Heta Katajisto

INTERNET-SIVUSTO DRUPALILLA

Tietojenkäsittelyn koulutusohjelma

2015

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

INTERNET-SIVUSTO DRUPALILLA

Katajisto, Heta
Satakunnan ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
marraskuu 2015
Ohjaaja: Grönholm, Jukka
Sivumäärä: 40

Asiasanat: Drupal, Drupal 7, sisällönhallintajärjestelmä, internet-sivusto

---

Tässä opinnäytetyössä esiteltiin Drupalia sisällönhallintajärjestelmänä, eli sitä miten Drupal tallentaa sisältöjä, miten sisältöjä käsitellään Drupalin avulla ja minkälaista sisältöä Drupalilla voi käsitellä ja hallita. Työssä esiteltiin myös miten Drupal-sivusto rakentuu ja mistä kaikista palasista se koostuu, eli kerrottiin solmuista, kentistä, luokittelusta, moduuleista, valikoista, lohkoista, käyttäjistä ja teemoista. Opinnäytetyössä esiteltiin myös Drupalin ytimeen kuulumaton Näkymät-moduuli, sillä sen todettiin olevan suuri osa monia Drupalilla tehtyjä sivustoja ja sitä myös käytettiin paljon opinnäytetyön projektissa.

Drupalin esittelyn jälkeen työssä kerrottiin projektista, jossa eräs internet-sivusto toteutettiin Drupalia käyttäen. Sivusto toteutettiin teatterille, ja sivustolle tuli näkyviin kyseisen teatterin, menneet, nykyiset ja tiedossa olevat tulevat esitykset. Sivustolle toteutettiin myös kalenteri, jossa näkyvät kaikki tiedossa olevat tulevat näytökset. Näitä varten sivustolle lisättiin Drupalin sisältötyyppejä, esitys ja esitysajankohta, joihin määriteltiin kyseisten sisältöjen tarvitsemia kenttiä.

Kalenteri toteutettiin niin, että kalenterista pystyy valitsemaan haluamansa esityksen, jonka jälkeen avautuu lomake, jota käyttäen voi varata liput kyseiseen esitykseen. Kalenterin toiminnallisuus toteutettiin Drupalin valmiita moduuleita ja JavaScriptiä käyttäen. Kalenteria varten tehtiin myös oma kustomoitu moduuli vain tätä sivustoa varten.

Sivustolle tehtiin myös sivuja, jotta tehtyjä varauksia pystyy tarkastelemaan sivustolta. Nämä sivut toteutettiin niin, että pystyy helposti katsomaan yhden esitysajankohdan varauksia kerrallaan. Sivustolle luotiin käyttäjärooli sisällöntuottamiseen, jotta projektin toteutuksen jälkeen teatterin väki pystyy itse lisäämään sivulle esityksiä ja esitysajankohtia. Sivustolle tehtiin myös käyttäjärooli pelkästään varausten tarkastelua varten, tämän käyttäjäroolin käyttäjillä ei ole oikeutta muuttaa, lisätä eikä poistaa mitään. Varausten tarkastelu –sivut määriteltiin näkymään vain näille kahdelle käyttäjäroolin omaaville käyttäjille.

A DRUPAL WEB SITE

Katajisto, Heta
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in Business Information Technology
November 2015
Supervisor: Grönholm, Jukka
Number of pages: 40

Keywords: Drupal, Drupal 7, content management system, web site

---

This thesis demonstrates Drupal as a content management system, namely how Drupal saves content, how the content is edited with Drupal and what different content types one can edit and manage with it. The thesis also shows how a Drupal web page is built and what kind of building blocks are used to make it work, such as nodes, fields, taxonomy, modules, menus, blocks, users and themes. An unofficial "Views" module, not integrated to the Drupal core is also demonstrated because it has proved to be very useful and it's used in several Drupal sites, including the project in the thesis.

After the Drupal showcase, the thesis talks about a project in which a web page was made using Drupal. The site was made for a theatre and the primary content was past, current and upcoming performances. A calendar that displays all known and upcoming performances was also implemented. For the calendar, content types were made for Performance and Date, and fields that the types expected were defined.

The calendar was made so that upon selecting a performance, a form opens and you can fill it to book the tickets to the performance. The functionality was made by using Drupal's pre-existing modules and JavaScript. For the calendar, a special customized module was made exclusively for this page.

Certain sub-pages were also made so you could inspect your bookings. The sub-pages were made so that you could easily view multiple bookings of the same date range. User roles for content creators were added so the theatre staff themselves can add performances and information to the site. A user role for viewing the bookings only was made as well, and these users have no permissions to modify, add or delete anything. "Booking inspection" pages were made visible to these two roles only.

SISÄLLYS

1 JOHDANTO 5
2 TOIMEKSIANTAJAN JA PROJEKTIN ESITTELY 5
3 PROJEKTIN TAVOITTEET JA LÄHTÖKOHDAT 6
4 DRUPALISTA 6
4.1 Drupalista yleisesti 6
4.2 Sisällön tallennus Drupal-sivustolle 8
4.2.1 Solmut 9
4.2.2 Sisältötyypit ja kentät 9
4.2.3 Luokittelu 11
4.3 Moduulit 11
4.4 Lohkot ja valikot 12
4.4.1 Lohkot 12
4.4.2 Valikot 14
4.5 Käyttäjät ja käyttöoikeudet 14
4.6 Teemat 15
4.7 Näkymät 16
5 SIVUSTON TOTEUTTAMINEN 18
5.1 Drupalin asentaminen 18
5.2 Ulkoasumääritykset 18
5.3 Teeman valinta ja määrittely 20
5.4 Moduulien asennus ja valinta 22
5.5 Sisältötyypin Esitys määrittely 22
5.6 Esitysten näyttäminen sivustolla 24
5.6.1 Tulevien ja nykyisten esitysten näkyminen 28
5.7 Esitysajankohtien ja kalenterin määritys 29
5.8 Varauslomakkeen teko 33
5.8.1 Varausten määrän rajoitus 34
5.9 Varausten tarkastelu 35
6 SIVUSTON JULKAISU 36
6.1 Käyttäjätunnukset 36
7 LOPUKSI 37
LÄHTEET 39

1 JOHDANTO

Tässä opinnäytetyössä käyn läpi sivuston toteuttamista Drupalilla, käytännössä Drupalin versiolla 7. Opinnäytetyössä käyn läpi myös projektia, joka ei ole aivan perinteinen internet-sivusto, jossa olisi etusivu, pääsivuja ja alasivuja. Tämän takia työstä saa hyvän käsityksen siitä, kuinka monipuolisesti Drupalia voi internet-sivuja tehdessä käyttää jo kohtuullisen hyvällä Drupal-osaamisella.

Työssä on aluksi käytä läpi Drupalin toimintaa, miten tiedot tallentuvat Drupalilla, miten niitä voi hyödyntää ja muita Drupalin perustoiminnallisuuksia. Tämän jälkeen selvitän miten Drupalia käyttäen toteutin internet-sivuston, jossa on toimiva ja helppokäyttöinen sisällönhallintajärjestelmä. Projektin läpikäynnissä tulee ilmi, kuinka Drupalilla sivusto tehdään niin, että sisällönhallinta on sivuston toteuttamisen jälkeen helppoa ja toimivaa.

2 TOIMEKSIANTAJAN JA PROJEKTIN ESITTELY

Projektin toimeksiantajana oli porilainen teatteri Cont'akti, joka palasi muutaman vuoden tauon jälkeen takaisin toimintaan. Jo vuodesta 1988 toiminut teatteri halusi palaamisen kunniaksi aivan uudet internet-sivut, jossa näkyisi käynnissä olevat esitykset ja joista löytyisi myös Cont'aktin pitkän historian aikana tehdyt vanhatkin esitykset. Projektissa oli mukana itseni lisäksi vain toinen henkilö, joka teki ulkoasusuunnitelman ja toimi myös teatterin puolelta asiantuntijana, eli kertoi esimerkiksi mitä tietoja teatteriesityksestä olisi hyvä löytyä. Hän toimi myös ”asiakkuusyhteyshenkilönä”, eli esitteli sivuston muulle teatterin väelle ja kertoi heidän tarpeitaan/toiveitaan minulle.

3 PROJEKTIN TAVOITTEET JA LÄHTÖKOHDAT

Projektin tavoitteena oli saada teatterille aivan uusi internet-sivusto, jossa olisi teatterin perustiedot, käynnissä olevat esitykset ja niiden esittely ja jo menneet esitykset. Sivustolle toivottiin myös jonkinlaista kalenteria, josta näkyisi käynnissä olevien esitysten esitysajankohdat ja mahdollisesti jonkinlainen varausjärjestelmä, jonka avulla voisi varata sivuilta lipun esitykseen.

Toimeksiantajan puolelta ei sivustosta ollut mitään erityisvaatimusta tai –pyyntöä, miten sivusto rakennetaan, kunhan se olisi jatkossa helppo teatteriväen itse päivittää. Koska minulla on jo vahva kokemus sisällönhallintajärjestelmä Drupalista, oli Drupal minulle helpoin ja luontevin valinta. Myöskään sivuston vaatimukset eivät asettaneet estoa Drupalin käytölle, sillä tiesin Drupalin taipuvan näihinkin tarpeisiin. Hyvin toteutettuna Drupalilla toteutettu sivusto on myös sisällöiltään äärimmäisen helposti hallittava, joka olikin tärkeää, sillä sivuston ylläpitäjät eivät omaa vahvaa tietoteknistä osaamista. Drupal on avoimen lähdekoodin ohjelmisto, se on ilmainen ja myös turvallinen, sillä Drupalin takana on monia erittäin osaavia ammattilaisia, jotka tekevät vapaaehtoisesti paljon työtä, että Drupal olisi tietoturvallinen. Myös WordPress olisi varmasti myöskin avoimen lähdekoodin ohjelmana ollut oiva valinta sisällönhallintajärjestelmäksi tälle sivustolle, mutta itsellä ei ole siitä niin hyvää osaamista eikä se mielestäni ole niin helposti muovattavissa kuin Drupal. WordPressin kanssa olen myös törmännyt huomattavasti useammin maksullisiin laajennuksiin, joita Drupal-yhteisössä ole käytännössä ollenkaan, joten tällaisessa 0 € budjetissa tehdyssä projektissa oli tästäkin syystä mielestäni turvallisempaa valita Drupal, koska laajennuksien hankkiminen Drupalille on lähes aina ilmaista.

4 DRUPALISTA

4.1 Drupalista yleisesti

Drupal on sisällönhallintajärjestelmä, jonka avulla voi luoda ja ylläpitää monia erilaisia www-sivustoja, ilman että tarvitsee osata mitään ohjelmointikieliä. Drupalin takana

on suuri yhteisö, sillä Drupal on avoimen lähdekoodin ohjelmisto, eli kuka tahansa osaava henkilö pystyy kehittämään Drupalia. (Redding 2010, 2-3; phulebar, tvn, Hunter & Nordhaug 2015.)

Ymmärtääkseen hyvin Drupalin toimintaa pitää ymmärtää, miten tieto kulkee Drupalin eri tasojen välillä. Kuvassa 1 on havainnollistettu Drupalin viiden päätason suhteita keskenään. Nämä päätasot ovat

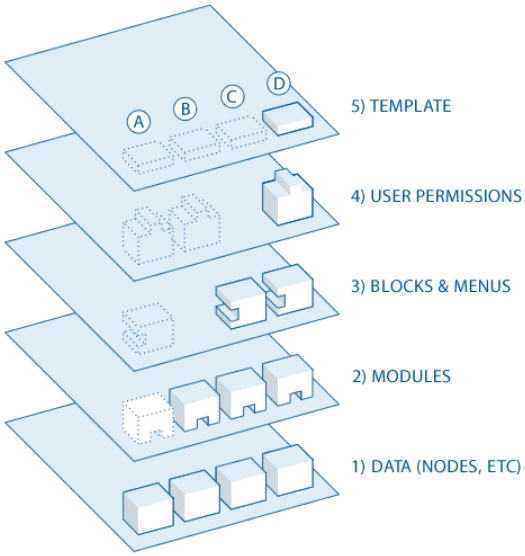
1. data (solmut ym.)
2. moduulit
3. lohkot ja valikot
4. käyttöoikeudet
5. teema.

