

# **KUVAMATERIAALIN HAKU PAIKKATIEDON PERUSTEELLA - DRUPAL-MODULIN KEHITYS**

Timo Sorakivi

Opinnäytetyö  
Marraskuu 2012  
Tietojenkäsittelyn koulutus-  
ohjelma  
TAMK

## TIIVISTELMÄ

Tampereen ammattikorkeakoulu  
Tietojenkäsittelyn koulutusohjelma

SORAKIVI, TIMO:

Kuvamateriaalin haku paikkatiedon perusteella - Drupal-modulin kehitys

Opinnäytetyö 41 sivua, joista liitteitä 3 sivua  
Marraskuu 2012

---

Kartta on ollut kautta ihmiskunnan historian yksi tärkeistä maailmankuvaa muokkaavista välineistä, jonka avulla on voitu suunnistaa haluttuun päämäärään, luoda uusia kauppayhteyksiä tai pohtia omaa suhdetta näkyvään maailmaan. Uudet tekniikat, kuten satelliittinavigointi ja tietoverkot, ovat mullistaneet tapamme käyttää karttatietoa. Yhdistämällä paikkatiedon sisältävä kuvamateriaali karttapohjaan saadaan muodostettua aivan uusi havainnollisuuden taso, jonka avulla arvioida todellisuutta.

Työssä on toteutettu yksi tapa koota eri kuvapalveluiden tarjoamaa paikkatietoon yhdistettyä kuvamateriaalia. Google Mapsin ohella käytettiin hyväksi avoimen lähdekoodin sisällönhallintajärjestelmä Drupalia ja siihen saatavia toimintoja laajentavia lisäosia eli moduleja. Työn tuloksena syntyi uuden sisältötyypin tarjoava moduli, joka tallettaa kohteen paikkatiedon ja hakee lähettyvillä olevaa kuvamateriaalia kahdesta eri kuvapalvelusta. Lähteinä suunnittelussa ja toteutuksessa hyödynnettiin eri internetsivustoja, joista tärkeimpänä Drupalin tarjoama laaja materiaali. Lisäksi alan kirjallisuus tarjosi havainnollisten kaavioiden ohella täsmällistä ja mietittyä tietoa aiheesta.

Moduli toteuttaa vähimmäisvaatimukset paikkatietoon perustuvalle kuvahaulle. Uusia kuvapalveluita voidaan lisätä modulin toimintoihin soveltamalla toteutettua rakennetta. Käyttämällä tunnettua sisällönhallintajärjestelmää pohjana voidaan parhaalla tavalla turvata työkalun jatkokehitys. Moduli on vapaasti saatavilla Drupalin sivustolla GNU General Public License -lisenssiehdoin.

---

Asiasanat: drupal, moduli, panoramio, flickr

## ABSTRACT

Tampereen ammattikorkeakoulu  
Tampere University of Applied Sciences  
Degree Programme in Business Information Systems

SORAKIVI, TIMO:

Finding Photos Tagged With Position Data - Developing a Drupal Module

Bachelor's thesis 41 pages, appendices 3 pages  
November 2012

---

Through the history of mankind a map has been one of the important tools that has helped to shape our view of the world. A map has made it possible to navigate to the desired goal, to create new trade links and to consider man's own relationship to the visible world. New technologies, such as satellite navigation and information networks, have revolutionized the way we use map data. By combining a map with photos tagged with location data has helped to form a whole different level of illustration that allows us to evaluate reality in a new way.

In this work one way to collect photos tagged with location data from different photo sharing sites was implemented. Google Maps and the open source Drupal content management system with its available add-ons were utilized to reach this goal. A new module was designed and implemented, including a new kind of content type storing the location with the capability to search photos located in the vicinity. The photos were collected from two different photo sharing sites. Most sources utilised in this work came from several Internet sites, of which the most important was the Drupal's own site for module developers. In addition, the literature provided with several illustrative diagrams and accurate and thought-out information on the subject.

Module implements minimal requirements for a location-based image search. To expand its capabilities new photo services can be added to the module functions by applying the implemented structure. By using a well-known content management system as a base, we can best ensure the further development of the tool. The module is freely available at Drupal site and shared under the GNU General Public License.

---

Key words: drupal, module, panoramio, flickr

## SISÄLLYS

1	JOHDANTO.....	7
2	Kuvapalvelut ja paikkatieto.....	9
2.1	Historiaa.....	9
2.2	Paikkatieto.....	9
2.3	Geotagging.....	10
2.4	Kuvapalvelut.....	11
3	Palvelut ja rajapinnat.....	12
3.1	Palvelun valintakriteerit.....	12
3.2	Tiedonvälitys.....	12
3.3	XML.....	13
3.4	JSON.....	13
3.5	AJAX.....	14
4	Drupal-ympäristö.....	15
4.1	Rakenne ja alusta.....	15
4.2	Asennus.....	16
4.2.1	Apache, MySQL, PHP.....	16
4.2.2	Drush.....	17
4.2.3	Drupal.....	17
4.3	Arkkitehtuuri.....	19
4.3.1	Ydin.....	19
4.3.2	Koukkufunktiot.....	20
4.3.3	Modulit.....	20
4.3.4	Tietokanta.....	21
4.4	Toiminta.....	22
4.4.1	Pyyntö ja vastaus.....	22
4.4.2	Index.php ja bootstrap-prosessi.....	23
5	Modulin kuvaus.....	26
5.1	Toiminta.....	26
5.2	Integrointi Drupaliin.....	29
5.3	Tiedostot ja toiminnot.....	30
5.3.1	Modulikansio.....	30
5.3.2	Tiedosto poi.install.....	30
5.3.3	Tiedosto poi.admin.inc.....	31
5.3.4	Tiedosto poi.module.....	31
5.3.5	Tiedosto poi_map.js.....	32
5.4	Projektin versionhallinta.....	34

5.5 Debuggaus ja testaus.....	34
6 POHDINTA.....	36
LÄHTEET .....	37
LIITTEET .....	39
Liite 1. poi.module: modulin toimintojen reititys. ....	39
Liite 2. poi.module: Flickr-hakulauseen muodostus. ....	40
Liite 3. Flickr-hakutulos JSON-muodossa. ....	41

**LYHENTEET JA TERMIT**

AJAX	Asynchronous JavaScript and XML, joukko web-sovelluskehityksen tekniikoita
CMF	Content Management Framework, sovelluskehys sisällönhallintajärjestelmälle
CMS	Content Management System, sisällönhallintajärjestelmä
CRUD	(Create, Read, Update, Delete), tietokantaan kohdistettavia toimintoja
DBMS	Database Management System, tietokannan hallintajärjestelmä
GIS	Geographical Information System, paikkatietojärjestelmä
GPS	Global Positioning System, paikannusjärjestelmä
JSON	Javascript Object Notation, yksinkertainen tiedonsiirtomuoto
MySQL	yleinen relaatiotietokantaohjelmisto
PHP	Hypertext Preprocessor, ohjelmointikieli
XML	Extensible Markup Language, rakenteellinen kuvauskieli

## 1 JOHDANTO

Idea työlle sai alkunsa eräänä sunnuntai-iltapäivänä, kun tutkin muutamaa mielenkiintoista matkakohdetta netissä. Olisin halunnut tutustua kohteisiin jo kotisohvalla, joten päätin etsiä sopivaa kuvamateriaalia internetin avulla. Ensimmäiseksi kokeilin Google Earth työpöytäsovellusta.

Kuvia Googlen sovellus toki löytää, mutta harmillisesti suuri osa kohteiden kuvista osoittautui joko epäinformatiivisiksi tai täysin turhiksi - auringonlaskuja ja pikkuruisia panoraamakuvia on enemmän kuin riittämiin. Sovelluksen kuvamateriaalin toimittaa Panoramio, joka on yksi suurimpia verkossa toimivia kuvapalveluita. Voisi näin ollen olettaa, että parempaakin materiaalia löytyy, ja ongelma onkin ehkä enemmän sovelluksen tavassa esittää kuvalinkit vailla ns. thumbnail-kuvia, jolloin käyttäjän on avattava jokainen linkki saadakseen tietää, millaisesta kuvasta on kyse.

Siirryin toiseen Googlen sovellukseen, verkossa toimivaan Google Mapsiin. Karttapohjan päälle voi valita kuvatason, jossa suosituimmat kuvat esitetään thumbnail-tyyppisinä, ja edelleen karttaa zoomaamalla useimmat kuvat tulevat näkyviin. Esitystapa on luonteva ja nopeuttaa huomattavasti mielenkiintoisen kuvamateriaalin löytämistä. Kuvamateriaali on kuitenkin edelleen Panoramion toimittamaa, joten ryhdyin etsimään muita kuvapalveluita, joissa kuvien paikkatieto olisi saatavilla.

Flickr on osa Yahoon palveluita, joka sisältää huomattavan määrään kuvamateriaalia ja tarjoaa myös ohjelmallisen rajapinnan palvelun toimintoihin (Flickr 2012). Olisi siis mukavaa, jos karttapohjassa voisi esittää myös näitä kuvia Panoramion lisäksi. Verkosta löytyy useita muitakin kuvapalveluita, joiden tarjontaa voisi hyödyntää. Miten saisin yhdistettyä nämä toiminnot siten, että käytössäni olisi työkalu, jossa voisin valita kohteen kartalta, tallentaa koordinaatit ja löytää toivottavasti mielenkiintoista kuvamateriaalia karttapohjaan kiinnitettynä?

Sovellus vaatii avukseen toimintoja, joiden avulla mielenkiintoisia paikkoja voi tallentaa pysyvästi. Tarvitaan siis tietokantaa ja jotakin palvelimella suoritettavaa ohjelmakoodia. Sovellus voitaisiin toteuttaa aivan alusta lähtien käyttäen PHP:tä ja MySQL:ää,

mutta tällä kertaa päätin hieman oikaista ja käyttää hyväkseni jotakin tarjolla olevista CMS-järjestelmistä.

