LAHDEN AMMATTIKORKEAKOULU
Lahti University of Applied Sciences

WEB-KEHITYS DRUPAL-YMPÄRISTÖSSÄ

Case: Hevostalli-portaali

LAHDEN
AMMATTIKORKEAKOULU
Tekniikan ala
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka
Opinnäytetyö
Kevät 2014
Markus Ylitalo

Lahden ammattikorkeakoulu
Tietotekniikka

YLITALO, MARKUS:

Web-kehitys Drupal-ympäristössä
Case: Hevostalli-portaali

Ohjelmistotekniikan opinnäytetyö, 41 sivua

Kevät 2014

TIIVISTELMÄ

Opinnäytetyö toteutettiin yhteistyössä lahtelaisten JS Median ja Korpimedia Oy:n kanssa. Työn tarkoituksena oli kehittää hevostalliyrittäjät ja ratsastajat yhdistävä verkkoportaali. Hevostalli-portaalin ideana on tarjota yrittäjille palvelu, jossa he voivat mainostaa omia palveluitaan ja ratsastajille helppokäyttöinen kanava ratsastuspalveluiden löytämiseen. Palvelu käyttää hyödykseen tallien paikkatietoja, jolloin käyttäjän on mahdollista löytää lähimmät palvelut oman sijaintinsa perusteella.

Portaali toteutettiin käyttäen avoimeen lähdekoodiin perustuvaa Drupal 7 -julkaisujärjestelmää. Tallien paikkatietojen geokoodauksessa käytettiin hyödyksi Googlen tarjoamaa Geocoding-rajapintaa. Portaalin hakuun ja sijainninhallintaan liittyvät toiminnallisuudet toteutettiin omina Drupal 7 -moduuleinaan.

Tämän opinnäytetyön teoriaosa käsittelee Drupal 7:n asentamista, keskeisimpiä komponentteja ja moduulikehityksessä tarvittavia tekniikoita. Moduulikehityksen esittely toimii pohjana käytännön osuudelle.

Drupal 7 moduulikehitykseen tutustutaan case-esimerkkien avulla työn käytännön osiossa. Case-esimerkkeinä toimivat toteutetut haku- ja sijainti-moduulit. Moduuleiden esittelyssä on painotettu Drupal 7 -moduulikehitykselle ominaisia piirteitä.

Asiasanat: Drupal, web-kehitys, koukkufunktiot, geokoodaus

Lahti University of Applied Sciences
Degree Programme in Information Technology

YLITALO, MARKUS: Web development in Drupal environment
Case: Horse stable portal

Bachelor's Thesis in Software Engineering, 41 pages

Spring 2014

ABSTRACT

This Bachelor's Thesis was made in cooperation with companies called JS Media and Korpimedia Oy. The purpose of the thesis was to develop a web portal which aims to connect the horse stable entrepreneurs with the equestrians. The idea is to offer a service for entrepreneurs, where they can advertise their own services, and where equestrians can search for a best place for horse riding. The service utilizes the geographic information of the stables, which helps equestrians to find the best place to ride for their purposes.

The portal was implemented using an open source based content management system called Drupal 7. Google's Geocoding API was used for geocoding the geographic information of the stables. The functionalities related with searching and location management were implemented in their own Drupal 7 modules.

The theoretical part of the Thesis deals with installing Drupal 7, its most essential components, and techniques used in Drupal module development. The introduction to module development serves as a basis for the practical part.

The practical part of the Thesis introduces Drupal 7 module development through case examples. The search module and the location module which were developed during the Thesis act as case examples. Introduction of these modules emphasizes the features that are specific to Drupal 7 module development.

Key words: Drupal, web development, hooking, geocoding

SISÄLLYS

1 JOHDANTO

Eri aihealueita käsittelevien verkkoportaalien määrä on kasvanut huomattavasti viimeisen vuosikymmenen aikana. Verkkoportaali on palvelu, joka yhdistää eri tahojen sidosryhmät toisiinsa. Yritysmaailmassa portaali käsitetään tavanomaisesti eräänlaisena markkinapaikkana, joka yhdistää palveluntarjoajat, ja asiakkaat keskenään.

Tämän opinnäytetyön käytännönosio käsittelee hevostalliyrittäjiä ja niiden asiakkaat yhdistävää verkkoportaalia. Työn hevostalli-portaali on lahtelaisen digimediatoimisto JS Median ideoima. Yritys toimi kehitysvaiheessa tiiviissä yhteistyössä lahtelaisen Korpimedia Oy:n kanssa.

Ennen opinnäytetyön aloittamista portaali oli jo päätetty toteuttaa Drupal 7 -julkaisujärjestelmää käyttäen. Drupal 7 oli asennettuna WWW-palvelimelle, ja järjestelmän alustavat konfiguraatiot oli tehty. Portaaliin oli myös luotu muutamia esittelysivuja hevostalleista, joiden kanssa portaali toimii kehitysvaiheessa yhteistyössä.

Portaalista haluttiin tehdä paikkatietoon perustuva, joten palvelusta täytyisi pystyä hakemaan hevostalleja niiden sijainnin perusteella, ja hakutulosten tulisi olla mahdollista järjestää sekä rajata käyttäjän oman sijainnin mukaan. Tämä tekisi oman lähiseudun hevostallien löytämisestä helppoa. Käyttäjällä olisi myös mahdollisuus määrittää itse oma sijaintinsa, mikäli tämä ei haluaisi jakaa sijaintitietojaan palvelulle.

Tämän opinnäytetyön tavoitteena on kehittää hevostalli-portaalin paikkatietoon perustuva hakutyökalu sekä sijaintitietoja hallitseva moduuli. Koska toteutusalustaksi valittu Drupal 7 ei ollut entuudestaan tuttu, ja se määrittelee moduuleiden kehittämisessä käytettävät metodit hyvin tiukasti, tutkimusongelmaksi muodostui Drupal 7 -moduulikehityksen ymmärtäminen.

Opinnäytetyön teoriaosassa käydään lävitse Drupal 7:n asentaminen ja käyttöönotto. Tämän jälkeen esitellään Drupal 7:n keskeisimmät komponentit, ja kerrotaan moduulikehityksessä käytettävistä tekniikoista.

Käytännön toteutusosa käsittelee Drupal 7 julkaisujärjestelmän moduulikehitystä, ja esittelee case-esimerkkejä hevostalli-portaaliin kehitetyistä moduuleista. Koska työn aloitusvaiheessa Drupal 7 oli jo asennettuna kehityspalvelimelle, on järjestelmän asentaminen esitelty teoriaosassa, mutta rajattu pois toteutusosasta.

2 DRUPAL

2.1 Mikä Drupal?

Drupal on avoimeen lähdekoodiin perustuva sisällönhallintajärjestelmä ja sovelluskehitysalusta, joka on kirjoitettu PHP ohjelmointikielellä (Drupal Suomi, Mikä on Drupal? 2013). Se on hyvin laajennettavissa useilla saatavilla olevilla laajennusmoduuleilla, ja niiden ansiosta Drupal sopii niin yksinkertaisten kuin laajojenkin verkkosivujen toteuttamiseen. Aktiivisen kehittäjäyhteisön myötä myös uusia laajennusmoduuleita kehitetään jatkuvasti.

Jos tavoitteena oli luoda dynaaminen verkkosivusto, joka sisältää esimerkiksi paljon lomakkeita, blogin ja keskustelualueen, oli kehittäjän hallittava ainakin HTML:ää, mahdollisesti JavaScriptiä ja CSS:ää, sekä skriptaamiseen PHP:tä tai ASP:tä, ennen sisällönhallintajärjestelmien yleistymistä. Jos tietoja haluttiin vielä taltioida istuntojen väliseksi ajaksi, oli hallittava SQL-kyselykieli, jonka avulla tietoja voidaan tallentaa tietokantaan ja myös lukea sieltä. (Beighley 2010, 9.)

Drupalin yhtenä tavoitteena onkin tarjota työkalu käyttäjälle, joka ei välttämättä hallitse edellä mainittuja kieliä. Drupalilla voi luoda verkkosivuja kirjoittamatta riviäkään koodia. (Beighley 2010, 9.)

Sivuston luojan näkökulmasta Drupal on helposti kustomoitavissa ja erittäin skaalautuva. Siinä on valmiina useita ominaisuuksia, joita voi kytkeä päälle tai pois yhdellä hiiren klikkauksella. Sivustoille voidaan myös asettaa teemoja valmiista pohjista tai niitä voidaan luoda itse. Lisää ominaisuuksia on saatavilla kolmansien osapuolten kehittämillä laajennusmoduuleilla, joiden avulla on mahdollista luoda esimerkiksi verkkokauppa tai vaikka kuvagallerioita (Beighley 2010, 9). Moduulikehitykseen palataan tämän työn toteutusosassa.

Drupalia käyttävät maailman laajuisesti monet suuret verkkosivustot ja -portaalit, kotimaisista mainittakoon YLE:n, Uuden Suomen ja Nelosen verkkosivut (Drupal Suomi, Kuka käyttää Drupalia? 2013).

2.1.1 Järjestelmävaatimukset ja asentaminen

Drupal 7 on mahdollista asentaa mille tahansa web-palvelimelle joka tukee PHP-ohjelmointikieltä. Suositeltuja palvelinohjelmistoja ovat Apache, Ngnix ja Microsoft IIS. Tässä työssä käsiteltävä hevostalli-portaali on toteutettu Apache-palvelimelle, joka on osa LAMP-palvelinympäristöä. LAMP on lyhenne sanoista Linux-Apache-MySQL-PHP, mutta P-kirjaimella voidaan viitata myös Perl- tai Python-ohjelmointikieliin (ONLamp.com, LAMP: The Open Source Web Platform 2001).

Tietokannaksi Drupal 7:lle suositellaan MySQL 5.0.15 -versiota tai uudempaa, PostgreSQL 8.3 -versiota tai uudempaa ja SQLite 3.3.7 -versiota tai uudempaa. Lisäksi tietokannalle tulee olla asennettuna PDO-ajuri (PHP Data Objects). Myös Microsoft SQL Serveriä tai Oraclen tuotteita on mahdollista käyttää. (System requirements 2013.)

Tietokantaan on suositeltavaa luoda Drupal 7:ää varten oma käyttäjänsä, jolla on vähintään seuraavat oikeudet: SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, INDEX, ALTER. (Database Server 2013.)

Käytettävän PHP-julkaisun versio tulee olla vähintään 5.2.5 tai uudempi, ja käytössä olevan tietokannan mukainen tietokantayhteys-laajennus (mysql, mysqli, pgsql) täytyy olla asennettuna ja kytkettynä päälle. PHP:n välimuistin käyttöraja Drupal 7 ytimelle täytyy olla asetettuna vähintään 32 megatavuun. Kuitenkin jos käytössä on Drupalin yleisimpiä laajennusmoduuleita, kuten CCK (Content Construction Kit) tai Views, käyttörajan täytyy todennäköisesti olla 64 megatavua tai enemmän. (Database Server 2013.)

