Turun ammattikorkeakoulu

Tietojenkäsittely

Tietojärjestelmät

2012

Tiia Toivonen

WEB-SOVELLUS INTRANET

– Suunnittelu ja toteutus Drupalilla

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

Turun ammattikorkeakoulu

Tietojenkäsittely | Tietojärjestelmät

5.6.2012 | 51

Anne Jumppanen

Tiia Toivonen

WEB-SOVELLUS INTRANET – SUUNNITTELU JA TOTEUTUS DRUPALILLA

Mitä asioita tulisi ottaa huomioon käytettäessä valmista järjestelmää intranetin pohjana? Kuinka Drupalia voidaan työstää erilaisiin tarkoituksiin, ja erityisesti sisäiseksi järjestelmäksi? Mikä on intranetille ominaista? Näihin kysymyksiin pyritään vastaamaan opinnäytteessä.

Opinnäyte tehtiin projektina ja osana suurempaa kokonaisuutta asiakkaalle. Projektina tämä oli ICT-Portissa ensimmäinen laatuaan. ICT-Portti avustaa paikallisia yrityksiä hyödyntämään tietotekniikkaansa paremmin ja implementoimaan uutta. Projektit on pääosin toteutettu opiskelijavetoisesti, mikä antaa opiskelijoille ainutlaatuisen tilaisuuden nähdä sellaisen kuvan paikallisesta yritystoiminnasta, jota he eivät ehkä muuten voisi kokea.

Tässä opinnäytteessä sovelletaan avoimen lähdekoodin sisällönhallintajärjestelmää, Drupalia intranet kokonaisuuden toteuttamiseen. Alun luvuissa tutustutaan Drupalin toiminnallisuuksiin ja funktionaalisuuksiin kokonaisuutena ja vertaillaan sitä muihin tämän hetken suosituimpiin avoimen lähdekoodin sisällönhallintajärjestelmiin. Rajaus luvussa mietitään järjestelmien eroja, jonka lisäksi perustellaan miksi Drupal valittiin tähän projektiin.

Opinnäytteessä tutustutaan myös moduulien kirjoittamiseen. Moduulit toteutetaan käyttämällä hyödyksi Drupalin omia funktionaalisuuksia ja yhdistetään ne järjestelmän osiksi koukuilla, joista kerrotaan opinnäytteen edetessä.

Käyttöliittymä-osuudessa suuressa osassa on ulkoasu ja sen koostaminen ohjeistusten mukaisesti. Toteutetussa käyttöliittymässä on otettu vahvasti huomioon käyttäjien tottumukset ja heille jo tutut järjestelmät. Hyödyntämällä yhtä Drupalin käytännöllisintä työkalua, Viewsiä voidaan käyttöliittymän osat esittää sekä Drupalin ohjeistuksen mukaisesti, että käyttäjälle sopivassa muodossa.

Yhteenvedossa pohditaan projektin kulkua, kompromisseja ja vastuunottoa. Erityisesti otetaan kantaa siihen, miten edetään projektissa, jossa ei ole näkyvää johtamista eikä ryhmän jäsenten näkemykset kohtaa. Lopuksi kerrotaan, mitä projektista jäi mieleen.

ASIASANAT:

Drupal, sisällönhallinta, intranet

BACHELOR´S THESIS | ABSTRACT
TURKU UNIVERSITY OF APPLIED SCIENCES

Business Information Technology | Information systems

5.6.2012 | 51

Anne Jumppanen

Tiia Toivonen

WEB APPLICATION INTRANET – DESIGN AND IMPLEMENTATION WITH DRUPAL

What should be taken into consideration when using a ready made system as a base for an intranet? How can Drupal be moulded into providing different functions, and especially as an internal system? What makes an intranet? These are the questions this thesis strives to answer.

The thesis was made as a part, and a result, of a project and a part of a bigger picture for a client. How I got involved in this project was via my job at ICT-Gate. This project was the first, and one of a kind for ICT-Gate. ICT-Gate supports local businesses by helping them make better use of their information and communications technology, and initialization of new ones. Projects are mainly done by students, which provides insight and experience of the local business scene in a way they might not otherwise experience.

This thesis is about applying an open source content management system Drupal to provide an intranet solution. The reader's are familiarized with Drupal in the following chapters, introducing a few of its differences compared to other content management systems.

Additionally, this thesis provides a short glance at the other popular content management systems. In the third chapter, content management systems are compared and the reasons of choosing Drupal as the system of choice is contemplated.

This thesis also covers the writing of modules in Drupal. Modules are implemented by utilizing Drupal's own functionalities and combining them with Drupal by adding hooks to them. More about these hooks can be found in the following chapters.

In the sections covering user interface, or the UI, the appearance of the system is given additional thought. Tapping into one of Drupal's most powerful tools called Views, it is possible to modify the output showed to the user with specific detail. Views module enables compiling the interface to the user in such a way that it complies to Drupal's instructions.

In the summary chapter, the project's course, compromises and responsibilities are pondered. How to advance In a project that has no clear goals or management. What is to be taken from this project for the future ones?

KEYWORDS:

Drupal, CMS, intranet

SISÄLTÖ

1 JOHDANTO 6

2 PROJEKTIN ESITTELY JA RAJAUS 8

3 ALUSTAN VALINTA 11

3.1 Vaihtoehdot 11

3.2 Valinta 12

4 DRUPALIN ESITTELY 14

4.1 Drupal toiminnallisuuksia 15

4.2 Drupal funktionaalisuuksia 18

5 MODUULIN KIRJOITTAMINEN 20

5.1 Tehtävälista TODO 22

5.2 Tiedostojen liittäminen sivuihin 29

5.2.1 Lomakepohjaiset ratkaisut – WYMeditori ja Drupal-tiedostonsiirto 29

5.2.2 Kansiopohjainen ratkaisu - File Manager –moduuli TTjako 32

6 DRUPAL-NÄKYMÄN HALLINTA 35

6.1 Lohkon muodostus 35

6.2 Views-moduuli 37

6.3 Kalenterinäkymä 39

7 KÄYTTÖLIITTYMÄ 41

7.1 Ulkoasu 43

7.2 HTML- ja CSS-koosto 44

7.3 HTML-kooston sovitus Drupal-teemaksi 46

8 JOHTOPÄÄTÖKSET 48

LÄHTEET 50

KUVAT

Kuva 1. Drupalin tietovirran rakenne (mukailtu Drupal.org 2011c). 15
Kuva 2. Käytössä olevien moduulien lataus module.inc TiliKoivu intranetissä. 18
Kuva 3. Drupalin sites-kansion rakenne. 20
Kuva 4. TODO- moduulin kutsu. 21
Kuva 5. Define-komento Drupalissa TODO-moduulissa. 23
Kuva 6. TODO-moduulista tuodut noden kentät. 24
Kuva 7. TODO-listauksen näkymä järjestelmässä. 25
Kuva 8. Nodeen liitetty aloituspäivämäärän määritys aikaleiman avulla. 26
Kuva 9. *hook_load()* moduulin toiminta. 27
Kuva 10. TinyMCE-editori. (Tinymce.com 2011) 29
Kuva 11. Käytössä oleva WYMeditor. 30
Kuva 12. Drupalin omat tiedostonsiirtoasetukset. 31
Kuva 13. Monitasoinen tiedostonhallinta liitettäessä tiedostoa uuteen solmuun. 32
Kuva 14. Kansion siirto moduulissa, $_post ja rawurldecode(). 34
Kuva 15. Juurisijainnin arvon määritys (Api.drupal.org 2011c). 34
Kuva 16. Drupal-intranetlohkot. 35
Kuva 17. Lohkon muodostava koodi Drupal 6:ssa. 36
Kuva 18. Dokumentit-näkymän asetukset. 38
Kuva 19. Kalenteri intranetin etusivulla. 39
Kuva 20. Kalenterinäkymän Views-moduulilla luodun näkymän asetukset. 40
Kuva 21. Päänavigaation ulkoasu. 44
Kuva 22. TiliKoivu .info tiedosto. 47

1 JOHDANTO

Monessa yrityksessä on käytössä sisäisiä verkkoja, ja vielä useammassa liiketoimintaan liittyvää sisäistä materiaalia. Jokainen yritys tarvitsee järjestelmän tai järjestelmiä, joilla hallita yrityksen sisäistä tietovirtaa. Yritystoimintaan liitetään nykyisin osaamisalueita useasta eri lähtökohdasta, miksi yrityksen työntekijöiden tulee olla laajalta osaamis-alueelta. Eri osaajien välillä tulee olla yhdistävä työkalu joka sitoo työmenetelmät yhtenäisiksi ja tekee työnteosta mahdollista eri ammattilaisten välillä. Tietopohja työntekijöiden kesken on laajempi kuin vasta muutama vuosikymmen sitten, ja ajan kanssa tämä tietopohja lisääntyy entisestään. Liiketoiminnan kannalta on kannattavaa jakaa tietopohja vapaammin työyhteisön kesken.

Tämän projektin aikana suunnitellaan ja toteutetaan intranet Drupal-järjestelmällä. Intranet on mm. yrityksen sisäiseen tiedon-, projektin-, ja ajanhallintaan tarkoitettu järjestelmä. Nämä järjestelmät voivat olla massatuotteita yrityksiltä, tai pitkälle räätälöityjä yksiköitä juuri kyseessä olevan yrityksen käyttöön. Tämä opinnäyte edustaa jälkimmäistä tyyliä toteuttaa intranet.

Alun kappaleiden teorioita hyödynnetään käytäntöön tavalla, jolla ne muodostavat räätälöidyn kokonaisuuden asiakkaalle käyttäen avointa sisällönhallintajärjestelmää. PHP-pohjaista Drupalia hyödyntämällä kootaan intranet, jota on helppo laajentaa, uudistaa ja ylläpitää. Samalla hahmotetaan moduulien toimintaa ja sitä miksi modulaarisuus on kasvattanut suosiotaan avoimen lähdekoodin sisällönhallintajärjestelmissä.

Tässä opinnäytteessä tarkkaillaan intranetin suunnittelua ja toteutusta Drupal-alustalla. Tavoitteena on käyttää järjestelmää pohjana, jolla tuoda käyttäjille intranetin toiminnallisuus. Näitä toiminnallisuuksia käsitellään myöhemmissä luvuissa, erityisesti luvuissa 5. ja 6. Moduulien avulla järjestelmään lisätyn toiminnallisuuden tavoitteena on sisällyttää valmiiseen intranetiin

projektihallinnollisia ominaisuuksia ja tarjota käyttäjilleen työntekoa tukeva ympäristö.

Tekniikan maailmassa käytettävyys on jäänyt vähemmälle huomiolle, ja peruskäyttäjille suunnattuja käyttöliittymiä on harvassa. Toisaalta peruskäyttäjälle suunnatut käyttöliittymän ominaisuudet eivät yllä nk. ”tehokäyttäjän” tasolle. Käyttöliittymien suunnittelijatkaan eivät aina ymmärrä käyttäjien vaatimuksia ja tarpeita, jolloin valmiin järjestelmän ominaisuudet eivät vastaa käyttäjän odotuksia.

2 PROJEKTIN ESITTELY JA RAJAUS

Viimeisenä opiskeluvuotenani toimin ICT-Portti – hankkeessa opiskelijaprojektipäällikkönä Turun ammattikorkeakoulun puolesta. ICT-Portin kautta tuli tietooni projekti, josta sain aiheen opinnäytteekseni. Projektin aikana tuli toteuttaa web-pohjainen intranet avoimen lähdekoodin alustalla. Asiakkaalla ei ollut aikaisempaa intranetiä, joten sain vapaasti päättää intranetin teknisestä toteutuksesta. Sisällön toimittamisesta intranetiin huolehtii asiakas.

Projektin asiakkaana oli tilitoimisto Tili-Koivu Oy, jonka yhteyshenkilönä toimi Procexon Oy:n Mika Kallioniemi. Projektissa oli siis osallisena itseni lisäksi kolme toimeksiantajaa: ICT-Portti, Procexon ja Tili-Koivu.

Mika Kallioniemi Procexon Oy:stä toteuttaa liikkeenjohdollista konsultointipalvelua yrityksen alasta riippumatta. Hän toimii konsulttina uudistaen asiakasyrityksen toimintaa tavoitteena sertifioida yritys vuoden 2012 aikana. Tili-Koivu Oy on jo yli 40 vuotta toiminut taloushallinnon alalla. Seuraavan vuoden kuluessa Tili-Koivu aikoo muokata toimintaansa ja haluaa intranetin avulla parantaa palveluaan asiakkaille, yhtenäistää nykyisin käytössä olevia ohjelmistoja yhteen portaaliin ja parantaa työntekijöiden välistä kommunikointia.

ICT-Portti on EU-rahoitteinen Turun alueen korkeakoulujen yhteistyöhanke, joka toteuttaa projekteja pk-yrityksille Varsinais-Suomen alueella. ICT-Portti antaa asiantuntevaa koulutusta ja tukea erinäisissä tietotekniikkaan ja tietoturvaan liittyvissä asioissa. Jokaiselle lukukaudelle valitaan Turun ammattikorkeakoulun opiskelijoista projektipäällikkö ICT-Porttiin toimimaan yhdessä yliopiston ja Åbo Akademin opiskelijaprojektipäällikön kanssa.

Tämän projektin tavoite oli intranetin suunnittelu ja pilotointi yritykseen. Sen ensisijainen tehtävä on esittää käyttäjille päivittäistä liiketoimintaa tukeva toimintajärjestelmä ohjeistuksineen ja materiaaleineen sopivassa graafisessa muodossa. Materiaalia on esimerkiksi laatukäsikirjan verkkoversio, prosessikaaviot hyperlinkkeineen, toiminta-ohjeistukset, kuvat ja muu

materiaali. Linkitykset johtavat ulkoisiin järjestelmiin, kuten palkanlaskennan Severa-ohjelmistoon, jotka jäävät muuten projektin ulkopuolelle.

Tarkoituksena oli mm. nopeuttaa ja selkeyttää yrityksen sisäistä kommunikointia, mikä projektin aloitushetkellä oli jäänyt hektisen työtahdin varjoon. Asiakasyrityksen materiaalin keskittäminen yhteen paikkaan oli myös tärkeä kriteeri. Intranetin tavoitteena on olla tarpeeksi yksinkertainen, jolloin käyttäjät pystyvät omaksumaan sen monipuolisen käytön nopeasti. Intranetin on erottauduttava edukseen käytettävyydessä ja huomioitava myös uuden oppimiseen ja omaksumiseen tarvittava aika.

