Tuomas Kämppi

RESPONSIIVISTEN TEEMOJEN SUUNNITTELU JA TOTEUTUS LIFERAY-JULKAISUJÄRJESTELMÄÄN

Case: Obshare.com-käyttöliittymätoteutus

Opinnäytetyö
Tietojenkäsittelyn koulutusohjelma

Toukokuu 2013

MIKKELIN AMMATTIKORKEAKOULU
Mikkeli University of Applied Sciences

KUVAILULEHTI

MIKKELIN AMMATTIKORKEAKOULU Mikkeli University of Applied Sciences	**Opinnäytetyön päivämäärä** **31.5.2013**
Tekijä(t) Tuomas Kämppi	**Koulutusohjelma ja suuntautuminen** **Tietojenkäsittelyn koulutusohjelma**

Nimeke

Responsiivisten teemojen suunnittelu ja toteutus Liferay-julkaisujärjestelmään Case: obshare.com käyttöliittymätoteutus

Tiivistelmä

Opinnäytetyön tavoitteena on suunnitella ja toteuttaa Liferay-julkaisujärjestelmän teemoitus Observis Oy:n obshare.com-palveluun responsiivisen verkkosuunnittelun periaatteella. Tavoitteena on, että palvelun käyttöliittymä vastaa graafikon ja suunnittelijoiden näkemyksiä ja sen käyttö on sujuvaa myös puhelimien ja tablettien selaimilta selailtuna.

Opinnäytetyön kehittämistehtävänä on perehtyä Liferayn teemojen toteutukseen ja kuinka palvelun sisäiset ohjelmat, portletit, saadaan vastaamaan teeman ulkoasua. Lisäksi tehtävänäni on suunnitella ja toteuttaa obshare.com-palvelun teema responsiivisen verkkosuunnittelun periaatteita hyödyntäen. Tietoa on hankittu Liferay-julkaisujärjestelmän virallisista ohjekirjoista, kuten Liferay User Interface Development ja Liferay In Action, sekä Liferayn kotisivuilta. Lisäksi responsiivisen verkkosuunnittelun lähteenä on käytetty Liferayn User Interface Engineer Nate Cavanaughin blogikirjoituksia sekä Ethan Marcotten vuonna 2010 ilmestynyttä kirjaa Responsive Web Design. Lisäksi haasteena on ollut toteutuksessa käytettyjen Primefaces-elementtien muokkaaminen teemaan ja mobiililaitekäyttöön sopiviksi.

Työ on toteutettu Observis Oy:n kansainvälisessä ohjelmointi- ja suunnittelutiimissä. Käytännön työ on tehty suurimmaksi osaksi yrityksen toimipisteessä ja työkielenä on ollut pääasiassa englanti. Tehtävänäni on ollut toteuttaa obshare.com-verkkopalvelun teema graafikon ohjeiston ja suunnittelutiimin ohjeiden mukaisesti. Lisäksi vastuulleni kuuluu ohjelmoijien tuottamien käyttöliittymäelementtien muokkaaminen tähän teemaan sopiviksi.

Opinnäytetyön alussa on esitelty käytännön työssä käytetyt responsiivisen verkkosuunnittelun ja Liferay-julkaisujärjestelmän tekniikat sekä niiden perusajatukset. Lopussa on käyty läpi teeman käytännön toteutus sekä esitelty obshare.com-palvelun havaintosivujen listanäkymän toteutus esimerkkinä portletin käyttöliittymän responsiivisesta toteutuksesta.

Asiasanat (avainsanat)

CSS3, HTML5, käytettävyys, julkaisujärjestelmät, mobiililaitteet

Sivumäärä	**Kieli**	**URN**
47	Suomi	

Huomautus (huomautukset liitteistä)

Ohjaavan opettajan nimi	**Opinnäytetyön toimeksiantaja**
Jukka Selin	Observis Oy

DESCRIPTION

MIKKELIN AMMATTIKORKEAKOULU Mikkeli University of Applied Sciences	**Date of the bachelor's thesis** **31 May 2013**
Author(s) Tuomas Kämppi	**Degree programme and option** Business Information Technology

Name of the bachelor's thesis

Theme creation in Liferay Portal through responsive web design

Abstract

This bachelor's thesis was about creating a theme for Observis Oy's product Obshare.com which was created on top of Liferay Portal web platform. The development task was to create a Liferay theme according to the user-interface designs and graphic design manuals. The user interface of the portal should also be designed for easy browsing with handheld devices.

The purpose of the development task was to learn Liferay's theme creation methods and to produce a theme for the Obshare.com service by using responsive web design. The task also included learning the methods and producing the correct look and feel for the portal's internal applications, called portlets. The instructions and guidelines for the theme creation were gathered from the official Liferay manuals such as Liferay User Interface Development and Liferay In Action. The guides and examples for responsive web design were from Ethan Marcotte's book titled Responsive Web Design and from blog-posts by Nate Cavanaugh, a Liferay's User Interface Engineer.

The development was done in an international development team at Observis Oy. Development team includes back end developers, designers, a graphical designer and me as a front end developer. My responsibility in the development team was to produce the look and feel of the obshare.com service according to the plans of the designers.

The thesis started with the introduction and concepts of responsive web design and Liferay Portal used in the development task. The rest of this thesis was about the development process. This included an example of theme creation and the use of responsive web design in the Obshare.com observation page. The development of the user interface of Obshare.com service continues with the knowledge gained from this work.

Subject headings, (keywords)

CSS3, HTML5, usability, content management systems, mobile devices

Pages	**Language**	**URN**
47	Finnish	

Remarks, notes on appendices

Tutor	**Bachelor's thesis assigned by**
Jukka Selin	Observis Oy

SISÄLTÖ

LIITTEET

1 JOHDANTO

Tämän opinnäytetyön aiheena on verkkopalvelu obshare.com käyttöliittymätoteutus responsiivisen verkkosuunnittelun periaatteella. Työssä esittelen toteutukseen käytetyt tekniikat ja niiden käytännön sovellukset. Opinnäytetyön tilaajana on Observis Oy.

Observis Oy on vuonna 2011 alusta toimintansa aloittanut mikkeliläinen verkko- ja mobiilipalveluita tuottava yritys. Yritys teettää pilvipalveluna työkaluja yrityksille sekä mobiili-, että verkkosovelluksina. Lisäksi yrityksen tuotteisiin kuuluvat erilaiset tapahtumasovellukset, kuten esim. JurassicApp 2012.

Opinnäytetyöni liittyy laajempaan kokonaisuuteen, Observis Oy:n kehitteillä olevaan tuotteeseen obshare.com. Obshare.com on yhteisöpalvelu, jonka tarkoitus on toimia vuorovaikutteisena viestintäkanavana yksityisten henkilöiden, järjestöjen, valtionhallinnon, yritysten, tutkimuslaitosten ja oppilaitosten välillä. Palveluun kuuluu mm. ympäristön havainnointi, jota on testattu jo Etelä-Savon pelastuslaitoksen Talvimyrsky-valmiusharjoituksessa. Palvelun käyttäjä voi lähettää mobiilisovelluksella havainnon ympäristöstä, joka vaatii esim. viranomaisten huomiota. Havainto sisältää mm. kuvan, kuvauksen, sijainnin ja havaintoajan. Lisäksi organisaatiot voivat toteuttaa yhteiskuntavastuutaan luomalla projekteja, esimerkiksi lähialueen joen puhdistushankkeen.

Idea työhön lähti kesällä 2012 ollessani syventävässä työharjoittelussa Observis Oy:n Summer Trainee 2012 -ohjelmassa. Esittelin yritykselle ajatukseni opinnäytetyöstä, jonka aiheena olisi responsiivinen verkkosuunnittelu. Yritys esitteli minulle kehitteillä olevan tuotteensa obshare.com, jonka toteutuksen pohjana käytetään Java-ohjelmointikieleen perustuvaa Liferay-julkaisujärjestelmää. Työtä tehdään kansainvälisessä tiimissä, yhdessä kolmen ohjelmoijan, graafikon ja kahden suunnittelijan kanssa. Minun osa-alueekseni kuuluu olla mukana suunnittelemassa palvelun käyttöliittymää ja toteuttaa käyttöliittymän ulkoasun teemoitus responsiivisen verkkosuunnittelun periaatteella.

Koska Observis Oy:n tuote obshare.com on hyvin laaja ja monisivuinen verkkopalvelu, keskityn pelkästään palvelun havaintosivujen käyttöliittymän toteutukseen. Sivuston muiden sivujen toteutuksessa noudatetaan samoja tässä

opinnäytetyössä esiteltyjä periaatteita. Opinnäytetyön tavoitteena on oppia toteuttamaan Liferayn teemoja ja oppia hallitsemaan toteutuksessa käytettyjä kehitysvälineitä ja kehitysympäristöä. Lopullisena tavoitteena on toteuttaa Liferayn teema, joka noudattaa responsiivisen verkkosuunnittelun periaatteita.

Opinnäytetyöni kehittämistehtävänä on suunnitella ja toteuttaa teema Liferay-julkaisujärjestelmän päällä toimivan obshare.com-verkkopalvelun käyttöliittymälle. Suunnittelua ja kehitystä tehdään yhdessä kolmen ohjelmoijan, graafikon ja useamman palvelun suunnittelijan kanssa. Jokainen kehitystiimin jäsen osallistuu jossain määrin palvelun toteutuksen suunnitteluun projektin aikana. Ohjelmointipuoli pääsääntöisesti suunnittelee toteutustavan ja suunnittelupuoli suunnittelee, mitä ja minkä näköisiä ominaisuuksia palveluun kuuluu. Minun tehtävänäni on toteuttaa verkkopalvelun ulkoasun teema graafikon graafisen ohjeiston ja suunnittelijoiden ohjeiden mukaisesti. Teemaan kuuluu myös ohjelmoijien toteuttamien käyttöliittymäkomponenttien ulkoasujen muokkaaminen teemaan sopiviksi.

Luvussa kaksi esittelen lyhyesti responsiivisen verkkosuunnittelun perusajatuksen ja perusteet sen käyttämiselle verkkopalvelun teeman toteutuksessa. Luvussa esitellään myös käytännön toteutuksessa käytetyt responsiivisen verkkosuunnittelun tekniikat. Kolmannessa luvussa esittelen lyhyesti Liferay-julkaisujärjestelmän, perusteet sen käytölle, sekä käsitteet, jotka ovat keskeisiä teeman toteutuksessa. Luvussa neljä käyn läpi obshare.com-verkkopalvelun kehitystiimin, työskentelyprosessin ja teeman toteutuksen. Päätäntöluvussa teen yhteenvedon opinnäytetyöstä.

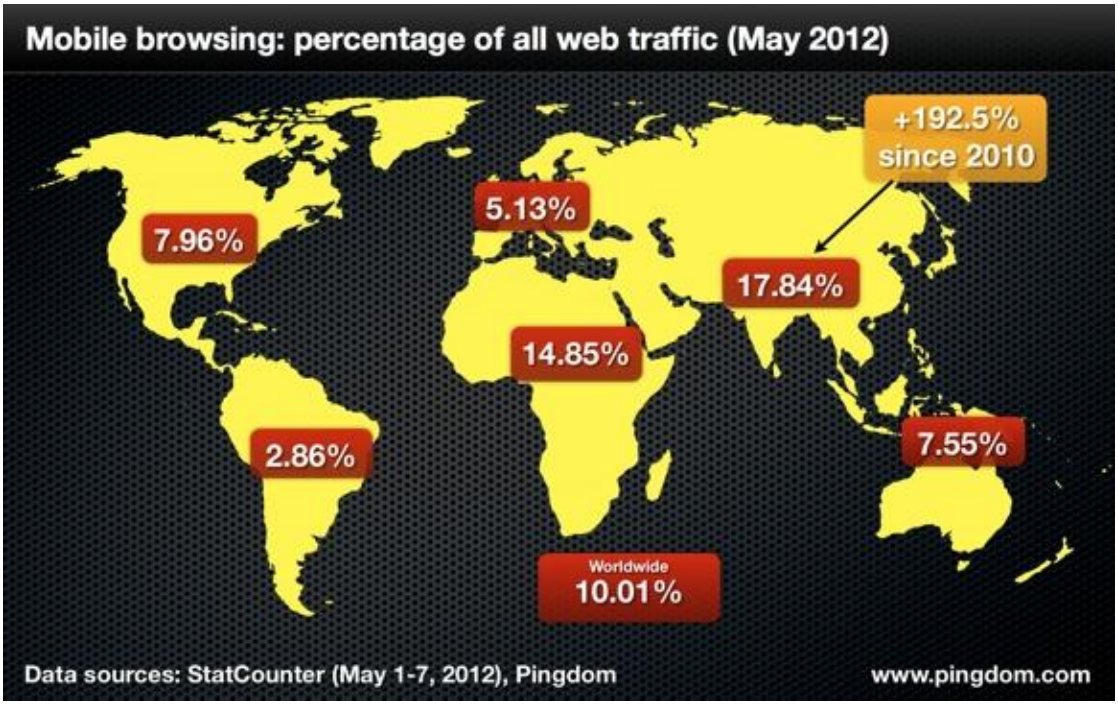
2 RESPONSIIVINEN VERKKOSUUNNITTELU

Tässä luvussa esittelen lyhyesti responsiivisen verkkosuunnittelun perusajatuksen ja perusteet sen käytölle teeman toteutuksessa. Lisäksi esittelen käytännön toteutuksessa käytettyjä responsiivisen verkkosuunnittelun tekniikoita.

Responsiivisella verkkosuunnittelulla tarkoitetaan eri näyttökokoihin mukautuvien verkkosivujen suunnittelua. Tarkemmin sanottuna verkkopalveluiden suunnittelussa otetaan huomioon sekä pienten mobiililaitteiden näyttökoot, kuin myös isommat 24-tuumaiset laajakuvanäytöt. Responsiivisella verkkosuunnittelulla pyritään esittämään

sama verkkosivujen sisältö erikokoisten laitteiden selaimilla mahdollisimman luontevasti. Suunnittelussa otetaan lisäksi huomioon kosketusnäytöt ja niiden haasteet.

Nykypäivänä yhä useammalla meistä on taskussamme laite, joka sisältää nettiselaimen. Danyl Bosomworth www.smartsights.com:sta kertoo artikkelissaan, että kolmen vuoden kuluessa mobiililaitteet tulevat ohittamaan perinteiset tietokoneet Internetin käyttäjien keskuudessa (Bosomworth 2013). Suurin kasvu tällä hetkellä on royal.pingdom.com:n mukaan Aasiassa, missä mobiililaitteiden Internetin käyttö on kolminkertaistunut vuodesta 2010 vuoden 2012 toukokuuhun, kuten kuvassa 1 on esitetty (Pingdom 2012).



KUVA 1. Mobiilisurffauksen prosenttiosuus kaikesta verkkoliikenteestä toukokuussa 2012 (Pingdom 2012)

Mobiililaitteiden lisääntynyt määrä Internetin käytössä ei näy kuitenkaan suurimmassa osassa verkkopalveluista. Michael Martin kertoo kolumnissaan, että vain alle 10 % maailman verkkopalveluista on räätälöity käytettäväksi mobiililaitteiden pienillä näytöillä. Yritysten markkinoinnin kannalta sivujen mobiilioptimointi on tärkeää. Adoben tutkimuksessa (Mobile Shopper Insights for 2011) todettiin, että huonosti mobiililaitteille räätälöidyt verkkokaupat karkottavat asiakkaat toisille, paremmin suunnitelluille sivuille. (Martin 2012.)

Nykyään yhä enemmän ihmiset tekevät hakuja mobiililaitteillaan. Samuli Kotilainen kirjoittaa Tietokone-lehden artikkelissa *”Googlea lähestytään yhä useammin mobiilisti”*, että ihmiset eivät jaksa kirjoittaa verkkopalvelun koko osoitetta osoiteriville, vaan mieluiten hakeutuvat niihin hakukoneiden kautta. Samassa artikkelissa Kotilainen kertoo myös, että Suomessa tehdään jo yli kolme miljoonaa hakua mobiililaitteilla päivittäin. (Kotilainen 2013.)

Keskeisintä responsiivisessa verkkosuunnittelussa on eri näyttökokojen tunnistaminen. Responsiivisuuteen kuuluu myös eri selainten sekä kosketusnäyttöjen huomioon ottaminen. Tässä luvussa esittelen tämän opinnäytetyön toteutuksessa näyttökokojen ja selainten tunnistamiseen käytetyt tekniikat: **CSS3** ja **JavaScript** sekä ehdollisen HTML-kommentin, jota käytetään Internet Explorerin ja sen eri versioiden tunnistamisessa.

2.1 CSS3

Koska pelkkä CSS-mediatyyppi *”handheld”* ei riitä kattamaan kaikkia eri kannettavien laitteiden näyttökokoja, on ratkaisuksi tuotu CSS3:n mediakyselyt. Mediakyselyt koostuvat kahdesta osasta jotka on näytetty alla kuvassa 2.

```
2
3  @media screen and (min-width: 1024px) {
4  /* Tänne tulee CSS, joka halutaan toteuttaa */
5    #container {
6        width: 960px;
7    }
8  }
```

KUVA 2. Esimerkki mediakyselystä

Ensimmäinen osa on CSS2 mukainen *”@media screen”*-sääntö, jonka jälkeen tulee ehtokysely: *”(min-width: 1024px)”*. Selaimelta siis kysytään, onko selailuun käytettävän laitteen selaimen näyttöalueen koko vähintään 1024 pikseliä. Jos selain vastaa kyselyyn myöntävästi, toteutetaan ehdon sisältämä sääntö. Ehdon voi myös sisällyttää HTML *”<link>”*-elementtiin, jolloin halutulle näyttökoolle voidaan osoittaa haluttu CSS-tiedosto. Mediakyselyillä voidaan kysyä myös, onko selaimen laite pysty- vai vaakasuorassa. Koko lista selaimilta kysyttävistä ominaisuuksista ovat taulukossa 1.

TAULUKKO 1. Lista ominaisuuksista (Marcotte 2010, 76–78)

Ominaisuus	**Määritelmä**	**Minimi- ja maksimiarvot**
width	Selaimen näyttöalueen leveys	Kyllä (esim. max-width, min-width)
height	Selaimen näyttöalueen korkeus	Kyllä
device-width	Laitteen piirtoalueen leveys	Kyllä
device-height	Laitteen piirtoalueen korkeus	Kyllä
orientation	Laitteen pysty- tai vaaka-asento. Arvot portrait tai landscape	Ei
aspect-ratio	Voidaan kysyä, onko selaimen kuvasuhde esim. 16:9	Kyllä
device-aspect-ratio	Voidaan kysyä, onko laitteen näytön kuvasuhde esim. 16:9	Kyllä
color	Voidaan kysyä, onko laitteen näytön väritoisto esim. 8-bittinen. Esim. @media screen and (color: 8).	Kyllä
color-index	Esim. @media screen and (min-color-index: 256).	Kyllä
monochrome	Samankaltainen kuin color	Kyllä
resolution	Esim. @media screen and (resolution: 72dpi) tai @media screen and (max-resolution: 300dpi).	Kyllä
scan	Tv selailuun	Ei
grid	Yksinkertaisesti @media screen and (grid).	Ei

Mediakyselyiden ehtolauseita voidaan myös yhdistellä kuvan 3 tavalla käyttämällä *”and”*-operaattoria.

```
@media screen and (min-device-width: 480px) and (orientation: landscape) {
/* Tänne tulee CSS, joka halutaan toteuttaa */
  #container {
      width: 100%;
  }
}
```

KUVA 3. Esimerkki yhdistetystä mediakyselystä

Testaamiseen riittää alkuun selaimen ikkunakoon muuttaminen, mutta jos mahdollista, testailuun kannattaa pitää käsillä ainakin yleisimmät puhelinmallit ja tabletit, koska pelkän selaimen koon muuttaminen ei välttämättä kerro koko totuutta.

2.2 JavaScript

JavaScriptillä voidaan simuloida CSS3 mediakyselyitä. Liferayssa sama on toteutettu Jonathan Nealin toteuttamalla JavaScriptillä. Liferayn AlloyUI-kirjastoon liitetty JavaScript-moduuli **aui-viewport** tutkii joka kerta selaimen latautuessa, tai sen koon vaihtuessa selaimen piirtoalueen koon pikseleinä (px). Etu JavaScriptin käytössä CSS3-mediakyselyiden sijaan on sen varmatoimisuus vanhemmillakin Internet Explorerin selainversioilla. (Cavanaugh 2011.)

JavaScriptillä voidaan tuoda myös uutta sisältöä selaimille. Esimerkiksi vanhemmat versiot Internet Explorerista eivät tue kaikkia HTML5-elementtejä. Microsoft on kuitenkin kehittänyt selaimilleen ainutlaatuisen tavan tunnistaa vain niille tarkoitettua sisältöä. Ehdollinen kommentti on HTML-kommentti, jonka avulla Internet Explorer –selaimille voidaan tuoda omaa sisältöä (kuva 4).

```
        <!--[if IE]> Vain Internet Explorerille tarkoitettua sisältöä! <![endif]-->
```

KUVA 4. Ehdollinen kommentti

Ehdolliseen kommenttiin on tässä opinnäytetyössä sijoitettu JQuery JavaSript-kirjastolla toteutettu tiedosto **html5shiv.js**, joka toteuttaa kaikki HTML5:n uudet ominaisuudet. Lisäksi ehdollisella kommentilla voidaan kysyä eri Internet Explorerin versioita ja ohjata niille omat sisällöt. Tässä työssä kaikki Internet Explorerin versiot 9

alaspäin saavat lisäksi **IE9.js**-tiedoston, joka korjaa useimpia Internet Explorerin HTML ja CSS ongelmia, sekä lisää läpinäkyvien PNG-kuvien tuen myös Internet Explorerin versioille 5 ja 6 (kuva 5).

```
20
21      <!--[if IE]>
22          <script src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>
23      <![endif]-->
24      <!--[if lt IE 9]>
25          <script src="http://ie7-js.googlecode.com/svn/version/2.1(beta4)/IE9.js"></script>
26      <![endif]-->
27
```

KUVA 5. JavaScript-kirjastojen tuominen IE:lle ehdollisen kommentin avulla

Tämän jälkeen uusimmilla HTML5-tekniikoilla toteutetut sivut toimivat todennäköisemmin myös vanhemmilla Internet Explorerin versioilla. Mallia ehdollisen kommentin käytöstä on otettu Wasim Shaikh:n Tech Blog:ssa ladattavissa olevasta responsiivisesta Liferayn versiolle 6.1 tehdystä esimerkkiteemasta: **preview-html5-responsive-theme.war**. (Shaikh 2011.)