Vaadittavien PHP:n asetusten ja laajennusten tarkistaminen onnistuu esimerkiksi PHP:n omalla phpinfo() -funktiolla. Palvelimelle luodaan PHP-tiedosto sisältäen kuvion 1 mukainen skripti hakemistoon, johon WWW-selaimella on pääsyoikeus. Selaimella tarkasteltaessa skripti näyttää PHP:n konfiguraation sekä käytössä olevat laajennukset.

```
<?php
phpinfo();
?>
```

KUVIO 1. Esimerkki phpinfo() -funktion käytöstä (drupal.org 2013, Viewing PHP settings using phpinfo())

Drupalin asentaminen aloitetaan lataamalla haluttu Drupalin versio palvelimelta. Siirrytään hakemistoon, johon WWW-selaimella on pääsyoikeus. Unix-ympäristössä tämä on tavallisesti /var/www/html. Drupal 7:n asennuspaketti ladataan käyttäen esimerkiksi wget-komentoa kuvion 2 mukaisella tavalla.

```
wget http://ftp.drupal.org/files/projects/drupal-7.0.tar.gz
```

KUVIO 2. Esimerkki Drupalin asennuspaketin lataamisesta palvelimelle käyttäen wget-komentoa

Latauksen valmistuttua asennuspaketti puretaan ja hakemisto nimetään halutun mukaiseksi. Lopuksi poistetaan ladattu asennuspaketti palvelimelta. Kuviossa 3 on esitetty komennot asennuspaketin purkamiseksi, hakemiston uudelleennimeämiseksi ja asennuspaketin poistamiseksi.

```
tar -zxf drupal-7.0.tar.gz
mv drupal-7.0 drupal-example
rm drupal-7.0.tar.gz
```

KUVIO 3. Drupal-paketin purkaminen ja hakemiston uudelleennimeäminen

Drupal 7 -asennuspaketin mukana tulee asennus-skripti, joka ajamalla tehdään sivuston oikean toiminnan kannalta välttämättömät konfiguraatiot. Ennen asennus-skriptin ajamista täytyy käytettävälle tietokantapalvelimelle luoda käyttäjä, ja tietokanta.

Uusi tietokanta luodaan kuvion 4 osoittamalla komennolla, jossa ’username’ on MySQL-käyttäjä, jolla on CREATE sekä GRANT oikeudet, ja ’databasename’ on luotavan tietokannan nimi. (Step 2: Create the database 2013.)

```
mysqladmin -u username -p create databasename
```

KUVIO 4. Käsky tietokannan luomiseen

Tietokannan luomisen jälkeen kirjaudutaan MySQL-palvelimelle. Kuviossa 5 on esitetty komento Drupal 7:n vaatimien oikeuksien asettamisesta MySQL-käyttäjälle, jossa ’databasename’ on aiemmin luodun tietokannan nimi, ’username’ on käyttäjä jolle oikeudet annetaan, ja ’password’ on käyttäjälle asetettava salasana. (Step 2: Create the database 2013.)

```
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, INDEX, ALTER
ON databasename.*
TO 'username'@'localhost' IDENTIFIED BY 'password';
```

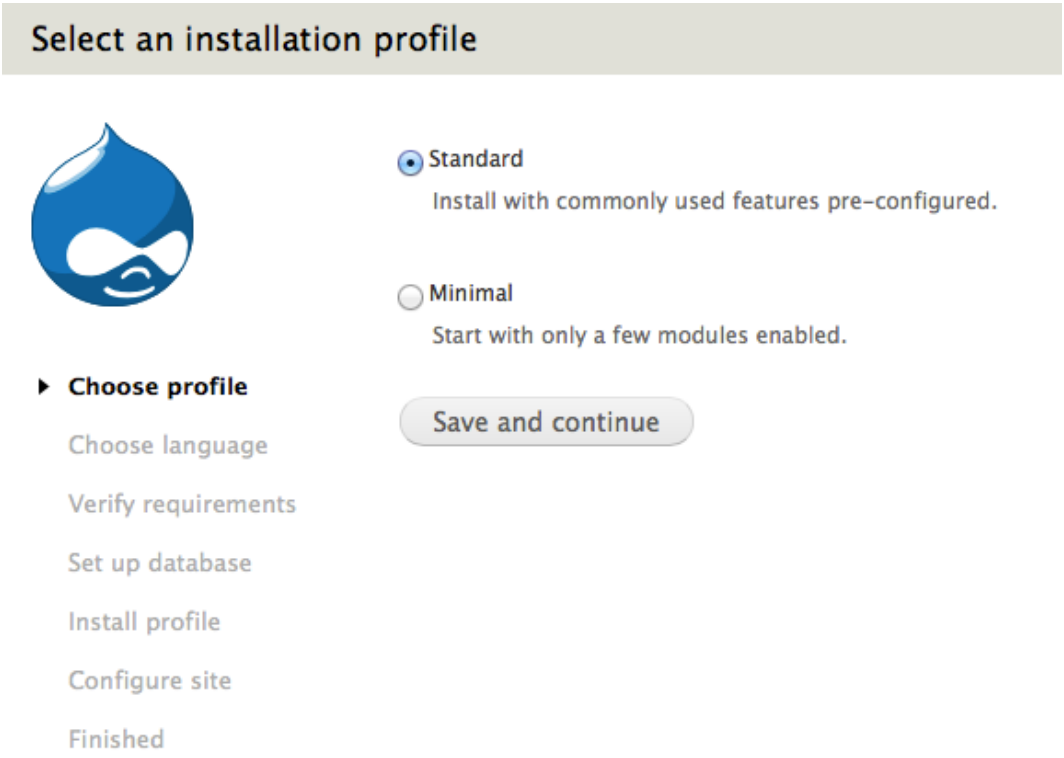
KUVIO 5. Oikeuksien asettaminen MySQL-käyttäjälle

Kun tietokantaan liittyvät konfiguraatiot on tehty, kopioidaan asennuspaketin mukana tullut default.settings.php-tiedosto, ja nimetään se settings.php-nimiseksi. Asennus-skriptin käyttäjältä kysymät konfiguraatiot tallennetaan settings.php-tiedostoon, joten WWW-palvelimella täytyy olla siihen luku- ja kirjoitus-oikeudet. Kuviossa 6 on esitetty komento tiedoston kopioimiseksi. (Step 3: The settings.php file 2013.)

KUVIO 6. Drupal 7:n asetus-tiedoston kopioiminen

Asennus-skriptin ajaminen aloitetaan siirtymällä WWW-selaimella osoitteeseen, josta Drupal-asennus löytyy. Selain näyttää kuvion 7 mukaisen aloitusruudun, josta valitaan asennettavaksi joko *Standard*, tai *Minimal* -profiili. (Step 4: Run the installation script 2012.)

Standard-profiili sisältää valmiiksi asennettuna yleisimpiä sisällön tyyppejä, kuten *Article* ja *Page*. Lisäksi valmiiksi on asennettuna joukko hyödyllisiä laajennusmoduuleita. *Minimal*-profiili on suunnattu enemmän kokeneille Drupal-kehittäjille, jotka haluavat itse määrittää sivustossa käytettävät sisällön tyypit, sekä laajennusmoduulit. (Step 4: Run the installation script 2012.)



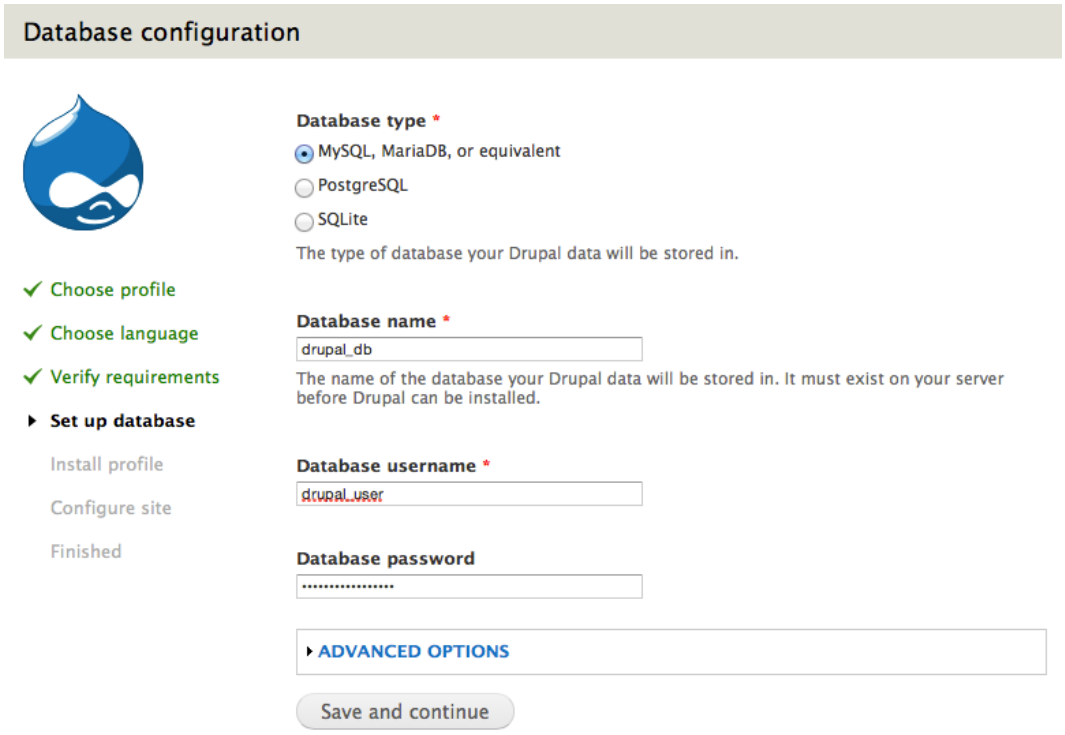
KUVIO 7. Drupal 7 -asennusvelhon aloitusruutu

Asennus-skriptin seuraavassa vaiheessa valitaan asennuksen käyttämä kieli. Englanti on valittuna oletuksena, mutta muiden kielten asentaminen on mahdollista asennus-skriptin tarjoaman ohjeiden avulla. Mikäli WWW-palvelin on konfiguroitu oikein, ja PHP:lle on asennettu kaikki Drupal 7:n vaatimat laajennukset, asennus-skripti ohittaa *Verify requirements* –osion, ja siirtyy suoraan *Set up database* –osioon. Jos palvelimen konfiguraatiossa tai laajennuksissa havaitaan puutteita, siitä informoidaan käyttäjää.