Olin tutustunut erilaisiin CMS- järjestelmiin jo aikaisemmin, ja näistä Drupal viehätti minua eniten. Päätin siis integroida työkaluni Drupaliin, jossa moduli-rakenteen ansios- ta kehitystyössä oli mahdollista hyödyntää muita kolmannen osapuolen tarjoamia modu- leita, ja näin saada jo valmiita ja testattuja toimintoja käyttöön. Lisäksi modulini tulisi koko Drupal-yhteisön käyttöön ja saattaisi hyödyttää muita tämänkaltaista toimintoa kaipaavia.

Tavoitteeni olisi siis yhdistää paikkatieto, eri kuvapalveluiden tarjonta sekä sisällönhal- linta yhdeksi kokonaisuudeksi, josta olisi hyötyä sekä käyttäjälleen että laajemmin suu- relle Drupal-kehittäjäyhteisölle. Modulin rakenteen tulisi myös sallia mahdollisimman monen kuvapalvelun liittäminen mukaan, ja näin lisätä tuotteen joustavuutta ja moni- käyttöisyyttä.



## 2 KUVAPALVELUT JA PAIKKATIETO

### 2.1 Historiaa

Ensimmäinen tunnettu kartaksi tulkittava kuvitus on toteutettu jo 16 500 eaa. Lascaux'n luolassa Lounais-Ranskassa. Kartan kohde ei kuitenkaan ollut maanpäällinen, vaan osa yötaivasta. Ei tunneta tarkemmin, mikä tämän kartan tarkoitus oli, tai mitkä olivat olleet kartografien motiivit piirtää teoksensa pimeään luolaan (History of cartography 2012).

Navigointi on varmaankin yksi tärkeimmistä tavoista hyödyntää kartta-aineistoa. Tyyneenmeren saarten asukkaat saattoivat purjehtia rannattomilla ulapoilla ainostaan tähti-taivaan ja havaittujen luonnonilmiöiden ohjaamina vailla piirrettyä karttaa tai kompassia. Antiikin ajan Välimerellä navigointi liittyi jo varhain kauppamerenkulkuun, ja reitit kulkivat tavallisesti pitkin rannikkoja. Tällöin tarvittavat kartat olivat enemmänkin purjehdusohjeita (lat. *periplus*), joissa lueteltiin järjestyksessä rannikon satamia etäisyyksi-neen ja matkaan kuluvine aikoineen (Johnson 2007, 48). Vasta kompassin käyttöönotto 1100-luvulla antoi avomeripurjehdukselle keinot tarkempaan ja turvallisempaan navigointiin.

Nykyaikainen kartografia sai alkunsa 1500-luvulla tarpeista löytää paremmin todellisuutta vastaava kuvaus, jota voitaisiin käyttää myös pitkillä merimatkoilla uusiin valloitetuihin siirtomaihin. Maailmankuvan muutos maakeskisestä nykyaikaiseen antoi osaltaan kehitykselle suuntaa, samoin kuin uudet tekniset keksinnöt kuten jousivetoinen siirrettävä kello ja kehittyneet navigointi-instrumentit. Yleinen luonnontieteiden kehitys on palvellut myös kartografiaa, ja nykyään karttatiedon pohjalle rakennettuja järjestelmiä voidaan hyödyntää hyvinkin erilaisissa kohteissa.

### 2.2 Paikkatieto

GIS (Geographic Information Systems eli paikkatietojärjestelmä) on yksi tällainen karttatiedon hyödyntämiseen kehitetty järjestelmä, joka on suunniteltu kaikentyyppisen karttatiedon keräämiseen, käsittelyyn ja analysointiin, sekä kerätyn tiedon hallintaan ja esittämiseen tyypillisesti jonkin päätöksentekoprosessin apuvälineeksi. (GIS 2012).

GIS voidaan määritellä kartografian, tilastollisen analyysin ja tietokantateknologioiden yhteenliittymäksi. Yhdistämällä tietoja eri lähteistä, voidaan luoda uusia näkökulmia ja saada arvokasta apua päätöksentekoon. Klassinen esimerkki kartan ja muusta lähteestä tulevan tiedon yhdistämisessä on Lontoossa v.1854 puhjenneen kolera-epidemian tartuntapisteen paikantaminen ja torjunta (Soho 2012). Lääkäri John Snow päätteli tautiin sairastuneiden osoitteiden ja kartta-aineiston avulla todennäköiseksi tartuntapaikak- si saastuneen kaivon, jota lähialueen asukkaat olivat käyttäneet. Kun kaivon käyttö es- tettiin, epidemia saatiin nopeasti hallintaan.

Web GIS ja Web mapping (karttapalvelu) ovat usein synonyymeinä käytettyjä termejä kuvaamaan verkossa toimivia karttatiedon analyysi- ja esityspalveluita. Web GIS pai- nottuu ehkä enemmän analyysiin, kun taas Web mapping paremminkin tiedon esittämi- seen. Rajat termien käytölle ovat kuitenkin melko löyhät. Karttapalvelut ovat laaja ko- konaisuus erilaisia teknologioita, joiden avulla karttatietoa voidaan tuottaa ja esittää WWW:n avulla.

Palvelut voivat olla julkisesti rahoitettuja, kuten Suomessa Maanmittauslaitoksen julkai- semat kartta-aineistot, tai kaupallisia kuten Google Maps ja Microsoftin Bing. Open Street Map edustaa vapaaehtoiseen tuotantoon perustuvaa mallia, jossa käyttäjät voivat itse lisätä paikkatietoa järjestelmään (OSM 2012). Suomessa Open Street Map on voi- nut vapaasti hyödyntää Maanmittauslaitoksen toukokuussa 2012 avaamia avoimen li- senssin karttoja.

### 2.3 Geotagging

Geotagging on menettely, jossa mediaan lisätään paikkatietoa. Lisäys voi tapahtua jo median luontivaiheessa, kuten kamerassa, tai se voidaan lisätä kohteeseensa jälkeen- päin. Automaattinen lisäys tapahtuu yleensä medialaitteeseen integroidun GPS- paikantimen avulla, joka lisää paikan koordinaatit median metatietoihin (Geotagging 2012).

Koordinaattijärjestelmä perustuu leveys/pituus-arvoihin, jotka lasketaan maapallon nol- la-meridiaanista (-180°) itään pituuden suhteen ja etelänavalta (-90°) pohjoiseen levey-

den suhteen. Koordinaatit voidaan esittää monella eri tavalla käytettävän sovelluksen vaatimusten mukaan. Joissain tapauksissa voidaan mediaan tallettaa myös korkeusasma.

Paikkatiedon liittämällä mediaan voi varomattomasti käytettynä olla ikäviäkin seurauksia. On raportoitu tapauksista, joissa netin kauppapaikoilla esiteltyjen myytävien esineiden kuvien paikkatiedot ovat jääneet poistamatta. Rikollisille tällainen tieto on luonnollisesti liiketoimien kannalta erittäin arvokasta.

## **2.4 Kuvapalvelut**

Kuvapalvelulla (photo sharing, hosting) tarkoitetaan verkossa toimivaa järjestelmää, joka yksinkertaisimmillaan tarjoaa ainoastaan tilaa käyttäjien lataamille kuville. Palvelu voi myös tarjota rajapinnan, jonka avulla järjestelmän sisältöä päästään hyödyntämään erilaisten hakujen ja filttareiden avulla. Kuviin voidaan liittää erilaista metatietoa, kuten paikkakoordinaatteja, tietoa kuvassa olevista kohteista, kameran teknisiä tietoja jne.

Tyypillisesti palveluiden perustoiminnot ovat käyttäjälle ilmaisia, tai oikeammin mainosrahoitteisia, tosin tällöin käyttöön liittyy joitain rajoituksia, kuten kuvakoko tai aikajaksona ladattujen kuvien määrä. Maksullisissa palveluissa nämä rajoitukset joko poistuvat tai ne ovat hyvin lieviä. Tunnettuja kuvapalveluita ovat esim. Panoramio, Flickr, Photobucket ja Imageshack.

### 3 PALVELUT JA RAJAPINNAT

#### 3.1 Palvelun valintakriteerit

Karttapalvelun valinta oli tässä tapauksessa yksinkertaista, sillä eri palveluvaihtoehtoista Google Maps oli saatavana modulina Drupalin nykyiseen jakeluversioon. Valitettavasti moduli ei tue karttapalvelun uusinta 3-versiota, joten toistaiseksi on tyydyttävä käyttämään vanhempaa APIa. Päivitys on kehittäjän mukaan kuitenkin tulossa.

Kuvapalvelujen paremmuutta tai sopivuutta tähän tehtävään piti arvioida jollakin tavoin. Oli päätettävä vähimmäiskriteerit, jotka palvelun ainakin oli täytettävä. Sovelluksen suunniteltujen toimintojen perusteella määrittelin seuraavat ehdot:

- kuvamateriaalia oli oltava runsaasti tarjolla
- palvelun oli tarjottava rajapinta (API) kuvamateriaalin käyttöön
- kuvia oli oltava mahdollista hakea paikkatiedon perusteella

Tutkimistani palveluista Panoramio, Flickr, Photobucket ja Imageshack täyttivät nämä ehdot, mutta työn rajaamiseksi valitsin näistä vain kaksi ensimmäistä. Lisäksi Google Maps tarjosi valmiiksi integroidun tason Panoramion materiaalille.

Löytämäni suomalaiset palvelut oli hylättävä jo pelkästään suppean sisältönsä vuoksi, eikä rajapintaakaan juuri mainostettu etusivuilla. Onkin luultavaa, että monipuolinen ja laajasisältöinen kuvapalvelu pystyy toimimaan taloudellisesti kestäväällä pohjalla ainoastaan erittäin laajan käyttäjäkunnan voimin.

