HAAGA-HELIA
ammattikorkeakoulu

Yrityksen verkkoportaalin ja ajanvarausjärjestelmän prototyypin kehitys

Olli Selamaa

Opinnäytetyö
Tietojenkäsittelyn koulutusohjelma
2012

HAAGA-HELIA
ammattikorkeakoulu

Tietojenkäsittelyn koulutusohjelma

Tiivistelmä

8.11.2012

Tekijä tai tekijät Olli Selamaa	**Ryhmätunnus tai aloitusvuosi** 2008
Raportin nimi Yrityksen verkkoportaalin ja ajanvarausjärjestelmän prototyypin kehitys	**Sivu- ja liitesivumäärä** 32 + 8
Opettajat tai ohjaajat Sirpa Marttila	

Tämän opinnäytetyön tarkoituksena oli uudistaa Tikkurilan Hammaslääkärikeskuksen verkkosivut Liferay-pohjaisiksi, sekä prototyyppi sähköisestä ajanvarausjärjestelmästä. Projekti toteutettiin vuoden 2012 kesän ja syksyn aikana.

Projekti sisälsi kaksi pääosaa: Liferay portaalin teeman luominen sekä ajanvaraus-portletin prototyypin kehitys.

Verkkosivu-uudistuksen tärkeimpiä tavoitteita oli helppokäyttöinen sisällönhallintajärjestelmä sekä yleiskäyttöisyys. Sivujen kehitykseen ei kuitenkaan kuulunut sisällöntuotto eikä käyttöönotto. Ajanvarausjärjestelmän prototyyppi esittelee mahdollisuuden korvata nykyinen puhelinpohjainen toimintamalli. Integrointia potilastietojärjestelmään ei tässä projektissa toteutettu.

Ajanvaraus-portletti toteutettiin uusimmilla Java-teknologioilla ja ohjelmistokehyksillä. Spring Roo –työkalua käytettiin sovelluksessa tehostamaan kehitystä. Sovelluksen arkkitehtuuri hyödyntää täten laajasti Spring Framework –kehystä sekä aspekteja. Vaikka portlet on konfiguroitu Liferaylle, voi sitä käyttää muissa portaalituotteissa minimaalisen konfiguraation avulla.

Portaalin teemaan integroitiin Twitter Bootstrap –ulkoasukehys, joten verkkosivuissa käytetään uusimpia HTML5 ja CSS3 ominaisuuksia. Tuloksena syntyi moderni ulkoasu, joka on optimoitu myös mobiililaitteille.

Asiasanat
Java, Liferay, Portaali, ohjelmistokehitys, Spring Roo

HAAGA-HELIA
University of Applied Sciences

Abstract

8.11.2012

Degree programme in Information Technology

Authors Olli Selamaa	**Group or year of entry** 2008
The title of thesis Developing a web-portal and a prototype booking system for a company	**Number of pages and appendices** 32+8
Supervisor(s) Sirpa Marttila	

The goal of this thesis was to upgrade the website of Tikkurilan Hammaslääkärikeskus to a Liferay-based solution. Additionally, a prototype of an electronic appointment booking system was to be created. The project was carried out during the summer and autumn of 2012.

The project included two main parts: the creation of a theme for Liferay Portal and the development of the appointment booking portlet.

Some of the important objectives for the new website platform was to provide a user-friendly content management system and enhance the overall usability. However, the development process did not include creating content or the deployment of the system. A possibility to replace the current phone-based booking system was introduced by the appointment booking prototype. The integration to the existing patient datasystem was not planned or implemented in this project.

State-of-the-art Java technologies and frameworks were used to develop the appointment booking portlet. Spring Roo was used to boost productivity. Therefore, the software architecture made use of the Spring Framework and aspects extensively. Although the portlet was configured for Liferay, it can be used in other portal products with minimal configuration.

Twitter Bootstrap was integrated to the portal theme, hence new HTML5 and CSS3 features were used to create the website. A modern website design that is also optimized for mobile devices was produced as a result of the development.

Key words
Java, Liferay, Portal, Software Development, Spring Roo

Sisällys

1 Johdanto

1.1 Tarkoitus ja rajaus

Oris Oy, joka tunnetaan paremmin nimellä Tikkurilan Hammaslääkärikeskus on yksityinen hammashoitoklinikka, joka tarjoaa kaikki keskeiset modernit hammaslääketieteen palvelut. Yritys on harjoittanut toimintaansa Tikkurilassa vuodesta 1989 lähtien. (Oris Oy 2011).

Tämän opinnäytetyön aiheena on päivittää yrityksen verkkosivusto Liferay-portaali – pohjaiseen ratkaisuun, sekä kehittää prototyyppi portaaliin liitettävästä ajanvarausjärjestelmä-portletista. Kirjoittaja teki yritykselle vuonna 2007 nykyiset kotisivut, joten uuden alustan kehitys oli melko luonnollinen aihevalinta opinnäytetyölle.

Pyrkimyksenä on, että yrityksen verkkosivut saavat modernimman ulkoasun sekä lisätoiminnallisuuksia. Sisällönhallintajärjestelmän sisältyminen sivustoon on tärkeä osa uudistusta, jotta yrityksen työntekijät pystyvät itse hallinnoimaan sisältöä. Muun muassa tämän vuoksi alustaksi valittiin Liferay-portaali. Se sisältää useita sisäänrakennettuja ominaisuuksia, jotka ovat hyödyllisiä yritykselle. Valinnan kriteereihin kuului myös tulevaisuusvarmuus, millä tarkoitettaan joustavaa uusien sovellusten integroinnin mahdollisuutta ilman ylitsepääsemättömiä ongelmia.

Portaalin kehitys sisältää ulkoasuteeman luonnin sekä tarvittavat konfiguraatiot. Sivustoille tehdään alustavaa sisältöä, mutta koska sisältö tulee toimeksiantajalta, ei lopullisen sisällön tuottaminen kuulu opinnäytetyöhön. Sivuston käyttöönotto ei myöskään kuulu projektiin, jotta aikataulu ei viivästy muista riippumattomista syistä. Siitä sovitaan erikseen toimeksiantajan kanssa.

Ajanvaraus-sovellusprototyypin tarkoituksena on esitellä vaihtoehto nykyiselle puhelinpohjaiselle varausjärjestelmälle. Sovellus on portlet-pohjainen ja Liferay-yhteensopiva, joten se myös havainnollistaa portaalialustan toimintaa ja integraatiomahdollisuuksia. Toimeksiantajalla on olemassa sähköinen työpöytäversio

ajanvaraus- ja potilastietojärjestelmästä, mutta integraatiota tähän järjestelmään ei kehitetä tässä projektissa. Prototyyppi toimii täten lähinnä vaihtoehtona puhelimelle varausten vastaanottajana, vaatien kuitenkin manuaalista tiedonsyöttöä vanhaan järjestelmään. Pyrkimyksenä on, että asiakkaat pystyisivät tekemään sovelluksella varauksia ja että työntekijät pystyisivät vahvistamaan niitä.

1.2 Dokumentin rakenne

Opinnäytetyö on jaettu rakenteeltaan tekniseen teoriaosuuteen sekä toteutuksen kuvaukseen. Teoriaosuudessa esitellään projektissa käytettyjen teknologioiden ominaisuuksia ja hyötyjä. Teoriaosuus selkeyttää täten tiettyjä teknologiavalintoja. Toteutuksen kuvauksessa esitellään työn tavoitteet sekä saavutetut tulokset. Kuvaus sisältää kaksi huomattavaa työn osaa: verkkoportaalin sekä siihen liitettävän ajanvarausjärjestelmän prototyypin toteutus. Dokumentin loppuosiossa on myös analyyttinen pohdintaosuus, jossa pohditaan opinnäytetyön toteutusta.

2 Tekniikka

Tässä kappaleessa esitellään projektissa käytettyjä teknologioita. Käytettyjä teknisiä termejä on listattu liitteessä 1. Oleellisimmat teknologiat käsittelevät Liferay -portaalikehitystä sekä eri ohjelmistokehyksiä- ja työkaluja. Ohjelmistokehyksellä tarkoitetaan valmista ohjelmistokomponentti–kirjastoa, mikä tehostaa kehitystä vähentämällä ylimääräisen työn määrää. Kirjastot ovat yleisesti usean vuoden työn tuloksia, ja niitä käyttämällä parannetaan usein myös ohjelmiston laatua, hyviä käytäntöjä ja toimintavarmuutta. Niitä ei kuitenkaan voi käyttää suoraan ilman muokkauksia, vaan haluttu ohjelma voidaan rakentaa niiden avustuksella. (Mike Baker 2009)

2.1 Liferay Portaali

Portaalilla tarkoitetaan ohjelmistokehityksessä yleisesti palvelua, joka mahdollistaa erillisten sovellusten toiminnan yhdessä paikassa. Se tarkoittaa myös usein paikkaa, josta on pääsy muihin verkkopalveluihin. Portaalia käytetään useimmiten yrityksissä esimerkiksi www-sivustona, intranettinä ja extranettinä; käyttötarkoituksia on useita.(Liferay Inc 2012a) Valmiita portaaliratkaisuja on tarjolla useita. Liferay Portaalilla on paljon kilpailevia tuotteita. Samankaltaisia ominaisuuksia tarjoaa muun muassa Oraclen Web Logic Portal, JBoss Enterprise Portal, eXo Portal Framework, Apache JetSpeed 2 sekä monta muuta. (Java-Source.net) Aikaisemman kokemuksen ja valmiin osaamisen perusteella projektissa päädyttiin kuitenkin Liferay Portaaliin.

Liferay on avoimen lähdekoodin portaalituote. Se on kirjoitettu Javalla ja on täysin kustomoitavissa. Sen yleisin käyttö on luoda verkkosivustoja ja hallinnoida niiden sisältöä. Sille voi myös kirjoittaa lähes kaikilla suosituilla ohjelmointikielillä portletteja, jotka ovat irrallisia verkkosovelluksia. (Liferay Inc 2012b)

Liferayta voi laajentaa muun muassa asentamalla siihen portletteja. Portletilla tarkoitetaan sovellusta, jonka voi liittää esimerkiksi portaaliin. Portlettien toiminnallisuus keskittyy usein hyvin tiettyyn toimintoon. Esimerkiksi Liferay portaaliin voi kirjautua sisäänkirjautumis-portletin kautta, ja portaalissa voi tehdä hakuja

sivustohaku-portletilla. Yhdellä sivulla voi olla useita portletteja. Ne voivat myös kommunikoida keskenään mikäli siihen on tarvetta. Yleisesti portleteilla on kuitenkin omat itsenäiset toiminnallisuudet. (O'Reilly Media 2005)

Liferay sisältää useita sisäänrakennettuja portletteja. Ne kattavat laaja-alaisesti eri toimintoja. Näitä toimintoja voi olla esimerkiksi sisällön esittäminen, pikaviestintä tai sähköpostien hallinta.
Kustomoitujen portlettien teko on myös mahdollista. Liferay portletteja voi kehittää useilla moderneilla ohjelmointikielillä kuten Javalla. Portletit vaativat vain hieman Liferay-kohtaista konfiguraatiota, mutta esimerkiksi toisella portaalialustalla pyörivä portlet on mahdollista liittää Liferay-portaaliin ilman muutoksia itse sovellukseen. (Liferay Wiki 2009a)

Muokattavuus on usein ongelma varsinkin sisällönhallintajärjestelmien kanssa. Liferayn ydinominaisuuksia on suhteellisen helppo muokata "Hook" –pluginien ja teemojen avulla. Jopa alustan lähdekoodia ja muita ominaisuuksia voi muuttaa laajemmalla mutta tehokkaalla "EXT" –pluginilla. Liferayn mukana tulee kymmeniä sisäänrakennettuja ominaisuuksia, kuten kalenteri, sähköposti, chat, wiki, web-sisällön esitys, LDAP -integrointi sekä useita muita. Kaikki nämä valmiit portletit ja ominaisuudet on täysin muokattavissa. Yhteisön tekemiä portletteja, teemoja ja hook-plugineja on myös saatavilla ilmaiseksi lukuisia. (Liferay Wiki 2008)

Portaaleissa halutaan usein esittää vain tietyille käyttäjille. Liferayssa on sisäänrakennettu laaja käyttäjä- ja roolihallinta. Sisältöä sivuilla voi rajoittaa esimerkiksi vain tietyille käyttäjäryhmille- tai rooleille. On myös mahdollista tehdä kokonaan erillinen sivusto, joihin tietyillä käyttäjillä tai ryhmillä on pääsy. Esimerkkinä yrityksen intranet, joka on sallittu vain yrityksen työntekijöille. (Liferay Inc 2012c)