Seuraavassa kuviossa (KUVIO 8) on esitetty Drupal 7:n vaatimien tietokanta-asetusten syöttäminen. *Database name* -kenttään syötetään aiemmin luodun tietokannan nimi, *Database username* -kenttään luodun MySQL-käyttäjän nimi, ja *Database password* -kenttään MySQ-käyttäjälle määritelty salasana.

Advandec options -osio sisältää kentät MySQL-palvelimen osoitteen ja portin määrittämiseksi. Jos näitä kenttiä ei muuteta, asenneus-skripti olettaa, että MySQL-palvelin sijaitsee samassa osoitteessa kuin WWW-palvelin ja että

MySQL-palvelimen käyttämä portti on 3306. Kilkkaamalla *Save and continue* -nappia asennus-skripti aloittaa asentamisen.



KUVIO 8. Drupal 7 -asennusvelhon lomake tietokanta-asetuksia varten

Mikäli profiilin asentamisessa ei tapahtunut virheitä, asennus-skripti näyttää *Configure site* -osion, jossa luodaan sivuston pääkäyttäjän tunnukset ja annetaan sivustolle nimi. Osiossa on myös mahdollista määritellä WWW-palvelimen maakohtainen sijainti sekä aikavyöhyke. Seuraavassa osiossa asennus-skripti ilmoittaa asennuksen onnistumisesta. Asennuksen jälkeen on tietoturvan kannalta syytä muuttaa settings.php-tiedoston oikeudet kuvion 9 mukaisella komennolla. (Step 4: Run the installation script 2012.)

```
chmod 644 sites/default/settings.php
```

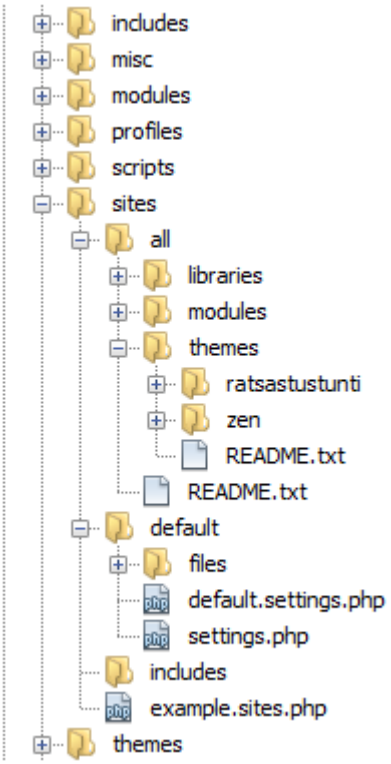
KUVIO 9. Komento settings.php-tiedoston oikeuksien muuttamiseen

2.1.2 Hakemistorakenne

Drupal 7:n hakemistorakenne muodostuu Drupalin ytimestä sekä itse sivuston tarvitsemista tiedostoista. Kuviosta 10 voidaan havaita, että sites-hakemiston yläpuolella sijaitsevat hakemistot muodostavat Drupalin ytimen, ja sites-hakemiston sisällä sijaitsevat tiedostot ovat verkkosivuston spesifisiä tiedostoja.

Yhdellä Drupal-asennuksella voidaan toteuttaa verkkosivustot useammalle domainille. Se onnistuu luomalla sites-hakemiston sisälle esimerkiksi esimerkki1.com niminen hakemisto, joka sisältää samat tiedostot ja hakemistot kuin default-hakemisto. Lisäksi hakemisto voi sisältää modules- ja themes-hakemistot, joihin sijoitetaan verkkosivun spesifiset teemat ja laajennusmoduulit. Hakemiston ’all’ alihakemistoihin sijoitetaan kaikkien domainien yhteiset moduulit ja teemat. Default-hakemiston tulee sisältää files-hakemisto, sekä settings.php-tiedosto siltä varalta, että käytettävälle domainille ei ole määritelty omaa hakemistoa. WWW-palvelimella täytyy olla luku- ja kirjoitus-oikeudet files-hakemistoon, koska sinne sijoitetaan esimerkiksi kaikki sisällönhallinan kautta lisätyt kuvat. (Setup of sites directory for multi-site 2010.)

Kuvion 10 mukaisessa toteutuksessa ei käytetä useampaa kuin yhtä domainia, joten kaikki sivuston tarvitsemat tiedostot sijaitsevat all-hakemistossa.



KUVIO 10. Drupal 7 -hakemistorakenne

2.1.3 Sisällön tyypit

Kuten muidenkin sisällönhallintajärjestelmien, myös Drupal 7:n ydinidea perustuu sisällön tuottamiseen. Drupal 7:llä sisältöä voidaan luoda usean eri sisällön tyypin *(content type)* kautta, ja uusia sisällön tyyppejä voidaan itse luoda rajattomasti. Drupal 7:n perusasennus sisältää valmiina kaksi sisällön tyyppiä: artikkelin ja tavallisen sivun.

Jokainen luotu artikkeli tai sivu tallennetaan järjestelmässä rakenteeseen, jota kutsutaan kirjoitukseksi *(node)*. Muita sisällön tyyppejä voivat olla esimerkiksi blogi-kirjoitus tai vaikkapa kysely. Kaikki Drupal 7:n hallintapaneelin kautta tuotettu sisältö muodostuu käytännössä kirjoituksista. Kirjoituksia eivät kuitenkaan ole esimerkiksi blogi-kirjoitusten kommentit, koska ne eivät ole sivuston ylläpitäjien tuottamaa sisältöä. (Beighley 2010, 79.)

Jokainen sisällön tyyppi muodostuu siis kirjoituksesta. Jokaiselle kirjoitukselle muodostuu järjestelmässä näkymä ja uniikki osoite, johon järjestelmässä voidaan viitata sisäisillä linkeillä. Kirjoituksen näkymä on muokattavissa sen mukaan,

minkä tyyppistä sisältöä sillä on tarkoitus esittää. Sisällön tallentaminen kirjoituksiin tapahtuu eri tyyppisillä kentillä *(field)*, ja jokaiselle kentän tyypille on määritelty omat lomake-elementtinsä *(form widget)*, jolla dataa syötetään tai tallennetaan.

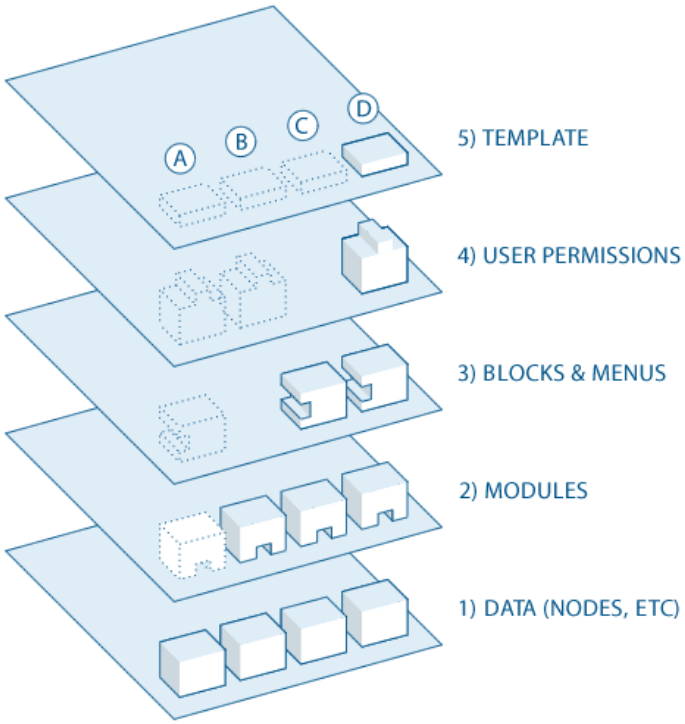
Kenttiä voidaan liittää kirjoitukseen vapaasti mielivaltainen määrä, joten kirjoitukset voidaan käsittää erityyppisten kenttien kokoelmina. Kaikki Drupal 7 perusasennuksen mukana tulevat kenttätyypit, ja niille valittavissa olevat lomake-elementit, on esitetty taulukossa 1.

TAULUKKO 1. Drupal 7 perusasennuksen kenttätyypit (drupal.org 2010)

Field	**Form widget**
Boolean	Check boxes / radio button Single on / off checkbox
Decimal	Text field
File	File
Float	Text field
Image	Image
Integer	Text field
List (float)	Select list Check boxes / radio button
List (numeric)	Select list Check boxes / radio button
List (text)	Select list Check boxes / radio button
Long text	Text area (multiple rows)
Long text and summary	Text area with a summary
Term reference	Select list Check boxes / radio button Autocomplete term widget (tagging)
Text	Text field

Lisää kehittäjäyhteisön luomia kenttätyyppejä on ladattavissa internetistä, tai niitä voidaan kehittää myös itse. Suuri kenttätyyppi-valikoima tekee omien sisällön tyyppien luomisesta erittäin joustavaa, minkä ansiosta kirjoituksilla voidaan esittää lähes minkä tyyppistä sisältöä tahansa.

Sisällön tyyppien tarkoitusta voi havainnollistaa esimerkin kautta, jossa puhelimen yhteystiedot toteutettaisiin Drupal-sivustona. Yhteystietoja varten luotaisiin uusi sisällön tyyppi nimeltään ”Yhteystieto”. Tälle uudelle tyypille määriteltäisiin tallennettavat tietueet, kuten nimi ja puhelinnumero. Nimeä ja puhelinnumeroa varten Yhteystieto-tyyppiin liitettäisiin näille tietueille soveltuvat kentät. Kun Yhteystieto-tyyppi on saatu toteutettua, voitaisiin kaikki puhelimen yhteystiedot syöttää sivustolle käyttäen tätä uutta sisällön tyyppiä. (Working with content types and fields 2009.)



KUVIO 11. Drupal 7 järjestelmän viisi pääkerrosta (drupal.org 2014.)

Kuvio 11 kuvaa tiedon välitystä järjestelmän eri kerroksien välillä. Drupal 7:n ydin muodostuu viidestä pääkerroksesta, joiden käyttäytyminen suhteessa toisiinsa on hyödyllistä ymmärtää selkeämmän kokonaiskuvan saamiseksi. (The Drupal overview 2014.)

Kuviossa pohjimmainen kerros kuvaa kirjoitusten kokoelmaa – datavarastoa, joka on koko järjestelmän perusta. Ennen kuin sivulla voidaan näyttä mitään, täytyy se ensin syöttää datana järjestelmään. (The Drupal overview 2014.)

Seuraavassa kerroksessa ylöspäin sijaitsevat moduulit – toiminnalliset liitännäiset, jotka voivat olla joko osa Drupalin ydintä, tai kolmansien osapuolien toimittamia. Moduulien avulla muokataan esimerkiksi sisällön tyyppien sisältämiä kenttiä. (The Drupal overview 2014.)