2.3 Responsiivinen sisältö

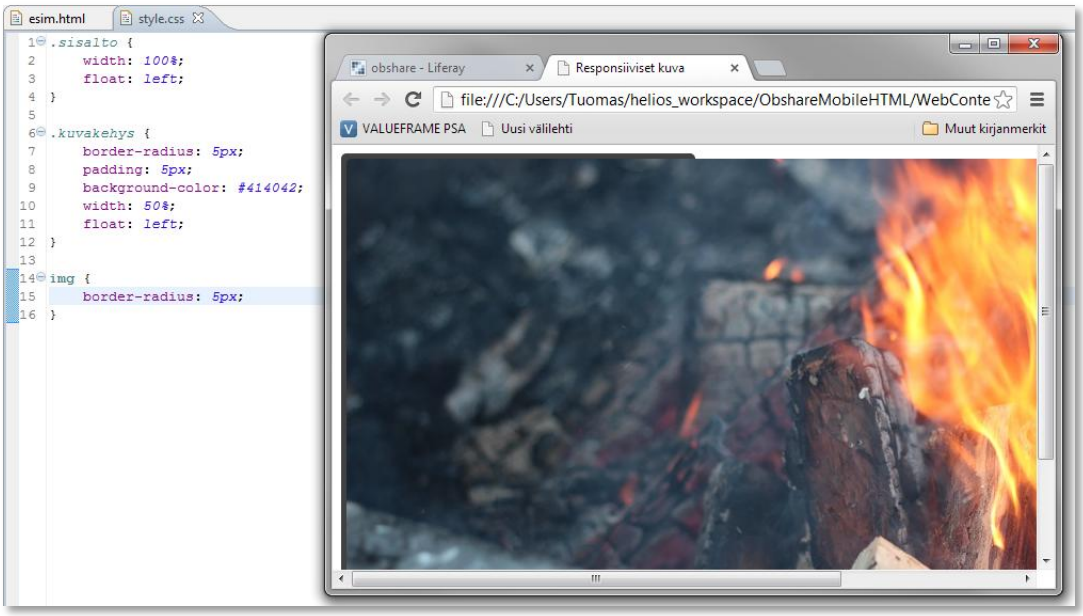
Keskeinen osa sivuja on ulkoasun lisäksi tietenkin itse sisältö. Kiinteiden, epämukautuvien sivujen sisältö on usein tehty tarkkojen pikselikokojen (px) mukaan. Esimerkiksi 200px leveään sivupalkkiin tehtiin 200px leveä kuva. Nykyisin eri näyttökokoihin mukautuvien sivujen yhtenä kehityshaasteena ovat usein leveyttä vaihtelevien elementtien reunojen yli vuotavat kuvat ja tekstit.

Mukautuvat kuvat

Kuvien vuotaminen elementtien yli on yksi ratkaistavista haasteista näyttökokoihin mukautuvien sivustojen toteutuksessa. Näyttöjen koon pienentyessä sivuston kuvienkin on pienennyttävä, säilyttäen silti oikeat mittasuhteet suhteessa koko sivustoon. Esimerkkinä toteutettakoon puolet sisällön tilasta vievä kuvakehys, jonka sisään sijoitetaan 884px x 496px kokoinen kuva (kuva 6). Isolla näytöllä kuva pysyy kehyksen sisäpuolella, mutta selainta pienennettäessä kuva alkaa vuotaa kuvakehyksen yli (kuva 7).

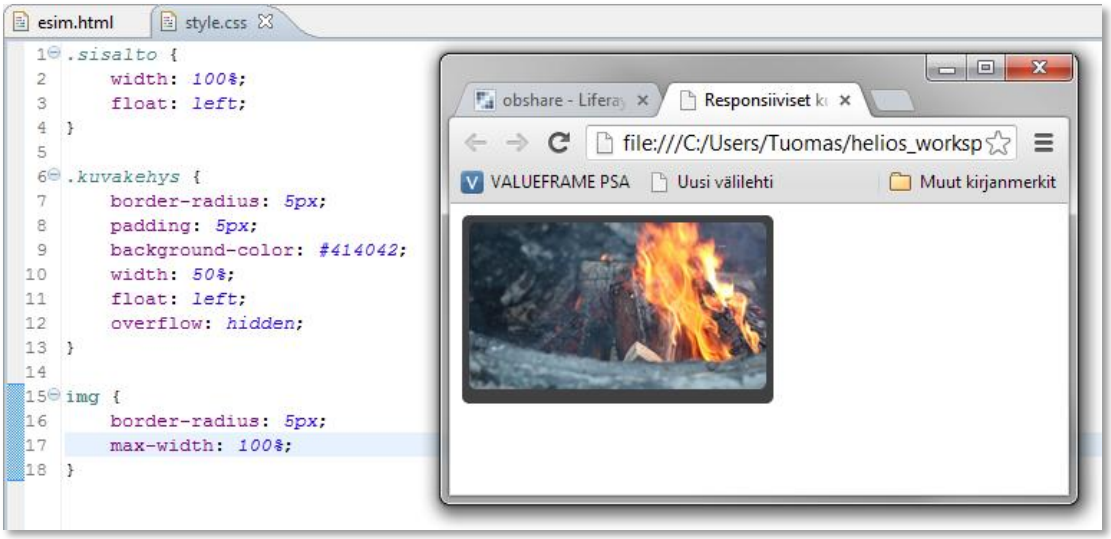
```
11    <div id="sisalto">
12        <div class="kuvakehys">
13            <img src="esim.jpg" alt="">
14        </div>
15    </div>
```

KUVA 6. Kuvaesimerkin HTML-rakenne



KUVA 7. Kuvaesimerkin ylivuotava kuva

Kuten kuvasta 7 näkyy, 884px leveä kuva vuotaa täysin sitä ympäröivän kuvakehyksen yli ja peittää koko muun näkyvän tilan. Lisäämällä kuvan 7 näkyvään *”<img>”*-elementin CSS-tyylin sisään: *”max-width: 100%;”* saadaan kuva mahtumaan kuvakehyksen sisään (kuva 8).



KUVA 8. Korjattu skaalautuminen

Sen lisäksi, että lisäämällä *”max-width: 100%;”*, saadaan kuvan leveys sopimaan sitä ympäröivään kuvakehykseen, osaavat modernit selaimet myös säilyttää kuvan oikean kuvasuhteen. Ylivuodon ehkäisemisen varmistamiseksi voidaan kuvakehykseen lisätä vielä *”overflow: hidden;”*, joka piilottaa elementin yli vuotavan sisällön.

Mukautuva teksti

Yksi ratkaistavista haasteista oli vaihtelevan pituisen tekstin tulostamisen näyttäminen niin, että teksti ei vuoda missään tapauksessa mistään yli, eivätkä sanat katkea ikävästi keskeltä. Tekstin ylivuotamisen näkymisen pystyy estämään lisäämällä sen ympäröivän elementtiin sama *”overflow: hidden;”* , kuten kuvien kanssa (kuva 9).

```
3  #container {
4      overflow: hidden;
5  }
```

KUVA 9. Elementin sisällön ylivuotamisen estäminen CSS:llä

Tämä estää elementin yli vuotavan tekstin näkymisen elementin ulkopuolella, mutta luettavuuden kannalta tämä ei ole hyvä ratkaisu, sillä teksti menee vieläkin elementin yli, mutta sitä ei enää näy ollenkaan.

Kenneth Auchenberg kertoo kehittämänsä ratkaisun tähän ongelmaan blogissaan: *Word wrapping/hyphenation using CSS* (kuva 10).

```
5  -ms-word-break: break-all;
6       word-break: break-all;
7
8  /*       // Non standard for webkit */
9       word-break: break-word;
10
11 -webkit-hyphens: auto;
12    -moz-hyphens: auto;
13         hyphens: auto;
```

KUVA 10. Jokaisella selaimella toimiva tekstimuotoilu (Auchenberg 2012)

Lisäämällä edellä olevat säännöt saadaan jokaisella selaimella toimivat määritykset tekstin näyttämiseen siten, että ne eivät katkea keskeltä tai jatku reunojen yli. (Auchenberg 2012.)

3 LIFERAY JULKAISUJÄRJESTELMÄ

Tässä luvussa esittelen yleisesti Liferay-julkaisujärjestelmän, käyttöperusteet ja peruskäsitteet, jotka ovat olleet olennaisia palvelun teeman toteutuksessa.

Liferay Portal on Java EE:n pohjautuva open-source julkaisujärjestelmä. Sitä voidaan käyttää esimerkiksi kirjautumista vaativien, laajojen verkkopalveluiden ja yritysten intranettien toteuttamisen alustana. Sovellus koostuu kehysohjelmasta, johon voidaan liittää useita pienempiä verkkosovelluksia, eli portletteja. Samankaltainen julkaisujärjestelmä on mm. PHP-pohjainen Drupal. (Liferay. What is a Portal? 2013.)

3.1 Liferayn valintaperuste

Perusteet Liferayn valinnasta obshare.com:n julkaisualustaksi ovat sen Java-pohjaisuus ja avoin lähdekoodi, eli sitä on mahdollista muokata vastaamaan paremmin omia tarpeita. Java-pohjaisuudesta on taas käytännön hyötyä, sillä kaikki Observis Oy:n palvelut ja tietokantayhteydet pohjautuvat Javaan. Valmiina olevien komponenttien ja kirjastojen helpon integroinnin takia Java-teknologia on luonnollinen valinta myös julkaisupuolelle. (Kanerva 2013.)

Toisena syynä Kanerva tuo haastattelussa esille Liferayn laajan käytön nimekkäiden organisaatioiden julkaisualustana maailmalla. Sillä on hyvä maine monipuolisuutensa vuoksi, vaikka se ei ole helpoin alusta kehittää ruohonjuuritasolta. Java-pohjaisuus ja laaja käyttö maailmalla tukivat Liferayn valintaa. PHP-pohjaiset Drupal ja wordpress ovat Kanervan mielestä kohdistettu enemmän pienien toteutuksien luomiseen. Lisäksi PHP-pohjaisuus vaatisi ylimääräisen integraatiokerroksen ja työntekijöiden tarkemman perehtymisen PHP:hen. Liferayhin voidaan myös upottaa tarvittaessa aikaisemmin Observiksen käyttämiä GWT (Google Web Toolkit)-tekniikalla luotuja komponentteja sekä jopa tarvittaessa PHP-komponentteja. Liferayn kanssa on myös mahdollista käyttää kehittäjän kannalta nopeaa PrimeFaces-käyttöliittymäkirjastoa Liferay portlettien toteutuksessa. (Kanerva 2013.)

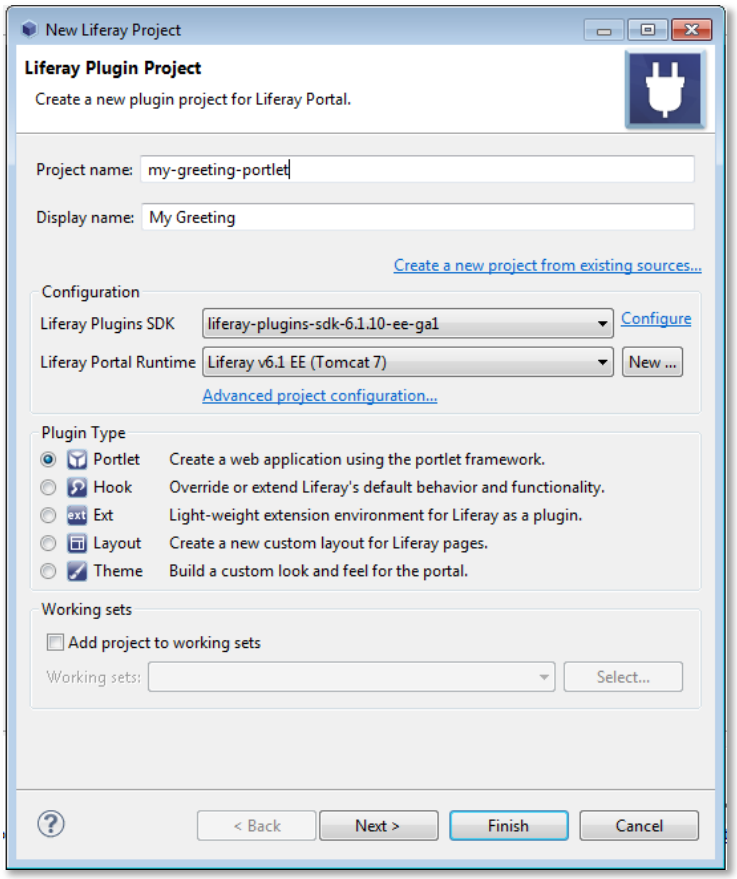
3.2 Kehitystehtävän kannalta keskeiset käsitteet

Liferay sisältää useita käsitteitä, joiden tunteminen on tärkeää Liferayn teeman toteutuksessa. Tässä opinnäytetyössä keskitytään varsinkin teemojen ja sisällön muokkaamiseen, joten käsitteet *”theme”* ja *”portlet”* ovat erityisen paljon esillä. Seuraavaksi esittelen lyhyesti keskeisimmät obshare.com-verkkopalvelun käyttöliittymän toteutukseen liittyvät käsitteet.

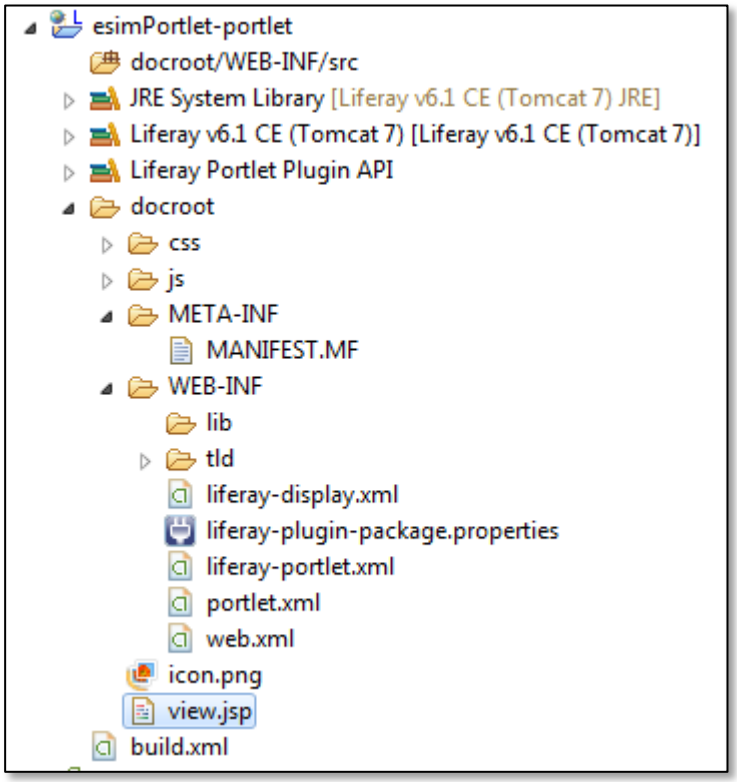
3.2.1 Portlet

Portlet on Javalla toteutettu verkkokomponentti. Liferayn näkyvät sivut koostuvat käytännössä portleteista. Portletteja voi olla yksi tai useampi ja niiden sijaintiin voidaan vaikuttaa Layout-jäsennystiedostolla. Portlet voi olla käytännössä itsenäisesti toimiva Java-sovellus. Liferayssa on jo lukuisia valmiita portletteja, mutta ne eivät monesti pelkästään riitä. (Liferay. About Portlets 2013.)

Portletteja voidaan luoda helposti Liferay IDE (Integrated Development Environment) Eclipse-laajennuksen avulla (kuva 11). Liferay IDE:n projektiluontityökalu generoi valmiin pohjan portletille, joka sisältää JRE-, Liferay- ja Liferay Portlet Plugin-kirjastot. Lisäksi portlet-projekti sisältää **docroot**-kansion, joka sisältää kehittämistehtävän kannalta olennaisen kansion **css**, jonka sisällä on tiedosto **main.css** (kuva 12). Tiedostoa voidaan käyttää portletin sisäisten tyylien määrittelemiseen. Sama voidaan kuitenkin toteuttaa myös teema-paketin sisällä, mikä on omien mieltymyksien ja kokemuksien mukaan helpompaa, koska kaikki tyylitiedostot ovat samassa paikassa. Lisäksi muutoksien näkyminen muille kehittäjille vaatii vain pelkän teema-paketin päivittämisen.



KUVA 11. Liferayn projektinluontityökalu (Liferay. Liferay Portal 6.1 – Development Guide, Creating a Portlet 2013)



KUVA 12. Liferayn portlet projektin kansiorakenne

Oletuksena portlet-projekti Java-projektin tapaan sisältää tiedoston *”view.jsp”*, joka on portletin näkyvä osa. Kaikki näkyvän käyttöliittymän kehittäjän kannalta oleellinen tapahtuu *”view.jsp”*-tiedostossa (kuva 13). (Liferay. Liferay Portal 6.1 – Development Guide, Creating a Portlet 2013.)

view.jsp

```
1  <%
2  /**
3   * Copyright (c) 2000-2012 Liferay, Inc. All rights reserved.
4   *
5   * This library is free software; you can redistribute it and/or modify it under
6   * the terms of the GNU Lesser General Public License as published by the Free
7   * Software Foundation; either version 2.1 of the License, or (at your option)
8   * any later version.
9   *
10  * This library is distributed in the hope that it will be useful, but WITHOUT
11  * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS
12  * FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more
13  * details.
14  */
15 %>
16
17 <%@ taglib uri="http://java.sun.com/portlet_2_0" prefix="portlet" %>
18
19 <portlet:defineObjects />
20
21 This is the <b>EsimPortlet</b> portlet.
22
```

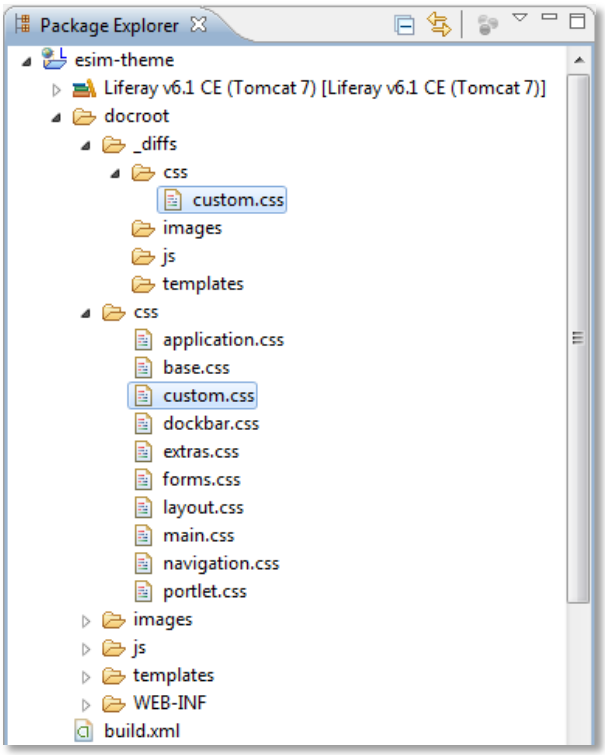
KUVA 13. Liferayn portlet projektin view.jsp

Näkyvä osa voidaan toteuttaa muillakin tekniikoilla. Tässä opinnäytetyössä muokattavien portlettien näkyvä osa on toteutettu esimerkkiportletista poiketen XHTML-tiedostoilla.

3.2.2 Theme

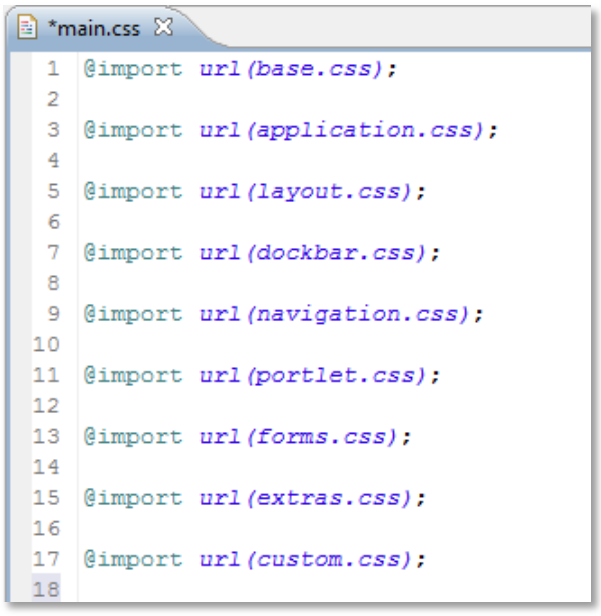
Theme määrittää Liferayn yleisen teeman, eli ulkoasun. Teemoja voi olla useita ja niitä hallitaan Liferayn hallintantyökalun avulla. Eri sivuilla voi olla myös omat teemansa. Teemoja voidaan luoda samalla Liferay IDE-laajennuksen avulla, kuten portletteja (Liferay. Liferay Portal 6.1 – Development Guide, Creating Liferay Themes 2013). Theme-paketin sisällä toteutuksen oleellisin kansio on **docroot**, jonka sisältä löytyvät kaikki teeman kuva-, CSS-, JavaScript- ja velocity-tiedostot. Tiedostot on järjestetty omiin kansioihinsa: **images**, **css**, **js** ja **templates**. Kansion docroot sisältä löytyy myös kansio **_diffs**, joka sisältää samat kansiot, kuin docroot. Kaikki teemaan itse tehdyt muutokset tallennetaan *”_diffs”*-kansion alla oleviin kansioihin, josta

Liferay automaattisesti ylikirjoittaa muutokset *”docroot”*-kansion alla oleviin tiedostoihin (kuva 14). (Yuan ym. 2010, 53-58.)



