OULUN SEUDUN
AMMATTIKORKEAKOULU

Ville Jänkälä

HOITOAIKAVARAUSTOIMINNON TOTEUTTAMINEN DAISY-JÄRJESTELMÄÄN

HOITOAIKAVARAUSTOIMINNON TOTEUTTAMINEN DAISY-JÄRJESTELMÄÄN

Ville Jänkälä
Opinnäytetyö
Kevät 2013
Tietotekniikan koulutusohjelma
Oulun seudun ammattikorkeakoulu

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu
Tietotekniikan koulutusohjelma, ohjelmistokehitys

Tekijä: Ville Jänkälä
Opinnäytetyön nimi: Hoitoaikavaraustoiminnon toteuttaminen Daisy-järjestelmään
Työn ohjaaja: Kari Laitinen
Työn valmistumislukukausi ja -vuosi: Kevät 2013 Sivumäärä: 36

Daisy-järjestelmä on päiväkodin arkeen suunniteltu digitaalinen järjestelmä. Opinnäytetyön aiheena oli suunnitella ja toteuttaa hoitoaikavaraustoiminto Daisy-järjestelmään. Varaustoiminnon avulla pystytään tekemään hoitoaikavarauksia päiväkoteihin, joissa on käytössä Daisy-järjestelmä. Internet-pohjaisen varausjärjestelmän tavoitteena on vähentää sekä päiväkodin henkilökunnan että vanhempien työmäärää ja helpottaa päiväkodin työntekijöiden työaikasuunnittelua.

Hoitoaikavaraustoiminto rakennettiin web-standardien mukaisesti HTML-, CSS- ja JavaScript-tekniikoita hyväksikäyttäen. Työssä käytettiin Microsoftin ASP.NET-ohjelmistokehystä Microsoft Visual Studio -ohjelmointiympäristön yheydessä.

Projektin aikana saatiin toteutettua toimiva ja käyttövalmis hoitoaikavaraustoiminto DaisyNet-sovellukseen. Vaatimusmäärittelyn mukaiset toiminnallisuudet saatiin tehtyä, vaikka aikataulu oli suhteellisen tiukka. Testijakson aikana käyttäjiltä saatiin palautetta ja parannusehdotuksia toimintoon ja hoitoaikavarausprosessia tullaan kehittämään tulevaisuudessa käyttäjäystävällisempään suuntaan.

Asiasanat: HTML, CSS, ASP.NET, web-sovellus

ABSTRACT

Oulu University of Applied Sciences
Information technology, software development

Author: Ville Jänkälä
Title of thesis: Implementation of nursery hour booking feature in to Daisy system
Supervisor: Kari Laitinen
Term and year when the thesis was submitted: Spring 2013 Pages: 36

Daisy system is a digital environment designed for day care centers. The target of the thesis was to implement a nursery hour booking feature in to Daisy system. Feature is used to book nursery hours in day care centers which are using the Daisy system. Online booking reduces the work load of both parents and day care center staff and it makes it easier to manage and plan the daily working time of employees.

Nursery hour booking feature was built using standardized web techniques such as HTML, CSS and JavaScript. Project was created using Microsoft ASP.NET framework and Microsoft Visual Studio IDE.

During the project a working and ready-to-use version of the booking feature was developed in to the DaisyNet application. All the features included in the requirements list were successfully developed, even though the schedule was somewhat tight. During a test phase users gave feedback and improvement ideas and in the future the feature will be developed further in to more user friendly direction.

Keywords: HTML, CSS, ASP.NET, web application

ALKULAUSE

Kiitos SPDesign OY:lle mahdollisuudesta toteuttaa opinnäytetyönä mielenkiintoinen ja hyödyllinen sovellus. ASP.NET-ohjelmistokehyksen käyttö web-suunnittelussa ei ollut kovinkaan tuttu ympäristö entuudesta, joten opinnäytetyö antoi minulle mahdollisuuden tutustua monipuolisesti uudenlaiseen tapaan toteuttaa web-sovelluksia.

5.5.2013

Ville Jänkälä

SISÄLLYS

SANASTO

ASP.NET	Microsoftin kehittämä ohjelmistokehys
CSS	Määrittelee web-sivujen ulkoasun
HTML	Web-sivujen rakenteen kuvauskieli
IDE	Ohjelmistokehitysympäristö
JavaScript	Selaimessa suoritettava dynaaminen koodi
jQuery	JavaScript-kirjasto
Qt	Alustariippumaton ohjelmistokehitysympäristö

1 JOHDANTO

Opinnäytetyöni aiheena on suunnitella ja toteuttaa hoitoaikavaraustoiminto web-pohjaiseen DaisyNet-sovellukseen. DaisyNet on osa suurempaa kokonaisuuttaa, Daisy-järjestelmää, joka on suunniteltu helpottamaan päiväkotien henkilökunnan ja lasten huoltajien arkea. DaisyNet on vanhemmille suunniteltu työkalu, jonka avulla he voivat mm. päivittää lastensa tietoja ja lähettää viestejä päiväkodille. Uutena toimintona sovellukseen toteutetaan hoitoaikavaraustyökalu, joka vähentää sekä vanhempien että päiväkodin työntekijöiden työmäärää.

Aiemmin hoitoaikavaraukset on toimitettu päiväkodeille paperisina lomakkeina. Digitaalisten varausten tekemisessä on huomattavia etuja paperisiin varauksiin nähden: digitaalisesta arkistosta on helppo hakea ja muokata tietoa ja niistä saadaan helposti tehtyä erilaisia koonteja, raportteja ja yhteenvetoja. DaisyNetin kautta varauksia tehdessä huoltajat näkevät reaaliajassa, kuinka monta hoitotuntia he ovat millekin kuukaudelle varanneet. Tämä on tärkeä tieto huoltajille, sillä päivähoitosopimukset ja hinnat määräytyvät hoitotuntimäärien perusteella. Päiväkodin henkilökunnan taas on helpompi suunnitella työntekijöiden työtunteja, kun he näkevät paikallaolevien lasten määrän päivän ja kellonajan perusteella.