Liferay on käännetty useille kielille. Yhteisö on erittäin aktiivinen, joten lähes kaikki tunnetut kielet on tuettuja. On myös mahdollista tehdä omia käännöksiä ja korvata olemassa olevat. Tuki mobiilialustoille on myös tehty helpoksi toteuttaa (puhelimet, tabletit). (Liferay Inc 2012c)

Eräs tärkeimmistä ominaisuuksista Liferayssa on sen sisällönhallintajärjestelmä. Sivujen lisäys ja muokkaus on tehty helpoksi ja tehokkaaksi. Sisällön julkaisuun on myös mahdollista lisätä varmistus, jolloin tietyn roolin omaava henkilö voi tarkistaa sisällön ennen sen julkaisua. (Liferay Inc 2012c)

Myös Liferayn yhteisö- ja organisaatioarkkitehtuuri on tehty selkeäksi, mikä tekee sivustojen luonnin eri ryhmille helpoksi. Tiimit, ali-organisaatiot ja kaveriryhmät voivat luoda omat sivunsa täysin erillisiksi muista.
Sovelluksiin on myös mahdollista sisällyttää sosiaalisten verkostojen integrointi. Tuettuja verkostoja on Liferayn oman alustan lisäksi OpenSocial, jota käyttää useat sosiaaliset verkostot. (Liferay Inc 2012c)

2.2 Liferay teema

Ulkoasun kehittäminen Liferay –portaalille noudattaa pääsääntöisesti normaaleita ulkoasusuunnittelun sääntöjä. Kehitys on kuitenkin hieman monimutkaisempaa kuin tavallisella www-sivulla. Liferay sisältää valmiita ”teemoja”. Usein kuitenkin halutaan valmista teemaa persoonallisempi ulkoasu, joten tarvetta on kehittää uusi teema. Teemalla tarkoitetaan ulkoasua muuttavaa ohjelmistokomponenttia jonka voi helposti lisätä Liferay-portaaliin ja poistaa sen siitä. (Liferay Documentation 2012a)

Teemoja voi luoda Liferay Plugins SDK:lla, ja ne hyödyntävät muun muassa Velocity- ja Freemarker templaattimoottoreita. Teeman avulla Liferayn ulkoasu on täysin muokattavissa. Portaalin sivuja ja rakennetta voi muokata haluamakseen, ja tyylejä voi muokata käyttäen CSS –kieleltä. Pohjateema generoidaan käyttäen Liferay Plugins SDK:ta, ja pohjateemaksi voi valita olemassa olevan teeman tai käyttää tyhjää teemaa. Tyhjän teeman avulla saavutetaan paras kustomoitavuus, mutta se vaatii myös enemmän työtä. (Liferay Documentation 2012a)

Kuviossa 1 esitellään Liferay teeman tiedostorakennetta. Se sisältää kansiot CSS –tiedostoille, kuville, JavaScriptille sekä Velocity –templaateille. CSS –muokkaukset tehdään normaalisti *custom.css* –tiedostossa. Tässä tiedostossa olevat tyylimääritelmät ylikirjoittavat mahdolliset olemassa olevat tyylit, mikä mahdollistaa valmiin teeman

muokkauksen. Ylimääräisiä CSS-tiedostoja voi lisätä *main.css* tiedostoon, joka lataa siinä määritellyt tiedostot lisäysjärjestyksessä. Viimeiseksi ladatuilla CSS-tiedostoilla on suurempi tärkeys, joten ne ylikirjoittavat samat tyyliluokat aikaisemmin ladatuissa tiedostoissa. Tämän vuoksi *custom.css* ladataan aina viimeiseksi. (Liferay Documentation 2012b)

Templates-kansiossa luetellut Velocity-templaatit mahdollistavat Liferayn perusrakenteen muokkauksen. Eri templaattien avulla voi muokata muun muassa navigaatio-komponenttia, portlettien ulkoista rakennetta sekä itse portaalin rakennetta. Muokkaus on mahdollista perus HTML-osaamisella, mutta monimutkaisemmat muokkaukset vaativat myös Velocity-templaattikielen hyödyntämistä. Templaatit sisältävät myös valmiiksi esitellyissä muuttujissa tietoja portaalista, kuten kirjautuneesta käyttäjästä ja portaalin asetuksista. Näitä Velocity-muuttujia voi lisätä teemaan *init_custom.vm* –tiedostossa. Niiden avulla voidaan hyödyntää portaalin tietoja ilman tarkempaa osaamista miten tietoihin pääsee käsiksi. (Liferay Wiki 2009b)

```
/TEEMAN_NIMI/
    /css/
        application.css
        base.css
        custom.css
        dockbar.css
        extras.css
        forms.css
        layout.css
        main.css
        navigation.css
        portlet.css
    /images/
        (useita kansioita)
    /js/
        main.js
    /templates/
        init_custom.vm
        navigation.vm
        portal_normal.vm
        portal_pop_up.vm
        portlet.vm
```

Kuvio 1. Liferay teeman rakenne (Liferay Documentation 2012b)

Portaaliin voidaan liittää useita eri portletteja. Usein nämä komponentit tuottavat portaalin sivuille sisältöä tai jonkinlaisen käyttöliittymän. Liferay sisältää valmiiksi useita

rakenne-templaatteja[1], joiden avulla voidaan määritellä sijainnit, mihin näitä portletteja voidaan sijoittaa. Templaattien avulla voidaan joustavasti muokata, millainen rakenne sivuilla on. Ne muokkaavat kuitenkin vain portlettien sijoittelun asetuksia, ei portaalin perusrakennetta. (Liferay Documentation 2012c)

Rakennetemplaatin pohja voidaan generoida Liferay Plugins SDK:lla, ja tehdä sen jälkeen tarvittavat muutokset. Templaatissa täytyy olla tietyt komponentit määriteltynä, mihin portletteja voidaan sijoittaa, mutta muutoin määrittely on täysin kustomoitavissa. Kuvio 3 havainnoillistaa templaatin vaadittuja komponentteja. Määrittely voidaan tehdä myös mobiililaitteille erillisessä ”wap” –tiedostossa. Kuviossa 2 esitellään rakennetemplaatti-projektin tiedostorakennetta. (Liferay Documentation 2012c)

```
/RAKENNE_TEEMA/
docroot/
    WEB-INF/
        liferay-layout-templates.properties
        liferay-plugin-package.properties
    layout_template_name.png
    layout_template_name.tpl
    layout_template_name.wap.tpl
```

Kuvio 2. Rakennetemplaatin tiedostorakenne (Liferay Documentation 2012c)

```
<div id="main-content" role="main">
    <div class="portlet-layout">
        <div class="portlet-column" id="column-1">
            $processor.processColumn("column-1")
        </div>
        <div class="portlet-column" id="column-2">
            $processor.processColumn("column-2")
        </div>
    </div>
</div>
```

Kuvio 3. Esimerkki rakennetemplaatin vaadituista komponenteista (Liferay Documentation 2012c)

[1] Käännös englanninkielisestä termistä *layout template*

2.3 Twitter Bootstrap

Ohjelmistokehityksessä erilaiset kehykset ovat olleet suosittuja jo pitkään. Myös ulkoasukehityksen avuksi on ilmestynyt lukuisia kehyksiä. Nämä kehykset sisältävät yleisesti valmiita työkaluja, tyylikomponentteja, JavaScript –toimintoja, HTML-templaatteja sekä muita sisällön esitykseen liittyviä apuvälineitä. Ne parantavat varsinkin selainyhteensopivuutta sekä tukea mobiililaitteille. (Javier Cuevas 2012)

Yksi tämän hetken suosituimmista ulkoasukehyksistä on Twitter Bootstrap. Sen tarkoitus on helpottaa kehitystä tarjoamalla työkaluja ulkoasun kehittämiseen. Se sisältää eri selaimille yhteensopivia valmiita CSS-tyylejä useimmille HTML –elementeille, ja valinnaisia JavaScript-lisätoimintoja. Sivuston rakenteen luonnin avuksi se sisältää myös 12-kolumnisen ruudukkorakenteen, minkä avulla verkkosivulle voi määritellä hyvinkin monimutkaisen rakenteen ilman laajempaa CSS-osaamista. (Javier Cuevas 2012)

Twitter Bootstrapin uusimmassa julkaisussa hyödynnetään myös 'reagoivaa suunnittelua[2]'. Sen periaatteisiin kuuluu sisällön joustava esittäminen kaikille laitteille ilman suurempia muokkauksia sivustossa. Tämä tarkoittaa sitä, että sisällön tulisi mukautua käytettävän tilan mukaan. Usein tämä vaatii kuvien koon dynaamista muokkaamista ja rakenteen dynaamista vaihtamista. Esimerkiksi mobiililaitteella osa sisällöstä voidaan haluta piilottaa tilan säästämiseksi. Nykyaikana mobiililaitteilla selailu on hyvin suosittuja, joten verkkosivun yhteensopivuus mobiililaitteille on hyvin tärkeää. (Kayla Knight 2011)

Twitter Bootstrap hyödyntää LESS[3]-kieltä. LESS on dynaaminen tyylisivu-kieli, joka mahdollistaa muun muassa muuttujien, sisäkkäisten tyylien, css-funktioiden ja operaattorien käytön tyylisivujen teossa. LESS on pohjimmiltaan CSS esikäsittelijä, joka luo LESS-tiedostoista validia CSS koodia. Tämä muunnos voidaan tehdä joko JavaScriptin avulla selaimessa tai renderoida LESS-tiedostot etukäteen CSS-tiedostoiksi.

---

[2] Käännös englanninkielisestä termistä *responsive design*
[3] LESS – dynamic stylesheet language

Kilpaileva teknologia on SASS[4], joka on hyvin samanlainen kuin LESS. (Javier Cuevas 2012; Alex Sellier)

2.4 Spring Framework

Spring Framework on yksi johtavista avoimen lähdekoodin Java-ohjelmistokehyksistä. Sen pyrkimys on tehostaa ja helpottaa Java –websovellusten kehitystä tarjoamalla valmiita ja joustavia Java –ohjelmistokomponentteja. Sen laajuuden vuoksi se toimii usein korvaajana monille Java EE ominaisuuksille. Kehyksellä on erittäin aktiivinen ja laaja käyttäjäyhteisö, ja alustaa kehitetään jatkuvasti.
Sen suosion johdosta Spring Framework -pohjaisia lisäprojekteja on syntynyt useita, lähes jokaiseen tarpeeseen. Esimerkiksi Spring Security tarjoaa web-sovelluksille järeää tietoturvaa, Spring Mobile tuo mobiilikehitykseen suositun ohjelmointimallin, Spring Web Services helpottaa SOAP -palveluiden kehitystä ja Spring Social integroi sovelluksia sosiaalisiin palveluihin. (SpringSource 2012a)

Spring korostaa modulaarisuutta, minkä vuoksi se on helppo integroida olemassa oleviin sovelluksiin. Sen ydinosa sisältää useita erillisiä moduuleita, jotka tarjoavat erilaisia palveluita. Näitä yksittäisiä moduuleita voi ottaa vähitellen käyttöön tarpeen mukaan. (SpringSource 2012a)

Inversion of control, IoC, on olennainen osa Spring kehystä. Se koostuu useasta eri tekniikasta, joiden tarkoitus on yksinkertaistaa liiketoimintalogiikkaa. Spring kehyksessä sen toiminta koostuu resurssien ja riippuvuuksien injektoimisesta[5] ohjelmaan ajon aikana. Java objekteja tai muita resursseja voidaan määritellä Springin IoC –säiliöön, josta niitä voidaan käyttää ohjelman komponenteissa. (SpringSource 2012b)

Olennainen projektissa hyödynnetty tekniikka on aspekti-ohjelmointi, lyhenteeltään AOP[6], mikä on ohjelmointimalli jonka tarkoituksena on varsinkin parantaa modulaarisuutta. Sen avulla voidaan erottaa tehokkaasti toisistaan eroavat toiminnot,

[4] SASS – Syntatically Awesome Sylesheets
[5] käännös englanninkielisestä termistä Dependency Injection
[6] AOP = Aspect Oriented Programming

mutta jotka kulkevat toistensa läpi. Näitä toimintoja voi olla esimerkiksi ohjelman lokitus, mikä on hyvä esimerkki ohjelmiston yhtymäkohdasta joka usein halutaan erottaa. Lokitus vaikuttaa lähes jokaiseen ohjelman osaan, joten se läpileikkaa useimpaa kohtaa ohjelmassa. Aspekti yhdistää sivuavat toiminnot neuvoilla ja liittymäkohta-määritysten avulla. Aspektiohjelmointi on myös tärkeä osa Spring kehystä, sillä se pohjautuu laajasti aspektien käyttöön. (SpringSource 2012c)