KUVA 14. Liferayn teemaprojektin kansiorakenne

CSS-kansio sisältää kaikki teeman tyylitiedostot. Tiedosto **custom.css** on tarkoitettu omien tyylien tuomiseen teemaan. Tyylitiedostot nidotaan lopuksi yhteen tiedostossa **main.css** (kuva 15).



KUVA 15. CSS-tiedostojen liittäminen main.css-tiedostoon

Tiedosto *”custom.css”* tuodaan *”main.css”*-tiedostoon vasta viimeisenä muutoksien ylikirjoittamiseksi, kuten kuvassa 15 on esitetty. Tiedostoon voidaan myös liittää suoraan portlettien tyylitiedostot, jos ne on toteutettu teeman sisällä.

3.2.3 XHTML

XHTML on merkkauskieli, johon törmätään obshare.com-verkkopalvelun sisältämissä portleteissa. Portlettien näkyvät osat koostuvat *”view.jsp”*-tiedoston sijaan usein useista XHTML:llä toteutetuista näkymistä. XHTML on HTML:ää, joka on kirjoitettu, kuten XML-tiedostot.

XML tulee sanoista e**X**tensible **M**arkup **L**anguage, mikä tarkoittaa laajennettavissa olevaa merkkauskieltä. XML-tiedostoihin voidaan luoda täysin omia elementtejä kuten: *”<tuote></tuote>”*. XML-tiedostot täytyy kirjoittaa täydellisesti, eli sen elementit pitää sulkea *”</ >”*. Tiedoston tyypin ilmoitus, eli doctype, määritellään tiedoston alussa: *”<?xml version=1.0”?>”*. XML-elementit kirjoitetaan pelkästään pienillä kirjaimilla ja sisällöllä on oltava yksi juuri (kuva 16). (W3Schools. XML Tutorial 2013.)

```
1  <?xml version=1.0"?>
2  <tietokanta>
3      <tuote>Omena</tuote>
4      <hinta>0.20</hinta>
5  </tietokanta>
```

KUVA 16. Esimerkki XML-tiedostosta

HTML tulee sanoista **H**yper **T**ext **M**arkup **L**anguage, joka tarkoittaa hyperteksin merkkauskieltä. Hyperteksti tarkoittaa lyhyesti dokumenttia, joka voidaan linkittää toiseen dokumenttiin. HTML on XML:ää, jonka perusteella selaimet rakentavat näkyvät sivut. Useimmat selaimet sallivat ns. ”lepsumpaa” merkkaustapaa eli HTML-elementtejä ei ole aina pakko sulkea toimiakseen. Esimerkiksi: *” Pientä tekstiä”* saattavat toimia moitteetta. (W3Schools. HTML Introduction 2013.)

XHTML tulee sanoista e**X**tensible **H**yper **T**ext **M**arkup **L**anguage eli on siis XML:n ja HTML:n yhdiste. XHTML-tiedostot kirjoitetaan samoin, kuten XML-tiedostot. Esimerkiksi jokainen HTML-elementti on suljettava: *” Pientä tekstiä ”*. (W3Schools. HTML – XHTML 2013.)

3.2.4 CSS3, AlloyUI ja Velocity

CSS eli **C**ascading **S**tyle **S**heets on verkkosivujen tyylien määrittelemiseen käytettävä merkkauskieli. CSS:n avulla voidaan määritellä mm. sivuston fontit, värit, taustakuvat, elementtien koot jne. CSS:n versio 3, eli CSS3 tuo näiden perusmäärittelyjen lisäksi mukanaan uusia ominaisuuksia. CSS3:lla voidaan mm. luoda animaatioita HTML-elementeillä, joiden luomiseen tarvittiin aikaisemmin JavaScriptiä. Lisäksi CSS3 mahdollistaa mm. omien fonttien tuomisen, sekä HTML-elementtien ulkonäön muokkaamisen, kuten reunojen pyöristämisen, varjostuksen ja rotaation. CSS3:n avulla voidaan siis toteuttaa graafisia koristeluja, jotka ennen vaativat kuvankäsittelyä. Uudet CSS3 version tuomat ominaisuudet on listattu liitteessä 1. (W3Schools. CSS3 Tutorial 2013.)

AlloyUI on Yahoon YUI3 Jquery JavaScript-kirjastoon ja CSS3: n pohjautuva alusta. AlloyUI:n avulla voidaan tuoda valmiita käyttöliittymäelementtejä, kuten kuvagallerioita, painikkeita jne. sekä selvittää, minkä kokoinen on selaimen leveys kuvapisteinä. Liferay käyttää AlloyUI:ta käyttöliittymäelementeissään. (Yuan ym. 2010, 40-42.)

Liferay käyttää sivustojen rakenteissa **Apache Velocity** merkkauskieltä. Velocityllä voidaan viitata suoraan Java-koodiin ja käyttää sitä yhdessä HTML:n kanssa. Esimerkiksi eri kieliversioiden käännökset toteutetaan hakemalla velocityn avulla Liferayn hallintatyökalulla määritellyt sanat ja niiden käännökset otsikoihin ja valikoihin.

Liferayn teemapaketti sisältää oletuksena viisi velocity-tiedostoa, jotka määrittelevät sivuston HTML-rakenteet. Tiedostot löytyvät teeman **/docroot/templates** kansiosta:

- **init_custom.vm**: Tiedostoon voidaan ylikirjoittaa ja luoda uusia velocity-muuttujia.
- **navigation.vm**: Tiedosto kutsutaan portal_normal.vm tiedostossa ja käsittää navigaation HTML rakenteen.
- **portal_normal.vm**: Määrittelee sivuston HTML perusrakenteen.
- **portal_pop_up.vm**: Määrittelee ponnahdusikkunoiden rakenteen.
- **portlet.vm**: Määrittelee portlettien ulkoisen kehyksen rakenteen.

Tässä opinnäytetyössä keskitytään erityisesti portal_normal.vm-, portlet.vm- ja navigation.vm-tiedostojen rakenteiden muokkaamiseen. (Yuan, ym. 2010, 43-51.)

3.2.5 HTML5

Liferay portaalin uusimmat versiot (6.0, 6.1) käyttävät oletusteemansa velocity-tiedostoissa HTML5-rakenteita. HTML5 on uusin versio HTML-merkkauskielestä, joka on tuettuna jo uusimmissa selaimissa. Liferayssa HTML5-tuki näkyy oletusteeman HTML-version ilmoituksesta <!DOCTYPE html>, lisäksi sivuston otsikoissa ja navigaatioissa on käytetty <header>- ja <nav> -elementtejä (kuva 17). (Yuan, ym. 2010, 42-51.)

portal_normal.vm

```
<!DOCTYPE html>

#parse ($init)

<html class="#language("lang.dir")" dir="#language("lang.dir")" lang="$w3c_language_id">

<head>
    <title>$the_title - $company_name</title>

    $theme.include($top_head_include)
</head>

<body class="$css_class">

$theme.include($body_top_include)

#if ($is_signed_in)
    #dockbar()
#end

<div id="wrapper">
    <a href="#main-content" id="skip-to-content">#language("skip-to-content")</a>

    <header id="banner" role="banner">
        <div id="heading">
            <h1 class="site-title">
                <a class="$logo_css_class" href="$site_default_url" title="#language("go-to") $site_name">
                    <img alt="$logo_description" height="$site_logo_height" src="$site_logo" width="$site_logo_width" />
                </a>

                #if ($show_site_name)
                    <span class="site-name" title="#language("go-to") $site_name">
                        $site_name
                    </span>
                #end
            </h1>

```

KUVA 17. HTML5:n käyttö oletusteemassa

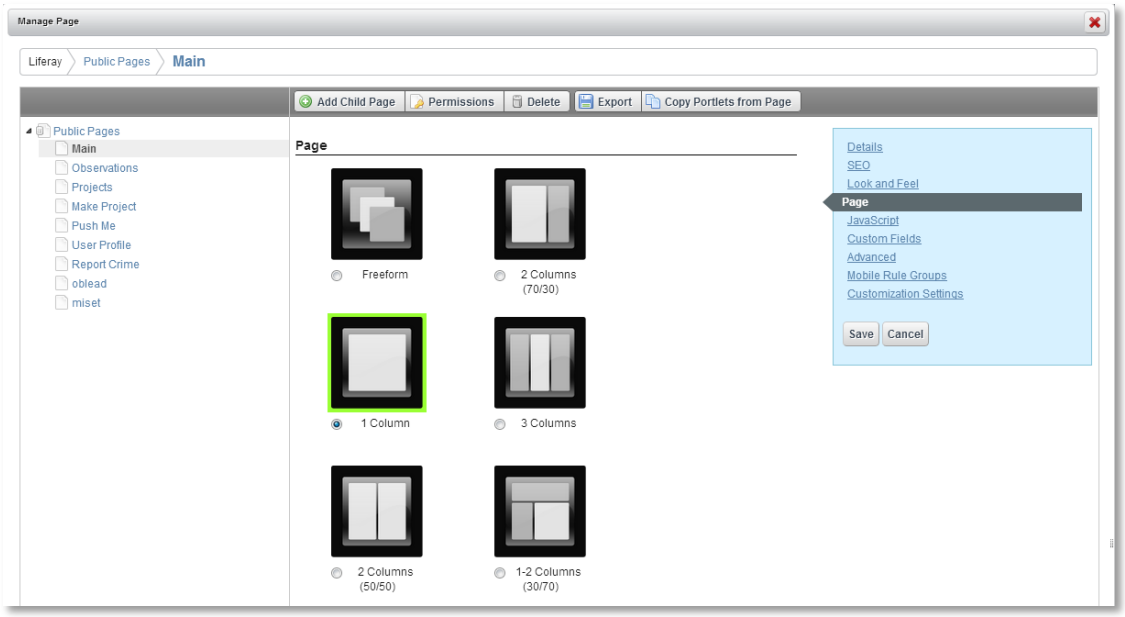
HTML5 tuo kehittäjälle käyttöön uusia elementtejä, jotka mahdollistavat verkossa toimivien sovellusten toteuttamisen entistä monipuolisemmin ja vähemmällä vaivalla.

3.2.6 Layout

Liferayn sivujen sisällön jäsentäminen hoidetaan layout-tiedostojen avulla. Sivuston jäsentymisen määrittävä tiedosto valitaan Liferayn hallintapaneelin vasemmasta laidasta: **Manage** –> **Page Layout** (kuva 18).

KUVA 18. Sivuston jäsentämistyökalun sijainti hallintapaneelissa

Jäsentämistyökalun avulla voidaan hallita koko varsinaisen sivun jäsentämistä palstoihin. Liferay sisältää oletuksena jo lukuisia eri tavoin sivua palstoittavia layout-rakenteita, joiden mukaan sivustojen portlettien jakaman tila suhteet muuttuvat (kuva 19). Portletteja voidaan sijoittaa suoraan vapaana oleviin palstoihin raahaamalla. (Liferay. Layout Template 2013.)



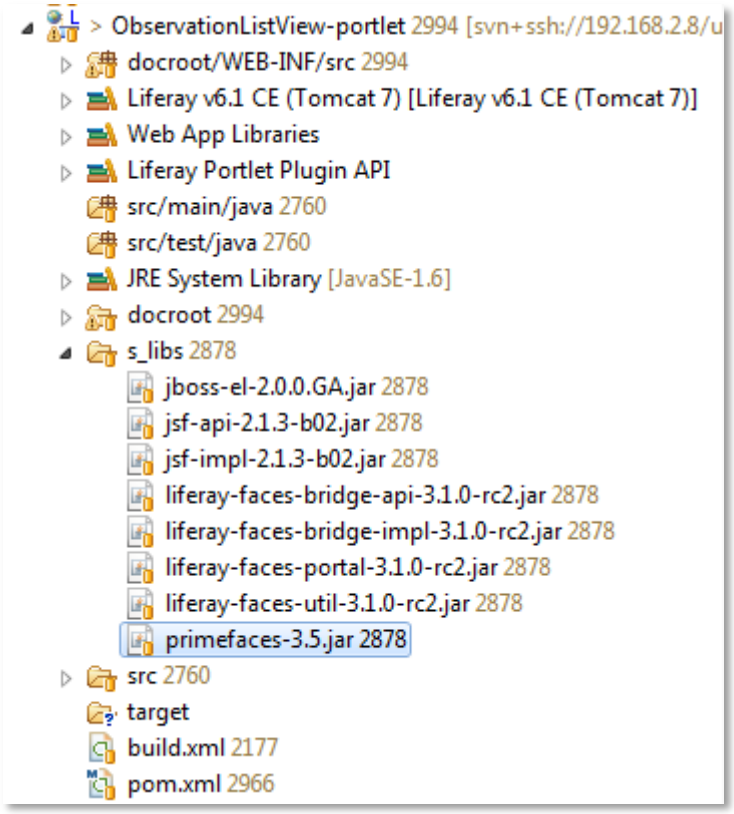
KUVA 19. Sivuston jäsennystavan valintatyökalu

Tässä opinnäytetyössä toteutuksessa on käytetty yhden kolumnin layout-rakennetta (kuva 19), sillä sivu tulee sisältämään vain yhden, koko tilan täyttävän havaintoportletin ObservationListView-portlet. Tarvittaessa layout-tiedostoja voidaan

rakentaa lisää visuaalisen editorin avulla Eclipsen Liferay IDE-laajennuksella. Tässä työssä sitä ei ole kuitenkaan tarvinnut tehdä.

3.2.7 PrimeFaces

PrimeFaces on obshare.com-verkkopalvelun käyttöliittymäelementtien toteuttamisessa käytetty avoimen lähdekoodin käyttöliittymäkirjasto, joka pohjautuu JSF (Java Server Faces) teknologiaan (Çivici 2013, 12). PrimeFaces on helppo ottaa käyttöön Liferayssä, koska se pohjautuu Liferayn lailla Java-teknologiaan. PrimeFaces otetaan käyttöön lataamalla pakattu tiedosto: primefaces-[version numero].jar, joka lisätään projektiin kuvan 20 mukaan. (PrimeFaces, Getting Started 2013.)



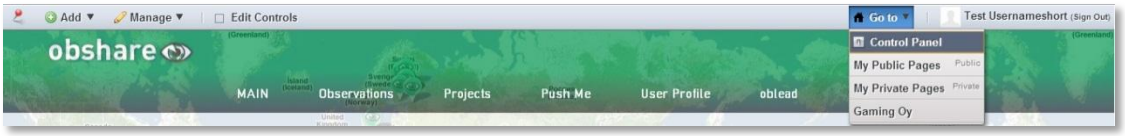
KUVA 20. PrimeFaces-paketti ObservationListView-portletissa

PrimeFaces liittyy keskeisesti tähän opinnäytetyöhön, koska sitä on käytetty apuna portlettien toiminnallisuuden toteutuksessa. Monet PrimeFacesin elementit eivät kuitenkaan sellaisenaan kelpaa sovelluksen käyttöliittymään, joten niitä täytyy muokata CSS:llä tai vaihtaa tilalle sopivampia elementtejä. Esimerkiksi PrimeFaces-ryhmittelyelementti **PanelGrid** soveltuu erinomaisesti eri komponenttien asemoimiseen nopeasti, mutta se ei sovellu hyvin responsiivisen palvelun

toteuttamiseen sen generoimien HTML-taulukkoelementtien takia. Monesti tiettyjen komponenttien täytyy pystyä siirtymään toistensa päälle tai rinnalle katseltavan laitteen näyttökoosta riippuen, mihin taulukkoelementit eivät sovellu. Vaihtamalla PanelGrid-elementtien tilalle elementin **Panel**, saadaan aikaiseksi normaali HTML *” ”*-elementti, jota on huomattavasti helpompi muokata. Elementeille voidaan lisätä omia tyyliluokkia tavalla: *”<pf:panel styleClass=”luokan_nimi”>”*. (Çivici 2013, 300-305.)

3.3 Sisällönhallintatyökalu

Liferay-julkaisujärjestelmän sisältöä hallitaan siihen sisäänrakennetulla hallintatyökalulla. Hallintatyökalu on käytettävissä ylläpito-oikeudet omaavilla käyttäjillä. Hallintatyökalun avulla ylläpitäjä voi lisätä mm. sivustolle uusia sivuja, portletteja, hallita sivustojen käyttäjäoikeuksia ja hallita käyttäjiä, sekä ottaa käyttöön teemoja. Käytännössä hallintatyökalun avulla suoritetaan kaikki sivuston ylläpitoon liittyvät tehtävät. Hallintapaneeli (Control Panel) löytyy kirjautumisen jälkeen sivuston ylälaidasta hallintatyökalusta (kuva 21).



KUVA 21. Hallintapaneelin valikko

Sivuston hallintaan päästään valitsemalla **Control Panel** kuvan 21 osoittamalla tavalla. Hallintapaneelin kautta pystytään kontrolloimaan koko Liferayn sisältöä, rakennetta, käyttäjiä ja teemoja. Hallintapaneelin vasen laita sisältää sivuston nykyisen sivun pika-asetukset kohdan **Manage** alta. **Edit Controls** valittuna ylläpitäjä voi siirrellä ja muokata sivun portletteja. (Sezov ym. 2013, 183-188.)

4 CASE: OBSHARE.COM TEEMAN TOTEUTUS

Obshare.com-yhteisöpalvelu on Observis Oy:n projekti, jonka on tarkoitus valmistua keväällä 2013. Tässä luvussa esittelen kehittämisen aikaista työskentelyprosessia ja omaa rooliani kehitystiimissä. Työtä tehdään kansainvälisessä tiimissä, yhdessä ohjelmoijien, graafikon ja palvelun suunnittelijoiden kanssa. Lisäksi esittelen teeman

toteutuksen menetelmät ja toteutukseen käytetyt kehitysvälineet ja testausympäristön. Koska palvelu on laaja, keskityn pelkästään yhden sivun näkymän käyttöliittymän toteutuksen esittelemiseen. Palvelun muut sivut tullaan toteuttamaan samalla menetelmällä. Kehittämistehtävänäni on toteuttaa palvelun julkaisualustana käytettävän Liferayn ulkoasun määrittelevä teema-paketti responsiivisen verkkosuunnittelun periaatteella.

4.1 Työskentelytiimi ja työskentelyprosessi

Toteutusta tekevään tiimiin kuuluvat graafikko, suunnittelijat, ohjelmoijat (back end) ja WEB-kehittäjä (front end). Työtä tehdään pääosin englannin kielellä, työtiimin kansainvälisyydestä johtuen. Minun tehtäväni projektin WEB-kehittäjänä on sekä suunnitella, että toteuttaa palvelun ulkoasua ja käyttöliittymää yhdessä graafikon, suunnittelijoiden ja ohjelmoijien kanssa (kuva 22).