#### 3.2 Tiedonvälitys

Palvelun tarjoama rajapinta määritellään tavallisesti API-dokumentaatioissa, joka sisältää kuvaukset funktioista ja tietorakenteista, joiden avulla palvelun sisältämään materiaaliin voidaan kohdistaa erilaisia toimenpiteitä. Tyypillisesti toimenpiteen suoritettava lauseke muodostuu web-sivun lomakkeen sisällöstä käyttäen API:n kielioppia, ja pyyntö lähetetään joko GET tai POST HTTP-funktion avulla. Haluttaessa voidaan käyttää

asynkronista (AJAX) tiedonvälitystapaa. Palvelun vastauksen muoto riippuu kyselyn parametreista, jotka valitaan API:n kuvauksen ja sovelluksen vaatimusten mukaan. Tavallisimpia tiedonvälitysmuotoja verkkopalveluissa ovat XML ja JSON.

### 3.3 XML

XML on metakieli ja W3C standardi, jolla määritetään rakenteellisia merkkauškieliä. XML:ää käytetään siis jonkin tiedon kuvailemiseen, ja se määrittelee sovelluksessa käytetyn tiedon rakenteen. Kieltä käytetään yleisesti tiedonjakoon verkkoympäristössä, sillä se on tekstipohjainen ja siten riippumaton sovellusalustasta (Introduction to XML).

XML kykenee tarvittaessa esittämään hyvin monimutkaisia tietorakenteita. Usein kuitenkin sovellusten väliselle tiedonsiirrolle riittäisi yksinkertaisempikin malli, ja tähän tarpeeseen kehitettiin JSON. Voidaankin todeta, että XML on omiaan kokonaisten dokumenttien tiedonsiirtoon, kun taas JSON sopii paremmin useimpiin arkisiin ja yksinkertaisiin tehtäviin.

### 3.4 JSON

JavaScript Object Notation eli JSON on tiedonvälityformaatti, joka perustuu nimensä mukaisesti Javascriptin objektimalliin. Se on kuitenkin täysin kieliriippumaton, ja samoin kuin XML, myös tekstipohjainen (JSON). Sen on osuvasti kuvattu olevan "fat-free alternative to XML" (Fat-free JSON).

JSON käyttää kahta eri ohjelmointikielissä hyvin yleistä tietorakennetta:

- avain - arvo -pareista koostuva objekti (associative array, hash table)
- arvoista koostuva taulukko (array)

Alla olevassa koodiesimerkissä 1 ovat käytössä molemmat näistä rakenteista.

```
{
  "avain 1": "arvo",
  "avain 2": [
    "taulukko",
    "kaikenlaisia",
    "arvoja"
  ]
}
```

### Koodiesimerkki 1. Yksinkertainen JSON-rakenne

Rakenteita yhdistämällä voidaan toteuttaa hyvinkin mutkikkaita kokonaisuuksia, jotka kuitenkin ovat ihmiselle helposti ymmärrettävässä muodossa. Ohjelmakoodissa objekti-muodossa olevaa dataa on yksinkertaista lukea ja kirjoittaa.

## 3.5 AJAX

AJAX on lyhenne sanoista Asynchronous JavaScript and XML. Toiminnallisesti määriteltynä AJAX on keino hakea sisältöä palvelimelta asiakkaalle ilman näkyvää sivun uudelleenlatausta tai -päivitystä. Yksinkertaisimmillaan siihen sisältyy seuraavien teknologioiden käyttö (Chaffer 2011, 139):

- JavaScript (vuorovaikutus)
- XMLHttpRequest-objekti (XHR)
- Palvelimella saatavissa oleva useimmiten XML tai JSON muotoinen data
- JavaScript (datan käsittely)

JavaScriptin avulla sivu reagoi käyttäjän toimiin vuorovaikutteisesti ja suorittaa pyynnön XMLHttpRequest-objektin avulla. XMLHttpRequest-objekti on toteutettu selaimessa, ja se mahdollistaa pyynnot palvelimelle asynkronisessa muodossa, ts. muut selaimessa käynnissä olevat prosessit eivät keskeydy toiminnon ajaksi. Pyydetty data voi olla staattisessa muodossa, eli valmiina palvelimella, tai se voidaan koostaa pyynnön parametrien perusteella. Selain vastaanottaa esim. XML- tai JSON-tyyppisen datan ja käsittelee sen haluttuun muotoon.

## 4 DRUPAL-YMPÄRISTÖ

### 4.1 Rakenne ja alusta

Drupal on PHP-pohjainen avoimen lähdekoodin sisällönhallintajärjestelmä eli CMS, joka sisältää kaikki tavalliselta verkkopohjaiselta julkaisujärjestelmältä edellytettävät toiminnot. Drupal on toteutettu moduli-periaatteella, joka mahdollistaa järjestelmän laajentamisen kontrolloidusti. Drupalia voidaankin paremmin luonnehtia termillä CMF (Content Management Framework), eli järjestelmään voidaan lisätä uusia toimintoja joustavasti muuttuvien tarpeiden mukaan.

Drupal on rakennettu PHP:tä käyttäen, ja tämänhetkinen julkaisuversio 7.15 vaatii vähintään PHP version 5.2. Historiallisista ja toiminnallista syistä johtuen järjestelmä on pääosin kirjoitettu proseduraalista ohjelmointitapaa käyttäen, jossa lukuisat funktiot vastaavat toiminnoista (Butcher 2010, 9). Oliopiiirteitä löytyy kuitenkin monista alijärjestelmän moduleista.

Tietojen tallennukseen käytetään tietokantaa, mutta järjestelmän voi suhteellisen vapaasti valita. Päätös on kuitenkin tehtävä ennen asennusta. Tavallisin valinta on edelleen MySQL, mutta monia muitakin vaihtoehtoja on olemassa: MariaDB, PostgreSQL, SQLite jne. Lisäksi tarvitaan luonnollisesti palvelinohjelmisto, joka Drupalin tapauksessa on lähes aina Apache. Muita tuettuja ovat Nginx ja Microsoftin IIS (System requirements 2012).

Kehitystyössä on vaivattominta asentaa jokin tarjolla olevista valmiista XAMP-tyyppisistä palvelin-tietokanta-php ympäristöistä. Paketti on saatavilla yleisimmille käyttöjärjestelmille. Jos palvelimen asennus ei jostain syystä kuitenkaan ole toivottua, tai edes mahdollista, on varteenotettava vaihtoehto käyttää jotakin virtualisointisovellusta, kuten Oraclen VirtualBox, jonka avulla on helppo asentaa useita erilaisia käyttöjärjestelmiä virtuaalisina. Siten esim. Windowsiin voidaan tuoda Linux-ympäristössä toimiva www-palvelin, jonka asennus, käyttö tai poistaminen eivät vaikuta mitenkään käyttöjärjestelmän asetuksiin tai toimintaan.

## 4.2 Asennus

### 4.2.1 Apache, MySQL, PHP

Edellä kuvatun XAMP-paketin asennus on suoraviivaista, mutta muutamiin kohtiin on syytä kiinnittää huomiota. Madel (2012) ehdottaa muutoksia seuraaviin PHP:n asetuksiin, jotka löytyvät tiedostosta php.ini (koodiesimerkki 2):

```
; Maximum execution time of each script, in seconds
max_execution_time = 60
; Maximum amount of time each script may spend parsing request data
max_input_time = 120
; Maximum amount of memory a script may consume
memory_limit = 128M
; Show all errors, except for notices and coding standards warnings
error_reporting = E_ALL & ~E_NOTICE
```

Koodiesimerkki 2. Muutokset tiedoston php.ini asetuksiin (Madel 2012).

Myös MySQL tarvitsee joitakin muutoksia asetuksiinsa tiedostossa my.ini (koodiesimerkki 3). Erityisesti asetuksen max\_allowed\_packet väärä oletusarvo on Madelin (2012) mukaan ollut syypäänä moniin vaikeasti paikannettaviin virheilmoituksiin. Editoinnin jälkeen on palvelin käynnistettävä uudelleen, jotta muutokset tulevat voimaan.

```
# * Fine Tuning
#
key_buffer = 16M
key_buffer_size = 32M
max_allowed_packet = 16M
thread_stack = 512K
thread_cache_size = 8
max_connections = 300
```

Koodiesimerkki 3. Muutokset my.ini tiedoston asetuksiin (Madel 2012).



### 4.2.2 Drush

Drush on Drupaliin kehitetty komentoriviltä toimiva työkalu, jonka avulla monet tehtävät voidaan hoitaa kätevästi. Työkalun nimi on johdettu enemmän tai vähemmän osuvasti sanoista Drupal ja shell. Drush.org (<http://drush.ws/>) tarjoaa sivustollaan kaikkea mahdollista tukea käyttäjille.

Drushin voi ladata Drupalin virallisilta sivuilta (<http://drupal.org/project/drush>) muille käyttöjärjestelmille, mutta Windowsille on parempi käyttää valmista asennuspakettia, joka löytyy myös Drushin sivuilta. Kun Drush on asennettu, voidaan toimivuus testata kirjoittamalla komentoriville `C:\>drush --help`, joka esittää lyhyen yhteenvedon työkalun käytöstä ja komennoista.

### 4.2.3 Drupal

Drupalin asennuksessa on tehokasta käyttää Drushia. Koodiesimerkki 4 esittelee kaikki tarvittavat komennot toimivan Drupal-järjestelmän asentamiseksi. Tässä komennot kirjoitetaan komentoriville `www-palvelimen juuri-hakemistossa`.

```
drush dl drupal --drupal-project-rename=drupal_test -y
cd drupal_test
cp sites/default/default.settings.php sites/default/settings.php
chmod -R o+w sites/default
```

Koodiesimerkki 4. Drupalin asennus komentoriviltä.