Hoitoaikavaraustyökalu toteutettiin ASP.NET-tekniikalla web-pohjaiseen DaisyNet-sovellukseen. Sovellus on toimiva kokonaisuus, johon luodaan oma sivu hoitoaikavaraustoimintoa varten. Huoltajat voivat käyttää sovellusta kotikoneiltaan selaimen välityksellä. Kaikki huoltajien tekemät varaustiedot tallennetaan palvelimella sijaitsevaan tietokantaan, josta tiedot synkronoituvat Daisy-tuoteperheen muiden sovellusten kesken.

2 DAISY-JÄRJESTELMÄN YLEISKUVAUS

Daisy-järjestelmä on laaja kokonaisuus, johon kuuluu useita eri internet- ja mobiilisovelluksia. Sovellukset toimivat eri osa-alueilla ja ne on suunniteltu eri käyttäjäryhmille päiväkotiympäristössä. Sovellukset välittävät tietoa toisillensa, joten tiedon muokkaaminen yhdessä laitteessa vaikuttaa myös muiden laitteiden näkymiin.

2.1 DaisyManager

DaisyManager on päiväkodin työntekijöiden käyttöön tarkoitettu web-sovellus. DaisyManagerin kautta työntekijät pääsevät katselemaan hoidossa olevien lasten tietoja sekä omia työaikatietoja. DaisyManagerin kautta työntekijät voivat myös lähettää viestejä muille työntekijöille ja lasten huoltajille. Sovelluksella voidaan myös katsella erilaisia raportteja ja yhteenvetoja päiväkodin työntekijöistä ja lapsista.

2.2 DaisyNet

DaisyNet on päiväkodissa hoidossa olevien lasten huoltajien työkalu, jonka avulla he voivat päivittää lastensa tietoja, kuten esim. allergioita ja huoltajien yhteystietoja. Sovelluksen avulla huoltajat voivat myös lähetää viestejä ja tiedoksiantoja päiväkodin työntekijöille. Sovellukseen integroidaan hoitoaikavaraustoiminto, jonka avulla lapsen hoitoaikojen ilmoittaminen päiväkodille helpottuu huomattavasti.

2.3 DaisyMobile

Daisy-järjestelmään kuuluu Qt:llä toteutettu mobiilisovellus, jonka avulla päiväkodin henkilökunta voi tehdä erilaisia kirjauksia järjestelmään. Kirjausten tekeminen tapahtuu NFC-tageja käyttämällä, joten ylimääräisten valintojen ja painallusten määrä vähenee. Puhelimella tehdyt kirjaukset siirtyvät suoraan Daisy-tietokantaan, ja tietoja voidaan katsella DaisyNetin ja DaisyManagerin

kautta. Kirjausten tekemisen lisäksi sovelluksella voidaan tarkistaa esimerkiksi päiväkodissa olevien lasten lukumäärä.

2.4 DaisyServer

Daisy-järjestelmän kulmakivi on DaisyServer, joka mahdollistaa koko järjestelmän olemassaolon. Serverillä on yhteys tietokanta, johon kaikki järjestelmässä kulkeva tieto tallennetaan. Kaikki Daisy-järjestelmään kuuluvat sovellukset ovat yhteydessä DaisyServeriin ja tieto synkronoituu sovellusten välillä sen kautta. DaisyServerillä on myös mahdollisuus luoda yhteys kahteen ulkopuoliseen tietokantaan, Efficaan ja Titaniaan ja tiedot näiden tietokantojen välillä voidaan synkronoida päivittäin. DaisyServer toimii Microsoft Windows Server 2008 R1 -ympäristössä.

3 TYÖSSÄ KÄYTETYT TEKNOLOGIAT

3.1 HTML

HTML (Hypertext Markup Language) on standardoitu kieli, jolla kuvataan web-sivujen rakennetta. HTML-standardia ylläpitää yritysten ja yhteisöjen yhteenliittymä W3C. Tällä hetkellä maailmanlaajuisesti käytössä on HTML 4.01 -standardi, mutta HTML 5 tekee jo kovasti tuloaan. HTML 5:n spesifikaatio ei kuitenkaan ole vielä täysin valmis, joten useat selaimet eivät osaa käyttää kaikkia sen tuomia uusia ominaisuuksia. (HTML. 2013.) Tämän työn toteutuskieleksi valittiin HTML 4.01 -kieli, sillä ohjelmiston toimivuus haluttiin taata mahdollisimman monella alustalla.

HTML-kielen syntaksi on hyvin yksinkertainen. Kaikki HTML-kielellä kuvattavat elementit alkavat ja päättyvät kulmasulkeisiin kirjoitetulla tagilla (kuva 1). Tagien väliin voidaan kirjoittaa esimerkiksi tekstisisältöä, joka näky näytöllä normaalina tekstinä. HTML-syntaksia voidaan kirjoittaa millä tahansa tekstieditorilla, eikä se vaadi minkäänlaisia lisäosia toimiakseen.

```
<a href="http://www.google.com">Linkki Googleen.</a>
```

KUVA 1. Esimerkki HTML-elementistä, joka näyttää tekstilinkin Googleen

Kun käyttäjä kirjoittaa selaimensa osoiteriville osoitteen, lähettää selain HTTP-pyynnön palvelimelle ja palvelin palauttaa selaimelle sivun HTML muodossa. Selain tulkitsee saapuneen HTML-tiedon ja renderöi tämän perusteella sivun käyttöliittymän käyttäjälle. Koska HTML ei itsessään ota kantaa siihen, miltä sivun pitäisi ulkoasullisesti näyttää, tarvitaan siihen erillinen kuvauskieli, CSS.