Tiimin jäsenten tehtävät:

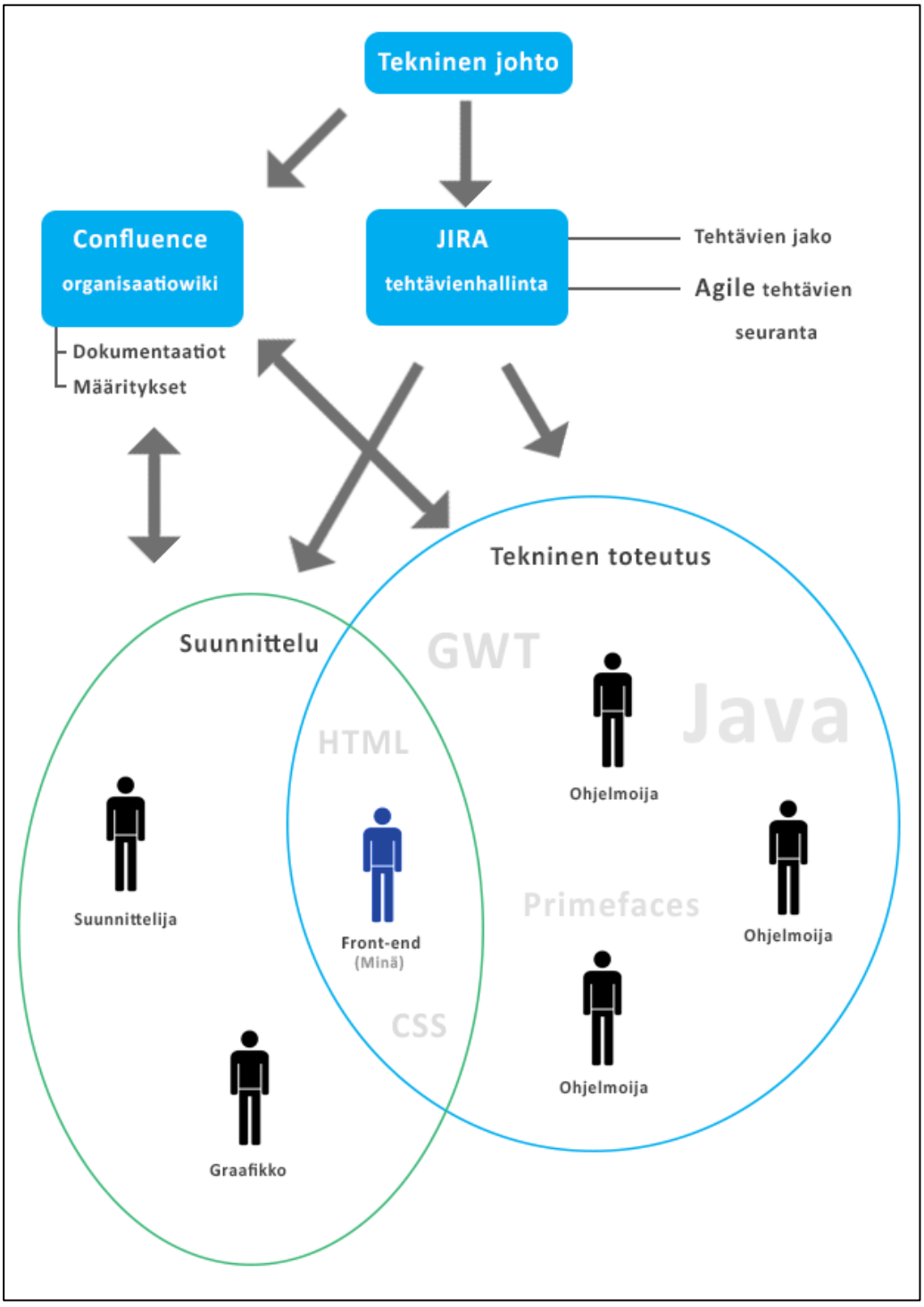
Suunnittelija
Suunnittelee ja kuvaa palvelun käyttötapaukset. Tekee käyttöliittymäsuunnittelua yhdessä WEB-kehittäjän ja graafikon kanssa (rautalankamallit). Ei ota kantaa tekniseen toteutukseen.

Graafikko
Toteuttaa palvelun graafisen ohjeiston. Suunnittelee käyttöliittymän ulkoasua yhdessä WEB-kehittäjän ja palvelun suunnittelijan kanssa.

WEB-kehittäjä (front end)
Toteuttaa palvelun käyttöliittymän ei-toiminnallisen toteutuksen graafisen ohjeiston ja rautalankamallien mukaan. Suunnittelee käyttöliittymän rakennetta ja ulkoasua yhdessä graafikon ja palvelun suunnittelijan kanssa. Työskentelee myös tiiviisti yhdessä ohjelmoijien kanssa käyttöliittymätoteutuksen yhteydessä.

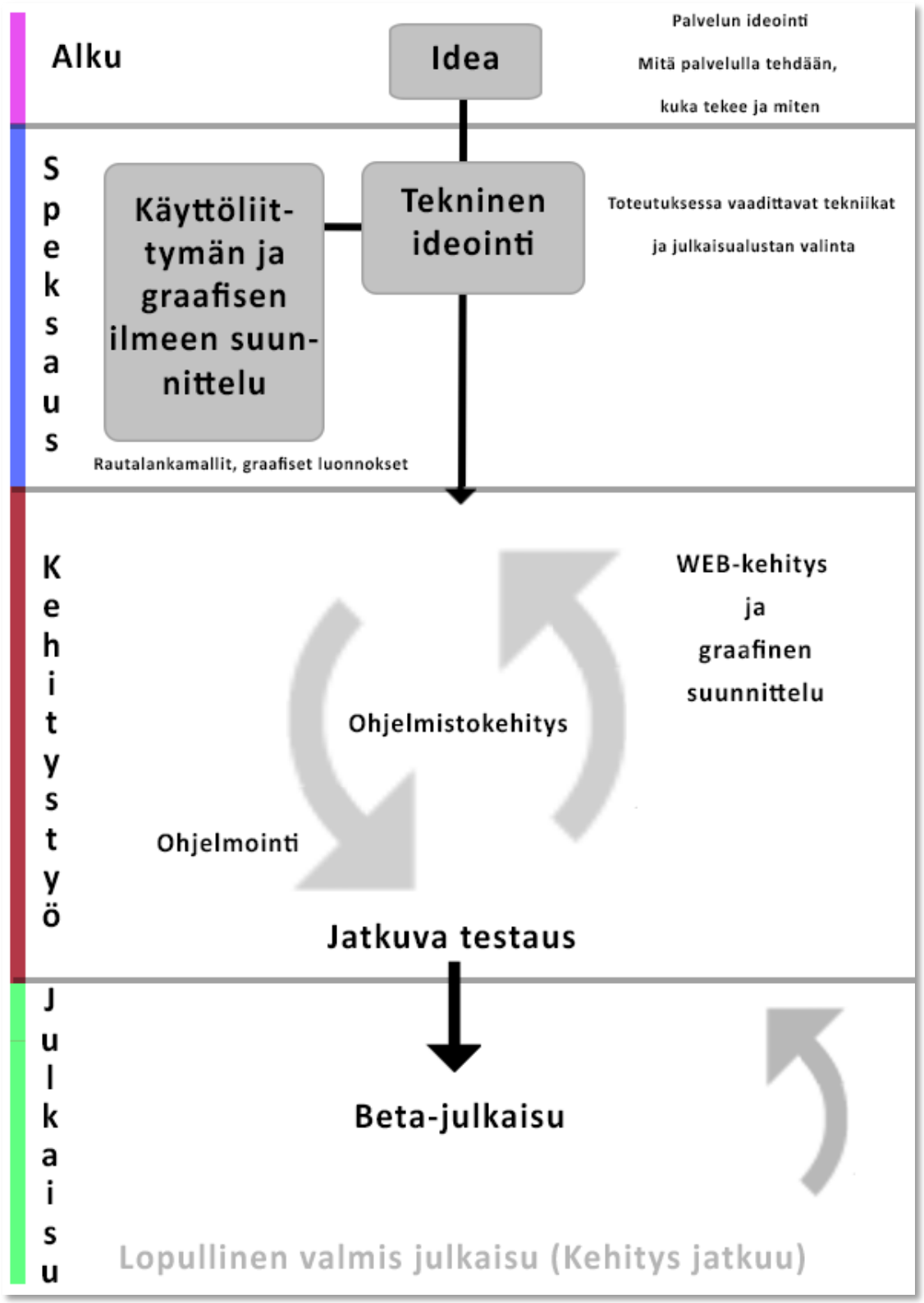
Ohjelmoijat (back end)
Toteuttavat suunnittelijoiden määrittelemät palvelun toiminnallisuudet. Osallistuvat myös teknisen toteutuksen suunnitteluun.

Työskentelyssä työnjako tapahtuu projektipäällikön toimesta **JIRA**-tehtävienhallintaohjelman kautta. Tiimin jäsenet kirjaavat tehtäviensä toteutumisen kulun JIRA:ssa olevaan **Agile**-tehtävienseurantaohjelmaan. Toteutuksen aikana tuotetut palvelun kuvaukset ja ohjeistot voidaan tallentaa **Confluence**-organisaatiowikiin. Jokainen työntekijä pystyy lukemaan ja lisäämään dokumentaatioita palvelusta ja sen toteutuksesta (kuva 22).



KUVA 22. Projektin työskentelytiimi

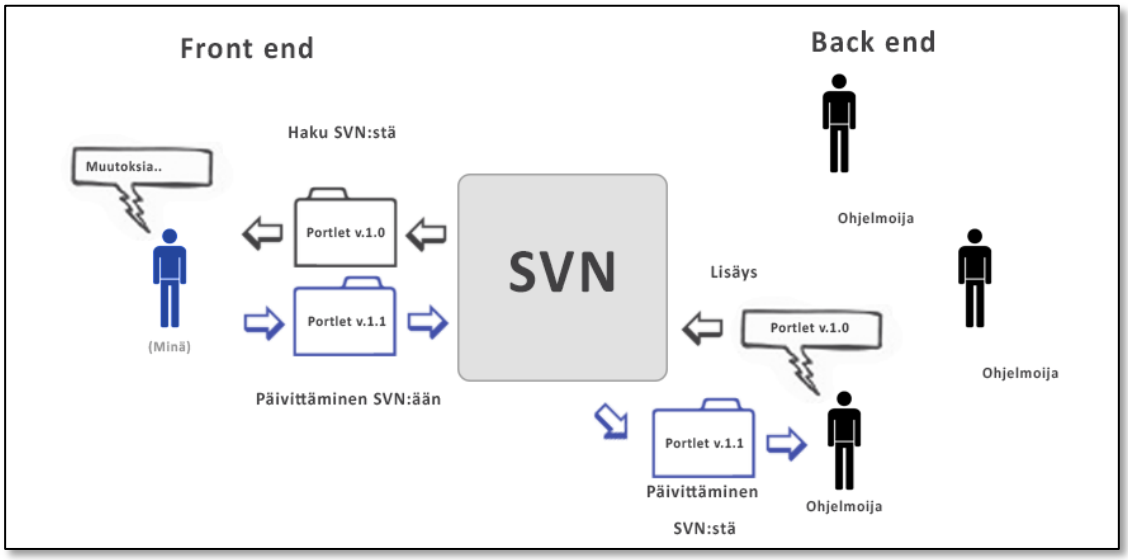
Käytännön toteutusta tehdään yhdessä graafikon ja ohjelmoijien kanssa. Graafikko suunnittelee palvelun yleisen graafisen ilmeen ja typografian rautalankamallien avulla. Kehitystä tehdään ketterän ohjelmistokehityksen menetelmillä, joten alkuperäisten suunnitelmien ja lopputuloksen välillä voi olla suuriakin eroja (kuva 23).



KUVA 23. Projektin työskentelyprosessi

Portaalin yleisen teeman suunnittelun ja luonnin lisäksi, tehtäväni on muokata ohjelmoijien toteuttamien palvelun sisältämien portlettien ulkoasua ja käyttöliittymää vastaamaan graafista ohjeistoa.

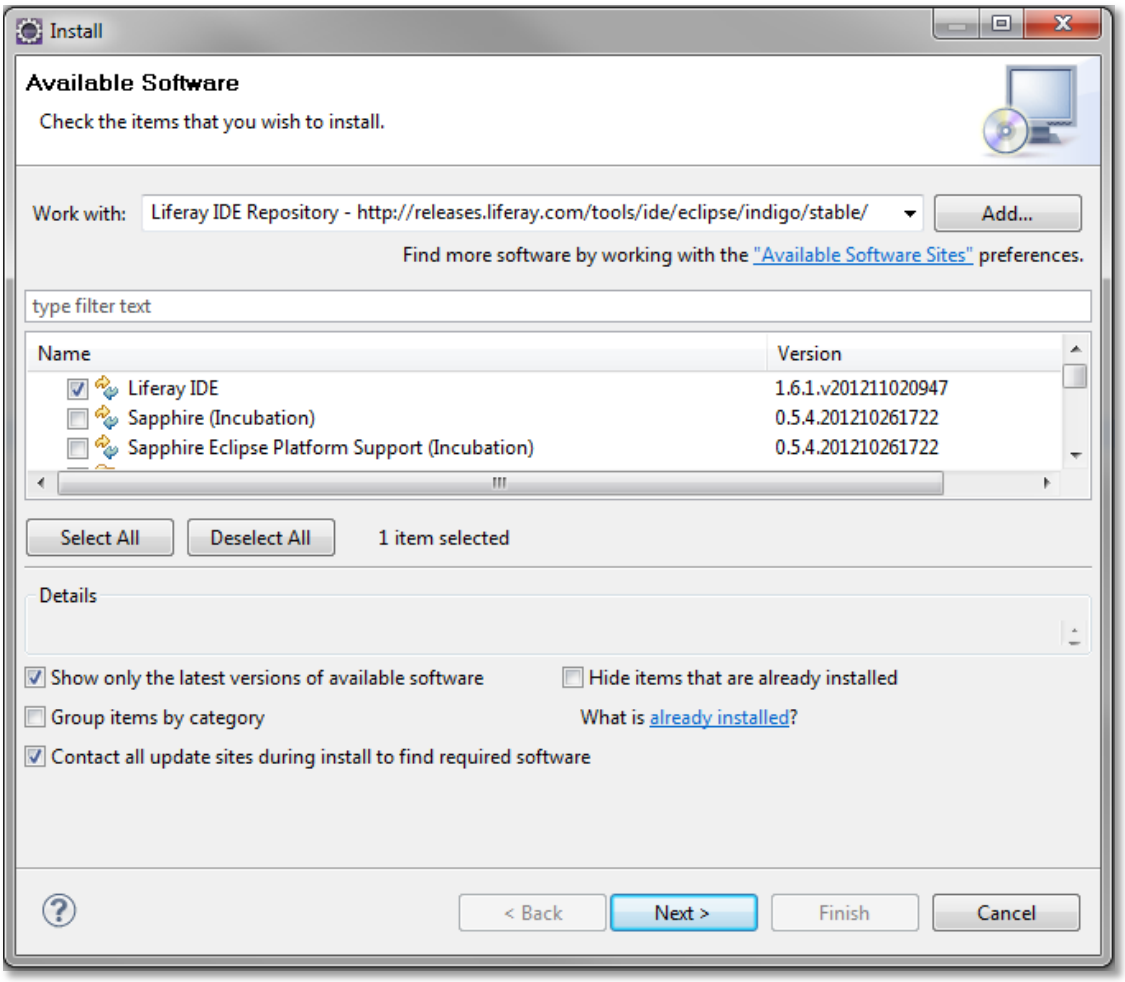
Teknisen toteutuksen tiimityöskentelyssä (back end & front end) käytetään apuna **SVN**-versionhallintajärjestelmää. Käytännössä ohjelmoija toteuttaa portletin, jonka hän lähettää SVN:ään. SVN:stä WEB-kehittäjä lataa portletin omaan kehitysympäristöönsä ja tekee siihen tarvittavat muutokset, jotka vastaavat ulkoasultaan graafista ohjeistoa. Tämän jälkeen WEB-kehittäjä päivittää muutokset SVN:ään, mistä muut ohjelmoijat pystyvät päivittämään muutokset omille kehitysympäristöilleen (kuva 24).



KUVA 24. SVN:n hyödyntäminen teknisessä toteutuksessa

4.2 Kehitys- ja testausympäristö

Teeman toteutuksessa on käytetty kehitysvälineenä **Eclipse Java EE IDE for Web Developers** versiota **Helios Service Release 2**, johon on asennettu **Liferay IDE** (**I**ntegrated **D**evelopment **E**nvironment) kehitysympäristö. Liferay-kehitysympäristön asentaminen onnistuu suoraan Eclipsen ylävalikosta **Help > Install New Software**. *"Work with"*-kenttään kirjoitetaan Liferayn sivuston ohjeiden mukaan osoitekenttään URL: *http://releases.liferay.com/tools/ide/eclipse/indigo/stable/*, josta Liferay IDE löytyy (kuva 25).



KUVA 25. Liferay IDE asennus Eclipseen

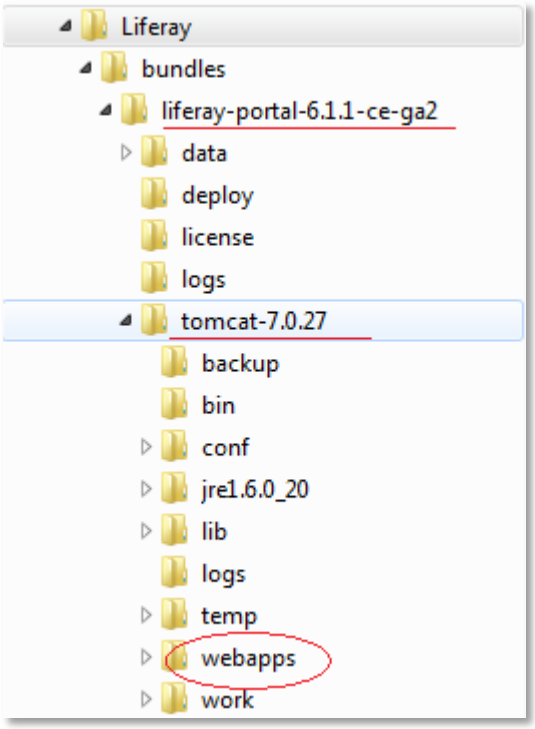
Asentamisen ja Eclipsen uudelleenkäynnistyksen jälkeen Liferay IDE-kehitysympäristö on käytettävissä. Tämän jälkeen on tietokoneelle asennettava itse Liferay.

Liferayn asennus

Verkkopalvelu obshare.com käyttää alustanaan Liferay v6.1 CE versiota, joten sitä käytetään luonnollisesti myös kehityksessä. Liferay 6.1 CE:n voi ladata Liferayn kotisivuilta osoitteesta: *http://www.liferay.com/downloads/liferay-portal/available-releases*. Kehitystyössä olen käyttänyt sivuston oletuksena tarjoamaa Tomcat 7-pakettia, jota Liferay IDE-kehitysympäristö tukee (Liferay, Liferay IDE Features 2013). Liferay Tomcat paketti ladataan omalle tietokoneelle pakattuna zip-tiedostona. Liferayn kotisivujen lataussivun mukaan Windows-käyttäjien kannattaa purkaa tiedosto jollain kolmannen osapuolen zip-pakettien purkuohjelmalla, paketin sisältämien pitkien tiedostonimien takia (Liferay. Get Liferay Portal 2013). Paketti

voidaan purkaa minne tahansa tietokoneelle, mutta varsinkin Windows-käyttäjille suositellaan suoraan esimerkiksi C-juureen asentamista, koska Windowsin NTFS-tiedostojärjestelmän kokonaisen polun pituus voi olla maksimissaan 256 merkkiä (Sezov 2012, 33).

Olen itse luonut kansion *"Liferay"* suoraan C-aseman juureen: **C:\Liferay**. Sezov:n ohjeen mukaan tähän kansioon luodaan kansio *"bundles"*, jonka sisään Liferayn Tomcat 7 zip-paketti puretaan. Purkamisen jälkeen kansiossa *"bundles"* pitäisi olla kansio: **liferay-portal-6.1.1-ce-ga2**. Tämä kansio sisältää Liferayn ja Tomcat 7-serverin (kuva 26).



KUVA 26. Liferay kansiorakenne

Kaikki Eclipsen Liferay-projektit tallennetaan suoraan Tomcat 7-serverin webapps-kansioon, mistä ne ovat suoraan käytettävissä (kuva 26). Tallennuskansio valitaan projektin luonnin yhteydessä.

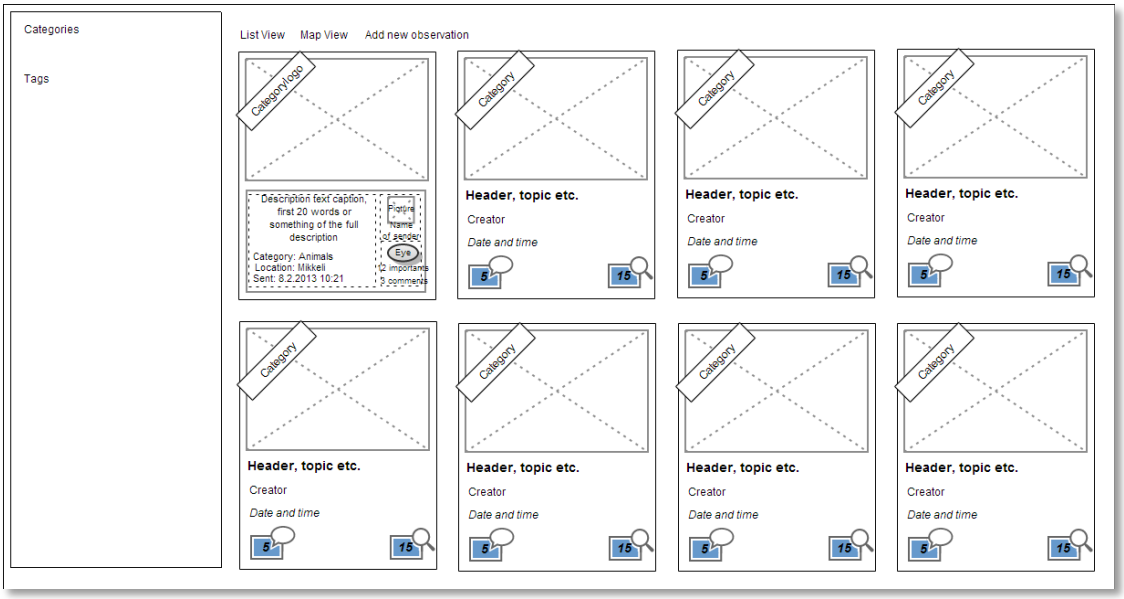
4.3 Käyttöliittymän kuvaus ja vaatimusmäärittely

Obshare.com-yhteisöpalvelun sivujen käyttöliittymä koostuu yhtenäisestä otsikko- ja ylä- ja alanavigaatioelementeistä. Näiden elementtien tyylit ja rakenteet määritellään

pelkästään Liferayn teemapaketissa. Ylä- ja alanavigaatiopalkkien väliin sijoitetaan havaintosivujen havaintojen haku-, listaus- ja selausohjelman sisältävä portletti: *ObservationListView-portlet*.

