä. Asiakkaan tarve sivuston tai verkkopalvelun, kuten intranetin, tilaamiseen liittyy oman toimialan kilpailuasetelman parantamiseen, eikä asiakkaalla ole näin lainkaan tietoa verkkoteknologioista. Tämä saattaa aiheuttaa osapuolten välille vaikeuksia kommunikoida tehokkaasti. On olemassa tapauksia, joissa asiakas ei ole osannut kuvailla tarpeeksi hyvin omia vaatimuksiaan, eikä lopputuote ollut valmistuessaan vastannut lainkaan toimeksiantajan alkuperäistä näkemystä. [5, s. 6–8, 21–24.]

Sivuston tilaavalla osapuolella ei kuitenkaan välttämättä tarvitse olla verkkoteknologioita koskevaa tietoa saadakseen palveluntarjoajalta vaatimuksien mukaisen tuotteen. Tämä voidaan varmistaa käyttämällä oikeanlaista kehitysmenetelmää. Aiemmin yleisesti käytetyn niin sanotun vesiputousmallin mukaisesti etenevä projekti on hyvin täsmällinen, ja aikataulu määritellään projektin sisältämien työvaiheiden mukaan etukäteen hyvin tarkasti. Työvaiheiden järjestys verkkosivuston kehittämisessä on hyvin ennakoitavissa, mutta kun edellisen työvaiheen on oltava valmis ennen seuraavan aloit-

tamista, ei työn laadusta ole välttämättä takuita. Vesiputousmalli on nimensä mukainen prosessi, jossa työvaiheet virtaavat jatkuvasti eteenpäin kohti lopullista määräaikaa. Projektin alussa asiakas asettaa vaatimukset, ja vasta lopuksi voidaan todeta, onko lopputulos halutunlainen. [5, s. 13–15.]

Nykyisin yleistynyt vaihtoehto perinteiselle vesiputousmallille on menetelmä, jota kutsutaan ketteriksi kehitysmenetelmiksi. Ennalta määrätyn projektiaikataulun sijaan palveluntarjoaja kehittää tilattua tuotetta lyhyissä sykleissä. Varmistuakseen prosessin tehokkuudesta verkkopalvelua kehittävä osapuoli pitää asiakkaan ajan tasalla heti projektin alusta lähtien. Ensimmäinen toimiva versio rakennettavasta sivustosta saatetaan asiakkaan kokeiltavaksi hyvin pian projektin aloitusajankohdan jälkeen. Tämän jälkeen asiakas voi tehdä huomautuksia tuotteeseen, ja tarvittavat muutokset on mahdollista toteuttaa nopealla aikataululla. Ketterät kehitysmenetelmät lisäävät avoimuutta asiakkaan ja palveluntarjoajan välillä, kun ongelmakohdista voidaan keskustella projektin ollessa vielä kesken. Niin kutsuttuihin sprintteihin, eli lyhyiden päämäärien jaksoihin, jaettu projekti on todennäköisemmin asiakkaan toiveiden mukainen, kun osapuolten välinen kommunikaatio keskittyy kerrallaan pienempiin osa-alueisiin sivustoon liittyen. [5, s. 13–14.]

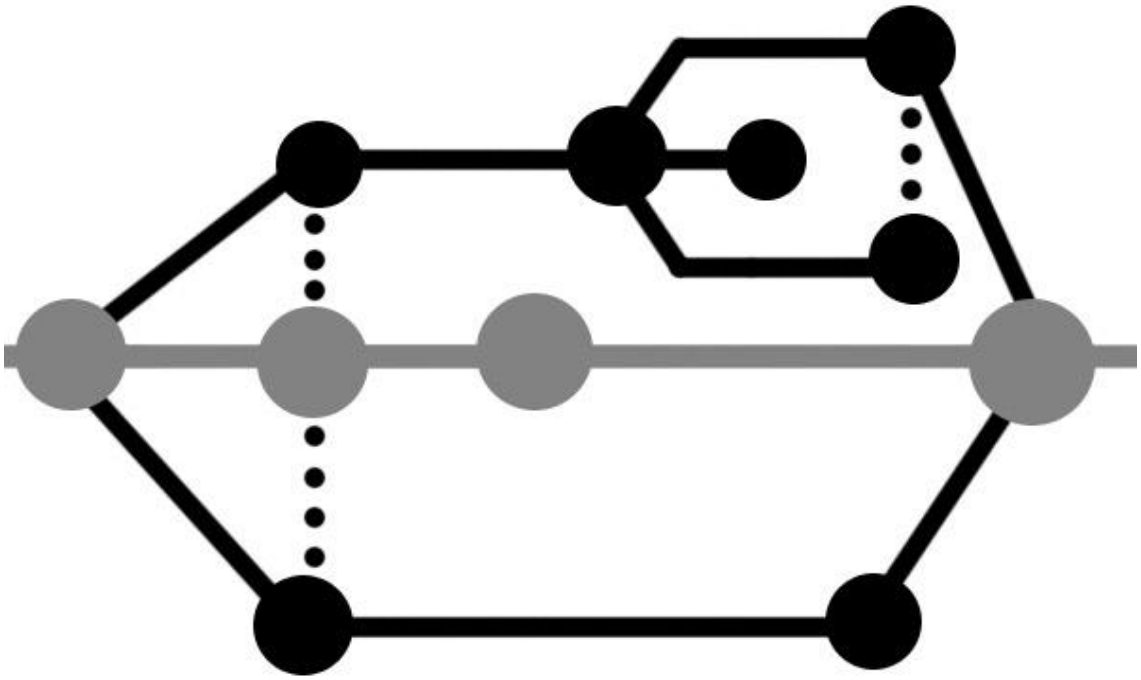
2.2 Moduulikehityksestä

Asiakkaan tarpeista riippuen Drupal-sivusto voidaan saada rakennettua valmiiden, jo olemassa olevien, järjestelmän mukana tulleiden tai muiden kehittäjien julkaisemien moduulien avulla. Drupal-kehittäjän tietämys ja osaaminen sisällönhallintajärjestelmään liittyen auttavat kääntämään sivuston vaatimukset verkkoteknologioiden mahdollistamaksi toiminnallisuudeksi ja löytämään kutakin ominaisuutta vastaavat moduulit. Asiakkaan vaatimukset ovat kuitenkin epärationaalisia olemassa olevien ratkaisuiden suhteen, sillä ne eivät pohjaudu tietämykseen tai olettamuksiin verkkoteknologioiden tarjoamista mahdollisuuksista, vaan ne ovat parhaimmillaan uusia ajatuksia, jotka vievät osaltaan verkkopalveluiden kehitystä eteenpäin. Olemassa olevien moduulien muokkaaminen tai uuden moduulin luominen tulisi nähdä haasteena, ratkaisuna asiakkaan esittämään ongelmaan. [5, s. 43–44, 56–57.]

2.3 Versionhallinta moduulikehityksessä

Yksi ketterän kehityksen tärkeimmistä työkaluista on versionhallinta. Se tarkoittaa sanamukaisesti järjestelmää, jolla saadaan pidettyä kehitysprojektin jokainen vaihe tallennettuna esimerkiksi erilliseen pilvipalveluun. Versionhallinnan avulla projekti voidaan jakaa sekä temporaalisesti että dimensionaalisesti haaroihin, mikä mahdollistaa monen kehittäjän välisen tehtävänjaon ja yhtäaikaisen kehitystyön. [4, s. 21; 6, s. 1.]

Temporaalinen, eli ajallinen, versionhallinta on eräänlainen lokikirja, jossa projektin kaikki vaiheet ovat nähtävissä kronologisesti kehitystyön aloitushetkestä projektin valmistumiseen. Perinteiseen tiedostojen korvaamiseen perustuvaan varmuuskopiointimenetelmään verrattaessa dimensionaalinen, toisin sanoen eri ulottuvuuksien välinen, versionhallinta mahdollistaa kehitysprojektin jakautumisen moniin eri haaroihin, joissa samat lähtökohdat ovat saaneet erilaisia lopputuloksia, kuten kuvan 2 harmaasta poikkeavat haarat. Dimensionaalisen versionhallinnan eri haaroja voidaan myöhemmin yhdistää, ja näin saadaan projektin eri osa-alueet koottua yhdeksi toimivaksi kokonaisuudeksi. [6, s. 43; 7, s. 58, 80.]



Kuva 2. Versionhallinnan puurakenne: harmaan pääjanan ympärille versoo oksia.

Versionhallintajärjestelmät, kuten Github ja BitBucket, tarjoavat käyttöliittymän, jonka avulla saadaan synkronoitua projektin eri vaiheet ja osa-alueet lokaalin kehitysympäristön ja pilvivaraston välillä. Paikallisesti tehdyt muutokset liitetään versionhallintapalvelun uuteen haaraan, ja näin projektin jokainen versio saadaan samalla varmuuskopioitua. Tarvittaessa aiempiin versioihin voidaan myöhemmin palata, ja tämän vuoksi on tärkeää sisällyttää jokaiseen luotuun versioon kattava selvitys tehdyistä muutoksista. [6, s. 131–132.]

Versionhallinnan lisäksi myös projektin lähdekoodi on kommentoitava yksityiskohtaisesti, jotta sen sisältö olisi mahdollisimman vaivattomasti ymmärrettävissä. Avoimen lähdekoodin ansiosta muut kehittäjät voivat myöhemmin jatkaa tai laajentaa projektia omien tarpeiden mukaan, ja hyvin kommentoitu koodi selkeällä kielellä auttaa heitä pääsemään alkuun ja löytämään tarvitsemansa osiot koodin seasta. Tästä syystä itseään varten tehdyt huomiot ja kommentit on kirjoitettava siten, että kuka tahansa voi niistä hyötyä. [4, s. 35–36.]

2.4 Verkkoteknologioista moduulikehityksessä

Drupalin rajapinta mahdollistaa mihin tahansa käytössä olevaan verkkoteknologiaan pohjautuvien moduulien luomisen, ja kehittäjän on hallittava järjestelmän hyödyntämistä skriptauskielistä ainakin CSS (Cascading Style Sheets) ja JavaScript. Reaaliaikaista toiminnallisuutta saadaan moduuliin edelleen AJAXin avulla. Itse moduulin rakenne koodataan Drupalin rakentamiseen käytetyllä PHP (PHP: Hypertext Preprocessor) -kielellä. [4, s. 10.]

Verkkosivustoissa on kaksi eri puolta: front-end, joka tarkoittaa selainpuolella tapahtuvaa HTML-koodin suorittamista, sekä back-end, joka mahdollistaa palvelimen ja selaimen välisen tiedonsiirron esimerkiksi PHP-kielen avulla. Sivuston ulkoasu tuodaan selaimesta riippuen tietyllä tavalla käyttäjän nähtäväksi, ja CSS-tyylitiedostossa on tärkeää ottaa huomioon mahdollisimman monta eri valmistajan selainta sekä tukea eri versioita. JavaScript-kielen avulla saadaan käyttäjän toimintaa seuraamalla tuotua sivustolle toiminnallisuutta. Tieto on kuitenkin olemassa vain selaimen ylläpitämän selaussession ajan, eikä dataa saada tallennettua ilman palvelinpuolen ohjelmointia ja jonkinlaista tietokantajärjestelmää. Tämä on eräs syy, miksi Drupal on rakennettu pal-

velinpuolen back-end-teknologian päälle hyödyntämällä PHP-ohjelmointikieltä. [4, s. 11.]