Lähes jokainen sovellus kommunikoi tietovaraston kanssa. Hyviin käytäntöihin kuuluu myös käyttää transaktioita virhetilanteiden käsittelyyn tämän kommunikoinnin aikana. Spring Framework yksinkertaistaa tietokanta-transaktioiden toteutusta. Transaktioiden tekoon Java-ohjelmoinnissa vaaditaan usein paljon täytekoodia, joten Spring tehostaa tätä prosessia valmiilla transaktion hallinnalla. Metodit tai luokat, jotka vaativat transaktioita, voidaan yksinkertaisesti merkitä @Transactional –annotaatiolla, jonka jälkeen Spring hoitaa automaattisesti transaktioihin liittyvät komplikaatiot. (SpringSource 2012d)

Oleellinen osa toiminnallisia sovelluksia on käyttöliittymän toimintojen hallinnointi. Model-view-controller, lyhyesti MVC, on ohjelmointimalli, jossa ohjelman näkymä-tieto- ja ohjauskerros erotetaan toisistaan. Se selkeyttää ohjelmiston arkkitehtuuria ja jälleen parantaa modulaarisuutta sekä tehostaa ohjelmointia. Spring MVC on kehyksen toteutus tästä ohjelmointimallista. Se käsittelee web-ohjelmien HTTP pyynnöt ohjaamalla ne oikeisiin paikkoihin tai näkymiin ohjelmassa. (SpringSource 2012e) Portlet ympäristöön on myös olemassa Spring Portlet MVC, minkä toimintamalli on hyvin samankaltainen kuin web-ympäristön toteutus. (SpringSource 2012f)

Spring sisältää lisäksi sisäänrakennetun tuen monikielisyydelle. Tämä tarkoittaa, että ohjelman käyttöliittymä voidaan näyttää useilla eri kielillä vaikuttamatta itse toteutukseen. Toteutustapoja on muutamia, mutta yleisesti käännökset haetaan tietojärjestelmästä, joka voi olla esimerkiksi tiedosto tai tietokanta. Tämä informaatio välitetään ohjelman näkymäkerrokselle, joka osaa näyttää siten oikean tekstin lokalisaation mukaisesti. (SpringSource 2012g)

Tiedon validointi on olennainen osa Java-ohjelmointia. Tallennettavat tiedot tulisi olla oikeanlaisia, jotta tietorakenne pysyy eheänä ja vältytään virheiltä. Spring tukee Java-objektien validointia JSR-303[7] standardin mukaisesti. Tämä tarkoittaa sitä, että ennen kuin tietoja voidaan käyttää liiketoimintalogiikassa, tiedot validoidaan asetettujen sääntöjen mukaan. Web-sovelluksissa validointi on usein kaksikerroksinen; ensimmäinen validointi tehdään web-lomakkeella, minkä jälkeen objektit validoidaan vielä tietotasolla. (SpringSource 2012h)

Spring hyödyntää laajasti Java-annotaatioita nopeuttamaan ja yksinkertaistamaan kehitystä. Annotoidut luokat, metodit tai objektit saavat automaattisesti lisätoimintoja tai attribuutteja tarpeesta riippuen. Esimerkiksi @Autowired-annotaatiolla Spring osaa automaattisesti asettaa objektille oikeat riippuvuudet IoC-säiliöstä. Java-luokan annotoiminen @Controller-annotaatiolla kertoo Springille, että luokka toimii pyyntöjen ohjaajana. Annotaatioita on lukuisia, ja niiden käyttö vähentää huomattavasti XML-konfiguraation määrää. Uusimmissa Spring kehyksen versioissa on jopa mahdollista korvata web-ohjelman XML-konfiguraatio kokonaan Java-konfiguraatiolla. (Perficient 2009)

2.5 Spring Roo

Spring Roo on SpringSourcen kehittämä avoimen lähdekoodin työkalu nopeaan ohjelmistokehittämiseen. Ensimmäistä kertaa se esiteltiin julkisesti SpringOne konferenssissa huhtikuussa vuonna 2009. (SpringSource, Spring Roo 2012a)
Sen toimintaperiaatteisiin kuuluu niin sanottu 'convention-over-configuration', minkä tarkoituksena on vähentää kehittäjän tarvitsemia päätöksiä. Tämä yksinkertaistaa ja nopeuttaa kehitystä ilman suuria haittavaikutuksia. Sillä voi luoda muutamissa minuuteissa pohjan web-sovellukselle, ja lisäominaisuuksia voi lisätä hyvin tehokkaasti. Se hyödyntää useita suosittuja Java-teknologioita; muun muassa Spring Framework, JPA, Apache Maven, AspectJ sekä monta muuta. (SpringSource, Spring Roo 2012b)

[7] Java Specification Requests, viralliset määrittelydokumentit Java-alustalle tehdyille ehdotuksille. JSR-303 määrittelee Java objektien validoinnin.

Roon toiminta koostuu pääosin ohjelmakoodin generoinnista käyttäjän antamien komentojen avulla. Roo generoi Java-luokkia sekä niihin kytkettyjä AspectJ-tiedostoja. Roo skannaa myös käyttäjän tekemiä muokkauksia, ja muuttaa generoitua koodia tarpeen mukaan. Käyttäjä ei ole myöskään sidottu käyttämään Roota. Sen tavoitteena on olla irtonainen apuväline, jonka käytön voi halutessaan lopettaa vaikuttamatta projektiin. Käyttäjä voi lopettaa Roon käytön ohjelmistoprojektissa yksinkertaisesti lakkaamalla komentolinjan käytön. Pysyvämpiin tuloksiin pääsee poistamalla tai siirtämällä generoidut luokat ja poistamalla Roo-kohtaiset annotaatiot. Käytön aloittaminen uudelleen on tehty myös yhtä yksinkertaiseksi. (SpringSource, Spring Roo 2012b)

Spring Roon käyttöliittymä on pääsääntöisesti komentorivipohjainen. Komentorivi tarjoaa useita toimintoja, ja niitä voi lisätä liitännäisillä. Liitännäisiä on sisäänrakennettuna useita, ja käyttäjät voivat myös kehittää omia liitännäisiä. Komentorivi sisältää myös apuvälineitä, kuten konteksti-riippuvaiset vinkit sekä tekstin automaattitäydennyksen.
Käyttäjä ei ole kuitenkaan sidottu käyttämään komentolinjaa. Spring Roon tavoite onkin olla sitomatta käyttäjää tiettyihin menetelmiin. Kehittäjä voi käyttää ohjelmoidessa haluamaansa editoria, ja Roo voidaan jättää pyörimään taustalle. Kun ohjelmakoodiin tehdään muutoksia, Roo huomaa muutokset ja tekee tarvittavat toimenpiteet niiden perusteella. (SpringSource, Spring Roo 2012b)

Kuviossa 4 havainnollistetaan Spring Roon tehokkuutta. Muutaman komennon jälkeen käyttäjä on luonut valmiin web-sovelluksen, sisältäen yksikkötestit, Selenium testit, käyttöliittymän käyttäen sekä Spring MVC:tä että GWT[8]:tä. Sovellus sisältää myös tietokantakonfiguraatiot sekä normaalit CRUD[9] tietokantafunktiot. (SpringSource 2012i)

[8] GWT = Google Web Toolkit. Ohjelmistokehys rikkaiden verkkosovellusten tekoon
[9] CRUD = Create, Read, Update, Delete

```
mkdir hello
cd hello
roo
roo> hint
roo> project --topLevelPackage com.foo
roo> jpa setup --provider HIBERNATE --database HYPERSONIC_IN_MEMORY
roo> entity jpa --class ~.Timer --testAutomatically
roo> field string --fieldName message --notNull
roo> hint web mvc
roo> web mvc setup
roo> web mvc all --package ~.web
roo> selenium test --controller ~.web.TimerController
roo> perform tests
roo> quit
```

Kuvio 4. Spring Roo komentolinjaesimerkki (SpringSource 2012i)

Spring Roo on laajennettavissa useilla liitännäisillä, jotka lisäävät komentoja Roon komentolinjaan. Sisäänrakennettuja liitännäisiä on kuitenkin lukuisia. Niiden avulla voi konfiguroida sovellukseen muun muassa Maven-projektirakenteen, JPA-tuen, Java-objektien validoinnin, Log4J lokituksen, yksikkötestien teon sekä web-käyttöliittymän useilla eri teknologioilla kuten Spring MVC, GWT tai Adobe Flex. Yhteisön tekemät liitännäiset laajentavat jo vaikuttavaa toiminnallisuutta. (SpringSource 2012i)
Aivan kaikkea Spring Roo ei tue. Esimerkiksi tuki portleteille on melko olematon eikä valmiita liitännäisiä ole vielä kehitetty portlettien tukemiseksi. Roon avulla sovellukselle voi kuitenkin tehdä pohjan ja kustomoida toiminnan sen jälkeen haluamaksensa kaltaiseksi.

2.6 Kehitysympäristö

Sovelluskehityksen tehostamiseksi on kehitetty useita eri apuvälineitä. Työkaluja on olemassa miltei jokaiseen käyttötarpeeseen, kuten projektinhallintaan, versionhallintaan ja ohjelmakoodin kirjoittamiseen. Lähes kaikki ohjelmistoprojektit hyödyntävät eri apuvälineitä ainakin jossain määrin.

Projektin kehitystyö tapahtui lähinnä käyttämällä Eclipse IDE[10] -ympäristöä. Eclipse IDE on avoimen lähdekoodin ohjelmointiympäristö. Se tukee erityisesti Java-kieltä. Tuki löytyy myös C, C++ ja PHP –ohjelmointikielille.

[10] IDE – Integrated Development Environment

Eclipselle on lukuisia liitännäisiä, ja sille on mahdollista myös kehittää omia liitännäisiä. Hyödyllisiä liitännäisiä Java-kehityksessä on esimerkiksi Maven integraatio, sekä integrointi eri versionhallintajärjestelmien kanssa kuten Git ja SVN. (The Eclipse Foundation)

Git on ilmainen, avoimen lähdekoodin ohjelmiston versionhallintatyökalu, jonka pääpainona on tehokkuus ja hajautettu toiminta. Tässä projektissa käytettiin Git-versionhallintaa säilyttämään ohjelmakoodi turvallisesti pakettivarastossa[11]. Projektissa ei muihin versiohallinnan ominaisuuksiin juuri ollut, sillä vain yksi henkilö teki muutoksia ohjelmaan, joten päällekkäisyyksien mahdollisuus oli minimaalinen. (Git 2012)

Ohjelmoidessa tuotettu ohjelmakoodi täytyy ennen käyttöä muuntaa järjestelmälle käytettäväksi pakkaamalla se sopivaan muotoon. Useat ohjelmointiympäristöt sisältävät mahdollisuuden suorittaa tämän operaation, mutta hyviin käytäntöihin kuuluu erillisten rakennustyökalujen käyttö. Maven on yleisesti Java-projekteissa käytetty koodin rakennustyökalu[12]. Se hoitaa kuitenkin useita muita tehtäviä kuin pelkästään ohjelmakoodin kompilaation ja pakkauksen. Sitä käytetään usein ohjelmistoprojektien hallintaan, sillä sen avulla useita projekteja voi yhdistää toisiinsa vaivatta. Se käsittelee myös projektin ulkoiset riippuvuudet ja kirjastot. Aikaisemmin kolmannen osapuolen ohjelmistokirjastot täytyi manuaalisesti kopioida projektiin, mutta Mavenin avulla riippuvuudet voi määritellä yksinkertaisella XML-konfiguraatiolla. Maven tarjoaa ohjeita hyvien ohjelmistokehitys-käytäntöjen seuraamiseen, varsinkin liittyen projektin tiedostorakenteeseen. (Apache Maven Project 2012a)

Mavenin XML konfiguraatiossa voi määritellä projektin asetukset, riippuvuudet, liitännäiset koodin pakkaus-asetukset sekä useita muita konfiguraatioita. Liitännäisten avulla voi kustomoida rakennusprosessia ja tuoda siihen lisäominaisuuksia. (Apache Maven Project 2012b)

---

[11] käännös englanninkielisestä termistä Repository
[12] englanninkielinen termi build automation tool

Maven käyttää taulukossa 1 esiteltyä ennaltamääriteltyä hakemistorakennetta, mikä selkeyttää tiedostojen tarkoitusta. Monimutkaisemmatkin rakenteet ovat tuettuja, kuten multi-moduuliset projektit, joissa yksi pääprojekti voi yhdistää alleen useita muita liittyviä projekteja. Tiedostorakennetta voi muuttaa määrittelemällä kansioiden sijainnit, mutta usein on hyvien käytäntöjen mukaista käyttää oletusasetuksia. (Apache Maven Project 2012c)