(ccgearig, kalamityjane, Young & RonnieRayB 2014.)

Ensimmäisenä on datataso, sillä ennen kuin mitään voi näyttää sivustolla, pitää se tallentaa sivustolle datana, esimerkiksi *solmuina*. Toisena tasona on *moduulit*, joilla rakennetaan kaikki Drupalin toiminnallisuus. Moduulit, joita käytetään voi olla Drupalin ytimeen kuuluvia moduuleja, moduuleita jotka jotkut muut ovat tehneet tai ihan itse tehtyjä moduuleita. Moduulien avulla voi kustomoida solmujen ym. sisältämää dataa. (ccgearig ym. 2014.)

Kolmantena tasona on *lohkot* ja *valikot*. Lohkojen avulla data saadaan näkymään moduuleiden määrittelemällä tavalla. Lohkoihin voi myös asettaa suoraan omaa sisältöä. Lohkojen näkyvyyttä voi määritellä usealla eri tavalla, sen saa asettaa vapaasti sivun *mallinteeseen* haluamaansa kohtaan, sen voi määritellä näkymään tietyille sivuille tai tietyille *käyttäjärooleille*. Valikot puolestaan ovat Drupalin navigaatio, joka määrittää kaikelle sisällölle valikkopolun. (ccgearig ym. 2014.)

Käyttöoikeuksien avulla määritellään mitä mikäkin käyttäjärooli saa ja mitä ei saa sivustolla nähdä. Sivuston näkymä puolestaan luodaan päällimäisellä tasolla, joka on sivun teema. Tämä on rakennettu lähinnä XHTML:ää ja CSS:ää käyttäen, mukana myös hieman PHP-muuttujia, jotta Drupalin sisällöt menevät asianmukaisille paikoilleen. Teeman avulla voi myös yliajaa moduulien funktioita ja näin muokata moduulien tapaa näyttää dataa sivustolla. (ccgearig ym. 2014.)



Kuva 1. Drupalin eri tasot (ccgearig ym. 2014).

Kuvassa 1 näkyy myös neljä eri tilannetta (A, B, C ja D) siitä, miten tasot toimivat keskenään. Tilanteessa A on dataa, jota näyttää, muttei moduulia, jolla datan näyttäminen määritellään, joten sitä ei saa näkymään mitenkään. Tilanteessa B puolestaan on dataa ja moduuli, jonka avulla määritellään sen näkyminen, käyttöoikeudet ovat kunnossa, mutta lohkoa ei ole asetettu näkymään missään, joten datalla ei ole mahdollisuutta näkyä sivustolla. Tilanteessa C puolestaan on data, jota näyttää, moduuli, jonka avulla datan näkyminen määritellään, lohko, jossa data näkyisi, mutta käyttöoikeudet ovat pielessä, joten data ei näy sivustolla. Nämä viisi tasoa siis tarvitsevat kaikki toisiaan, ja niiden asetukset tarvitsee olla kunnossa, että sivustolle saadaan näkymään halutut asiat. (ccgearig ym. 2014.)

4.2 Sisällön tallennus Drupal-sivustolle

Kaikki sisältö Drupal-sivustolla tallennetaan ja käsitellään *solmuina (eng. node)*. Solmu voi olla yhtä monista *sisältötyypeistä*. Solmu on mikä tahansa yksilöllinen sisältö sivustolla, kuten sivu, artikkeli tai blogiteksti. Solmuilla on muutamia oletus*kenttiä*, mutta niitä, kuten myös sisältötyyppejä, voi tehdä itse lisää niin paljon kuin haluaa. (Choudhury, Feidt, Foley & Ringwood 2015.)

4.2.1 Solmut

Solmujen käsittelyn takana on Solmu-moduuli. Moduulin avulla pystyy listaamaan, järjestämään ja hallitsemaan kaikkea sisältöä sivustollasi. Moduulin avulla hallitaan myös sitä, miten solmujen tiedot näkyvät sivustolla. Jokainen solmu sisältää seuraavat tiedot:

- **NID** (Node ID, Solmu ID) – jokaisella sivuston solmulla oleva yksiöllinen ID. Kaikkien solmujen polku muodostuu NID:in avulla, niin että polku on muotoa esimerkkisivu.fi/node/[id], esimerkiksi esimerkkisivu.fi/node/5.
- **Otsikko** – jokaisella solmulla on otsikko, mutta sen ei tarvitse olla yksilöllinen.
- **Tila** – tieto siitä, onko solmu julkaistu vai ei. Julkaisematon sisältö ei näy kuin sivuston ylläpitäjille (käyttöoikeuksien mukaan).
- **Luomisajankohta ja muokkausajankohta** – jokaisellle solmulle tallentuu automaattisesti aikaleima, kun se tehdään ja kun sitä muokataan (vain viimeisin muokkausajankohta tallentuu).

(Redding 2010, 92; ccgearig, phulebar, Guleria & Doesborg 2014.)

Näiden lisäksi Drupal pitää yllä versiohistoriaa, jos kyseinen ominaisuus on sallittu. Jokaiselle solmulle tallentuu tieto myös mitä kieltä se on. (Redding 2010, 92; ccgearig, phulebar, Guleria & Doesborg 2014.)

4.2.2 Sisältötyypit ja kentät

Sisältötyyppejä (eng. content type) hallitaan myös Solmu-moduulilla. Erilaisten sisältötyyppien avulla pystyy luomaan Drupal-sivustolle erilaisia sisältöjä erilaisiin tarkoituksiin. Kun Drupal-sivustolle lisätään sisältöä, sen tarvitsee aina olla jotakin sisältötyyppiä. Drupalin perusasennuksen mukana tulee valmiiksi kaksi sisältötyyppiä: Artikkeli ja Sivu. Sisältötyyppejä voi hankkia lisää asentamalla moduuleita tai tekemällä itse. (ccgearig ym. 2014a; snehi, Kumar, Bah & Doesborg 2015.) Sisältötyyppien ei tarvitse olla vain tekstisivuja tai vastaavia, vaan sisältötyyppejä voidaan tehdä monipuolisemminkin. Esimerkiksi jos sivustolle halutaan tallentaa ihmisten yhteystietoja,

sivustolle voi tehdä Henkilö-sisältötyypin tai jos sivustolle halutaan tallentaa tietoja jääkiekko-otteluista, sisältötyyppi Ottelu olisi täysin mahdollinen.

Kenttiä (eng. fields) hallinnoidaan Kenttä-moduulilla. Kenttämoduulin avulla hallitaan *kenttätyyppejä*, joita ovat esimerkiksi teksti-, kuva- ja numerokentät. Kenttätyyppejä saa lisää asentamalla uusia moduuleita. Tehdessä uutta kenttää tarvitsee tietää ainakin seuraavat asiat: kentän nimi, millaista tietoa kenttään tallennetaan, miten tieto tallennetaan ja kuinka monta arvoa yhteen kenttään voi tietoa asettaa. Kentän nimi on se etiketti, joka näkyy sisältöä luodessa, mutta kentällä on myös koneluottava nimi, jonka perusteella se tallentuu tietokantaan. Kentän koneluettavaa nimeä ei voi jälkikäteen muuttaa, mutta etiketti on muokattavissa. Myöskään kenttään tallennettavan tiedon muotoa ei voi jälkikäteen muokata. Tieto voi olla vain yhtä tyyppiä, esimerkiksi kuva, teksti ja numero, ja tietotyyppejä saa lisää asentamalla uusia moduuleita. Jokaisella tietotyypillä on yksi tai useampi ”widget”, jonka avulla tietoa lisätään; esimerkiksi tekstilaatikko, valikkolista, valintaruudut ja tiedoston lataus. Tämän voi vaihtaa myöhemmin toiseen, jos tietotyyppi tarjoaa enemmän kuin yhden vaihtoehdon. Näitäkin pystyy joillekin tietotyypeille lisäämään asentamalla moduuleita. Viimeisenä mainittu kentän arvojen määrän voi määritellä olemaan jokin tietty numero tai rajattomaksi, antaen käyttäjän lisätä niin monta arvoa kuin haluaa. Tätä voi muokata jälkikäteen, mutta jos rajaa pienemmäksi arvojen määrää, voi jo tallennetuista sisällöistä kadota joitakin arvoja. (Choudhury, Brown, Gutierrez & Liskey 2015.)

Kenttiä ei aina tarvitse tai kannata tehdä alusta asti uudelleen. Jo tehtyjä kenttiä voi lisätä toisiin sisältötyyppeihin helposti. Kenttien jotkut arvot (kuten kentän arvojen lukumäärä) ovat kaikissa sisällöissä samat, mutta joitakin arvoja (kuten kentät etiketti) voi muokata jokaiselle sisältötyypille omakseen. Kaksi hyvää syytä käyttää kenttiä uudelleen ovat ensinnäkin se, että säästää aikaa, kun ei tarvitse alusta asti määritellä kentän asetuksia uudelleen, ja toiseksi käyttämällä samoja kenttiä eri sisältötyypeissä voi helposti näyttää, rajata, ryhmitellä ja järjestää samankaltaisia kenttiä riippumatta siitä mihin sisältötyyppiin ne ovat tallennettuna. Kuitenkin jos halutaan säätää näkyvyyttä eri käyttäjille erilaisiksi eri sisältötyypeissä, kannattaa myös käyttää eri kenttiä, jotta on helpompi määritellä aina tietty kenttä näkymään tietyille käyttäjärooleille. (Choudhury ym. 2015.)

Kentät ovat käytössä muuallakin kuin sisältötyypeissä. Muun muassa *luokitteluissa* ja *käyttäjätileissä* voi myös käyttää kenttiä. Esimerkiksi luokittelussa voi asettaa ikonikentän luokittelulla, jotta jokaiselle termille voi asettaa oman ikonin kun taas käyttäjätilille voidaan lisätä esimerkiksi biografia. (Choudhury ym. 2015.)

4.2.3 Luokittelu

Luokittelun (eng. taxonomy) avulla pystyy liittämään, yhdistämään ja luokittelemaan sivuston sisältöjä. Luokittelu vastaa muiden järjestelmien kategoriointia, tageja, metadataa tai muuta vastaavaa. Drupalissa *termit (eng. terms)* on koottu aina *sanastoihin (eng. vocabulary)*. Luokittelu-moduulin avulla voi luoda, hallita ja käyttää sanastoja. Sanastoilla on aina oma yksilöllinen ID ja kaikilla sanaston termeillä on oma sanastokohtainen yksilöllinen ID. Jokainen sanasto muodostuu termeistä ja jokaisella sanastolla voi olla rajaton määrä termejä. Myös sanastoja voi olla sivustolla rajaton määrä. Sanaston termit voi järjestää hierarkkisesti. (SERVANT14, Hunter, crawleyde & Tracy 2013; Brown, Hunter, Tracy & Daglees 2014.)

Jotta luokittelua voi käyttää Drupalin sisällöissä, tarvitsee haluttuun sisältötyyppiin lisätä Termireferenssi-kenttä, jonka avulla luokittelu tapahtuu. Kentän käyttöönottoasetuksissa tarvitsee valita sanasto, jonka perusteella sisältö luokitellaan. Luokittelu tulee tarpeeseen esimerkiksi silloin, jos haluaa lisätä sivustolle uutisia eri maista, niin että uutisesta tulee ilmi, mistä maasta kyseinen uutinen on. Sen sijaan että tekisi jokaiselle maalle oman Uutinen-sisältötyypin, kannattaa tehdä uusi Uutismaat-sanasto, jossa on termeinä kaikki maat joista uutisia tulee, ja yksi Uutinen-sisältötyyppi, johon lisää termireferenssikentän, joka käyttää sanastoa Uutismaat. Näin esimerkiksi uutisten sivuja on helpompi ylläpitää saman näköisinä, mutta kuitenkin pystyy luokittelemaan ne maittain. (Redding 2010, 93; Matthews, Hunter, Curtain 2013.)