3.2 CSS

CSS (Cascading Style Sheets) on kieli, jolla HTML-elementeille voidaan luoda erilaisia tyyliasetuksia. CSS:n avulla voidaan muuttaa esimerkiksi elementin väriä, kokoa ja sijaintia (kuva 2). Kuten HTML:ää, myös CSS:ää ylläpitää W3C.

Tällä hetkellä laajasti käytössä on CSS2-versio. Osittain selaimet tukevat jo CSS3:n joitain ominaisuuksia, mutta niitä ei tule käyttää sovelluksen toiminnan kannalta kriittisiin ominaisuuksiin. (CSS. 2013.)

Yksi CSS-kielen vahvuuksista on se, että sitä käytettäessä itse HTML-tiedoston rakenne ja syntaksi pysyvät siistinä. HTML-tiedostoon voidaan linkittää erillinen CSS-tiedosto, josta HTML-elementtien tyylit ja asettelu ladataan. Näin ollen sivuston ulkoasua, teemaa ja värejä voidaan helposti vaihtaa vain muuttamalla CSS-tiedoston määrityksiä.

CSS-kielen syntaksi on selkeää. CSS-tiedostossa viitataan HTML-elementteihin niiden taginimen tai *class*- ja *id*-attribuuttien perusteella. Jos halutaan useita samanlaisia HTML-elementtejä, voidaan niille kaikille antaa sama *class*-attribuutti ja määrittää tyyliasetukset sen avulla. *Id*-attribuutilla taas voidaan yksilöidä elementtejä, ja niihin viittaamalla voidaan määrittää yksittäisen elementin tyyliasetuksia. CSS-tiedostojen luontiin voidaan käyttää mitä tahansa tekstieditoria.

```
29  #footer {
30      background: #95c235;
31      position: absolute;
32      width: 100%;
33      bottom: 0px;
34      height: 100px;
35      color: #484848;
36      font-weight: bold;
37      font-size: 16px;
38  }
39
40  .custom {
41      display: block;
42      float: left;
43      margin-left: 120px;
44  }
45
46  #footer .custom div {
47      float: left;
48      margin: 20px;
49      margin-top: 30px;
50  }
51
```

KUVA 2. CSS-tiedosto Notepad++-ohjelmassa

3.3 JavaScript

JavaScript on käyttäjän selaimessa ajettava ohjelmointikieli, jonka avulla HTML-sivuille voidaan lisätä dynaamisia toimintoja, kuten esimerkiksi hiirellä liikuteltavia laatikoita. JavaScriptin kehitti alun perin Netscape Communications Corporation ja nykyisin se on yksi suosituimmista ohjelmointikielistä web-pohjaisissa sovelluksissa (JavaScript. 2013). JavaScriptiä tukevat kaikki suurimmat ja tunnetuimmat selaimet, ja nykyisin onkin enemmän sääntö kuin poikkeus, että internet-sivut sisältävät jonkinlaista JavaScript-koodia. Palveluiden ylläpitäjien kannalta yksi JavaScriptin parhaimmista ominaisuuksista on se, että koodi suoritetaan selaimen kautta käyttäjän omalla tietokoneella eikä koodin suorittaminen vie laskentatehoa palvelimella.

3.4 jQuery

JQuery on JavaScriptillä toteutettu avoimen lähdekoodin kirjasto, jonka tarkoituksena on helpottaa dynaamisten toimintojen toteuttamista JavaScript-ohjelmointikielellä. Ensimmäinen versio kirjastosta julkaistiin tammikuussa 2006 ja sen kehittäjänä toimi yhdysvaltalainen ohjelmoija John Resig (jQuery History. 2013). JQuery yksinkertaistaa ja yhtenäistää erilaisten sovellusten toteuttamistapoja, ja sen syntaksi on pyritty tekemään mahdollisimman ymmärrettäväksi ja helppolukuisaksi. JQueryä tukevat käytännössä kaikki selaimet, jotka tukevat JavaScriptiä, ja se onkin nykyään maailman suosituin JavaScript-kirjasto (jQuery. 2013).

3.5 ASP.NET

Microsoftin kehittämän ASP.NET-ohjelmistokehyksen avulla voidaan luoda dynaamisia web-sovelluksia käyttäen mitä tahansa tuettua .NET-ohjelmointikieltä, kuten esimerkiksi C#:a. Ensimmäinen 1.0 versio ASP.NET-ympäristöstä julkaistiin tammikuussa 2002, ja tällä hetkellä viimeisin julkaistu versio on ASP.NET 4.5, joka julkaistiin elokuussa 2012 (ASP.NET. 2013). ASP.NET erottaa käyttöliittymän ja toiminnallisuuskoodin eri tiedostoihin, joten ulkoasun ja toimintojen hallinnoiminen onnistuu helposti. Käyttöliittymän

rakenne luodaan omaan tiedostoon käyttäen yleistä HTML-kuvauskieltä, jonka jälkeen tähän pohjaan voidaan asettaa haluttu sisältö toisen tiedoston kautta käyttäen esimerkiksi C#-ohjelmointikieltä. ASP.NET sisältää myös lukuisia valmiita kontrolleja, joita sovelluksissa voidaan käyttää, mikä helpottaa ja nopeuttaa toimintojen toteuttamista sovelluksiin. Yksinkertaista ASP-koodia voidaan kirjoittaa millä tahansa tekstieditorilla, mutta todellisen hyödyn siitä saa irti käyttämällä Microsoftin omaa Visual Studio -ohjelmistokehitysympäristöä.