Tuloksena on `www-palvelimen juuren alihakemisto drupal_test`, johon ensin ladattiin viimeisin Drupalin jakeluversio, jonka jälkeen `drupal_test/sites/default/settings.php` kopioitiin oletusasetuksista, sekä sallittiin rekursiivisesti kirjoitusoikeudet ryhmälle 'muut' (other) hakemistoon `drupal_test/sites`. Ennen asennuksen jatkamista on vielä luotava tietokanta, jota Drupal tulee käyttämään. Tehtävään voitaisiin käyttää XAMP-paketin mukana asentunutta phpMyAdmin-sovellusta, mutta yleensä on nopeampaa antaa tarvittavat käskyt suoraan komentoriviltä (koodiesimerkki 5).

```
mysql -u root -p
DROP DATABASE IF EXISTS testdb;
CREATE DATABASE testdb;
CREATE USER 'test_user'@'localhost' IDENTIFIED by 'pw';
GRANT ALL ON testdb.* TO 'test_user'@'localhost' IDENTIFIED BY 'pw';
FLUSH PRIVILEGES;
```

#### Koodiesimerkki 5. Tietokannan luonti Drupalin asennuksessa.

Koodiesimerkissä 5 käynnistetään ensin MySQL-shell-asiakasohjelma ja kirjaudutaan root-käyttäjänä sisään. Luodaan sen jälkeen tietokanta nimeltään testdb ja käyttäjä test\_user, sekä annetaan käyttäjälle kaikki oikeudet luotuun kantaan. Nyt valmiina on sekä tietokanta, että oikeudet kantaan omistava käyttäjä. Nämä tiedot on annettava Drupalin seuraavassa asennusvaiheessa, joka käynnistyy selaimella osoitteessa [http://localhost/drupal\\_test/](http://localhost/drupal_test/). Avautuvalla sivulla Drupalin asennus etenee vuorovaikutteisesti, kunnes kaikki tarvittavat tiedot on syötetty järjestelmään.

Drupal on nyt perusasetuksissaan valmis. Tässä vaiheessa voi olla hyvä ottaa varmuuskopio käytetystä tietokannasta, jotta järjestelmän voi helposti palauttaa alkutilaansa. On myös hyvä suunnitella ja päättää sopiva varmistusstrategia sivuston tulevaa käyttöä ajatellen. Koodiesimerkki 6. näyttää, miten varmuuskopiointi voidaan suorittaa komentoriviltä.

```
md db
mysqldump -u root -p testdb > db/testdb.sql
```

#### Koodiesimerkki 6. Tietokannan varmistus.

Esimerkissä luodaan ensin hakemisto nimeltä db, ja tämän jälkeen kopioidaan mysqldump-sovelluksella tähän hakemistoon haluttu kanta sql-muotoisena. Hakemisto on hyvä luoda kyseessä olevan Drupal-asennuksen sisään sekaannusten välttämiseksi. Koska Drupal kirjoittaa periaatteessa kaiken sisältönsä käytössään olevaan kantaan, on järjestelmän palautus suoraviivaista. Koodiesimerkki 7 näyttää miten palautus varmistettuihin asetuksiin tapahtuu. Ennen palautusta on kuitenkin ensin poistettava vanha

kanta ja koodiesimerkin 5. mukaan luotava uusi, jossa olemassaolevalle käyttäjälle annetaan kaikki oikeudet. Käyttäjää ei kuitenkaan tarvitse luoda uudelleen.

```
mysql -u root -p testdb < path\to\testdb.sql
```

Koodiesimerkki 7. Tietokannan palautus.

Palautuksessa tietokanta täytetään käymällä läpi varmistetun kannan sql-lausekkeet, jolloin tuloksena on alkuperäiset tiedot sisältävä kanta. Virheiden välttämiseksi on usein tapahtuvat tietokannanvarmistus- ja palautusoperaatiot hyvä kirjoittaa komentojonotiedoiksi. Varsinkin testausvaiheessa tämä säästää sekä aikaa että hermoja.

## 4.3 Arkkitehtuuri

### 4.3.1 Ydin

Ydin eli Drupal Core Libraries tarjoaa kaikki perustoiminnot, joita järjestelmä tehtäviä suorittaakseen vaatii. Tiedostolistauksen näkökulmasta kaikki muut paitsi siteshakemiston alla olevat tiedostot ovat Drupalin ydintä. Ydin huolehtii siitä, että kaikki on kunnossa järjestelmän osalta, mutta ei ota kantaa itse sivupyynnön prosessointiin (Butcher 2010, 13). Sivupyyntö voi käsittää mitä tahansa CRUD-toimintoja, tai käyttäjän sisään- ja uloskirjauksen.

Sivupyynnöllä voidaan katsoa olevan erityinen elinkaari, jonka aikana eri modulit käsittelevät pyyntöä ja palautettavaa sisältöä. Ydin valvoo tätä prosessia ja tarjoaa moduleille mahdollisuuden osallistua pyynnön käsittelyyn. Modulit käyttävät ns. koukkufunktioita ilmoittaakseen ytimelle, että kysytty toiminto on toteutettu modulissa ja valmis otettavaksi mukaan prosessiin. Ytimen käytössä olevat koukkufunktiot on määritelty tiedostossa `system.api.php`.

### 4.3.2 Koukkufunktiot

Koukkufunktioiden nimeämiskäytäntö perustuu kaavaan `hook_funktion_nimi()`, jossa hook-osa korvataan modulin nimellä. Koukkufunktioiden suoritusjärjestys pyynnön kohteena olevan materiaalin rakentamisen elinkaareissa alkaa funktiosta `hook_init()` ja päättyy funktioon `hook_exit()`. Init-funktiossa asetetaan tavallisesti globaaleja muuttujia, joita tarvitaan myöhemmin pyynnön käsittelyssä, ja exit-funktiossa suoritetaan järjestelmän siivoukseen liittyviä tehtäviä, kun sisältö on jo lähetetty selaimelle (Hooks Drupal API 2012). Muut sivupyyntöön liittyvät koukkufunktiot suoritetaan näiden välissä jokainen vuorollaan, kunnes prosessi on valmis ja sisältö lähetetty vastaanottajalle. Kaikki modulit voivat myös tarjota omia koukkufunktioitaan, jolloin muut modulit voivat käyttää näitä hyväkseen ja pystyvät näin vaikuttamaan pyynnön prosessointiin juuri halutussa kohdassa elinkaarta.

### 4.3.3 Moduilit

Kaikki Drupalin perusasennuksen mukana tulevista moduleista eivät ole valmiiksi aktivoituja, vaan käyttäjä voi itse valita, mitä toimintoja hän järjestelmältään haluaa. Esimerkiksi blog, forum ja contact form ovat kaikki valmiina käyttöön, mutta vaativat modulien käyttöönoton aktivoinnin admin-paneelistä.

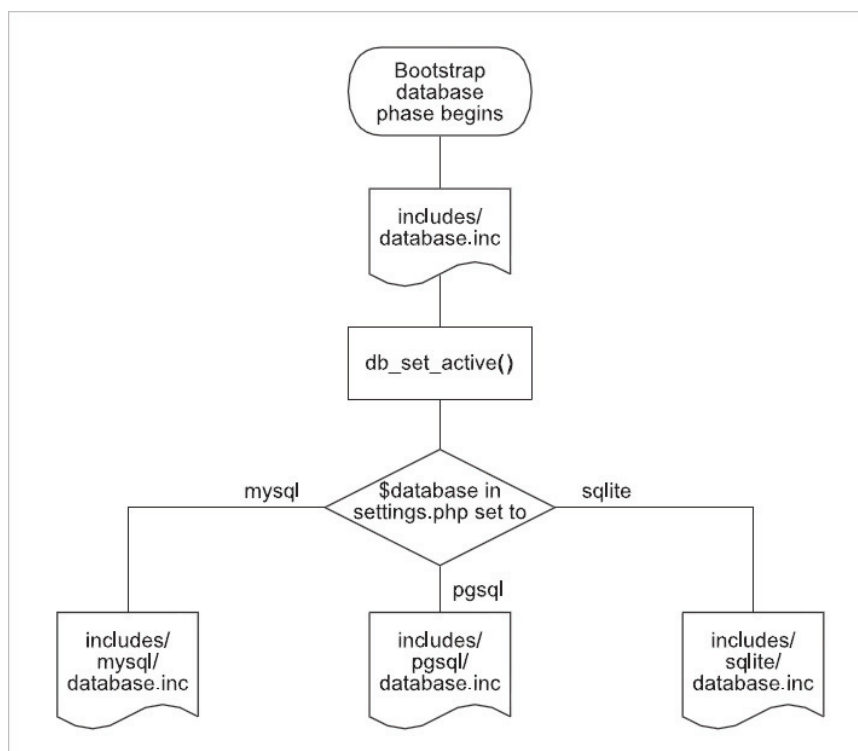
Osa moduleista on kuitenkin välttämättömiä, eikä niiden poistaminen ole käyttöliittymän kautta mahdollistakaan. Tällaisia moduleita ovat:

- filter                      sisällön suodatustoiminnot
- node                        sisällön perusyksikkö
- system                    yleinen sivuston hallinta ja konfigurointi
- text                        yksinkertainen tekstikenttä
- user                        käyttäjänhallinta

#### 4.3.4 Tietokanta

Järjestelmässä on mahdollista käyttää lähes mitä tahansa tietokantaa, sillä PHP:n PDO-kirjasto toimii abstraktiotasona tietokannan ja Drupalin välillä. Jos käytössä on jokin eksoottinen kanta, voidaan tarvittava ajuri kirjoittaa itse noudattaen Drupalin kehittäjäsvuston ohjeita. MySQL lienee kuitenkin kaikkein yleisin vaihtoehto, ja on käytössä myös tässä työssä. Drupalissa tietokanta sisältää tiedot kaikesta järjestelmässä olevasta materiaalista, menetelmät sisällön koostamiseen sekä eri käyttäjien oikeudet sisältöön (Tomlinson 2010, 89).

Drupal ottaa automaattisesti yhteyden asetuksissa määritettyyn tietokantaan ns. bootstrap-prosessissa, joka suoritetaan aina järjestelmän saadessa pyynnön asiakkaalta (kuva 1).