4.3 Moduulit

Moduulit (eng. moduls) ovat liitännäisiä, jotka laajentavat, rakentavat tai parantavat Drupalin toiminnallisuuksia. Moduulit voidaan jakaa kolmeen eri kategoriaan:

- **ytimen moduuleihin (core)**, eli moduuleihin, jotka tulevat Drupalin asennuksen mukana
    - Näistä jotkut ovat pakollisia Drupalin toiminnan kannalta ja jotkut saa ottaa vapaasti käyttöön.
- **Drupal-yhteisön tekemiin (contributed)**, eli moduuleihin, jotka ovat ladattavissa drupal.org sivuilta ja jotka voi asentaa Drupal-sivustolle
- **omiin moduulieihin (custom)**, jotka ovat itse tehtyjä moduuleita omalle sivustolleen sopiviksi.

(Mercer 2010, 50; dcdenison, Brown, pbulebar & Gerndt 2015.)

Moduulit otetaan käyttöön asentamalla moduulin sivustolle, jollei se ole asennettuna, minkä jälkeen moduulien hallintasivulla moduuli otetaan käyttöön. Moduulin voi aina ottaa pois käytöstä, jos sivusto ei sitä tarvitsekaan. Ne sivustot, jotka eivät tarvitse mitään erikoistoiminnallisuuksia, eivät tarvitse mitään lisättyjä moduuleita, kun taas puolestaan erikoistoiminnallisuuksia täynnä oleva sivusto saadaan tehtyä juuri oikeanlaiseksi moduuleita asentamalla ja ottamalla käyttöön. Moduuleiden avulla voi esimerkiksi lisätä sisältötyyppejä, kuten resepti, blogikirjoitus tai tiedosto ja myös toimintoja, kuten sähköposti-ilmoitukset. (Tomlinson & VanDyk 2010, 3-5.)

4.4 Lohkot ja valikot

4.4.1 Lohkot

Lohkot (eng. blocks) sisältävät toiminnallisuuksia tai sisältöä joka on näkyvillä monissa paikoissa sivustolla sen mukaan, millaiset näkyvyysasetukset lohkolle on määritelty. Lohkot ovat usein moduulien muodostamia, mutta ne voi luoda itsekin, niin että esimerkiksi lohko näyttää upotetun YouTube-videon. (Redding 2010, 70; Mercer 2010, 69.)

Lohkon saa näkyviin sivustolle ottamalla lohkon käyttöön, eli asettamalla sen johonkin teeman *alueelle*. Ne lohkot, joita ei ole asetettu mihinkään alueelle, eivät näy sivustolla. Koska lohkot asetetaan teeman alueelle, voi lohkon asettaa eri alueille eri tee-

moissa, jos on käytössä useampi kuin yksi teema. Lohkoja voi myös järjestellä pystysuunnassa alueen sisällä, eli mikä lohko on ylimpänä kyseisellä alueella, mikä toisena ja niin edelleen. Lohkon näkyvyyden saa määriteltyä sivu-, käyttäjärooli-, sisältötyyppi- ja käyttäjäkohtaisesti. (Redding 2010, 73-74; dags, James, Maat & Singh 2014.)

Sivukohtainen näkyvyys tarkoittaa sitä, että valittavana on joko näyttää lohko kaikilla, paitsi listatuilla sivuilla, tai näyttää lohko ainoastaan listatuilla sivuilla. Oletuksena lohko näkyy kaikilla sivuilla. Sivut listataan sivujen polkujen mukaan. Käytössä on myös ’*’-merkki, jonka voi laittaa merkkijonon perään, ja näin ollen määritys koskee kaikkia kyseessä olevalla merkkijonolla alkavia polkuja. Poikkeuksena on etusivu, joka määritellään merkkijonolla ’<front>’. (Redding 2010, 73; dags ym. 2014.) Esimerkiksi jos merkkijonona on ’user’, määritys koskee sivua esimerkki.fi/user, kun taas jos merkkijono on ’user*’, niin määritys koskee sivun esimerkki.fi/user lisäksi myös muita samanalkuisia sivuja, kuten esimerkki.fi/user/login. Jos puolestaan haluaa, ettei määritys koske sivua esimerkki.fi/user, mutta kuitenkin kaikkia sen alasivuja, kuten esimerkki.fi/user/login, niin tämän saavuttaa laittamalla merkkijonoksi ’user/*’.

Käyttäjäroolikohtainen näkyvyys tarkoittaa sitä, että lohkon voi määrittää näkymään yhdelle tai useammalle seuraavista käyttäjärooleista: ylläpitäjä, anonyymi (käyttäjä joka ei ole kirjautunut sivustolle) tai todennettu käyttäjä (sivustolle kirjautunut käyttäjä). Oletuksena lohko näkyy kaikille käyttäjille. (Redding 2010, 74.) Esimerkiksi jos lohkoa ei halua vielä julkaista näkymään sivuston käyttäjille, niin asettaa lohkon näkymään vain ylläpitäjäroolissa olevilla. Toisaalta jos haluaa näyttää sivustolla mainoksia käyttäjille, jotka eivät ole kirjautuneet sisään, niin voi määrittää lohkon näkymään vain anonyymeille käyttäjille, joten kun kirjautuu sisään, mainokset katoavat.

Sisältötyyppikohtainen näkyvyys tarkoittaa sitä, että lohkon voi määrittää näkymään vain tietyillä sisältötyyppien sivuilla. Koska lohkot oletuksena näkyvät kaikilla sivuilla, näkyvät ne myös kaikkien sisältötyyppien sivuilla. Jos kuitenkin määrittää lohkon näkymään vain tietyillä sisältötyyppien sivuilla, niin lohko ei näy myöskään sivuilla, jotka eivät ole solmun sivuja, esimerkiksi käyttäjätilin hallintasivu tai etusivu.

Käyttäjäkohtainen näkyvyys tarkoittaa sitä, että lohkon voi määrittää niin, että käyttäjä näkee lohkon jos haluaa. Määrittäessä pitää myös määrittää se, näkyykö lohko, vai onko se piilotettu oletuksena. Kun tämä määritys on päällä, voi käyttäjä itse joko piilottaa tai näyttää lohkon. (Redding 2010, 74; dags ym. 2014.)

4.4.2 Valikot

Valikot (eng. menus) ovat kokoelma linkkejä, joiden avulla navigoidaan sivustolla. Valikkoja hallinnoidaan ja luodaan Valikko-moduulilla, joka tekee aina automaattisesti kaikista valikoista lohkon. Lohko koostuu valikon linkeistä hierarkkisessa järjestyksessä. Valikkoja ja valikon elementtejä (linkkejä) voi lisätä, poistaa ja uudelleen nimetä. Luodessa uutta sisältöä voi samalla määrittää kyseisen sisällön valikkolinkin. (Carlile, Chopra, deeparandad, Singh 2015.)

4.5 Käyttäjät ja käyttöoikeudet

Sen sijaan että yksittäiselle *käyttäjälle* asetetaan omat *käyttöoikeudet*, Drupalissa käyttöoikeudet määrätään *roolin* mukaan, ja jokaiselle käyttäjälle annetaan tietty rooli. Oletuksena Drupalissa on roolit ylläpitäjä, anonyymi ja todennettu käyttäjä. Anonyymin käyttäjän rooli on kaikilla sivuston käyttäjillä, jotka eivät ole kirjautuneet sisälle, ja puolestaan todennettu käyttäjä on kaikkien sisälle kirjautuneiden käyttäjien rooli. Rooleja voi sivustolla lisätä niin paljon kuin tarvitsee, ja roolien käyttöoikeudet saa määrittää aivan kuten tahtoo. Kuitenkin se käyttäjätunnus, joka luodaan myös sivustoa luodessa ja joka on sivuston tekijän käyttäjätunnus, on poikkeus, koska sillä on oikeus aina muokata mitä vain sivuston asetuksia, eikä näitä oikeuksia voi ottaa pois. Jokaisella sivustolla on vain yksi tällainen käyttäjätunnus. Kaikilla muilla rooleilla voi olla niin monta käyttäjätunnusta kuin sivusto vain tarvitsee. (Gruber, Bah, Matthews & Tadisetty 2014; ccgearig, Wilson & Ryan 2014.)

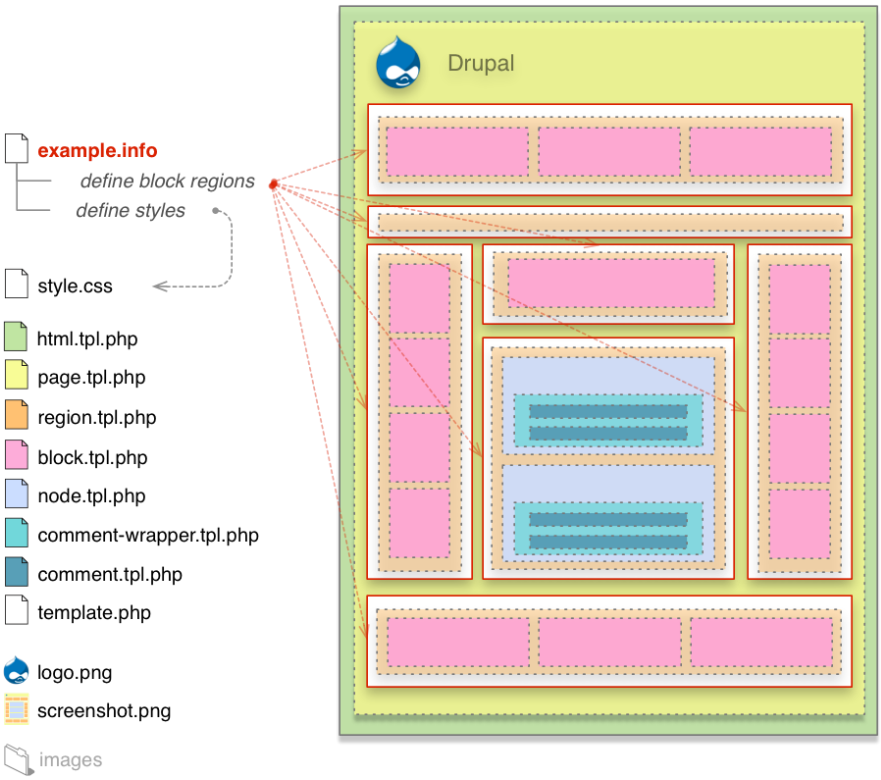
Esimerkiksi jos tehdään sivusto, jossa jotkut käyttäjät voivat kirjoittaa artikkeleita, mutta toiset käyttäjät eivät, kannattaa tehdä oma käyttäjärooli heille, jotka voivat artikkeleita kirjoittaa, esimerkiksi ”Kirjoittaja”. Kun yksi ”Kirjoittaja”-roolin omaava

käyttäjä kirjoittaakin jotakin sivustolle sopimatonta, ja häneltä halutaan ottaa kirjoitusoikeus pois, hänen roolinsa vain vaihdetaan normaaliin sisäänkirjautuneen rooliin (todennettu käyttäjä).

4.6 Teemat

Valittu *teema (eng. theme)* määrittää sivun ulkoasun. Teematiedostot määräävät elementtien sijainnin ruudulla ja vaikuttavat sekä siihen, miten sisältö sivustolla esitetään että siihen, miten käytettäviä sivuston toiminnallisuudet ovat. Teemat ovat Drupal-sivuston näkyvin osa. (Shrevers 2011, 8.)