4 HOITOAIKAVARAUSTOIMINNON VAATIMUKSET

4.1 Käyttöliittymä

Koska hoitoaikavaraustoiminto on tarkoitettu kaikkien päiväkodin lasten huoltajien käyttöön, on selvää, että käyttäjäkunta tulee olemaan hyvin laaja. Ihmisten tietotekninen taso vaihtelee hyvinkin paljon, joten käyttöliittymästä täytyy tehdä mahdollisimman yksinkertainen ja ymmärrettävä. Tämä asettaa suuren haasteen toiminnon toteuttamiseen, sillä tasapainon löytäminen yksinkertaisen, mutta tarpeeksi kattavan toiminnon välillä on hankalaa. Lisärajoitteita luo myös sovelluksen ennaltamääritelty ulkoasu ja suhteellisen pieni pinta-ala, johon varaustoiminto on toteutettava.

Selkeyden vuoksi hoitoaikavarausnäkymässä käytetään tavallista kuukausittaista kalenterinäkymää, johon merkataan värikoodein, jos päivälle on tehty merkintöjä.

4.2 Toiminnallisuus

Hoitoaikojen varaus on toimintarpeiltaan suhteellisen yksinkertainen. Käyttäjän tulee pystyä valitsemaan päivä tai useampia päiviä, joille haluaa varauksia tehdä, sekä varauksia koskevat kellonajat. Päivät valitaan kalenterista päivämäärää klikkaamalla tai käyttäjän halutessa koko viikko viikkonumeroa klikkaamalla. Varauksia pitää myös pystyä muokkaamaan ja poistamaan. Varausten tekemistä, poistamista ja muokkaamista tulee rajoittaa järjestelmässä niin, että muutoksia pystyy tekemään vain tulevaisuuteen. Päivämäärän lisäksi käyttäjän tulee pystyä määrittämään varauksille alku- ja loppukellonajat. Kaikki varaukset merkitään kalenterin päivämääriin punaisella taustavärillä ja kalenterin alapuolelle listataan aktiivisen kuukauden varausten päivämäärät ja kellonajat.

5 HOITOAIKAVARAUSTOIMINNON TOTEUTUS

5.1 Käyttöliittymän toteuttaminen

Koska varaustoiminto on osa DaisyNet järjestelmä, olivat raamit hoitoaikavaraustoiminnon käyttöliittymälle jo olemassa. DaisyNetin kaikki sivut käyttävät pohjanaan samaa ulkoasutiedostoa, ja eri sivuilla sisältölaatikoihin ladataan eri sisältöä (kuva 3). Keltaiselle pohjalle toteutettiin kalenterinäkymä, josta voidaan valita päivämääriä, ja sen vierelle aseteltiin pudotusvalikot kellonaikavalintaa varten. Sisältölaatikko 2:een sijoitettiin tarvittavat panikkeet varausten tallentamista ja poistamista varten.

KUVA 3. DaisyNet-käyttöliittymän rakenne

5.1.1 Kalenterinäkymä

Jotta hoitoaikavarausten tekeminen ja hahmottaminen olisi mahdollisimman helppoa, päätettiin varaussivulle toteuttaa tietokoneen kalenteria muistuttava komponentti, jonka kautta päivämääriä voidaan valita. Kalenterin tuli myös näyttää värikoodein päivämäärät, joille merkintöjä on tehty. Kalenterinäkymän

mahdollisti helposti ASP.NETissä valmiina oleva kalenterikontrolli, jota voitiin käyttää pohjana hoitoaikavaraustoiminnon rakentamisessa.

Valmiin ASP.NET-kalenteri-kontrollin näyttäminen sivulla on yksinkertaista. Sivun HTML-koodin sekaan upotetaan ASP-tagi, joka kuvaa kalenteria. Tagin sisään voidaan määritellä erilaisia ominaisuuksia kalenterille. Ominaisuudet määritellään attribuuttien avulla. Esimerkiksi *OnVisibleMonthChanged*-attribuutti määrittää, minkä nimistä funktiota kutsutaan, kun käyttäjä vaihtaa kalenterissa näytettävää kuukautta (kuva 4). Varsinainen funktion toiminnallisuus määritellään omassa tiedostossa erillään ulkoasusta, jolloin koodin rakenne pysyy selkeänä ja koodia on helppo ylläpitää.

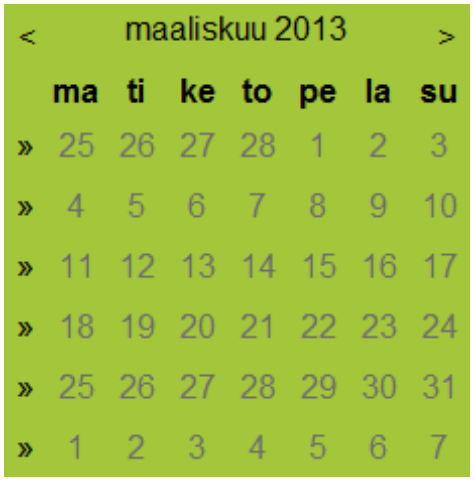
```
<asp:Calendar ID="Calendar1" OnVisibleMonthChanged="VisibleMonthChanged" OnInit="setDefaultDate"
OnSelectionChanged="updateCalendarSelection" OnDayRender="checkForPlacements" runat="server"
class="datecalendar" SelectionMode="DayWeek" TitleStyle-BackColor="#a4c63b" SelectWeekText="»">
<SelectedDayStyle BackColor="#325e24" CssClass="tdSelected" ForeColor="#ffffff" />
</asp:Calendar>
```

KUVA 4. ASP.NET-kalenterin näyttämiseen tarvittava koodi

Koska valmiin ASP.NET kalenteri-kontrollin ominaisuuksin ei kuulu viikkonumeroinnin näyttäminen, täytyi tämä ominaisuus toteuttaa itse. Viikkonumeroinnin näyttäminen päätettiin toteuttaa JavaScriptin avulla, sillä silloin ei tarvitse muokata varsinaista ASP.NET kalenteri-luokkaa.Tällöin varmistutaan siitä, että kontrolli on edelleen yhteensopiva kaikkien järjestelmässä olevien sivujen kanssa, eikä viikkonumeroinnin lisääminen aiheuta ongelmia muiden ominaisuuksien kanssa. Koska projektissa on käytössä jQuery-kirjasto, voidaan kirjaston määrittelemiä metodeita käyttää helpottamaan JavaScript-koodin kirjoittamista.