Kuva 1. Tietokantatuki bootstrap-prosessissa (Tomlinson 2010, 91)

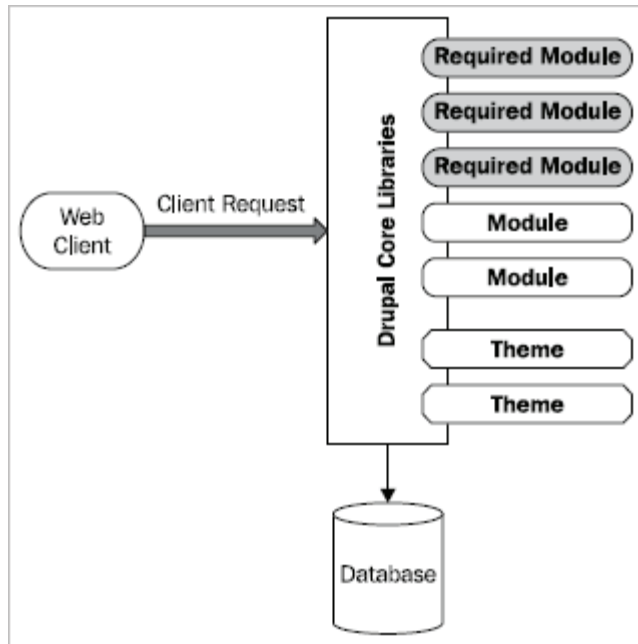
Kattavan tietokantatuken takia on uutta modulia kirjoitettaessa määriteltävä kaikki modulin vaatimat tietokantataulut yhdenmukaisella tavalla. Määrittely tehdään modulin install-tiedostossa, jossa taulujen piirteet annetaan Schema API:n mukaisesti. Näin Drupal osaa kääntää syntaksin käytössä olevan tietokantajärjestelmän ymmärtämään muotoon (Tomlinson 2010, 99).

## 4.4 Toiminta

### 4.4.1 Pyyntö ja vastaus

Kuvassa 2. esitetään Drupalin rakenne ja toiminnot selaimen lähettäessä sivupyyntöä. Butcher (2010, 12) kuvaa proseduurin seuraavasti:

- Käyttäjä kirjoittaa selaimen osoitekenttään URLin `http://example.com/node/123` ja painaa Enter.
- Selain pyytää palvelimelta osoitteessa `example.com` dokumenttia `/node/123`.
- Palvelin toteaa, että pyyntö tulee käsitellä PHP:n avulla, ja siirtää pyynnön PHP-tulkille.
- PHP-tulkki suorittaa `index.php` -tiedoston (kaavio 2.) sisältämän skriptin ja ohjaa sille osoitteen `/node/123`.
- Drupal suorittaa ns. bootstrap-prosessin, alustaa resurssit ja menu-järjestelmän avulla etsii tavan käsitellä osoitteesta `/node/123` löytyvää dokumenttia.
- Node-moduli vastaa käsittelypyyntöön hakemalla noden, jonka tunnus on 123 (node on yksi Drupalin alijärjestelmistä tai -moduleista, jolla tarkoitetaan talletettua sisältöyksikköä). Noden sisältö haetaan yleensä tietokannasta.
- Theme-funktiot vastaavat noden sisällön esittämisestä ja muuntamisesta tyylitiedostojen avulla HTML-muotoon.
- Drupalin ydin (core) antaa vielä eri moduleille mahdollisuuden käsitellä sisältöä ja lähettää tämän jälkeen saadun tuloksen takaisin selaimelle.
- Selain muuntaa HTML- ja CSS-sisällön visuaaliseen muotoon ja suorittaa vielä mahdolliset javascript-toiminnot.
- Käyttäjä näkee dokumentin.



Kuva 2. Drupalin komponentit (Butcher 2010, 11)

#### 4.4.2 Index.php ja bootstrap-prosessi

Bootstrap-prosessi suoritetaan joka sivupyynnön yhteydessä, sillä kaikki pyynnot ohjautuvat tiedostolle index.php. Jos mod\_rewrite-moduli on asennettuna Apachessa, voidaan käyttöön ottaa Clean Url-asetus, jolloin edellämainittu osoite on muodossa `http://example.com/node/123`. Ilman modulia Clean Url-asetusta ei voida käyttää, ja tällöin osoite näkyy muodossa `http://example.com/?q=node/123`, joka on myös Drupalin omassa prosessoinnissaan käyttämä muoto. Index.php on hyvin kompakti ja sisältää vain muutaman rivin koodia (kuva 3). Koodin toiminnot voidaan kuvailla lyhyesti:

- Kutsutaan `getcwd()`-funktioita, joka palauttaa Drupalin asennuksen hakemistopolut palvelimen tiedostojärjestelmässä, ja asetetaan tämä vakion `DRUPAL_ROOT` arvoksi.
- Luetaan `bootstrap.inc` -tiedosto. Tiedostossa määritetään noin 80 funktiota ja lähes 40 vakiota, joista seuraavaksi `bootstrap()` funktiota kutsutaan vakioargumentilla `DRUPAL_BOOTSTRAP_FULL`.

- Viimeiseksi kutsutaan `menu_execute_active_handler()` -funktiota, joka etsii oikean tavan käsitellä pyyntöä.

```
/**
 * Root directory of Drupal installation.
 */
define('DRUPAL_ROOT', getcwd());

require_once DRUPAL_ROOT . '/includes/bootstrap.inc';
drupal_bootstrap(DRUPAL_BOOTSTRAP_FULL);
menu_execute_active_handler();
```

Kuva 3. Index.php -tiedoston sisältö

Bootstrap-prosessi koostuu kahdeksasta vaiheesta (kuva 4), jotka suoritetaan yksi toisensa perään while-silmukassa siten, että kaikki vaiheet aina parametrina annettuun saakka suoritetaan järjestyksessä. Siten funktion `drupal_bootstrap` (`DRUPAL_BOOTSTRAP_FULL`) suoritus automaattisesti käy läpi ensin kaikki muut bootstrap-vaiheet ja vasta viimeisenä parametrina annetun.

```
static $phases = array(
    DRUPAL_BOOTSTRAP_CONFIGURATION,
    DRUPAL_BOOTSTRAP_PAGE_CACHE,
    DRUPAL_BOOTSTRAP_DATABASE,
    DRUPAL_BOOTSTRAP_VARIABLES,
    DRUPAL_BOOTSTRAP_SESSION,
    DRUPAL_BOOTSTRAP_PAGE_HEADER,
    DRUPAL_BOOTSTRAP_LANGUAGE,
    DRUPAL_BOOTSTRAP_FULL,
);
```

Kuva 4. Bootstrap-vaiheet funktiossa `bootstrap()` tiedostossa `bootstrap.inc`

1. Configuration-vaiheessa luetaan `settings.php` sites/default kansiota ja asetetaan tärkeimmät globaalit muuttujat, kuten käytettävä tietokanta jne.
2. Page-cache-vaiheessa yritetään löytää pyydetty sivu välimuistista.
3. Database-vaiheessa tietokantaan liittyvät luokat ja funktiot käydään läpi, mutta varsinaista yhteyttä kantaan ei vielä avata.
4. Variables-vaiheessa haetaan muuttujat tietokannan variable-tilusta laajaan `$conf` taulukkoon, johon ne talletetaan avain-arvo-pareina, ja voidaan lukea tai kirjoittaa myöhemmin `variable_get()` ja `variable_set()` funktioilla.



5. Session-vaiheessa rekisteröidään session-käsittelijät ja kytketään sessionit kirjautuneisiin käyttäjiin.
6. Page-header vaiheessa muodostetaan HTTP-headerit ja lähetetään ne output-bufferiin odottamaan prosessin valmistumista. Lisäksi tässä vaiheessa annetaan moduleille ensimmäinen mahdollisuus osallistua sivun muodostukseen kutsuamalla koukkufunktiota `hook_boot()`.
7. Language-vaiheessa rekisteröidään moni-kielisen sivuston kieliasetukset ja annetaan koukkufunktion `hook_language_init()` käsitellä esim. kieliriippuvaisia muuttujia.
8. Full-vaiheessa kaikki käyttöön tulevat modulit ladataan ja theme-määritykset luetaan. Viimeiseksi suoritetaan koukkufunktio `hook_init()`, jonka avulla mm. asetetaan tarvittavia globaaleja muuttujia ja asetetaan päivämäärän esitysmuoto vastaamaan kieliasetuksia.


Näiden vaiheiden jälkeen Drupal on valmis rakentamaan pyydetyn materiaalin ja lähettämään sen eteenpäin. Kuvassa 3. listattu `menu_execute_active_handler()` -funktio saa menu-järjestelmän kautta tiedon callback-funktiosta, joka pystyy vastaamaan pyydetyn tyyppisen materiaalin koostamisesta. Luvussa 5.2.2 esittelen tarkemmin, miten tämä käytännössä tapahtuu.

## 5 MODULIN KUVAUS

### 5.1 Toiminta

Moduli tarjoaa uuden sisältötyypin perusasennuksessa tarjolla olevien Article- ja Basic page-tyyppien rinnalle nimeltään Point of interest (POI). Käyttäjä lisää uutta sisältöä järjestelmään valitsemalla Add content navigointiblokista, jonka jälkeen valittavana ovat kaikki asennetut sisältötyypit. Klikattaessa Point of interest-tyyppiä, avautuu valitun sisältötyypin editointisivu uuden sisällön lisäämiseen, jossa tarvittavat tiedot voidaan syöttää lomakkeen kenttiin. Paikkatiedon lisäämiseksi POI tarjoaa mahdollisuuden hakea koordinaatit Google Maps-kartan avulla. Koska kaikki kuvamateriaali ei sisällä paikkatietoa, voidaan sisältöön lopuksi liittää vielä omia merkkeamattomia kuvia.