Seuraavat luvut kuvaavat yleisellä tasolla intranetin ominaisuuksia, mutta toteutuksen osalta tämä intranet voisi olla käytössä lähes minkä tahansa alan yrityksessä. Näiden alun kappaleiden jälkeen en siis viittaa Tili-Koivun toimialaan tai sen erityispiirteisiin.

Intranetin ja opinnäytteen yhdistäminen samaan projektiin oli uusi ulottuvuus ICT-Portissa, joten projektin rajausta sopeutettiin kattamaan opinnäytteen kriteerit. Aikataulullisesti projektia tuli myös rajata tiukemmaksi, minkä vuoksi kaikkia asiakkaan vaatimuksia ei pystytty toteuttamaan tässä projektissa.

Dokumenttien hallintaan liittyvät toiminnot olivat tärkein kokonaisuus intranetin toteutuksessa. Tähän tuli sisällyttää eri muodossa olevia dokumentteja, kaavioita ja kuvioita. Toisena kokonaisuutena toteutettiin laatukäsikirjan runko elektronisessa muodossa siten, että yritysasiakas pystyy ISO9001-sertifioimaan yrityksen vuoden 2012 loppuun mennessä.

Muita pienempiä kokonaisuuksia muodostui yritysasiakkaan omien toiveiden mukaan. Tilinpäätösmuistio toteutettiin intranetiin muistiona, johon käyttäjät voivat tehdä merkintöjä edetessään tilinpäätösprosessissa. Jokaiselle tilitoimiston asiakkaalle luodaan oma muistio tilinpäätöksestä, joka lopuksi tulostetaan allekirjoitettavaksi tilinpäätöksen hyväksymiseksi. Tulostaminen on välttämätön osuus, koska yrityksen tulee otattaa asiakkaan allekirjoitus tähän dokumenttiin merkiksi tilinpäätöksen hyväksynnästä.

Lisäksi intranetiin haluttiin lisätä projektinhallinnallisia elementtejä. Näitä toteutettiin mm. kalenterilla, johon lisättiin erillisiä toimintoja, jotka mahdollistavat merkintöjen teon. Tässä haluttiin välttää päällekkäisyyttä jo käytössä olevaan Severa-järjestelmään, johon työntekijät merkitsevät työaikansa, poissaolonsa ja muut työajan kannalta merkittävät huomiot. Kalenterin päätehtäväksi rajattiin tehtävien esittäminen intranetin pääsivulla, minkä lisäksi käyttäjät voivat lisätä kalenteriin omia merkintöjään.

Projektinhallinnasta tehtiin mahdollisimman yksinkertainen käyttäen vain perusominaisuuksia. Asiakkaan toiveena oli, että intranetissä pystyttäisiin seuraamaan tehtävien etenemistä, ja jakamaan vastuualueita työntekijöiden kesken. Tämä toteutettiin erillisellä moduulilla, jonka avulla käyttäjät voivat jakaa toisilleen nk. vastuutehtäviä. Kokonaisuudessaan ratkaisu kattaisi tehtäville eri prioriteetteja, kuvauksia ja vaihtoehtoja, joilla yksilöidä tehtäviä.

Asiakastietoja tai muuta salassapidettävää materiaalia ei tulla sisällyttämään intranetiin niiden vaatiessa esimerkiksi henkilötietolain vaatimusten mukaisen käsittelyn. Tällöin myös tietoturvalle tulisi lukuisia uusia vaatimuksia. Näiden toteuttaminen on loogisinta tehdä erillisenä projektina intranetin käyttöönoton jälkeen, jos sinne päätetään keskittää uusia toimintoja.

Asiakas oli aluksi kiinnostunut myös toteuttamaan verkostoitumiseen ja tiedottamiseen liittyviä osioita intranetiin. Nämä toteutukset jäivät projektin ulkopuolelle, koska niillä ei koettu saavutettavan lisähyötyä olemassa olevien kotisivujen lisäksi.

3 ALUSTAN VALINTA

Intranetin alustaa ei aikataulusyistä lähdetty toteuttamaan tämän projektin yhteydessä, vaan päätettiin ottaa käyttöön jokin olemassa oleva järjestelmä. Kriteereinä oli mm. käyttöönoton nopeus, ylläpidon yksinkertaisuus, olemassa olevat ominaisuudet, muunneltavuus, laajennettavuus ja soveltuvuus intraksi. Projektin alussa asetettiin päämääräksi, että kesään 2011 mennessä intranet on yrityksen käytössä. Tämä loi luonnollista painetta saada alusta työstettäväksi mahdollisimman nopeasti. Tämän jälkeen aikataulusta suuri osa kuluu alustan soveltamiseen tarkoitukseensa.

Alustan tulee olla joustava, jotta siihen saadaan asiakkaan pyytämiä ominaisuuksia kohtuullisella työ- ja aikapanoksella. Koska tämän projektin osuuteen kuuluu intran teko, käyttöönotto ja testaus, jää ylläpito asiakkaalle. Tämä tarkoittaa vaatimuksellisesti sitä, että ylläpidon on luonnistuttava myös henkilöltä, joka ei ole teknisen alan ammattilainen. Alustassa on erotettava nk. frontend ja backend. Normaalin käyttäjätason omaavat käyttäjät voivat käsitellä intraa rajoitetummin oman käyttöliittymän läpi, kun taas ylläpito voi hallita intraa erillisellä heille suunnatulla käyttöliittymällä, josta on mahdollista hallita intran toimintaa ja muokata sisältöä.

3.1 Vaihtoehdot

Vapaan lähdekoodin sisällönhallintajärjestelmiä on useita. Suosituimpia näistä ovat jo useamman vuoden olleet WordPress, Joomla! ja Drupal (Water & Stone 2010). Water & Stonen julkaisemassa raportissa on mukana 20 avoimen lähdekoodin sisällönhallintajärjestelmää.

Wordpress on henkilökohtainen blogialusta joka lupaa webstandardin mukaista, helppokäyttöistä työkalua kaikkien käyttöön (Wordpress.org 2011a). Lisäosilla ja muokattavilla teemoilla tästä alustasta on mahdollista koota kattava intranet. Wordpress hyödyntää interaktiivista ajattelutapaa ja mahdollistaa useita sosiaalisen median käyttömahdollisuuksia RSS-syötteistä blogeihin ja foorumiin. Kokonaisuutena sen on tarkoitus yhdistää työyhteisöä ja toimia

monipuolisena kommunikaatiovälineenä. Wordpress toimii PHP- pohjaisena ja käyttää tietojen tallentamiseen MySQL-tietokantaa. Tällä hetkellä Wordpressiä alustana käyttäviä sivustoja on tuhansia, näiden joukossa mm. Axl Smithin kotisivut, The Royal Ballet-sivusto ja University of Toronto Alumni Website-sivusto (Wordpress.org 2011b).

Joomla! on tällä hetkellä käytetyin avoimeen lähdekoodiin perustuva sisällönhallintajärjestelmä. Joomlalla voidaan luoda kattavia järjestelmiä joiden avulla voidaan hallita sivustoja ja portaaleja internetissä. Sivustojen lisäksi sillä on tehty laajalti intranetratkaisuja, verkkokauppoja sekä esimerkiksi valtiohallinnollisia sovelluksia (Joomla.org 2011a). Joomla! pohjautuu Mambo-ohjelmistoon josta se erkaantui omaksi projektikseen vuoden 2005 aikana (Analogik.org 2011). Joomlan suosio on hiipunut hiljattain, mutta syyksi vaikuttaa nousseen uuden version julkaisupäivämäärän läheneminen. Käyttäjien kasvun jatkuminen uuden version tullessa käyttöön saattaa nostaa Joomlan taas suosituimmaksi järjestelmäksi. Tällä hetkellä Joomlaa alustana käyttäviä sivustoja on mm. Oklahoma State University, Olympus Australia ja Menhec Blog (Joomla.org 2011b).

Drupalin suosio perustuu osittain sen teknisiin ominaisuuksiin, mutta samalla sen tekninen vaativuus pudottaa Drupalin suosiossa esimerkiksi Joomlan alapuolelle. Perusasennuksella sen saa käyttöönsä hetkessä ja sivuston julkaistavaksi muutamassa tunnissa. Moduulien avulla sivustoa on mahdollista muokata lähes rajattomasti. Itse tehdyillä moduuleilla sivustoa voi laajentaa entisestään ja julkaista omia moduuleja myös muiden käyttöön. Moduulien teko Drupaliin on yksinkertaista ja sen avoimessa yhteisössä on ohjeita, kuinka toteuttaa moduuleja jaettavaksi muiden käyttöön. Drupalia alustana käyttäviä sivustoja ovat mm. MTV United Kingdom, Penn State University ja AOL Corporate (Drupal.org 2011a).

3.2 Valinta

Vaihtoehdoista Drupal osoittautui opinnäytteen kannalta parhaaksi. Drupal kiinnosti teknisyytensä vuoksi, mutta myös oma osaamiseni Drupalista oli

lähtötilanteessa alkeellista ja teoriatasoista. Halu oppia käyttämään järjestelmää oli siis suuressa osassa valintaa tehdessä. Tällä alustalla pystyin myös hyödyntämään tietokantaosaamistani, sekä PHP- ja suunnittelutaitojani intran toteutuksessa.

Drupal tuntemus toimii myös hyvänä pohjana muiden järjestelmien opetteluun, jotka ovat toiminnoiltaan yksinkertaisempia, rajoitetumpia ja tarjoavat lähtökohdiltaan valmiimpia kokonaisuuksia käytettäväksi. Drupalin perusasennus antaa käyttöön lähes tyhjän alustan, jota laajentamalla voi luoda niin yksinkertaisia kuin monimutkaisiakin kokonaisuuksia.

Alustalla on myös uskollinen ja aktiivinen yhteisö internetissä, ja näiden avulla Drupal laajenee ja monimuotoistuu jatkuvasti. Yhteisössä on peruskäyttäjien lisäksi myös lukuisia lahjakkaita ohjelmoijia, jotka etsivät ja poistavat moduuleista virheitä. Yhteisössä on myös mahdollista julkaista omia moduulejaan muiden käyttöön GNU-lisenssillä. Lisenssi on yleisesti käytetty julkaistaessa avoimen lähdekoodin materiaalia.

4 DRUPALIN ESITTELY

Drupal otti ensiaskeleensa vuonna 2000 Antwerpenin yliopistossa, kun internetyhteydet tulivat käyttöön kampusalueella, jossa se aluksi suoritti intranetin toimea kahdeksan opiskelijan kesken. Varsinaisina perustajina pidetään Dries Buytaertia ja Hans Snijderiä. Drupal julkaistiin verkossa alunperin nimellä drop.org erinäisten kommellusten kautta. Vähitellen yhteisön jäsenet alkoivat keskustella uusista web-teknologioista ja niiden kehittämisestä eteenpäin. Näitä kehittämisideoita testattiin itse drop.orgiin. Vuonna 2001 tämä ohjelmisto päätettiin julkaista nimellä Drupal, joka juontuu englantilaisesta tavasta lausua hollantilainen sana ”druppel” eli pisara (Drupal.org 2011b).

Drupalin asennukseen tarvitaan vain core-tiedostot, jotka ladataan palvelimelle tai halutulle testialustalle. Tämän jälkeen voi suorittaa automaattisen PHP-skriptin, joka täyttää tietokannan taulut, asentaen Drupalin perusosat valittuun sijaintiin. Vaihtoehtoisesti asennuksen voi tehdä myös komentoriviltä.

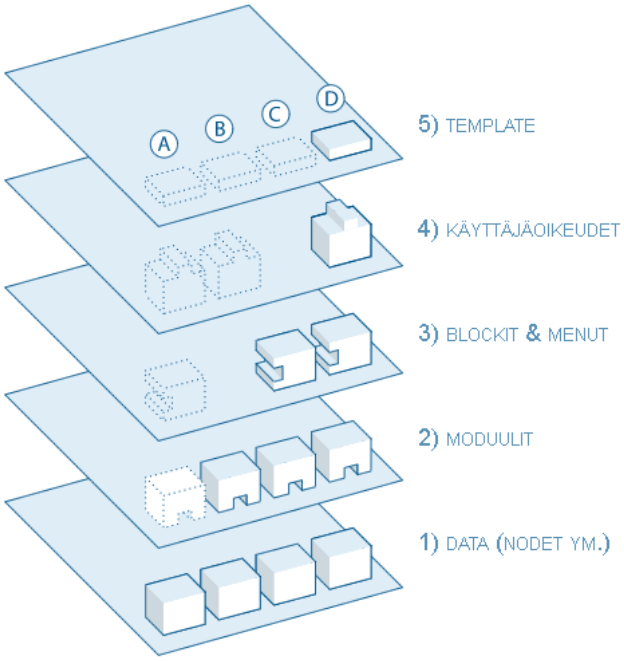
Drupal coreksi kutsuttu perusasennus sisältää vain tärkeimmät toiminnalliset osat, ja sellaisenaan core on melko vaatimaton käytettäväksi. Toiminnallisuus muodostuu pääosin moduuleista, jotka lisätään Drupaliin. Core kuitenkin tarjoaa graafisen käyttöliittymän, jonka läpi on helppo aloittaa sivuston rakentaminen. Myös käyttöliittymää on mahdollista muokata teemoiksi kutsuttujen HTML-, PHP- ja CSS- kokonaisuuksien avulla.

Erilaiset sisällönhallinta-, eli CMS-järjestelmät toimivat periaatteella, jossa yksi jakaa kaikille käyttäjille sisällön ja mahdollisuudet toimia tietyn kaavan mukaan. Drupal toimii kehittäjilleen yhtäaikaisesti CMS-järjestelmänä ja frameworkinä, eli CMF-järjestelmänä. Käyttäjäoikeuksien avulla on mahdollista tuoda käyttäjille vaihtoehto muokata sivuston sisältöä ja rakennetta kääntäen yleisen ”yhdeltä monelle” –periaatteen ”monelta monelle” -periaatteeksi. (Drupal.org 2011c)

Drupal toimii vapaana abstraktiona tarjoten käyttäjälleen vapauden muokata järjestelmän käyttötarkoitusta. Perusasennuksen yhteydessä tulee mukana osia, joilla käyttäjä voi nopeasti rakentaa itselleen toimivan sivuston, mutta järjestelmässä ei ole oletuksia sen lopullisesta käytöstä. (Drupal.org 2011c)

4.1 Drupal toiminnallisuuksia

Drupalin sisäinen tiedonkulku koostuu viidestä tasosta. Tietovirran suunta on aina alhaalta ylöspäin, jolloin ylempänä olevat tasot voivat muokata alempia tasoja. Tämä koskee mm. ulkoasun toimintoja ja käsittelyä (Drupal.org 2011c).