Käytännössä viikkonumeroinnin näyttäminen tapahtuu niin, että ensiksi selain renderöi sivulle kalenterin ilman viikkonumeroa (kuva 5). Tämän jälkeen selain suorittaa JavaScript-koodin (kuva 6), jonka seurauksena kalenteriin tulee viikkonumerot näkyviin. Todellisuudessa JavaScriptin suorittamiseen kuluu niin

vähän aikaa, että käyttäjä näkee sivulla ainoastaan kalenterin, jossa viikkonumerot ovat näkyvissä (kuva 7).



KUVA 5. ASP.NET-kalenteri

```
var $calendar = $('#ctl00_MainContent_Calendar1');
var swn = $('#ctl00_MainContent_startWeekNumber').val();

$('td', $calendar).each(function (i) {

    var $td = $(this);
    var $link = $('a', $(this));

    if ($link.text() == '»') {

        if (swn > 52) {
            swn = 1;
        }
        $link.addClass('weekSelectLink');
        $link.text(swn);
        swn++;
    }

});
```

KUVA 6. jQuery/JavaScript-koodi viikkonumeroinnin lisäämisestä kalenteri-kontrolliin

KUVA 7. ASP.NET-kalenteri, johon on lisätty viikkonumerot

5.1.2 Kellonajan valinta

Hoitoaikavarauksia koskevien kellonaikojen määritystä varten sivuille toteutettiin omat kontrollit. Koska käyttäjän itse esimerkiksi tekstikenttään syöttämässä datassa on virhenäppäilyn mahdollisuus suuri, päätettiin kellonaikojen määrittäminen toteuttaa alasvetovalikoilla, joihin on listattu vuorokauden kellonajat viidentoista minuutin välein. Tämä mahdollistaa hoitoaikojen varaamisen riittävällä tarkkuudella, eikä käyttäjän virhenäppäilylle jää mahdollisuutta. Alasvetovalikoiden toteutukseen käytettiin ASP.NETin omaa *DropDownList*-komponenttia, joka renderöityy sivulle tavallisena HTML:n standardoituna *select*-tagina (kuva 8). *DropDownList*-komponentin käytön etuna on kuitenkin se, että valikkoon voidan lisätä valittavia vaihtoehtoja helposti sekä valikkoa voidaan muokata ja hallinnoida kätevämmin toiminnallisuuskoodin kautta.

```
<table>
    <tr>
        <td>
            <asp:Literal ID="Literal4" Text="<%$Resources:DaisyNetResource, Start %>" runat="server" /><br />
            <asp:DropDownList ID="startTime" runat="server" Width="140px"></asp:DropDownList>
        </td>
        <td>
            <asp:Literal ID="Literal5" Text="<%$Resources:DaisyNetResource, Finish %>" runat="server" /><br />
            <asp:DropDownList id="endTime" runat="server" Width="140px"></asp:DropDownList>
        </td>
    </tr>
</table>
```

KUVA 8. Tyhjien alasvetovalikoiden näyttämiseen tarvittava koodi

5.1.3 Viikon valinta ja väriselitteet

Koska kalenterista on mahdollista valita kaikki viikon päivät yhdellä klikkauksella, haluttiin kalenterin viereen tuoda mahdollisuus määrittää, koskeeko koko viikon valinta ainoastaan arkipäiviä vai myös viikonloppua. Tämä valintamahdollisuus toteutettiin *RadioButton*-elementtien avulla (kuva 9). Kun kahdelle *RadioButton*-elementille asetetaan sama *GroupName*-attribuutti, osaa selain käsitellä komponentteja niin, että vain toinen komponentti voi olla valittuna kerrallaan.

```
<asp:RadioButton id="weekSelectTypeFive" Text=" <%$Resources:DaisyNetResource, MonFri %>"
Checked="true" AutoPostBack="false" GroupName="weekSelectType" runat="server"/>
<br />
<asp:RadioButton id="weekSelectTypeSeven" Text=" <%$Resources:DaisyNetResource, MonSun %>"
AutoPostBack="false" GroupName="weekSelectType" runat="server"/>
```

KUVA 9. Kahden RadioButton-komponentin luonti

Koska kalenterin päivät merkitään erilaisilla väreillä sen mukaan, minkätyyppisiä merkintöjä päiville on tehty, kalenterin viereen täytyi tehdä lista väreistä ja niiden selitteistä. Listaus toteutettiin yksinkertaisesti HTML:n div- ja span-tagien avulla (kuva 10). Värikoodiit voidaan ilmaista HTML:n ja CSS:n yhteydessä heksadesimaalilukuina.

```
<div><span style="background:#325e24;"></span> Valittu päivä</div>
<div><span style="background:#ff0300;"></span> Varattu hoitoaika</div>
<div><span style="background:#06aded;"></span> Poissaolo</div>
<div><span style="background:#ffa500;"></span> Loma-aika</div>
```

KUVA 10. Värien selitelaatikoiden luonti

5.1.4 Toimintopainikkeet

Hoitoaikavaraussivulle täytyi toteuttaa kaksi toimintopainiketta, Tallenna ja Poista, joiden avulla varauksia voidaan tallentaa ja poistaa. Painikkeiden

luomiseen käytettiin ASP.NETin *Button*-kontrollia, koska siihen voidaan helposti kiinnittää funktio, joka suoritetaan, kun kontrollia klikataan (kuva 11). Painikkeet sijoittuvat käyttäliittymäkuvassa (kuva 3) Sisältölaatikko 2:een.

```
<asp:Button ID="SaveDayCareTimes" runat="server" Text="Tallenna"
    OnClick="SaveDayCareTimes_OnClick"/>
<asp:Button ID="DeleteDayCareTimes" runat="server" Text="Poista"
    OnClick="DeleteDayCareTimes_OnClick"/>
```