Kolmannesta kerroksesta löytyvät lohkot *(blocks)* ja valikot *(menus)*. Lohkojen yleisin tarkoitus on tarjota tuloste moduulien välittämälle datalle. Lohkot voidaan määritellä esittämään dataa monella eri tapaa, kuten myös näkymään vain tietyillä sivuilla tai vain tietyille käyttäjille. Valikot ovat järjestelmän navigaattoreita, jotka määrittelevät näytettävän sisällön kullekin polulle. (The Drupal overview 2014.)

Seuraava kerros sisältää käyttäjäoikeudet, jossa rajataan, mitä erityyppiset käyttäjät saavat tehdä ja nähdä. Ylin kerros käsittää sivuston esittävän teeman *(theme)*, joka muodostuu pääasiassa XHTML:stä, CSS:stä, ja joukosta PHP-muuttujia, jotka ohjaavat Drupalin tuottaman sisällön määrättyihin pakkoihin teeman sisällä. (The Drupal overview 2014.)

Edellä kuvattu alhaalta ylöspäin kulkeva virta ohjaa Drupalin toimintaa. Jokainen matkan varrella oleva kerros tulee olla toiminta kunnossa, jotta sisältö välittyy pohjimmaisesta datavarastosta ylimmäiseen näyttöteemaan. Kuvion 11 A-sarake kuvaa tilannetta, jossa moduuli on asennettu, mutta sitä ei ole vielä aktivoitu. Tästä syystä datan esittäminen katkeaa moduuli-kerrokseen.

Sarakkeessa B on kuvattu tilanne, jossa moduuli on asennettu ja aktivoitu, mutta moduulia vastaavaa lohkoa ei ole asetettu näkymään. C-sarakkeessa moduuli ja lohko ovat asetettu oikein, mutta käyttäjäoikeudet estävät tiedon välittymisen käyttäjän nähtäville.

Ennen kentän määrittelyä uudelle sisällön tyypille on syytä ottaa huomioon muutamia seikkoja. Kentälle annetaan kaksi erityyppistä nimeä: käyttöliittymässä näytettävä nimi ja koneluettava nimi. Käyttäjille näytettävää nimeä voi muuttaa luomisen jälkeenkin, mutta koneluettavaa nimeä ei.

Kentän tyypin tulisi soveltua mahdollisimman hyvin tallennettavalle datalle, koska yhteen kenttään voidaan tallentaa vain yhden tyyppistä dataa, kuten numeroita. Kentän tyyppi ei niin ikään ole vaihdettavissa luomisen jälkeen.

Joillekin kenttätyypeille on valittavissa muutamia erilaisia lomake-elementtejä, joiden avulla data syötetään kenttään. Lomake-elementin tulisi vastata mahdollisimman hyvin kentän käyttötarkoitusta esimerkiksi lista-tyyppiselle kentälle, jossa halutaan mahdollistaa useamman kuin yhden arvon valitseminen, radio-nappeja ei kannata valita lomake-elementiksi.

Viimeinen huomioon otettava seikka on kenttään tallennettavien arvojen määrä. Jokaiseen kenttään voidaan tallettaa yksi arvo, ennalta määritelty määrä arvoja tai rajaton määrä arvoja. Tätä asetusta voidaan muuttaa jälkeenpäin, mutta se voi aiheuttaa tietojen katoamista, jos kenttään on jo tallennettu dataa. (Working with content types and fields 2009.)

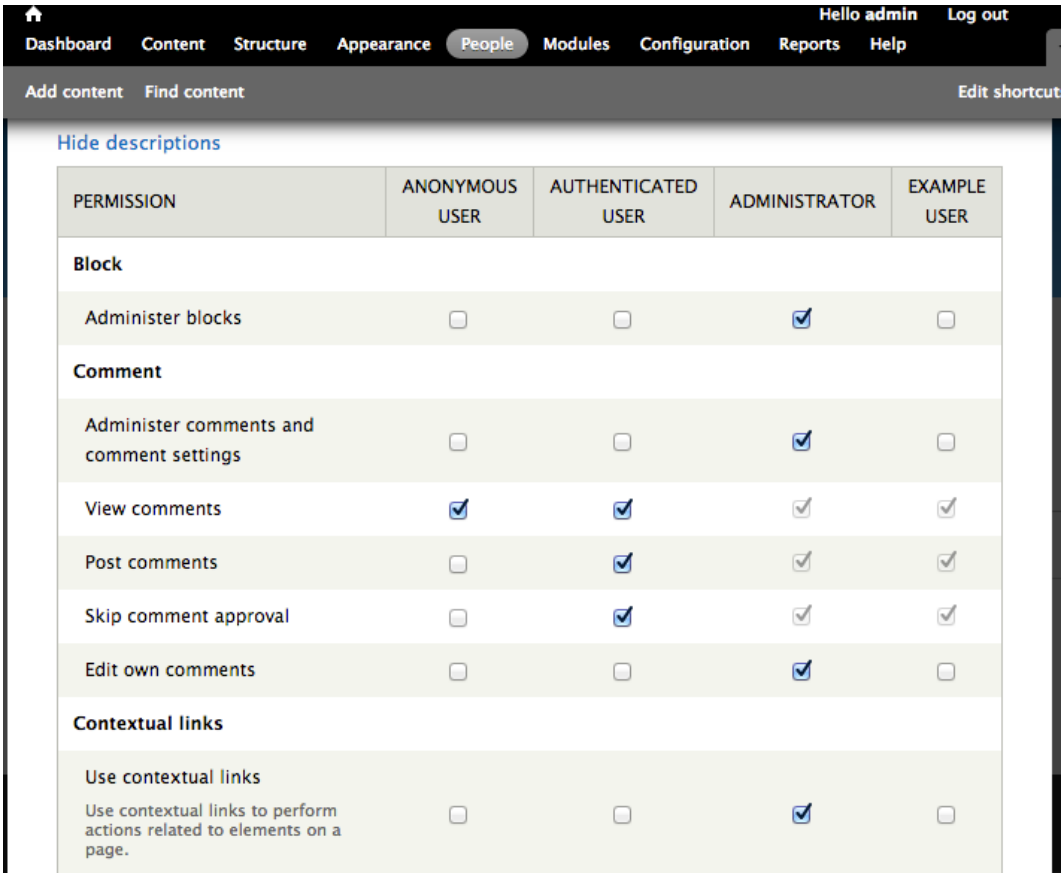
2.1.4 Käyttäjät ja oikeudet

Olennaisena osana sisällönhallintajärjestelmiin kuuluvat myös käyttäjät sekä niiden roolit ja oikeudet. Jo sivuston toiminnallisuutta suunniteltaessa olisi hyvä miettiä, millaisia erilaisia käyttäjiä tarvitaan, ja mitkä ovat niiden oikeudet. Sallitaanko esimerkiksi anonyymiltä käyttäjältä sisällön kommentoiminen, vai täytyykö käyttäjän tyytyä vain katselemaan. Käyttäjälle tulisi antaa vain sallittujen tehtävien vaatimat oikeudet, ei yhtään enempää. (Mercer 2010, 118.)

Käyttäjien oikeuksista puhuttaessa, on helpompaa lähestyä asiaa käyttäjäroolien kautta: Drupal 7:ssä käyttäjärooli määrittelee joukon oikeuksia, joita kaikki rooliin kuuluvat käyttäjät noudattavat. Asennuksen jälkeen järjestelmään on luotu jo valmiiksi kolme eri käyttäjäroolia: anonyymi ja tunnistautunut käyttäjä, sekä

pääkäyttäjä. Anonyymillä käyttäjällä tarkoitetaan käyttäjää, joka vain selaa sivustoa kirjautumatta sisään. Anonyymi ja tunnistautunut käyttäjä ovat järjestelmätasolla lukittuja, joten niitä ei ole mahdollista poistaa. Uusia rooleja on mahdollista lisätä mielivaltainen määrä. (Mercer 2010, 121.)

Kuviossa 12 on esitetty kuvankaappaus Drupal 7:n käyttäjäoikeuksienhallinnasta. Kuvion taulukossa kolme ensimmäistä käyttäjää ovat järjestelmään valmiiksi luodut käyttäjäroolit, ja viimeinen, *EXAMPLE USER*, on ylläpitäjän lisäämä käyttäjärooli. Kuvakaappauksesta voidaan havaita, että jokainen uusi käyttäjärooli perii samat oikeudet kuin tunnistautunut käyttäjä, eli *AUTHENTICATED USER*. Tämä tarkoittaa sitä, että jokainen uuden käyttäjäroolin käyttäjä perustuu tunnistautuneeseen käyttäjään.



KUVIO 12. Drupal 7:n käyttäjäoikeuksien hallinta

Käyttäjäoikeudet perustuvat muutamiin toisistaan täysin poikkeaviin termeihin, ja oikeudet määritellään aina moduulikohtaisesti. Seuraavassa on lista näistä kaikkein yleisimmin käytetyistä oikeuksista:

- *Administer* -oikeus antaa käyttäjälle mahdollisuuden muokata tietyn moduulin toiminnallisuutta. Esimerkiksi kuvakaappauksen *Administer blocks* antaa käyttäjälle oikeuden lisätä, poistaa ja muokata lohkoja. *Administer* –tason oikeutta ei siis tule antaa muille kuin sivuston ylläpitäjille. (Mercer 2010, 124.)
- *View* –oikeus sallii käyttäjän pelkästään katsella moduulin tuottamaa sisältöä, ilman oikeuttaa vaikuttaa siihen millään tapaa. Esimerkiksi *View* –oikeus *Comment* –moduulissa sallii pelkästään kommenttien lukemisen, ei niiden kirjoittamista tai muokkaamista. (Mercer 2010, 124.)
- *Create* –oikeus mahdollistaa jonkin sisällön luomisen, mutta ei sen muokkaamista tai poistamista jälkeenpäin. (Mercer 2010, 124.)
- *Edit any/own* –oikeus antaa käyttäjälle mahdollisuuden muokata joko kenen tahansa, tai pelkästään käyttäjän itsensä luomaa sisältöä, riippuen siitä onko kyseessä *Edit any* vai *Edit own* –oikeus. (Mercer 2010, 124.)
- *Delete any/own* –oikeus antaa käyttäjälle mahdollisuuden poistaa joko kenen tahansa, tai pelkästään käyttäjän itsensä luomaa sisältöä. (Mercer 2010, 124.)

Käyttäjäroolien avulla käyttäjätunnukselle voidaan antaa juuri ne oikeudet, joita sille halutaan. Drupal 7:n käyttäjä ei itsessään ole vielä yhtään mitään, ellei sille ole annettu käyttäjäroolia, joka määrittelee tavat, joilla käyttäjä voi sivustolla toimia. Tästä syystä asiaa lähestyttiin käyttäjäroolien kautta. (Mercer 2010, 127.)