Drupal-teema koostuu tiedostoista, jotka määrittävät ja kontrolloivat Drupal-sivuston ominaisuuksia käyttäen PHP:tä, HTML:ää, CSS:ää ja kuvia. Eri Drupal-*mallinetiedostot (template files)* kontrolloivat eri *osioita* sivuilla. Mallinetiedostot ovat Kuvassa 2 näkyvät '.tpl.php' –päätteiset tiedostot. Nämä tiedostot käsittelevät kaikki tietyn datakokonaisuuden näyttämistä sivustolla. Ne eivät ole pakollisia, ja jos teemassa ei ole näitä määrityksiä, määritykset otetaan Drupalin ytimen oletuksista. Teemalla on kuitenkin pakko olla .info-päätteinen tiedosto, jossa pitää vähintään kertoa teeman nimi ja se, mihin Drupalin ytimen pääversioon se on tarkoitettu (esimerkiksi 'core = 7.x'). (Needham, Riemenschneider, Hannemyr & Rosenfeld 2013; James, dustise, Rana & drupalshrek 2015.)



Kuva 2. Drupal-teeman tiedostot (James ym. 2015).

4.7 Näkymät

Näkymät-moduulin *(eng. Views)* avulla voi tietokannasta hakea sisältöä ja esittää sen käyttäjälle listana, galleriana, taulukkona, valikkolinkkeinä, lohkoina ja monilla muilla eri tavoilla. Näkymien avulla voidaan näyttää solmuja, käyttäjä ja muita sisältöjä mitä sivustolle on tallennettu. Näkymillä voi muodostaa sivun määrittelemällä näkymiin sivun polun, lohkon, syötteen tai muita erilaisia näkymiä. Näkymien avulla voi myös käyttää suhderiippuvuuksia, esimerkiksi voi luoda listan käyttäjistä, jossa näkyy myös linkit kunkin käyttäjän luomiin artikkeleihin. (Pachabhaiya, Feidt, Foley & Guo 2015.)

Näkymien avulla voi luoda, hallita ja näyttää sisältölistoja. Jokaista listaa kutsutaan *näkymäksi*. Näkymässä tärkeitä käsitteitä ovat muun muassa *kentät, suhteet, argumentit, järjestyskriteerit, suodattimet, näytöt, yläosa, alaosa* ja *tyhjä teksti*.
Kentät, tai muu näyttävänä data, määrittää sen, mitä tarkalleen näkymä näyttää. Esimerkiksi voi valita kentät ’Solmu: Otsikko’, ’Solmu: Tyyppi’, ’Solmu: Julkaisupäivämäärä’ solmunäkymään ja näin ollen näkymän lopputuloksessa on kaikkien valittujen

solmujen otsikko, tyyppi ja julkaisupäivämäärä. (Tolboom, McGuire & Watcher 2015.)

Jos kahden tietotyypin välissä on jonkinlainen **suhde** tai datat liittyvät jotenkin toisiinsa, voidaan näitä käyttää näkymässä hyväksi. Esimerkiksi solmun ja solmun kirjoittajan välillä on suhde, ja tätä voidaan hyödyntää, kun tehdään näkymä, jossa on näkyvissä tietyn käyttäjän kaikki tekemät solmut. (Tolboom ym. 2015.)

Argumentit tulevat sivun polusta. Esimerkiksi jos lisätään argumentti ’Solmu: Tyyppi’ solmunäkymään, jonka polku on ”sisalto”, voidaan dynaamisesti näyttää solmuja sisältötyypeittäin. Näin voidaan määrittää, että sivulla esimerkki.fi/sisalto/sivu näkyvät kaikki solmut joiden sisältötyyppinä on sivu, kun taas sivulla esimerkki.fi/sisalto/blogi näkyvät kaikki blogi-sisältötyyppiä edustavat solmut. Sivulla esimerkki.fi/sisalto näkyvät puolestaan kaikki solmut huolimatta niiden sisältötyypistä. (Tolboom ym. 2015.)

Järjestyskriteerien avulla voi datalistaa järjestää niiden sisältämän datan perusteella. Esimerkiksi jos lisätään järjestyskriteeri ’Solmu: Julkaisupäivämäärä’ laskevassa järjestyksessä solmunäkymään, niin lopullisessa tuloksessa solmut ovat laskevassa järjestyksessä sen mukaan milloin ne ovat luotu. (Tolboom ym. 2015.)

Suodattimien avulla voi suodattaa joitakin tuloksia pois lopullisesta näkymästä. Esimerkiksi solmunäkymästä saa suodatettua kaikki julkaisemattomat solmut asettamalla suodattimen ’Solmu: Julkaistu’ olemaan yhtä suuri kuin ’Julkaistu’. Näkymän voi esittää muun muassa lohkona tai sivuna, ja tämä määritellään **näyttöjen** avulla. Näkymällä on aina yksi oletus näyttö, joka ei näy minnekään, mutta jonka asetukset ovat koko näkymän oletusasetukset. (Tolboom ym. 2015.)

Yläosaan voi asettaa yhden tai useamman tekstialueen, joka siis tulee näkymän lopputulokseen tuloksien yläpuolella. Myös **alaosaan** voi lisätä tekstialueita, ne näkyvät lopputuloksessa tuloslistauksen alapuolella. Näkymään voi asettaa jonkin tekstin, joka näkyy vain silloin jos tuloksia ei näkymästä tule. Tätä kutsutaan **tyhjäksi tekstiksi.** (Tolboom ym. 2015.)

5 SIVUSTON TOTEUTTAMINEN

5.1 Drupalin asentaminen

Projektin toinen jäsen, joka oli teatterin puolen yhteyshenkilö, oli hankkinut webhotellipalvelu Neobitiltä palvelintilaa, johon kuului myös tietokantatilaa. Koska palvelintila oli heti ohjattu oikeaan domainiin, tein ensimmäiseksi palvelimelle yksinkertaisen index.html-tiedoston css-tyylitiedostoineen, jottei keskeneräinen sivusto näkyisi liian helposti muille ihmisille. Kuvassa 3 näkyy tämä väliaikainen sivusto.

Kuva 3. Väliaikainen sivu

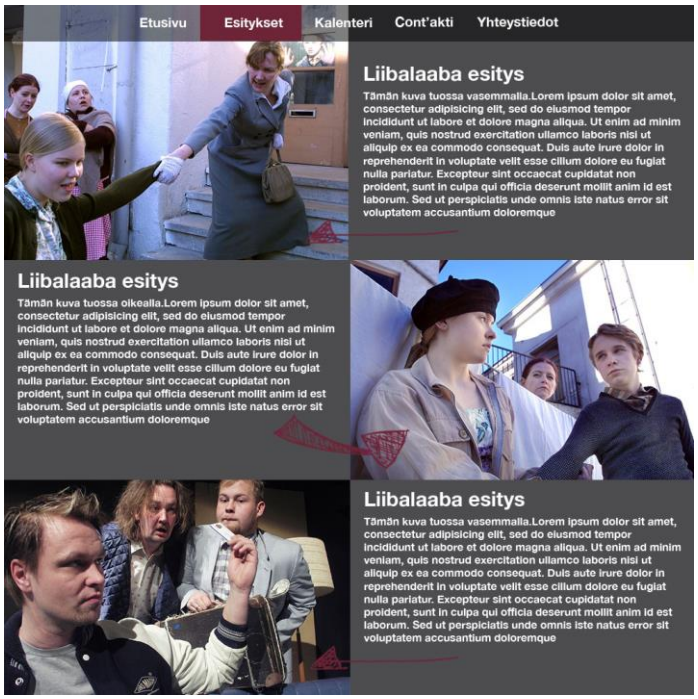
Varsinaisen Drupalin alustuksen tein palvelimelle sandbox-kansioon käyttäen Drupal-sivuilta saatavaa latauspakettia. Drupalin alustusta varten minun piti myös luoda tyhjä sql-tietokanta, mikä onnistui helposti cPanelin avulla. Alustuksen jälkeen minulla oli hyvin vajavainen pohja sivuston tekemiseen.

5.2 Ulkoasumääritykset

Ulkoasupohjan sain projektin toiselta henkilöltä. Kuvassa 4 näkyy etusivu ja myös valikon ja logon paikat. Valikossa muista poikkeavasti oleva punainen ”Etusivu”-teksti osoittaa millainen valikkolinkin pitäisi olla aktiivisena, eli kun käyttäjä on kyseisellä sivulla.

Kuva 4. Etusivun ulkoasupohja

Kuvassa 5 on puolestaan ideoitu miten esitykset listautuisivat sivustolle. Tarkoituksena on siis, että jokaisesta esityksestä olisi aina aika iso kuva, jonka vieressä olisi kaikki tiedot esityksestä (nuolet vain havainnollistamassa mikä teksti kuuluu mihinkin kuvaan). Kuvassa näkyy myös miten sivuston yläosa käyttäytyy kun rullataan hiirellä alemmaksi sivustolla; yläosa säilyy aina näkyvissä ylhäällä, kuitenkin niin että logo jää pois.



Kuva 5. Esitysten ulkoasupohja

Pohdimme yhdessä projektin toisen henkilön kanssa miten sivuston rakenne toteutetaan, teenkö sisäsivuja ja jos niin miten. Päädyimme kuitenkin siihen tulokseen, että yhden sivun sivusto riittää kaikelle sisällölle aivan hyvin, eikä sisäsivuja näin ollen tarvita. Sivustolle olisi siis ensimmäiseksi näkyvissä kuvassa 5 näkyvä etusivu jonka alapuolella olisi kuvassa 6 näkyvät esitykset, jonka alla puolestaan jonkinlainen kalenteri ja lippujen varaus toiminto, jonka jälkeen Cont'akti-teatterista yleisesti ja vielä alimpana yhteystietoja. Valikko toimisi niin, että valikkolinkit olisivat ankkurilinkkejä kuhunkin osaan, ja ne olisivat niin kauan aktiivisia, eli kuvassa 5 näkyvällä tavalla värillisiä, kunnes käyttäjä olisi rullannut seuraavaan osioon, jolloin tähän osioon johtava valikkolinkki puolestaan vaihtuisi aktiiviseksi.

5.3 Teeman valinta ja määrittely

Koska toimeksiantajan puolesta en saanut mitään erityisvaatimuksia, eikä ulkoasupohjakaan ollut todella tiukka, päätin valita teemaksi Drupalin Boostrap-teeman. Valintaan päädyin Boostrapin helpon responsiivisuuden vuoksi, Boostrapista voi lukea lisää osoitteesta http://getbootstrap.com/ ja siellä kannattaa tutustua erityisesti http://getbootstrap.com/css.