Kuva 1. Drupalin tietovirran rakenne (mukailtu Drupal.org 2011c).

Alimpana kuvassa 1. on kaikki järjestelmässä oleva data. Tätä dataa jalostetaan järjestelmän tietovirrassa läpi tasojen toiminnallisuuksien, kyselyjen ja ulosannin avulla mm. teemojen välityksellä. Tietokannan tauluista voidaan kyselyjen avulla tuottaa käyttäjälleen hyödyllistä infoa. (Drupal.org 2011c)

Tiedonkulun lisäksi Drupalissa on konsepteja, jotka jokaisen Drupalin kanssa työskentelevän tulee ymmärtää ennen työskentelyn aloittamista.

- Moduuli

  Drupalin osa jolla järjestelmään tuodaan toiminnallisuutta. Myös käyttäjistä muodostuva yhteisö voi luoda moduuleja ja näin laajentaa omaa Drupal-järjestelmäänsä. Intranetin tapauksessa moduuleilla on laajennettu järjestelmää siten että siinä on intranetille ominaisia

toiminnallisuuksia (Drupal.org 2011d). Näistä toiminnallisuuksista lisää kappaleissa 5. ja 6.

- Käyttäjä, lupa, rooli

Jokaisella järjestelmällä on oletettavasti käyttäjiä. Heillä on usein erilaisia tehtäviä ja toimintoja, joita he haluavat toteuttaa järjestelmässä. Tietoturvan kannalta käyttäjien toimintaa on hyvä rajata, joten järjestelmään tulee luoda lupia ja rooleja eri käyttäjille (Drupal.org 2011d).

- Node

Kaikkea Drupaliin syötettyä ja tallennettua tietoa kohdellaan nodeina, eli solmuina. Sivut, blogi-kirjaukset ja päävalikot ovat järjestelmässä kaikki nodeja. (Drupal.org 2011d)

- Kommentti

Kommentit ovat Drupalin sisällön tyyppi, joka on yleensä käyttäjän syöttämä tieto johonkin tiettyyn nodeen. (Drupal.org 2011d)

- Taxonomy

Drupalin tapa luokitella sisältöä. Käyttäjä voi luoda taksonomioita tarpeiden mukaan ryhmiksi, jotka sisältävät toisiinsa liittyvää tietoa. Taksonomioilla voidaan luokitella monimutkaisia tai vaikeaselkoisia rakenteita käyttäjäystävällisiksi kategorioiksi. (Drupal.org 2011d)

- Tietokanta

Jokaisella Drupalin sisällön tyypillä on oma tietokantataulunsa. Moduulien avulla luotavat uudet sisältötyypit luovat tietokantaan aina uuden taulun. Nodeilla, käyttäjillä, rooleilla, ja muilla sisältötyypeillä on omat taulunsa tietokannassa. (Drupal.org 2011d)

- Polku

Drupal luo URL -polun tietokannan kyselyn avulla. Yleisesti Drupal ohjaa käyttäjän jollekin nodelle, joka moduulien avulla päättää mitä sisältöä käyttäjälle esitetään. (Drupal.org 2011d)

- Teema

Teema päättää Drupal-sivuston ulkoasun. Se sisältää usean PHP-tiedoston sekä CSS-tiedostoja. (Drupal.org 2011d)

- Alue, blokki

Drupal jakaa sivun sisällön sille osoitettuun alueeseen. Blokit ovat sivustoon kiinnitettyjä informaation ja toiminnallisuuksien esittämiseen erikoistuneita alueita. Blokeissa voidaan esittää esimerkiksi sivuston navigaatio tai valikko. (Drupal.org 2011d)

- Valikko

Drupalissa on kolmentyyppisiä valikoita: ensisijaiset linkit, toissijaiset linkit sekä navigaatio. Kaksi ensimmäistä valikkoa on sivuston ylläpitäjän kokoamia, jotka esitetään käyttäjälle. Kolmas valikko eli navigaatio noudattaa catch-all –skriptiä, joka esittää kaikki järjestelmän linkit. Linkit saadaan kolmannessa vaihtoehdossa sekä ylläpitäjältä että asennetuilta moduuleilta. (Drupal.org 2011d)

4.2 Drupal funktionaalisuuksia

Drupalissa tapahtumat ovat pitkälti tietokantakyselyjen tuloksia ja niiden esittämistä käyttäjän haluamaan muotoon. Nämä tapahtumat voidaan löyhästi jakaa ulkoisiin ja sisäisiin tapahtumiin. Ulkoiset tapahtumat olisivat esim. käyttäjän tekemiä kyselyjä, moduulien asennuksia ja sivuston selaaminen. Sisäisiä tapahtumia kutsutaan myös nimellä *hooks*, *callbacks* tai koukku. Nimi callback on tosin harhaanjohtava nimeämisperiaatteiden takia, sillä koukut eivät rekisteröidy kuuntelijalla. (VanDyk & Westgate 2007, 4-5.)

Drupalin asennuksen mukana tulee joukko *include*- funktioita, joilla Drupalin perustoiminnot ovat käytettävissä. Kuvassa 2 nähdään moduulit järjestelmän käyttöön tuova funktio. Järjestelmään lataamisen lisäksi on käytännöllistä listata ne myös hallintapaneeliin nimen ja kuvauksen kanssa. Toistamalla funktiota alkaen riviltä 6. saadaan moduulien nimet latauksen aikana, minkä jälkeen funktio lisää niihin kuvauksen.

```
1  function module_load_all() {
2    foreach (module_list(TRUE, FALSE) as $module) {
3      drupal_load('module', $module);
4    }
5  }
6  function module_iterate($function, $argument = '') {
7    foreach (module_list() as $name) {
8      $function($name, $argument);
9    }
10 }
11
```

Kuva 2. Käytössä olevien moduulien lataus module.inc TiliKoivu intranetissä.

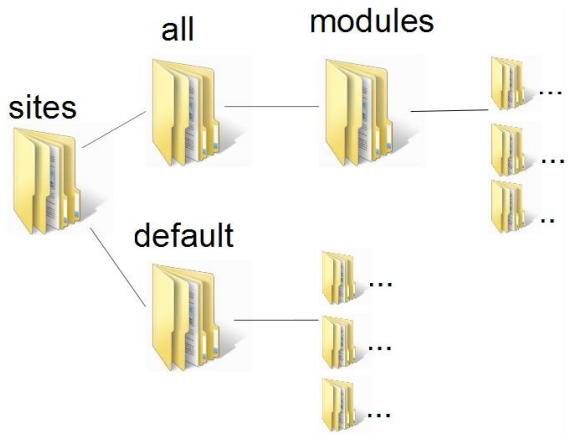
Module.inc tiedosto sisältää Drupal-moduulien lataukseen ja interaktioihin liittyvät funktiot. Core-asennuksen yhteydessä tulee myös lukuisia muita funktioita, joiden tehtävänä on huolehtia ja toteuttaa Drupalin perustoiminnot. Omia sivuja on mahdollista luoda myös kirjoittamalla omia funktioita, joihin voi vapaasti yhdistellä ja rakentaa kokonaisuuksia, jotka parhaiten vastaavat käyttäjän tarpeita. (Packtpub.com 2011)

Drupal tarjoaa myös kevyen rajapinnan, jossa kehittäjien on mahdollista ylläpitää useaa tietokantapalvelinta vähällä vaivalla säilyttäen saman ohjelmointipohjan. Tätä rajapintaa kutsutaan Drupal abstraction layeriksi. Rajapinnan tarkoituksena on säilyttää SQL-syntaksi ja tehokkuus samalla kun Drupal ohjaa kyselyjä joiden syntaksi eroaa toisistaan, eri palvelimille suorittaen samalla olennaiset turvatoimenpiteet. (Drupal.org 2011e.)

Drupalilla kyselyjä voidaan suorittaa suoraan tietokantaan PHP:llä. Vaihtoehtoisesti kutsutaan Drupaliin kuuluvia funktioita. Suurin osa näistä kyselyistä voidaan hoitaa funktioilla *db_query()* tai *db_query_range()*. Funktiolla *db_query()* suoritetaan tavanomaisia kyselyitä kun taas *db_query_range()* on ennalta määritetyn joukon kysely (Api.drupal.org 2011a).

5 MODUULIN KIRJOITTAMINEN

Drupalissa moduulit sijaitsevat yleisesti omassa kansiossaan sites/all/modules. Kuvassa 3. on havainnollistettu sites-kansion rakennetta. Tämä mm. siksi, että lisätyt moduulit eivät aiheuta konflikteja Drupalin coren kanssa, jolloin järjestelmän päivittäminen on turvallisempaa.



Kuva 3. Drupalin sites-kansion rakenne.

Kuten teemat, myös moduulit tarvitsevat .info-tiedoston, jossa järjestelmälle kerrotaan moduulin perustiedot. Tietojen tulee olla PHP-standardin mukaisessa muodossa. Tämä erillinen .info-tiedosto auttaa jatkossa muistin tarpeen minimoimista, kun sivuja ladataan selaimeen. (VanDyk & Westgate 2007, 12)

Kuvassa 3. esitetyssä sijainnissa ja oikealla tiedostopäätteellä olevat moduulit ovat käytössä järjestelmän hallintapaneelista asennuksen jälkeen. Varsinainen moduulitiedosto ilmaistaan Drupalille lisäämällä tiedostonimen perään .module. (VanDyk & Westgate 2007, 12)

Kun nämä tarvittavat tiedostot on luotu ja tallennettu, moduuli on näkyvissä Drupal-hallintapaneelissa. Tätä tyhjää moduulia ei vielä ole sisällytetty mihinkään, eikä moduuli palauta mitään toiminnallisuutta käyttäjälle. Luvussa 4.2 on maininta Drupalin sisäisistä kutsuista, eli hooks –tai callback-toiminnoista. Drupalin ja uuden moduulin välille on tehtävä uusi sisäinen kutsu, jotta järjestelmä voi tuoda käyttäjälle moduulin toiminnallisuuden. (VanDyk & Westgate 2007, 11-12)

Yhteyden luomiseksi tulee käyttää erästä Drupalin perustoimintoa. Käyttäjän luodessa järjestelmään pyynnön, järjestelmä kysyy moduuleilta kaikki niiden tarjoamat vastaukset. Kun järjestelmä löytää pyyntöön sopivan vastauksen se lähettää vastauksen sisältävään moduuliin vaihtoehtoisen parametrin kysyen voidaanko vastaus tallentaa välimuistiin. Koukkujen *function()* nimi muodostuu aina järjestelmän luomasta loppuosasta, joka riippuu kutsun ominaisuudesta sekä vastauksen antavan moduulin nimestä. (VanDyk & Westgate 2007, 13)

Kuvassa 4. on esitelty moduulin navigaation lataava ja noutava koukku. Intranetissä on ylläpitäjille myös erillinen moduuli, joka tarjoaa heille oman navigaation helpottamaan ylläpidon toimenpiteitä. TODO-moduulin tapauksessa navigaatiolla tarkoitetaan ylläpitäjille suunnattua moduulin asetukset sisältävää valikkoa. Kuvassa ensimmäisellä rivillä aloitetaan funktio TODO-moduulin navigaation lataamiseen. Rivillä kaksi näkyy Drupalin *include()*-funktio, johon sisällytetään halutut navigaatiot niiden nimien mukaan. Funktio yhdistää TODO-moduulin navigaation ylläpitäjille suunnatun moduulin navigaatioon.

```
1  function to_do_menu() {
2    module_load_include('admin.inc', 'to_do');
3    return to_do_menu_array();
4  }
5
```

Kuva 4. TODO- moduulin kutsu.

Luotu moduuli on nyt havaittu järjestelmässä ja se otetaan huomioon käsiteltäessä pyyntöjä. Moduulin vastauksen on kuitenkin hyvä sisältää myös jotakin käyttäjälle hyödyllistä. Näiden järjestelmälle ominaisten toimenpiteiden ja kutsujen jälkeen moduuli voi sisältää lähes mitä tahansa tai ainakin mitä tahansa mikä on toteutettavissa PHP-kielellä. (VanDyk & Westgate 2007, 13)

5.1 Tehtävälista TODO

TODO-moduuli on yksi tärkeä ominaisuus intranetissä. Moduuli käyttää hyväkseen mm. Drupalin muita moduuleja, kuten cck-, date- ja views-moduuleja. Kokonaisuudessaan moduuli antaa käyttäjän lisätä joukon tietoja tietokannan tauluun, jotka yhdistetään moduulin avulla. Yhdistävän moduulin avulla tiedot voidaan myöhemmin hakea ja ajoittaa niille toimenpiteitä. Moduuli laajentaa järjestelmän peruselementtiä node-tyyppiä.

Drupalissa on monia projektinhallinnallisia moduuleja, jotka tarjoavat kokonaisvaltaisia ratkaisuja, joilla projektin seuraaminen ja aikataulutus siirtyisivät kokonaisuudessaan järjestelmän sisäisiksi. Asiakkaalla oli kuitenkin käytössään jo mm. työajanseurantaan tarkoitettu työkalu Severa-järjestelmässä. Tämän TODO-listauksen tuli välttää turhaa päällekkäisyyttä näiden kahden järjestelmän välillä. Severan ulkoisesta järjestelmästä ei ollut mahdollista siirtää näitä toiminnallisuuksia näkymään Drupalissa.

Case tracker- ja Storm- projektihallinnan moduulit ovat tällä hetkellä Drupalin yhteisön suosituimpia työkaluja projektinhallintaan. Näiden tutkimisesta oli hyvä aloittaa ratkaisun hakeminen. Vaihtoehtoina nämä moduulit osoittautuivat liian laajoiksi asiakkaalle, sillä ne tarjosivat liian syvällisiä toimintoja tämän intranetin tarkoituksiin. Lisäksi toiminnot tarjosivat liikaa samankaltaisia ominaisuuksia Severa-järjestelmän kanssa. Lopullinen ratkaisu on hyvä, riisuttu, asiakkaan tarpeet täyttävä yhdistelmä molempien järjestelmien oivalluksista. Drupalin projektinhallintamoduuleista on yhteisössä toteutettu vertailu osoitteessa http://groups.drupal.org/node/17948.