Uusia käyttäjiä voidaan luoda kahdella tapaa: käyttäjän itsensä toimesta rekisteröitymällä, tai ylläpitäjän toimesta hallintapaneelin kautta. Rekisteröityneet käyttäjät saavat aina oletuksena tunnistautuneen käyttäjän käyttäjäroolin, hallintapaneelin kautta luodulle käyttäjille voidaan lisätä mikä tahansa käyttäjärooli. Uusille käyttäjille täytyy määrittää aina vähintään käyttäjänimi, sähköpostiosoite ja salasana.

Riippuen sivuston toimintalogiikasta käyttäjien rekisteröitymistä voidaan rajoittaa kolmella eri tapaa. Tiukin rajoitus sallii uusien käyttäjien rekisteröimisen vain ylläpitäjiltä. Kaksi muuta rajoitusta sallivat anonyymien vierailijoiden rekisteröitymisen, mutta toisessa näistä vaihtoehdoista vaaditaan lisäksi ylläpitäjän hyväksyntä.

2.2 Moduulit

Drupal 7:ssä moduuli käsitetään liitännäisenä, joka laajentaa, lisää, tai parantaa järjestelmän toiminnallisuutta (Mercer 2010, 50). Moduuleiden päälle ja pois kytkeminen tapahtuu hallintapaneelista sivuston ylläpitäjien toimesta. Drupal 7 perusasennuksen mukana tulee joukko moduuleita. Näitä moduuleita kutsutaan järjestelmän ydinmoduuleiksi.

Ydinmoduuleita on olemassa kahdenlaisia: valinnaisia ja pakollisia. Valinnaiset ydinmoduulit toimitetaan asennuksen mukana, mutta ne eivät ole järjestelmän toiminnan kannalta välttämättömiä. Osa näistä moduuleista on oletuksena kytketty päälle. Pakolliset moduulit taas ovat järjestelmän toiminnan kannalta välttämättömiä, eikä niitä ole mahdollista kytkeä pois päältä. (Beighley 2010, 152.)

Jotta Drupal 7:n toimintalogiikka tulisi selväksi, on olennaista ymmärtää näiden pakollisten ydinmoduulien tarjoamat toiminnallisuudet.

Block-moduulin avulla luodaan eräänlaisia sisällöstä muodostuvia laatikoita, jotka renderöidään ennalta määriteltyyn alueeseen yhdellä tai useammalla sivulla. Moduulia voidaan käyttää sisällön asetteluun sivuilla. Lisäksi sillä voidaan hallita sisällön näkyvyyttä vain tietyllä sivulla, tietyille käyttäjärooleille, tai vain sivuilla, jotka esittävät tiettyä sisällön tyyppiä.

Field-moduuli mahdollistaa mukautettujen datakenttien määrittämisen sisällön tyypeille. Moduuli huolehtii kentän sisältämän datan tallentamisesta, lataamisesta, muokkaamisesta ja tulostamisesta. Useimmat käyttäjät eivät ole moduulin kanssa tekemisissä suoraan, mutta ne käyttävät valinnaista Field UI –moduulia kenttien liittämiseen ja hallinnoimiseen.

Field-moduuli itsessään tarjoaa perustan kenttien hallinnointiin, mutta se ei määrittele yhtään kenttätyyppiä tai niiden lomake-elementtejä; niiden määrittely tapahtuu omissa moduuleissaan. Drupal 7:n ydin sisältää seuraavat kenttätyyppimoduulit:

- Number
- Text
- List
- Taxonomy
- Image
- File
- Options.

Field SQL storage –moduuli huolehtii kenttien tietokantayhteydestä. Se on pakollinen ydinmoduuli, mutta vallinnaisia vastaavia moduuleita on saatavilla vaihtoehtoisille tietokannoille.

Filter-moduuli mahdollistaa tekstiformaattien määrittämisen. Tekstiformaatti määrittelee HTML-tagit, koodin, tavallisen tekstin ja muun syötteen, jotka ovat sallittuja sisällössä ja kommenteissa. Näin ollen Filter-moduuli on yksi avaintekijöistä suojauduttaessa vahingolliselta syötteeltä, joka on peräisin esimerkiksi tietoisesti haittaa tekeviltä käyttäjiltä. Sivustolla anonyymeille käyttäjille voidaan määritellä esimerkiksi pelkästään tavallinen teksti sallituksi tekstiformaatiksi, mutta tunnistautuneille käyttäjille sallitaan lisäksi joukko HTML-tageja käytettäväksi tekstin seassa. Vaikka yksi määritelty filtteri voi poistaa ei toivottuja HTML-tageja tekstistä, voi toisen filtterin tarkoitus olla www-osoitteiden muuntaminen klikattaviksi linkeiksi.

Node-moduuli hoitaa sisällön näyttämisen, luonnin, muokkaamisen, poistamisen ja asetukset. Sisällön tyypit, kuten tavallinen sivu tai artikkeli, muodostuvat kirjoituksesta, eli *node*:sta. Kirjoitus sisältää otsikon, ja joitakin metatietoja, kuten kirjoittajan, luomisajan, ja sisällön tyypin. Lisäksi kirjoitukseen on mahdollista liittää valinnaisia kenttiä, joihin voidaan tallentaa tekstiä tai muuta dataa. Node-moduulin avulla hallitaan myös sisällön julkaisuasetuksia ja näkyvyyttä sivustolla. Uusien sisällön tyyppien luominen tapahtuu myös tämän moduulin kautta.

System-moduuli on olennaisin moduuli sivuston toiminnan kannalta, ja se tarjoaa perustoiminnallisuudet toisten moduuleiden ja teemojen laajennettaviksi. System-moduuli sisältää järjestelmän useimpia oleellisimpia toimintoja, kuten välimuistin hallinnan, moduuleiden ja teemojen hallinnan, hallintasivujen näyttämisen, ja sivuston perusasetusten konfiguroinnin.

User-moduuli vastaa kaikista käyttäjiin liittyvistä toiminnallisuuksista, kuten rekisteröitymisestä sekä sisään- että uloskirjautumisesta. Sen kautta ylläpitäjät hallitsevat käyttäjärooleja sekä näiden käyttäjäoikeuksia.

3 DRUPAL-KEHITYS

3.1 Koukkufunktiot

Ohjelmistoja ja sovelluksia kehitettäessä on usein tarpeellista vaikuttaa käyttöjärjestelmän, tai käytettävän sovelluskehyksen toimintaan ilman, että itse sovelluskehykseen tai käyttöjärjestelmään tehdään muutoksia. Tämä on mahdollista toteuttaa sieppaamalla funktiokutsuja, viestejä, tai tapahtumankäsittelijöitä ohjelman osien välillä. Tällaista menetelmää kutsutaan ohjelmistotekniikassa *koukuksi*. (Understanding the hook system for Drupal modules 2013.)

Drupal 7:ssä moduulien ja järjestelmän ytimen toiminnallisuuksien välinen kanssakäyminen tapahtuu koukkufunktioilla. Drupal 7:n ydin määrittelee kattavan valikoiman erilaisia koukkufunktioita, joilla on mahdollista esimerkiksi määritellä URL-osoitteita, lisätä sisältöä sivuihin tai luoda omia tietokantatauluja. Moduulit voivat määritellä myös omia koukkufunktioitaan muiden moduulien käytettäväksi. (Understanding the hook system for Drupal modules 2013.)

Koukkufunktiot suoritetaan ohjelman ajonaikaisesti ennalta määrätyissä pisteissä, joissa järjestelmä etsii koukkufunktioiden toteutuksia kaikista päälle kytketyistä moduuleista. Esimerkiksi kun käyttäjä vierailee Drupalin hallintapaneelin apu-sivulla sitä muodostettaessa järjestelmä etsii jokaisesta aktiivisesta moduulista *module_help* nimistä koukkufunktiota, jossa *module* on korvattu moduulin nimellä. Kyseisen koukkufunktion toteutuksessa moduulin tulee palauttaa dokumentaatio itsestään. (Understanding the hook system for Drupal modules 2013.)

3.2 Menu-järjestelmä

Drupal 7:ssä käyttäjille näytettävien navigaatio-valikoiden määrittely ja URL-osoitteiden reititys tapahtuu menu-järjestelmässä. Varsinaisten navigaatio-valikoiden hallinta hoidetaan erillisessä menu-moduulissa, joten menu-järjestelmän nimitys on hieman harhaanjohtava. Menu-järjestelmän pääpaino on

HTTP-kutsujen ohjaaminen ennalta määrätyille callback-funktioille. (Tomlinson & VanDyk 2010, 57.)

Kun www-selaimella vieraillaan Drupalilla toteutetulla sivustolla, Drupal saa ensimmäisenä tiedon URL-osoitteesta, jota kutsutaan. Tästä osoitteesta menu-järjestelmän täytyy päätellä, mitä koodia suoritetaan, ja kuinka selaimen pyyntöä käsitellään. Drupal leikkaa pois URL-osoitteen perusosan, ja käyttää päättelyssä jälkimmäistä osaa, jota kutsutaan poluksi.

Esimerkiksi jos osoite on *http://example.com/?q=node/3*, käytettävä polku on *node/3*. Drupalissa on mahdollista käyttää myös selkokielisiä osoitteita, jolloin edellisen esimerkin osoite muuttuisi muotoon *http://example.com/node/3*. Jos web-serveri on muuttanut osoitteen samaan muotoon kuin ensimmäisessä esimerkissä, Drupalin tarvitsee käsitellä vain yhden muotoisia osoitteita. (Tomlinson & VanDyk 2010, 57.)

Kaikkien käytössä olevien moduulien täytyy toimittaa Drupalille tieto menu-nimikkeistä. Tämä tapahtuu koukku-funktiossa nimeltä *hook_menu()*. Jokainen nimike koostuu taulukosta, jossa avaimena on polku, johon viitataan. Taulukko sisältää lisäksi erilaista metadataa polusta, kuten suoritettavan takaisinkutsu-funktion, tässä tapauksessa PHP-funktion nimen, joka suoritetaan, kun avaimena määritellyssä polussa vieraillaan. (Tomlinson & VanDyk 2010, 57.)

Kuviossa 13 on kuvattu esimerkki *hook_menu()* koukkufunktion toteutuksesta. Rivillä 69 lisätään *$items*-nimiseen taulukkoon uusi solu avaimella *karttahaku*. Solu sisältää uuden taulukon, joissa eri avaimilla määritellään tietoa kyseisestä menu-nimikkeestä. Rivillä 70 on määritelty kutsuttava PHP-funktio, kun *karttahaku* nimisessä polussa vieraillaan.