Drupalin ydin (engl. core) ja moduulit suorittavat PHP-koodin. PHP:n sisällä haetaan tarvittavat CSS- ja JavaScript-tiedostot, ja ne ladataan selaimeen. Drupalissa PHP:n lisäksi toinen tärkeä rakenteellinen elementti on SQL (Structured Query Language) -pohjainen tietokanta. Tietokantaan tallennetaan lähes kaikki tieto sivustolta, muun muassa tiedot käyttäjistä ja sivustolla käytetyn median hakemistorakenteesta. Tietokannan ansiosta Drupal-sivustoa voidaan laajentaa lähes rajattomasti, ja tietokannan varmuuskopioinnilla saadaan sivuston toiminta turvattua ja palautettua mahdollisten käyttökatkosten tai palvelimen vaihtamisen jälkeen. Eräs Drupalin merkittävä piirre on sen yhteensopivuus monenlaisten eri tietokantatyyppien kanssa. [4, s. 9–10; 8, s. 26.]

Tietokannan toimiminen abstraktiotasolla mahdollistaa eri lähestymistapoja rakennettaessa verkkopalveluja. Riippumatta siitä, onko tietokantapalvelimella käytössä esimerkiksi MySQL vai SQLLite, tietokantakyselyt ovat Drupal-kehittäjän näkökulmasta samat. Tämä mahdollistaa myös sivuston siirtämisen toisentyyppisen tietokannan piiriin. Moduulikehityksessä ei tulisi käyttää tavallista SQL-syntaksia, vaan Drupalin omia funktioita, jotka välittävät tietokantakutsut abstraktiotason läpi oikeassa muodossa kussakin tietokantatyypissä. Tauluja luodaan, muokataan ja poistetaan Drupalin Schema-rajapinnan välityksellä, jonka vaatima syntaksi on suunniteltu huomattavasti yksinkertaisemmaksi kuin perinteinen SQL-skripti. Myös tallennettu tieto haetaan Drupalin omien tietokantakyselyjen kautta. Tietokannan abstraktiotaso mahdollistaa tietokannan jakamisen monen eri palvelimen kesken, mikä on varsinkin suurten kävijämäärien suosimilla verkkosivustoilla käytännöllinen vaihtoehto. [1, s. 360–361.]

Drupalissa voidaan hyödyntää monenlaisia verkkoteknologioita, ja eri moduulit käyttötarkoituksesta riippuen käyttävät niiden tuomia mahdollisuuksia hyödyksi. Asynkronisoitu JavaScript ja XML (AJAX) on sivuston taustalla vaikuttava tekniikka. XML on pitkään käytössä ollut merkkikieli, jonka avulla saadaan luotua sisältöön sääntöjä ihmisen ja koneen luettavaan muotoon. AJAX on moderni sovellus, joka hyödyntää XML-tyyppistä tiedonvälitystä. Siinä tieto kootaan selkeään muotoon ennen sen lähetystä. XML:n avulla voidaan välittää informaatiota riippumatta siitä, onko kyseessä musiikkia, soittolistoja, tai video. XML-merkkikieli tuleekin olemaan joustavuutensa ansiosta hyödyllinen työkalu myös tulevaisuudessa. AJAXin avulla tietoa saadaan haettua palvelimelta se-

laimeen huomaamattomasti, eikä käyttökokemus katkea. Tämä on käytännöllinen vaihtoehto sille, että tiedon hakeminen vaatisi koko sivuston päivittämistä ja uudelleen lataamista. Nyt sisällön osan muuttuessa päivitetään vain muuttunut osa. Esimerkiksi Facebookin reaaliaikaisesti päivittyvä syöte perustuu AJAXin hyödyntämiseen. [4, s. 305–306; 9, s. 57.]

Drupalissa moduulien tuoman funktionaalisuuden vastapainona toimivat teemat, joiden kautta rakennetaan sivuston ulkoasu. Verkkosivuja selataan nykyisin pääosin muilla kuin perinteisillä työasemilla, ja tämän vuoksi eräs kehitettävän sivuston perusvaatimuksista on niin kutsuttu responsiivisuus. Sivuston ulkoasu on suunniteltava mukautumaan eri näytönkokoihin ja laitetyyppeihin ilman, että sivuston toiminnallisuus kärsii. Laitteiden kirjo on vakiinnuttanut niin sanottua ”mobile first” -ajattelua, jonka mukaan verkkosivut tulisi suunnitella ensisijaisesti mobiililaitteita ajatellen ja lisätä sitten näytön koon kasvaessa sivuston näyttävyyttä ja toiminnallisuutta. [10.]

Drupal-sivusto tuodaan käyttäjälle HTML-muodossa, ja selainten kehittäjätyökalujen avulla saadaan sivuston teemaa rakennettaessa selville Drupalin muodostamien id-tunnusten ja luokkien (engl. class) nimet erityyppisille elementeille. Saatua tietoa voidaan hyödyntää CSS-tyylien avulla liittämällä halutut tyylit elementtien tunnuksiin. Teemaan liittyy ulkoasun lisäksi toiminnallisuus, ja monet apumoduulit keskittyvätkin JavaScript- ja jQuery-toiminnallisuuden tuomiseen Drupal-sovellukseen. Tämä voidaan toteuttaa kahdella eri tavalla – joko antamalla kehittäjälle työkalu oman JavaScript-koodin ajamiseen sivustolla tai vaihtoehtoisesti luomalla suoraan jokin valmis toiminnallisuus kehittäjän käyttöön. Jälkimmäisestä tavasta esimerkkinä voisi toimia tyylikäs ponnahdusikkuna kalenterinäkymään tiedon syöttämisen yhteydessä päivämääräkenttään. [11, s. 36–38; 3, s. 25; 4, s. 288.]

2.5 Moduulijärjestelmä

Drupalin rakenne on modulaarinen, mikä tarkoittaa kokonaisuuden jakautumista yksittäisiin osiin. Kukin näistä osista, moduuleista, mahdollistaa jonkin toiminnallisuuden ja näin laajentaa Drupalin käytettävyyttä. Moduulit voidaan tarvittaessa korvata toisella, ja toisaalta niiden määrää on mahdollista lisätä lähes rajattomasti. Kussakin tapauksessa käytettävien moduulien määrä ja tarkoitus on suunniteltava tarkkaan, sillä kaikki käytössä olevat moduulit vaikuttavat lopullisen sivuston latausnopeuteen. Huonosti suunniteltu Drupal-sivusto on raskas käyttää, ja mitä suurempi kävijämäärä sivustolla on, sitä paremmin sen rakenne tulisi optimoida myös moduulien kannalta. Moduulin sisältämä koodi suoritetaan jokaisella latauksella, ja näin ollen myös itse moduulin tulisi olla mahdollisimman hyvin koodattu. [1, s. 315–316; 3, s. 50.]

2.6 Sisältötyyppi

Sisältötyyppi (engl. content type) tarkoittaa erikseen määriteltyä tapaa luoda yhtenäistä sisältöä. Esimerkiksi blogikirjoitus on oma sisältötyyppinsä, jossa on otsikko ja leipätekstialue. Sivustoa suunniteltaessa on tärkeää huomioida samankaltaisen sisällön keskittäminen yhden sisältötyypin alaisuuteen, sillä tämän avulla saadaan kunkintyyppinen sisältö eristettyä muusta erilaisissa yhteyksissä, kuten esimerkiksi tuotelistausta tehtäessä. Sisältötyyppejä käytetään vaihtelevan sisällön luomiseen, ja Drupalin tarjoamat oletussisältötyypit eivät aina täytä sisällölle asetettuja kriteerejä. Uusia sisältötyyppejä voidaan Drupalissa luoda kahdella tavalla, joko suoraan käyttöliittymän kautta tai moduulin asennuksen yhteydessä. Jälkimmäinen tapa on hyvin käytännöllinen, mikäli moduuli vaatii tietyn yksilöidyn tietotyypin olemassaoloa ja etsii sitä kautta edelleen hyödynnettävää sisältöä. [2, s. 119.]

2.7 Node

Yksi Drupalin tärkeimmistä rakenteellisista ominaisuuksista on node-järjestelmä. Node tarkoittaa suomeksi solmua, ja Drupalin suomalaisessa ammattiterminologiassa se toimii alkuperäisessä muodossaan sateenvarjoterminä kaikelle sivustolla julkaistavalle tekstisisällölle, aina etusivusta yksittäiseen blogijulkaisuun saakka. Sisällön tuottami-

seen tarkoitettuja alajärjestelmiä yhdistävät tietokentät, joihin syötetään julkaistavan solmun sisältämä informaatio. Näin ollen myös moduulin luoma sisältötyyppi on yhtä lailla tapa luoda solmuja, toisin sanoen sisältöä sivustolle. Node-järjestelmä antaa moduuleille mahdollisuuden laajentaa solmujen sisältöä lisäämällä niihin tarvittaessa uudenlaisia tietokenttiä. Lisäksi Node-järjestelmän avulla on mahdollista tarttua julkaistuihin solmuihin ja muokata niiden sisällön esitystapaa. Node-järjestelmän ansiosta kaikki Drupal-sivuston sisältö on yhdenmukaista ja tallennettuna tietokantaan samassa muodossa: id-tunniste, otsikko, sisältö, julkaisuajankohta ja noden luonut käyttäjä. Näin tietokannasta voidaan myös hakea mitä tahansa luotua sisältöä samanlaisin parametrein riippumatta solmun sisältötyypistä. [4, s.17; 1, s. 92–93.]

2.8 Entiteetti

Drupal-sivuston sisältöä on pystyttävä hallitsemaan mahdollisimman tehokkaasti. Edellä mainitun node-järjestelmän ulkopuolelle jää todellisuudessa informaatiota, joka joissakin tapauksissa vaatii samantapaista hallinnointia kuin muu julkaistu sisältö, kuten esimerkiksi sivustolla julkaistut kommentit. Entiteetti, toisin sanoen abstrakti olio, on yläkäsite, joka sisältää nodet ja niiden ulkopuoliset sisältötyypit. Esimerkiksi kommentit ja käyttäjät ovat entiteettijärjestelmän instansseja eli kyseisen pääluokan ilmentymiä. Siinä missä node on sivuston yhteydessä luotu instanssi, entiteettijärjestelmä mahdollistaa sivuston ulkopuolisen materiaalin hyödyntämisen ja uusien kenttien lisäämisen muihinkin kuin node-järjestelmän tietotyyppeihin, kuten kuvan tai vapaan kuvauksen liittämisen käyttäjäprofiiliin. [4, s. 151.]