Kansio	**Sisällys / Tarkoitus**
kotikansio	pom.xml konfiguraatiotiedosto ja alikansiot
src/main/java	projektin Java-lähdekoodi
src/main/resources	resurssitiedostot kuten sovelluskonfiguraatiot
src/test/java	testiluokat, esimerkiksi JUnit yksikkötestit
src/test/resources	resurssitiedostot joita voi käyttää testiluokissa

Taulukko 1. Mavenin normaali tiedostorakenne (typistetty) (Apache Maven Project 2012c)

Maven suorittaa projektien rakennuksen tiettyjen vaiheiden[13] mukaan. Oletusarvoisesti Maven käyttää kuviossa 5 esiteltyä elinkaarta tavoitteiden suorittamiseen. Käyttäjä voi määrittää suorittamisen päättymisen tiettyyn vaiheeseen. Esimerkiksi komentolinja-komennolla '*mvn test*' Maven suorittaa tavoitteet vain testivaiheeseen asti. Eri liitännäiset voi konfiguroida suorittamaan tiettyjä asioita tietyissä vaiheissa. Suoritusvaiheita voi myös kustomoida, mutta useimmissa projekteissa oletusvaiheet ovat varsin riittävät. (Apache Maven Project 2012d)

```
1. process-resources
2. compile
3. process-test-resources
4. test-compile
5. test
6. package
7. install
8. deploy
```

Kuvio 5. Mavenin suoritusvaiheet pakatessa projektia (Apache Maven Project 2012d)

[13] englanninkielinen termi build lifecycle - Mavenin elinkaari joka sisältää eri vaiheita

3 Toteutus

Tässä luvussa esitellään projektin lähtökohtaiset vaatimukset ja tavoitteet, sekä niiden perusteella toteutuneet tulokset. Luku on jaettu teemoittain tavoitteisiin ja vaatimuksiin, ajanvaraus-portletin kuvaukseen sekä Liferay-teeman esittelyyn.

3.1 Tavoitteet ja vaatimukset

Lähtökohtana projektille oli, että asiakas toivoi verkkosivujen ulkoasun päivittämistä modernimmaksi. Toiveena oli myös mahdollisuus helposti muokata sisältöä ilman laajempaa tietoteknistä osaamista. Pyrkimyksenä oli myös siirtyä järjestelmään, jota voisi tulevaisuudessa laajentaa ja johon voisi integroida helposti muita palveluita. Ajanvarausjärjestelmän päivittämisestä sähköiseen versioon oli myös puhetta, joten ylimääräisenä vaatimuksena oli tehdä prototyyppi sähköisestä ajanvarausjärjestelmästä, joka voisi toimia vaihtoehtoisena menetelmänä puhelinvarausten kanssa.

Projektin alussa vertailtiin erilaisia sisällönhallintajärjestelmiä. Päätettiin käyttää Liferay portaalia, sillä olemassa oli jo valmiiksi kokemusta kyseisestä järjestelmästä. Liferayssa on sisäänrakennettu sisällönhallintajärjestelmä, jota voi käyttää ilman suurempaa teknistä osaamista. Portaalituotteena sen suunnitteluperiaatteisiin kuuluu myös helppo sovellusten integrointi.

Ulkoasun modernisoimiseksi suunniteltiin käytettäväksi uusimpia ulkoasu-teknologioita, kuten HTML5:ää ja CSS3:a. Näiden lisäksi ja laadun varmistamiseksi päädyttiin hyödyntää Twitter Bootstrap –ulkoasukehystä, sillä se tarjoaa paljon uudelleenkäytettäviä ominaisuuksia modernin ulkoasun rakentamiseen.

Ajanvaraus-portletissa pyrittiin valitsemaan myös uusimpia Java-teknologioita, jotka kuitenkin sopivat palvelun toimintaperiaatteisiin. Tavoitteena oli, että asiakkaat pystyvät tekemään varauspyyntöjä internetin kautta, joita yrityksen henkilökunta pystyy vahvistamaan. Teknologiavalinnat pohjautuivat suuresti Spring Framework –kehyksen ja siihen liittyvien projektien ympärille. Spring Roo –työkalua päätettiin hyödyntää sovelluksen pohjan rakentamisessa. Ongelmana oli sen heikko portlet-tuki, mutta

integraatio katsottiin mahdolliseksi ilman suurempia ongelmia. Portletin toiminnallisuus koostuu pitkälti CRUD[14] –toiminnoista, joiden luomisessa Spring Roo on varsin tehokas.

Prototyypin tavoitteet muuttuivat projektin aikana hieman, mutta lopullisten vaatimusten lähtökohdat olivat kuitenkin alkuperäisten toivomusten kaltaiset. Toiminnallisia vaatimuksia suunniteltiin myös vertailemalla kilpailevien yritysten olemassa olevia varausjärjestelmiä. Toiminta pyrittiin täten pitämään sovelluksessa melko yksinkertaisena sisältämällä vain tärkeimmät toiminnot. Näitä on muun muassa ajanvarauksen teko sekä varauksen vahvistus.

3.2 Liferay-teema

Tässä luvussa esitellään Liferay-teeman kehityksen lopputulosta sekä keinoja joilla tulokset saatiin aikaan.

3.2.1 Projektin alustus

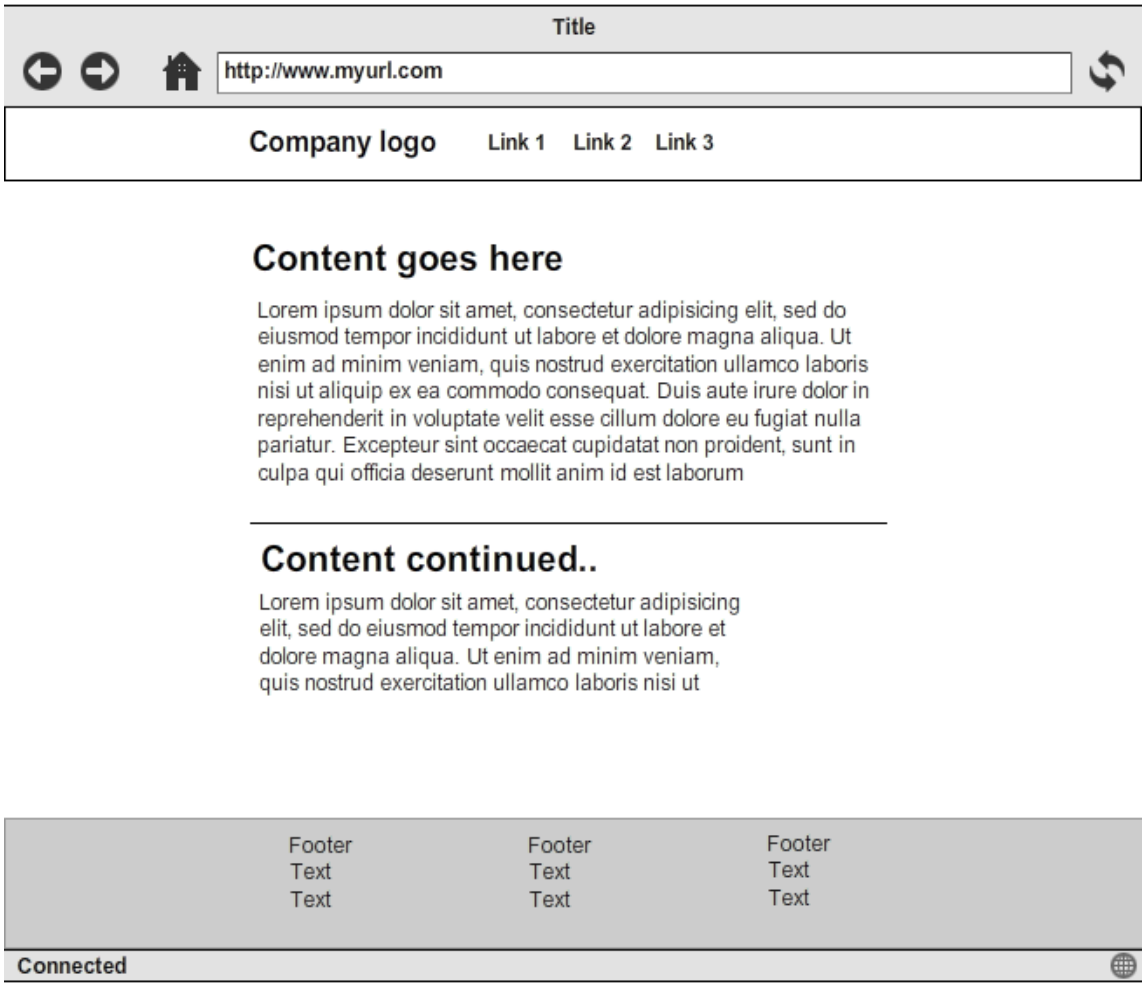
Liferay teemaprojektin pohja luotiin Mavenilla käyttäen Liferay Maven pluginia sekä arkkityyppiä[15] 'liferay-theme-archetype'. Mavenin arkkityypit ovat templaattipohjia erityyppisille projekteille. (Mika Koivisto 2012) Mavenin konfiguraatiossa voi myös valita pohjateeman, jonka päälle voi tehdä muokkauksia. Tässä projektissa valittiin kuitenkin *unstyled* –pohja, joka ei sisällä valmiita tyylejä. Tämä varmistaa maksimaalisen joustavuuden ulkoasun kehityksessä. Liferay Maven plugin tuo Liferay Plugins SDK:n ominaisuudet Mavenin käyttöön, minkä avulla pystytään käyttämään Mavenia teeman kehityksessä. Tiedostorakenne on normaalin Liferay-teeman mukainen, yhdistettynä Mavenin standardiin hakemistorakenteeseen.

Teeman lisäksi luotiin myös Liferay-rakennetemplaatti Mavenilla käyttäen arkkityyppiä 'liferay-layouttpl-archetype'. (Mika Koivisto 2012) Rakennetemplaatti mahdollistaa

[14] Create, Read, Update, Delete
[15] käännös englanninkielisestä termistä archetype

Liferayn sisällön rakenteen joustavan muokkauksen. Joustavuudella tarkoitetaan, että sivujen templaatteja pystyy helposti vaihtamaan, asentamaan ja poistamaan portaalista.

Ennen sivuston kehityksen aloitusta suunniteltiin karkea rautalankamalli sivun komponenttien asettelusta (Kuvio 6). Sivuston rakenne koostuu täten kolmesta pääosasta: navigointi, sisältö sekä footer[16]-osio. Rautalankamalli helpottaa ulkoasun kehitystä sekä havainnoillistaa asiakkaalle suunniteltua rakennetta.



Kuvio 6. Sivuston rautalankamalli

[16] Footer = englanninkielinen termi tarkoitettaessa verkkosivun alaviitettä

3.2.2 Ulkoasun toteutus

Ulkoasun teknologiaperustana päätettiin käyttää Twitter Bootstrap –ulkoasukehystä. Sen integraatio Liferayn kanssa vaatii portaalin Velocity-templaattien muokkausta, jotta Bootstrapin hyödylliset ominaisuudet saataisiin käyttöön. Tavoitteena oli käyttää Bootstrapin 12-kolumnista ruudukkorakennetta, sen navigointi-palkkia sekä reagoivaa suunnittelua. Footer tulisi olla myös sisältöön mukautuva, joten pyrittiin että se pysyy sivun ala-laidassa vaikka sisältöä ei olisi riittävästi pitämään sitä siellä. Tätä alaviite-suunnittelumallia kutsutaan yleisesti nimellä 'Sticky Footer'[17] (Ryan Fait 2012).

Bootstrapin avulla verkkosivulle on helppo luoda toimiva rakenne lisäämällä sisällön ympärille HTML-elementti * *. Tämän sisältö-elementin oletusarvoinen leveys on 940 pikseliä. Responsiiviset ominaisuudet saa myös helposti käyttöön lisäämällä tarvittava meta-tagi sivun *<head>* -elementtiin:
<meta name="viewport" content="width=device-width, initial-scale=1.0">
Tämä optimoi sisällön ruudun koon reagoivan suunnittelun periaatteiden mukaisesti, mikä on hyödyllistä varsinkin mobiililaitteilla. (Twitter Bootstrap 2012)
Liitteessä 3 havainnollistetaan, kuinka sivun esitystapa muuttuu pienemmällä ruudulla.