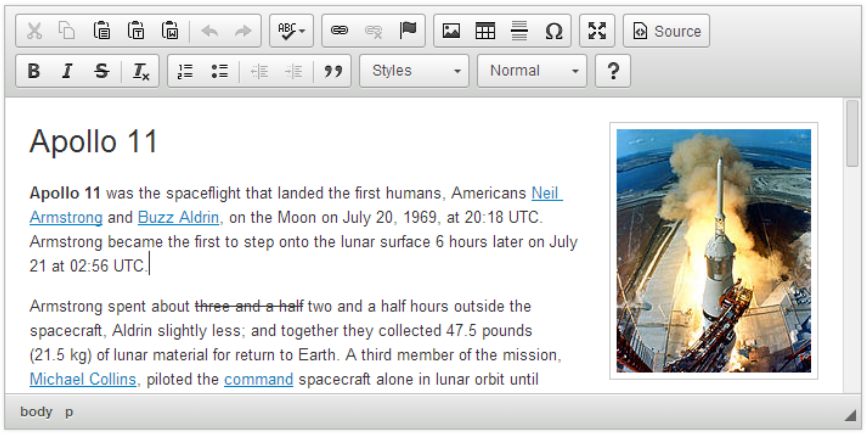
Tein Bootsrap-teemasta sivuja varten aliteeman Contakti, jonne määritin kaikki tarvitsemani alueet: esitykset, kalenteri ja contakti. Otin myös seuraavat oletusalueet pois käytöstä, etteivät ne näkyisi missään turhanpäiten: higlighted, help, primary and secondary. Kuvassa 6 näkyy miten laitoin kyseiset alueet page.tpl.php-tiedostoon, jolloin ne tulevat myös sivuille näkymään.

```
<dic class="wrapper-container">
  <div class="bg">
  </div>
  <div class="main-container container">

    <header role="banner" id="page-header">
      <?php if (!empty($site_slogan)): ?>
        <p class="lead"><?php print $site_slogan; ?></p>
      <?php endif; ?>

    </header> <!-- /#page-header -->

    <div class="row">

      <?php if (!empty($page['sidebar_first'])): ?>
        <aside class="col-sm-3" role="complementary">
          <?php print render($page['sidebar_first']); ?>
        </aside>  <!-- /#sidebar-first -->
      <?php endif; ?>

      <section<?php print $content_column_class; ?>>
        <?php if (!empty($page['highlighted'])): ?>
          <div class="highlighted jumbotron"><?php print render($page['highlighted']); ?></div>
        <?php endif; ?>
        <!--<?php if (!empty($breadcrumb)): print $breadcrumb; endif;?>-->
        <a id="main-content"></a>
        <?php print $messages; ?>
        <?php print render($title_prefix); ?>
        <?php if (!empty($title)): ?>
        <div class="col-md-8 col-md-push-2">
          <h1 class="page-header">Tervetuloa Cont'akti-teatteriin</h1>
        </div>
        <?php endif; ?>
        <?php print render($title_suffix); ?>
        <?php if (!empty($tabs)): ?>
          <?php print render($tabs); ?>
        <?php endif; ?>
        <?php if (!empty($page['help'])): ?>
          <?php print render($page['help']); ?>
        <?php endif; ?>
        <?php if (!empty($action_links)): ?>
          <ul class="action-links"><?php print render($action_links); ?></ul>
        <?php endif; ?>
        <div class="col-md-8 col-md-push-2">
          <?php print render($page['content']); ?>
        </div>
      </section>

      <?php if (!empty($page['sidebar_second'])): ?>
        <aside class="col-sm-3" role="complementary">
          <?php print render($page['sidebar_second']); ?>
        </aside>  <!-- /#sidebar-second -->
      <?php endif; ?>

    </div>
  </div>
</dic>
<div class="wrapper-esitykset">
  <div class="esitykset-container" id="esitykset">
    <?php print render($page['esitykset']); ?>
  </div>
</div>
<div class="wrapper-kalenteri">
  <div class="kalenteri-container container" id="kalenteri">
    <?php print render($page['kalenteri']); ?>
  </div>
</div>
<div class="wrapper-contakti">
  <div class="contakti-container container" id="contakti">
    <div class="col-md-8 col-md-push-2">
      <?php print render($page['contakti']); ?>
    </div>
  </div>
</div>
<div class="wrapper-footer" id="yhteystiedot">
<footer class="footer container">
  <div class="col-md-8 col-md-push-2">
    <?php print render($page['footer']); ?>
  </div>
</footer>
</div>
```

Kuva 6. Osa page.tpl.php-tiedostosta

5.4 Moduulien asennus ja valinta

Drupalin tarjoama asennus ei ole ennen moduulien asennusta erityisen käytettävä eikä varsinkaan muokattava, joten Drupalin alustamisen aloitin erinäisten moduulien asentamisella. Näistä tärkeimpiä oli Näkymät, jotka on käsitelty aiemmin luvussa 4 ja ckeditor, jonka avulla saa käyttöön paremman tekstinmuokkaustyökalun, kuten kuvasta 7 näkyy.



Kuva 7. Kuva CKEditorista (Walc 2009).

5.5 Sisältötyypin Esitys määrittely

Moduulien asentamisen jälkeen tein sivustolle ilmiselvästi kaivatun sisältötyypin Esitys. Kuten kuvasta 5 näkyy, niin esityksellä on ainakin otsikko, tekstiä ja kuva. Koska Drupal tekee automaattisesti aina uuden sisältötyypin luodessa sisällölle otsikko- ja sisältö-kentän, ei minun tarvinnut lisätä kuin kuva-kenttä, johon määritin laitettavaksi vain yhden kuvan esitystä kohti.

Kuitenkin sivustolle haluttiin myös jonnekin menneet esitykset näkyviin, muttei kuitenkaan kuvassa 5 näkyvänä listana, vaan piti tehdä tapa erottaa mennyt esitys tulevasta/nykyisestä esityksestä. Päätin etten tee menneille esityksille omaa sisältötyyppiä, sillä jokainen esityshän tulee joskus olemaan mennyt esitys, ja näin esitykset tarvitsisi ajan myötä tallentaa kahteen kertaan järjestelmään. Projektin teatteria edustava henkilö selvensi myös, että esityksillä ei useinkaan ole tarkkaa päättymispäivää tiedossa paljon sitä ennen, hyvin menestyneitä esityksiä kun voidaan jatkaa vielä seuraavalle

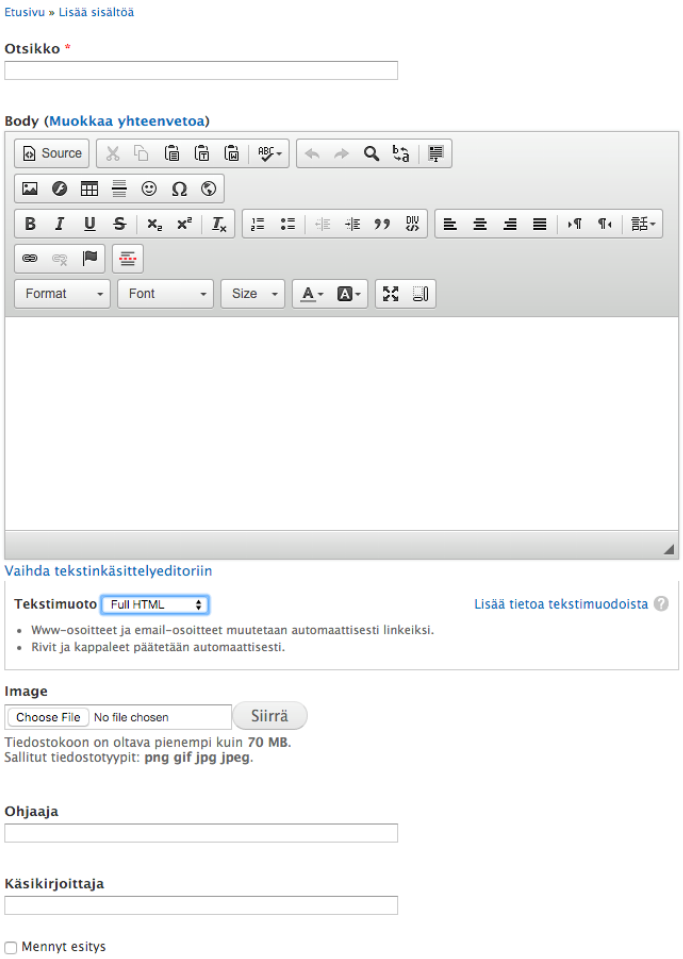
kaudelle. Tämän johdosta tein rajauksen menneisiin ja nykyisiin esityksiin todella yksinkertaisella tavalla, sillä lisäsin esitys-sisältötyyppiin totuusarvo-kentän ”Mennyt esitys”, jonka arvo ”true” tarkoittaa että esitys on mennyt esitys ja puolestaan ”false” tarkoittaa nykyistä/tulevaa esitystä.

Jo menneet esitykset sain tekstilistana, josta osa näkyy kuvassa 8. Esitykset olivat listattu kaikki samalla tavalla; vuosiluku, käsikirjoittaja, esityksen nimi ja ohjaaja. Tämän listan innoittamana päätin myös tehdä esitykselle tekstikentät ohjaajalle ja käsikirjoittajalle, jotta saisin helposti menneet esitykset kuvan 8 listan mukaisesti sivustolle. Vuosiluvun päätin ottaa Drupaliin automaattisesti tallentuvasta tallennusajankohdasta. Menneisiin esityksiin jouduin muokkaamaan tämän ajankohdan käsin oikeaan vuoteen.

1995 Hely Kallio, UNIKKOMERI, ohjaus Hely kallio
1995 Horace Mc Coy, MARATONTANSSIT, ohjaus Paul Salonen
1996 Jari Tervo, PYHIESI YHTEYTEEN, ohjaus Paul Salonen
1996 Henrik Ibsen, KANSANVIHOLLINEN, ohjaus Pekka Saaristo
1996 Tarmo Koivunen, UNELMARISTEILY, ohjaus Tarmo Koivunen
1997 Tracey Letts, KILLER JOE, ohjaus Paul Salonen
1998 Maria Jotuni, HUOJUVA TALO, ohjaus Paul Salonen
1998 Stephen King, PIINA, ohjaus Anne Mäkinen
1999 RAKKAUTTA, VALHEITA, VAIKENEMISIA, ohjaus Paul Salonen
1999 Martin McDonagh , VUORISTON KAUNOTAR, ohjaus Paul Salonen
1999 Seppo Parkkinen, UNOHDUKSEN TANGO, ohjaus Paul Salonen
1999 Valentin Chorell , KISSANKÄPÄLÄT, ohjaus Viola Westerlund
2000 Sean O´Casey, JUNO JA RIIKINKUKKO, ohjaus Ulla-Maija Aho-Harnett
2000 Irina Krohn, KINKKU PAHANA, ohjaus Paul Salonen
2000 ÄLKÄÄ JÄÄKÖ MEIRÄN ALLE, ohjaus Paul Salonen
2000 Hely Kallio, OMAISHOITAJAN PUOLUSTUSPUHE, ohjaus Hely kallio
2001 Dario Fo , RUUMIS MYYTÄVÄNÄ, ohjaus Mikko Salminen
2001 Strindberg, AAVESONAATTI, ohjaus Marko Saario
2002 Lassi Sinkkonen, SOLVEIGIN LAULU, ohjaus Paul Salonen
2002 Kenneth Grahame , KAISLIKOSSA SUHISEE, ohjaus Pasi Toivanen
2002 V-MÄISTÄ SUKKULOINTIA
2003 Minna Leino, PULUISTA JA IHMISISTÄ, ohjaus Anne Mäkinen

Kuva 8. Osa menneiden esitysten listasta.

Valmiissa esitys-sisältötyypissä oli siis seuraavat kentät: otsikko, sisältö, kuva, ohjaaja, käsikirjoittaja ja mennyt esitys. Kuvasta 9 näkyy minkälaisella lomakkeella esityksiä lisätään sivustolle.



Kuva 9. Esityksen lisäyssivu.

5.6 Esitysten näyttäminen sivustolla

Kun kaikki tarvittavat määritykset esitysten lisäämiseen olivat valmiit, tarvitsi vielä määrittää paljon, että esitykset näkyivät sivustolla halutusti. Nykyisten/tulevien esitysten määrittely oli helpompaa, koska ulkoasupohja oli jo määritelty (kuva 6). Tulevien esitys näyttämiseen päätin luoda uuden sisältö-näkymän, jonka näytöksi tein lohkon. Näytön muodoksi asetin muokkaamattoman listan, ja sisällön näytin sisältönä, käyttäen oletus esitysmuotoa. Eli näkymä tulostaa jokaisen solmun oletus-näkymän omalle rivilleen, kuvasta 10 näkyy miltä rivit näyttävät html-koodissa. Eli jokaisen solmun koodi on aina oman div-tagin, eli rivinsä sisällä. Kuvasta näkyy myös, miten Drupal automaattisesti tekee jokaiselle riville luokat sen mukaan kuinka mones rivi on kyseessä, onko rivi parillinen vai pariton, ja myös viimeinen ja ensimmäinen rivi ovat saanet omat luokkansa.

Kuva 10. Esitys-näkymän lohkon koodi, kommentit helpottamassa havainnointia

Esitys-näkymässä on rajattu näkymään ainoastaan solmut, jotka ovat julkaistussa tilassa ja tietysti myös ainoastaan esitys-sisältötyypin solmut. Koska kyseessä on tulevien/nykyisten esitysten näyttö, on tässä rajattu näkymään ainoastaan ne esitys-tyypin solmut, joiden ”Mennyt esitys”-kenttä on arvoa ”false”. Esitykset on näkymässä myös järjestetty näkymään lähetysajankohtansa mukaan laskevaan järjestykseen, jolloin viimeiseksi lisätty esitys näkyy listalla ylimpänä.