2.9 Field

Tietokenttä (engl. field) on node-järjestelmän perusta. Sisältötyyppiin voidaan liittää haluttu määrä kenttiä, sekä myös tarvittaessa paljon erityyppisiä kenttiä. Tavallisimmat kenttätyypit ovat teksti-, kokonaisluku- ja liukulukukenttä. Nodeen voidaan liittää myös erilaisia valintakenttiä tarpeen mukaan, ja tavallisimmin HTML-lomakkeen yhteydessä käytettyjä ”select”- ”radio button”- ja ”checkbox”-elementtejä vastaavat tietokentät ovat olemassa Drupalin kenttäjärjestelmässä. Niiden lisäksi sisältötyyppeihin voidaan liittää eri moduulien aikaansaamia kenttiä, kuten mahdollisuus lisätä kuva tai päivämäärä. Erilaisten kenttien avulla saadaan luotua sivustolle rikkaampaa sisältöä. Toistuva in-

formaatio, kuten osoitetiedot jonkin sisältötyypin sisällä, on käytännöllisempää tallentaa erilliseen kenttään sen sijaan, että kaikki tieto olisi hajallaan leipätekstissä. Kenttiin jäsennelty tieto on lisäksi helpommin löydettävissä ja hallittavissa. CSS-luokat mahdollistavat myös muusta sisällöstä erottuvien tyylien asettamisen halutuille kentille. [3, s. 188; 1, s. 106–107.]

3 Custom Queries -moduulin suunnittelu

Tämän insinöörityön tavoite on luoda Drupal-moduuli, jonka avulla saadaan luotua dynaamisia linkkejä ulkopuolisille sivustoille käyttämällä etukäteen tallennettuja hakuparametreja. Kuten sanottu, moduulit ovat Drupalin toiminnan perusta. Moduulien avulla sivuston rakennetta saadaan muokattua ja samalla saa tehtyä sen toimintaa monipuolisemmaksi. Kuka tahansa voi kehittää ja halutessaan julkaista moduuleita. Yleisesti ottaen Drupal-kehittäjät antavat moduulien muodossa takaisin Drupalista saamansa hyödyn. Erilaisia moduuleita on yli 10 000 erilaista, ja ne kaikki ovat ilmaiseksi ladattavissa Drupal.org-sivustolta. Olemassa olevilla moduuleilla saa toteutettua monia asioita, mutta joissakin asiakasprojekteissa omaa moduulikehitystä tarvitsee silti tehdä. Tällöin syntyneestä moduulista voi tehdä asiakkaan luvalla julkaistavan version, ja moduuli lähetetään Drupal.org-sivustolle arvioitavaksi. Ennen julkaisua yhteisön ydinryhmä testaa moduulit tietoturva-aukkojen sekä koodissa olevien virheiden varalta, ja tämän vuoksi moduulien julkaisu on usein hyvinkin pitkäkestoinen prosessi. [1, s. 315–316; 3, s. 50.]

Moduulit voidaan jakaa kolmeen eri ryhmään, Drupal-asennuksen mukana tulevat niin sanotut ”core” (ydin) -moduulit, yhteisön jäsenten vapaaseen käyttöön kehittämät ”contrib” (yhteisön tukemat) -moduulit ja tiettyyn tarpeeseen mittatilaustyönä kehitetyt niin sanotut ”custom” (räätälöidyt) -moduulit. Viimeinen ryhmä sisältää sekä julkaisemattomia että vapaaseen käyttöön julkaistuja moduuleita, riippuen projektin luonteesta. Sivuston tilannut asiakasyhtiö voi esimerkiksi kieltää moduulin julkaisemisen ja pitää kaikki oikeudet sen ylläpitoon ja jatkokehitystyöhön. Yleisesti ottaen kuitenkin kaikki julkiset moduulit ovat vapaasti muokattavissa ja kenen tahansa käytettävissä. [1, s. 316; 12, s. 17–18.]

Toiset moduulit keskittyvät tiettyyn toiminnallisuuteen, kuten esimerkiksi haluttujen RSS (Really Simple Syndication) -syötteiden hakemiseen ja tuomiseen sivustolle, toiset moduulit tarjoavat kehittäjälle työkaluja monipuoliseen sisällönhallintaan. Näiden niin sanottujen supermoduulien ominaisuudet ovat enemmänkin abstrakteja viitekehyksiä kuin tiettyjen rajojen sisälle laadittuja konkreettisia käyttömalleja. Supermoduulien avulla saadaan luotua sivustolle sääntöjä ja erilaisia olosuhteita sekä hyödynnettyä luokiteltua sisältöä varsin tehokkaasti. [4, s. 10; 13, s.13.]

3.1 Moduulin vaatimukset

Insinöörityöprojektina rakennettavan Custom Queries -moduulin avulla on pystyttävä luomaan räätälöityjä hakuparametreja, joita kutsutaan verkkosivun käyttöliittymässä lisäämällä tallennetun parametrin loppuun käyttäjän tekstikenttään kirjoittama hakusana. Käyttäjä valitsee verkkosivun, josta haluaa hakea tietoa, ja tästä syntynyt dynaaminen linkki avataan uudessa välilehdessä. Moduulille on luotava oma sisältötyyppi, jonka avulla hakuparametrit saadaan tallennettua Drupal-sivuston tietokantaan oikeassa muodossa. Sisältötyypillä on oltava jokaista tallennettavaa hakuparametria kohden kaksi tekstikenttää, joista ensimmäiseen kirjoitetaan sivuston nimi ja toiseen sivuston vaatima hakuparametri.

Lisäksi moduulilla on oltava oma piensovellus, joka voidaan upottaa sivustoon lomakkeen muodossa. Piensovelluksen on kyettävä soveltamaan sisältötyyppiin tallennettua tietoa sivustolle syötettävään hakusanaan ja näin luomaan dynaamisia hakuparametreja. Widget-sovelluksen on listattava kaikki tietokantaan tallennetut hakuparametrit helposti haettavaan muotoon. Tämä on toteutettavissa kuvassa 3 näkyvän HTML-lomakkeen avulla, jossa on select-alasvetovalikko, tekstikenttä ja lähetyspainike. Select-valikon sisältö haetaan elementti kerrallaan tietokantaan tallennetusta tiedosta. Tekstikenttään syötettävä hakusana viedään sellaisenaan lomakkeen lähetyksen yhteydessä luotuun linkkiin, joka avataan uudessa välilehdessä.

Kuva 3. Hahmotelma ”widget-sovelluksen” lomakkeesta.

3.2 Moduulin ominaisuuksien kartoitus

Drupalin ensimmäinen versio julkaistiin vuonna 2001, ja tällä hetkellä yhteensopivan sisällön ja ylläpidettävien moduulien kehitystyö on keskittynyt pääosin seitsemänteen versioon. Kahdeksas versio on vielä beta-vaiheessa. Avoimeen lähdekoodiin perustuva Drupal on alusta alkaen rakennettu hyvin perusteellisen ja yksityiskohtaisen dokumentaation ympärille, ja kuka tahansa voi tarvittaessa tehdä järjestelmään lisäyksiä uusien moduulien muodossa. [1, s. 1–2.]

Drupalin ydintä, corea, ja sen mukana tulevia moduuleja ylläpidetään tietyn kehittäjäryhmän toimesta, ja kaikki tehdyt muutokset käydään tarkasti läpi tietoturvan ja koodin tehokkuuden ja virheettömyyden osalta. Stabiili Drupal-asennus on testattu toimivaksi perustaksi sivustoa rakennettaessa, ja tehdyt muutokset voivat aiheuttaa odottamattomia ja hallitsemattomia ongelmia koodin rikkoutuessa. Lisäksi myöhemmin päivitettäessä Drupal-asennuksen uusi versio yliajaa koko hakemistorakenteen ja kaikki olemassa olevat tiedostot korvataan uusilla. Tämä aiheuttaa sen, että kaikki coreen tehdyt muutokset katoavat. Näiden kahden syyn vuoksi ei voida suositella muutosten tekemistä suoraan Drupalin ytimeen tai valmiisiin moduuleihin. [14, s. 63.]

Räätälöity sisältö ja toiminnallisuus tulisi tehdä erilliseen moduuliin. Drupal.org sisältää paljon erilaisiin tarkoituksiin suunniteltuja moduuleita, ja on hyvin todennäköistä, että joku on ratkaissut vaatimusten kaltaisen ongelman. Ongelman kuvaus muutamalla sanalla tai verkkosivulle tuotavan elementin rakenne ovat hyviä lähtökohtia selvitettäessä, onko olemassa jo sopivan toiminnallisuuden mahdollistava moduuli. Moduulikirjastossa julkaistavilta moduuleilta vaaditaan dokumentaatiota, metatietoa, kuten kategorioita, avainsanoja ja informaatiota projektiin liittyvistä muista moduuleista. Dokumentaatiossa on oltava kattava kuvaus moduulin toiminnallisuudesta, ohjeet sen asentamisesta, tai

kuvakaappauksia tai videoita, kuinka moduuli toimii käytännössä. [13, s. 550; 15, s. 171–172.]

Drupal.org-sivuston yksittäinen moduuliin liittyvä sivu sisältää tietoa julkaistuista versioista ja aiemmin korjatuista ongelmista. Kirjautumalla sivustolle kuka tahansa kehittäjistä voi tehdä huomautuksia kohtaamistaan ongelmista moduulien käytössä tai vaihtoehtoisesti tehdä parannuksia vikalistalla oleviin kohtiin. Drupaliin liittyy vahvasti avoimen lähdekoodin ajattelumalli, sillä kaikki siihen julkaistut laajennukset ovat vapaasti kenen tahansa käytettävissä ja edelleen kehitettävissä. Jotkin moduulit ovatkin aiempien moduuliprojektien johdannaisia ja pohjautuvat johonkin yksittäiseen toiminnallisuuteen, mutta soveltavat sitä uudella tavalla. Parhaassa tapauksessa kehittäjän tekemä lisäominaisuus otetaan osaksi virallista moduulia, jolloin sen ylläpidosta vastaa moduulin alkuperäinen kehittäjä. Lisäksi muut moduulin käyttäjät saavat uuden ominaisuuden käyttöönsä. [14, s. 72.]

3.3 Monia tapoja osallistua

Drupal.org-sivusto on avoin foorumi, jossa laadukkaan ja loputtomasti laajennettavan järjestelmän ympärille julkaistut työkalut ovat vapaasti kenen tahansa käytettävissä, ja jokainen yhteisön jäsen voi halutessaan tuottaa materiaalia muiden hyödynnettäväksi. Hyvän kiertokulku on ajatuksena naiivi varsinkin nykypäivänä, jolloin kaiken ammattimaisen toiminnan on tuotettava tekijälleen tuloja. Mutta kuten Linux ja muut menestyneet vapaaseen lähdekoodiin perustuvat yhteisölliset projektit ovat osoittaneet, ihmiset haluavat auttaa kehittämään itse käyttämiään työkaluja. [13, s. 8–9.]