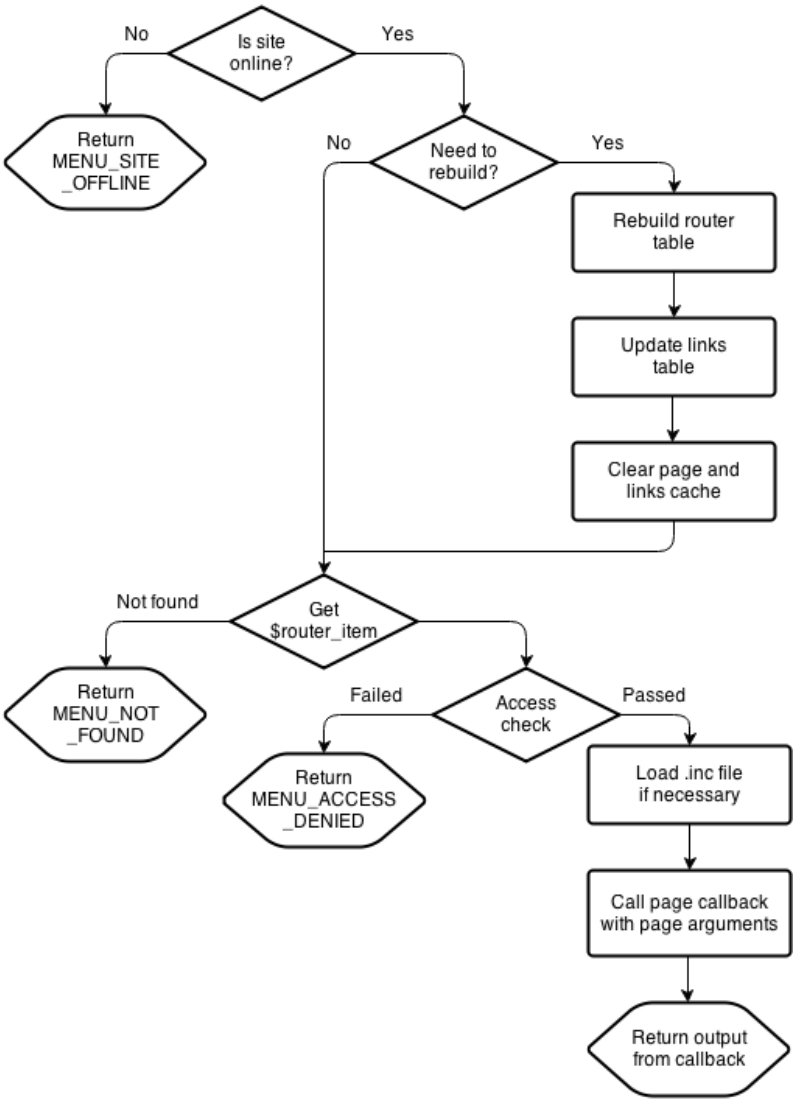
```php
/**
 * Implements hook_menu().
 */
function ratsastustunti_main_menu() {
    $items['karttahaku'] = array(
        'page callback' => 'ratsastustunti_main_map_view',
        'access arguments' => array('access content'),
        'type' => MENU_CALLBACK,
    );
}
```

KUVIO 13. Esimerkki *hook_menu()* -funktion toteutuksesta

Drupal käy läpi seuraavanlaiset vaiheet, kun HTTP-pyyntö saapuu:

1. Menu-järjestelmä päättelee URL-osoitteen perusteella kutsuttavan polun.
2. Drupal tallentaa polut, ja niitä vastaavat takaisinkutsu-funktiot omaan tietokanta-tauluunsa. Tässä vaiheessa tarkistetaan, tarvitseeko kyseinen taulu muutosten seurauksena rakentaa uudelleen.
3. Etsitään tietokanta-taulusta polkua vastaavaa riviä ja kutsutaan sen perusteella oikeaa koukku-funktiota.
4. Ladataan kaikki takaisinkutsu-funktion suorittamiseen vaadittavat objektit
5. Tarkistetaan onko käyttäjällä tarvittavia oikeuksia takaisinkutsun suorittamiseen.
6. Lokalisoidaan menu-nimikkeen otsikko ja kuvaus.
7. Ladataan tarpeelliset laajennus-tiedostot
8. Kutsutaan takaisinkutsu-funktiota, ja palautetaan tulos *index.php* tiedoston käsiteltäväksi.

Kuviossa 14 vastaavat vaiheet on kuvattu prosessikaavion avulla.



KUVIO 14. Menu-järjestelmän prosessikaavio

3.3 Entiteetit

Drupalin aiemmissa versioissa kenttiä oli mahdollista liittää vain sisällön tyyppeihin. Drupal 7 esitteli uuden mekanismin nimeltä entiteetti *(entity)*. Entiteettityypit *(entity types)* ovat tapa luoda abstrakteja ryhmiä, joihin voidaan liittää kenttiä. Entiteettityyppi on etenkin moduuli-kehittäjälle käytännöllinen tapa luoda erityyppisiä ryhmiä, jotka eivät kuitenkaan ole varsinaisia sisällön tyyppejä, joita haluttaisiin näyttää käyttäjälle sivuilla. Entiteettityyppejä ovat myös Drupalin ytimen komponentit, kuten kirjoitukset, kommentit ja käyttäjäprofiilit.

Entiteetit voidaan selittää myös olio-ohjelmoinnin termein seuraavan taulukon 2 mukaisesti.

TAULUKKO 2. Drupal entiteetin ja olio-ohjelmoinnin vastaavat termit

Drupal	**OOP**
Entiteettityyppi *(entity type)*	Kantaluokka
Nippu *(bundle)*	Periytetty luokka
Kenttä *(field)*	Jäsenmuuttuja
Entiteetti *(entity)*	Kantaluokan tai periytetyn luokan instanssi

Niput ovat entiteettityyppien toteutuksia, tai niitä voidaan ajatella entiteettityyppien alaluokkina. Otetaan esimerkiksi kirjoitus – tässä tapauksessa entiteettityyppi – josta voidaan tuottaa artikkeleita, blogikirjoituksia, tai vaikka tuotteita. Tässä tapauksessa näitä edellisiä voidaan ajatella nippuina, eli alaluokkina. Kaikista entiteettityypeistä ei kuitenkaan voi luoda nippuja.

Kentät toimivat entiteeteistä puhuttaessa samalla tavalla kuin aiemmin käsitellyssä sisällön tyyppien tapauksessa. Kenttiä voidaan liittää siis entiteettityyppeihin, kuten myös nippuihin, helpottamaan datan organisointia.

Yksittäinen entiteetti on tietyn entiteettityypin, tai nipun, instanssi – eli edustaja. Entiteettityypin entiteettejä ovat esimerkiksi yksittäiset kommentit tai käyttäjän luoma profiili. Nippujen entiteettejä taas ovat yksittäiset artikkelit tai blogikirjoitukset.

4 HEVOSTALLI-PORTAALI

4.1 Vaatimukset

Portaaliin on tavoitteena kerätä mahdollisimman kattava osa Suomen hevostalleista, jotka tarjoavat ratsastuspalveluita asiakkaille. Jokaiselle portaalin hevostallille tulee oma sivunsa, jossa kerrotaan yleisesti tallin toiminnasta, sen tarjoamista palveluista ja varustelutasosta. Hevostalliyrittäjät saavat käyttäjätunnukset portaaliin ja näin ollen voivat muokata oman tallinsa tietoja.

Loppukäyttäjältä ei vaadita kirjautumista palveluun, ja kaikki portaalin sisältämät tallit ovat käyttäjän selailtavissa. Palvelusta halutaan tehdä täten helposti lähestyttävä. Käyttäjän paikkatietoa halutaan käyttää perustellusti haun lisäkriteerina, mutta sijainnin ilmoittamista ei vaadita.

4.2 Rakenne

Kun Drupal 7 oli saatu asennettua ja konfiguroitua, määriteltiin järjestelmään uusi sisällön tyyppi nimeltä talli, jota jokainen portaaliin tuleva hevostalli tulisi käyttämään. Taulukossa 3 on listattu kaikki talli-typpiin liittyvät kentät. Pelkästään kenttiä tarkastelemalla saa selkeän kuvan talli-sivujen rakenteesta sekä sen sisällöstä. Taulukon kaksi viimeistä kenttää, latitude ja longitude, eivät kuulu Drupalin ytimeen, vaan ne saatiin käyttöön asentamalla kolmansien osapuolten kehittämä *Location*-niminen moduuli.

TAULUKKO 3. Talli sisältötyyppiin liitetyt kentät

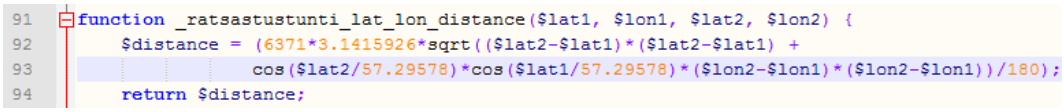
Kentän nimi	Kentän tyyppi	Lomake-elementti
Nimi	Title field	Text field
Kuvaus	Long text	Text area (multiple rows)
Kuva	Image	Image
Katuosoite	Text	Text field
Postinumero	Integer	Text field
Postiosoite	Text	Text field
Päivystysnumero	Text	Text field
Ryhmäkoko	Term reference	Check boxes
Varustelu	Term reference	Check boxes
Opettajien taso	Term reference	Check boxes
Latitude	Latitude / longitude	
Longitude	Latitude / longitude	

4.3 Valmistelut

Koska hevostalli-portaalin kantavana ideana pidettiin käyttäjän paikkatietoon perustuvaa hakua, täytyi jokaisen tallin sijainti määrittää pituus- ja leveysasteina maantieteellistä koordinaatistoa käyttäen. Kun jokaisen tallin tietoihin oli syötetty myös tallin osoite, voitiin koordinaatit määrittää Googlen tarjoaman Geocoding-rajapinnan avulla. Geocoding-rajapintaa käytetään HTTP-kutsun välityksellä, ja osoitetiedot sille annetaan GET-parametrina. Vastauksena rajapinta palauttaa joko JSON- tai XML-muotoista dataa.

Tallit käytiin ohjelmallisesti lävitse, ja pituus- sekä leveyskoordinaatit tallennettiin omiin kenttiinsä, jotka talli-tyypille oli määritelty. Näitä kenttiä ei kuitenkaan missään vaiheessa ole tarkoitus näyttää loppukäyttäjälle, vaan niitä käytetään apuna käyttäjän ja tallin välisen etäisyyden laskennassa.

Kuviossa 15 on esitetty PHP:lla toteutettu funktio etäisyyden laskemiseksi käyttäjän, ja tallin välillä. Funktiossa on hyödynnetty Harversinen kaavaa, joka laskee kahden koordinaattipisteen välisen etäisyyden ottaen huomioon myös maapallon kaarevuuden.