Case tracker on lähemmin katseltuna usein asiakaspalveluissa käytetyn tikettijärjestelmän korvaaja Drupalissa. Moduuli tarjoaa mahdollisuuden järjestää kyselyjä niiden kiireellisyyden mukaan. Se hyödyntää myös Drupalin muita moduuleja, kuten views –moduulia koostaakseen käyttäjälle listauksen tiketeistä tärkeysjärjestyksessä. Case tracker oli lähtökohtaisesti hyvä toteutus asiakkaan määrityksistä, sillä se sisälsi jo toimivan ratkaisun tehtävän osoituksesta vastuuhenkilölle sekä tehtävän eri tilat aikajanalla. Sisäiselle

järjestelmälle asiakaspalvelu ei ole kriittistä toimintaa, eikä tähän toiminnalliseen ominaisuuteen siten tule keskittyä liiaksi. Case tracker-moduulia voi testata vapaasti osoitteessa http://drupal6-casetracker.module-demos.org/casetracker.

Storm tarjoaa moduulina vielä laajempaa projektihalllintaa lukuisilla eri sisältötyypeillä varustettuna. Storm on jaettu useaan osa-alueeseen, joista voi valita sopivan kokonaisuuden. Stromista asennetaan hallintapaneeli Drupaliin, johon liitetään muita moduuleja kooten Stormin projektihallinnon ominaisuudet. Stormista saatiin projektihallinnallisten ominaisuuksien ideat TODO -moduulille. Näihin kuuluu mm. date –moduulin käyttö aloituksen ja takarajan asettamiseen. Stormin kehitystä voi seurata osoitteessa http://drupal.org/project/storm.

Moduuli tarvitsee myös muutamia tietokantaan ennalta tallennettuja arvoja ja määrityksiä. TODO –moduulin tapauksessa tehtävälle haluttiin määrätä joitakin käyttäjää informoivia kohtia. Näitä olivat tehtävän tila, prioriteetti, tehtävien järjestelemisessä avustava valintakaava ja määräajan toimintoja määrittävät arvot ja tilanteet. Nämä kaikki voidaan kirjoittaa tietokantaan yksinkertaisen *define()*-komennon avulla, joita esitellään kuvassa 5. *Define()*-komento muistuttaa syntaksiltaan SQL-lausetta. Kuvassa *define()*-komento luo järjestelmälle listat TODO-moduulin tila- ja prioriteetti-ominaisuuksille. Komennolle luetellaan oikean listan nimi, alaviivalla erotellaan uuden listauksen kohdan otsikko, ja pilkun jälkeen sen painoarvo, jolla määritetään otsikon sijainti listassa. Tuloksena syntyy lomakkeen alasvetovalikko. Painosta puhutaan lisää kappaleessa 5.2.2.

```
5  define('TO_DO_STATUS_NOT_STARTED',        0);
6  define('TO_DO_STATUS_STARTED',            1);
7  define('TO_DO_STATUS_FINISHED',           2);
8  define('TO_DO_STATUS_AWAITING_MORE_INFO', 3);
9  define('TO_DO_STATUS_DELAYED',            4);
10 define('TO_DO_STATUS_CANCELED',           5);
11 define('TO_DO_STATUS_IN_PROGRESS',        6);
12
13
14 define('TO_DO_PRIORITY_LOW',       0);
15 define('TO_DO_PRIORITY_MEDIUM',    1);
16 define('TO_DO_PRIORITY_HIGH',      2);
17 define('TO_DO_PRIORITY_IMMEDIATE', 3);
```

Kuva 5. Define-komento Drupalissa TODO-moduulissa.

Ennen moduulin luontia on käytännöllistä tehdä tämän kaltaiselle tiedolle oma sisältötyyppi. Käyttämässämme Drupal-versiossa tämä vaatii toisen moduulin, CCK-moduulin asennuksen. Tämä mahdollistaa räätälöityjen kenttien luonnin jotka eivät kuulu Drupalin oletuksiin (Drupal.org 2011f). Näihin uusiin kenttiin voidaan tallentaa valintalistoja, tekstikenttiä, kuvia tai ääntä tarpeen mukaan. TODO-moduulin tapauksessa tuli luoda uusia kenttiä, joihin tallettaa päivämääriä sekä date-moduuliin yhteydessä olevaa dataa.

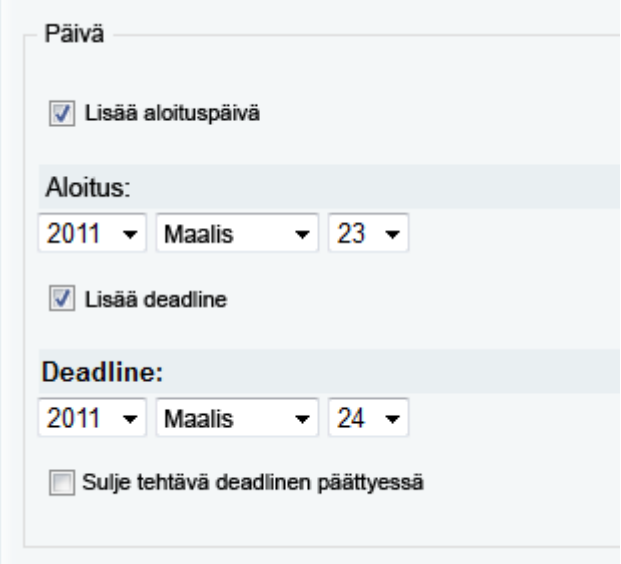
Aluksi moduulissa on lomake, johon luodaan käyttäjälle täytettäväksi muutama kenttä. Nämä kentät tallennetaan myöhemmin kokonaisuutena, joista tehtävä muodostuu. Järjestelmän näkökulmasta nämä käyttäjän tehtävät eivät tässä vaiheessa edellytä Drupalilta mitään jatkotoimenpiteitä, joten ne ovat vain joukko tallennettua dataa Drupalin peruselementtiin nodeen.

Kuvassa 6. nodeen lisätään taulukon muodossa joukko räätälöityjä kenttiä, joihin määritellään niiden nimet ja tyypit. Taulukosta voidaan näin jatkossa tulostettaessa saada kenttien nimet ja niihin lisätyt tietokannan tallenteet. Tämän tyyppinen *_info()* päätteinen tieto tarvitaan Drupalissa jokaiselle uudelle sisältötyypille (Phpeveryday.com 2011). Kuvassa 6. on Drupalin sääntöjen mukaisesti toteutettu taulukko. Noden valmiista kentistä poimitaan TODO-moduulille sopivat kentät, jolloin uusia kenttiä ja sisältötyyppejä ei tarvitse luoda erikseen moduulin tarkoituksiin. Riveillä 46-52 on lueteltu vasemmalla puolella noden sisältämiä parametrejä, ja oikealla on TODO-moduulin määritykset niille parametreille. Tämän taulukon tiedot vaikuttavat siihen mitä käyttäjä voi TODO-merkintään lopulta tallentaa.

```
43  function to_do_node_info() {
44    return array(
45      'to_do' => array(
46        'name' => t('To do'),
47        'module' => 'to_do',
48        'description' => t('tehtävä lisätään käyttäjälle'),
49        'has_title' => TRUE,
50        'title_label' => t('tehtävä'),
51        'has_body' => TRUE,
52        'body_label' => t('tehtäväkuva'),
53      )
54    );
55  }
```

Kuva 6. TODO-moduulista tuodut noden kentät.

Seuraavaksi lisättiin tähän lomakkeeseen toiminnallisuus, joka erottaa sen Drupalin muista lomakkeista, eli kyky määrittää tälle sisältötyypille aloitus –ja lopetuspäivämäärät. PHP:n mukainen *mktime()* –funktio sopii tarkoitukseen mainiosti sen palauttaessa Unixin mukaisen aikaleiman (PHP.net 2011a). Lisäksi tämä aloituspäivä olisi epäkäytännöllistä asettaa johonkin ennaltamäärättyyn päivään, joten se jätetään käyttäjän täytettäväksi. Määräaika on määritelty samalla if-lauseella, ja myös se on käyttäjälle vaihtoehtoinen kohta lomakkeella. Kuvissa 7. ja 8. tätä on havainnollistettu kuvakaappauksella ja ohjelmalistauksella.



Kuva 7. TODO-listauksen näkymä järjestelmässä.

Käyttäjän toimintaa tulee ohjata lomakkeen täytössä. Käyttäjän ei tarvitse nähdä lomakkeella ylimääräisiä kenttiä, vaan ne kentät jotka käyttäjä valitsee valintaruuduista ovat näkyvissä. Kentät ovat kronologisessa järjestyksessä joka helpottaa lomakkeen täyttöä. Päivämäärien asettaminen on käyttäjälle valinnainen optio.

Kuvassa 8. on ohjelmalistaus päivämäärien asettajasta TODO-moduulissa. Drupal käyttää luonnostaan runsaasti taulukoita, joten myös päivämäärät on asetettu taulukkoon. Päivämäärän asetuksessa halutaan myös varmistaa ettei käyttäjä voi syöttää virheellistä arvoa. Jos käyttäjä ei syötä oikeanlaista arvoa

antaa järjestelmä virheen. Oletusarvoksi järjestelmä hakee kuluvan päivän järjestelmän sisäisestä päivämäärästä.

Rivillä 182. aloitetaan *if()*-lause joka tarkistaa onko nodeen asetettu parametri aloituspäivämäärälle. Seuraavalla rivillä tarkistetaan päivämäärien taulukko nodessa, joka sisältää kalenterin päivämäärät. Jos tässä kentässä on päivämäärä tallennetaan se tietokantaan tehtävän yhteyteen. Rivin 184. *if()*-lause tarkastaa lomakkeen valintaruudun aloituspäivämäärän asettamisesta. Jos valintaruutu on valittuna, siirrytään riville 187. Kaikki tämä on vapaaehtoista, joten *if()*-lauseen avulla kenttä voidaan jättää tyhjäksi. Rivillä 187. käyttäjän napsauttaessa lomakkeen valintaruutua hakee järjestelmä *mktime()*-funktion avulla Unix aikaleiman, josta valitaan näytettäväksi kolme argumenttia. Lomakkeessa ne esitetään käyttäjälle alasvetovalikkoina kuten kuvassa 7.

```
182  if (isset($node->start_date)) {
183    if (is_array($node->start_date)) {
184      if (empty($node->include_start_date) || empty($node->start_date)) {
185        $node->start_date = 0;
186      }
187      else {
188        $node->start_date = mktime(
189          0, 0, 0,
190          $node->start_date['month'],
191          $node->start_date['day'],
192          $node->start_date['year']
193        );
194        $node->start_date -= $timezone;
195      }
196    }
197  }
198  else {
199    $node->start_date = 0;
```

Kuva 8. Nodeen liitetty aloituspäivämäärän määritys aikaleiman avulla.

Määriteltyjen toimintojen asetuksen jälkeen niiden kutsuminen jätetään Drupalille kutsumalla sille ominaisia *hook()* toimintoja, kuten *hook_update(), hook_insert()* ja *hook_delete()* (Drupal.org 2011g). Nämä toiminnot ottavat yhteyden Drupalin tietokantaan ja tekevät tarvittavat muutokset. Kuvan 9. esimerkissä on osa *hook_load()*-funktiota, mikä noutaa tietokannasta SQL-lauseen avulla aiemmin tietokantaan määritetyt kentät luettavaksi näkymään.

```
294   $sql = "SELECT item_status, priority, start_date, deadline, date_finished,"
295       . " deadline_event, auto_close, mark_permissions"
296     . " FROM {to_do} WHERE vid = %d";
297   $result = db_query($sql, $node->vid);
298   $return_object = db_fetch_object($result);
299   if (empty($return_object)) {
300     $return_object = new stdClass;
301     $return_object->item_status = TO_DO_STATUS_STARTED;
302     $return_object->priority = TO_DO_PRIORITY_MEDIUM;
303     $return_object->start_date = 0;
304     $return_object->deadline = 0;
305     $return_object->date_finished = NULL;
306     $return_object->deadline_event = 0;
307     $return_object->auto_close = 0;
308     $return_object->mark_permissions = 0;
309   }
```

Kuva 9. *hook_load()* moduulin toiminta.

Kuvassa 9. SQL-lause tarkistaa tietokannan ensin kutsuttavasta datasta, minkä jälkeen *if()*-lauseella täytetään tyhjät kentät. Kuvassa 9. riveillä 300-308 näkyvät TODO-moduulin kaikki käyttäjän näkymään palautetut parametrit. Käyttäjälle funktio muodostaa tallennetuista kentistä intranetiin näkymän, jossa esitetään tallennetut tiedot käyttäjän oman tärkeysjärjestyksen mukaan kokonaisuutena. *If()*-lause estää myös tyhjien arvojen tulostuksen (Daniweb.com 2011).

Käyttäjän yksilöinti moduulissa

Jotta tehtäviä voitaisiin kohdistaa tietylle käyttäjälle sekä tunnistaa tehtävän laatinut käyttäjä, käytetään Drupalin *$user*-objektia. Tähän objektiin voidaan tallentaa käyttäjälle tarpeellista tietoa ja ensisijaisia asetuksia. Tätä objektia voidaan käyttää tunnistamaan järjestelmän käyttäjiä ilman yksilöivän id:n käyttöä. Tämä tunnistaa aina sisäänkirjautuneen käyttäjän ilman että käyttäjän tarvitsee koskaan syöttää omaa id-tunnustaan järjestelmälle. (Api.drupal.org 2011b)

Muille käyttäjille osoitettuja tehtäviä voidaan määrittää yksilöllisen *$uid*:n avulla, joka osoittaa yksittäiseen käyttäjään määritellyn id:n mukaan. Tähän id:hen viittaaminen toimii vain sen olemassaolon ajan. Poistamalla käyttäjä poistuu myös kyseinen id käytöstä. Lisäksi tämän käyttäminen viittamaan omaan käyttäjäänsä voi aiheuttaa konflikteja ja avaa käytetyn käyttäjän id:n tietoturvauhille. Liitettäessä käyttäjän suora id johonkin toimintoon, jokainen *$user*-objekti näkee *$uid*-objektin sisällön. Käytännössä kuka tahansa muu käyttäjä voi käsitellä paljastetun käyttäjän tietoja. Moduulissa *$uid*-objektia käytetään vain muiden käyttäjien viittaamiseen. (Api.drupal.org 2011b)

Drupalissa käyttäjä id 0 ja 1 ovat nk. varattuja tunnuksia, joista toinen viittaa aina järjestelmän ensimmäiseen ylläpitäjään ja toinen kirjautumattomaan käyttäjään joka luodaan asennettaessa Drupalia (Mostrey 2011). Tällä mm. varmistetaan, ettei järjestelmä ole koskaan ilman ylläpitäjän oikeuksia olevaa käyttäjää. Id on tunnus, johon käyttäjän harvoin tarvitsee samaistua, vaan käyttäjä voi tunnistaa muita käyttäjiä minkä tahansa $user-objektiin tallennetun tiedon avulla. Selkokielinen käyttäjänimi järjestelmässä on yksi tällainen tieto.