Drupal-yhteisö rohkaisee jäseniään antamaan takaisin järjestelmästä saamansa hyödyn jossakin muodossa, ja moduulikehityksen lisäksi on olemassa monia muitakin keinoja kehittää Drupal-järjestelmää. Verkkosivujen kehittäminen millä tahansa sisällönhallintajärjestelmällä vaatii pitkän opetteluprosessin läpikäymisen, ja kun kokeneemmat kehittäjät auttavat aloittelijoita, kaikki osapuolet voittavat. Foorumille jätetty kysymys mahdollisine vastauksineen auttaa todennäköisesti myös muita saman ongelman parissa painiskelevia, ja avoin keskustelu tuo usein esiin myös vaihtoehtoisia ratkaisuja ja erilaisia näkökulmia käsillä olevaan asiaan. [1, s. 4–5.]

Moduulikehitykseen liittyy itse koodaamisen lisäksi myös testaus ja esiintyvien ongelmien raportointi. Drupal.org-sivustolla jokaiseen moduuliin liittyvällä projektisivulla on erillinen osio, jossa eritellään avoimet ongelmakohdat ja ehdotetut lisäominaisuudet moduulin jatkokehitystä ajatellen. Arvostelujen kirjoittaminen ja käyttöohjeiden parantaminen, moduulin soveltaminen ja eri käyttömahdollisuuksien esitteleminen on myös tärkeää moduulikehityksen kannalta. Youtuben, oman sivuston tai blogin kautta jaetut kurssit, joissa opetetaan Drupalin käyttöä, ovat kehittäjälle hyvä keino jakaa omaa osaamistaan ja aloitteleville Drupal-kehittäjille rikas tiedonlähde. [3, s. 15; 13, s. 19.]

Yhteisön jäsen voi auttaa kehittämään Drupalia myös moduulikehityksen ja Internetin ulkopuolella. Sisällönhallintajärjestelmän menestyksen kannalta on tärkeää saada mahdollisimman paljon ihmisiä kiinnostumaan Drupalin tarjoamista mahdollisuuksista. Drupalin taustalla vaikuttaa maailmanlaajuinen yhteisö, jonka jäsenet ovat useimmissa tapauksissa erittäin aktiivisia ja lähes jokainen osallistuu omien mahdollisuuksiensa mukaan alan jatkuvaan kehittämiseen. Yhteisön sisällä järjestetään erityyppisiä ja -laajuisia tapahtumia, yksittäisten kaupunkien tapahtumista aina kokonaisten mannerten laajuisiin tapahtumiin. [3, s. 21; 13, s. 16.]

Tapahtumien perusajatus on ajankohtaisen tiedon välittäminen ja verkostoituminen alan toimijoiden kesken. Tapahtumien yhteydessä tarjotaan luentoja ja koulutusta, joiden lomassa ihmiset samalla tutustuvat keskustelemalla yhteisesti kiinnostavista asioista. Yksittäiset luennot ovat pääasiassa ilmaisia, mutta laajemmat koulutukset maksavat sisällöstä riippuen hyvinkin paljon. Drupal-yhteisön jäsenet ovat kuitenkin erittäin motivoituneita kehittämään itseään ja omaa osaamistaan, ja tällaiset koulutustilaisuudet ovat hyviä mahdollisuuksia oppia uusia asioita tekemällä. Drupal-järjestelmän opettelun aloitettuaan henkilön olisi hyvä hakeutua aluksi paikallisten toimijoiden järjestämiin tapahtumiin ja sitä kautta oppia uutta, saada kontakteja ja motivaatiota omakohtaiseen opiskeluun. [1, s. 3–4.]

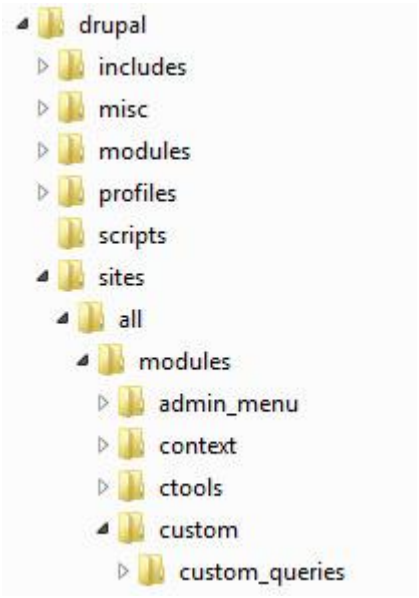
4 Moduulikehitysprojekti

4.1 Custom Queries -moduuli

Insinöörityössä rakennettiin räätälöity moduuli, nimeltään Custom Queries, joka mahdollistaa dynaamisten linkkien luomisen sivustolle asennettavan piensovelluksen avulla. Tietokantaan tallennetaan verkkosivustojen hakuparametreja, joihin piensovelluksen tekstikenttään syötetty hakusana yhdistetään. Moduuli toteutettiin Drupalin hook-funktioiden avulla, jotka ovat erikoistuneet kukin yksittäisen toiminnallisuuden luomiseen. Funktioiden avulla luotiin moduulille info- ja asetussivut sekä moduulin asennuksen yhteydessä syntyvä sisältötyyppi hakuparametrien tallennusta varten. Piensovellus rakennettiin hyödyntämällä Drupalin block-järjestelmää.

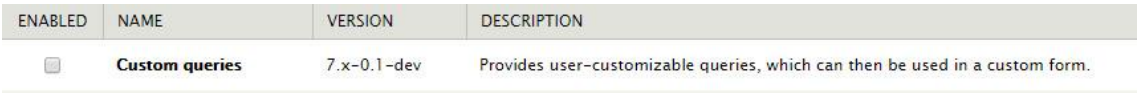
4.2 Moduulin rakenne

Jotta Drupal löytäisi lisäosat, eli moduulit ja teemat, ne on asennettava tiettyyn hakemistopolkuun. Drupal-sivuston asennuksen juuresta lähtevä /sites/all/-hakemisto sisältää kaksi kansiota, Themes ja Modules. Kuten kuvassa 4 on esitetty, kaikki yhteisön tukemat ja räätälöidyt moduulit asennetaan hakemistoon /sites/all/modules. Sittemmin vakiintuneen käytännön mukaisesti kehittäjän omat moduulit, itsetehdyt lisäosat, asennetaan vielä erikseen luotuun /sites/all/modules/custom-hakemistoon, jotta ne saadaan eroteltua Drupal.org-sivustolta ladatuista yhteisön tukemista moduuleista. Drupal löytää kaikki moduulit modules-hakemistosta, ja edellä mainittu käytäntö on tarkoitettu sivuston kehittäjää itseään varten – räätälöidyt moduulit ovat kehittäjän omia tuotteita, ja niiden varmuuskopiointi ja ylläpito on tämän itsensä vastuulla. Yhteisön tukemat moduulit sen sijaan löytyvät Drupal.org-sivustosta, ja ne saadaan päivitettyä ja ladattua uudelleen, mikäli tarve vaatii. [1, s. 317; 4, s. 27–28.]



Kuva 4. Moduulin hakemistorakenne.

Drupal-moduulin hakemiston sisältämien tiedostojen suhteen on tietyt minimivaatimukset. Moduulin nimellä varustetut .module- ja .info-päätteiset tiedostot on oltava luotuna, jotta Drupal löytää sen ja tunnistaa moduuliksi. Info-tiedosto, tässä tapauksessa custom_queries.info, sisältää määrittelyt, jotka kuvan 5 mukaisesti tulostetaan Drupal-käyttöliittymän moduulilistassa. Info-tiedostossa ovat lueteltuna moduulin nimen ja kuvauksen lisäksi myös ulkoiset tiedostot, kuten erikseen ladattavat JavaScript-liitännäiset. Info-tiedosto on tarkoitettu Drupalia itseään varten, mutta sen rakenne ja sisältö on myös ihmisen helposti luettavissa ja muokattavissa. Tiedoston sisältämät määritelmät vievät enintään toistakymmentä riviä koodia. [4, s. 29–33; 13, 551–553.]



Kuva 5. Listauksessa näkyvät moduulin Info-tiedostossa määritelty nimi, versio ja kuvaus.

Hook, sanatarkasti suomennettuna koukku, on yleisnimitys funktiolle, jolla tartutaan Drupalin rajapintaan. Hook-funktioiden tarkoitus on tarjota kehittäjälle valmiita työkaluja rakentaa räätälöityä toiminnallisuutta Drupal-järjestelmän ympärille. Jokainen olemassa oleva hook-funktio löytyy rajapintaa koskevasta dokumentaatiosta, jossa kerrotaan seikkaperäisesti, mitä kukin funktio saa aikaan ja minkälaisia parametreja ne vaativat

toimiakseen. Hook-funktiot ovatkin moduulikehityksen perusta, sillä niiden avulla saadaan moduulit reagoimaan Drupalin sisällä tapahtuviin asioihin halutulla tavalla ja luotua tarkoituksenmukaista toiminnallisuutta järjestelmään. Räätälöidyn moduulin sisältämä koodi ajetaan Drupalin ytimen, coren, olemassa olevaan prosessiin, ja näin hook-kutsut lisäävät prosessointikapasiteettia jakamalla tehtäviä myös Drupalin ytimen ulkopuolisille osa-alueille. [1, s. 317; 4, s. 13; 14, s. 46.]

Module-tiedosto, tässä tapauksessa custom_queries.module, sisältää moduulin toiminnallisuuden ja sisäänrakennetut ominaisuudet. Module-tiedoston runko rakentuu Drupalin rajapintaa koskevista hook-funktioista. Näiden Drupalin omien PHP-funktioiden avulla saadaan moduulille luotua haluttu toiminnallisuus, ja kokonainen funktioiden luettelo sisältää monta sataa eri tarkoituksiin sopivaa hook-funktiota. [4, s. 33–34; 13, s. 553–554.]

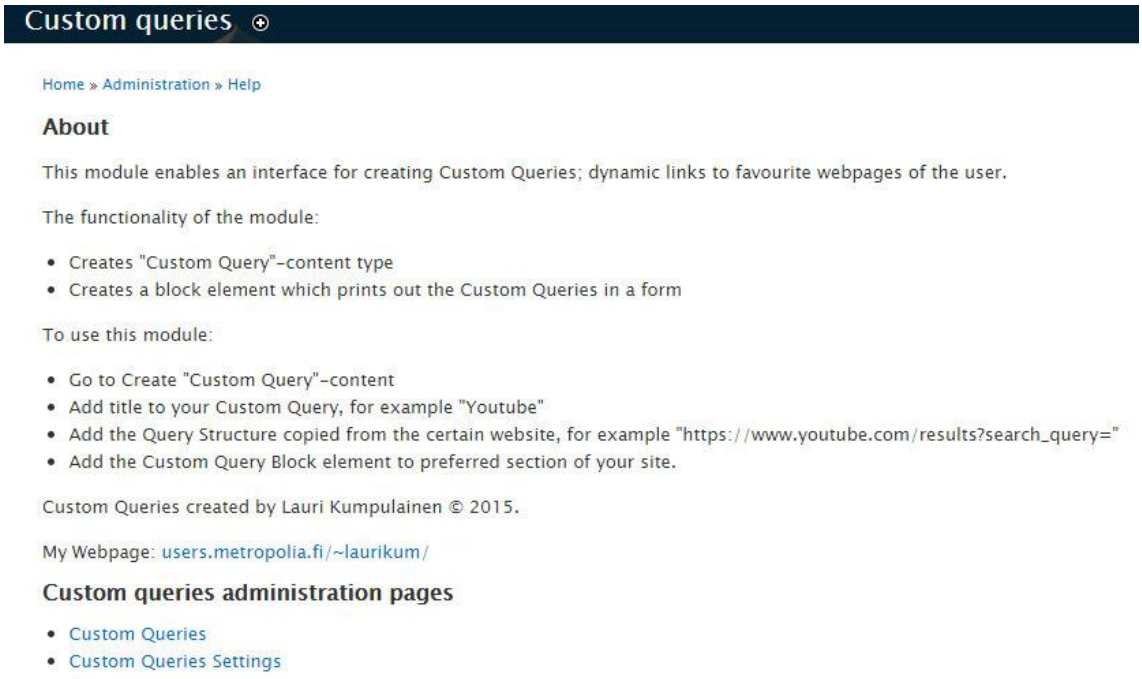
Custom Queries-moduulissa käytetään viittä funktiota:

- hook_help(), joka luo moduulille oman infosivun
- hook_menu(), jonka avulla saadaan luotua moduulille oma asetussivu
- hook_form(), joka mahdollistaa moduulin asetussivun tietokenttien määrittämisen
- hook_block_info(), joka määrittelee moduulin luoman block-elementin
- hook_block_view(), jonka kautta saadaan block-elementti näkymään halutulla tavalla.