KUVIO 15. PHP-funktio etäisyyden laskemiseen

5 KEHITETYT MODUULIT

5.1 Haku-moduuli

Loppukäyttäjän kannalta hevostalli-portaalin ja samankaltaisten sivustojen, tärkein työkalu on haku. Ilman hyvin toimivaa ja kattavia hakuvaihtoehtoja sisältävää hakua, käyttäjän on mahdotonta löytää kriteereihinsä sopivaa vaihtoehtoa satoja tietueita käsittävästä tietokannasta. On siis tärkeää suunnitella niin käyttöliittymältään kuin tekniseltä toteutukseltaankin sujuvasti toimiva haku.

Työn hevostalli-portaalin haun haluttiin toimivan käyttäjän paikkatietoon perustuen, koska voitiin olettaa, että käyttäjä haluaa löytää parhaan mahdollisen vaihtoehdon mahdollisimman läheltä kotipaikkakuntaansa. Näin ollen tärkeimpänä hakukriteerinä pidettiin kilometrimääräistä sädettä käyttäjälle määritellystä tai käyttäjän itsensä määrittelemästä paikkakunnasta katsoen. Valittavissa olisi 20 km, 100 km, 300 km tai koko Suomi. Muita hakukriteerejä olisivat esimerkiksi tallin varustelu ja ryhmäkoot. Kuviossa 16 on esitetty kuvankaappaus haun suunnitellusta käyttöliittymästä.

KUVIO 16. Hevotalli-portaalin haku

Hevostalli-portaalin haku toteutettiin Drupal-kehitykselle ominaiseen tapaan omana moduulinaan. Moduuli sisältää kaiken haun vaatiman koodin, ja valmis moduuli voidaan kytkeä päälle hallintapaneelin kautta, samaan tapaan kuin Drupalin sisältämät valinnaiset ydinmoduulitkin.

Jotta järjestelmä tunnistaa moduulin, täytyy sille kirjoittaa .info-tiedosto. Tiedosto sisältää moduulia kuvailevaa metadataa, joka annetaan INI-standardin mukaisesti avain-arvo –pareina. Kuviossa 17 on esitetty haku-moduulin .info-tiedosto. *name*- ja *description*-ominaisuudet näytetään hallintapaneelissa moduulin tiedoissa. *core*-ominaisuudella määritellään moduulin vaatima Drupalin minimiversio.

```
1   name = Stable Search
2   description = A block module that is used for searching stables.
3   core = 7.x
```

KUVIO 17. Haku-moduulin .info-tiedosto

5.1.1 Lohko

Jotta hakulomake saadaan näkyviin sivulla, täytyy lomaketta varten määritellä oma lohkonsa. Uusi lohko määritellään *hook_block_info()* –nimisessä koukkufunktiossa, ja lohkon näyttämä sisältö määritellään *hook_block_view()* –nimisessä funktiossa.

Kuviossa 18 on esitetty haku-lohkon määritelevä koukkufunktio. Lohko määritellään $blocks-muuttujan sisältämään taulukkoon uutena soluna, jonka avain on lohkon koneluettava nimi. Lohkon ainoa pakollinen ominaisuus on *info*, johon annetaan lohkon nimi, joka näytetään ylläpitäjälle hallintakäyttöliittymässä. Haku-lohkolle annettiin nimeksi *stable_search*.

```
3   /**
4    * Implements hook_block_info().
5    */
6   function stable_search_block_info() {
7       $blocks['stable_search'] = array(
8           'info' => t('Stable search'),
9           'cache' => DRUPAL_CACHE_PER_ROLE,
10      );
11
12      return $blocks;
13  }
```

KUVIO 18. Koukkufunktio haku-lohkon määrittelemiseksi

Lohkolle täytyy vielä kertoa, mitä sisältöä sillä halutaan esittää. Kuviossa 19 riveillä 20-24, *stable_search* -lohkon sisällöksi asetetaan *stable_search_from* -niminen lomake. Osa kyseisen lomakkeen toteutuksesta käydään lävitse seuraavassa kappaleessa (kappale 5.1.2).

```
15  /**
16   * Implements hook_block_view().
17   */
18  function stable_search_block_view($delta = '') {
19      switch($delta) {
20          case 'stable_search':
21              $block['subject'] = null;
22              $block['content'] = drupal_get_form('stable_search_form');
23              break;
24      }
25
26      return $block;
27  }
```

KUVIO 19. Koukkufunktio, jossa määritellään haku-lohkon näyttämä sisältö

5.1.2 Lomake

Haku-lohkon näyttämä haku-lomake määritellään Drupalin lomake-rajapinnan mukaisesti taulukoilla. Kuviossa 20 on esitetty osa haku-lomakkeen toteutuksesta.

Kuviossa näkyvät vain valintapainikkeiden ja lähetys-napin määrittelyt, mutta loput lomake-elementeistä määritellään saman periaatteen mukaisesti.

Kuviossa riveillä 31-36 luodaan taulukko, joka sisältää lomakkeelle tulevat valintaoptiot hakualueen rajaamiseksi, ja asetetaan se $radiOptions-nimiseen muuttujaan. Riveillä 38-44 asetetaan lomakkeen valintapainikkeet $radiOptions-muuttujan perusteella. Rivillä 39 *type*-ominaisuudella kerrotaan, että lomake-elementiksi halutaan valintapainikkeet. Rivillä 40 *default_value*-ominaisuus asettaa ”Koko Suomi” –vaihtoehdon valituksi oletuksena, kun lomakkeelle saavutaan.

Riveillä 46-50 määritellään lomakkeen lähettämis-painike. Rivillä 48 asetetaan funktio, jota kutsutaan, kun käyttäjä klikkaa lähetä-painiketta. Kuvion tapauksessa kutsutaan *stable_search_form_submit* –nimistä funktiota.

```
29  /* Search form for frontpage */
30  function stable_search_form($form_state) {
31      $radiOptions = array(
32          'all' => 'Koko Suomi',
33          30 => '30 km säteellä',
34          100 => '100 km säteellä',
35          300 => '300 km säteellä'
36      );
37
38      $form['distance'] = array(
39          '#type' => 'radios',
40          '#default_value' => 'all',
41          '#options' => $radiOptions,
42          '#attributes' => array('class' => array('styled')),
43          '#title' => 'Näytä kaikki Suomen kohteet',
44      );
45
46      $form['submit_list'] = array(
47          '#type' => 'submit',
48          '#submit' => array('stable_search_form_submit'),
49          '#value' => 'Näytä listalla',
50      );
51
```

KUVIO 20. Haku-lomakkeen muodostaminen

Varsinainen hakutulosten suodattaminen tapahtuu omalla tulos-sivullaan. Tulos-sivulle on asetettu polku *node/2*. Tulosjoukkoa rajaavia suodattimia voidaan asettaa URL-osoitteen mukana toimitettavilla parametreilla.

Kuviossa 21 on esitetty lähetä-painikkeen takaisinkutsufunktion toteutus. Funktio saa parametrina lähetetyn lomakkeen tiedot muuttujassa nimeltä $form_state. Rivillä 135 $form_state-muuttujasta parsitaan halutut hakuarvot, ja rivillä 137 arvot välitetään hakusivulle. Samalla myös käyttäjä ohjataan samalle sivulle.

```
134 function stable_search_form_submit($form, &$form_state) {
135     $r = stable_search_get_options($form_state);
136 
137     drupal_goto('node/2',array('fragment' => 'haku='.$r['prox'].'&'.$r['se
138 }
```

KUVIO 21. Haku-lomakkeen lähetä-painikkeen takaisinkutsufunktio

5.2 Sijainti-moduuli

Sijainti-moduulilla on tarkoitus hallita käyttäjälle määriteltyä sijaintia, ja se toimii yhteistyössä Location-moduulin kanssa. Location-moduuli hoitaa käyttäjän paikantamisen ja paikkatiedon taltioimisen. Paikantaminen tapahtuu joko HTML5:n tarjoaman Geolocation rajapinnan avulla, tai mikäli käyttäjän selain ei ole HTML5-yhteensopiva, voidaan suuntaa antava paikkatieto arvioida käyttäjän IP-osoitteen perusteella. IP-pohjainen paikantaminen vaatii kuitenkin erillisen tietokannan asentamisen.

5.2.1 Entiteettityyppi location_cities

Kehitetyn moduulin tarkoitus on mahdollistaa sijainnin vaihtaminen käyttäjälle. Käyttäjä voi valita sijainnikseen minkä tahansa Suomen kunnan. Kuntien sijaintien taltioimista varten luotiin uusi entiteettityyppi nimeltä *location_cities*. Uusi entiteettityyppi määritellään kuvion 22 mukaisella koukkufunktiolla. Samassa koukkufunktiossa voidaan määritellä useampia entiteettityyppejä yhdellä kertaa, mutta kuvion toteutuksessa määritellään vain yksi.

```
3  /**
4   * Implements hook_entity_info().
5   */
6  function location_cities_entity_info() {
7      $return = array(
8          'location_cities' => array(
9              'label' => t('City locations'),
10             'entity class' => 'Entity',
11             'controller class' => 'EntityAPIController',
12             'base table' => 'location_cities',
13             'entity keys' => array(
14                 'id' => 'pid',
15             ),
16             'label callback' => 'entity_class_label',
17             'uri callback' => 'entity_class_uri',
18         ),
19     );
20
21     return $return;
```

KUVIO 22. Koukkufunktio uuden entiteettityypin määrittelemiseksi

Määrittely tapahtuu taulukon avulla, jossa jokaisen alkion avain vastaa entiteettityypin koneluettavaa nimeä. Kuviossa *location_cities* tyypin määrittely alkaa riviltä 8. Alkio sisältää vastaavasti uuden taulukon, jossa entiteettityypin ominaisuudet on annettu avain-arvo –pareina.

Rivit 12-15 ovat järjestelmän kannalta kaikkein oleellisimmat. *base table* –ominaisuus määrittää entiteettityypin käyttämän tietokantataulun, johon myöhemmin luotavien entiteettien data tallennetaan. *entity keys* –ominaisuus sisältää taulukon, jossa on määriteltävä vähintään tietokantataulun primaariavain. Drupalin kenttä-rajapinta *(Field API)* käyttää tätä tietoa, kun entiteettiin liitetään kenttiä.

5.2.2 Entiteettien luominen

Jotta käyttäjä voisi vaihtaa sijainnikseen haluamansa kunnan, täytyvät kunnat olla listattuna jossakin. Kuntien ja niiden sijaintien taltioimiseen käytetään *location_cities* entiteettityyppiä. Kuten käyttäjänkin kohdalla, kuntien sijainti taltioidaan koordinaattipisteinä, jotka voidaan määrittää Googlen Geocoding rajapinnan avulla.