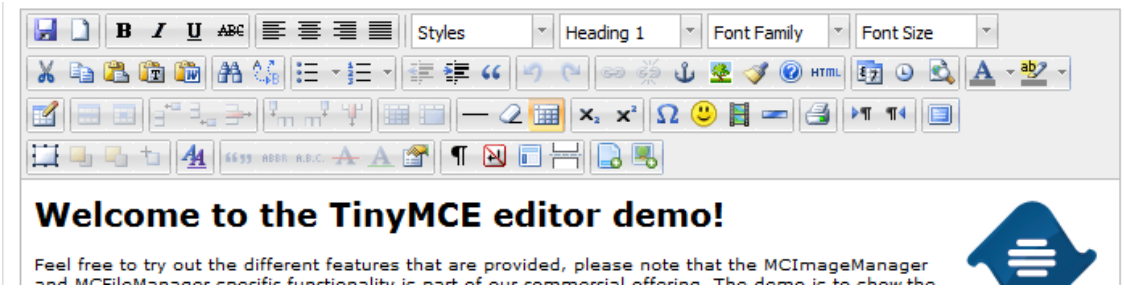
5.2 Tiedostojen liittäminen sivuihin

Intranetissä tärkeäksi osoittautui myös tiedostojen lataus ja hallinnointi. Intranetissä tähän käytettyjä ratkaisuja on kaksi. Menetelmiltään ratkaisut ovat suuressa suosiossa laajassa järjestelmien skaalassa. Mahdollisia ratkaisuja oli alunperin kolme ja haluaisin esitellä myös tämän lopullisesta järjestelmästä puuttuvan ratkaisun. Intranetissä käytössä olevat ratkaisut pohjautuvat vahvasti vallitseviin käytäntöihin internetissä ja Drupalissa. Tiedostonsiirron osuudesta määrityksissä oli suuria muutoksia vielä projektin loppuvaiheilla.

5.2.1 Lomakepohjaiset ratkaisut – WYMeditori ja Drupal-tiedostonsiirto

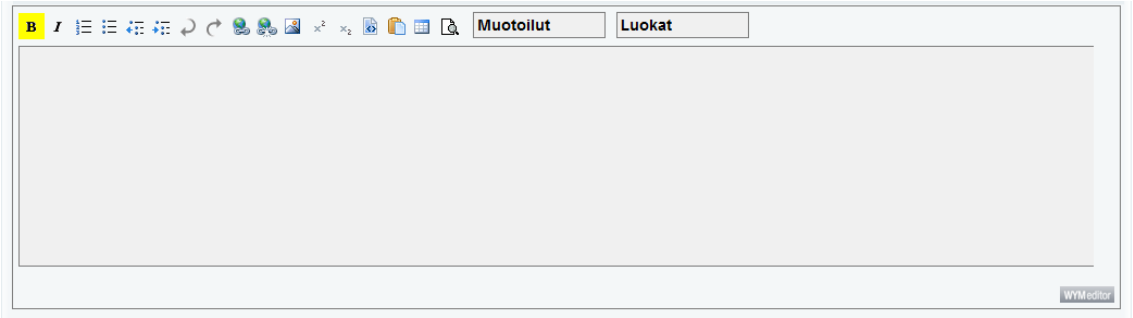
Tämän johdosta projektissa päädyttiin käyttämään mm. internetin sähköpostiohjelmista tuttua editoria, joka hoitaa tiedonsiirron liitetiedostoina. Näitä editoreita on lukuisia ja niitä on listattu mm. osoitteessa http://news.softpedia.com/news/The-Most-Popular-Free-WYSIWYG-Editors-85015.shtml. Editorilla voi muokata tekstiä ja käyttäjän luomaa sisältöä.

Pelkän tiedonsiirron ratkaisuna tälläinen WYSIWYG-editori on sisäisessä järjestelmässä paitsi turha, myös tarjoaa käyttäjälle suuren määrän uusia ominaisuuksia, jotka käyttäjien tulee opetella. Ratkaisuperiaatteena tämä luottaa että jokainen käyttäjä pystyy samaistumaan editorin toimintaperiaatteisiin. Kuvista 10. ja 11. saa idean tyypillisestä WYSIWYG-editorista ja sen ulkoasusta. Nykyinen ratkaisu muistuttaa suuresti suosittua TinyMCE-editoria (kuva 10.), mutta karsitummalla ulkoasulla (kuva 11.).



Kuva 10. TinyMCE-editori. (Tinymce.com 2011)

TinyMCE-editoria ei kuitenkaan käytetty tässä projektissa sillä se on korvattu Wysiwyg API-editorilla (kuva 11.). Editorina sen parhaat puolet muodostuvat sen ominaisuuksien runsaudesta. Pelkistettyjen verkkosivujen ylläpidon voisi esimerkiksi hoitaa yksinomaan kuvassa 10. esitetyllä editorilla.



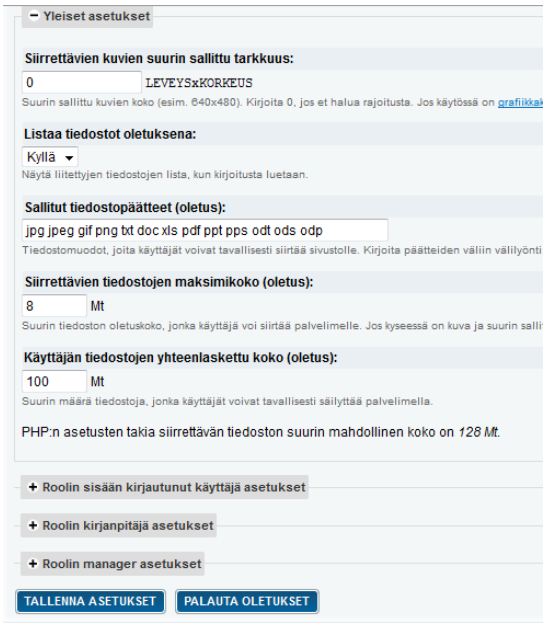
Kuva 11. Käytössä oleva WYMeditor.

Editorin haluttu ominaisuus oli yksinkertainen tiedonsiirtojärjestelmä. Tältäkin osin ratkaisu jää vaitonaiseksi. WYMeditori liittää tiedoston järjestelmässä eri sijaintiin kuin olisi toivottavaa. Editorin kautta lisätyt tiedostot jäävät järjestelmän kansiorakenteessa moduulin sisäiseen kansioon. Moduulin vaihtuessa, tai päivitettäessä moduulia suuri määrä tiedostoja voi kadota, ja vähintään ne tulee varmuuskopioida erilliseen sijaintiin. Lisäksi ratkaisun vaihtaminen toiseen tuo lisätyötä ylläpitäjälle. Lisää WYMeditorista osoitteesta http://www.wymeditor.org/. Kyseisen editorin kehitys on lopetettu, joten uusia päivityksiä tai korjauksia tähän ei tulla enää julkaisemaan yhteisön toimesta. Järjestelmäkehityksessä tämän tulisi myös olla painava kriteeri alun valinnassa, mutta tämä implementoitiin asiakkaan ja konsultin määritysten muuttumisen johdosta.

Drupalin oman tiedonsiirron käyttö on yksinkertainen vaihtoehto, ja ylläpitäjälle sopivampi vaihtoehto, sillä se tarjoaa mahdollisuuden muokata tiedonsiirron vaihtoehtoja ilman PHP-ohjelmointitaitojen opettelua. Ylläpitäjä voi määrittää yksittäisen tiedoston latauskoon tiettyyn palvelimen sallimaan rajaan sekä määrittelemään hyväksytyt tiedostomuodot tarpeiden mukaan. Tiedonsiirtoja Drupalissa voidaan määrittää myös käyttäjätason mukaan, jolloin hallitaan kyseessä olevan käyttäjätason käyttäjien mahdollisuuksia ladata tiedostoja järjestelmään. Drupalin yhteisösivut ovat täynnä tiedonsiirtoon suunnattuja

ratkaisuja, joten tyytyminen tähän yksinkertaiseen tiedonsiirtoon jonka kehitys on lopetettu on tässä tapauksessa lyhytnäköinen ratkaisu. (Drupal.org 2011h)

Kuvassa 12. näkyvät Drupalin hallintapaneelin asetukset tiedostonsiirroille käyttäen Drupalin sisäistä vaihtoehtoa. Ylläpitäjä voi näillä asetuksilla määrittää mitä käyttäjät saavat ladata ja jakaa, minkä kokoisia tiedostoja ja mikä on niiden sallittu yhteiskoko. Näitä määrityksiä voidaan yliajaa PHP:n asetuksilla palvelintasolla. Jokainen käyttäjätaso voidaan myös määritellä erikseen samassa järjestelmässä.



Kuva 12. Drupalin omat tiedostonsiirtoasetukset.

Editorin sekä Drupal järjestelmän sisäisen tiedostosiirron yhteisenä ongelmana on tiedostokoko sekä tiedostokokonaisuuksien hallinta. Molemmissa tapauksissa tiedostot liitetään johonkin tiettyyn sivuun. Tässä tapauksessa yhteen sivuun on vaikea kerryttää kokonaisuuksia, sillä kumpikaan vaihtoehto ei tue versiointia, korjauksia tai tiedostojen liittämistä toisiinsa millään tavalla. Edes tarkoilla nimeämisperiaatteilla on mahdotonta ylläpitää tiedostohierarkioita tai seurata tiedostojen käyttöä ja käyttäjiä. Molemmissa vaihtoehdoissa saa ladattua järjestelmään kerrallaan vain yhden tiedoston, minkä jälkeen käyttäjä joutuu läpikäymään moniosaisen prosessin, jotta käyttäjä voisi lisätä uuden tiedoston.

5.2.2 Kansiopohjainen ratkaisu - File Manager –moduuli TTjako

Omassa ratkaisussani ongelmaa on lähestytty toisella tapaa, joka on kokonaisuudessaan visuaalisempi kuin aikaisemmat vaihtoehdot. Näitä tiedonsiirtoratkaisuja kutsutaan myös yleisemmällä nimellä *file manager*. Ratkaisuja on myös vähintään yhtä paljon kuin niiden käyttäjiä, joten valta-osa näistä ratkaisuista käyttää jotakin tiedonsiirtoprotokollaa, joka standardoi käytännöt tiedonsiirrossa. Protokolla ei kuitenkaan ota kantaa siihen, miten asia esitetään käyttäjälle, joten käyttöliittymäsuunnittelu tuli tässäkin ratkaisussa esille.

Kuvassa 13. on käyttäjän näkymä tiedonsiirron TTjako-moduulista. Tästä rakentuu hierarkkinen kansiorakenne jossa voi olla useita alikansioita. Vasemmalla kuvassa näkyvät myös tallennuspaikat, joita voidaan tarvittaessa luoda useita Drupalin kansiorakenteen sisään. Tämän enempää rajoituksia moduuli ei luo tallennuksen sijainnille. Näkymän alapuolella ovat myös latauksen optiot, joissa voidaan käyttää raahausta tai selaamista halutusta sijainnista kuten työpöydältä ikkunoiden välillä Windows-ympäristössä.

Kuva 13. Monitasoinen tiedostonhallinta liitettäessä tiedostoa uuteen solmuun.

Kansiopohjainen ratkaisu korjaa aiempien ratkaisujen suurimpia puutteita. Kansiopohjalla käyttäjälle voidaan esitellä kansiorakenteita puumuotoisina, Windows-ympäristöstä tuttuina ratkaisuina, joka on hyvin omaksuttu muoto esittää hierarkioita sisältäviä tiedostokokonaisuuksia. Lisäksi moduuli hyödyntää

käyttöliittymistä ominaisuuden, jossa käyttäjä voi raahata tiedostoja työpöydältään suoraan järjestelmän kansioon.

TTJako-moduulissa on *case() –lauseke,* joka kattaa yleisimmät toiminnot kuten myös raahauksen kuvassa 14. Raahaus voi olla harhaanjohtava sanonta, sillä varsinaista raahausta ei koskaan tapahdu. Sen sijaan käyttämällä *$_POST* (PHP.net 2011b) ja *rawurldecode()* muuttujia (PHP.net 2011c), voidaan tutkia missä kansio oli ennen ja jälkeen käyttäjän toiminnon. Muuttuja saa itselleen kaksi parametriä jotka edustavat kansion sijainteja molemmissa tilanteissa. Drupalissa toimii myös painon periaate sivuissa ja navigaatioissa jota hyödynnetään myös tässä ratkaisussa. Paino tai *weight* määrittää moduulin kokonaisuuden sijainnin sivussa. Tätä painoarvoa voidaan käyttää järjestämään elementtejä sivussa vaikuttamalla missä järjestyksessä moduulit kutsutaan. (Drupal.org 2011i)

Kuvassa 14. on *case()*-lausekkeen tapaus tiedoston siirtämisestä kansioiden välillä. Siirron aikana lauseke huolehtii että sijainnit ovat olemassa, ja että käyttäjällä on oikeus molempiin siirrossa tarvittaviin kansioihin. Itse siirrossa tarkistetaan *if()*-lausekkeella siirron todella tapahtuvan, eli että kohdekansio on eri kuin tiedoston nykyinen kansio. Siirron onnistuessa lauseke lisää nykyiseen kansioon uuden sijainnin juuresta osoitteen, josta tämän seurauksena tiedoston sijainti muuttuu järjestelmässä. Siirrosta kirjataan myös sen suorittaja ja aikaleima viimeisimmästä siirrosta tiedostolle.

```
case "move":
        if (isset($_POST["param0"]) && isset($_POST["param1"])) {
          $source = $root_dir . trim(rawurldecode($_POST["param0"]));
          $dest = $root_dir . trim(rawurldecode($_POST["param1"]));
          if (is_dir($source) && ($ttjako_perm != ttjako_ADMIN)) {
            ttjako_json(array('status' => FALSE, 'data' => t('Pääsy kielletty')));
            exit();
            break;
          }
          if ($source != $dest) {
            if (!ereg('\.\.', $dest)) {
              $err_arr[] = array();
              $ret = ttjako_move($source, $dest, ($ttjako_perm == ttjako_USER) ? $user->uid : 1, $err_arr);
              ttjako_json(array('status' => $ret, 'data' => $err_arr));
            }
            else {
              ttjako_json(array('status' => FALSE, 'data' => t('Et voi siirtää tähän kansioon')));
            }
          }
          else {
            ttjako_json(array('status' => FALSE, 'data' => t('Kansio ei ole siirrettävissä')));
          }
        }
        else {
          ttjako_json(array('status' => FALSE, 'data' => t('Osoita kohde')));
        }
        exit();
        break;
```

Kuva 14. Kansion siirto moduulissa, $_post ja rawurldecode().