Edellisten kahden pakollisen tiedoston lisäksi voidaan luoda moduulille myös asennustiedosto, tässä tapauksessa custom_queries.install. Moduulin määritelmät sisällytetään tällöin tiedostossa käytettyihin hook_install- ja hook_uninstall-funktioihin. Esimerkiksi moduulin asetussivun kenttien oletusarvot voidaan määritellä install-tiedostossa. Tässä tapauksessa moduulin asentamisen yhteydessä luodaan uusi tietotyyppi räätälöidyille hakuparametreille sekä tietotyypin sisältämät tietokentät. Kun moduulin asennus puretaan, samalla install-tiedoston määritelmien mukaisesti myös poistetaan tietotyyppi ja siihen liittyvä sisältö. Näin moduulin poistamisen jälkeen ei tietokantaan jää mitään ylimääräistä dataa, joka myöhemmin saattaisi aiheuttaa virheitä järjestelmän toiminnassa. Hook_install-funktio suoritetaan, kun moduuli aktivoidaan Drupalin käyttöliittymässä. Hook_uninstall-funktio taas suoritetaan, kun moduuli otetaan pois käytöstä ja valitaan käyttöliittymästä kohta ”Uninstall”. [13, 555–556; 1, s. 363.]

4.3 Moduulin infosivu

Drupal tarjoaa mahdollisuuden luoda moduulille oman ohjesivun, johon moduulin tarjoamat ominaisuudet saadaan dokumentoitua. Selkokielinen esitys moduulin toiminnasta mahdollistaa tehokkaan käyttöönoton, ja infosivun sisältö on tärkeää suunnitella tarkkaan ennen toteuttamista. Verrattuna moduulin kommentoituun koodiin, joka tarjoaa tarvittaessa samat mahdollisuudet toimia moduulin käyttöoppaana, kuvassa 6 näkyvä info-sivu on avattavissa suoraan sisällönhallintajärjestelmän käyttöliittymästä. [1, s. 339.]

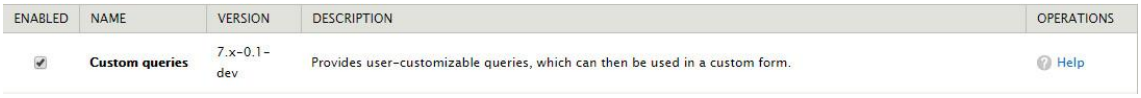


Kuva 6. Moduulin info-sivu sisältää tietoa moduulista sekä linkin asetussivulle.

Moduulikehityksessä usein käytetty työkalu on t-funktio, jonka tarkoitus on luoda helposti eri kielille käännettävää sisältöä, joka halutaan käyttäjälle esitettävän. Funktion sisälle kirjoitettu selkokielinen teksti saadaan näkyviin sellaisenaan Drupalin käyttöliittymässä, ja sitä käytetäänkin ilmaistaessa kenttien otsikoita, kuvauksia ja muuta käännettävää sisältöä. T-funktiota voidaan käyttää käännettävän sisällön lisäksi muun muassa PHP-kielessä ruudulle tulostukseen tarkoitetun print-funktion sijaan. T-funktio on suunniteltu siten, ettei sen läpi pääse koodia, kuten JavaScript-injektioita. Näin tulostettaessa esimerkiksi kirjautuvan käyttäjän nimi ei syötteen mahdollisesti sisältämää koodia suoriteta. T-funktiota voidaan käyttää myös Drupalissa suoritettavan JavaScript-sisällön kääntämiseen kutsumalla skriptissä Drupal.t()-funktiota. [1, s. 340; 4, s. 41, 302.]

Infosivun sisältö luodaan hyödyntämällä t-funktion käännösominaisuutta, ja sivulla voidaan käyttää tavallista HTML-syntaksia. Tämän ansiosta moduulin toiminnallisuutta voidaan havainnollistaa kaavioiden ja kuvankaappausten kaltaisten kuvasarjojen avulla. Infosivu on myös mahdollista rakentaa ohjekirjan muotoon luomalla pienimuotoinen sisällysluettelo, jossa linkit toimivat joko ankkureina sivun eri osiin tai ohjaavat moduuliin liittyvälle sivustolle muualla verkossa. [4, s. 38–39.]

Moduuli-tiedoston ensimmäinen funktio on nimeltään hook_help, jonka avulla infosivu luodaan. Funktion sisällä määritellään infosivun hakemistopolku ja t-funktiota käyttäen tulostetaan sivun sisältö. Kuten kuvasta 7 näkyy, siirryttäessä takaisin Drupalin käyttöliittymän moduulilistaukseen hook_help-funktion lisäämisen seurauksena moduulin kohdalle on ilmestynyt ”Help”-kuvake, joka toimii linkkinä määritetylle infosivulle. [1, s. 357.]

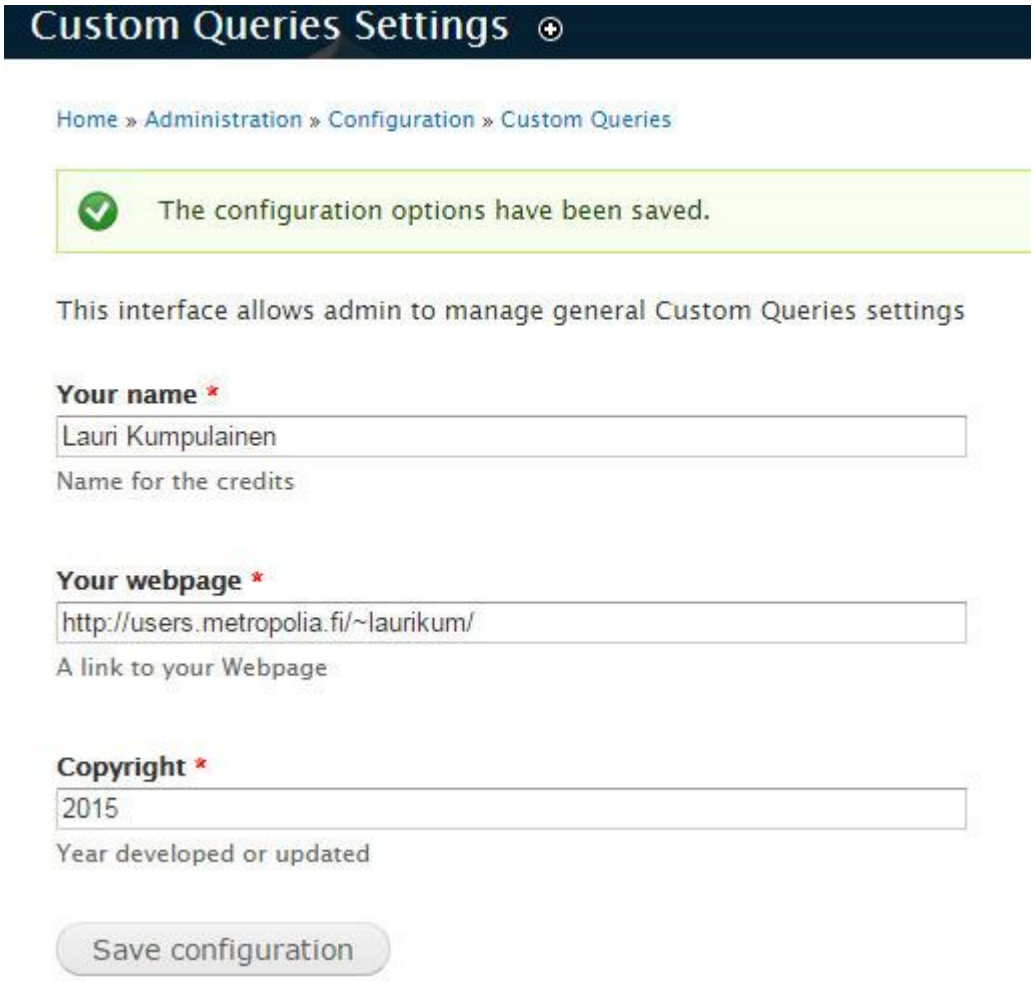


Kuva 7. Moduulilistauksen oikeaan reunaan ilmestyy linkki infosivulle.

4.4 Moduulin asetussivu

Moduulin käyttötarkoituksesta riippuen sille voidaan luoda Drupal-käyttöliittymään erillinen sivu, jossa asetuksia muokataan mahdollisuuksien mukaan. Esimerkiksi räätälöidyn karttaelementin mahdollistama moduuli sisältäisi oletusarvot koordinaateille, joiden perusteella kartta keskitettäisiin ladattaessa. Nämä arvot tallennettaisiin aina muutettaessa tietokantaan, ja asetussivun kautta ne olisivat tarvittaessa helposti vaihdettavissa. [1, s. 333.]

Asetussivun sisältö, eli itse lomake ja sen kentät, luodaan hook_admin_settings_form()-funktiossa. Varsinainen asetussivu luodaan kutsumalla hook_menu()-funktiota, jossa määritellään sivun sijainti Drupalin käyttöliittymässä ja sivuston osoiterakenteessa sekä tulostetaan edellä mainitun hallintalomakkeen sisältö sivulle. Tässä tapauksessa moduulin toiminnallisuus perustuu sisältötyyppiin, jonka sisällön avulla räätälöityjä hakukyselyjä voidaan luoda. Toisin sanoen, asetussivun sisältö ei vaikuta moduulin toimintaan suoraan. Kuvassa 8 näkyvä Custom Queries -moduulin asetussivu toimii graafisena käyttöliittymänä, jonka kautta luodaan omaleimainen ilme moduulin info-sivulle. [1, s. 352.]