Seuraavassa kuviossa esitellään silmukka entiteettien luomiseksi jokaisesta Suomen kunnasta. Kuviossa 23 rivillä 44 $cities-muuttuja sisältää taulukon kuntien nimistä, ja $city-muuttujaan asetetaan vuorollaan kunkin kunnan nimi. Riveillä 46-51 asetetaan Geocoding rajapinnan palauttama JSON-muotoinen data $data –nimiseen muuttujaan.

Varsinainen entiteetin datan asettaminen tapahtuu riveillä 55-60. *type*-ominaisuus kertoo käytettävän entiteettityypin, *name, latitude* ja *longitude* ovat tallennettavia kenttiä. Rivillä 64 data tallennetaan tietokantaan, ja rivillä 65 uuden entiteetin kentät tallennetaan.

```
44  foreach ($cities as $city) {
45    do {
46      $data = json_decode(
47        file_get_contents(
48            'http://maps.googleapis.com/maps/api/geocode/json?address='.$city.'+Finl
49        ),
50        TRUE
51      );
52
53      if ($data['results'][0]['geometry']['location']) {
54        $loc = $data['results'][0]['geometry']['location'];
55        $city = (object) array(
56          'type' => 'location_cities',
57          'name' => $city,
58          'latitude' => $loc['lat'],
59          'longitude' => $loc['lng'],
60        );
61      }
62    } while (!$data['results'][0]['geometry']['location']);
63
64    $result = drupal_write_record('location_cities', $city);
65    field_attach_insert('location_cities', $city);
66  }
```

KUVIO 23. Silmukka *location_cities*-entiteettien luomiseksi

5.2.3 Toiminnalliset osat

Sijainnin vaihtaminen haluttiin toteuttaa omassa valikko-polussaan. Poluksi valittiin *sijainti/kunta*, jossa *kunta* korvataan halutulla kunnan nimellä. Esimerkiksi *sijainti/Lahti* vaihtaisi käyttäjän sijainnin Lahteen. Tätä varten moduulille täytyi määritellä oma menu-nimike, jonka takaisinkutsufunktioksi määriteltiin kuvion 24 mukainen funktio.

Kuvion funktio saa parametrina polun jälkimmäisen osan, eli kunnan nimen. Riveillä 84-88 haetaan tietokannasta kunnan nimeä vastaava entiteetti joka asetetaan $result-nimiseen muuttujaan. Muuttuja sisältää taulukon, jossa on määritelty kunnan nimi sekä koordinaatit.

Käyttäjän sijainti tallennetaan PHP:n SESSION-muuttujaan taulukkona. Taulukko sisältää tiedot maasta, maakoodista, kaupungista / kunnasta, sekä koordinaatit. Riveillä 91-96 asetetaan $result-muuttujassa olevat tiedot käyttäjän SESSION-muuttujaan. Lopuksi käyttäjä ohjataan takaisin sivulle, jossa sijainninvaihtamis-

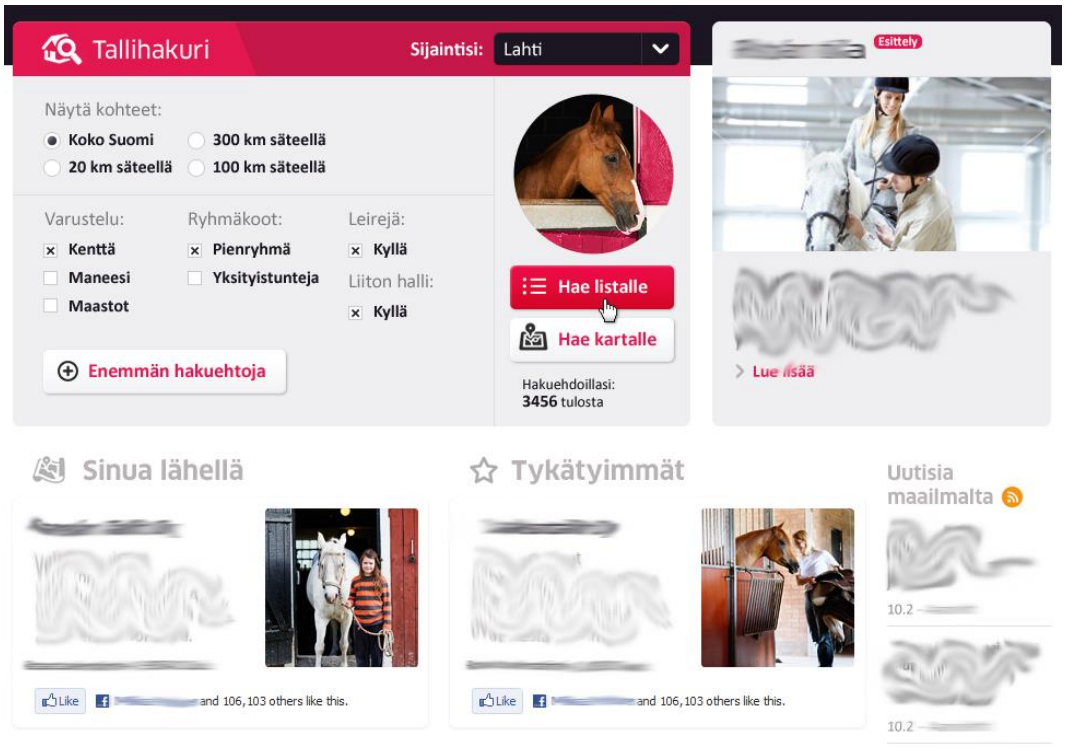
linkkiä klikattiin, joten operaatio näkyy käyttäjälle vain sivun uudelleen lataamisena.

```
82  function location_change_callback($location = NULL) {
83      if (isset($location)) {
84          $result = db_select('location_cities', 'c')
85          ->fields('c')
86          ->condition('name', $location,'=')
87          ->execute()
88          ->fetchAssoc();
89
90          if (isset($result)) {
91              $_SESSION['user']['location'] = array (
92                  'country' => 'Finland',
93                  'country_code' => 'FI',
94                  'city' => $result['name'],
95                  'latitude' => $result['latitude'],
96                  'longitude' => $result['longitude']
97              );
```

KUVIO 24. Takaisinkutsufunktio käyttäjän sijainnin vaihtamiseksi

Käyttäjän sijainti ilmoitetaan etusivun hakulomakkeen yhteydessä. Samalla lomakkeella onnistuu myös sijainnin vaihtaminen, mikäli paikantaminen ei ole onnistunut, tai käyttäjä on estänyt sen. Paikkakunnan nimeä klikkaamalla aukeaa valmiiksi määritelty lista Suomen suurimmista kaupungeista.

Lisäksi lomakkeella on hakukenttä, johon voi kirjoittaa minkä tahansa Suomen kunnan nimen. Kuviossa 25 on esitetty hevostalli-portaalin etusivu, jossa sijaitsevat myös haku ja sijainnin vaihtamiseen tarkoitettu painike.



KUVIO 25. Hevostalli-portaalin etusivu

6 YHTEENVETO

Drupal 7 -moduulikehitys luo omat haasteensa ensimmäistä moduuliaan kehittävälle ohjelmoijalle. Vaikka moduulikehityksessä käytettävä PHP-ohjelmointikieli olisikin kehittäjälle entuudestaan tuttu, saattavat Drupalin määrittelemät käytännöt tuntua kankeilta, koska käytännössä kaikki toiminnallinen koodi täytyy toteuttaa omissa koukkufunktioissaan. Tällainen ohjelmistokehityksen lähestymistapa vaatii totuttelua aiheeseen ensimmäistä kertaa tutustuvalta.

Työn haku- ja sijainti-moduulit toteutettiin Drupal 7 moduulikehityksen periaatteita noudattaen. Jälkeenpäin tarkasteltaessa esiin nousee joitakin asioita, joita olisi voitu tehdä enemmän Drupalille ominaisella tavalla ja valmiita moduuleita olisi voitu hyödyntää tehokkaammin. Kuitenkin lopputuloksena syntyneet moduulit toimivat määritellyllä tavalla ja täydentävät hevostalli-portaalin paikkatietoon perustuvan palvelun ominaisuuksia onnistuneesti.

Moduuleiden valmistuttua hevostalli-portaali on edelleen kehitysvaiheessa, ja sen sisältämiä ominaisuuksia täytyy edelleen täydentää. Sijaintiin perustuvaa hakua pystyttiin kuitenkin testaamaan, koska portaaliin oli jo aikaisessa vaiheessa syötetty satoja ratsastuspalveluja tarjoavia yrityksiä. Käyttäjän paikkatiedon hyödyntämisestä haussa todettiin olevan huomattavaa hyötyä, etenkin kun palveluita haluttiin etsiä lähiseudulta.

Työn käytännönosan valmistuttua, on Drupal 7:n moduulikehityksestä syntynyt selkeä mielikuva, ja lähtötasoon verrattuna kynnys uuden moduulin kehittämiseen on laskenut huomattavasti.

LÄHTEET

Drupal Suomi. 2013. Mikä Drupal? [viitattu 10.3.2013]. Saatavissa: http://drupal.fi/fi/drupal-suomi

Beighley, L. 2010. Drupal For Dummies. Yhdysvallat: Wiley Publishing, Inc.

Dougherty, D. 2001. ONLamp.com [viitattu 10.3.2013]. Saatavissa: http://www.onlamp.com/pub/a/onlamp/2001/01/25/lamp.html

System requirements. 2013. drupal.org [viitattu 10.3.2013]. Saatavissa: http://drupal.org/requirements

Database Server. 2013. drupal.org [viitattu 10.3.2013]. Saatavissa: http://drupal.org/requirements/database

Step 2: Create the database. 2013. drupal.org [viitattu 10.3.2013]. Saatavissa: http://drupal.org/documentation/install/create-database

Step 3: The settings.php file. 2013. drupal.org [viitattu 15.3.2013]. Saatavissa: http://drupal.org/documentation/install/settings-file

Step 4: Run the installation script. 2012. drupal.org [viitattu 15.3.2013]. Saatavissa: http://drupal.org/documentation/install/run-script

Setup of /sites directory for multi-site. 2010. drupal.org [viitattu 10.3.2013]. Saatavissa: http://drupal.org/node/53705

Working with content types and fields. 2010. drupal.org [viitattu 15.3.2013]. Saatavissa: https://drupal.org/documentation/modules/field-ui

The Drupal overview. 2014. drupal.org [viitattu 18.4.2014]. Saatavissa: https://drupal.org/getting-started/before/overview

Mercer, D. 2010. Drupal 7 Second Edition. Iso-Britannia: Packt Publishing

Understanding the hook system for Drupal module. 2013. drupal.org [viitattu 29.3.2014]. Saatavissa: https://drupal.org/node/292

Tomlinson, T., VanDyk, JK. Pro Drupal 7 Development Third Edition. Yhdysvallat: Apress