ObservationListView-portlet on portletti, joka toteuttaa verkkopalvelun lähetettyjen havaintojen listauksen ja näyttämisen toiminnallisuuden. Havaintosivut koostuvat käyttäjien lähettämistä havainnoista ja niiden suodattamiseen vaadittavista työkaluista. Sivujen vasemmalla laidalla on suodattimet, joiden avulla käyttäjän tulee pystyä suodattamaan näytettäviä havaintoja kategorian, ajan (uusin - vanhin) ja sijainnin perusteella. Demovaiheessa käytössä on pelkästään kategorian perusteella suodattaminen.

Oikealla puolella suurimman osan sivusta vie itse havaintolista, johon listataan havainnot. Havainnot esitetään kortteina, joista löytyy lyhyt kuvaus, kategoria, kuva, aika, lähettäjän kuva ja lähettäjän nimi. Lisäksi kortista näkyy havainnon kommenttien määrä, sekä käyttäjien antamat ”tärkeys”-pisteet (kuva 25).



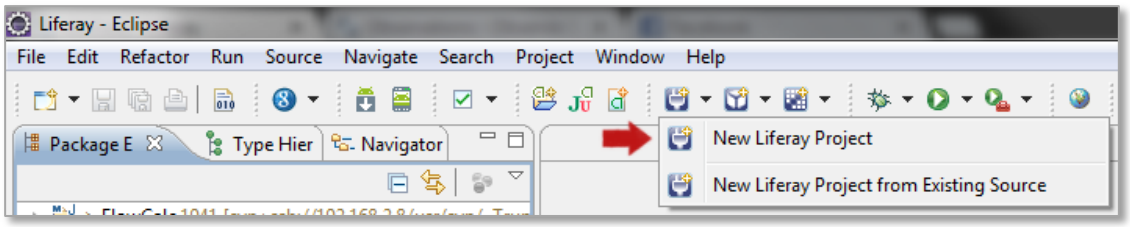
KUVA 27. ObservationListView-portlet listausnäkymän rautalankamalli

Listanäkymän lisäksi *ObservationListView-portlet* sisältää karttanäkymän ja yksittäisen havainnon selausnäkymän. Portletin sisäinen navigaatio toteutetaan listanäkymän yläpuolelle. Näkymä voidaan vaihtaa listanäkymästä karttanäkymään valitsemalla joko *”List View”* tai *”Map View”* Yksittäistä havaintokorttia klikkaamalla joko kartalla, tai listalla, avautuu se samaan tilaan suurempana

yksityiskohtaisempaa tarkastelua varten. Navigaatiossa on lisäksi linkki toiseen portlettiin, missä luodaan uusia havaintoja.

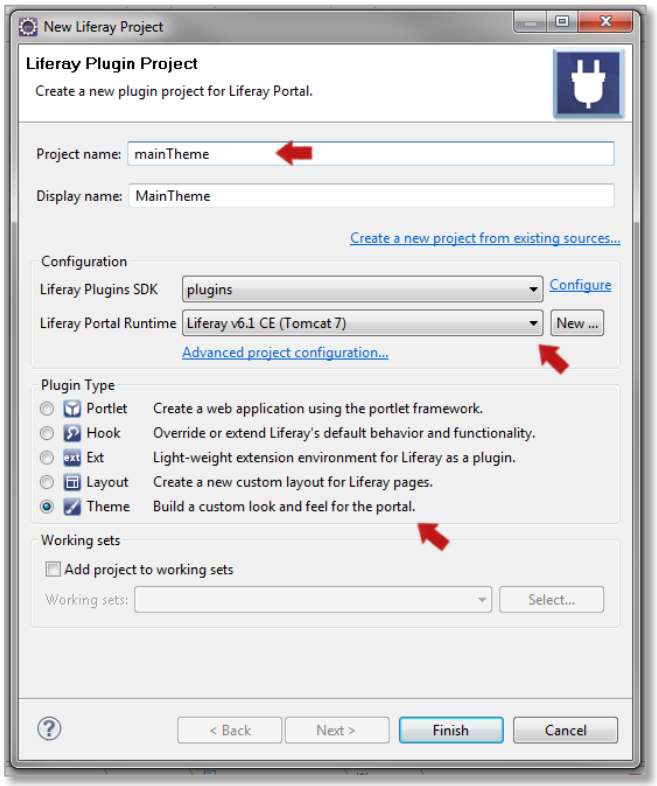
4.4 Teeman toteutus

Liferayn teemat luodaan Liferayn IDE Eclipse-laajennuksen projektinluontityökalulla. Teeman luonti aloitetaan valitsemalla Eclipsen ylävalikosta uusi Liferay-projekti kuvan 28 mukaan.



KUVA 28. Uusi Liferay-projekti

Tämän jälkeen aukeaa projektien, eli Liferay-laajennuksien (Plugin Project) luontityökalu (kuva 29).



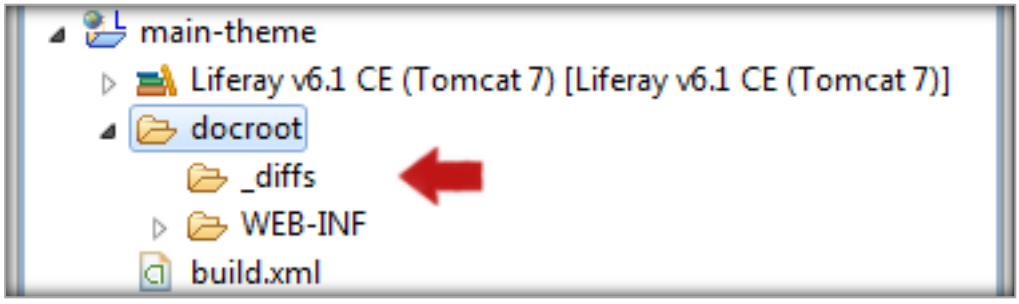
KUVA 29. Liferay laajennuksien luontityökalu

Listasta valitaan laajennuksen tyyppi, mikä teemoja tehdessä on: **Plugin Type: Theme**. Tämän jälkeen projekti nimetään. Tässä esimerkkikuvassa projekti on nimetty nimellä ”*mainTheme”*, mutta lopullisessa työssä olen käyttänyt projektinimeä *”main”*. Nimeämisen kannattaa olla kuvaava, sillä kuten jo aiemmin mainitsin, teemoja voi olla Liferayssa useita eri käyttötarkoituksia varten. Tässä tapauksessa tarkoituksena on luoda palvelun yleinen pääteema.

Editori lisää tämän jälkeen nimen perään *”-theme”*-päätteen. Omien kokemuksieni mukaan väliviivojen käyttö teeman nimen sisällä sotkee teeman lukemisen Liferayssa. Missään dokumentaatiossa tästä ei ollut mitään mainintaa, mutta varmuuden vuoksi suosittelen välttämään väliviivoja.

Nimeämisen jälkeen varmistetaan, että käytössä on haluttu Liferayn versio. Tässä työssä käytetään ilmaista **Liferay v6.1 Community Editionia**, eli CE, joka on tällä hetkellä uusin versio. Tämän opinnäytetyön ja toteutuksen kannalta keskeisintä on, että uusimmassa Liferay v6.1 CE versiossa on lisättynä **AlloyUI**-moduuli **aui-viewport**, jonka avulla voidaan tunnistaa eri näyttökokoja. Aikaisemmin kyseinen moduuli oli saatavilla vain Liferayn version 6.0 maksullisessa Enterprise Editionissa (EE). Esimerkiksi Jonas X. Yuanin, Xinsheng Chenin ja Frank Yu:n kirja: *Liferay User Interface Development* ei sisällä lainkaan mainintaa kyseisestä moduulista, koska kirja perustuu Liferayn Community Editionin versioon 6.0.

Valintojen jälkeen teema-laajennus luodaan painamalla *”Finish”*-painiketta (kuva 18). Eclipsen Liferay IDE luo uuden teemapaketin *”main-theme”*, joka sisältää oletuksena *”docroot”*-, *”_diffs”*- ja *”WEB-INF”*-kansiot (kuva 30).



KUVA 30. Liferay-laajennuksen luontityökalu

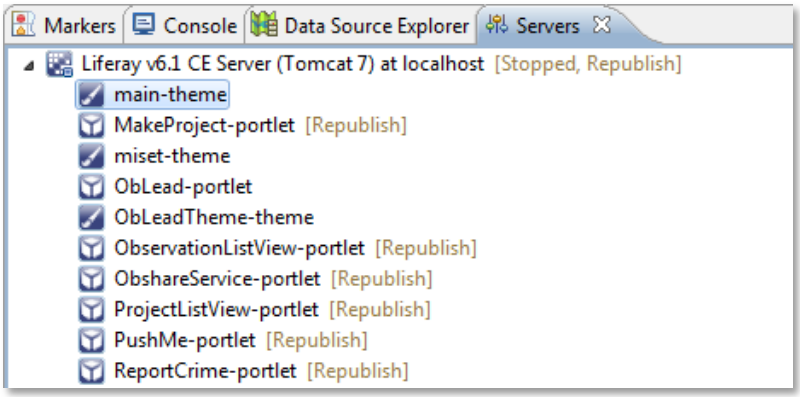
Teeman luonnin kannalta keskeisin kansio on **docroot** ja sen sisältä löytyvä kansio **_diffs**. Jotta teeman toteutusta ei tarvitsisi aloittaa täysin tyhjästä, käytän apuna Liferayn omaa oletusteemaa *”Classic”*. Oletusteeman tyylitiedostot sisältävät kansiot löytyvät Liferayn asennuskansion sisältä polusta: **C:\Liferay\bundles\liferay-portal-6.1.1-ce-ga2\tomcat-7.0.27\webapps\ROOT\html\themes\classic**. Kansiosta **classic** kopioidaan kaikki kansiot **css**, **images**, **js** ja **templates** uuden *”main-theme”*-projektin *”_diffs”*-kansioon. Tämän jälkeen Liferay IDE automaattisesti generoi oletusteeman sisällön *”_diffs”*-kansiosta *”docroot”*-kansioon. Kansioiden sisältöjen määritelmä on taulukossa 2. Theme-paketin kansiorakenne.

TAULUKKO 2. Theme-paketin kansiorakenne

Kansio:	**Sisältö:**
css	css-tyylitiedostot
images	kuvatiedostot
js	JavaScript-tiedostot
templates	Velocity-tiedostot

Kaikki teemaan tehtävät muutokset tehdään *”_diffs”*-kansion alla oleviin kansioihin ja tiedostoihin.

Teemaa voidaan heti kokeilla ottamalla se käyttöön Liferayn hallintatyökalulla. Teema on ensin liitettävä Liferayn serveriin raahaamalla *”main-theme”*-paketti Eclipsen Package Explorerista käynnissä olevan serverin: *”Liferay v6.1 CE Server (Tomcat 7) at Localhost”* päälle (kuva 31).



KUVA 31. Liferay v6.1 Tomcat 7 serveri ja siihen liitetyt lisäosat

Tämän jälkeen Liferay IDE luo automaattisesti Liferayn Tomcat 7-serverille kansion: **C:\Liferay\bundles\liferay-portal-6.1.1-ce-ga2\tomcat-7.0.27\webapps\main-theme**, jonka jälkeen teema on käytettävissä.

4.4.1 Teeman ulkoasun toteutus

Teeman ulkoasun toteutuksen pohjana on käytetty Liferayn *”Classic”*-teemapaketin tyylitiedostoja. Ennen tyylin muokkaamista on ensin muokattava *”portal_normal.vm”*-tiedostoa, joka sisältää sivuston HTML perusrakenteen (LIITE 2). Muokkaaminen aloitetaan poistamalla kaikki turhat rakenteet pois ja lisäämällä muutama lisäsääntö. Toteutuksessa on haluttu aivan ensimmäiselle sivulle yläbannerin taustalle liikkuva karttakuva. Karttakuvan liike on toteutettu erillisellä JavaScriptillä, joka tuodaan ensin normaalisti *”<head>”* osioon tutkimalla velocityn *”#if”*-kyselyllä, ollaanko etusivulla (kuva 32).

```
#if ($page.friendlyURL == "/main")
    <script type="text/javascript" src="http://maps.googleapis.com/maps/api/js?sensor=false"></script>
    <script type="text/javascript" language="javascript" src="/main-theme/observations/observations.nocache.js"></script>
#end
```

KUVA 32. Karttaohjelman tuominen pääsivulle velocityn avulla

Etusivu saadaan tutkittua kysymällä etusivun ns. ystävällistä osoitetta, eli *”friendlyURL”*. FriendlyURL on osoiterivillä heti *”/web/ymp”* jälkeen, mistä se on nopeasti selvitettävissä (Liferay. Wiki, Friendly URLs 2013). Itse kartta lisätään *”<header>”* osion sisään omassa *” ”*-elementissä, jos ollaan etusivulla (kuva 33).

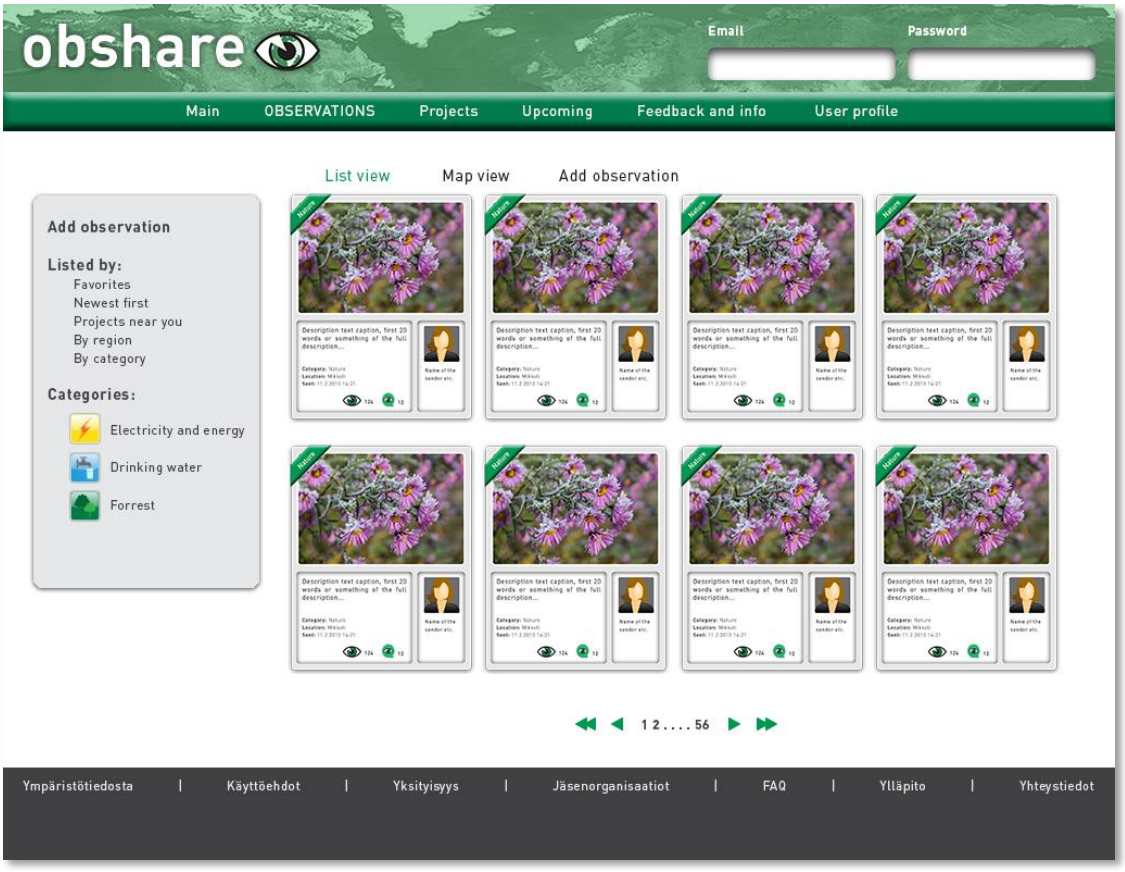
```
    <header id="banner" role="banner">
        <div id="heading">
            #if ($page.friendlyURL == "/main")
                <div id="uniqueId" class="map-banner"></div>
            #end
                <div class="logo"></div>
        </div>
```

KUVA 33. Liikkuvan kartan tuominen pääsivulle velocityn avulla

Samalla tavalla teemaan voidaan halutessa toteuttaa muitakin erikoissääntöjä eri sivuja varten.

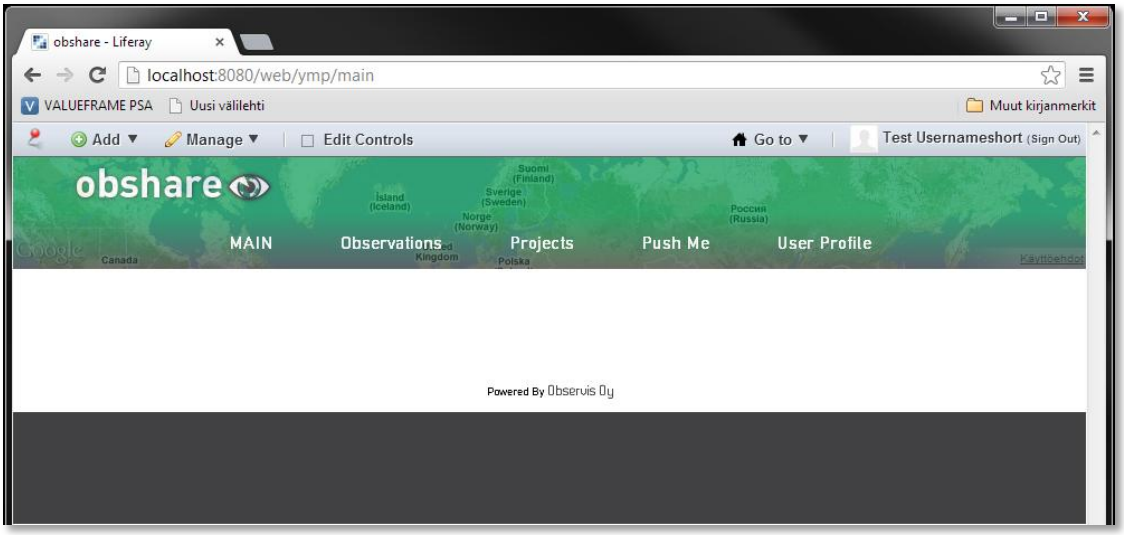
Tiedostosta *”portal_normal.vm”* on muokattu lisäksi hiukan *”<footer>”*-elementin sisältöä. Tämän jälkeen on muokattu *”portlet.vm”*-tiedostoa, joka sisältää portlettien ulkoisen kehyksen. Portleteista on poistettu otsikkorivit poistamalla *”<h1>”*-tekstielementti, jonka luokka on *”portlet-title”* (vrt. LIITE 4 ja LIITE 5). Samalla tavalla on muokattu vielä *”navigation.vm”*-tiedostoa, joka määrittelee sivuston navigaation HTML-rakenteen (LIITE 6).

Velocity-rakenteiden muutoksien jälkeen olen muokannut teeman CSS-tyylejä omaan ”*custom.css”*-tiedostoon keräämällä halutut muokattavat tyyliluokat niiden alkuperäisistä CSS-tiedostoista. Esimerkiksi portlettien reunaviivojen poistaminen on toteutettu kopioimalla luokan *”.portlet”* tyylit suoraan *”portlet.css”*-tiedostosta *”custom.css”*-tiedostoon ja ylikirjoittamalla ominaisuudet halutun kaltaiseksi. Samalla periaatteella on toteutettu muiden valmiiden elementtien ulkonäön muokkaus. Muutokset pitää muistaa tehdä ohjeiden mukaisesti **_diffs**-kansioon. Yleisen teeman muokkauksessa on käytetty graafikon graafisia ohjeita ja typografiaa (kuva 34). Kaikki fontteihin liittyvät tyylitt on muokattu *”fonts.css”*- tiedostossa.



KUVA 34. Graafikon luonnos sivuston yleisestä ilmeestä ja listanäkymästä

Kehityksen aikana palvelun graafinen ilme on kokenut jatkuvia pieniä muutoksia, mutta yleinen värimaailma on suurin piirtein sama. Tämän hetkinen teema on jo käynyt läpi useita muutoksia ensimmäisistä luonnoksista ja kokeiluista (vrt kuva 34 & kuva 35).



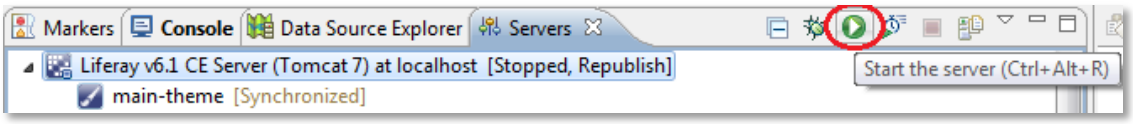
KUVA 35. Sivuston tämän hetkinen kehitteillä oleva teema ilman portletteja

Teeman graafista ilmettä toteutetaan jatkuvasti yhteistyössä graafikon kanssa, joten alkuperäisen luonnoksen jälkeen uusia luonnoksia ei ole ollut syytä tehdä.

4.4.2 Teeman käyttöönotto

Kun teema on luotu, se pitää ottaa käyttöön. Teemat otetaan käyttöön Liferayn hallintapaneelista, joka on näkyvissä kaikille ylläpitotason omaaville käyttäjille. Teema on käytettävissä, kun se on liitetty serveriin (ks. kuva 31 sivulla 30).