Myöhemmin esitän esitysten tarkemman toteuttamisen, mutta sitä ennen käyn läpi tämän näkymän toisen näytön, eli menneet esitykset. Menneissä esityksissä en käyttänyt samaa sisällön näyttötapaa, eli en tulostanut solmuja solmuina, vaan kenttinä. En myöskään käyttänyt muokkaamatonta listaa muotona, vaan latasin Views Responsive Grid-moduulin, jonka avulla pystyy määrittelemään paremmin miten solmut järjestetään. Asetin solmut menemään kahteen sarakkeeseen pystysuoralla tasauksella, jolloin järjestelmä tekee kuvan 11 koodista näkyvät kaksi saraketta, joihin tulee yhtä monta riviä (tai ensimmäiseen yksi enemmän jos on pariton määrä), ja riveihin puolestaan tulostuu määritetyt kentät.

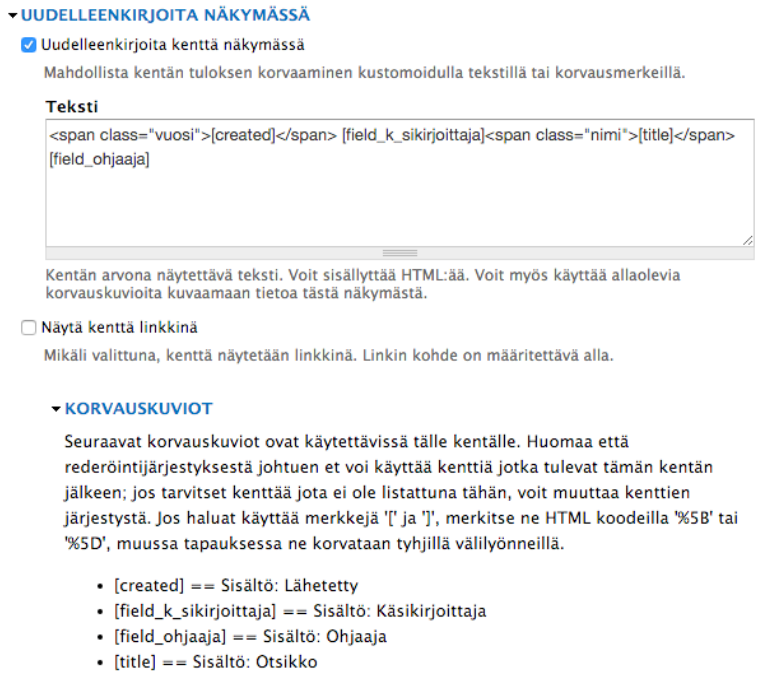
```
<!-- Esitykset näkymän lohko alkaa -->
<section id="block-views-esitykset-block-1" class="block block-views
   clearfix">
   <h2 class="block-title">Menneet esitykset</h2>
   <div class="view view-esitykset view-id-esitykset view-display-id-
      block_1 view-dom-id-4a2b1f4e2b6c50094dd98a00c81ad6f7">
      <div class="view-content">
         <div class="views-responsive-grid views-responsive-grid-vertical
            views-columns-2 container">
            <!-- Ensimmäinen sarake -->
            <div class="views-column col-sm-6 clearfix views-column-1
               views-column-first alpha">
               <!-- Ensimmäisen sarakkeen ensimmäinen rivi -->
               <div class="views-row clearfix views-row-1 views-row-
                  first">
                  <!-- Ensimmäisen solmun otsikko-kenttä -->
                  <div class="views-field views-field-title">
                     <span class="field-content">
                        <span class="vuosi">1989</span> Stephen Bill,
                           <span class="nimi">Syntymäpäivä</span>, ohjaus
                           Marjo-Riitta Ventola</span>
                     </div>
                  </div>
               <!-- Toinen rivi -->
               <div class="views-row clearfix views-row-2">
                  <!-- ......... -->
               </div>
               <!-- ......... -->
               <!-- Ensimmäisen sarakkeen viimeinen rivi -->
               <div class="views-row clearfix views-row-25 views-row-
                  last">
                  <!-- ......... -->
               </div>
            </div>
            <!-- Toinen sarake -->
            <div class="views-column col-sm-6 clearfix views-column-2
               views-column-last omega">
               <!-- Toisen sarakkeen ensimmäinen rivi -->
               <div class="views-row clearfix views-row-1 views-row-
                  first">
               <!-- ......... -->
               </div>
            <!-- ......... -->
            </div>
         </div>
      </div>
   </div>
</section>
```

Kuva 11. Esitys-näkymän menneiden esitysten näytön koodi, kommentit helpottamassa havainnointia

Kuten kuvasta 11 näkyy, tulostuu ainoastaan otsikko-kenttä. Tämä on kuitenkin vain puoli totuus, sillä koodista näkyy myös, otsikko-kenttä kertoo otsikon lisäksi vuosiluvun, käsikirjoittajan ja ohjaajan. Kuvassa 12 näkyvät menneiden esitysten näytön asetukset, ja näiden lisäksi kaikki kentät, jotka haen solmuista tätä näyttöä varten. Kuitenkaan kaikkia kenttiä ei näy omana kenttänään, kuten kuvasta 11 näkyy.

Kuva 12. Menneiden esitysten näkymän asetukset

Käytin tässä näkymässä hyväkseni näkymän kenttien uudelleenkirjoittamismahdollisuuksia. Asetin kentät lähetetty, käsikirjoittaja ja ohjaaja ”Piilota näkymässä”-tilaan, jolloin kentät eivät tulostu ollenkaan. Otsikko kentässä puolessa käytin hyväkseni kuvassa 13 näkyvää ”Uudelleenkirjoita näkymässä”-asetusta. Kuvassa näkyvät korvauskuviot tarkoittavat, että niiden tilalle lopullisessa tuloksessa tulee se, mitä kyseisen kentän ulostuloksi on määritelty. Esimerkiksi lähetetty kentässä jouduin uudelleen määrittelemään miten päivämäärä tulostetaan, sillä en kaivannut muuta kuin vuoden näkyviin. Otsikon uudelleenkirjoittamisella sain kaiken tiedon yhteen riviin ja näin ollen yhden div-tagin sisälle.



Kuva 13. Menneiden esitysten näytön otsikko-kentän uudelleenkirjoituksen määrittelyt

Tämän jälkeen asetin tekemäni lohkot näkymään etusivulle teemassa tekemässäni esitykset-alueeseen. Sijainti sivun koodissa näkyy kuvasta 6, eli lohkot tulostuvat näkymään ”$page[’esitykset’]”-kohtaan.

5.6.1 Tulevien ja nykyisten esitysten näkyminen

Tulevat ja nykyiset esitykset laitoin näkymässä näkymään oletussisältönä. Tämän sisällön ulkonäön määritykseen käytin Display Suite –moduulia ja sen pohjalta tehtyä Display Suite Bootsrap –moduulia, jonka avulla sain kätevästi tehtyä esityksistä Bootstrapin toiminnoilla toimivan responsiivisen ulostulon. Käytin Display Suite Bootsrapin valmiista pohjista Bootstrap Two Columns: 6-6 –pohjaa, jonka avulla sisällön saa jaettua kahdelle palstalle.

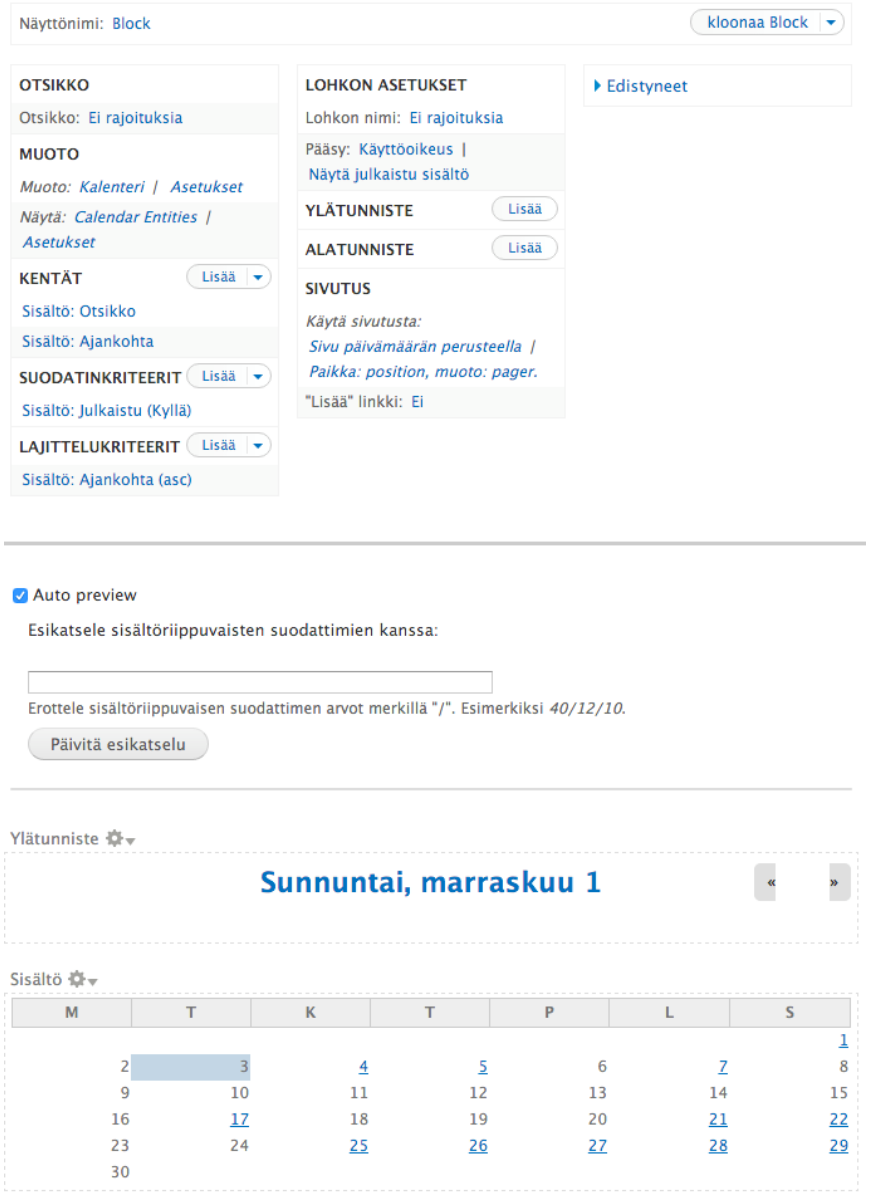
Kuva 14. Esityksen ulkoasumääritykset

Kuvassa 14 näkyy, miten kuvan olen laittanut toiselle puolelle ja loput sisällöt toiselle. ”Käsikirjoitus ja ohjaus” on Display Suiten avulla tehty kenttä, joka on muotoa ” Käsikirjoitus [node:field-k-sikirjoittaja], ohjaus [node:field-ohjaaja] ” missä [node:field-k-s-ikirjoittaja] korvautuu esityksen käsikirjoittajalla ja [node:field-ohjaaja] puolestaan ohjaajalla.