KUVA 11. Kahden Button-elementin luonti ja funktioiden kiinnittäminen OnClick-tapahtumaan

KUVA 12. Valmis käyttöliittymä hoitoaikavarauksille

5.2 Toiminnallisuuksien implementointi

Toiminnallisuuksien toteuttamista varten ASP.NET-projekteissa on erillinen ns. code behind -tiedosto, johon voidaan kirjoittaa mitä tahansa tuettua .NET-ohjelmointikieltä. Tässä projektissa ohjelmointikielenä käytettiin C#:a. Code behind -tiedostossa voidaan viitata käyttöliittymän elementteihin ja kontrolleihin niiden ID-attribuutin avulla ja niiden sisältöä voidaan muuttaa tai niihin voidaan

kiinnittää erilaisia funktioita, joita suoritetaan, kun kontrollia esimerkiksi klikataan.

5.2.1 Päivämäärien valinta kalenterikontrollista

Hoitoaikavaraustoiminnossa käytettävä ASP.NET-kalenterikontrolli tukee valmiiksi päivän valintaa päivänumerosta klikkaamalla tai viikon valintaa viikkonumerosta klikkaamalla. Kalenterista ei kuitenkaan oletuksena pysty valitsemaan kuin yhden päivän tai viikon kerrallaan, ja uuden valinnan tekeminen kadottaa vanhan valinnan. Koska kalenteri haluttiin sellaiseksi, että siitä voidaan valita niin monta päivää tai viikkoa kuin käyttäjä haluaa, täytyi tämä toiminto toteuttaa kalenteriin itse.

Kalenterikontrollissa on *OnSelectionChange*-niminen event listener, joka tarkkailee milloin käyttäjä muuttaa kalenterin päivämäärävalintaa. Tähän event listeriin kiinnitettiin oma funktio, *updateCalendarSelection* (kuva 13), johon toteutettiin koodinpätkä huolehtimaan siitä, että edelliset päivämäärävalinnat eivät katoa uusia valintoja tehdessä. Koodi toimii niin, että joka valinnan jälkeen kalenterin senhetkinen valinta tallennetaan käyttäjän sessioon. Kun käyttäjä tekee uuden valinnan, luetaan sessiosta edelliset valitut päivämäärät ja ne lisätään nykyiseen valintaan. Näin päivämäärävalintoja voi tehdä useita menettämättä vanhoja valintoja. Jotta käyttäjä pystyisi myös poistamaan haluamiaan päivämääriä valinnasta, koodiin tehtiin toiminto, joka tarkistaa löytyykö tehdystä valinnasta samoja päiviä, joita on tallennettuna jo käyttäjän sessioon. Mikäli samoja päivämääriä löytyy, nämä päivämäärät poistetaan valinnoista. Käyttäjä voi siis tehdä valinnan klikkaamalla haluamaansa päiväämäärää ja poistaa valinnan klikkaamalla valittua päiväämäärää uudelleen.

```
protected void updateCalendarSelection(object sender, EventArgs e)
{
    List<DateTime> oldList = new List<DateTime>();

    try
    {
        oldList = (List<DateTime>)Session["SelectedDates"];
        oldList = oldList.GroupBy(d => d.Date).Select(g => g.First()).ToList();
    }
    catch { }

    List<DateTime> newSelection = new List<DateTime>();

    foreach (DateTime dt in Calendar1.SelectedDates)
    {
        if ((!IsWeekEnd(dt) || Calendar1.SelectedDates.Count < 2)
                            || weekSelectTypeSeven.Checked)
        {
            if (dt.Date > maxDisabledDay)
                newSelection.Add(dt);
        }
    }

    Calendar1.SelectedDates.Clear();

    List<DateTime> finalSelection = new List<DateTime>();

    try
    {
        var DifferencesList = oldList.Where(x => !newSelection.Any(x1 => x1.Date == x.Date))
        .Union(newSelection.Where(x => !oldList.Any(x1 => x1.Date == x.Date)));

        finalSelection = DifferencesList.ToList();
    }
    catch { }

    foreach (DateTime fs in finalSelection)
    {
        Calendar1.SelectedDates.Add(fs);
    }

    Session["SelectedDates"] = Calendar1.SelectedDates;

    list.Clear();
}
```

KUVA 13. updateCalendarSelection-funktion toteutus

5.2.2 Kellonaikojen valinta alasvetovalikoista

Käyttäjien syöttämien virheellisten arvojen välttämiseksi haluttiin kellonaikavalinnat toteuttaa alasvetovalikoina ASP.NET:in *DropDownList*-kontrollin avulla. Alasvetovalikoihin tarvittiin vuorokauden kellonajat viidentoista minuutin tarkkuudella, joten valikoiden täyttämistä varten tehtiin