Sisällön tueksi tehtiin myös kaksi Liferay-rakennetemplaattia. Liitteessä 2 esitellään rakennetta, joka käyttää sisällön esitykseen kahta kolumnia. Sivummainen kolumni on hieman pienempi, sillä sen tarkoitus on toimia sivupalkkina joka sisältää useimmiten lisäinformaatiota. Lisäksi tehtiin yksikolumninen rakenne, jolloin sivupalkkia ei ole. Rakennetemplaatit vaikuttavat vain portlettien asetteluun ja niiden sisällön esittämiseen. Portaalin ulkoasun ydinkomponentteihin niillä ei ole suurempaa vaikutusta, vaan niitä voidaan muokata teeman ja sen Velocity-templaattien avulla.

12-kolumnisen rakenteen käyttö toteutetaan myös HTML div-elementtien avulla. Sisällössä voi määritellä div-elementtejä luokka-attribuutilla[18] '.row', mikä luo uuden rivin. Rivin sisälle voi määritellä useita div-elementtejä sisältöä varten. Nämä elementit

[17] Sticky Footer – tahmainen alaviite. Verkkosivun alaviite-elementti, joka pysyy aina sivun ala-laidassa.
[18] luokka-attribuutti = class attribute. HTML elementin luokitteleva attribuutti jolla voi lajitella helposti elementtejä.

vaativat luokka-attribuutin ’.spanX’, jossa X on numero väliltä 1-12. Annettu numero kertoo elementin koon ruudukossa, ja elementtien yhteiskoko tulisi olla maksimissaan 12. Esimerkiksi ’.span4’ ja ’.span8’ rivin sisällä täyttää sen halutulla 12 kolumnin määrällä. Elementin sijaintia voi myös vaihtaa antamalla sille ylimääräisen luokka-attribuutin ’.offsetX’, jossa X on määrä, kuinka monta kolumnia elementtiä halutaan siirtää oikealle. Rivejä voi myös asettaa sisäkkäisesti lisäämällä sisäkkäinen rivi ulomman kolumniin. Sisäkkäisen rivin yhteiskoko tulisi olla maksimissaan sen sisältävän kolumnin koko. (Twitter Bootstrap 2012)
Rakenneominaisuudet lisättiin Liferay-teemaan muokkaamalla asianmukaista Velocity-templaattia lisäämällä Bootstrapin elementit oikeisiin paikkoihin.

Navigointipalkki on Twitter Bootstrapin perus-ominaisuus. Kuviossa 7 on esitelty sen perusrakenne. Se koostuu muutamasta sisäkkäisestä HTML –elementistä ympäröivän div-elementin sisällä, jolla on luokka-attribuutti ’.navbar’. Muita olennaisia elementtejä on ’.navbar-inner’ sekä ’.nav’, jotka sisältävät itse sisällön.

```
<div class="navbar">
  <div class="navbar-inner">
    <a class="brand" href="#">Title</a>
    <ul class="nav">
      <li class="active"><a href="#">Home</a></li>
      <li><a href="#">Link</a></li>
      <li><a href="#">Link</a></li>
    </ul>
  </div>
</div>
```

Kuvio 7. navigointipalkin perusrakenne (Twitter Bootstrap 2012)

Navigointipalkin voi myös tehdä reagoivaksi lisäämällä sisältöä ympäröivälle elementille luokan ’.nav-collapse.collapse’ sekä lisäämällä palkkiin painikkeen ’.btn-navbar’. Tämä mahdollistaa navigointipalkin sisällön kokoonpainumisen pienemmillä ruuduilla, jolloin linkit sekä muun sisällön voi halutessa näyttää painamalla painiketta. (Liite 3)

Nämä navigointipalkin ominaisuudet lisättiin Liferayn teemaan muokkaamalla navigointi-templaattia. Koska templaatti sisältää dynaamista sisältöä, täytyi navigointi myös mukauttaa siihen. Esimerkiksi mikäli sivu sisältää ali-sivuja, näytetään ali-sivut pudotusvalikossa, mikä on myös Bootstrapin ominaisuus. Ali-sivujen olemassaolo

voidaan tarkistaa kätevästi olemassa olevalla Velocity-muuttujalla *$nav_item.hasChildren()*. Muuttujat ovat osa Velocityn templaattikieltä[19], ja niitä voi käyttää Velocityssa esimerkiksi ehdollisten lauseiden yhteydessä, analysoida tietoa tai esittää informaatiota. Muuttujien käyttö on samantyyppistä muiden ohjelmointikielten kanssa. (Apache Velocity Project)

Sticky Footer –alaviitteen käyttö mahdollistettiin CSS-tyylien avulla. Olemassa olevia toteutuksia on useita erilaisia. Perusideana kuitenkin on työntää elementti ensin sivun ala-laitaan asettamalla sisällölle 100% maksimikorkeus, ja asettamalla sisällölle negatiivinen marginaali. Marginaali tulisi olla saman suuruinen alaviitteen koon kanssa.

3.3 Ajanvaraus-portlet

Tässä luvussa esitellään ajanvaraus-prototyypin suunnittelu ja toteutus. Ajanvarausjärjestelmän prototyypin tarkoitus oli esitellä toimeksiantajalle ideaa, miten uudistetulla verkkosivustolla olisi mahdollista käyttää sähköistä ajanvarausjärjestelmää. Tavoitteena oli myös oppia uusia teknologioita sekä uusia trendejä sovelluskehityksessä. Tämän vuoksi työssä hyödynnettiin muun muassa Spring Roo-työkalua, joka nopeuttaa uusien teknologioiden käytön alkuunpääsyä huomattavasti.

Sähköisten varausjärjestelmien tärkeys korostuu nykypäivänä, kun yhä useampi asiakas käyttää hyvin aktiivisesti internetin mahdollistamia ja osa jopa olettaa esimerkiksi ajanvarauksen olevan mahdollista yrityksen verkkosivuilla. Varausjärjestelmä myös vähentää työntekijöiden taakkaa tarjoamalla asiakkaalle mahdollisuuden syöttää tietonsa järjestelmään itse.

Toimeksiantajalla on olemassa oleva potilastietojärjestelmä, mutta tässä projektissa ei toteutettu integraatiota tähän järjestelmään aikarajotteiden sekä sovelluksen prototyyppiluonteen vuoksi.

---

[19] Käännös englanninkielisestä termistä - Velocity Template Language, VTL.

3.3.1 Suunnittelu ja arkkitehtuuri

Ajanvarausjärjestelmä on Java-pohjainen portlet-sovellus, jolla on web-käyttöliittymä. Sovellus tehtiin Liferay-portaalialustalle, mutta on yhteensopiva myös muiden portaalituotteiden kanssa tuotekohtaisten konfiguraatioiden avulla.

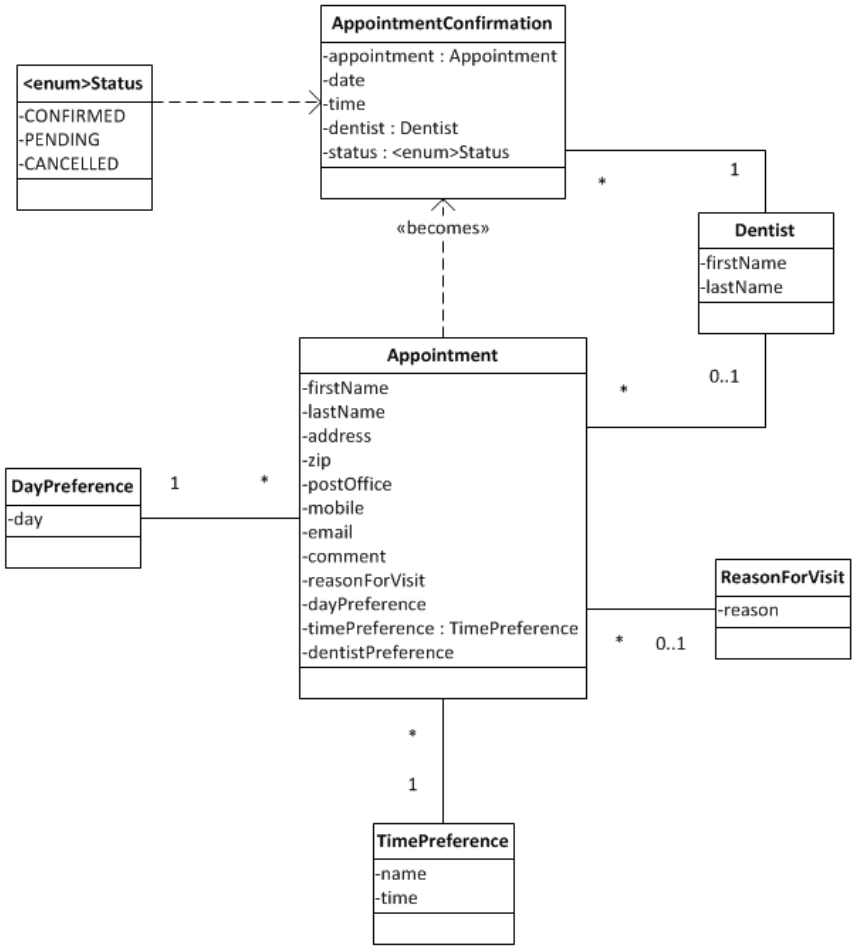
Teknologia perustuu pitkälti Spring Framework –ohjelmistokehykseen ja siihen pohjautuviin tekniikoihin. Suunnitteluvaiheessa päätettiin tutustua Spring Roo –työkaluun, joka mahdollistaa CRUD-sovelluksien nopean kehityksen tietomallin perusteella. Käyttöliittymä toimii Spring Portlet MVC[20]–teknologian avulla, jossa @Controller –annotaatiolla merkityt kontrolleri-luokat hoitavat pyyntöjen ohjaamisen asianmukaisiin näkymiin ja asettaa näkymiin vaadittavat tiedot. Näkymät ovat JSPX[21]-tiedostoja, jotka vaativat validin XML-rakenteen. Spring Roo käyttää oletuksena Apache Tiles –templaattikehystä ja JSPX-tiedostoja web-sovelluksen näkymien esitykseen. Roo tukee myös muita näkymäteknologioita, kuten GWT[22]:tä. GWT ja useimmat muut Roon tukemat näkymäteknologiat kuitenkin perustuvat JavaScriptin käyttöön, jolloin käyttäjät tarvitsevat selaimessa sille tuen.

Spring Roon oletusarvoinen sovellusarkkitehtuuri on kaksitasoinen, joka sisältää näyttökerroksen ja tietomallikerroksen. Useimmissa sovelluksissa ei vaadita erillistä palvelu-kerrosta, jotta arkkitehtuuri pysyy asianmukaisen yksinkertaisena. Suurin osa liiketoimintalogiikasta liittyy yleisesti johonkin tietomallin luokista, joten toiminnot voidaan liittää täten tietomallikerrokseen tai näyttökerroksen kontrollereihin.

Kuviossa 8 esitellään sovelluksen tietomallin rakennetta. Toiminta pyrittiin pitämään hyvin yksinkertaisena; täten sovellus on rakennettu suhteellisen yksinkertaisen ajanvaraus-luokan ympärille. Kaikki sovelluksen luokat sisältävät Spring Roon luomat CRUD-toiminnot, sekä käyttöliittymän niiden käyttöön. Ei-kirjautunut käyttäjä näkee ainoastaan ajanvaraus-lomakkeen jolla voi luoda uuden ajanvarauksen. Portaaliin kirjautunut käyttäjä puolestaan näkee kaikki sovelluksen ominaisuudet. Tällä hetkellä

[20] MVC – Model View Controller
[21] JSPX = JSP Document, Java Server Pages Document. Javan standardeihin kuuluva näkymäteknologia
[22] GWT = Google Web Toolkit

ainoastaan työntekijöillä on oikeus kirjautua portaaliin, joten rekisteröityminen portaaliin on poistettu käytöstä.