Revisioinnin avulla voidaan seurata tiedostojen muutoksia ja kuka niitä on viimeksi käsitellyt jokaisen tiedoston kohdalla. Kaikki moduulin kautta jaetut tiedostot jaetaan samaan juurikansioon, jota voi järjestelmän kautta selata halutessaan ja jo ladattuja tiedostoja voidaan selata uutta sivua tehdessä. Juuren alla voi olla useita muita kansioita, mikä on tämän vaihtoehdon parhaimpia puolia.

```
$root_dir = file_directory_path() . ttjako_get_root_path();
```

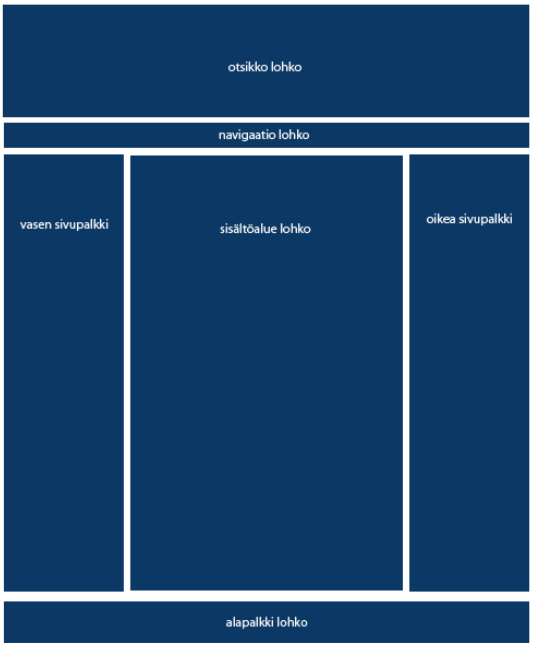
Kuva 15. Juurisijainnin arvon määritys (Api.drupal.org 2011c).

Kuvassa 15. on esitys juurisijainnin määrityksestä. Se tuli myös toteuttaa menetelmällä jossa arvoa voitiin käyttää vapaasti koko moduulin ajon ajan. Pitkällä aikavälillä tiedostojen kerrytys samaan tasoon tuottaisi lähes ylitsepääsemättömiä vaikeuksia rakenteen loogisessa hallinnassa.

6 DRUPAL-NÄKYMÄN HALLINTA

6.1 Lohkon muodostus

Drupalissa on useita tapoja muokata ja rajata sitä mikä on näkyvillä käyttäjäkohtaisesti. Tässä projektissa päädyin käyttämään pääasiallisesti kahta ratkaisua, jotka esitellään seuraavissa luvuissa. Kuvassa 16. on esitelty intranetin sisältölohkot ja niiden nimet.



Kuva 16. Drupal-intranetlohkot.

Hallintapaneelin avulla on myös mahdollista toteuttaa omia lohkoja, joiden sisältö voi olla mitä tahansa tekstistä ohjelmaa suorittavaan PHP-koodiin. Lohkoja on käytännössä kahdentyyppisiä. Toiset ovat nk. moduulilohkoja, eli moduuleja, jotka funktioillaan luovat lohkoja, esimerkiksi navigaation moduuli on tällainen. Nämä ovat moduuleja, joita yleensä esitetään useammalla sivuston osalla ja alasivuilla. Toisaalta tarvitaan myös vaihtuvalle sisällölle tarkoitettuja lohkoja, joita normaalisti ovat mm. sivun leipätekstit. Kaikkien lohkojen luomiseen käytetään Drupalille ominaisia koukkuja. (Drupal.org 2011j)

Intranetissä käytetyssä Drupal 6:ssa lohkon luomiseen tarvitaan moduuli ja delta-arvo. Delta-arvo on halutun lohkon nimi, jolla se voidaan jatkossa tunnistaa.

Kuvassa 17. nähdään Drupalin PHP-muotoinen koodi vasemman sivupalkin lohkon luonnista. Valmiilla komennolla $*block* voidaan järjestelmälle julistaa uusi lohko, ja antaa sille ominaisuuksia. Uudet lohkot ovat lähes poikkeuksetta uusia objekteja järjestelmässä. Lohkoilla on myös taulukoita joissa sen tiedot esitellään järjestelmälle. Järjestelmän kannalta tärkeitä tietoja ovat mm. lohkon delta-arvo, lohkon sijainti sivussa eli *region* johon se sijoittuu ja tuleeko kyseisen lohkon sisältö moduulista.

Region-arvolla määritetään moduulien sisältöjen mahdolliset sijainnit sivustolla. Nämä region-alueet ovat paikkoja, joihin Drupalissa voidaan sijoittaa sisältöä. (Drupal.org 2011k)

```
1  $block = new stdclass; // uusi tyhjä objekti
2  $module = 'system';
3  $delta = 0; // uuden lohkon nimi
4  //  hook_block()
5  // module_invoke()
6  $array = module_invoke($module, 'block', 'view', $delta);
7
8  // muunnetaan objektiksi
9  // block_list()
10 if (isset($array) && is_array($array)) {
11   foreach ($array as $k => $v) {
12     $block->$k = $v;
13   }
14 }
15
16 $block->module = $module;
17 $block->delta = $delta;
18 $block->region = 'vasensivupalkki';
19
20 echo theme('block',$block);
21
```

Kuva 17. Lohkon muodostava koodi Drupal 6:ssa.

hook_block()-funktioilla voidaan välittää dataa Drupalin coren eli ytimen ja moduulien välillä. Funktiolla identifioidaan lohko, minkä jälkeen siihen voidaan kohdistaa toimenpiteitä. (Drupal.org 2011j)

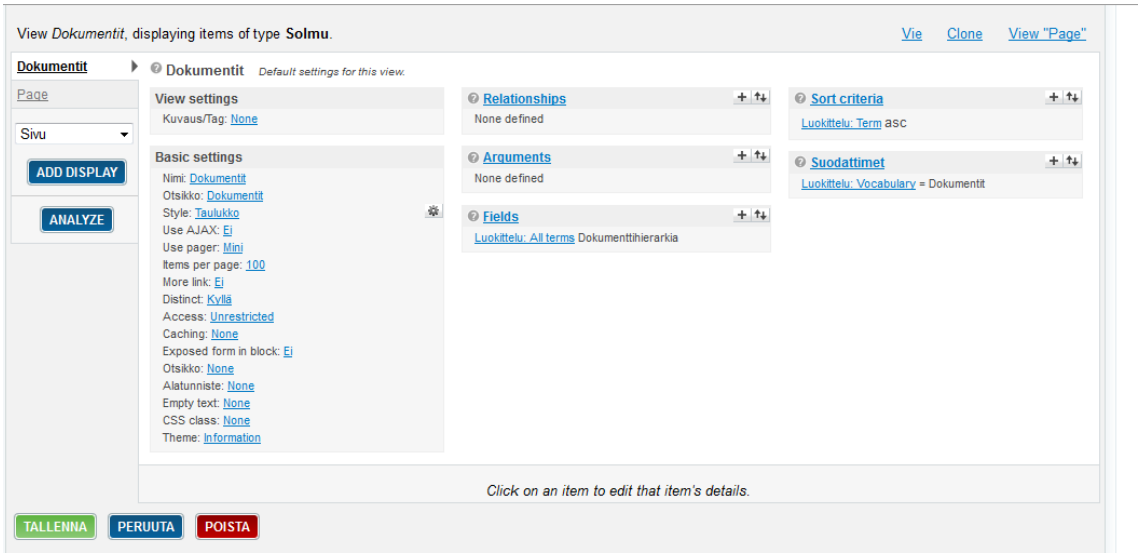
Tehtävänä on sijoittaa moduulien suoritteet lohkoihin näytettäväksi käyttäjälle. Yksi lohko voi sisältää useiden moduulien suoritteita. Hallitsemalla lohkojen

näkyvyyttä voidaan määritellä eri käyttäjille erilaisia näkymiä. Drupalin hallintapaneelin kautta voidaan määritellä eri käyttäjäroolien mukaan mitä lohkoja kullekin käyttäjälle tai roolille näytetään. Tämä tapa on vahvasti ylläpitäjän vastuulla hänen määrittäessä jokaiselle roolille lohkokohtaisia näkymiä. Käyttäjiä voidaan erotella myös yksittäisten sääntöjen mukaan jolloin tietyt lohkot eivät näy tietyille käyttäjille tai laajemmin käyttäjärooleille.

6.2 Views-moduuli

Views-moduuli on hyvin olennainen osa Drupalia, ja sillä voidaan luokitella, osioida ja järjestellä dataa käyttäjälle. Viewsin avulla minkä tahansa moduulin näkymää voidaan muokata uudestaan. Osa Drupalin liitännäisistä moduuleista myös vaatii tätä moduulia toimiakseen. Views toimii yhdessä CCK –moduulin kanssa luoden yhden Drupalin tehokkaimmista työkaluista kehittäjille. Jokaiselle sisältötyypille, kentälle tai lohkolle voidaan määrittää omanlaisensa näkymä näiden avulla.

Kuvassa 18. on haluttu luoda oma näkymänsä ryhmälle dokumentteja, joka havainnollistaisi myös dokumenttien hierarkiaa käyttäjälle. Kuvassa 18. näkyvät Views-moduulin tarjoamat vaihtoehdot kenttien järjestämiseen. Views-moduulia on helppo hallinnoida Drupalin hallintapaneelin kautta, josta kuvakaappaus on otettu.

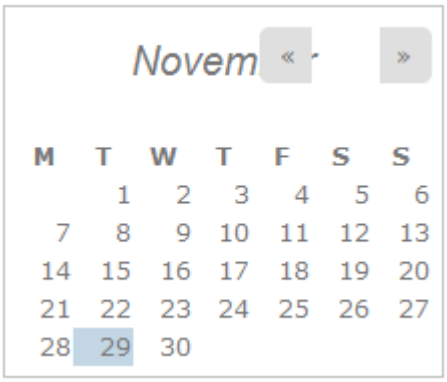


Kuva 18. Dokumentit-näkymän asetukset.

Koska TODO –moduulissa on oma sisältötyyppinsä, tarvitaan Views –moduulia luomaan haluttuja näkymiä tietokannan kyselyistä. Tärkeimpänä halutaan päättää näkymässä esitettävät kentät. Myös eri moduuleihin liittyviä kenttiä on mahdollista liittää toisiinsa näkymillä. Kenttien ulosantia voidaan muokata ja ulosannin järjestystä voidaan helposti hallita moduulin mukana tulevan hallintapaneelin avulla.

6.3 Kalenterinäkymä

Kalenterinäkymä intranetissä on toteutettu Views-moduulilla. Kalenterin tehtävänä on antaa käyttäjälle nopea näkymä hänelle suunnatuista tehtävistä. Tähän on mahdollista yhdistää myös muita merkintöjä, kuten käyttäjän omia muistutuksia. Kuvassa 19. kalenterin esitysmuoto järjestelmän etusivulla. Date-moduuli antaa järjestelmälle aikaleiman ja kalenteripäivämäärän, josta ne poimitaan Views-moduulin avulla etusivulle, muodostaen käyttäjälle tunnistettavan kalenterin.



Kuva 19. Kalenteri intranetin etusivulla.

Kalenterinäkymä voidaan toteuttaa luomalla tarvittavat kentät CCK-moduulin avulla. CCK-moduuli on sisällönluonti/työkalu-moduuli. Drupalin moduulin hallintapaneelin kautta luodaan yksittäisiä kenttiä valikoitua dataa varten, joihin voidaan jatkossa tallettaa käyttäjän syötteitä esitettäväksi muille käyttäjille. Näitä kenttiä voidaan siten yhdistää Views-moduulin avulla näkymäksi.

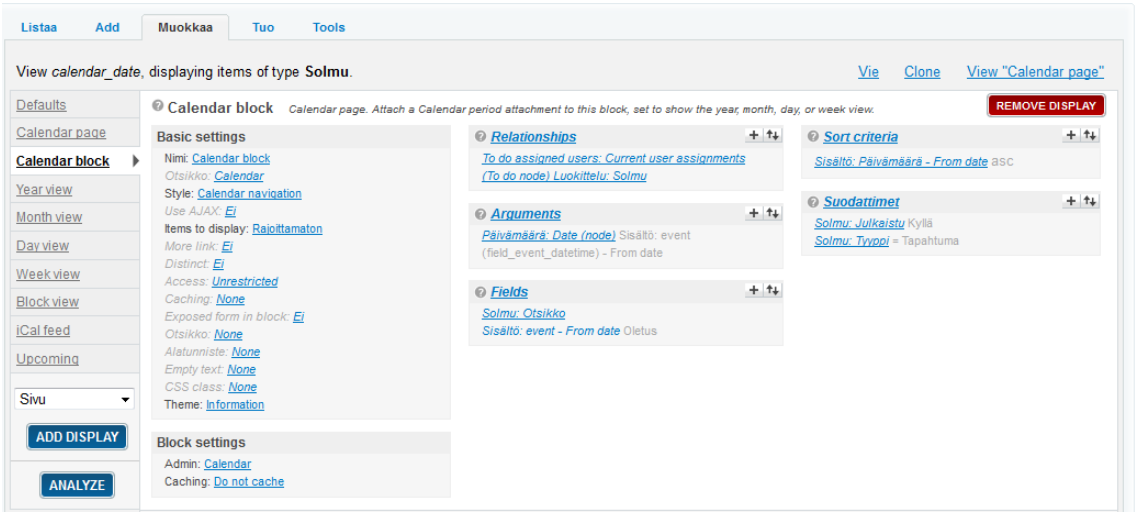
Kuvassa 20. näkyvät kalenterinäkymän asetukset. Vasemmalla kuvassa on perusasetukset. Perusasetuksiin kuuluu mm. näkymän nimi, sen tyyppi ja tyylit. Nämä perusasetukset tulevat Views-moduulin mukana. Kalenterinäkymän kenttiin voidaan hakea oikeat tiedot suoraan järjestelmän sisäisestä päivämäärästä.