Kuva 8. Kuvassa 6 näkyneet kehittäjän tiedot ovat vaihdettavissa asetussivun kautta.

Asetussivun kenttiin syötettävä tieto on tarkastettava ennen tietokantaan tallentamista, ja Drupal tarjoaa tätä varten funktion hook_admin_settings_form_validate(). Jokaiselle kentälle on sen sisällä mahdollista määrittää validointisääntö, jolla varmistetaan kunkin syötteen vastaavan kentän vaatimaa muotoa. Tämä toteutetaan PHP-funktion preg_match() avulla, jossa luodun säännön sisältämistä merkeistä eroava syöte aiheuttaa virheilmoituksen. Esimerkiksi kokonaislukuja vaativaan kenttään ei tule olla mahdollista syöttää kirjaimia, ja käyttäjää on myös hyvä informoida tästä. Form_set_error-funktion avulla voidaan kohdentaa kuhunkin kenttään oma virheilmoitus, jolloin käyttäjä saa sekä informatiivisen viestin että näkee käyttöliittymästä punaisen reunuksen ympäröimänä kentän, johon liittyvästä virheestä on kyse. [4, s. 127–128, 170.]

4.5 Moduulin luoma sisältötyyppi

Drupalin sisältö on mahdollista rakentaa monella eri tavalla. Eräs sisällönhallintajärjestelmän tärkeä ominaisuus on sen rakenteellinen joustavuus. Sisäänrakennetut kokonaisuudet, taksonomia ja sisältötyypit antavat vapauden toteuttaa sisällönhallinnan kunkin tapauksen vaatimalla tavalla. Taksonomia tarkoittaa sanastoa, ja perinteisesti sen yhteydessä puhutaan avainsanoista, tageista, joilla voidaan luokitella sisältöä. Nykyisin verkossa yleistynyt avainsanojen käyttö on eräs sovellus taksonomian hyödyntämisestä. Taksonomiaa voidaan kuitenkin käyttää myös paljon monipuolisemmin. Sen avulla luodaan uusia sanastoja, joiden sisälle lisätään sanastoon liittyviä termejä. Taksonomian käyttötarkoitus Drupal-sivustolla voi olla kuitenkin myös toisenlainen, ja sen avulla voidaan esimerkiksi ylläpitää sivuston päävalikkoa. Tällöin sanasto ”main menu” sisältäisi termit valikon elementeistä. Tässä tapauksessa sanaston asetuksissa määrättyyn URL-kenttään tulisi joka termiä luotaessa linkki haluttuun osoitteeseen, ja näin sanastoa voitaisiin hyödyntää rakentamalla sivuston päävalikko. [1, s. 157–159; 2, s. 37.]

Sisältötyypit ovat monipuolinen vaihtoehto sisällön hallintaan, sillä sisältötyyppi voidaan suunnitella minkälaiseksi tahansa. Yksinkertaisimmillaan sisältötyyppi sisältää kaksi kenttää, otsikon ja sisältöalueen tekstille, mutta erilaisia kenttiä voidaan lisätä lähes rajattomasti. Sisältötyypin luominen parantaa sivuston käytettävyyttä, kun Drupalin käyttöliittymästä käsin saadaan helposti tuotettua uutta sisältöä. Verrattuna taksonomian käyttämiseen, joka sopii paremmin rakenteellisiin yhteyksiin, sisältötyypin avulla saadaan intuitiivinen tapa lisätä ja muokata sisältöä. Moduulin oma sisältötyyppi ja sen sisältämät kentät luodaan Install-tiedostossa – tässä tapauksessa siis sivuston nimi ja sivustolle ominainen hakuparametri, URL-osoitekenttä. Kuten kuvasta 9 nähdään, sisältötyypin olemassaolo mahdollistaa uudenlaisen sisällön luomisen Drupalin graafisesta käyttöliittymästä käsin. [1, s. 111–112; 3, s. 152.]

Kuva 9. Yksittäiset hakuparametrit luodaan sisältötyypin avulla.

Kentän oma rajapinta, Field API, mahdollistaa uusien kenttien luomisen. Yksittäinen kenttä syntyy kaksivaiheisessa prosessissa, ja ensin se on määriteltävä sisältönsä puolesta. Kentän tyyppi, koneenluettava nimi, ja sen sisäinen datarakenne tallennetaan. Jotta varsinainen kenttä saadaan näkyviin Drupalin käyttöliittymässä, edellä määriteltyyn kenttään täytyy viitata toisaalla. Toinen vaihe onkin instanssin luominen, jossa syntyy niin sanottu fyysinen tietokenttä. Tällä tavoin käyttöliittymän puolella saadaan tulostettua haluttu kenttä oikeassa muodossa, ja sen avulla voidaan tallentaa tietoa. Instanssi sisältää kentän tyypin ja otsikon lisäksi kuvauksen, eli kentän yhteyteen tulostettavan ohjeistuksen, joka koskee kentän täyttämistä. [4, s. 184–185.]

4.6 Block-järjestelmä

Yksi Drupal-sivuston tärkeä työkalu on block-elementti, johon voidaan tuoda mukautettua sisältöä ja näin lisätä sivuston toiminnallisuutta. Block-elementtejä voidaankin kutsua Drupal-sivuston rakennuspalikoiksi. Sivusto jaetaan osiin, jotka tulostetaan teeman avulla halutussa muodossa. Block-elementtejä voidaan sijoittaa näihin määriteltyihin osiin, muun muassa sivun otsakeosaan (header), navigaatiopalkin alle (featured), sisältöalueen ylle (highlighted) tai korvaamaan kokonaan varsinainen sisältö jollakin sivulla

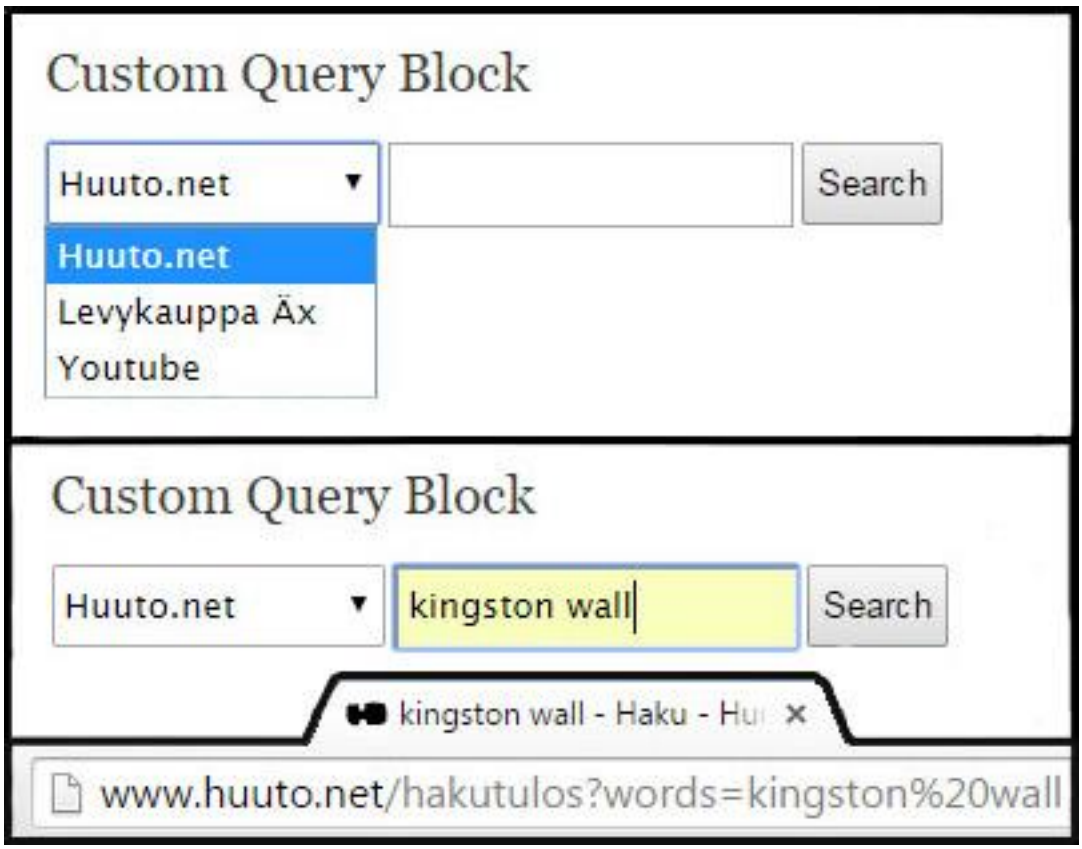
(content). Valmiiksi määritellyt palkkialueet täytetään block-elementeillä kuvassa 10 näkyvän käyttöliittymän kautta. Core-ominaisuuksien, kuten sisäänkirjautumisen ja etsin-kentän, lisäksi monet yhteisön tukemat moduulit luovat omia piensovelluksia, kuten esimerkiksi Twitter-syötteen sivulle liitettävän block-elementin muodossa. Drupalin yksi erittäin hyödyllinen ominaisuus on mahdollisuus määritellä konteksti, jossa kukin palkkialueen sisältämä block-elementti tahdotaan esittää. Tällöin tietyt elementit näkyvät vain kirjautuneelle käyttäjälle tai pääkäyttäjälle. Myös yksittäisten sivujen suhteen voidaan toteuttaa jokaisen block-elementin kohdalla yksilöllinen säännöstö, ja näin olosuhteista riippuen rakennuspalikat järjestyvät aina halutulla tavalla. [2, s. 71; 13, s. 123–124; 1, s. 33–34.]

Kuva 10. Block-listauksessa näkyvien elementtien sijoittelua sivustolla voidaan muuttaa.

Block-elementin suunnittelussa ja sijoittelussa on otettava huomioon sivuston toimivuus ja käytettävyys. Keskeisen sisällön on oltava hyvin esillä, jotta käyttäjä löytää sen helposti. Toissijainen sisältö on usein sijoiteltu sivuston laidoille, ja täytemateriaali halutaan minimoida. Sivuston tarkoitus on välittää informaatiota, ja käyttäjällä on oltava selkeä kuva sivuston tarjoamasta sisällöstä. Block-elementin on näin oltava yksinkertainen käyttää tai vaihtoehtoisesti sisältää hyvin kiteytettyä informaatiota. Sivuston sisältämät vaikeaselkoiset osiot hankaloittavat käytettävyyttä ja etäännyttävät käyttäjän alkuperäisestä tarkoituksesta vierailla sivustolla. Moduulin sisällä luotu block-elementti on käytännöllinen tapa tuoda uutta sisältöä sivustolle. Custom Queries -moduulin oma sisältötyyppi, jonka avulla hakuparametrit tallennetaan, olisi itsessään hieman epäkäytännöllinen ilman block-elementtiä. Jos moduulin avulla luotu sisältö vain tulostettaisiin sivustolle sellaisenaan, ei moduuli palvelisi alkuperäisiä tarkoitusperiä. [3, s. 69; 1, s. 70–71.]