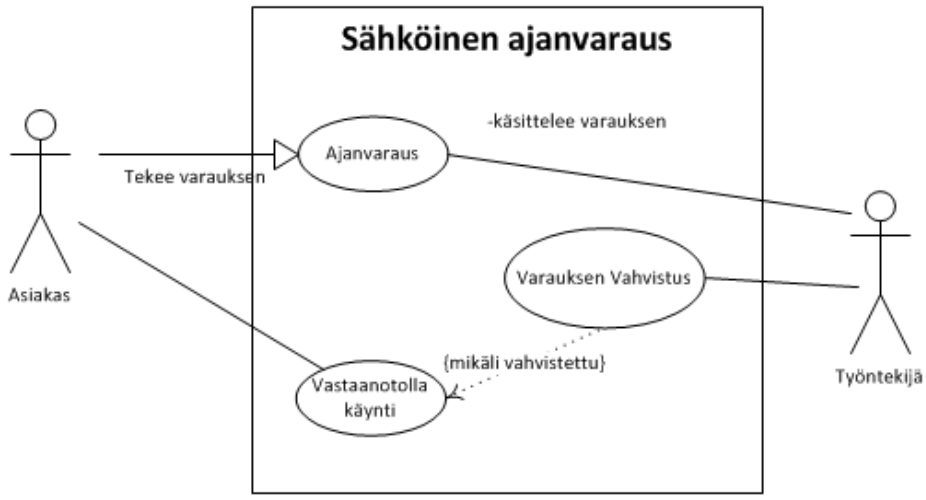


Kuvio 8. Sovelluksen tietomallin luokkakaavio

Järjestelmä mahdollistaa asiakkaille ajanvarauksen luomisen, sekä työntekijälle varauksen vahvistamisen. Kuviossa 9 esitelty käyttötapauskaavio havainnoillistaa ajanvarauksen toimintakulkua. Tapahtumien kulku tapahtuu seuraavasti: Asiakas saapuu ajanvaraus-sivulle. Tämän jälkeen asiakas täyttää varauslomakkeen ja lähettää sen järjestelmään. Seuraavaksi työntekijä käsittelee varauksen ja tekee sille vahvistuksen, jonka jälkeen asiakas saa vahvistuksesta sähköpostilla ilmoituksen ja käy vastaanotolla vahvistettuun aikaan.

Koska sovelluksen tarkoitus on ainoastaan demota varausten tekoa, ei käyttäjällä ole mahdollisuutta peruuttaa varausta sovelluksen kautta. Mikäli varausta halutaan muuttaa, täytyy käyttäjän soittaa vastaanotolle ja pyytää varauksen peruuttamista tai tietojen

muuttamista. Toiminnallisuus varauksen muuttamiseen on olemassa sovelluksessa, mutta se on piilotettu asiakkailta.



Kuvio 9. Ajanvarauksen käyttötapauskaavio

3.3.2 Käyttöliittymä & Spring Roo portlet-integraatio

Käyttöliittymässä käytetään Spring Portlet MVC –teknologiaa näkymien esitykseen. Spring Roo ei tue tätä teknologiaa täysin, joten kontrolleri-luokat tehtiin manuaalisesti. Normaalille web-sovellukselle Roo osaa luoda Spring MVC:n mukaiset kontrollerit sekä JSPX-näkymät automaattisesti. Koska Spring MVC on melko samanlainen Portlet MVC:n kanssa, voitiin yhteisiä toiminnallisuuksia käyttää jossain määrin generoiduista JSPX-tiedostoista.

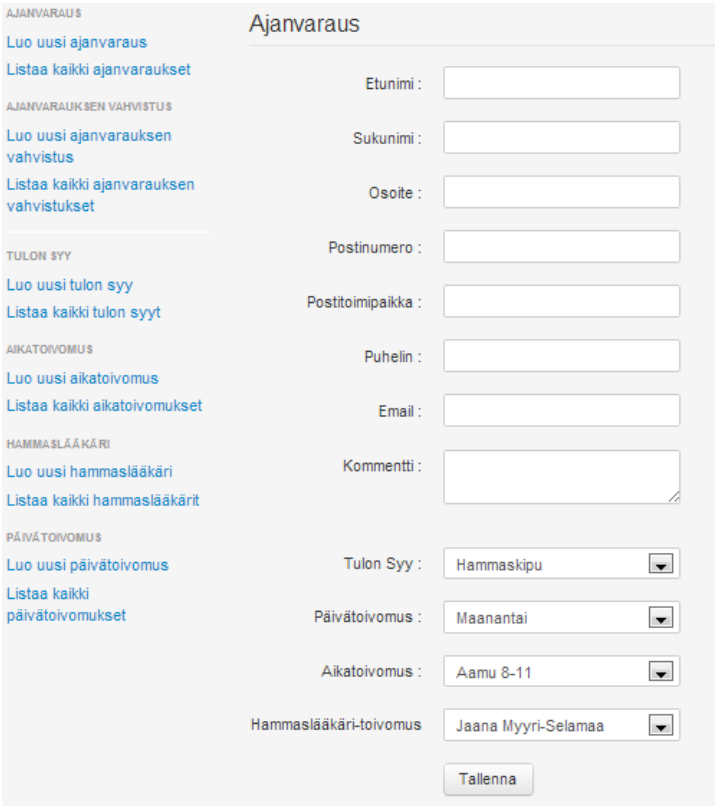
Spring Roo käyttää Apache Tiles –templaattikehystä näkymien rakentamisen tehostamiseksi. (SpringSource, Spring Roo 2012b) Tiles ei kuitenkaan toimi odotetulla tavalla portlet-ympäristössä, joten sen käyttö jouduttiin korvaamaan vaihtoehtoisella teknologialla. Spring Framework sisältää useita eri ViewResolver -luokkia, joita käytetään kontrollereissa näkymien päättelemiseen nimen perusteella sekä portlet- että servlet-ympäristöissä. Sovelluksessa käytettiin *org.springframework.web.servlet.view.JstlView* –luokkaa, joka etsii näkymän tiedoston sijainnin ja päätteen perusteella. Kuvio 10 on ote sovelluksen Spring –konfiguraatiosta, jossa määritellään käytettäviksi näkymiksi JSPX-tiedostoja */WEB-INF/views/* kansiossa. (SpringSource 2012i)

```
<bean id="viewResolver"
    class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="viewClass"
        value="org.springframework.web.servlet.view.JstlView" />
    <property name="prefix" value="/WEB-INF/views/" />
    <property name="suffix" value=".jspx" />
</bean>
```

Kuvio 10. Spring JstlView käyttö

Spring Roon generoimia Tiles-näkymiä voitiin käyttää ilman muokkauksia, mutta tällöin ne eivät sisällä muuta kuin näytettävän tiedoston ilman Tilesin määrittelemää rakennetemplaattia. Halutun rakenteen saavuttamiseksi näkymiin lisättiin manuaalisesti halutut komponentit. Oleellisia komponentteja on muun muassa menu-palkki, joka sisältää linkit perustoiminnallisuuksien käyttöön.

Käyttöliittymä sisältää toiminnallisuuden CRUD-operaatioiden tekemiseen jokaiselle tietomallin luokalle. Nämä toiminnot ovat listaus, yksityiskohtainen näkymä, uuden kohteen luonti, kohteen päivitys ja poisto. Kuviossa 11 esitellään ajanvaraus-lomaketta jossa voi luoda uuden ajanvarauksen kirjautuneen käyttäjän näkökulmasta. Käyttöliittymän vasemmassa laidassa sijaitseva menu-palkki näkyy ainoastaan kirjautuneelle käyttäjälle.

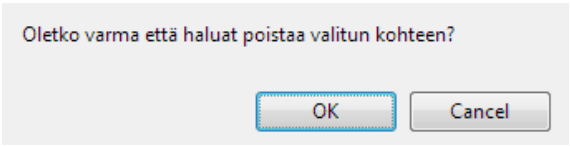


Kuvio 11. Ajanvaraus-lomake (kirjautunut käyttäjä)

Listaustoiminto listaa kaikki luodut instanssit tietystä kohteesta. Näkymä sisältää painikkeet tietojen yksityiskohtaiseen tarkasteluun, päivitykseen, poistoon sekä uuden kohteen luontiin. Kuviossa 12 on ote hammaslääkäri-objektien listausnäkymästä. Näkymä on kaikille muille kohteille rakenteeltaan sama.

Kuvio 12. Hammaslääkäri-objektien listaus-näkymä

Mikäli käyttäjä painaa poista-nappia, esitetään tapahtumasta varmistus ennen poistoa (Kuvio 13). Poistoa ei kuitenkaan suoriteta, mikäli poistettavaa objektia käytetään toisessa kohteessa. Tämä varmistaa tietojen eheyden.



Kuvio 13. Tiedon poiston varmistus

Näytettäessä objekteja käyttöliittymässä, täytyy tiedot usein muuntaa merkkijonoksi. Syötetystä informaatiosta myös usein halutaan luoda objekti. Tätä muunnostoimintaa kutsutaan tyyppikonversioksi. Spring Roo luo näkymiä generoitaessa automaattisesti tyyppikonversio –palveluluokan, joka osaa käsitellä muunnokset oikein. Sovelluksen Spring-konfiguraatiossa luokka määritellään käytettäväksi konversiopalveluksi. Servlet-ympäristössä tyyppikonversion konfigurointi on triviaalia, mutta portlet-ympäristössä konfigurointi vaatii tarkemmat määritykset. Kuviossa 14 esitellään ote sovelluksen konfiguraatiosta, jossa määritellään käytettävä tyyppikonversiopalvelu. Servlet-ympäristössä sama konfiguraatio voidaan tehdä yhdellä rivillä: <*mvc:annotation-driven conversion-service="applicationConversionService" /*>. (SpringSource, Spring Roo 2012c)

```
<mvc:annotation-driven validator="validator" />

<bean id="validator"
    class="org.springframework.validation.beanvalidation.LocalValidatorFactoryBean" />
<bean id="applicationConversionService"
    class="com.selamaa.portlet.appointments.web.ApplicationConversionServiceFactoryBean" />

<bean id="annotationMethodHandlerAdapter"
    class="org.springframework.web.portlet.mvc.annotation.AnnotationMethodHandlerAdapter">
    <property name="webBindingInitializer">
        <bean id="configurableWebBindingInitializer"
            class="org.springframework.web.bind.support.ConfigurableWebBindingInitializer">
            <property name="validator">
                <ref bean="validator" />
            </property>
            <property name="conversionService">
                <ref bean="applicationConversionService" />
            </property>
        </bean>
    </property>
</bean>
```

Kuvio 14. Tyyppikonversion määrittely Spring-portletissa

3.3.3 Validointi

Validoinnilla tarkoitetaan tietojen oikeellisuuden tarkistamista luodessa uusia objekteja. Validointi on tärkeää missä tahansa sovelluksessa tietojen eheyden varmistamiseksi. Ajanvaraus-sovelluksessa käytetään JSR-303 Java-papujen[23] validointia. Java-pavulla tarkoitetaan Sun Microsystemsin määrittelemää JavaBeans –rajapintaa ja –nimeämiskäytäntöä. Java-papu ei sisällä toimintalogiikkaa, vaan pelkästään muuttujia ja metodit niiden käyttämiseen. Java-papujen päätarkoitus on yleisesti toimia säiliönä useille muuttujille, jolloin tiedon käsittely helpottuu. Esimerkiksi sovelluksessa tehtävät ajanvaraukset ovat Java-papuja jotka sisältävät useita muuttujia tarvittaville tiedoille. Sovelluksen Java-papujen validoinnin implementaationa käytetään Hibernate Validatoria, joka on JSR-303 rajapinnan referenssitoteutus. Validointisäännöt määritellään Java-papujen muuttujissa asettamalla niille annotaatioiden avulla määritelmät. Esimerkiksi vaaditut kentät tulisi annotoida @NotNull –merkinnällä. Tällöin tietoja ei tallenneta mikäli tällä annotaatiolla merkitty muuttuja on tyhjä. Annotaatioita on lukuisia, kattaen lähes kaikki tarpeet. Muun muassa @Size –merkinnällä voi määritellä tekstikentän minimi- ja maksimipituuden; @Email –annotaatiota voi käyttää sähköpostiosoitteen oikean muodon tarkistukseen; @Future päivämääräkentässä varmistaa, että päivämäärä on tulevaisuudessa. Mikäli mikään

[23] käännös englanninkielisestä termistä JavaBean, usein käytetään myös nimitystä POJO, Plain Old Java Object

annotaatio ei vastaa vaadittuja validointisääntöjä, on mahdollista tehdä myös omia annotaatioita. (Nicolas Frankel 2010)

Java-papujen validointi yhdistetään myös yleisesti käyttöliittymään, jotta käyttäjä tietää syöttäneensä virheellistä tietoa. Spring Roo käyttää Dojo Toolkit JavaScript-kirjastoa lomakkeiden validoinnissa. JavaScript –validointi on käyttäjäystävällinen, sillä sivua ei tarvitse ladata uudelleen virheiden havaitsemiseksi. Dojo Toolkitillä on kuitenkin joitain yhteensopivuusongelmia portlet-ympäristössä, joita ei pystytty projektin aikana ratkaisemaan. Sen lomakkeiden validointi toimii sovelluksessa oikein, mutta ulkoasu ei näy oikein. Tämä tekee Dojon käytön tällöin epäkäytännölliseksi, joten sovelluksessa se on otettu väliaikaisesti pois käytöstä. Sen sijaan mikäli tiedoissa on virheitä, ohjataan käyttäjä takaisin lomake-sivulle ja mahdollisesti kerrotaan, missä virheelliset tiedot sijaitsevat. Tällöin sivu ladataan uudelleen vähentäen käytettävyyttä. Dojo on mahdollista korvata esimerkiksi Roo –liitännäisellä, mutta tässä projektissa tätä ei pystytty tekemään. (SpringSource, Spring Roo 2012d)