Relaatioilla eli kohdassa *Relationships* saadaan kytkös TODO-moduulin kenttään kalenterinäkymään. Tämä relaatio näyttää kirjautuneen käyttäjän sen hetkisten tehtävien määräajan kalenterissa. Jos tehtävällä ei ole määräaikaa,

voidaan kenttä vaihtaa osoittamaan aloituspäivämäärää, tai se voidaan jättää tyhjäksi jolloin kalenterissa ei näy merkintää.

Näillä relaatioilla kalenterista saadaan käyttäjälle interaktiivinen. Kalenterinäkymän relaatio lähettää tässä tapauksessa tapahtuman TODO-moduulille, jossa pyyntö toteutetaan ilman Views-moduulin ajoa. Views-moduuli ohjaa käyttäjän linkin kautta TODO-moduulin antamaan tapahtuman vastaukseen oikealle sivulle intranetissä. Tarvitaan vain relaatio kalenterinäkymän ja TODO-moduulin välille jota hallitaan Views-moduulilla.

Liittämällä näkymään kenttiä voidaan siihen tehdä linkkejä, tai vaikkapa linkin kanssa olevia nk. työkalupalkkeja. Toiminnallisuus tulee kuitenkin kentän ja näkymän ulkopuolelta moduulilta johon kenttä on liitetty. Yksinkertaistetusti nämä kentät ovat toiminnaltaan PHP:n mukaisia lomakekenttiä, joilla on lähetys- ja palautustoimintoja.



Kuva 20. Kalenterinäkymän Views-moduulilla luodun näkymän asetukset.

Näkymä ei suorita järjestelmässä mitään tapahtumia, vaan napsautettavien tapahtumien kohdalla tulee tapahtumaan vastaavalla moduulilla olla toimenpide. Yleisin vastaus kalenterin napsauttamiseen intranetissä on antaa käyttäjän tarkastella näkymää koko näytöltä. Napsautus siirtää käyttäjän kyseisen TODO-moduulin tapahtuman sivuun, jossa voidaan tarkastella tehtävän yksityiskohtia.

7 KÄYTTÖLIITTYMÄ

Käyttöliittymän tarkoitus on tuoda käyttäjän ja koneen välille rajapinta, jossa käyttäjä ja kone voivat kommunikoida keskenään. Jakob Nielseniä on jo pitkään pidetty käytettävyysguruna käyttöliittymäsuunnittelun alalla. Nykyisellään on vaikea puhua käyttöliittymien käytettävyydestä mainitsematta Jakob Nielseniä. Nielsenin huomiot ja suositukset pätevät yhä huolimatta siitä, että käyttöliittymiä koskevien teosten julkaisuajankohdasta on kulunut runsaasti aikaa. Mainittakoon että teoksia on painettu useita painoksia vuosien kuluessa.

Nielsen pyrki tuomaan laajempaan huomioon tutkimustuloksia, jotka olivat jo tuolloin tiedossa, mutta huonosti noteerattuna käytännössä. Nykyään Nielsenin julkaisuja huomioidaan laajasti ja sen arvostus on noussut esittelevästä ohjeistavaan. Tekniikka on kehittynyt, ja nykyiset ratkaisut ovat jatkuvassa murroksessa. Selaimet ovat siirtyneet kauemmas 1990-luvulla hallinneista tekstipohjaisista, Mosaicista, ja Netscapesta (Nielsen 2000, 37). Nykyisin lähes kaikki selaimet tukevat yhteisiä standardeja (Web Devout 2011).

Tältä pohjalta haluan opinnäytteessäni huomioida Nielsenin oppeja, vaikka oman näkemykseni mukaan käyttäjistä on tullut taitavampia ja graafiset web-käyttöliittymät ovat arkipäiväistyneet. Käyttäjät odottavat myös käyttökokemuksiltaan enemmän kuin kaksikymmentä vuotta sitten. Korkeisiin odotuksiin on vastattava nykytekniikoiden voimin vaarantamatta liikaa uudelleenkäytettävyyttä, käytännöllisyyttä tai yhteensopivuutta selaimiin koodin tasolla. Intranet-järjestelmän käyttöliittymää suunniteltaessa tiedetään yleensä melko täsmällisesti käyttäjien tarpeet. Tässä tapauksessa käyttäjien ryhmä on määrältään pieni, lisäksi käyttökokemuksiltaan, taidoiltaan ja odotuksiltaan samankaltainen.

Nielsenin kannalta internetissä olevien sivustojen sisältö on suunnattu ulkoisille asiakkaille eikä sen tulisi orjallisesti noudattaa organisaation sisäistä rakennetta (Nielsen 2000, 15). Sisäiseen käyttöön tulevassa intranetissä tulee kuitenkin ajatella myös organisaation sisäistä rakennetta ja sen toimintatapoja. Toisaalta käyttäjien tottumuksen vuoksi näyttöpäätteellä visualisoitavien asioiden tulee

noudattaa käyttäjälle tuttua muotoa ja järjestystä. Koska projektin asiakkaalla on käytössä Windows-käyttöjärjestelmä, pyritään se huomioimaan toiminnoissa ja esimerkiksi käytetyissä kuvakkeissa. Käyttäjän näkökulmasta pienetkin vaihtelut toiminnoissa vaikuttavat merkittävästi sivuston käytön oppimiseen, ja sen mieleenpainuvuuteen. Asiakkaalla ei ole käytössä aikaisempaa intranetiä, eikä juurikaan kokemusta niistä, joten käyttö opitaan uutena, sen sijaan että pyrittäisiin muokkaamaan käyttäjän toimintaa poikkeamaan aiemmasta.

Pohdittavaksi jää esitelläänkö intranet käyttäjille sisäisenä vai ulkoisena työkaluna. Päätöksestä riippuen käyttäjien kokemukset samasta järjestelmästä voivat poiketa. Sisäisenä työkaluna käyttäjät rinnastaisivat järjestelmän kenties Windows- käyttöjärjestelmään, kun taas ulkoisena käyttöliittymä rinnastettaisiin kaikkiin käyttäjän internetissä kohtaamiin käyttöliittymiin. Jälkimmäisessä vertauskohtia saattaa olla liikaa eikä käyttäjä pysty paikantamaan haluttuja toimintoja järjestelmässä. Täten myös kokemus järjestelmästä on käyttäjälle pettymys ja sen käyttö jatkossa on huteraa ja tuloksetonta. Käyttäjälle tuttuja käyttöliittymiä on hyvä hyödyntää omassa suunnittelussa. Tämä on helpoin ja luotettavin tapa saada käyttäjät totuttautumaan uuteen järjestelmään. Parhaat käyttöliittymät pysyvät poissa käyttäjän tieltä, eivätkä ohjasta käyttäjää tarjoten tälle toimintoja joita käyttäjä ei välttämättä hakenut. (Thinkvitamin.com 2011)

Nielsen jakaa nettisivujen suunnittelun kahteen puoleen, graafiseen ja tekniseen puoleen. Nielsen keskittyy vahvasti teknisyyteen vaikkakin myöntää että visuaalisuudella on osansa sivustojen suunnittelussa. Toiminnallisuus on yhä tärkein osa käyttöliittymässä, mutta käyttäjillä on myös visuaalisia odotuksia enemmän kuin aiemmin. Käyttöliittymän osien sijainnilla ja niiden väliin jäävällä tyhjällä tilalla on suuri merkitys käytettävyyteen. Nielsenin mukaan käyttäjää kiinnostavan tiedon tulisi viedä mielellään 80 % näyttöalasta. Vielä nykyisinkin löytää helposti sivustoja joissa käyttäjän kannalta merkityksellisintä asiaa ei huomioida. Näissä tapauksissa sisällön osuus sivusta jää jopa alle 20 %. (Nielsen 2000, 18)

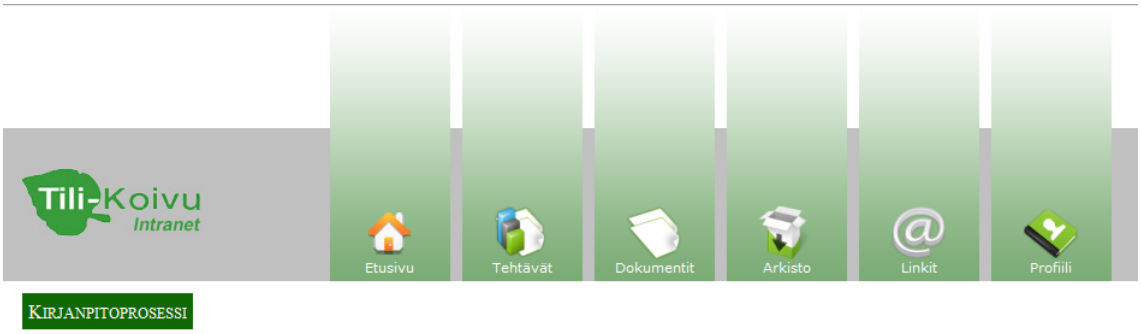
Web-käyttöliittymässä tulee kuitenkin olla myös pysyviä elementtejä vaihtuvan sisällön lisäksi. Peruselementtejä sivustolla katsotaan olevan neljä. Nämä ovat

kokooja (ts. header tai ylätunniste), sisältöalue (content), alatunniste (footer) ja navigaatiopalkki. Jos nämä elementit eivät ole helposti tunnistettavissa sivustolla, sivustoa pidetään yleisesti epäonnistuneena.

7.1 Ulkoasu

Ulkoasu on jaettu neljään toiminnalliseen osa-alueeseen. Ulkoasu noudattaa lukusuuntaa joka etenee ensisijaisesti vasemmalta oikealle ja ylhäältä alas. Tämä on erityisesti länsimaisille sivustoille tai länsimaisille käyttäjille suunnatulla sivustolla vakiintunut asettelu lukusuunnaksi. Sivun yläosassa painottuu navigaatio ja keskellä sisältö. Käyttöliittymässä pyritään myös hyödyntämään mahdollisimman paljon näytön poikkilinjaisesta tilasta. Suositeltavaa on, että sisällölle jätetään suhteessa suurempi osa kokonaisuudesta kuin muille käyttöliittymän osille (Nielsen 2000, 18). Sivun elementtien sijoittelussa on myös pyritty löytämään loogisia kokonaisuuksia ja sijoittamaan käyttäjän kannalta samanarvoiset toiminnot lähelle toisiaan. Tässä pyritään hyödyntämään läheisyyden lakia, jonka mukaan kaksi toisiaan lähellä olevaa visuaalista ärsykettä mielletään yhteenkuuluviksi (Ratol.fi 2011).

Kuvassa 21. on havainnollistettu intranetin päänavigaation ulkoasua. Navigaatiossa esiintyvät kaikki intranetin tärkeimmät toimintokokonaisuudet käyttäjän näkökulmasta. Nämä valikon linkit toimivat myös pikalinkkien tapaan tarjoten käyttäjälle nopean pääsyn intranetin tärkeisiin osiin. Tämä valikko on aina käyttäjän saatavilla riippumatta tämän sijainnista intranetissä. Lisäksi logoalueen alapuolelle kerrytetään laatukäsikirjan osia joita klikkaamalla avautuu toiminnallinen ohjeistus.



Kuva 21. Päänavigaation ulkoasu.

Ulkoasussa haluttiin minimoida verkkosivujen suurimpia kompastuskiviä, kuten liiallista animointia ja monimutkaisia sivurakenteita. Väreinä käytettiin runsaasti valkoista ja tekstit ovat linkkejä lukuun ottamatta mustia. Asiakkaan toiveesta yrityksen edustusväri vihreä esiintyy intranetissä logoissa, linkeissä ja taustojen korostusvärinä. Lisää visuaalisuutta saatiin lisäämällä intranetin linkkeihin toimintoihin liittyviä kuvakkeita.

Ulkoiset verkkosivut ovat kokeneet suuria muutoksia viime vuosina, ja nyt muutokset ovat siirtymässä myös sisäisiin järjestelmiin. Vaikka suurin osa tämän intranetin onnistumisen mittareista ei riipu ulkonäöstä, käytännöllisyys ja oikea työnkulun mallintaminen ovat erittäin tärkeitä kriteerejä.

7.2 HTML- ja CSS-koosto

Ennen kuin ulkoasu saatiin toivotun näköiseksi, tuli siitä tehdä koosto. Koostossa suunniteltu sivuston ulkoasu muokataan valitulla ohjelmointikielellä olevaksi sivuston rakenteeksi, jota selain voi lukea ja tulkita. Koosto on siis HTML/PHP-rakenne ja joukko CSS-sääntöjä selaimelle. Näitä kokonaisuuksia eri selaimet tulkitsevat eri tavoin. Kaikki selaimet eivät tulkitse sääntöjä samalla tavalla, mistä johtuen sivustolla voi esiintyä poikkeavuuksia eri selaimissa katseltuina. Sääntöjen käyttöönottoon ja levinneisyyteen kehittäjien keskuudessa vaikuttaa vahvasti suosituimpien selainten kanta kyseiseen sääntöön (Quirksmode.org 2011).

Jokaisen selaimen julkaisija päättää mitä sääntöjä selainten versioiden tulisi noudattaa. Hyvän ohjelmointitavan mukaisesti tämä tulee ottaa huomioon esim.

siten, että eri selaimille kirjoitetaan sääntöjä sellaisella tavalla jota selain osaa tulkita. Tällöin ulkoasu on käyttäjälle sama kaikilla selaimilla. Valmis koosto tuotiin FTP -yhteyden kautta palvelimelle Drupalin kansiorakenteeseen.

Drupal käyttää koostoa sellaisenaan teeman ulkoasuna. Järjestelmä tarvitsee suuren joukon sääntöjä ulkoasulle. Käytetyn säännön puuttuessa koostosta aiheuttaa teemassa poikkeavuuksia tai puuttuvia osia, joilla ei ole oikeanlaista esitystapaa teemassa. Näitä poikkeavuuksia on tärkeää estää eikä niitä tulisi esiintyä valmiissa koostossa. Huolellinen suunnittelu auttaa välttämään näitä tilanteita lopullisessa järjestelmässä. Drupal järjestelmänä suosii myös tiettyjä esitysmuotoja, jolloin nämä tulee ottaa huomioon suunnittelussa.