4.7 Block-elementin toiminnallisuus

Moduulin luoma sisältötyyppi sisältää kaksi kenttää, verkkosivuston nimen ja sivuston hakuparametrin. Kun nämä tiedot yhdistetään käyttäjän syötteeseen, saadaan luotua räätälöityjä hakulausekkeita. Tätä varten on luotava käyttöliittymä, jossa tietotyypin sisältämät nodet tulostetaan valikkoon ja käyttäjän kirjoittamaan hakusanaan reagoidaan avaamalla valitun sivuston hakutulos. Liitteessä 1 esitetty Custom Queries -moduulin luoma block-elementti perustuu ajatukseen HTML-lomakkeesta, jossa tietotyyppiin tallennetut node-entiteetit tulostetaan select-elementin sisälle. Kuten kuvasta 11 näkyy, alasvetovalikon viereen on sijoitettu tekstikenttä, johon käyttäjä kirjoittaa haluamansa hakusanan. Lomakkeen lähetyksen yhteydessä vetovalikosta sillä hetkellä valittu sivuston nimi edustaa sen yhteyteen tietotyypissä liitettyä hakuparametriä. Näin syntyy dynaaminen linkki uuden ikkunan avatessa haun tuloksen valittuun hakuparametriin yhdistetyllä hakusanalla varustettuna. [16, s. 193–194, 200, 203–204.]



Kuva 11. Block-elementin toiminta: Valitaan parametri ja syötetään hakusana, ja linkki avautuu.

Form on lomakkeen luomiseen tarkoitettu HTML-elementti, jonka sisälle voidaan sijoittaa tekstikenttiä ja erilaisia valintoja. Form-elementin sisällön käsittely voidaan ulkoistaa erilliseen PHP-tiedostoon (käsittelijä.php), ja tietojen lähetys haluttuun käsittelijään on mahdollista kahdella eri tavalla, joko POST- tai GET-protokollalla. Näistä ensimmäinen sopii arkaluontoisen tiedon, kuten salasanojen ja muun henkilökohtaisen materiaalin, välittämiseen, sillä dataa ei tulosteta sellaisenaan vasta, kun käsittelijä.php on vastaanottanut sen. GET-protokolla sen sijaan tulostaa lomakkeen tiedot suoraan selaimen osoitekenttään muotoon *käsittelijä.php?tekstikenttä=”hakusana”*, joten sen käyttäminen räätälöidyissä hakuparametreissa on näistä kahdesta parempi vaihtoehto. [16, s. 189.]

HTML:n GET-protokolla vie lomakkeeseen syötetyt tiedot eteenpäin näkyvässä muodossa selaimen osoitekentässä, mutta liittää mukaan jokaisen kentän tunnisteen eli id-parametrissa olevan nimen. Tämän vuoksi form-elementtiä ei voida sellaisenaan käyttää räätälöityjen kyselyjen tuottamiseen muilla verkkosivuilla. Verkkosivustojen hakuparametrit eroavat toisistaan huomattavasti, ja käytettäessä GET-protokollaa oikeanlaisen hakutuloksen aikaansaaminen vaatisi itse hakusanan yksinkertaistamista. Lomakkeen lähetysosoite-parametriin liitettävä räätälöity hakuparametri halutulle sivulle, esimerkiksi ”http://muusikoiden.net/tori/haku.php?keyword=”, muuttuu lomakkeen lähetyksen yhteydessä linkiksi, johon lisätään hakusana muodossa *tekstikenttä=”hakusana”*. Tämä ei sellaisenaan tuota haluttua tulosta, sillä kyseisessä tapauksessa hakusanan tulisi olla pelkistettynä muotoon ”hakusana”, ilman tekstikentän nimeä. PHP sisältääkin monia tapoja käsitellä ja muuttaa tekstimuodossa olevaa dataa, kuten funktiot substr_replace() tai explode(), mutta koska hakuparametri muodostetaan form-elementin lähetyksen yhteydessä, toimintaprosessi monimutkaistuu. [17, s. 63–64, 216–218, 222–224.]

PHP:n avulla voitaisiin ensin tallentaa käyttäjän syöte muuttujaan, poistaa valitun funktion avulla hakuparametrista tekstikenttä-elementin id, tarkemmin kohta ”tekstikenttä=”, tallentaa jäljelle jäänyt hakusana toiseen muuttujaan, vasta tämän jälkeen lisätä se haluttuun hakuparametriin, ja viimein avata linkki oikealle sivustolle vaaditussa muodossa. Tällainen prosessi on mahdollista toteuttaa PHP:n avulla, mutta ratkaisun tarjoamat hyödyt ovat monimutkaisuuteensa nähden pienet. Palvelinpuolen sijaan samanlainen lomakkeenkäsittelytapa on mahdollista luoda käyttämällä selainpuolen skriptausta, tarkemmin JavaScriptiä. [17, s. 287–288.]

4.8 JavaScriptin mahdollistama toiminnallisuus

JQuery on JavaScript-kirjasto, jonka avulla saadaan ilmaistua alkuperäistä JavaScriptiä tehokkaammin skriptausta vaativat asiat huomattavasti vähemmällä määrällä koodia. Drupal on jQuery-yhteensopiva sellaisenaan, sillä Drupal-asennus sisältää kyseisen kirjaston. Monet Drupalin moduulit perustuvat jQueryn hyödyntämiseen, ja näin ollen kirjaston on hyvä olla valmiiksi asennettuna. Yksi tapa luotaessa uutta jQuery-skriptiä on korvata tavallinen jQuery-syntaksi, jossa tiedosto alkaa ”$”-merkillä, sanalla ”jQuery”. Tämä johtuu siitä, että Drupal varaa $-merkkiä muille JavaScript-kirjastoille, jotka mahdollisesti käyttävät tätä muuttujaa. [4, s. 288.]

JavaScriptin perusajatus on reagoida käyttäjän tekemiin asioihin verkkosivustolla, ja sen avulla voidaan luoda erilaisille elementeille niin sanottuja kuuntelijoita. Kuuntelijat keskittyvät seuraamaan niille osoitettujen elementtien tapahtumia, kuten tekstikentän syötettä ja painikkeiden eri asentoja, ja voivat laukaista tapahtumia niihin liittyen. JavaScript tarjoaa siis mahdollisuuden reagoida HTML-elementtien muutoksiin. Edellä mainitun form-elementin käyttämisen sijaan voidaan Custom Queries -moduulissa jäljitellä lomakkeen käyttäytymistä luomalla tekstikentän viereen tavallinen painike, jonka painaminen vie senhetkisen tekstikenttään syötetyn hakusanan eteenpäin JavaScript-muuttujan arvoksi. Samalla myös select-elementistä valittu sivuston nimi viedään eteenpäin. Liitteessä 2 on esitetty, kuinka näistä tiedoista saadaan JavaScriptin avulla muodostettua oikeassa muodossa oleva hakuparametri ja luotua dynaaminen linkki. [18, s. 199, 230, 235–238.]

5 Tulevaisuuden näkymät

Drupalin suosio on kasvanut viime vuosina merkittävästi. Tätä on edesauttanut sisällönhallintajärjestelmän soveltuvuus käytettäväksi kaikenkokoisissa verkkosovelluksissa. Drupal-pohjaisen alustan päälle on rakennettu keskisuuria sivustoja, toteutettu intranetratkaisuja sekä tehty laajoja rinnakkaisten sivustojen kokonaisuuksia. Drupal mahdollistaa laaja-alaisten asiakasprojektien toteuttamisen, mutta suurten kokonaisuuksien suunnitteluprosessi vie myös paljon aikaa. Joissakin tapauksissa sivuston valmistumisen jälkeen jatketaan projektin jatkokehityksen parissa. Palveluntarjoajat myyvät asiakkailleen osaamistaan ja tehdyistä projekteista saamaansa kokemusta, ja

asiakassuhteista tulee parhaassa tapauksissa pysyviä. Tämän ansiosta Drupal-kehittämiseen keskittyneet yritykset saavat yhä enemmän työtilauksia, ja näin tarvitaan lisää osaavia työntekijöitä. [19, s. 838.]

Drupal-kehittämiseen liittyvä työnkuva voidaan jakaa kahteen profiiliin: front-end- ja back-end-kehittäjään. Kuten edellä mainittiin, termi ”front-end” liittyy verkkosivuston näkymiseen selaimessa. Front-end-kehittäjä toimiikin Drupalin ulkoasun ja toiminnallisuuden parissa ja on keskittynyt CSS-tyylitiedostoihin sekä JavaScript-ohjelmointiin. Yleensä front-end-kehittäjän työnkuvan kohdalla puhutaan teemoittamisesta. Tällöin tarkoitetaan teeman luomista, useimmiten jonkin olemassa olevan puhtaan pöydän periaatteella rakennetun tyhjän teeman päälle. Drupal.org-sivustolla ladattavissa olevien valmiiden teemojen käyttö on toki mahdollista, mutta useimmiten teema luodaan varta vasten tiettyä sivustoa varten alusta loppuun. [19, s. 835–836.]

Front-end-kehittäjän selainpuolen sijaan back-end-kehittäjä työskentelee palvelinpuolen parissa. Tällöin työnkuvaan liittyy PHP-ohjelmointikielellä tehtävä moduuli- ja ohjelmistokehitys sekä mahdollisten uusien julkaistujen moduulien testaus. Pääasiassa kaikki Drupalia työkseen kehittävät voidaan jakaa ainakin pääpiirteittäin jompaan kumpaan näistä profiileista, ellei jopa molempiin samanaikaisesti. Kuitenkin on tärkeää muistaa, että merkittävä osa Drupal-kehityksestä sisältää työskentelyä järjestelmän graafisessa käyttöliittymässä. Toisin sanoen, myös henkilö jolla ei ole taustaa ohjelmoinnista, voi oppia hallitsemaan Drupalin ja päästä työskentelemään alalle. [19, s. 835–836, 851.]

Suomessa yhä useammat asiakkaat haluavat verkkosivustoja, jotka toteutetaan juuri Drupal-sisällönhallintajärjestelmällä. Verrattuna muihin markkinoilla oleviin järjestelmiin Drupal on hyvin monipuolinen, ja sen avulla voidaan luoda kaikenkokoisia ja -näköisiä verkkosovelluksia. Lisäksi Drupalilla luotuja sivustoja voidaan myöhemmin laajentaa haluttuun suuntaan ongelmitta. Asiakkaan kannalta on edullista valita kestävä, helposti päivitettävissä oleva järjestelmä, jonka käyttö on kehitystyötä lukuun ottamatta täysin maksutonta. Suosion myötä myös uusia työpaikkoja syntyy alalle jatkuvasti lisää, ja osaamista tarvitaan yritysten työmäärän kasvaessa. Asiakkaat ovat pääosin vakavaraisia yrityksiä ja tahoja, mikä edelleen auttaa luomaan myönteisiä näkymiä alan työtilanteessa. [19, s. 843–844; 20, s. 3–4; 21.]