3.3.4 Tietoturva

Sovelluksen tietoturva käyttää Liferayn tunnistautumista ja käyttäjähallintaa hyväkseen. Sovelluksella ei ole tarvetta itsenäiselle käyttäjähallinnalle tai tunnistukselle. Käytännössä tämä tarkoittaa sitä, että portletilla mahdollisuus käyttää käyttäjän tietoja näkymissä tai ohjelmalogiikassa. Käyttäjätiedot sijaitsevat Liferay-kohtaisessa ThemeDisplay –objektissa, joka on mahdollista hakea portlet-pyynnöstä. Kuviossa 15 on ote ohjelmakoodista, jossa käytetään @ModelAttribute –annotaatiolla merkittyä metodia kontrolleri-luokassa lisäämään tietomalliin tieto, onko käyttäjä kirjautunut. Lisättyä tietoa voidaan siten käyttää sille annetulla nimellä *signedIn* esimerkiksi näkymissä piilottamaan sisältöä, tai estämään toimintoja kontrollerin metodeissa. ThemeDisplay –objekti sisältää myös muun muassa käyttäjän roolitiedot sekä portaalikohtaisia tietoja. (Liferay 6.0 API)

```
@ModelAttribute("signedIn")
public boolean isSignedIn(PortletRequest request){
    ThemeDisplay td  =(ThemeDisplay)request.getAttribute(WebKeys.THEME_DISPLAY);
    return td.isSignedIn();
}
```

Kuvio 15. Käyttäjän tietojen hakeminen portlet-pyynnöstä

Sovelluksessa tietoja käytetään piilottamaan näkymissä menu, joka sisältää linkkejä tietojen muokkaamiseen. Ainoastaan kirjautuneilla käyttäjillä, eli työntekijöillä, on oikeus nähdä linkit. Muokkausoperaatioita ei ole kuitenkaan estetty tuntemattomilta käyttäjiltä ohjelmatasolla, sillä tietoturva ei tarvitse olla kattava vielä esittelyversiossa. Täten sisällön ehdollinen näyttäminen on riittävä esittelytarpeisiin. Portlet ympäristössä linkit ovat kuitenkin hyvin pitkiä ja vaikealukuisia, joten tunnistamattoman käyttäjän on silti hyvin vaikea päästä käsiksi muokkaustoimintoihin, mikä lisää tietoturvaa.
Tietoa voi käyttää näkymässä JSTL[24]-ehtolausekkeella *<c:if test="${signedIn}">*. JSTL on standardisoitu tagikirjasto, joka tarjoaa yleisiä toiminnallisuuksia JSP –sivuilla, kuten ehtolausekkeita, iteraattoreita ja muita yleishyödyllisiä tageja. (Oracle 2012)

[24] JSTL = JavaServer Pages Standard Tag Library

4 Pohdintaa

4.1 Saavutetut tulokset

Lähes puoli vuotta kestäneen projektin aikana saatiin paljon aikaiseksi. Tuotetut tulokset eivät kuitenkaan ole täysin käyttöönottovalmiita, mikä ei tosin ollut tavoitteenakaan. Toivotut ominaisuudet saatiin kuitenkin toteutettua, mikä on tärkeää jatkokehityksen kannalta. Projekti oli kokonaisuudessaan erittäin opettavainen, ja varsinkin projektinhallinnasta sekä uusista teknologioista opittiin paljon.

Projekti oli myös teknisesti haastava, sillä siinä käytettiin aikaisemmin tuntemattomia teknologioita kuten Spring Roo –työkalua. Uudet teknologiat aiheuttivat myös ennalta-arvaamattomia vaikeuksia, joiden selvittämiseen käytettiin huomattavasti aikaa. Useimmat ongelmat liittyivät yhteensopimattomuuteen portlet-ympäristössä, joten ratkaisut saattavat ratketa mikäli portlet-yhteensopivuutta parannetaan näiden teknologioiden uusissa julkaisuissa. Dokumentaatiota näiden teknologioiden käytöstä portleteissa on myös niukasti, mikä hankaloitti ongelmien ratkomista.

Ajanvarausjärjestelmä vaatii vielä hiomista, mutta esittelytarkoitukseen se on täysin riittävä. Sen avulla voidaan tehdä yksinkertaisen käyttöliittymän avulla varauksia, ja kirjautumalla portaaliin sisään voi varauksia myös vahvistaa. Sovellus sisältää kuitenkin joitain bugeja, joita ei ehditty projektin aikana korjaamaan. Esimerkiksi Spring Roon käyttämä Dojo-toolkit ei ole täysin yhteensopiva portlet-ympäristössä, joten se päätettiin poistaa käytöstä esittelyä varten.

Sivuston ulkoasu saatiin onnistuneesti uudistettua modernimpaan muotoon. Ajankohtaisten ulkoasuteknologioiden käyttö varmisti paremman yhteensopivuuden mobiililaitteiden kanssa, mutta myös edesauttoi tyylikkään vaikutelman luomista. Sisällön luonti ei kuulunut projektin vaatimuksiin, joten se vaatii vielä töitä. Materiaalia tarvitaan myös toimeksiantajalta, jotta sisältöä on mahdollista lisätä. Sivuston värimaailmaa tullaan tulevaisuudessa muuttamaan, joten tuotetussa teemassa pyrittiin hyödyntämään paljon neutraaleja värejä kuten mustaa, harmaata ja valkoista.

Myös vaatimukset vaihtelivat kesken projektia useaan kertaan, mikä vaikutti aikataulun lievään hidastumiseen. Alkuperäistä ajoitussuunnitelmaa venytettiin noin kuukaudella, jotta tulokset saataisiin tuotettua paremmalla laadulla. Aikataulun parempi suunnittelu oli kuitenkin opettavaista projektinhallinnolliselta kannalta, ja kunnollisella suunnittelulla projektin aikatauluongelmat saatiin vaikeuksien kautta minimoitua. Projektin tekeminen kokoaikaisen työpaikan ohessa oli kuitenkin stressaavaa, mikä myös korostaa aikataulutuksen tärkeyttä.

4.2 Jatkotoimenpiteet

Projektin päätyttyä toimeksiantajan kanssa suunnitellaan verkkosivuston käyttöönottoa. Käyttöönotto ei kuulunut projektiin, sillä se olisi saattanut viivästyttää projektia itsestä riippumattomista syistä. Ennen käyttöönottoa tulisi tuottaa sisältöä sivuille ja tarvittaessa tehdä ulkoasumuokkauksia vastaamaan tarkemmin yrityksen imagoa. Mahdollista on myös luoda teemalle ylimääräinen väri-templaatti, joka mahdollistaa teeman värimaailman helpon vaihdon, pitäen alkuperäisen version tallessa. Käyttöönotossa Liferay tulisi optimoida tuotantokäyttöön sopivaksi parantamalla sen suorituskykyä, johon on useita keinoja. Ilman mitään muutoksia Liferay sisältää paljon turhia ominaisuuksia, jotka on hyvä poistaa käytöstä. Liferayn konfiguraatio vaatii myös kehitysasetuksien vaihdon tuotantoasetuksiin, mikä vaikuttaa suorituskykyyn positiivisesti.

Ajanvarausjärjestelmä on käyttöönoton kannalta prototyyppivaiheessa. Ennen mahdollista jatkokehitystä tulisi korjata olemassa olevat yhteensopivuusongelmat ja muita bugeja. Lomakkeiden validointi käyttöliittymäkerroksessa sisältää JavaScriptin kanssa ongelmia, mikä on yksi suurimmista esteistä testikäytölle. Jotta sovellusta olisi kannattava käyttää, olisi hyvä lisätä myös joitain ominaisuuksia kuten tiedon etsiminen ja sisällön filtteröinti.

4.3 Arviointi

Opinnäytetyö suoritettiin HAAGA-HELIAn opinnäytetyöprosessin mukaisesti. Opinnäytetyö oli yksi tekijän vaativimmista yksilöprojekteista sekä tekniseltä että

projektinhallinnolliselta kannalta. Projekti sisälsi kaikki opinnäytetyöprosessin osat, ja aikaa siihen käytettiin yhteensä noin 400 tuntia.

Projektin toteutustarkoitus oli mielenkiintoinen, ja sen toteuttamisesta oltiin jo keskusteltu lähes vuosi sitten toimeksiantajan kanssa. Tekijä teki vuonna 2007 yrityksen nykyiset kotisivut, joten oli luonnollista ehdottaa päivityksen toteuttamista, jolle oli jo olemassa tarve. Jotta projektissa olisi hieman enemmän töitä kuin tarvetta, päätettiin kehittää myös prototyyppi ajanvarausjärjestelmästä. Päätös oli hyvä, sillä sovelluksen kautta oli mahdollista oppia paljon uutta sovelluskehityksestä ja uusista teknologioista. Osa teknologioista oli aiemmin tuttuja, mutta projektin aikana myös niistä saatiin paljon enemmän tietoa. Varsinkin Liferay –teeman kehityksen saloihin pääsi projektissa syvemmälle. Projektinhallinnasta opittiin myös paljon, sillä kaikki hallinta ja dokumentointi tehtiin kokonaan itse. Dokumentaation määrä oli välillä yllättävä, ja usein siihen kului yhtä paljon aikaa kuin itse kehittämiseen.

Toimeksiantajayritykselle projekti oli erittäin hyödyllinen. Uuden verkkosivualustan avulla verkkosivujen käyttö mahdollistaa uusia käyttötarkoituksia. Nykyiset sivut ovat staattiset, joten niiden muokkaaminen ei ole helppoa ilman hieman teknistä osaamista. Uudistetuissa sivuissa tämä on enemmän kuin mahdollista, ja laajentamisvaraa käytössä on myös. Mahdollista olisi esimerkiksi luoda yrityksen intranet-sivusto kätevästi portaaliin. Toimeksantaja on erittäin innokas uusista ominaisuuksista, joten projektin päättymisen jälkeen useita uusia ideoita on mahdollista toteuttaa. Myös käyttöönottoprosessi aloitetaan projektin päättyessä, mikä tuo lisäkokemusta sovelluskehityksen koko elinkierrosta.

Mielenkiintoisimpiin osuuksiin kuului ajanvarausjärjestelmän toteutus, sillä se sisälsi runsaasti uusia teknologioita. Spring Roo oli aikaisemmin melko tuntematon teknologia, mutta projektin myötä sen tarjoamat mahdollisuudet tuli hyvinkin tutuiksi. Myös teknologien rajoitteet tulivat esille, kuten useiden teknologioiden ongelmat portlet-ympäristössä. Kokonaisuudessaan projekti oli opettavainen, ja sen avulla saavutettiin odottamattomiakin hyötyjä. Työ oli hyvin työelämälähtöinen, sillä sen kautta saatiin paljon kokemusta kokonaisvaltaisesta kehitysprojektityöstä.

Lähteet

Oris Oy kotisivu. Saatavilla: http://www.oris.fi/. Viitattu 5.11.2012

Mike Baker: *What is a Software Framework? And why you should like 'em?* 10.2009. Saatavilla: http://info.cimetrix.com/blog/bid/22339/What-is-a-Software-Framework-And-why-should-you-like-em

Liferay Inc: *What is a Portal?* 2012a. Saatavilla: http://www.liferay.com/products/what-is-a-portal/web-platform. Viitattu 26.10.2012

Java-Source.net: Open Source Portals in Java. Saatavilla: http://java-source.net/open-source/portals. Viitattu 26.10.2012

Liferay Inc: *Liferay Portal*. 2012b. Saatavilla: http://www.liferay.com/products/liferay-portal/overview. Viitattu 26.10.2012

O'Reilly Media: *What is a Portlet*. 9.2005. Saatavilla: http://oreilly.com/java/archive/what-is-a-portlet.html

Liferay Wiki: *Portlets*. 9.2009a. Saatavilla: http://www.liferay.com/web/guest/community/wiki/-/wiki/Main/Liferay+Portlets. Viitattu 27.10.2012

Liferay Wiki: *Portal Hook Plugins*. 7.2008b. Saatavilla: http://www.liferay.com/community/wiki/-/wiki/Main/Portal+Hook+Plugins

Liferay Inc: *Portal Features*. 2012c. Saatavilla: http://www.liferay.com/products/liferay-portal/features/portal. Viitattu 27.10.2012.