populateDropdown-funktio (kuva 14). Funktioon syötetään parametrinä *DropDownList*-kontrolli, joka halutaan täyttää, sekä indeksi, joka määrittää, kuinka mones arvo on oletuksena valittuna listassa. Funktiossa luodaan kaksi *DateTime*-tyyppistä oliota, joista ensimmäinen (*dt*) asetetaan aikaan 1.1.2012 klo 00:00:00 ja toinen (*dt2*) 2.1.2012 klo 00:00:00. Tämän jälkeen käynnistetään while-silmukka, jossa *dt*-olion kellonajan arvo lisätään parametrinä syötettyyn listaan. Silmukan lopussa *dt*:n arvoon lisätään 15 minuuttia. Silmukkaa suoritetaan niin kauan, kun *dt*:n arvo on pienempi kuin *dt2*:n, ja näin saadaan valikko täytettyä kellonajoilla viidentoistaminuutin välein. Tämä funktio suoritetaan erikseen sekä alku- että loppu-kellonaikavalikoille sivulatauksen yhteydessä. Muuta toimintoa alasvetovalikoihin ei tarvitse tehdä, sillä valittujen arvojen lukeminen valikoista tapahtuu vasta hoitoaikavarauksen tallennuksen yhteydessä.

```
protected void populateDropdown(DropDownList dropdown, int selectedIndex)
{
    DateTime dt = new DateTime(2012, 1, 1,0,0,0);
    DateTime dt2 = new DateTime(2012, 1, 2,0,0,0);

    while (dt < dt2)
    {
        ListItem item = new ListItem();
        item.Text = dt.ToString("HH:mm");
        item.Value = dt.ToString("HH:mm");
        dropdown.Items.Add(item);
        dt = dt.AddMinutes(15);
    }

    dropdown.SelectedIndex = selectedIndex;
}
```

KUVA 14. populateDropdown-funktion toteutus

5.2.3 Hoitoaikavarauksien tallentaminen

Hoitoaikavarauksien tallennus tietokantaan tapahtuu siinä vaiheessa, kun käyttäjä painaa sivulla olevaa Tallenna-painiketta. Painikkeen *OnClick*-tapahtumaan on kiinnitetty *SaveDayCareTimes_OnClick*-funktio (kuva 15), joka siis suoritetaan silloin, kun käyttäjä klikkaa painiketta.

SaveDayCareTimes_OnClick-funktio on toteutettu code behind -tiedostossa kaikkien muiden toiminnallisuusfunktioiden tapaan.

Kun käyttäjä klikkaa Tallenna-painiketta, luetaan käyttäjän valitsemat päivät sessiosta *List<DateTime>*-tyyppiseen *newList*-nimiseen olioon. Heti funktion alussa tarkistetaan, että käyttäjä on valinnut edes yhden päivän. Mikäli valittuja päivämääriä ei löydy yhtään kappaletta, näytetään käyttäjälle virheviesti. Jos taas valittuja päivämääriä löytyy, siirrytään funktiossa seuraavaan vaiheeseen. Foreach-silmukalla käydään läpi yksitellen kaikki *newList*-listan oliot. Silmukan alussa tarkistetaan, että kyseinen päivämäärä ei ole menneisyydessä. Jokaisella kierroksella tarkistetaan, löytyykö jo kyseiselle päivälle käyttäjän tekemiä varauksia. Mikäli varauksia löytyy, luetaan vanhan varauksen tiedot tietokannasta ja päivitetään varauksen kellonaikatiedot. Jos varauksia ei löydy kyseiselle päivälle, luodaan uusi *NursingTime*-olio, johon asetetaan päivämääräksi kyseinen päivä ja kellonajoiksi alasvetovalikoista valitut kellonajat.

Kellonaikavalikot on rakennettu käyttöliittymään ASP.NETin oman *DropDownList*-kontrollin avulla, joten niistä voidaan lukea valitut arvot helposti myös code behind -tiedostossa. Kellonaika-arvot ovat valikoissa muodossa hh:mm, joten valituista arvoista täytyy erotella tunnit ja minuutit, jotta todellinen aikakomponentti saadaan muodostettua *DateTime*-tyyppiseen olioon. Tämä onnistuu *Split*-funktion avulla, johon syötetään parametrinä merkki, jonka perustella merkkijono halutaan jakaa. Funktio palauttaa yksiulotteisen taulukon, jonka alkiot sisältävät alkuperäisen merkkijonon pilkottuna parametrina syötetyn merkin kohdalta. Pilkkomisen jälkeen luodaan uudet *DateTime*-oliot varauksen alku- ja loppupäivämäärille, joihin syötetään parametreinä vuosi-, kuukausi-, päivä- sekä kellonaikakomponentit. Nämä oliot sijoitetaan *NursingTime*-olion *start*- ja *end*-jäsenmuuttujiin, jonka jälkeen hoitoaikavaraus on valmis tallennettavaksi.

```
protected void SaveDayCareTimes_OnClick(object sender, EventArgs e)
{
    string errorMessage = "";

    if (childInfo.SelectedChildId != null)
    {
        List<DateTime> newList = new List<DateTime>();

        newList = (List<DateTime>)Session["SelectedDates"];

        if (newList.Count < 1)
        {
            errorMessage = "Ei valittuja päivämääriä";
        }

        foreach (DateTime dt in newList)
        {
            if (dt > maxDisabledDay)
            {
                List<NursingTime> nursingTimes = PersonService
                    .Instance.LoadNursingTimes(childInfo.SelectedChildId, null, dt.Date, dt.Date);

                NursingTime newNursingTime;

                if (nursingTimes.Count < 1)
                {
                    newNursingTime = new NursingTime();
                    newNursingTime.PersonId = childInfo.SelectedChildId;
                    newNursingTime.Date = dt.Date;

                    List<Placement> childPlacement = PersonService.Instance
                        .LoadPlacements(childInfo.SelectedChildId, true, null);

                    try
                    {
                        newNursingTime.PlacementId = childPlacement.First().Id;
                    }
                    catch { }
                }
                else
                {
                    newNursingTime = nursingTimes.First();
                }

                string[] startTimeFromInput = startTime.SelectedValue.Split(':');
                string[] endTimeFromInput = endTime.SelectedValue.Split(':');

                try
                {
                    newNursingTime.Start = new DateTime(dt.Date.Year, dt.Date.Month, dt.Date.Day,
                                                        Int32.Parse(startTimeFromInput[0]),
                                                        Int32.Parse(startTimeFromInput[1]), 0);
                    newNursingTime.End = new DateTime(dt.Date.Year, dt.Date.Month, dt.Date.Day,
                                                      Int32.Parse(endTimeFromInput[0]),
                                                      Int32.Parse(endTimeFromInput[1]), 0);
                    PersonService.Instance.SaveNursingTime(newNursingTime);
                }
                catch
                {
                    errorMessage = "Muutoksia ei voitu tallentaa";
                }
            }
        }

        list.Clear();
        Calendar1.SelectedDates.Clear();

        if(errorMessage == "")
            this.Master.ShowPopupBox(true, Resources.DaisyNetResource.NoteSaved);
        else
            this.Master.ShowPopupBox(false, errorMessage);
    }
}
```