Siirryttäessä editointitilasta pääsivulle, lisätty sisältö on ilmestynyt sisältölistaan ns. teaser-muodossa, jossa ainoastaan otsikko ja lyhennelmä sisällön tekstimateriaalista ovat näkyvissä. Otsikko on linkki itse sisältöön, joka koostuu otsikosta, sisältötekstistä, Flickr-kuvien asetuslomakkeesta (kuva 5) sekä kahdesta Google Maps-objektille varustusta alueesta, joiden välissä on kolme painiketta kontrolloimassa eri komponenttien näkyvyyttä karttapohjalla. Ylempi Maps-alue on varattu kartalle ja alempi Street View-näkymälle (kuva 6). Lisäksi Flickrin kuvahaun tuloksille on varattu alue, joka vielä tässä vaiheessa on tyhjä.



▼ Flickr options

Number of images per page

10 ▼

Size of images

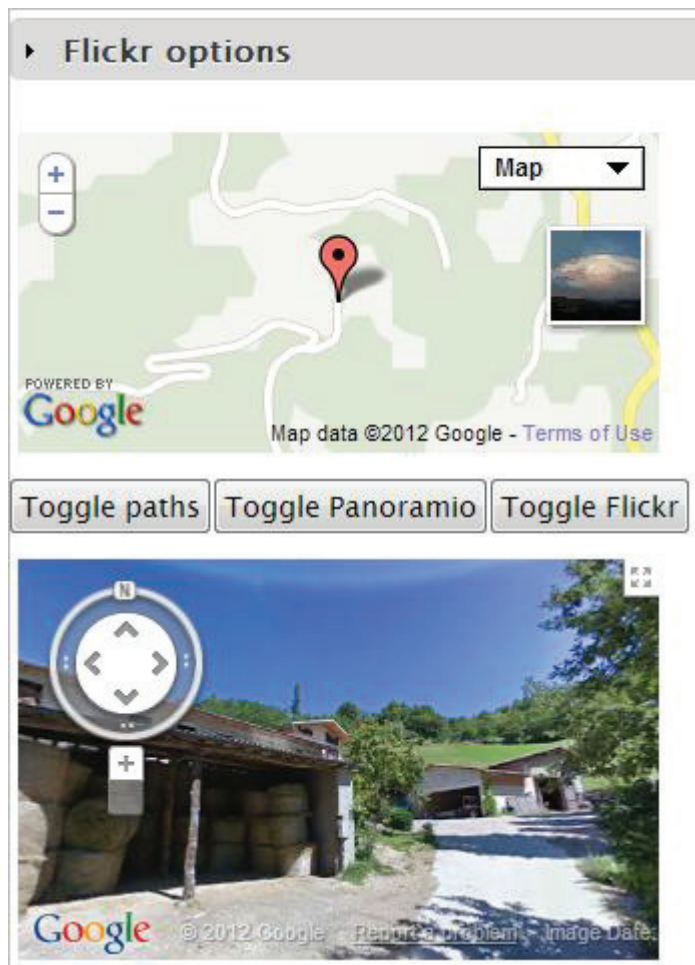
Medium ▼

Kuva 5. Flickr-asetukset.

Käyttäjä voi liikkua kummassa tahansa alueessa markkerin ja näkymän päivittyessä molemmissa vastavuoroisesti. Painikkeet ovat on/off-tyyppisiä, eli näkymän tila vaihtuu kahden eri vaihtoehdon välillä. Toggle paths näyttää karttapohjalla reitit, joilla Googlen Street View kuvauskalusto on vieraillut. Näkymä on oletuksena pois päältä. Toggle Pa-

noramio piilottaa Panoramion tarjoaman kuva-aineiston, ja alkuasetuksena on kuvien näyttäminen. Toggle Flickr toimii samoin, mutta käsittelee Flickrin aineistoa.

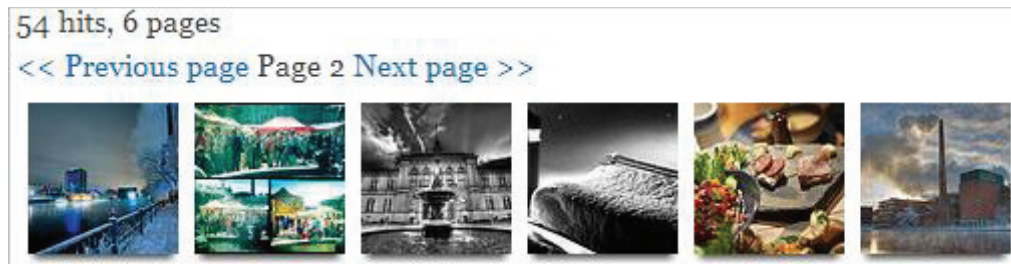
Kuvan 5. Flickr-asetukset määrittävät, miten monta ja miten suurta kuvaa haetaan yhdellä kyselyllä. Kartan nurkkakoordinaatit määräävät sen alueen, jonka sisällä etsittyjen kuvien tulee sijaita. Haun tulokset esitetään alueella karttojen yläpuolella asetuksissa valitun määrän mukaisesti, sekä lisäksi karttapohjalla Panoramion tapaan. Lisäksi jos Flickr-kuvia löytyy enemmän kuin mitä sivuasetukseen on määritetty, näytetään navigointipainikkeet kuvasivujen välillä liikkumiseen (kuva 7).



Kuva 6. Google Maps, painikkeet ja Street View.

Flickrin kuvahaku tapahtuu taustalla asynkronisesti AJAXin avulla. Klikkaus karttaan liikauttaa vastaavasti punaista markkeria ja lähettää Flickrin API:n mukaisen flickr.photo.search-metodin mukaisen kyselyn. Lauseen tärkeimmät parametrit saadaan karttapohjan nurkkakoordinaateista ja asetuslomakkeen kentistä. Halusin minimoida

hakujen määrän kuormittaakseni Flickrin palvelua mahdollisimman vähän, minkä vuoksi sovellus ei esimerkiksi hae kuvia ennen kuin käyttäjä on klikannut karttaa, ja siten ilmoittanut haluavansa tehdä haun.



Kuva 7. Flickr ja navigointitoiminnot.

Kuvat esitetään kartalla hakutuloksista löytyneiden koordinaattien perusteella pieninä thumbnail-kuvina, jotka ovat linkkejä avautuvaan infoikkunaan. Panoramio tarjoaa valmiin kuvatason Google Mapsiin, mutta Flickrin tapauksessa esitysmuoto joudutaan rakentamaan käyttäen Flickrin API-funktioita.

Eri kuvapalveluiden tarjoaman materiaalin erottamiseksi toisistaan karttapohjalla oli keksittävä jokin menetelmä. Tyydyin tässä esittämään Flickrin kuvat hieman Panoramioa pienempinä (kuva 8), mutta ratkaisu ei ole kovin toimiva, jos lisää kuvapalveluita otettaisiin mukaan. Jokin toinen menetelmä olisi siis siinä tapauksessa otettava käyttöön.



Kuva 8. Kuvien esitys karttapohjalla paikkatiedon perusteella.

Kuvien esitysmenetelmät thumbnail-kuvien linkeistä on tässä jätetty avoimeksi, sillä tyypillisesti sovelluksessa käytettäisiin todennäköisesti jotakin Lightbox-tyyppistä modulia, joita Drupaliinkin löytyy lukuisia. Tämä liittyy niin olennaisesti sivuston muuhun

ulkoasuun ja toteutukseen, etten ottanut kantaa asiaan. Jotta sovellus toimisi oletetusti tässä prototyypissä, on kuvat siis syytä avata uuteen ikkunaan, joka tunnetusti tapahtuu klikkaamalla ja painamalla samalla vaihto-näppäintä

## 5.2 Integrointi Drupaliin

Modulin käyttöönotto edellyttää, että Drupal voi löytää tarvittavat tiedostot oletetusta paikasta. Drupalin asennushakemiston alla kansio sites/all/modules tai tämän alihakemistot ovat sallittuja paikkoja, joihin modulutiedostot on mahdollista sijoittaa. Drupalin admin-osion modules-sivu näyttää kaikki järjestelmässä olevat modudit, sekä käytössä olevat, että käyttämättömät. Käyttöönotto tapahtuu asettamalla moduli enabled-tilaan ja tallentamalla asetukset (kuva 9).

ENABLED	NAME	VERSION	DESCRIPTION
<input checked="" type="checkbox"/>	Point of interest		Point of interest. Requires: GMap ( <a href="#">enabled</a> ), Location ( <a href="#">enabled</a> ), GMap Location ( <a href="#">enabled</a> ), Location CCK ( <a href="#">enabled</a> )

Kuva 9. Modulin käyttöönotto.

Usein moduli tarvitsee muiden modulien toimintoja tehtävissään. Nämä modudit listataan jokaiselta modulilta vaadittavassa info-tiedostossa dependencies-tilaukkuun (kuva 10). Tiedostossa määritellään muitakin modulin asennuksen kannalta olennaisia tietoja. Tarvittavat modudit näkyvät myös Description-kentässä (kuva 9). Jos modudit on listattu info-tiedostossa, osaa Drupal ottaa ne käyttöön automaattisesti, eli tässä tapauksessa rastittamalla Point of interest -moduli enabled-tilaan saadaan kaikki sen tarvitsemat muut modudit myös käyttöön olettaen, että ne ovat Drupalin saatavilla.

Modulit poistetaan käytöstä samalla admin/modules -sivulla poistamalla raksi enabled-kohdasta. Tämä ei kuitenkaan vielä poista modulin tekemiä muutoksia tietokantaan, vaan tätä varten on suoritettava poistotoimenpiteet admin/modules/uninstall -sivulla. Tavallisesti jokaisen modulin poistofunktio on määritelty install-tiedostossa, jossa hook\_uninstall -koukkufunktio vastaa tarvittavien toimenpiteiden suorittamisesta.

```

1 name = Point of interest
2 description = "Point of interest."
3 package = groovekonna
4 core = 7.x
5 files[] = poi.install
6 files[] = poi.module
7 files[] = poi.admin.inc
8 dependencies[] = gmap_location
9 dependencies[] = location_cck

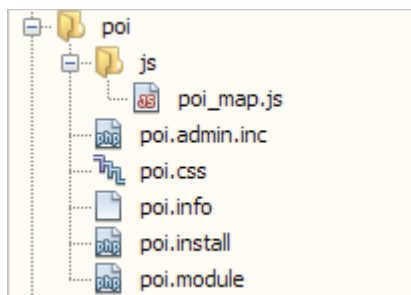
```

Kuva 10. Tiedoston poi.info sisältö.