Liferay Documentation: *Creating Liferay Themes*. 2012a. Saatavilla: http://www.liferay.com/documentation/liferay-portal/6.0/development/-/ai/creating-liferay-them-5. Viitattu 20.10.2012.

Liferay Documentation: *Creating Liferay Themes – Anatomy of a Theme.* 2012b. Saatavilla: http://www.liferay.com/documentation/liferay-portal/6.0/development/-/ai/anatomy-of-a-theme. Viitattu 25.10.2012.

Liferay Wiki: *Access Objects from Velocity.* 4.2009b. Saatavilla: http://www.liferay.com/community/wiki/-/wiki/Main/Access+Objects+from+Velocity. Viitattu 26.10.2012.

Liferay Documentation: *Creating Lifeary Layout Templates.* 2012c. Saatavilla: http://www.liferay.com/documentation/liferay-portal/6.1/development/-/ai/lp-6-1-dgen05-creating-liferay-layout-templates-0. Viitattu 26.10.2012

Javier Cuevas. Diacode.com: *Front-End Frameworks: a quick overview.* 3.2012. Diaesitys saatavilla: http://www.slideshare.net/Diacode/frontend-frameworks-a-quick-overview. Viitattu 27.10.2012.

Kayla Knight, Smashing Magazine: *Responsive Web Design: What It Is and How To Use It.* 1.2011. Saatavilla: http://coding.smashingmagazine.com/2011/01/12/guidelines-for-responsive-web-design/. Viitattu 25.10.2012.

Alex Sellier, LESS. Saatavilla: http://lesscss.org. Viitattu 27.10.2012

SpringSource: *Spring Projects - Spring Framework.* Saatavilla: http://www.springsource.org/spring-framework. Viitattu 24.10.2012a.

SpringSource: *Spring Framework Reference Documentation – The IoC container.* Saatavilla: http://static.springsource.org/spring/docs/3.0.x/spring-framework-reference/html/beans.html. Viitattu 26.10.2012b.

SpringSource: *Spring Framework Reference Documentation – Aspect Oriented Programming with Spring.* Saatavilla: http://static.springsource.org/spring/docs/current/spring-framework-reference/html/aop.html. Viitattu 26.10.2012c.

SpringSource: *Spring Framework Reference Documentation – Transaction Management.* Saatavilla: http://static.springsource.org/spring/docs/current/spring-framework-reference/html/transaction.html. Viitattu 26.10.2012d.

SpringSource: *Spring Framework Reference Documentation – Web MVC framework*. Saatavilla: http://static.springsource.org/spring/docs/current/spring-framework-reference/html/mvc.html. Viitattu 26.10.2012e.

SpringSource: *Spring Framework Reference Documentation – Portlet MVC framework*. Saatavilla: http://static.springsource.org/spring/docs/current/spring-framework-reference/html/portlet.html. Viitattu 26.10.2012f.

SpringSource: *Spring Framework Reference Documentation – Additional Capabilities of the ApplicationContext – Internationalization using MessageSource*. Saatavilla: http://static.springsource.org/spring/docs/current/spring-framework-reference/html/beans.html#context-functionality-messagesource. Viitattu 26.10.2012g.

SpringSource: *Spring Framework Reference Documentation – Spring 3 Validation*. Saatavilla: http://static.springsource.org/spring/docs/current/spring-framework-reference/html/validation.html#validation-beanvalidation. Viitattu 27.10.2012h.

SpringSource: *Spring Framework Reference Documentation – View technologies*. Saatavilla: http://static.springsource.org/spring/docs/3.0.x/spring-framework-reference/html/view.html. Viitattu 7.11.2012i

Perficient: *Spring – ADD Developer – Annotated Driven Development*. 2.2009. Saatavilla: http://www.slideshare.net/kensipe/spring-3-annotated-development. Viitattu 27.10.2012.

SpringSource: *Spring Roo Reference Documentation – History*. Saatavilla: http://static.springsource.org/spring-roo/reference/html/background.html#background-history. Viitattu 26.10.2012a.

SpringSource: *Spring Roo Reference Documentation – Introduction*. Saatavilla: http://static.springsource.org/spring-roo/reference/html/intro.html. Viitattu 26.10.2012b.

SpringSource: *Spring Roo Reference Documentation – Web MVC Add-On – Application Conversion Service*.Saatavilla: http://static.springsource.org/spring-roo/reference/html/base-web.html#conversion-service. Viitattu 7.11.2012c.

SpringSource: *Spring Roo Reference Documentation – Beginning With Roo: The Tutorial*. Saatavilla: http://static.springsource.org/spring-roo/reference/html/beginning.html. Viitattu 7.11.2012d

SpringSource: *Spring Projects – Spring Roo*. Saatavilla: http://www.springsource.org/spring-roo. Viitattu 26.10.2012i.

The Eclipse Foundation: *Eclipse documentation.* Saatavilla: http://www.eclipse.org/documentation/. Viitattu 27.10.2012.

Apache Maven Project: *What is Maven?* 10.2012a. Saatavilla: http://maven.apache.org/what-is-maven.html. Viitattu 27.10.2012.

Apache Maven Project: *Maven Getting Started Guide.* 2012b. Saatavilla: http://maven.apache.org/guides/getting-started/index.html. Viitattu 27.10.2012.

Apache Maven Project: *Introduction to the Strandard Directory Layout.* 2012c. Saatavilla: http://maven.apache.org/guides/introduction/introduction-to-the-standard-directory-layout.html. Viitattu 27.10.2012

Apache Maven Project: *Introduction to the Build Lifecycle.* 2012d. Saatavilla: http://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html. Viitattu 27.10.2012.

Git: *About.* 2012. Saatavilla: http://git-scm.com/about. Viitattu 26.10.2012

Mika Koivisto: *Developing Liferay Plugins with Maven*. 5.2012. Saatavilla: http://www.slideshare.net/koivimik/developing-liferay-plugins-with-maven. Viitattu 29.10.2012.

Ryan Fait: *Make the Footer Stick to the Bottom of a Page.* 2012. Saatavilla: http://ryanfait.com/resources/footer-stick-to-bottom-of-page/. Viitattu 7.11.2012.

Twitter Bootstrap kotisivu. Saatavilla: http://twitter.github.com/bootstrap/. Viitattu 7.11.2012.

Apache Velocity Project: *User Guide.* Saatavilla: http://velocity.apache.org/engine/releases/velocity-1.5/user-guide.html. Viitattu 7.11.2012.

Nicolas Frankel, Java DZone: *Bean Validation and JSR 303*. 3.2010. Saatavilla: http://java.dzone.com/articles/bean-validation-and-jsr-303. Viitattu 7.11.2012.

Oracle: *The Java EE 5 Tutorial – JavaServer Pages Standard Tag Library*. 2010. Saatavilla: http://docs.oracle.com/javaee/5/tutorial/doc/bnakc.html. Viitattu 7.11.2012.

Liferay 6.0 API: *ThemeDisplay.* Saatavilla: http://docs.liferay.com/portal/6.0/javadocs/com/liferay/portal/theme/ThemeDisplay.html. Viitattu 7.11.2012.

Liitteet

Liite 1 Termistö

CMS

Content Management System.

Java EE

Java Enterprise Edition. Ohjelmointialusta Java-pohjaisten yritys-sovellusten kehittämiseen. Laajentaa Java Standard Editionin ominaisuuksia.

Portlet

Portaaliin liitettävä ohjelmistokomponentti.

CSS

Cascading Syle Sheets. Käytetään tiedostojen ulkoasun muokkaukseen, yleisimmin HTML-verkkosivuissa. Voidaan myös käyttää muiden tiedostomuotojen kanssa.

HTML

HyperText Markup Language. Yleinen teknologia verkkosivujen näyttämiseen selaimella. Selaimen tarkoitus on lukea HTML sivuja ja esittää ne näkyviksi sivuiksi.

LDAP

Lightweight Directory Access Protocol. Käytetään pääasiassa käyttäjäoikeuksien tarkistamiseen.

Liferay Plugins SDK

Työkalu, jota käytetään Liferay-lisäyksien kehittämiseen. Sisältää komentosarjoja[25], joiden avulla voidaan luoda Liferaylle muun muassa teemoja, portletteja ja hook- sekä layout-plugineja

[25] komentosarja = script – kokoelma suoritettavia komentoja tietylle järjestelmälle.

Apache Velocity

Java-pohjainen avoimen lähdekoodin templaattimoottori joka tarjoaa templaattikielen objektien määrittelyyn ja käyttöön Java-ohjelmassa. Sen käyttötarkoituksia on muun muassa ohjelmakoodin generoiminen templaattien perusteella ja web-sovellusten käyttöliittymän rakentamisen tehostaminen tarjoamalla sivulle paikan dynaamiselle informaatiolle.

SOAP

Simple Object Access Protocol. Tietoliikenneprotokolla, jonka tarkoituksena on vaihtaa informaatiota verkkopalveluiden välillä. Toimii pääasiassa HTTP-protokollan yli, mutta toimii myös useiden muiden tekniikoiden kanssa, kuten Java Message Service-teknologian (JMS) välityksellä.

JSP

Java Server Pages. Java-teknologia, jonka avulla voidaan luoda dynaamisesti generoituja verkkosivuja.

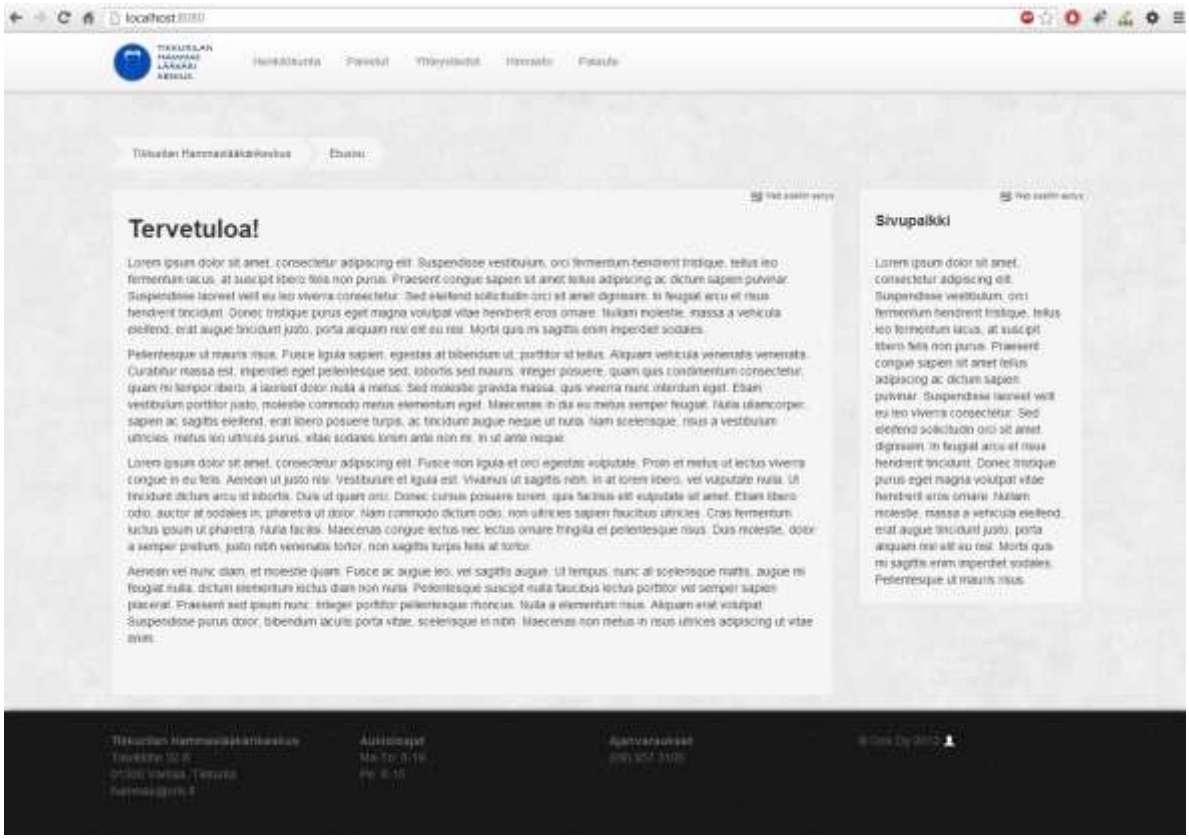
JSPX

JSP Document. Java-teknologia JSP-sivujen kehitykseen XML-pohjaisina.

CRUD

Create, Read, Update, Delete. Tarkoittaa perustoimintoja, joilla voi muokata tietoja. (Luonti, Luku, Päivitys, Poisto)

Liite 2 Esimerkki kaksikolumnisesta sivurakenteesta



Liite 3 Esimerkki sivun mobiilinäkymästä