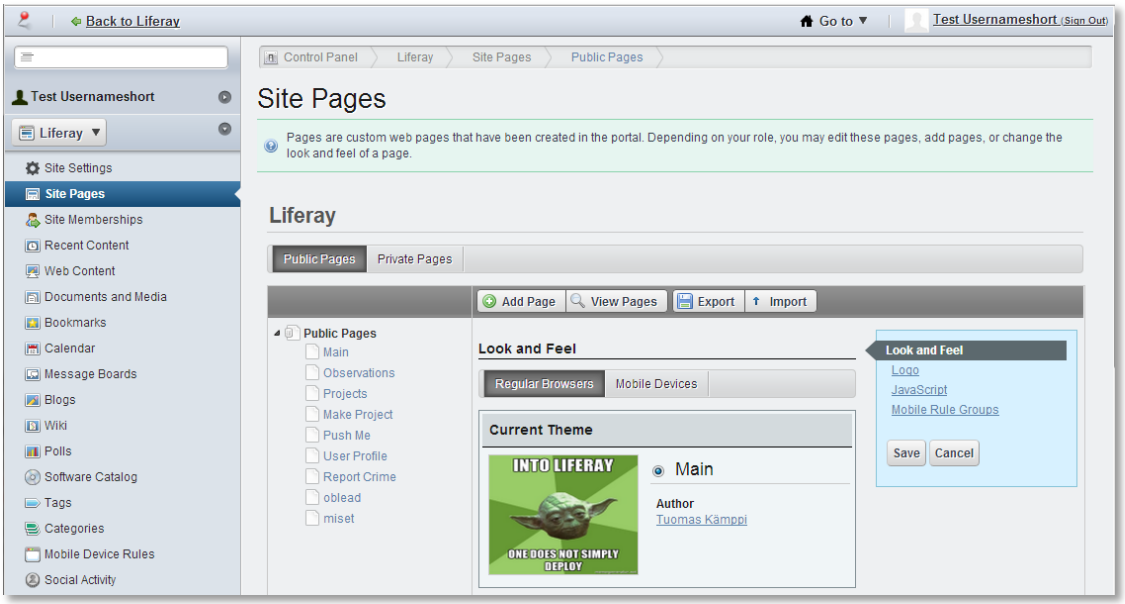
Tämän jälkeen Liferay IDE-laajennus luo automaattisesti teemalle kansion serverille: **C:\Liferay\bundles\liferay-portal-6.1.1-ce-ga2\tomcat-7.0.27\webapps\main-theme,** joka on nyt käytettävissä. Liferay-serveri käynnistetään valitsemalla se Eclipsen alalaidasta *"Servers"*-välilehdeltä ja painamalla vihreää käynnistyspainiketta (tai *Ctrl+Alt+R*) (kuva 36).



KUVA 36. Liferay-serverin käynnistäminen

Serverin käynnistyminen kestää muutaman minuutin, riippuen lisäosien (portlettien, teemojen hookien jne.) lukumäärästä. Välilehdeltä *"Console"* pystytään seuraamaan käynnistymisen edistymistä (kuva 36). Serveri on käynnissä, kun serverin perässä lukee *"Started"*. Tämän jälkeen Liferay voidaan avata kirjoittamalla osoiteriville: **localhost:8080**. Hetken päästä Liferay on käyttövalmiina.

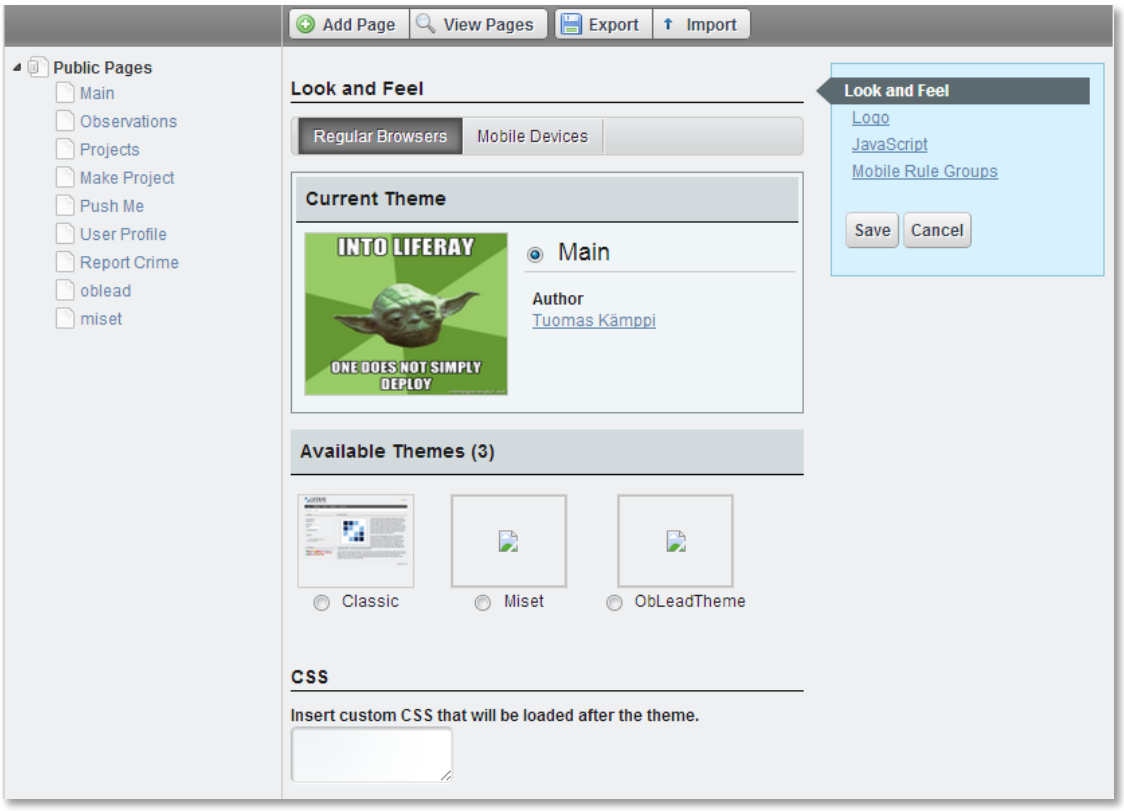
Hallintapaneeli avataan Liferayn ylälaidassa olevasta hallintatyökalupalkista (ks. kuva 21 sivulla 20). Hallintapaneelin (Control Panel) sisältä valitaan vasemmalta valikosta Site Pages. Site Pages vie julkaisujärjestelmän sivustonhallintaan (kuva 37). Sivustonhallinnata voidaan lisätä, poistaa, hallita käyttöoikeuksia ja vaihtaa teemoja.



KUVA 37. Hallintapaneelin sivustonhallinta

Vasemmalla **Public Pages**-välilehden alla on listattu kaikki sivuston tämän hetkiset sivut. Jokaiselle sivulle voidaan myös tarvittaessa määritellä oma teemansa. **"Look and Feel"**-kohdan alta voidaan valita normaalien selainten **"Regular Browsers"** ja mobiililaitteiden **"Mobile Devices"** teemat. Mobiililaitteille tarkoitettu teema on kuitenkin tämän opinnäytetyön kannalta turha, koska *"main-theme"*-teemapaketti on toteutettu responsiivisen verkkosuunnittelun periaatteella. Tämä ominaisuus on

todennäköisesti jäänne ajalta ennen mediakyselyitä. Aloitusnäkymässä näkyy tämän hetkinen, eli **”Current Theme”** ja saatavilla olevat teemat, eli **”Available Themes”** (kuva 38).



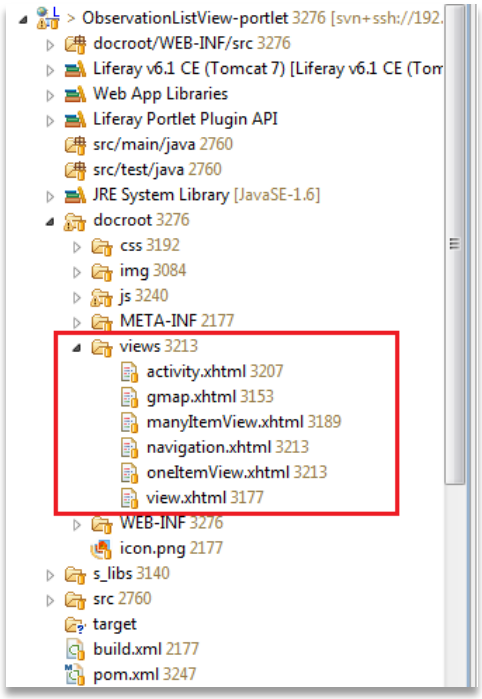
KUVA 38. Teeman valinta

Teeman valinnan jälkeen teema otetaan lopullisesti käyttöön tallentamalla kohdasta **”Save”**. Muutokset näkyvät heti palaamalla takaisin sivustolle painamalla ylälaidassa olevaa *”Back to Liferay”*-linkkiä (ks. kuva 37 sivulla 34).

4.5 Portletin rakenteen ja ulkoasun toteutus

Koska portletit ovat itsenäisiä verkkosovelluksia, niille on määriteltävä omat ulkoasun rakenteet ja tyylit normaalin verkkosivun tapaan. Muokattavan ObservationListView-portlet:n ulkoasun määrittelevät tiedostot on toteutettu XHTML:llä. Lisäksi käyttöliittymäelementtien toteutuksessa on käytetty PrimeFaces-käyttöliittymäkirjastoa. Projektin ohjelmoijat ovat toteuttaneet portletin toiminnallisuuden ja perusrakenteen rautalankamallien ja spesifikaatioiden mukaan. Tehtäväni on muokata ulkoasu vastaamaan graafikon antamaa ohjeistoa (ks kuva 34 sivulla 32).

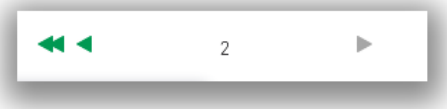
Ohjelmoijalta saamassani *ObservationListView-portlet*-paketissa on ulkoasun kannalta oleellinen kansio *”views”*. Tämä kansio sisältää kaikki portletin näkymät, joista olen ensimmäisenä lähtenyt muokkaamaan listanäkymää (kuva 39).



KUVA 39. ObservationListView-portlet:n kansiorakenne

Listanäkymä sijaitsee *”manyItemView.xhtml”*-tiedostossa. Käyttöliittymän rakenteissa on käytetty esimerkiksi normaalien HTML *” ”*- ja *””*-elementtien lisäksi PrimeFaces-elementtejä (kuva 40).

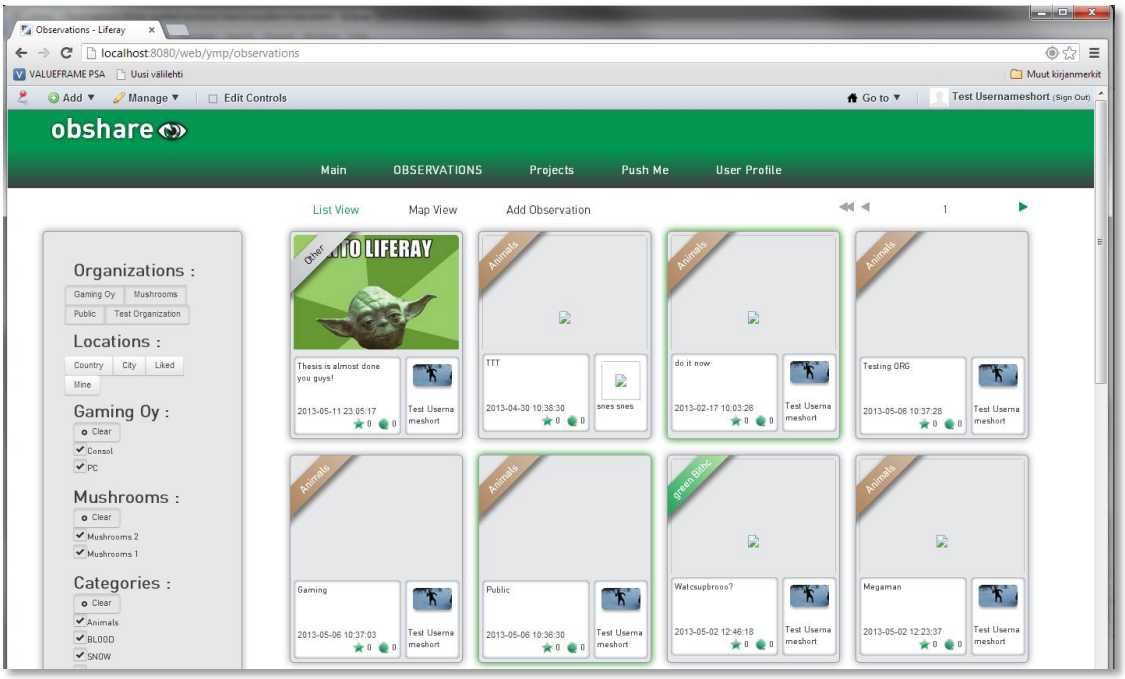
```
<div class="normal_navigation">
    <pf:commandLink rendered="#{observationPage.previousBtn}"
        actionListener="#{observationPage.first}"
        update=":form:manyItemTab">
        <span class="first_button"></span>
    </pf:commandLink>
    <pf:commandLink rendered="#{!observationPage.previousBtn}">
        <span class="empty_first"></span>
    </pf:commandLink>
    <pf:commandLink rendered="#{observationPage.previousBtn}"
        actionListener="#{observationPage.previous}"
        update=":form:manyItemTab">
        <span class="previous_button"></span>
    </pf:commandLink>
    <pf:commandLink rendered="#{!observationPage.previousBtn}">
        <span class="empty_previous"></span>
    </pf:commandLink>

    <span class="light_big">
        <h:outputText value="#{observationPage.page + 1}" />
    </span>

    <pf:commandLink rendered="#{observationPage.nextBtn}"
        actionListener="#{observationPage.next}" update=":form:manyItemTab">
        <span class="next_button"></span>
    </pf:commandLink>
    <pf:commandLink rendered="#{!observationPage.nextBtn}">
        <span class="empty_next"></span>
    </pf:commandLink>
</div>
```



KUVA 40. HTML-elementtien ja PrimeFaces-elementtien käyttö listan selailupainikkeiden toteutuksessa

Yllä olevassa kuvassa näkyy listanäkymän havaintolistan selailupainikkeiden toteutus *”manyItemView.xhtml”*-tiedostossa. Ulkoasun muokkaus on pääosin tehty lisäämällä *””*-elementit ja tyyliluokat *(class)* PrimeFaces elementtien ympärille, tai päinvastoin. Esimerkiksi *”<pf:commandLink>”* elementti on käytännössä HTML *”<a>”* linkkielementti, jonka sisään on luotu oma *””* elementti, jonka luokan *(class)* avulla on toteutettu painikkeen ulkoasu. PrimeFaces-elementeille voidaan lisäksi antaa suoraan omia tyyliluokkia: *styleClass=”luokan_nimi”*.

Listanäkymä koostuu kehityksen testausvaiheessa eri kehittäjien lähettämistä testihavainnoista. Ainoastaan omasta kehitysympäristöstä lähetetyt kuvat näkyvät testiympäristössä (kuva 41).



KUVA 41. Testiympäristön testihavainnot.

Yllä olevassa kuvassa 41 näkyy listanäkymän tämän hetkinen toteutus. Perusrakenteet ovat jo valmiina, joten tämän jälkeen olen siirtynyt testaamaan ja toteuttamaan näkymän muokkaamista mobiililaiteystävälliseksi.

4.6 Responsiivinen toteutus

Sivuston toteutus mobiililaiteystävälliseksi on toteutettu käyttämällä Liferayn sisäänrakennettua **AlloyUI** Jquery-kirjaston näyttökokojen tunnistamiseen käytettävää moduulia: **aui-viewport**. Moduuli perustuu 960 grideihin, joka tunnistaa, onko katseltavan selaimen piirtoalueen leveys joko **320px**, **480px**, **720px** tai **960px**. Moduuli otetaan käyttöön lisäämällä *”AUI().use(’aui-viewport’)”* minne tahansa kohtaan sivua (kuva 42). Lisäsin pätkän heti muiden JavaScriptien perään *”main-theme”*:n *”portal_normal.vm”* -tiedoston alkuun (Cavanaugh 2011).

```
17    <script type="text/javascript" charset="utf-8">
18    AUI().use('aui-viewport');
19    </script>
```

KUVA 42. AlloyUI aui-viewport:n käyttöönotto

Tämän jälkeen moduuli *”aui-viewport”* on käytössä ja sen sisältämiä luokkia voidaan kutsua CSS-tiedostossa. Sen lisäksi, että moduuli tunnistaa näyttökoot, se sisältää

luokkia, joilla voidaan määrittää tyylejä esimerkiksi näytöille, jotka ovat pienempiä kuin 960 pikseliä. Taulukossa 3 on lista kaikista moduulin käytettävistä CSS-luokista (class).

TAULUKKO 3. "aui-viewport" -moduulin CSS-luokat

Luokka (class):	Määritelmä:
.aui-view-320	320px leveät näytöt
.aui-view-480	480px leveät näytöt
.aui-view-720	720px leveät näytöt
.aui-view-960	960px leveät näytöt
.aui-view-lt320	320px kapeammat näytöt
.aui-view-lt480	480px kapeammat näytöt
.aui-view-lt720	720px kapeammat näytöt
.aui-view-lt960	960px kapeammat näytöt
.aui-view-gt320	320px leveämmät näytöt
.aui-view-gt480	480px leveämmät näytöt
.aui-view-gt720	720px leveämmät näytöt
.aui-view-gt960	960px leveämmät näytöt

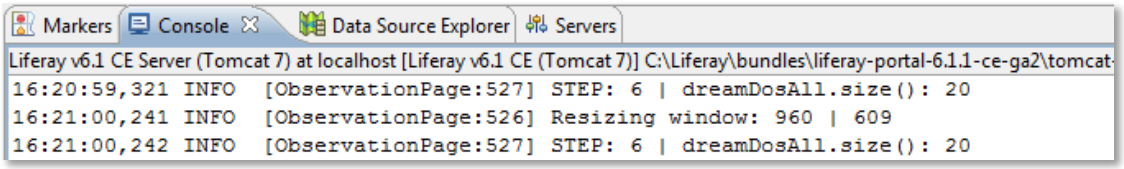
Loin teemaan main-theme uuden CSS-tiedoston: **mobile.css**. Selkeyden vuoksi olen toteuttanut kaikki mobiililaitteille räätälöidyt CSS-tyylit yhteen tiedostoon sen sijaan, että olisin lisännyt ne portletin omien tyylitiedostojen, esim. *"listview.css"*-tiedoston sisään. Tiedosto liitetään yhteen muiden teeman CSS-tiedostojen kanssa tiedostossa *"main.css"* (kuva 43).

```
@import url(base.css);

@import url(application.css);

@import url(layout.css);

@import url(dockbar.css);

@import url(navigation.css);

@import url(portlet.css);

@import url(forms.css);

@import url(extras.css);

@import url(custom.css);

@import url(login.css);

@import url(userpage.css);

@import url(listview.css);

@import url(ribbons.css);

@import url(reportcrime.css);

@import url(fonts.css);

@import url(mobile.css);

/* end */
```

KUVA 43. Teeman main-theme main.css

Koska kaikki Liferayn teemapaketin CSS-tiedostot liitetään yhteen main.css-tiedostossa, olen varmistanut sen, että *”mobile.css”*-tiedosto ylikirjoittaa kaikki tarvittavat säännöt lisäämällä sen aivan listan viimeiseksi

Sivuston reagointia eri selainten, eli laitteiden leveyksiin on helppoa testata muuttamalla selaimen kokoa. Kehitystä helpottaa ObservationListView-portletin lähettämät selaimen leveystiedot Eclipsen konsoliin (Console). ObservationListView-portlet sisältää ohjelman, joka tutkii, minkä kokoinen selain on ja määrittää sen mukaan listassa olevien havaintojen lukumäärän (kuva 44).



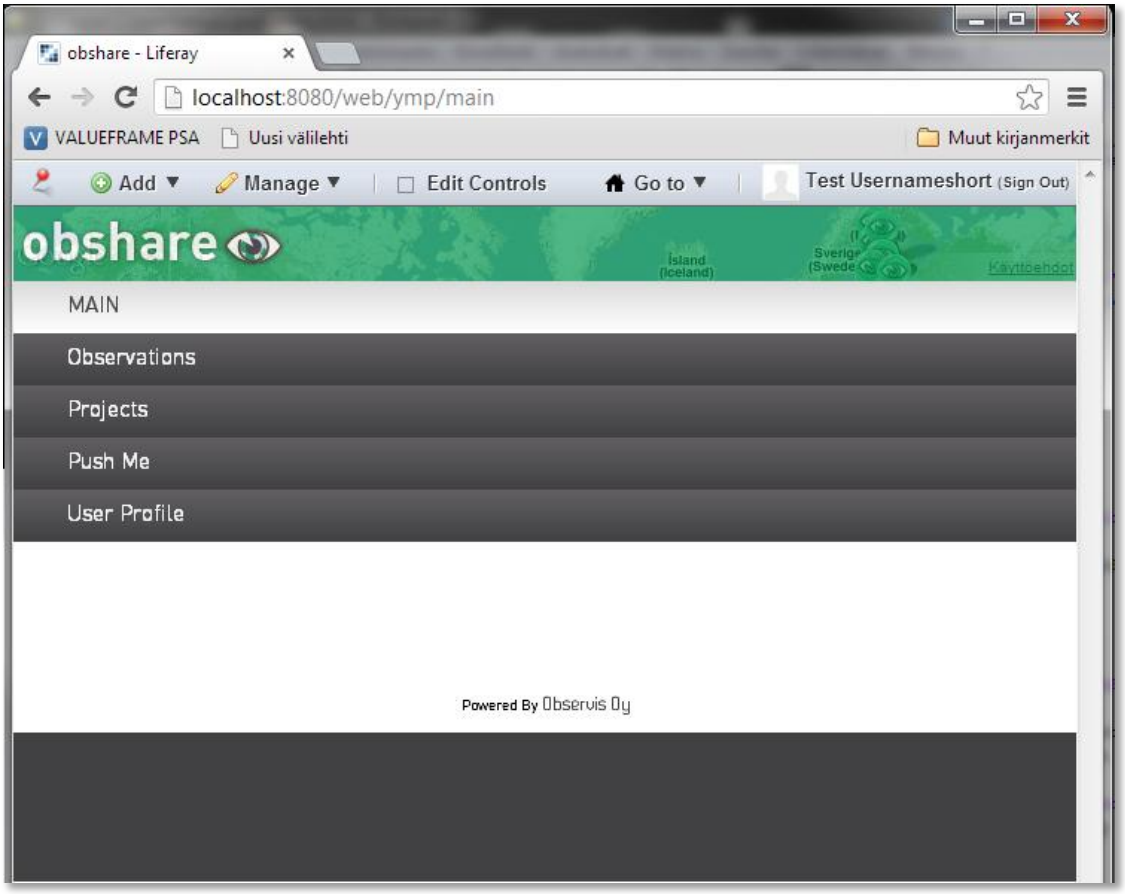
KUVA 44. ObservationListView-portletin lähettämät selaimen leveystiedot Eclipsen konsolissa

Muuttamalla selaimen kokoa, ohjelma lähettää tietoa selaimen sen hetkisestä koosta Eclipsen konsoliin. Yllä olevassa kuvassa 44 *”Resizing window”*-tekstin jälkeen

ensimmäinen luku *”960”* on selaimen leveys ja jälkimmäinen *”609”* on selaimen korkeus.