### 5.3 Tiedostot ja toiminnot

#### 5.3.1 Modulikansio

Modulin kaikki tiedostot ovat modulin nimeä kantavan hakemiston alla (kuva 11). Käyttönoton ja toiminnan kannalta ainakin info, install ja module-päätteiset tiedostot tulee nimetä modulin mukaisesti, jotta Drupal kykenee löytämään tarvitsemansa tiedot. Alihakemistoja voidaan luoda tarpeen mukaan jäsentämään tiedostohierarkiaa. Tässä ainoastaan javascript-tiedosto on siirretty omaan alihakemistoonsa.



Kuva 11. Modulikansion tiedostolistaus.

#### 5.3.2 Tiedosto poi.install

Modulin rakenne on määritelty poi.install-tiedostossa, jossa modulin sisältökentät ja -tyypit on rakennettu Drupalin tapaan array- eli taulukkomuodossa. Koukkufunktio hook\_install() suorittaa tarvittavat modulin määrittelytoiminnot. Tiedostosta löytyy

myös `hook_uninstall()`, jonka tehtävänä on modulia poistettaessa suorittaa järjestelmässä tarvittavat siivoustoimenpiteet.

### 5.3.3 Tiedosto `poi.admin.inc`

Modulin admin-tason konfiguraatio on määritelty tiedostossa `poi.admin.inc`. Liitteessä 1. esitetään modulin `hook_menu()` -funktio, joka tarjoaa reitityksen admin-tason konfiguraatioon osoitteessa `admin/config/poi/settings`. Tiedostossa määritellään lomakkeen kentät ja muut tarvittavat tiedot asetuksille. Asetuskenttiin on syötettävä Flickrin ja Google Mapsin API avaimet sekä otettava käyttöön Location-modulin asetus Google Mapsin käytöstä. Flickrin avainkenttään on toteutettu validointitoiminto, joka tarkistaa syötetyn avainluvun muodon. Lisäksi lomakkeessa on kaksi alasvetovalikkoa, joiden avulla voidaan haluttaessa asettaa Google Mapsin alueiden koot halutun suuruisiksi.

### 5.3.4 Tiedosto `poi.module`

Tiedosto `poi.module` on modulin sydän ja sisältää toiminnan kannalta tärkeimmät funktiot, mm. koukkufunktiot `hook_menu()`, `hook_node_view()` sekä lokitehtäviä suorittavat `hook_insert()`, `hook_update` ja `hook_delete()`, jotka reagoivat poi-tyyppisen sisällön lisäys-, muutos- ja poistotoimiin. Lisäksi tiedostoon sisältyy funktio `poi_search_fn()`, joka suorittaa Flickr-palveluun osoitetut kyselyt. Myös Flickrin kyselyparametrit tarjoava lomake on määritetty tässä tiedostossa.

Toimintojen reititys määritetään `hook_menu()` -funktiossa. Luvussa 4.4.2 bootstrap-prosessin yhteydessä käytiin läpi Drupalin tapaa käsitellä asiakkaalta tulevaa pyyntöä. Esimerkissä käytettiin osoitetta `http://example.com/?q=node/123`, jossa pyyntö kohdistui nodeen, jonka tunniste on 123. Menu-järjestelmä toimii siten, että Drupal pyytää kaikilta käytössä olevilta moduleilta listan tarjolla olevista menu-yksiköistä. Jokainen yksikkö sisältää taulukon, jossa avaimena on polku tarjottavaan toimintoon ja lisäksi muuta tietoa tästä yksiköstä. Liitteessä 1. havainnollistetaan, miten taulukon page callback -avain sisältää tiedon siitä, mitä toimintoja Drupalin tulee suorittaa pyydettyessä tämän yksikön osoitepolkua (Tomlinson 2010, 57). Drupal tallettaa osoitepolkujen ja callback -toimintojen vastaavuudet tietokantaan `menu_router`- tai `menu_links` -tauluun.

Tiedosto `poi.module` sisältää vielä yhden funktion, joka on vastuussa tarvittavien dokumenttirakenteiden lisäämisestä aineistoon, ennen kuin Drupal lähettää sisällön eteenpäin. Koukkufunktio `hook_node_view()` muokkaa noden sisältöä lisäämällä tyylitiedoston `poi.css` ja javascript-toiminnot sisältävän `poi_map.js` tiedoston. Lisäksi kutsutaan funktiota `drupal_add_js()`, jotta saadaan muutamia Drupalissa talletetut muuttujat näkyviin javascript-ympäristössä. Sisältöön lisätään vielä script-tageilla koodia, jossa suoritetaan tiedostossa `poi_maps.js` määritetyn javascript-objektin `initialize`-metodi. Samalla korvataan GMap-modulin tarjoama Google Mapsin esitysmuoto POI-modulin monipuolisemmalla versiolla.

### 5.3.5 Tiedosto `poi_map.js`

Drupal tarjoaa oletuksena jQuery-kirjaston funktiot kehittäjän avuksi. Näin voidaan monia dokumentin rakenteen muokkaukseen tarvittavia käskyjä yksinkertaistaa ja kirjoittaa ytimekkäämpään muotoon. Lisäksi käytössä ovat kaikki muutkin kirjaston edut, kuten selaintuki eri valmistajien tuotteille, tapahtumankäsittely ja AJAX-funktiot.

Tiedosto `poi_map.js` sisältää kaikki ne toiminnot, joiden avulla Google Maps -objekti pystyy esittämään erilaisia sisältötasoja kuvapalvelujen tarjoamasta materiaalista. To-teutuksessa on käytetty suunnittelumallia, jossa pyritään minimoimaan globaalien muuttujien määrä, ja näin parantamaan mallin uudelleenkäyttömahdollisuuksia. Mallia kutsutaan sattuvasti nimellä Module Pattern (Stefanov 2010, 97), ja sen toimintaperiaate selviää koodiesimerkistä 8.

Toiminnot on kääritty anonyymin funktion sisään, joka suoritetaan automaattisesti. Nimiavaruuden määrittävä muuttuja asetetaan osoittamaan funktion palauttamaan objektiin, jolla on omat yksityiset ominaisuutensa ja metodinsa. Näin selvittää ainoastaan yhdellä globaalilla muuttujalla. Objektin julkisiin attribuutteihin ja metodeihin voidaan viitata suoraan muuttujan kautta.



```
// rakenne
var my_obj = (function(){
    // yksityiset metodit ja attribuutit
    var prv_var;
    var prv_method = function(){ ... };
    return {
        // julkinen API
        prop : new Array(),
        init : function(){ ... }
    }
})();
...
// käyttö
my_obj.init();
```

#### Koodiesimerkki 8. Module Pattern.

Kuten jo aikaisemmin mainittiin, Panoramio tarjoaa valmiin tason Maps-objektille, joka kuitenkin on kytkettävä näkyviin Google Maps API:n `addOverlay()` -funktion avulla. Toinen valittu kuvapalvelu oli Flickr, jonka materiaali myös voidaan esittää Panoramion tavoin karttapohjalla. Esitys vaatii kuitenkin hieman enemmän ohjelmointityötä.

Liitteessä 1 esitetty menu-järjestelmän yksikkö `poi_search` on vastuussa oikean funktion suorittamisesta haettaessa materiaalia Flickr-palvelusta. AJAX-toiminnot sisältävälle jQueryn `post()` -funktiolle välitetään muun muassa tieto suoritettavasta funktiosta, eli tässä tapauksessa osoite `'http://localhost/drupal_test/poi_search'`. Tämän osoitteen kohdatessaan Drupal osaa suorittaa callback-avaimen määrittämän funktion `poi_search_fn()`, joka esitetään liitteessä 2. Esimerkkilistaus Flickrin palauttamasta hakutuloksessta on esitetty liitteessä 3.

Flickrin API tarjoaa haun tuloksen tässä tapauksessa JSON-muodossa, joka on yksinkertaista käydä läpi jQueryn `each()` -iteraatiofunktiolla, jossa iteraatiokierroksen käsittelevä callback-funktio voidaan asettaa samassa funktiokutsussa (jQuery API 2012). Hakutuloksen on sisällettävä ainakin kuvan ID sekä arvot `server ID`, `farm ID` ja `secret`, jotta kuvan url voidaan koostaa. Paikkatiedot ovat mukana `latitude` ja `longitude`- arvoina.

Käsittelyn jälkeen saadaan haun tulokset esitettynä html-muodossa. Samalla muodostetaan myös karttapohjalla esitettävät thumbnail-kuvat ja asetetaan ne näkyviin saadun paikkatiedon mukaisesti.

## 5.4 Projektin versionhallinta

Drupalissa versionhallinta on toteutettu käyttäen Git-järjestelmää. Kaikkien modulien lähdekoodi on saatavilla joko valmiina paketteina omilta sivuiltaan tai kloonattavissa Gitin kautta. Itse asiassa koko Drupal-järjestelmän versionhallinta on järjestetty Gitin avulla.

Modulien kehitystyö alkaa useimmiten ns. sandbox-vaiheella, jossa projekti kopioidaan Gitin kautta Drupalin tähän tarkoitukseen varaamaan paikkaan. Sandbox-projekti voidaan myöhemmin korottaa varsinaiseksi moduliksi tarkoin määritellyn hyväksymismenetelyn kautta.

Oma projektini löytyy osoitteesta [http:// drupal.org/ sandbox/ groovekonna/ 1802176](http://drupal.org/sandbox/groovekonna/1802176). Projektin kloonaus omalle työasemalle onnistuu 'git clone --branch master' -komennolla antaen parametriksi projektin osoitteen, sekä kansion nimen, johon paketti sijoitetaan. Osoite on 'http://git.drupal.org/sandbox/groovekonna/1802176.git' ja kansion nimi 'poi'.