5.7 Esitysajankohtien ja kalenterin määritys

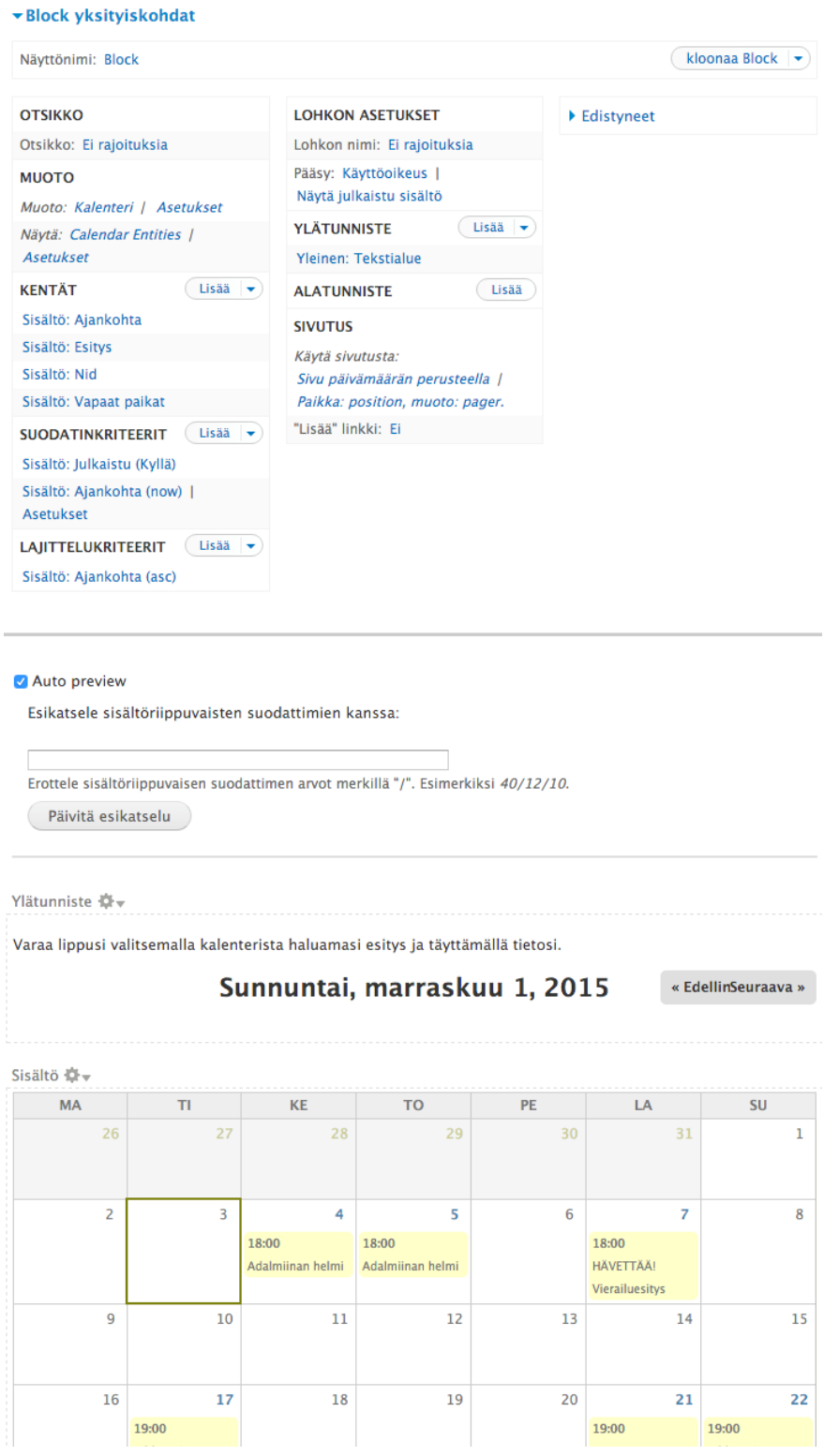
Esitysten lisäksi sivustolla tärkeä sisältö on esitysajankohdat. Esitysajankohdissa tarvittavat tiedot ovat esityksen nimi ja ajankohta, jolloin esitys pidetään. Ajankohtaa varten asensin Drupalin Date-moduulin, jonka avulla saa päivämääräkentän ja jonka apumoduulin avulla saa myös näkymiä rajattua päivämääräkenttien avulla. Esityksen nimeksi laitoin vain tekstikentän. Koska tässä sisältötyypissä ei ole varsinaista otsikkoa, eikä esityksen nimikään ole tarpeeksi yksilöivä, asensin Automatic Nodetitles-moduulin, jonka avulla muodostin sisältötyypin kahdesta kentästä otsikon. Automatic Nodetitle-moduulin avulla voi määrittää, haluaako otsikkokenttää näkyvän lainkaan, vai muodostotaanko se automaattisesti korvauskuvioita käyttäen. Tässä tapauksessa piilotin otsikkokentän ja määritin otsikon määriytyvän seuraavanlaisesti: ”[node:field_esitys] - [node:field-ajankohdat:custom:j.n.Y]”. Eli ensiksi tulee esityksen nimi, jonka jälkeen tulee ajankohta, PHP-käsikirjan mukaisesti tuossa muodossa, esimerkiksi ” Vaimoni on toista maata - 11.12.2015”.

Esitysajankohtien näyttämiseen latasin Näkymien tukena toimivan Calendar-moduulin. Calendar-moduuli loi asentaessa automaattisesti näkymämallinteita, eri päivämääräkenttien pohjalta. Tein oman näkymän siitä mallinteesta, joka oli tehty esityksen ajankohdan perusteelta. Kuvasta 15 näkyy millainen näkymä oli ennen kuin olin tehnyt siihen muutoksia; kalenteriin tulee kaikki sisällöt joissa on esitysajankohta määriteltynä. Kuvassa 16 näkyy puolestaan, että moduuli tarjoaa valmiina hyvän kalenterin, jota ei tarvitse kuin hieman tyylitellä sivuston ilmeeseen sopivaksi.



Kuva 15. Kalenterin mallinnenäkymä

Kuvasta 16 huomaa, että muutoksia on tehty: lisäsin myös kenttiä, joita en kuitenkaan tulosta näkymään, kuten kuvasta 16 näkyy. Rajasin myös näkymään vain esitysajankohdat, joiden ajankohta on tunnin tarkkuudella ”now” eli nyt, tämän rajauksen ja lisäkenttien tarpeellisuus tulee selväksi luvussa 5.8. Kuvasta 16 huomaa myös, että muutin kalenterin ulkoasua, näkymän muodon asetuksista otin pois päältä minikalenterin, jolloin kenttien tiedot näkyvät kalenterissa pelkkien numeroiden sijaan (vrt. kuva 15).



Kuva 16. Kalenterinäkymä

5.8 Varauslomakkeen teko

Varauslomakkeen tekoon latasin Drupalin Webform-moduulin, jonka avulla pystyy helposti tekemään lomakkeita. Latasin myös Webform References-moduulin, jonka avulla webform-lomakkeeseen saa kentän, jossa on kaikki sisällöt listana, valintanappeina/-ruutuina tai automaattisen täydentämisen kenttänä. Sisällöt saa rajattua tiettyyn sisältötyyppiin. Tein varauslomakkeeseen kyseisenlaisen kentän, jossa näkyvät valintalistana kaikki esitysajankohtaa olevat sisällöt. Lisäsin lomakkeeseen myös tekstikentät nimeä ja puhelinnumeroa varten, sähköpostikentän sähköpostia varten, ja kolme numerokenttää: peruslippu, opiskelija, eläkeläine… ja HARTEKA, joihin täytetään se luku, minkä verran haluaa varata kunkin hintaluokan lippuja.

Tutustuttuani Webform Reference-moduulin tarjoamaan valintalistaan paremmin, huomasin että se käyttää valintojen arvoina aina kyseisen sisällön ID:tä, eli NID:ä. Tätä tietoa pystyin käyttämään hyväksi kalenterin ja varauslomakkeen yhdistämisessä, sillä uudelleenkirjoitin kalenteri-näkymän esityksen nimen kentän seuraavanlaisesti: ” [field_esitys]”. Eli esityksen nimen ympärillä on aina span-tagi, jossa on luokkana kyseisen esitysajankohta sisällön NID. Tein kuvassa 17 näkyvät Javascript-koodaukset, jotka toimivat seuraavanlaisesti: Kun käyttäjä painaa kalenterista jotakin esitystä, niin varauslomakkeen valintalistan valittu valinta vaihtuu kyseiseksi painetuksi esitysajankohdaksi NID:n mukaan. Lisäsin lomakkeeseen perus html-koodia, jossa on tekstinä ” Valittu esitys ”. Kuten koodista näkyy kuvassa 17, niin span-tagien väliin tulee aina valintalistan valittu valinta. Näin pystyin piilottamaan valintalistan käyttäjiltä css:sää käyttäen.

```
$.fn.kalenteri = function() {
  var valinta = $(this).find('.views-field-field-vapaat-paikat span').attr('class');
  $('#edit-submitted-esitys').val(valinta);

  $('#webform-client-form-7 .webform-component--valittu-esitys p span').text($('#edit-
     submitted-esitys option:selected').text());

  $('#block-webform-client-block-7').slideDown();

  $('html,body').animate({
      scrollTop: $('#block-webform-client-block-7').offset().top - 180
  }, 1000);
};

$('.calendar-calendar .month-view .full td.single-day
   div.monthview').not('.loppu').click( function() {
     $(this).kalenteri();
});
```

Kuva 17. Kalenterin ja varauslomakkeen JavaScript-koodi

Tämän toiminnallisuuden takia kalenteri näkymästä on rajattu kaikki vanhat esitykset pois, jottei käyttäjä vahingossa varaa vanhoihin esityksiin lippuja. Koska käyttäjälle ei näy muuta varausmahdollisuutta kuin valita kalenterista ajankohta, niin tämä rajaus on riittävä, eikä tarvitse huolehtia siitä, että lomakkeen valintalistassa on oikeasti menneetkin esitysajankohdat valittavana.

5.8.1 Varausten määrän rajoitus

Tämä ei tietenkään ole vielä riittävä ratkaisu, sillä teatterisaleissa on rajattu määrä paikkoja, joten lippuja pitää pystyä varaamaan vain rajatun määrän esitysajankohtaa kohden. Tämän ongelman ratkaisemiseksi ensimmäiseksi lisäsin esitysajankohta-sisältötyyppiin numeerisen kentän ”Vapaat paikat”, johon laitoin (pienen tiedustelun jälkeen) oletusarvoksi 60. Ideana on, että aina kun tehdään uusi varaus, niin varattavien lippujen määrä vähennetään tuosta kentästä. Jos käykin niin, että vapaiden paikkojen ja varattavien lippujen määrän erotus on negatiivinen, niin varaus ei onnistu.

Tämän toiminnon rakentamiseen käytin hyväkseni melko monimutkaista mutta monipuolista Rules-moduulia jolla tein säännön, joka tarkistetaan aina kun lipun varaus –lomakkeella lähetetään vastaus. Sääntö tutkii kuinka monta lippua lomakkeen vastauksella yritetään varata ja vähentää sen kyseisen esitys ajankohdan ”Vapaat paikat” –arvosta; jos tulos on negatiivinen, tulee virhe, ettei lippuja voitu varata. Jos taas tulos on positiivinen, niin varaus onnistuu, ”Vapaat paikat”-arvo päivittyy uuteen ja varaajan sähköpostiin ja sivuston ylläpitosähköpostille lähetetään vahvistus varauksesta. Sivustolle ilmestyy myös onnistumisilmoitus.