4.6.1 Navigaatio

Ensimmäisenä keskityn portaalin yleisen ilmeen, eli teeman main-theme muokkaamiseen responsiiviseksi. Keskeisimpänä ja huomattavimpana ominaisuutena on mobiililaitteiden sovelluksille tyypillinen navigaatiopaneeli, joka tulee näkyviin, kun selaimen koko on alle 720 kuvapistettä (kuva 45).



KUVA 45. Alle 720 kuvapisteen laitteiden navigaatio

Navigaation toteutuksessa käytetään ensin **aui-viewport**-luokkaa: .**aui-view-lt720**. Sen perään otetaan ylikirjoitettava tyyliluokka (class) tai id (#) käsittelyyn, kuten kuvassa 46.

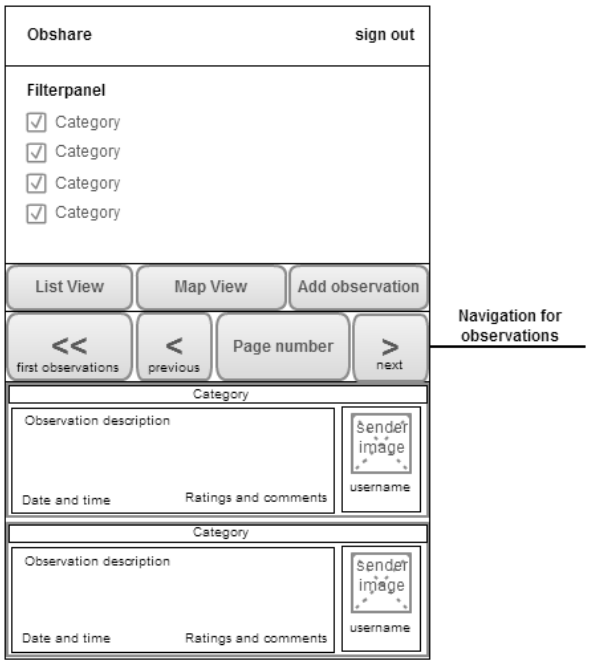
```
133 .aui-view-lt720 #navigation {
134     height: 0px;
135     background: none;
136     text-align: left;
137     padding: 0;}
```

KUVA 46. Alle 720 kuvapisteen näytön tyylien ylikirjoittaminen

Yllä olevan kuvan sääntö ylikirjoittaa kaikki hakasulkujen välissä olevat *"#navigation"* id:n omaavan elementin tyylit. Koko mobiilinavigaation CSS-toteutus on liitteessä 7. Muita muutoksia perusrakenteessa ovat madallettu *"<header>"*-osio ja pienennetty sivuston logo.

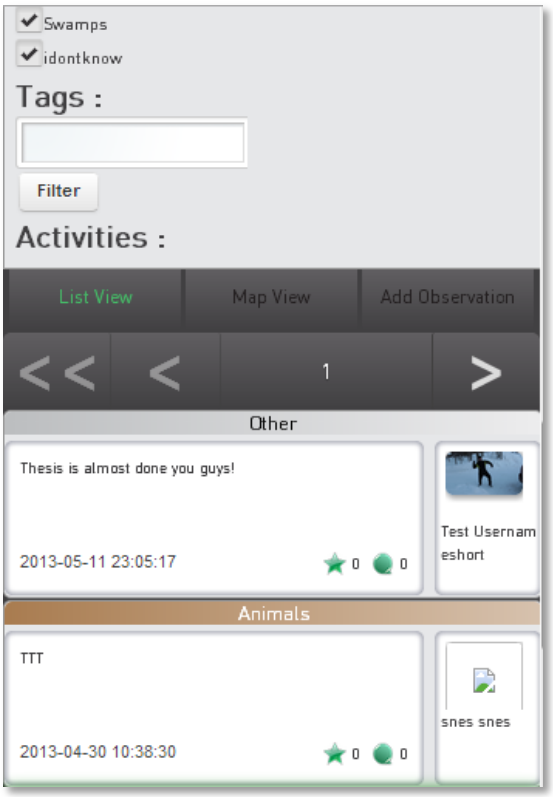
4.6.2 Listanäkymä

Portaalin yleisen ilmeen ja navigaatiopalkin muokkaamisen jälkeen siirrytään muokkaamaan itse listanäkymää. Navigaatio toimii ylhäällä hyvin, mutta havaintokortit alkavat muuttua pienillä näytöillä liian isoiksi, lisäksi vasemmanpuoleinen suodatuspaneeli jää listanäkymän kanssa päällekkäin. Haluan näytölle mahtuvan mahdollisimman monta havaintokorttia, joista pystyy vielä lukemaan havainnon perustiedot: lähettäjä, lähettäjän kuva, havainnon lisäysaika, havainnon kommenttien ja arviointipisteiden määrä, sekä itse havainnon kuvauksen. Pienillä näytöillä havaintokortin kuva piilotetaan kokonaan näkyvistä (kuva 47).



KUVA 47. Listanäkymän mobiilikäyttöliittymän rautalankamalli

Rautalankamallin mukaan olen muokannut listanäkymän navigaation ja havaintojen selailupainikkeet mobiilisovellusmaisesti listan yläpuolelle. Havaintojen suodattamisen olen siirtänyt ensimmäiseksi (kuva 48).



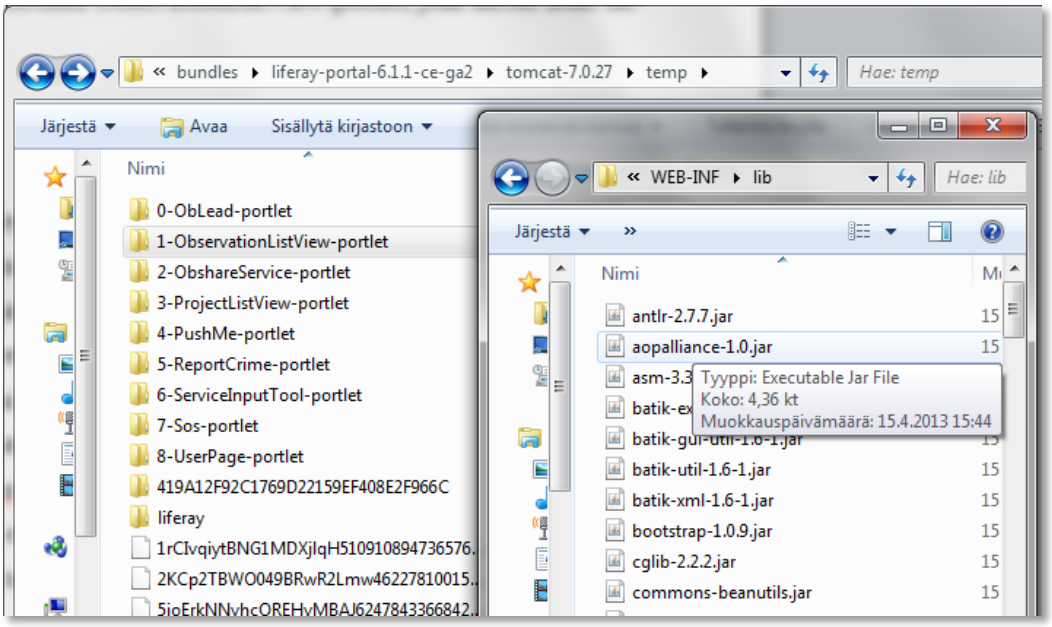
KUVA 48. Listanäkymän toteutus alle 480 kuvapisteen näytöille

Ensimmäisenä olen lähtenyt toteuttamaan listanäkymää alle 480 kuvapisteen levyisille näytöille (LIITE 8). Toteutus kaipaa vielä viimeistelyä, mutta suunta on oikea. Samaa toteutusperiaatetta jatketaan muiden näyttökokojen optimoinnissa.

4.7 Kehityshuomioita

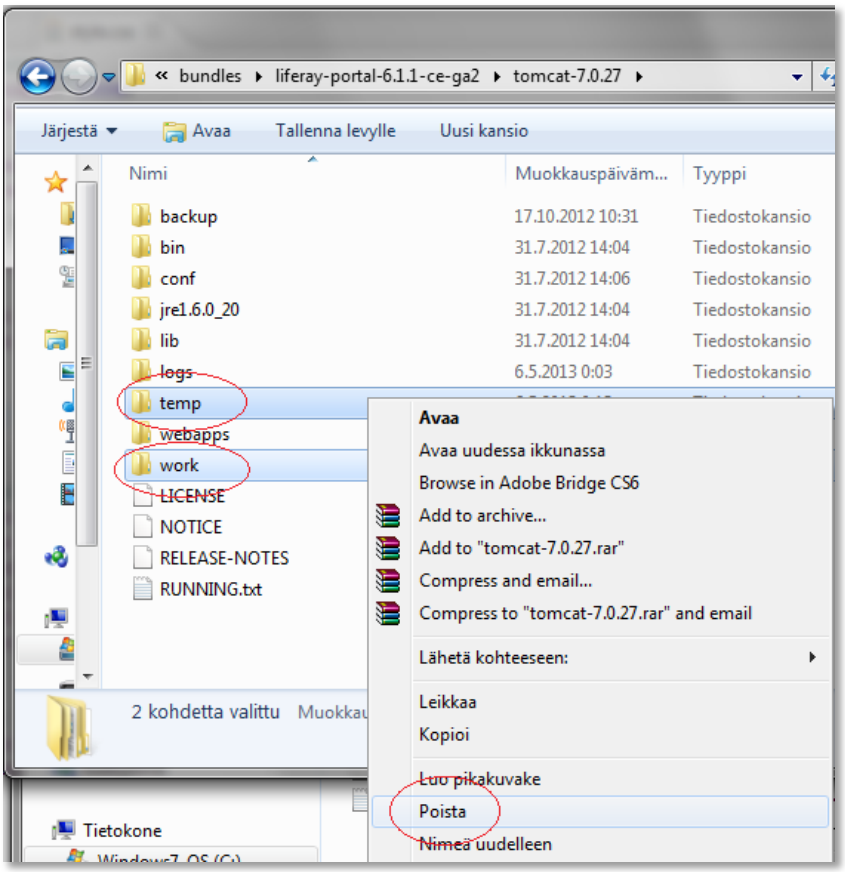
Suurimpia haasteita kehityksen alkuvaiheilla oli portlettien ja teemojen käyttöönotto päivityksien jälkeen. Kehitystyön aikana portettien sisällöt ja teeman rakenteet muuttuivat jatkuvasti. Pienet päivitykset aiheuttivat kuitenkin ongelmia niitä päivitettäessä Liferay-serverille. Päivitetyt portletit ja teemat aiheuttivat sen, että Liferay-serveri ei käynnistynyt kunnolla tai ei ollenkaan. Poistamalla lisäosat (portetit ja teemat) serveriltä, serveri käynnistyi, mutta takaisin lisättäessä, toiminta lakkasi. Myöskään serverin puhdistaminen (clean) ei auttanut. Syyksi paljastui lopulta Liferayn Tomcat 7-serverin **temp-** ja **work**-kansioiden sisältämät tiedostot.

Kansio *”temp”* sisältää jokaisen serverille ladatun lisäosan *”.jar”*-tiedostot (kuva 49). Liferay ei jostain syystä aina korvaa kaikkia vanhoja tiedostoja uusilla, mikä aiheuttaa joissain tapauksissa yhteentörmäyksiä. Poistamalla molemmat kansiot serveriltä, Liferay v6.1 Tomcat 7-serveri luo uudet *”temp”*- ja *”work”*-kansiot varmasti ajan tasalla olevilla tiedostoilla (kuva 50).



KUVA 49. Tomcat 7 serverin temp kansio

Kansiot *”temp”*- ja *”work”* löytyvät Liferayn asennuskansion alta. Jos Liferay on asennettu C-juureen, on polku kansioihin silloin: **C:\Liferay\bundles\liferay-portal-6.1.1-ce-ga2\tomcat-7.0.27**.



KUVA 50. Temp- ja work-kansioiden poisto serveriltä

Käytäntönä on ollut poistaa aina jokaisen suuren päivityksen jälkeen *”temp”*- ja *”work”*-kansiot kokonaisuudessaan varmuuden vuoksi. Kehityksen alkuvaiheilla suurin osa ajasta meni serverin tyhjentämiseen kaikista lisäosista, sen puhdistamiseen ja uudelleen käynnistämiseen.

5 PÄÄTÄNTÖ

Opinnäytetyön tavoitteena oli oppia toteuttamaan Liferayn teemoja ja oppia hallitsemaan toteutuksessa käytettyjä kehitysvälineitä ja kehitysympäristöä. Tavoitteena oli lisäksi toteuttaa obshare.com-verkkopalvelulle teema, joka noudattaa responsiivisen verkkosuunnittelun periaatteita. Alussa opettelua ja toteutusta hidastivat kehitys- ja testausympäristön käyttöön liittyvät ongelmat, joiden ratkaisussa kesti kehitystiimiltä jonkun aikaa. Tämä johtui siitä, että kenelläkään ei ollut aikaisempaa kokemusta Liferay-julkaisujärjestelmän käytöstä, saati sen kehitysympäristöstä. Alkuvaikeuksien jälkeen kehitys lähti kuitenkin nopeasti käyntiin.

Kehittämistehtävänä olevan teeman toteutusta opetellessa Liferayn kirjallisista lähteistä ja kotisivuista oli paljon hyötyä. Teemoja tuli luotua useita ja yritysten ja erehdysten kautta pääsin lopulta viimeisen, lopullisen teeman luomiseen. Teeman toteutuksessa kaikkein uusin ja tuntemattomin alue oli Apachen Velocity-merkkauskieli, jota käytettiin julkaisujärjestelmän sivujen HTML-rakenteiden rakentamiseen. Teeman ulkoasun tyylien määrittäminen oli täysin normaalia CSS-tyylien käyttöä. Responsiivisen toteutuksen keskeisenä apuvälineenä oli Ethan Marcotten esimerkit ja Nate Cavanaughin blogikirjoitukset. Teeman käyttöönotto ja testaus vaati lisäksi itse julkaisujärjestelmän hallintapaneelin käytön opettelua.

Opinnäytetyön lopputuloksena on teema, jonka pohjalta on hyvä lähteä rakentamaan palvelun lopullista ulkoasua. Tässä opinnäytetyössä esiteltyjä toteutustapoja tullaan käyttämään muiden sivuston sivujen käyttöliittymän toteutuksessa. Opinnäytetyön tietoja voidaan hyödyntää myös yrityksen muiden Liferay-julkaisujärjestelmään pohjautuvien verkkopalveluiden teemojen toteuttamisessa. Opinnäytetyössä esiteltyä responsiivisen verkkosuunnittelun mukaan rakennettua listanäkymän esimerkkiä tullaan käyttämään muidenkin palvelun sivujen toteutuksessa.

Opinnäytetyön aikana olen oppinut toteuttamaan teemoja Liferay-julkaisujärjestelmään, sekä oppinut hyödyllisiä tekniikoita responsiivisen verkkosuunnittelun toteutukseen. Erityisesti Liferayn mukana tulleen AlloyUI-kirjaston hyödyntäminen oli opettava kokemus. Opinnäytetyössä opittuja taitoja tullaan käyttämään ja hiomaan entisestään obshare.com-verkkopalvelun kehityksen

jatkuessa ja Observis Oy:n muissa Liferayhin pohjautuvissa toteutuksissa. Lisäksi kehitystyön aikana saatu oppi kansainvälisestä tiimityöskentelystä WEB-kehittäjänä yhdessä graafikon, suunnittelijoiden ja ohjelmoijien kanssa on ollut hyvin antoisaa ja opettavaista.

LÄHTEET

Auchenberg, Kenneth 2012. Word wrapping/hyphenation using CSS. WWW-dokumentti. http://blog.kenneth.io/blog/2012/03/04/word-wrapping-hypernation-using-css/. Päivitetty 4.3.2012. Luettu 14.1.2013.

Bosomworth, Danyl 2013. Mobile Marketing Statistics 2013. WWW-dokumentti. http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/. Päivitetty 8.1.2013. Luettu 5.2.2013.

Cavanaugh, Nate 2011. Liferay.com, mobile sites and responsive layouts. WWW-dokumentti. http://www.liferay.com/web/nathan.cavanaugh/blog/-/blogs/8907527. 22.5.2011. Luettu 20.2.2013.

Çivici, Çağatay 2013. User's guide 3.4. PDF. primefaces_users_guide_3_4.pdf. Luettu 1.5.2013.

Kanerva, Ville 2013. Haastattelu 25.4.2013. Tekninen johtaja, perustaja. Observis Oy.

Kotilainen, Samuli 2013. Googlea lähestytään yhä useammin mobiilisti. Tietokone 1, 15.

Liferay. About Portlets 2013. WWW-dokumentti. http://www.liferay.com/community/wiki/-/wiki/Main/About+Portlets. Luettu 2.5.2013

Liferay. Get Liferay Portal 2013. WWW-dokumentti. http://www.liferay.com/downloads/liferay-portal/available-releases. Luettu 6.5.2013.

Liferay. Layout Template 2013. WWW-dokumentti. http://www.liferay.com/community/wiki/-/wiki/Main/Layout+Template. Luettu 14.2.2013.

Liferay. Liferay IDE Features 2013. WWW-dokumentti. http://www.liferay.com/community/liferay-projects/liferay-ide/features. Luettu 7.5.2013.

Liferay. Liferay Portal 6.1 – Development Guide, Creating a Portlet 2013. WWW-dokumentti. http://www.liferay.com/documentation/liferay-portal/6.1/development/-/ai/creating-a-portl-3. Luettu 2.5.2013.

Liferay. Liferay Portal 6.1 – Development Guide, Creating Liferay Themes 2013. WWW-dokumentti. http://www.liferay.com/documentation/liferay-portal/6.1/development/-/ai/creating-liferay-them-7. Luettu 13.1.2013.

Liferay. What is a Portal? 2013. WWW-dokumentti. http://www.liferay.com/products/what-is-a-portal/web-platform. Luettu 14.3.2013.

Liferay. Wiki, Friendly URLs 2013. WWW-dokumentti. http://www.liferay.com/community/wiki/-/wiki/Main/Friendly+URLs. Luettu 10.5.2013.

Marcotte, Ethan 2011. Responsive Web Design. New York: A Book Apart.

Martin, Michael 2012. Less Than 10% Of The Web In 2012 Is Mobile Ready. WWW-dokumentti. http://searchengineland.com/less-than-10-of-the-web-in-2012-is-mobile-ready-112101. Päivitetty 20.2.2012. Luettu 5.2.2013.

Pingdom 2012. Mobile share of web traffic in Asia has tripled since 2010. WWW-dokumentti. http://royal.pingdom.com/2012/05/08/mobile-web-traffic-asia-tripled/. Päivitetty 8.5.2012. Luettu 5.2.2013.

PrimeFaces. Getting Started 2013. WWW-dokumentti. http://primefaces.org/gettingStarted.html. Luettu 1.5.2013.

Sezov, Richard Jr, Hinkey, Jim, Kostas, Stephen, Rao, Jesse & Hong, Cody 2013.

Using Liferay Portal 6.1. PDF. Using Liferay Portal 6.1.pdf. Luettu 12.5.2013.

Sezov, Richard Jr. 2012. Liferay In Action. Shelter Island, NY: Manning Publications Co.

Shaikh, Wasim 2011. WWW-dokumentti. http://blog.wasimshaikh.com/2011/09/01/preview-html5-responsive-liferay-theme/#. Päivitetty 1.9.2011. Luettu 23.1.2013.

W3Schools. CSS Properties 2013. WWW-dokumentti. http://www.w3schools.com/cssref/default.asp. Luettu 19.4.2013.

W3Schools. CSS Selectors 2013. WWW-dokumentti. http://www.w3schools.com/cssref/css_selectors.asp. Luettu 19.4.2013.

W3Schools. CSS3 Tutorial 2013. WWW-dokumentti. http://www.w3schools.com/css3/default.asp Luettu 22.4.2013.

W3Schools. HTML – XHTML 2013. WWW-dokumentti. http://www.w3schools.com/html/html_xhtml.asp. Luettu 22.4.2013.