## 5.5 Debuggaus ja testaus

Projekti toteutettiin käyttäen integroitua kehitystyökalua nimeltään NetBeans IDE. Kyssessä on hyvin paljon monille käyttäjille tutumman Eclipsen kaltainen kehitysympäristö, joka tarjoaa ennen kaikkea kattavat debuggausominaisuudet. Toimintojen testaus ja virheenjäljitys on erittäin helppoa ja nopeaa. Javascript-toimintojen testaukseen käytin Chromen Developer Toolsin työkaluja.

Varsinaista ohjelmisto- tai yksikkötestausta ei projektissa suoritettu, osin ajanpuutteen, mutta osin myös projektin ominaisuuksien takia, sillä suurin osa toiminnallisuudesta on toteutettu javascriptin avulla, eikä esim. omia php-luokkarakenteita tai uusia tietokanta-

tauluja ollut tarvetta luoda. Chromessa voitiin tehdä määriteltyyn javascript-objektiin erilaisia kyselyjä Console-paneelissa, sekä tutkia syntynyttä dokumenttirakennetta.

Modulin asentamista ja toimintaa testattiin luomalla uusi Drupal-asennus tietokantoi-  
neen ja käyttäjineen, sekä tämän jälkeen kopioimalla moduli paikoilleen. Asennuksen  
tuli sujua virheettömästi ja kaikkien tästä modulista riippuvien modulien tuli asentua  
automaattisesti. Lisäksi testattiin modulin poisto ja tarkistettiin, että kaikki modulin te-  
kemät muutokset tietokantaan poistuivat. Myös Apache-palvelimen `rewrite_module`-  
konfiguraation muutoksen vaikutukset testattiin modulin toiminnassa.

## 6 POHDINTA

Tavoitteena oli kehittää moduli Drupaliin, joka esittäisi paikkatiedon ja eri lähteistä koostuvan kuvamateriaalin samassa yhteydessä käyttäjäystävällisellä tavalla. Voidaan todeta, että tavoite täyttyi vähimmäismitoissaan ja vaaditut toiminnallisuudet kyettiin toteuttamaan.

Päätös ottaa karttapohjaksi Google Maps oli varmaankin nopean kehitystyön kannalta hyvä valinta, sillä Panoramion integraatio Googlen kanssa antoi tavallaan yhden materiaallähteen valmiina ilman merkittävää koodaustyötä. Drupalin GMap-moduli on vielä vailla virallista tukea Google Mapsin uusimmalle versiolle, joten muutoksia on tulevan päivityksen myötä odotettavissa myös muille tästä riippuville moduleille.

Flickr oli toinen valitsemistani kuvapalveluista, ja laajuutensa ansiosta varmasti perusteltu. Olisi kuitenkin ollut mielenkiintoista testata, miten hyvin muiden kuvalähteiden liittäminen mukaan olisi onnistunut Flickristä saatujen kokemusten pohjalta. Koska lähes koko javascript-osio keskittyi yksinomaan Flickrin kuvahaun toteutukseen, olisi tässä ehkä ollut valmis lähtökohta modulimaiseen kuvapalveluintegraatioon.

Voidaan todeta, että opinnäytetyön kannalta merkittävin kehitystyö suoritettiin toisaalta toimintojen integroimisessa Drupaliin, toisaalta taas Flickrin materiaalin tuomisessa karttapohjaan. Monia ulkonäköön ja kuvien esittämiseen keskittyviä seikkoja jouduttiin jättämään sivuun toiminnallisten vaatimusten ollessa etusijalla. Todellisessa käyttöympäristössä sivuston ulkonäkö ja kuvien esitys ovat kuitenkin ensimmäisiä asioita, joihin käyttäjä kiinnittää huomiota. Modulin on myös kyettävä tarjoamaan asetukset erilaisille käyttöympäristöille, kuten mobiililaitteet, tabletit jne. Nämä seikat monien muiden ohella tulevat varmaan ratkaistaviksi modulin jatkokehityksessä.

## LÄHTEET

Butcher, Matt 2010. Drupal 7 Module Development. Olton Birmingham, GBR. Packt Publishing Ltd

Chaffer, Jonathan 2011. Learning jQuery: 3rd ed. Packt Publishing

Fat-free JSON. [www-sivu].[Luettu 9.10.2012].<http://www.json.org/fatfree.html>

Flickr 2012. [www-sivu].[Luettu 6.10.2012].<http://en.wikipedia.org/wiki/Flickr>

Flickr api 2011. [www-sivu].[Luettu 9.10.2012]. <http://www.flickr.com/services/api/>

Geotagging 2012. [www-sivu].[Luettu 8.10.2012].<http://en.wikipedia.org/wiki/Geotagging>

GIS 2012. [www-sivu].[Luettu 5.10.2012].  
[http://en.wikipedia.org/wiki/Geographic\\_information\\_system](http://en.wikipedia.org/wiki/Geographic_information_system)

Google Maps api 2011. [www-sivu].[Luettu 9.10.2012].  
<http://code.google.com/apis/maps/>

History of cartography 2012. [www-sivu].[Luettu 5.10.2012].  
[http://en.wikipedia.org/wiki/History\\_of\\_cartography](http://en.wikipedia.org/wiki/History_of_cartography)

Hooks Drupal API 2012. [www-sivu].[Luettu 11.10.2012].  
<http://api.drupal.org/api/drupal/includes%21module.inc/group/hooks/7>

Introduction to XML 2008. [www-sivu].[Luettu 9.10.2012].<http://www.slideshare.net/pohjus/introduction-to-xml-presentation>

Johnson, Donald S 2007. Meritie - Navigoinnin historia. John Nurmisen Säätiö. Helsinki 2007

JSON. [www-sivu].[Luettu 9.10.2012].<http://www.json.org/>

jQuery API 2012. [www-sivu].[Luettu 18.10.2012]. <http://api.jquery.com/>

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

OSM 2012. [www-sivu].[Luettu 5.10.2012].  
<http://fi.wikipedia.org/wiki/OpenStreetMap>

Panoramio api 2011. [www-sivu].[Luettu 9.10.2012]. <http://www.panoramio.com/api/widget/api.html>

Peacock, Michael 2011. Drupal 7 Social Networking. Packt Publishing

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

Soho 2012. [www-sivu].[Luettu 5.10.2012].  
[http://en.wikipedia.org/wiki/Broad\\_Street\\_Pump#Broad\\_Street\\_pump](http://en.wikipedia.org/wiki/Broad_Street_Pump#Broad_Street_pump)

Stefanov, Stoyan 2010. JavaScript Patterns. O'Reilly Media

System requirements 2012. [www-sivu].[Luettu 9.10.2012].  
<http://drupal.org/requirements>

Tomlinson, Todd 2010. Pro Drupal 7 Development: 3rd ed. Apress

## LIITTEET

### Liite 1. poi.module: modulin toimintojen reititys.

```

/**
 * Implements hook_menu().
 */
function poi_menu() {
  $items = array();

  $items['poi_search'] = array(
    'title' => 'poi search',
    'page callback' => 'poi_search_fn',
    'page arguments' => array(),
    'access arguments' => array('access content'),
    'type' => MENU_CALLBACK
  );

  $items['admin/config/poi'] = array(
    'title' => 'POI',
    'description' => 'Adjust POI options.',
    'position' => 'right',
    'weight' => -5,
    'page callback' => 'system_admin_menu_block_page',
    'access arguments' => array('administer site configuration'),
    'file' => 'system.admin.inc',
    'file path' => drupal_get_path('module', 'system'),
  );

  $items['admin/config/poi/settings'] = array(
    'title' => 'POI settings',
    'description' => 'Change POI settings.',
    'page callback' => 'drupal_get_form',
    'page arguments' => array('poi_admin_settings'),
    'access arguments' => array('administer site configuration'),
    'type' => MENU_NORMAL_ITEM,
    'file' => 'poi.admin.inc',
  );

  return $items;
}

```

## Liite 2. poi.module: Flickr-hakulauseen muodostus.

```

/**
 * Flickr ajax function called from poi_map.js.
 */
function poi_search_fn($argument) {
$my_flickr_api_key = variable_get('poi_flickr_api_key');

$lat0 = $_POST['lat0'];
$lon0 = $_POST['lon0'];
$lat1 = $_POST['lat1'];
$lon1 = $_POST['lon1'];

$bbox = "$lon0,$lat0,$lon1,$lat1";

$url='http://api.flickr.com/services/rest/?method=flickr.photos.search
';
$url.= '&api_key='.$my_flickr_api_key;
$url.= '&tags=?';
$url.= "&bbox=".$bbox;
$url.= '&per_page='.$_POST['numImages'];
$url.= '&page='.$_POST['page'];
$url.= '&extras=geo';
$url.= '&format=json';
$url.= '&nojsoncallback=1';

header('Content-Type:text/json;');
echo file_get_contents($url);
}

```



### Liite 3. Flickr-hakutulos JSON-muodossa.

```
{ "photos":
  { "page":1,
    "pages":43,
    "perpage":10,
    "total":"423",
    "photo":
    [
      { "id":"8054301390",
        "owner":"41959761@N06",
        "secret":"4a130aa9b7",
        "server":"8462",
        "farm":9,
        "title":"2012-10-04-0027",
        "ispublic":1,
        "isfriend":0,
        "isfamily":0,
        "latitude":61.49857,
        "longitude":23.773094,
        "accuracy":"16",
        "context":0,
        "place_id":"hQ_V7V5UVbn_8JQ",
        "woeid":"573760",
        "geo_is_family":0,
        "geo_is_friend":0,
        "geo_is_contact":0,
        "geo_is_public":1
      },
      { "id":"8040858341",
        ...
      },
      ...
      ...
      ...
      { "id":"7919246064",
        ...
      }
    ]
  },
  "stat":"ok"
}
```