6 Yhteenveto

Verkkoteknologiat kehittyvät jatkuvasti, ja on hyvin haastavaa valita ja opetella käyttämään sopivaa vaihtoehtoa lukemattomien tekniikoiden joukosta, sillä monet niistä hiipuvat ennen pitkää ja ne korvataan toisella. Drupalin etu on siinä, että uuden verkkoteknologian hallitseminen saattaa hyvinkin innostaa yksittäistä yhteisön jäsentä integroimaan kyseiseen tekniikkaan liittyvän toiminnallisuuden Drupaliin. Kun toinen käyttäjä kuulee uudesta teknologiasta, on melko todennäköistä, että joku muu on jo ehtinyt tehdä siihen liittyvän moduulin. Toisin sanoen, keskittämällä huomionsa Drupaliin verkkosivukehittäjä pysyy kehityksen mukana suhteellisen pienellä vaivalla.

Oma alkuperäinen ajatukseni oli tehdä opinnäytetyö Drupal-moduulin kääntämisestä. Sen tarkoitus olisi ollut ottaa jokin olemassa oleva Drupal 7 -version moduuli ja koodata se alusta alkaen uudelleen Drupal 8:n vaatimalla tavalla. Tämä kuitenkin olisi tarkoittanut molempien järjestelmäversioiden syntaksin opiskelemista, sillä moduulikehityksestä minulla ei ollut aiempaa kokemusta. Lopulta päätin kehittää täysin uuden moduulin Drupal 7:lle. Idea räätälöityjen hakuparametrien tallentamisesta jalostui projektin myötä käytännön sovellukseksi, joka täyttää alkuperäiset vaatimukset: hakuparametreja voidaan tallentaa, ja lomakkeen kautta syötetyn hakusanan avulla luodaan dynaaminen linkki halutulle sivustolle. Projektin suunnitteluvaiheessa ajattelin parametrien tallennuksen tapahtuvan moduulin asetussivun kautta. Kuitenkin projektin edetessä ymmärsin luoda oman sisältötyypin, eikä tallennettavien hakuparametrien määrää tarvitse mitenkään rajoittaa.

Block-elementin rakentaminen osoittautui monimutkaisemmaksi kuin alun perin odotin, ja suurin yksittäinen ongelma oli ratkaista, kuinka käsitellä ja yhdistää käyttäjän syöttämä tieto jo olemassa olevaan, tietokantaan tallennettuun tietoon. Custom Queries -moduuli tulee toimimaan aluksi lokaalissa testiympäristössä osana selaimen aloitussivua, ja tulevaisuudessa moduulin ympärille on tarkoitus kehittää julkinen verkkopalvelu. Projektin aikana oppimani asiat moduulikehityksen perusteista ja järjestelmän rakenteesta syvensivät entisestään mielenkiintoani Drupaliin. Moduulikehityksen perusteisiin lukeutuvien moduulin asetus- ja infosivujen luominen mahdollistaa jo sellaisenaan erilaisia ratkaisuja. Moduulisivun yhteyteen voidaan liittää linkkejä ja muuta aiheeseen liittyvää tietoa, ja yksittäinen moduuli toimii näin parhaassa tapauksessa työnäytteenä

ja osana kehittäjän portfoliota. Custom Queries -projektin yhteydessä toteutettu sisältötyyppi ja block-elementti ovat Drupalissa yleisimmin käytettyjä rakennusaineksia, ja niiden taustojen oppiminen on ollut hyvin mielenkiintoista. Lisäksi valmiiksi tuttujen ainesosien luominen koodin kautta vähentää kynnystä syventävään moduulikehitysopiskeluun.

Ohjelmoinnin perusteet antavat teoreettiset valmiudet ymmärtää ja luoda koodia käytettyjen kielten puitteissa, mutta ilman käytännön sovelluksia hankitut taidot helposti pääsevät unohtumaan. Drupal on rakennettu PHP-kielen pohjalta, mutta sen yksilöllinen syntaksi ja rakenteelliset ratkaisut vaativat paljon opettelua. Järjestelmän pintapuolinen hallitseminen on mahdollista saavuttaa suhteellisen pienessä ajassa, mutta syvempi osaaminen saadaan vain yksilöllisen omistautumisen kautta. Moduulikehityksen perusteiden opiskelu kehittää samalla myös ohjelmointitaitoja, ja yksinkertaisen moduulin liittyessä osaksi suurempaa kokonaisuutta saa tällainen yksinkertaisempikin projekti enemmän merkitystä.

Drupalin taustalla vaikuttavien monimutkaisten prosessien ja teknologioiden hallitseminen vaatii pitkää perehtymistä, ja käsitteiden ja moduulien laaja kirjo saattaa vaikuttaa alkuun hyvinkin haastavalta. Vaikka Drupalin käyttäminen on mahdollista ilman koodaamista ja sivustoja sekä verkkopalveluja saadaan toteutettua hyödyntämällä olemassa olevia teemoja ja moduuleita, kiinnostus kurkistaa pintaa syvemmälle kasvaa viimeistään siinä vaiheessa, kun pelkän Drupalin käyttöliittymän avulla ei saada tuotettua toivotunlaista tulosta tai ominaisuutta sivustolle.

Yhteisön sisällä luodut lisäominaisuudet tekevät järjestelmästä vahvemman. Yleisesti moduulikehitys on tällä hetkellä keskittynyt yhä pääosin Drupalin seitsemännen version kanssa yhteensopiviin rakennuspalikoihin, mutta Drupal 8 on tätä opinnäytetyötä kirjoitettaessa jo beta-vaiheessa, ja sille on mahdollista luoda omia moduuleita sekä rakentaa sen avulla sivustoja. Moduulikehitys kulkee Drupalin kehityksen mukana, sillä suuri osa käytetyimmistä moduuleista rakennetaan uuden järjestelmäversion vaatimalla tavalla. Drupalin rakenne ja sen tarjoama rajapinta päivitetään jokaisen kantaversion julkaisun yhteydessä, minkä vuoksi vanhat moduulit eivät ole enää yhteensopivia uuden version kanssa. Moduuleista luotavat uudet versiot pitävät järjestelmän suojattuna, kun voidaan varmistua kaiken versiolle yhteensopivan materiaalin ajankohtaisuudesta.

Lähteet

1 Redding, Jacob. 2010. Beginning Drupal. Wiley Publishing.

2 Tomlinson, Todd. 2010. Beginning Drupal 7. Apress.

3 Mercer, David. 2010. Drupal 7. Packt Publishing.

4 Butcher, Matt; Farina, Matt; & co. 2010. Drupal 7 module development. Packt Publishing.

5 Emond, Justin; Steins, Chris. 2011. Pro Web Project Management. Apress.

6 Chacon, Scott; Straub, Ben. 2014. Pro Git. Apress.

7 Thomas, Shane. 2014. 5 secrets to becoming a Drupal 7 ninja. Code Karate.

8 Travis, Brian. 2011. Pro Drupal 7 for Windows developers. Apress.

9 Sikos, Leslie. 2011. Web Standards, Mastering HTML5, CSS3 and XML. Apress.

10 Thompson, Ben. 2015. Mobile first. Verkkodokumentti <http://stratechery.com/2015/mobile-first/>. Luettu 20.2.2015.

11 Powers, David. 2009. Getting StartED with CSS. Apress.

12 Douglass, Robert; Little, Mike; Smith, Jared. 2006. Building online communities with Drupal, phpBB, and WordPress. Apress.

13 Shreves, Ric; Dunwoodie, Brice. 2011. Drupal 7 Bible. Wiley Publishing.

14 Madel, Kurt. 2012. Drupal 7 Development by Example. Packt Publishing.

15 Poon, Dave. 2011. Drupal 7 Fields/CCK. Packt Publishing.

16 Schultz, David; Cook, Craig. 2007. Beginning HTML with CSS and XHTML. Apress.

17 Gilmore, W. Jason. 2010. Beginning PHP and MySQL. Apress.

18 Ferguson, Russ; Heilmann, Christian. 2013. Beginning JavaScript with DOM Scripting and AJAX. Apress.

19 Melançon, Benjamin; Luisi, Jacine; Négyesi, Károly. 2011. The Definitive Guide to Drupal 7. Apress.

20 James, Daniel. 2009. Crafting digital media: Audacity, Blender, Drupal, GIMP, Scribus, and Other Open Source Tools. Apress.

21 Drupal-sivut.fi. 2010. Verkkodokumentti. Mediatoimisto Pohjaton. <http://drupalsivut.fi/>. Luettu 19.3.2015.

Custom_queries.module-tiedostossa tulostettava block-elementin sisältö

Koodi mukailtu Jon Peckin Drupal 7 Custom Module Development -kurssin sisällöstä.

```
function custom_queries_block_contents($delta){
        $output = ‘’;
        switch ($delta){ case 'title':{
                // Luodaan tietokantakysely
                $query = new EntityFieldQuery();
                // Näytetään entiteeteistä vain node-tyyppi
                $query->entityCondition('entity_type', 'node');
                // Näytetään nodeista vain Custom Queries-tyyppiset
                $query->entityCondition('bundle', 'custom_query');
                // Järjestetään  Custom Queries-nodet nimen mukaan
                $query->propertyOrderBy('title', 'ASC');
                // Suoritetaan kysely
                $result = $query->execute();
                // Mikäli ei ole näytettäviä tuloksia, siirrytään eteenpäin
                if(!isset($result['node'])){
                return '';
        }

// Tietokannan sisältö listataan select-elementin sisälle
$nodes = node_load_multiple(array_keys($result['node']));
$output = '<select id="select" style="height:31px;">';
foreach($nodes as $node){
        // Jokaisen hakutuloksen osalta saadaan sivuston nimi ja parametri
        $title = $node->custom_query_title[LANGUAGE_NONE][0]['value'];
        $url = $node->custom_query_url[LANGUAGE_NONE][0]['value'];

        //Mikäli nodesta puuttuu nimi tai hakuparametri, se ohitetaan
        if(($title == '') || ($url == '')){
        continue;
        }
//Jokainen hakuparametri lisätään omana option-entiteettinä select-elementin sisälle
```

```
$output .= '<option value="'.$url.'">'.$title.'</option>';
}
$output .= '</select>

//Lisätään lomakkeen muut elementit, tekstikenttä ja hakupainike
<input type="text" name="userinput" id="input"/>
<button id="button">Search</button>';
break;
}
}
//Tulostetaan lomake sisältöineen block-elementiin
return $output;
}
```

Custom_queries.js-tiedostossa luotava dynaaminen linkki

Koodi mukailtu Drupal.org-sivuston, sekä Paolo Bergantinon kirjoittamista, ohjeista.

```
// Lisätään Drupaliin jQuery-funktio
(function ($) {
  Drupal.behaviors.custom_query = {
    attach: function (context, settings) {

                // Klikattaessa otetaan select- ja tekstikenttä-elementeissä olevat arvot
                $('#button').click(function(){
                $string = $('#input').val();
                $address = $('#select').val();
                // Saaduista tiedoista luodaan dynaaminen linkki
                $combi = window.open($address + $string, '_blank');

});
                // Varmistetaan toiminnallisuus, mikäli käyttäjä preferoi Enter-painiketta
                 $('#input').keypress(function(e){
                if(e.which == 13){
                $('#button').click();
        }
    });

    }
  };
}(jQuery));
```