Keino yliajaa Drupalin tyylitiedostoja on korvata sapluunassa oleva ylätunniste PHP-funktiolla. Tässä tapauksessa käytetään merkkijonon korvaavaa str_*replace()*-funktiota. Tämän avulla sapluunasta korvataan Drupalin oma ylätunniste kirjoittamalla funktioon HTML-muotoinen tyylitiedoston merkintä, jolla uusi tyylitiedosto liitetään sapluunaan. Tähän vaihtoehtoon voidaan sisällyttää myös alkuperäinen ylätunniste. Toinen keino on yliajaa Drupalin tuonti-funktio sapluunasta, jolloin teeman sapluunaan lisätään uusi PHP-funktio. Tällä funktiolla yliajetaan suoraan Drupalin tuonnin funktio joka tuo teemaan Drupalin omat tyylitiedostot. (Drupal.org 2011l)

7.3 HTML-kooston sovitus Drupal-teemaksi

HTML-kooston sovitus Drupalille tapahtuu lyhyesti muuttamalla tiedostot .PHP-tiedostomuotoon. Tämä sen sijaan ei vaadi itse koodin muokkausta, tiedostopäätteen muutos riittää tiedoksi siitä kuinka tämä kootaan selaimelle näytettäväksi. PHP-tiedoston ei tarvitse sisältää riviäkään PHP-koodia jotta se koostettaisiin selaimessa PHP:n tapaan. (Handyphp.com 2011)

Teema kertoo mihin data sisällytetään ja milloin, css päättää sen esitysmuodon, ja PHP:lla suoritetaan kyselyjä näiden ohjeiden mukaan. Asetelma on erittäin tarpeellinen kehittäjän näkökulmasta, sillä se takaa joustavuuden ja mahdollisuuden tehokkaasti eriyttää käyttäjän muokkaukset järjestelmän kehittäjän muokkauksista.

Drupal tarvitsee joitakin tietoja, jotta se voi tunnistaa kansioiden sisällön teemaksi. Ensimmäiseksi tiedostojen tulee sijaita Drupalin polussa *sites/all/themes.* Tämä on tarpeellista, koska lisätyt teemat halutaan pitää erillään Drupalin coresta, jolloin sen ylläpito helpottuu. Coren päivityksen ei haluta aiheuttavan lisättyyn sisältöön epätoivottuja muutoksia ja päinvastoin. Lisäksi teema on näin kaikkien Drupalissa olevien sivujen käytettävissä. (VanDyk & Westgate 2007, 6-7.)

Drupal toimii *template*-pohjaisena, ja tiedostoihin tulee tehdä myös tätä pohjamallia noudattavat muutokset. Käytännössä malli on sama kaikilla sivuilla, ja näihin malleihin tehdään tyhjät lohkot jotka täytetään tietokannasta haetuilla tiedoilla. (Handyphp.com 2011)

Käytössä oleva Drupal-versio tarvitsee toimiakseen myös .info tiedoston, joka kertoo Drupalille teemasta. Tämän tulee sijaita samassa sijainnissa muiden teemaan liittyvien tiedostojen kanssa, eli muiden teemojen kansioiden, ja kehittäjän lisäämien tiedostojen ja kansioiden kanssa erillään Drupalin perusasennuksesta.

Kuvassa 22. on esimerkki teeman .info tiedostosta ja sen sisällöstä. Teemasta löytyvät sen perustiedot kuten nimi, versio, kuvaus ja millaisella

ohjelmistopohjalla teema on toteutettu. Näitäkin olennaisempia ovat tiedot teeman sisällöstä kuten mitä alueita se sisältää ja millaisia tyylitiedostoja teema käyttää.

```
1  name = TiliKoivu
2  description = TiliKoivun oma teema
3  version = VERSION
4  core = 6.x
5  engine = phptemplate
6  stylesheets[all][] = style.css
7  stylesheets[all][] = menutree.css
8
9  regions[left] = Left sidebar
10 regions[right] = Right sidebar
11 regions[header] = Header
12 regions[content] = Content
13 regions[navigation] = Navigation
14 regions[footer] = Footer
15 version = "6.x-3.1"
16 core = "6.x"
17 project = "intra"
18 datestamp = "1299698767"
19
```

Kuva 22. TiliKoivu .info tiedosto.

Tiedosto on syntaksiltaan kuin .INI-tiedosto ja Drupal saa tästä teemaan tarvittavat konfiguraatiot. Jokainen rivi sisältää arvoparin joita Drupal hyödyntää järjestelmän ylläpidolle esitettäviin tietoihin. Nämä tiedot vastaavat luettavaa kieltä, eikä noudata varsinaisesti mitään ohjelmointikieltä. (Drupal.org 2011m)

8 JOHTOPÄÄTÖKSET

Projektin ratkaisut löytyivät monen kokeilukierroksen tuloksena. Varsinaisen projektiryhmän ja projektivedon puuttuessa keskustelut näistä ratkaisuista käytiin lähinnä sähköpostin ja puhelimen välityksellä. Silti kommunikointi oli hidasta eikä säännöllisiä tapaamisia ollut, mikä ei ole osana ketterien työmenetelmien kuvausta. Lopullisia ratkaisuja tullaan hakemaan vielä tämän opinnäytteen valmistuttua.

Päätöksenteko oli vaikeaa ja lopullisia määrityksiä ei päätetty projektin jo edetessä. Tämä viittaisi siihen, etteivät projektissa mukana olleet osapuolet löytäneet yhteistä säveltä. Kokonaisuudessaan projekti oli ensimmäinen laatuaan kaikille osapuolille, joten myös työtapoja haettiin osapuolien kesken.

Projektin edetessä myös varsinaisen vastuujaon puuttuminen kävi ilmeiseksi. Asiakkaan oli vaikea tuoda esiin omia vaatimuksiaan oikean kuuntelijan puuttuessa. Projektin muuttuessa myös tämän opinnäytteen painotukset muuttuivat. Kevään edetessä avoimella kaavalla toimiva projektiryhmä muutti määrityksiä näkemiensä tulosten perusteella monia kertoja. Lopullinen järjestelmä osoittautui kompromissien kirjoksi.

Alkuperäiset määritykset ja järjestelmässä olevat moduulit ja laajennokset eivät vastaa toisiaan. Moduulit ovat kuitenkin toimivia ja sopivat käyttötarkoitukseensa. Määritykset vaihtelivat ajan kuluessa laajasta kokonaisuudesta pieniin yksittäisiin toimintoihin. Päätös yhdessä määrityksessä pysymisestä tuli kenties liian myöhään, mikä vaikutti järjestelmän yhtenäisyyteen käyttäjän kannalta.

Opiskelijana tunsin asiantuntemukseni jäävän käyttämättä ja tarkoitukseni projektissa ”hyytyi”. Koin kuitenkin tämän opinnäytteen keinona esitellä myös omat, intranetissä hyödyntämättömät ratkaisuni. Nämä ratkaisut ovat kuukausien työn tuloksena syntyneitä, olemassa olevien moduulien inspiroimia, räätälöityjä moduuleja ja näkymiä, joilla koota intranetin olemus Drupal-järjestelmässä.

Drupalin kanssa toimiminen vaati sopeutumisen uuteen tapaan työskennellä, jonka opettelu vaati alussa paljon. Viikkojen kuluessa tulosten näkyminen antoi kuitenkin uutta potkua jatkaa projektissa sekä itseluottamusta omiin taitoihinsa. Opiskelijalle on tärkeää myös löytää kykyjä ja asioita joissa on hyvä, ja joissa voi kehittyä. Alaan kuuluu myös lukuisia muita osa-alueita kuin puhdas koodaaminen. Vaikka Drupalin parissa PHP-ohjelmointitaitoni kehittyi huomattavasti, en silti kutsuisi itseäni ”koodariksi”.

Suunnittelun osalla koen omaavani huomattavasti enemmän lahjoja, johtuen ehkä aikaisemmista kokemuksistani suunnitellessani web-sivustoja. Järjestelmän suunnittelu on havainnollistavaa, vaatii kykyä erkaantua toteuttajan roolista ja samalla pitää löytää sopiva ratkaisu toteutuksen ja määritysten välille. Voisin puhua konseptista jota haluaisin kutsua vaikkapa sisäiseksi organisoinniksi. Tätä kykyä tarvitaan nimenomaan pienissä projekteissa, jossa yksi tekijä joutuu omaksumaan useita rooleja. Ollakseen hyvä tiimin jäsen tulee projektiin tuoda uusia näkökulmia ja huomioita.

Omalta osaltani minun on kehitettävä tätä taitoa tuoda näkökulmani esiin etsittäessä oikeaa ratkaisua. Ei voi pitää itsestään selvänä omaa rooliansa asiantuntijana, vaan se on todistettava useaan otteeseen projektin aikana, oli se kuinka uuvuttavaa tahansa. Mielestäni tämä ei kuitenkaan kuulu minkään projektin henkeen, ja on lähinnä uuvuttavaa vieden energiaa pois projektilta itseltään. Oikea ratkaisu harvoin tulee yhden ihmisen ajatuksista, vaan siihen vaaditaan monen ihmisen osanotto.

Projektin ja samalla tämän opinnäytteen päätöksessä voin nähdä tämänkaltaiset projektit eri silmin. Lopullisena tuloksena Drupalista on muotoutunut intranet, jossa on käyttäjilleen yksilöllisiä työkaluja joilla helpottaa päivittäistä työntekoa. Seuraavassa projektissa tulen käyttämään erilaisia ratkaisuja niin kommunikointiin kuin järjestelmän ratkaisuihin.

LÄHTEET

Analogik.org 2011. History of Joomla!. Viitattu 20.4.2011 http://analogik.org/HistoryofJoomla.html.

Api.drupal.org 2011a. DB_query. Viitattu 1.10.2011 http://api.drupal.org/api/drupal/includes--database--database.inc/function/db_query/7.

Api.drupal.org 2011b. $user. Viitattu 27.8.2011 http://api.drupal.org/api/drupal/developer--globals.php/global/user/6.

Api.drupal.org 2011c. file_directory_path. Viitattu 27.8.2011 http://api.drupal.org/api/drupal/includes--file.inc/function/file_directory_path/6.

Api.drupal.org 2011d. Documentation. Viitattu 31.8.2011 http://api.drupal.org/api/examples/node_example--node_example.module/function/node_example_info/6.

Daniweb.com 2011. Supressing printing of no value. Viitattu 29.8.2011 http://www.daniweb.com/web-development/php/threads/154264.

Drupal.org 2011a. Cases. Viitattu 25.4.2011 http://drupal.org/cases.

Drupal.org 2011b. History. Viitattu 25.4.2011 http://drupal.org/about/history.

Drupal.org 2011c. Overview. Viitattu 8.5.2011 http://drupal.org/getting-started/before/overview.

Drupal.org 2011d. General Concepts. Viitattu 8.5.2011 http://drupal.org/node/19828.

Drupal.org 2011e. Drupal abstraction layer. Viitattu 26.5.2011 http://api.drupal.org/api/drupal/includes--database.inc/group/database/6.

Drupal.org 2011f. Working with content types and fields(Drupal 6 and earlier). Viitattu 31.8.2011 http://drupal.org/documentation/modules/cck.

Drupal.org 2011g. Hooks. Viitattu 18.8.2011 http://api.drupal.org/api/drupal/includes--module.inc/group/hooks/6.

Drupal.org 2011h. Search Results. Viitattu 17.8.2011 http://drupal.org/search/apachesolr_multisitesearch/file%20transfer?filters=tid%3A62%20drupal_core%3A87%20bs_project_sandbox%3A0%20%20ss_meta_type%3Amodule&text=file%20transfer.

Drupal.org 2011i. How to update a module's weight. Viitattu 1.10.2011 http://drupal.org/node/110238.

Drupal.org 2011j. hook_block. Viitattu 18.8.2011 http://api.drupal.org/api/drupal/developer--hooks--core.php/function/hook_block/6.

Drupal.org 2011k. Regions in PHPTemplate. Viitattu 1.10.2011 http://drupal.org/node/29139.

Drupal.org 2011l. Overriding drupal.css; two approaches. Viitattu 17.8.2011 http://drupal.org/node/23217.

Drupal.org 2011m. Structure of the .info file. Viitattu 19.5.2011 http://drupal.org/node/171205.

Handyphp.com 2011. How do I convert my HTML website to PHP? Viitattu 21.8.2011 http://www.handyphp.com/index.php/PHP-Resources/Handy-PHP-Tutorials/converting-html-to-php-basic.html.

Joomla.org 2011b. Showcase. Viitattu 20.4.2011 http://community.joomla.org/showcase/.

Joomla.org 2011a. What is Joomla?. Viitattu 20.4.2011 http://www.joomla.org/about-joomla.html.

Mostrey Wim. The second most important Drupal user. Viitattu 1.10.2011 http://mostrey.be/second-most-important-drupal-user.

Nielsen Jakob, 2000. WWW suunnittelu. Designing Web Usability. Suom. Haanpää, T. Jyväskylä: Gummerus.

Packtpub.com 2011. Creating our first module using Drupal 6. Viitattu 26.5.2011 http://www.packtpub.com/article/creating-our-first-module-using-drupal6-part1.

Phpeveryday.com 2011. Drupal Module: Writing .info files. Viitattu 20.8.2011 http://www.phpeveryday.com/articles/Drupal-Module-Writing-info-files-P1034.html.

PHP.net 2011a. mktime(). Viitattu 1.9.2011 http://php.net/manual/en/function.mktime.php.

PHP.net 2011b. PHP Manual. Viitattu 26.5.2011 http://fi2.php.net/manual/en/reserved.variables.post.php.

PHP.net 2011c. PHP Manual. Viitattu 26.5.2011 http://php.net/manual/en/function.rawurldecode.php.

Quirksmode.org 2011. Combatibility tables. Viitattu 29.8.2011 http://www.quirksmode.org/compatibility.html.

Ratol.fi 2011. Viitattu 2.4.2011 http://www.ratol.fi/opensource/klus/ihminen_hahmolait.html.

Thinkvitamin.com 2011. 10 User Interface Design Fundamentals. Viitattu 19.5.2011 http://thinkvitamin.com/design/10-user-interface-design-fundamentals.

Tinymce.com 2011. Full featured example. Viitattu 23.11.2011 http://www.tinymce.com/tryit/full.php.

VanDyk J. & Westgate M. 2007. Pro Drupal Development. New York: Springer-Verlag.

Wordpress.org 2011b. Showcase. Viitattu 20.4.2011 http://wordpress.org/showcase/archives/.

Water & Stone. 2010. Open source CMS market share report. Viitattu 8.4.2011 http://www.waterandstone.com/sites/default/files/2010%20OSCMS%20Report.pdf.

Wordpress.org 2011a. About Wordpress. Viitattu 20.4.2011 http://wordpress.org/about/.

Web Devout 2011. David Hammond. Creative Commons Attribution Share-Alike License. Viitattu 2.4.2011 http://www.webdevout.net/browser-support.

W3Counter 2011. Awio Web Services LLC. Viitattu 2.4.2011 http://www.w3counter.com/globalstats.php?year=2011&month=2.