W3Schools. HTML Introduction 2013. WWW-dokumentti. http://www.w3schools.com/html/html_intro.asp. Luettu 22.4.2013.

W3Schools. XML Tutorial 2013. WWW-dokumentti. http://www.w3schools.com/xml/default.asp. Luettu 22.4.2013.

Yuan, Jonas X, Chen, Xinsheng & Yu, Frank 2010. Liferay User Interface Development. Birmingham: Packt Publishing Ltd.

LIITE 1(1).

CSS3 ominaisuuslista (W3Schools. CSS Properties)

Property	Description	CSS
@keyframes	Specifies the animation	3
animation	A shorthand property for all the animation properties below, except the animation-play-state property	3
animation-name	Specifies a name for the @keyframes animation	3
animation-duration	Specifies how many seconds or milliseconds an animation takes to complete one cycle	3
animation-timing-function	Specifies the speed curve of the animation	3
animation-delay	Specifies when the animation will start	3
animation-iteration-count	Specifies the number of times an animation should be played	3
animation-direction	Specifies whether or not the animation should play in reverse on alternate cycles	3
animation-play-state	Specifies whether the animation is running or paused	3
background-clip	Specifies the painting area of the background	3
background-origin	Specifies the positioning area of the background images	3
background-size	Specifies the size of the background images	3
border-bottom-left-radius	Defines the shape of the border of the bottom-left corner	3
border-bottom-right-radius	Defines the shape of the border of the bottom-right corner	3
border-image	A shorthand property for setting all the border-image-* properties	3
border-image-outset	Specifies the amount by which the border image area extends beyond the border box	3
border-image-repeat	Specifies whether the image-border should be repeated, rounded or stretched	3
border-image-slice	Specifies the inward offsets of the image-border	3
border-image-source	Specifies an image to be used as a border	3
border-image-width	Specifies the widths of the image-border	3
border-radius	A shorthand property for setting all the four border-*-radius properties	3
border-top-left-radius	Defines the shape of the border of the top-left corner	3
border-top-right-radius	Defines the shape of the border of the top-right corner	3
box-decoration-break		3

LIITE 1(2).

CSS3 ominaisuuslista (W3Schools. CSS Properties)

box-shadow	Attaches one or more drop-shadows to the box	3
overflow-x	Specifies whether or not to clip the left/right edges of the content, if it overflows the element's content area	3
overflow-y	Specifies whether or not to clip the top/bottom edges of the content, if it overflows the element's content area	3
overflow-style	Specifies the preferred scrolling method for elements that overflow	3
rotation	Rotates an element around a given point defined by the rotation-point property	3
rotation-point	Defines a point as an offset from the top left border edge	3
color-profile	Permits the specification of a source color profile other than the default	3
opacity	Sets the opacity level for an element	3
rendering-intent	Permits the specification of a color profile rendering intent other than the default	3
bookmark-label	Specifies the label of the bookmark	3
bookmark-level	Specifies the level of the bookmark	3
bookmark-target	Specifies the target of the bookmark link	3
float-offset	Pushes floated elements in the opposite direction of the where they have been floated with float	3
hyphenate-after	Specifies the minimum number of characters in a hyphenated word after the hyphenation character	3
hyphenate-before	Specifies the minimum number of characters in a hyphenated word before the hyphenation character	3
hyphenate-character	Specifies a string that is shown when a hyphenate-break occurs	3
hyphenate-lines	Indicates the maximum number of successive hyphenated lines in an element	3
hyphenate-resource	Specifies a comma-separated list of external resources that can help the browser determine hyphenation points	3
hyphens	Sets how to split words to improve the layout of paragraphs	3
image-resolution	Specifies the correct resolution of images	3
marks	Adds crop and/or cross marks to the document	3
string-set		3
box-align	Specifies how to align the child elements of a box	3
box-direction	Specifies in which direction the children of a box are displayed	3

LIITE 1(3).

CSS3 ominaisuuslista (W3Schools. CSS Properties)

box-flex	Specifies whether the children of a box is flexible or inflexible in size	3
box-flex-group	Assigns flexible elements to flex groups	3
box-lines	Specifies whether columns will go onto a new line whenever it runs out of space in the parent box	3
box-ordinal-group	Specifies the display order of the child elements of a box	3
box-orient	Specifies whether the children of a box should be laid out horizontally or vertically	3
box-pack	Specifies the horizontal position in horizontal boxes and the vertical position in vertical boxes	3
@font-face	A rule that allows websites to download and use fonts other than the "web-safe" fonts	3
font-size-adjust	Preserves the readability of text when font fallback occurs	3
font-stretch	Selects a normal, condensed, or expanded face from a font family	3
crop	Allows a replaced element to be just a rectangular area of an object, instead of the whole object	3
move-to	Causes an element to be removed from the flow and reinserted at a later point in the document	3
page-policy	Determines which page-based occurance of a given element is applied to a counter or string value	3
grid-columns	Specifies the width of each column in a grid	3
grid-rows	Specifies the height of each column in a grid	3
target	A shorthand property for setting the target-name, target-new, and target-position properties	3
target-name	Specifies where to open links (target destination)	3
target-new	Specifies whether new destination links should open in a new window or in a new tab of an existing window	3
target-position	Specifies where new destination links should be placed	3
alignment-adjust	Allows more precise alignment of elements	3
alignment-baseline	Specifies how an inline-level element is aligned with respect to its parent	3
baseline-shift	Allows repositioning of the dominant-baseline relative to the dominant-baseline	3
dominant-baseline	Specifies a scaled-baseline-table	3
drop-initial-after-adjust	Sets the alignment point of the drop initial for the primary connection point	3

LIITE 1(4).

CSS3 ominaisuuslista (W3Schools. CSS Properties)

drop-initial-after-align	Sets which alignment line within the initial line box is used at the primary connection point with the initial letter box	3
drop-initial-before-adjust	Sets the alignment point of the drop initial for the secondary connection point	3
drop-initial-before-align	Sets which alignment line within the initial line box is used at the secondary connection point with the initial letter box	3
drop-initial-size	Controls the partial sinking of the initial letter	3
drop-initial-value	Activates a drop-initial effect	3
inline-box-align	Sets which line of a multi-line inline block align with the previous and next inline elements within a line	3
line-stacking	A shorthand property for setting the line-stacking-strategy, line-stacking-ruby, and line-stacking-shift properties	3
line-stacking-ruby	Sets the line stacking method for block elements containing ruby annotation elements	3
line-stacking-shift	Sets the line stacking method for block elements containing elements with base-shift	3
line-stacking-strategy	Sets the line stacking strategy for stacked line boxes within a containing block element	3
text-height	Sets the block-progression dimension of the text content area of an inline box	3
marquee-direction	Sets the direction of the moving content	3
marquee-play-count	Sets how many times the content move	3
marquee-speed	Sets how fast the content scrolls	3
marquee-style	Sets the style of the moving content	3
column-count	Specifies the number of columns an element should be divided into	3
column-fill	Specifies how to fill columns	3
column-gap	Specifies the gap between the columns	3
column-rule	A shorthand property for setting all the column-rule-* properties	3
column-rule-color	Specifies the color of the rule between columns	3
column-rule-style	Specifies the style of the rule between columns	3
column-rule-width	Specifies the width of the rule between columns	3
column-span	Specifies how many columns an element should span across	3
column-width	Specifies the width of the columns	3
columns	A shorthand property for setting column-width and column-count	3

CSS3 ominaisuuslista (W3Schools. CSS Properties)

fit	Gives a hint for how to scale a replaced element if neither its width nor its height property is auto	3
fit-position	Determines the alignment of the object inside the box	3
image-orientation	Specifies a rotation in the right or clockwise direction that a user agent applies to an image	3
page	Specifies a particular type of page where an element SHOULD be displayed	3
size	Specifies the size and orientation of the containing box for page content	3
ruby-align	Controls the text alignment of the ruby text and ruby base contents relative to each other	3
ruby-overhang	Determines whether, and on which side, ruby text is allowed to partially overhang any adjacent text in addition to its own base, when the ruby text is wider than the ruby base	3
ruby-position	Controls the position of the ruby text with respect to its base	3
ruby-span	Controls the spanning behavior of annotation elements	3
mark	A shorthand property for setting the mark-before and mark-after properties	3
mark-after	Allows named markers to be attached to the audio stream	3
mark-before	Allows named markers to be attached to the audio stream	3
phonemes	Specifies a phonetic pronunciation for the text contained by the corresponding element	3
rest	A shorthand property for setting the rest-before and rest-after properties	3
rest-after	Specifies a rest or prosodic boundary to be observed after speaking an element's content	3
rest-before	Specifies a rest or prosodic boundary to be observed before speaking an element's content	3
voice-balance	Specifies the balance between left and right channels	3
voice-duration	Specifies how long it should take to render the selected element's content	3
voice-pitch	Specifies the average pitch (a frequency) of the speaking voice	3
voice-pitch-range	Specifies variation in average pitch	3
voice-rate	Controls the speaking rate	3
voice-stress	Indicates the strength of emphasis to be applied	3
voice-volume	Refers to the amplitude of the waveform output by the speech syn-	3

CSS3 ominaisuuslista (W3Schools. CSS Properties)

	thesises	
hanging-punctuation	Specifies whether a punctuation character may be placed outside the line box	3
punctuation-trim	Specifies whether a punctuation character should be trimmed	3
text-align-last	Describes how the last line of a block or a line right before a forced line break is aligned when text-align is "justify"	3
text-justify	Specifies the justification method used when text-align is "justify"	3
text-outline	Specifies a text outline	3
text-overflow	Specifies what should happen when text overflows the containing element	3
text-shadow	Adds shadow to text	3
text-wrap	Specifies line breaking rules for text	3
word-break	Specifies line breaking rules for non-CJK scripts	3
word-wrap	Allows long, unbreakable words to be broken and wrap to the next line	3
transform	Applies a 2D or 3D transformation to an element	3
transform-origin	Allows you to change the position on transformed elements	3
transform-style	Specifies how nested elements are rendered in 3D space	3
perspective	Specifies the perspective on how 3D elements are viewed	3
perspective-origin	Specifies the bottom position of 3D elements	3
backface-visibility	Defines whether or not an element should be visible when not facing the screen	3
transition	A shorthand property for setting the four transition properties	3
transition-property	Specifies the name of the CSS property the transition effect is for	3
transition-duration	Specifies how many seconds or milliseconds a transition effect takes to complete	3
transition-timing-function	Specifies the speed curve of the transition effect	3
transition-delay	Specifies when the transition effect will start	3
appearance	Allows you to make an element look like a standard user interface element	3
box-sizing	Allows you to define certain elements to fit an area in a certain way	3
icon	Provides the author the ability to style an element with an iconic equivalent	3

nav-down	Specifies where to navigate when using the arrow-down navigation key	3
nav-index	Specifies the tabbing order for an element	3
nav-left	Specifies where to navigate when using the arrow-left navigation key	3
nav-right	Specifies where to navigate when using the arrow-right navigation key	3
nav-up	Specifies where to navigate when using the arrow-up navigation key	3
outline-offset	Offsets an outline, and draws it beyond the border edge	3
resize	Specifies whether or not an element is resizable by the user	3

LIITE 2(1).

Alkuperäinen “classic-theme”:n portal-normal.vm

```
<!DOCTYPE html>

#parse ($init)

<html class="#language("lang.dir")" dir="#language("lang.dir")"
lang="$w3c_language_id">

<head>
	<title>$the_title - $company_name</title>

	$theme.include($top_head_include)
</head>

<body class="$css_class">

$theme.include($body_top_include)

#if ($is_signed_in)
	#dockbar()
#end

<div id="wrapper">
	<a href="#main-content" id="skip-to-
content">#language("skip-to-content")</a>

	<header id="banner" role="banner">
		<div id="heading">
			<h1 class="site-title">
				<a
class="$logo_css_class" href="$site_default_url" ti-
tle="#language("go-to") $site_name">
					<img
alt="$logo_description" height="$site_logo_height" src="$site_logo"
width="$site_logo_width" />
				</a>

				#if ($show_site_name)
					<span
class="site-name" title="#language("go-to") $site_name">

	$site_name
					</span>
				#end
			</h1>

			<h2 class="page-title">
				<span>$the_title</span>
			</h2>
		</div>

		#if (!$is_signed_in)
			<a href="$sign_in_url" id="sign-in"
rel="nofollow">$sign_in_text</a>
		#end

		#if ($has_navigation || $is_signed_in)
			#parse
("$full_templates_path/navigation.vm")
		#end
	</header>

	<div id="content">
```

Alkuperäinen “classic-theme”:n portal-normal.vm

```
                        <nav class="site-breadcrumbs" id="breadcrumbs">
                                <h1>

            <span>#language("breadcrumbs")</span>
                                </h1>

                                #breadcrumbs()
                        </nav>

                        #if ($selectable)
                                $theme.include($content_include)
                        #else
                                $portletDisplay.recycle()

                                $portletDisplay.setTitle($the_title)

                                $theme.wrapPortlet("portlet.vm",
$content_include)
                        #end
            </div>

            <footer id="footer" role="contentinfo">
                        <p class="powered-by">
                                #language("powered-by") <a
href="http://www.liferay.com" rel="external">Liferay</a>
                        </p>
            </footer>
</div>

$theme.include($body_bottom_include)

</body>

$theme.include($bottom_include)

</html>
```

LIITE 3(1).

Teeman “main-theme” portal-normal.vm

```
<!DOCTYPE html>

#parse ($init)

<html class="#language("lang.dir")" dir="#language("lang.dir")"
lang="$w3c_language_id">

<head>
	<meta name="viewport" content="target-densitydpi=160dpi,
width=device-width, minimum-scale=1, maximum-scale=1">
	<link type="text/css" rel="stylesheet" href="/main-
theme/css/observations.css">
	<link href='http://fonts.googleapis.com/css?family=Armata'
rel='stylesheet' type='text/css'>
	<title>$the_title - $company_name</title>
#if ($page.friendlyURL == "/main")
	<script type="text/javascript"
src="http://maps.googleapis.com/maps/api/js?sensor=false"></script>

    <script type="text/javascript" language="javascript" src="/main-
theme/observations/observations.nocache.js"></script>
#end
	$theme.include($top_head_include)
	<script type="text/javascript" charset="utf-8">
	AUI().use('aui-viewport');
	</script>

<!--[if IE]>
	<script
src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>
<![endif]-->
<!--[if lt IE 9]>
	<script src="http://ie7-
js.googlecode.com/svn/version/2.1(beta4)/IE9.js"></script>
<![endif]-->

</head>

<body class="$css_class">

$theme.include($body_top_include)

#if ($is_signed_in && $permissionChecker.isCompanyAdmin($company_id))

	#dockbar()
#end

<div id="wrapper">

	<header id="banner" role="banner">
		<div id="heading">
			#if ($page.friendlyURL == "/main")
				<div id="uniqueId"
class="map-banner"></div>
			#end
				<div class="logo"></div>

		</div>
		#if (!$is_signed_in)
```

```
                                    <a href="$sign_in_url" id="sign-in"
rel="nofollow">$sign_in_text</a>
                        #else
                                    <a href="$sign_out_url" id="sign-in"
rel="nofollow">$sign_out_text</a>
                        #end

                        #if ($has_navigation || $is_signed_in)
                                    #parse
("$full_templates_path/navigation.vm")
                        #end
            </header>

            <div id="content">
                        #if ($selectable)
                                    $theme.include($content_include)
                        #else
                                    $portletDisplay.recycle()

                                    $portletDisplay.setTitle($the_title)

                                    $theme.wrapPortlet("portlet.vm",
$content_include)
                        #end
            </div>

            <footer id="footer" role="contentinfo">
                        <p class="powered-by">
                                    #language("powered-by") <a
href="http://www.observis.fi" rel="external">Observis Oy</a>
                        </p>
                        <div id="footer_nav">

                        </div>
            </footer>
</div>

$theme.include($body_bottom_include)

</body>

$theme.include($bottom_include)

</html>
```

LIITE 4.

Alkuperäinen “classic-theme”:n portlet.vm

```
#set ($portlet_display = $portletDisplay)

#set ($portlet_id =
$htmlUtil.escapeAttribute($portlet_display.getId()))
#set ($portlet_title = $portlet_display.getTitle())
#set ($portlet_back_url =
$htmlUtil.escapeAttribute($portlet_display.getURLBack()))

<section class="portlet" id="portlet_$portlet_id">
           <header class="portlet-topper">
                      <h1 class="portlet-title">
                                 $theme.iconPortlet() <span
class="portlet-title-text">$portlet_title</span>
                      </h1>

                      <menu class="portlet-topper-toolbar"
id="portlet-topper-toolbar_$portlet_id" type="toolbar">
                                 #if
($portlet_display.isShowBackIcon())
                                            <a class="portlet-icon-
back" href="$portlet_back_url">#language("return-to-full-page")</a>
                                 #else
                                            $theme.iconOptions()
                                            $theme.iconMinimize()
                                            $theme.iconMaximize()
                                            $theme.iconClose()
                                 #end
                      </menu>
           </header>

           <div class="portlet-content">
                      $portlet_display.writeContent($writer)
           </div>
</section>
```

LIITE 5.

Teeman “main-theme” portlet.vm

```
#set ($portlet_display = $portletDisplay)

#set ($portlet_id =
$htmlUtil.escapeAttribute($portlet_display.getId()))
#set ($portlet_title = $portlet_display.getTitle())
#set ($portlet_back_url =
$htmlUtil.escapeAttribute($portlet_display.getURLBack()))

<section class="portlet" id="portlet_$portlet_id">
           <header class="portlet-topper">

                        <menu class="portlet-topper-toolbar"
id="portlet-topper-toolbar_$portlet_id" type="toolbar">
                                    #if
($portlet_display.isShowBackIcon())
                                                <a class="portlet-icon-
back" href="$portlet_back_url">#language("return-to-full-page")</a>
                                    #else
                                                $theme.iconOptions()
                                                $theme.iconMinimize()
                                                $theme.iconMaximize()
                                                $theme.iconClose()
                                    #end
                        </menu>
           </header>

           <div class="portlet-content">
                        $portlet_display.writeContent($writer)
           </div>
</section>
```

LIITE 6.

Alkuperäinen ja muokattu navigation.vm

Alkuperäinen

```
<nav class="$nav_css_class" id="navigation">
    <h1>
        <span>#language("navigation")</span>
    </h1>

    <ul>
        #foreach ($nav_item in $nav_items)
            #if ($nav_item.isSelected())
                <li class="selected">
            #else
                <li>
            #end
                <a href="$nav_item.getURL()" $nav_item.getTarget()><span>$nav_item.icon() $nav_item.getName()</span></a>

                #if ($nav_item.hasChildren())
                    <ul class="child-menu">
                        #foreach ($nav_child in $nav_item.getChildren())
                            #if ($nav_child.isSelected())
                                <li class="selected">
                            #else
                                <li>
                            #end
                                <a href="$nav_child.getURL()" $nav_child.getTarget()>$nav_child.getName()</a>
                            </li>
                        #end
                    </ul>
                #end
            </li>
        #end
    </ul>
</nav>
```

Muokattu

```
<nav class="$nav_css_class" id="navigation">

    <ul>
        #foreach ($nav_item in $nav_items)
            #if ($nav_item.isSelected())
                <li class="selected">
                <a href="$nav_item.getURL()" $nav_item.getTarget()><span class="nav-item-selected">$nav_item.icon() $nav_item.getName()</span></a>
            #else
                <li>
                <a href="$nav_item.getURL()" $nav_item.getTarget()><span class="nav-item">$nav_item.icon() $nav_item.getName()</span></a>
            #end

                #if ($nav_item.hasChildren())
                    <ul class="child-menu">
                        #foreach ($nav_child in $nav_item.getChildren())
                            #if ($nav_child.isSelected())
                                <li class="selected">
                            #else
                                <li>
                            #end
                                <a class="nav_link" href="$nav_child.getURL()" $nav_child.getTarget()>$nav_child.getName()</a>
                            </li>
                        #end
                    </ul>
                #end
            </li>
        #end
    </ul>
</nav>
```

LIITE 7.

Teeman mobiilinavigaation CSS

```
.aui-view-lt720 #navigation {
    height: 0px;
    background: none;
    text-align: left;
    padding: 0;}
.aui-view-lt720 #navigation li {
    padding: 0;
    display: block;
    float: left;
    height: 35px;
    width: 100%;
    background: #414042;
    background: -moz-linear-gradient(#615F62, #414042);
    background: -webkit-gradient(linear, left top, left bottom, from(#615F62), to(#414042));
    background: -webkit-linear-gradient(#615F62, #414042);
    background: -o-linear-gradient(#615F62, #414042);
    background: -ms-linear-gradient(#615F62, #414042);
    background: linear-gradient(#615F62, #414042);}
.aui-view-lt720 #navigation a {
    padding: 0;
    display: block;
    height: 35px;
    width: 95%;
    margin-left: 5%;
    margin-top: 5px;}
.aui-view-lt720 #navigation .selected {
    background: #FFFFFF;
    background: -moz-linear-gradient(#DBDBDB, #FFFFFF);
    background: -webkit-gradient(linear, left top, left bottom, from(#DBDBDB), to(#FFFFFF));
    background: -webkit-linear-gradient(#DBDBDB, #FFFFFF);
    background: -o-linear-gradient(#DBDBDB, #FFFFFF);
    background: -ms-linear-gradient(#DBDBDB, #FFFFFF);
    background: linear-gradient(#DBDBDB, #FFFFFF);}
.aui-view-lt720 #navigation .nav-item-selected {color: #414042;}
.aui-view-lt720 #navigation ul {
    position: relative;
    display: block;
    list-style: none;
    margin: 0 auto;
    padding: 0 0 0 0;
}
```