Valitettavasti Rules-moduuli ei taipunut niin paljoa, että olisi poistanut aina lomakkeen jonka varaus ei mennyt läpi, vaan minun piti luoda lomakkeeseen ”Onko käypä” totuusarvo, jotta järjestelmässä pystyi rajaamaan kaikista tulleista lomakevastauksista ne, joiden varaus oli mennyt läpi. Tätä varten tein oman pienen moduulin, jonka koodi näkyy kuvassa 18. Kuvassa näkyvä funktio käynnistyy aina ennen kuin lomakkeen vastaus tallennetaan tietokantaan. Funktiossa tarkistetaan ensiksi, onko kyseessä halutun lomakkeen, eli varauslomakkeen, lomakevastaus. Tämän jälkeen lasketaan kaikki lomakevastauksen liput yhteen ja tehdään niistä muuttuja $varattavat_liput. $esitys-

muuttujaan puolestaan haetaan varattavan esityksen solmun ID:n avulla kyseinen solmu käyttämällä Drupalin node_load()-funktiota. Tämän jälkeen tarkistetaan, mikä on solmun tämän hetkinen vapaat paikat –arvo ja jos se on pienempi kuin varattavat liput, niin tallennetaan Onko käypä arvoksi kyseisellä lomakevastaukselle arvo ’ei’.

```php
<?php

function poista_varaus_webform_submission_presave($node, &$submission) {
    if ($submission->nid == 7) {
        $varattavat_liput = $submission->data[7][0] + $submission->data[8][0]
            + $submission->data[9][0];
        $esitys = node_load($submission->data[1][0]);

        if ($esitys->field_vapaat_paikat['und'][0]['value'] <
            $varattavat_liput) {
            $submission->data[10][0] = 'ei';
        }
    }
}
```

Kuva 18. Oman moduulin koodi

5.9 Varausten tarkastelu

Vaikkakin kaikista onnistuneista varauksista lähteekin sähköposti teatterin sähköpostiin, haluttiin myös sivustolle helppo tapa tarkastella varauksia. Tämän tähden tein näkymän, missä näkyvät kaikki lipun varaus –lomakkeen vastaukset. Lomakkeiden vastauksien näkymä ei ole aivan yhtä yksinkertainen kuin solmujen näkymät, sillä kentät lisätään kaikki ”Verkkolomakkeen lähetyksen tiedot: Arvo” kentällä, jonka asetuksista valitaan mitä kenttää kulloinkin haluaa kentän näyttävän. Kun tein rajauksen että listassa näkyvät vain ne lomakevastaukset, joiden onko käypä arvo on ”kyllä”, niin jouduin ensiksi luomaan näkymässä suhteen lomakevastauksen onko käypä –kenttään, jonka jälkeen tämän suhteen avulla pystyin vasta rajaamaan ei käyvät.

Tein näkymästä sivu-näkymän, jonka polku on varaukset/%, jossa %-merkkiä käytetään sisältöriippuvaisissa suodattimissa. Tein myös suhteen lomakevastauksen esityskenttään, joka on siis aina kyseisen esitysajankohdan solmun ID, ja tätä suhdetta käyttämällä tein sisältöriippuvaisen suodattimen. Laitoin niin, että jos polussa ei ole suodatinarvoa (eli polku on tässä tapauksessa varaukset), niin näkymän kaikki tulokset tulevat näkyviin. Puolestaan jos polku on varaukset/37, niin sivulla näkyvät vain ne varaukset jotka on varattu esitysajankohtaan, jonka solmun ID on 37.

Jotta näihin sisältöriippuvaisen suodattimen suodattamille sivuille pääsisi helposti, tein vielä yhden näkymän. Näkymässä yksinkertaisesti listataan kaikki esitysajankohdat,

joiden ajankohta on nyt tai myöhemmin. Näkymässä käytetään kahta kenttää, otsikkoa ja solmun ID:tä. Solmun ID:n piilotin näkymästä, ja otsikon uudelleenkirjoitin linkiksi, jonka polku on varaukset/[nid]. Näin tämän näkymän tuloksista pääsee katsomaan aina kunkin esityksen varauksia. Tämän sivunäkymän lisäsin admin-valikkoon, jotta sen helposti kirjautuneena löytää.

6 SIVUSTON JULKAISU

Sivuston julkaisu oli hyvin helppo projekti, piti vain siirtää sivusto sandbox-kansiosta public_html-kansioon. Käytin Drupalin ytimessä olevia pakkausmenetelmiä myös hyväkseni, joka siis tiivistää kaikki css- ja js-tiedostot, sekä myös laittaa sivuja välimuistiin. Näin käytössä oleva sivu on loppukäyttäjälle kevyempi käyttää.

6.1 Käyttäjätunnukset

Sivustolle tarvisi tehdä kahden tyyppisille käytäjille tunnukset; käyttäjille, jotka lisäävät sivustolle sisältöä ja katsovat varaukset ja toisille, jotka vain katsovat varauksia. Käytin ensimmäisten tekoon valmista käyttäjäroolia Sisällön tuottaja, jolle annoin oikeuden lisätä, poistaa ja muokata esityksiä ja esitysajankohtia. Toisen käyttäjäroolin loin itse, nimeltään Lippujen tarkastaja. Molemmille annoin oikeudet tarkastella lipun varaus –lomakkeen vastauksia ja Varaukset-sivuja.

Estin Varaukset-sivun pääsyn kirjautumattomilta henkilöiltä, etteivät sähköpostit esimerkiksi ole kaikkien saatavilla. Tein tämän kyseisen näkymän asetuksista, rajasin sivun pääsyn roolin mukaan, niin että käyttäjäroolin pitää olla administor, Sisällön tuottaja tai Lippujen tarkastaja. Muilla rooleilla sivulle yrittävät saavat vain virhesivun jossa lukee ” Sinulla ei ole käyttöoikeuksia tälle sivulle.”.

7 LOPUKSI

Projekti meni kaiken kaikkiaan yllättävänkin mukavasti, johtuen varmasti osaksi siitä, että oli vain toinen ihminen, jonka kanssa teki töitä ja hänkin antoi melko vapaat kädet. Toisaalta tämä vapaus toi mukanaan myös vastuita, sillä jouduin aivan itse selvittämään kaikki tekniset ongelmat, eikä ihan pienellä Drupal-osaamisella olisi tästä projektista tullut oikein mitään.

Projektin alussa annetut määritykset toteutuivat hyvin, esitysten ja esitysajankohtien lisääminen on helppoa teatteriväelle. Myöskään ulkoasusuunnitelmasta ei tarvinnut poiketa, vaan se pysyi hyvin sellaisena kuin millaiseksi se päätettiin.

Jälkeenpäin ajatellen, rakentaisin varauksen vastaanoton ja hylkäämisen jotenkin eritavalla, tekisin varmaankin siitä kokonaan oman moduulinsa, enkä käyttäisi Rules-moduulia ollenkaan. Eniten jäi vaivaamaan tuo ratkaisu, että varaus merkitään ”ei käyväksi”, ellei se mene läpi; olisin halunnut että varaus poistuu kokonaan.

Myös Boostrapin käyttö jäi vähän vajavaiseksi, ja nyt kun olen sitä enemmän käyttänyt, näen kuinka paljon paremmin sivuston olisi Bootsrapia voinut käyttää. Vaikka sivu onkin responsiivinen ja esimerkiksi Googlen hakutuloksissa sivusta lukee ”mobiililaitteisiin sopiva”, niin Boostrap olisi tarjonnut itsessään helpompia ja parempia ratkaisuja. Toisaalta myös opin tämän projektin aikana paljon Boostrapisa, kun minulle ei ennen tätä ollut oikein tietoa sen toiminnasta.

Olisin voinut myös perehtyä paremmin Drupalin muihin lomake-moduuleihin kuin Webforms, esimerkiksi toiseen suosittuun lomakkeiden tekemiseen tarkoitettuun moduuliin Entity Forms. Lomake joka sivustolla on toteutettu, ei kuitenkaan ole ihan perinteinen lomake, joten olisi ollut järkevämpää tutkia olisiko jokin toinen moduuli ollut tilanteeseen sopivampi. Koko lomakkeen olisi myös voinut toteuttaa kenties omalla moduulilla, mutta tähän eivät olisi välttämättä taidot projektin toteutushetkellä riittäneet.

Jatkossa joudun varmasti sivustolle tekemään pieniä parannuksia, ja ehkä isompiakin kun taidot vielä paranevat ja pystyy tekemään vielä parempia ratkaisuja. Sivustolle

halutaan vielä myös pieniä parannuksia, esimerkiksi että ison taustakuvan (näkyy kuvan 4 ulkoasusuunnitelmassa) vaihto onnistuu helposti sisällönhallintajärjestelmän kautta. Myös Drupal-päivitykset jäävät minun huolehdittavakseni, sillä aina päivityksiä tehdessä on pieni riski, että tuleekin jotakin versioiden yhteensopimattomuusvirheitä, joiden korjaamiseen tarvitaan jo Drupal-tuntemusta.

Kuitenkin sivujen ylläpito on teatterin puolelta suhteellisen itsenäistä. Anan uusien esitysten ja esitysajankohtien tullessa, saavat he itse lisättyä ne sivustolle. Tämä on hyvä molempien, sekä minun että teatteriväen, kannalta; minun ei tarvitse kuluttaa aikaani tälle sivustolle enää juuri ollenkaan ja heidän sivuston sisältöjen ajankohtaisuus ei ole riippuvainen minun aikataulustani.

LÄHTEET

Brown, J., Hunter, L., Tracy, C. & Daglees J. 2014. Organizing content with taxonomies. Viitattu 24.3.2015. https://www.drupal.org/documentation/modules/taxonomy

Carlile, K., Chopra, N., deeparandad & Singh, M. 2015. Working with Menus. Viitattu 29.3.2015. https://www.drupal.org/documentation/modules/menu

ccgearig, kalamityjane, Young, M. & RonnieRayB. 2014. The Drupal overview. Viitattu 21.3.2015. https://www.drupal.org/getting-started/before/overview

ccgearig, phulebar, Guleria, N. & Doesborg, B. 2014. About nodes. Viitattu 24.3.2015 https://www.drupal.org/documentation/modules/node

ccgearig, Wilson, D. & Ryan, M. 2014. User Roles. Viitattu 29.3.2015. https://www.drupal.org/node/1803614

Choudhury, S., Brown, J., Gutierrez, S. & Liskey, R. 2015. Working with content types and fields (Drupal 7 and later). Viitattu 24.3.2015. https://www.drupal.org/documentation/modules/field-ui

Choudhury, S., Feidt, D., Foley, D. & Ringwood, S. 2015. Working with nodes, content types and fields. Viitattu 24.3.2015 https://www.drupal.org/node/717120

dags, James, H., Maat, E., Singh M. 2014. Working with blocks (content in regions). Viitattu 29.3.2015. https://www.drupal.org/documentation/modules/block

Dcdenison, Brown, J., phulebar & Gerndt, L. 2015. General consepts. Viitattu 24.3.2015. https://www.drupal.org/node/19828

Gruber, E., Bah, T., Matthews, P. & Tadisetty H. 2014. Differentiating between kinds of ”users”. Viitattu 29.3.2015. https://www.drupal.org/node/660468

James, H., dustise, Rana, P. & drupalshrek. 2015. Overview of theme files. Viitattu 24.3.2015. https://www.drupal.org/node/171194

Matthews, P., Hunter, L., Curtain, K. 2013. Add a vocabulary to a content type. Viitattu 24.3.2015. https://www.drupal.org/node/1886980

Mercer, D. 2010. Drupal 7. Packt Publishing.

Needham, D., Riemenschneider, H., Hannemyr, G. & Rosenfeld, L. 2013. Viitattu 29.3.2015. https://www.drupal.org/node/171205

Pachabhaiya, C., Feidt, D., Foley, D. & Guo, L. 2015. Writing theme .info files. Viitattu 29.3.2015. https://www.drupal.org/documentation/modules/views

phulebar, tvn, Hunter, L. & Nordhaug, H. 2015. Understanding Drupal. Viitattu 21.3.2015. https://www.drupal.org/documentation/understand

Redding, J. 2010. Beginning Drupal. Hoboken: Wrox

SERVANT14, Hunter, L., crawleyde, Tracy, C. 2013. About taxonomies. Viitattu 24.3.2015. https://www.drupal.org/node/774892

Shreves, R. 2011. Drupal 7 Themes. Packt Publishing

snehi, Kumar, K., Bah, T. & Doesborg, B. 2015. Content types. Viitattu 21.3.2015. https://www.drupal.org/node/21947

Tolboom, C., McGuire, C. & Wachter, K. 2015. What is Views? Viitattu 29.3.2015. https://www.drupal.org/node/2287909

Tomlinson, T. & VanDyk K. 2010. Pro Drupal 7 Development: Third Edition. Apress.

Walc, W. 2009. CKEditor - WYSIWYG HTML editor. Viitattu 26.8.2015. https://www.drupal.org/project/ckeditor