KUVA 15. SaveDayCareTimes_OnClick-funktion toteutus

5.2.4 Hoitoaikavarausten poistaminen

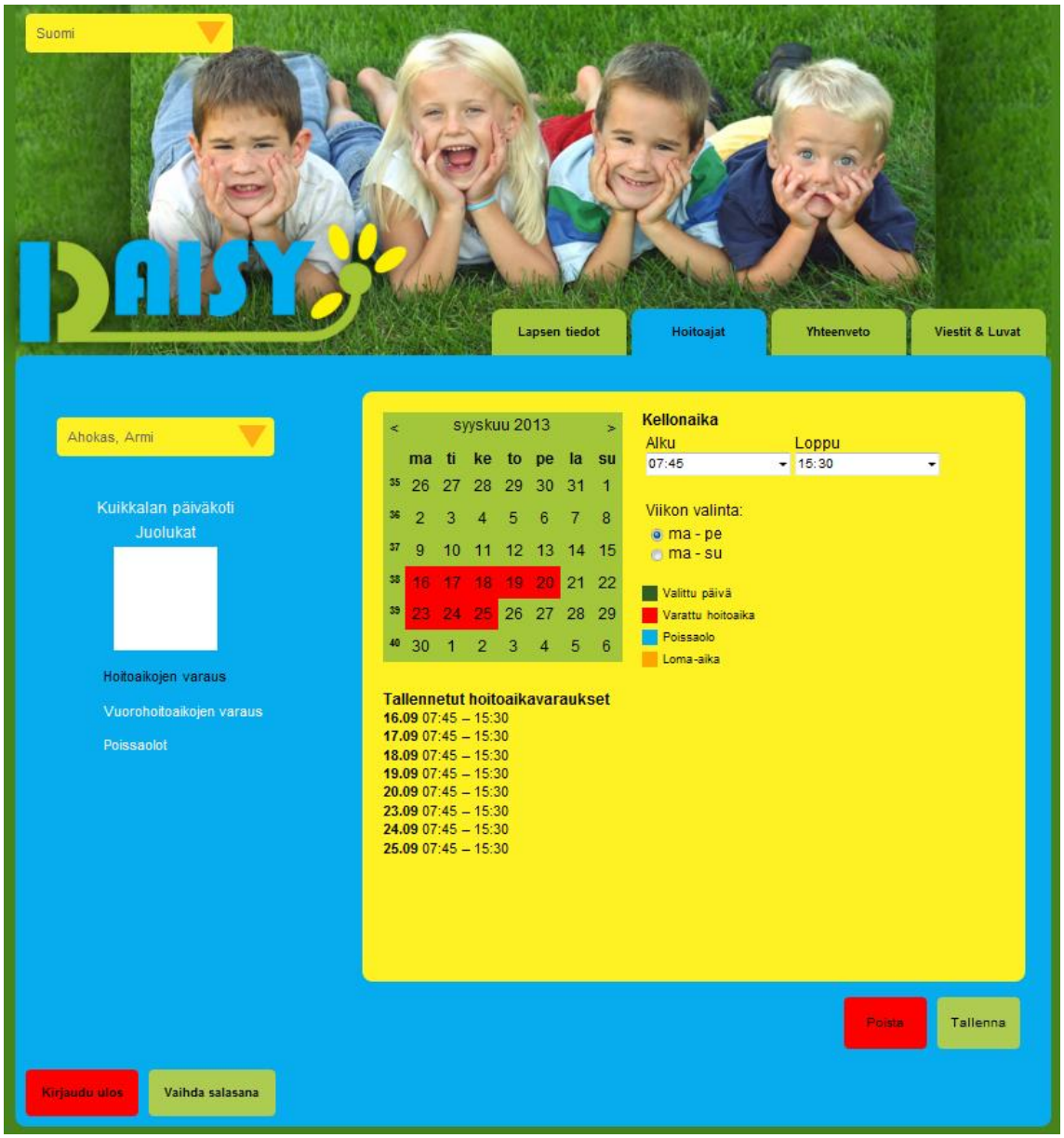
Käyttäjän tekemien hoitoaikavarausten poistaminen toteutettiin samalla periaatteella kuin varauksien tallentaminenkin. Käyttäjä valitsee kalenterista klikkaamalla päivämäärät, joilta haluaa varaukset poistaa, ja tämän jälkeen klikkaa Poista-painiketta. Painikkeen *OnClick*-tapahtumaan on kiinnitetty *DeleteDayCareTimes_OnClick*-funktio (kuva 16). Kun käyttäjä painaa Poista-painiketta, funktio lataa käyttäjän valitsemat päivämäärät sessiosta *List<DateTime>*-tyyppiseen olioon, jonka jälkeen listan päivämäärät käydään yksitellen läpi foreach-silmukan avulla. Jokaisella kierroksella tarkistetaan, löytyykö kyseiselle päivälle käyttäjän hoitoaikavarauksia. Mikäli varauksia löytyy eikä kyseinen päivämäärä ole menneisyydessä, varaukset poistetaan.

```
protected void DeleteDayCareTimes_OnClick(object sender, EventArgs e)
{
    string errorMessage = "";

    if (childInfo.SelectedChildId != null)
    {
        List<DateTime> newList = (List<DateTime>)Session["SelectedDates"];

        updateReserveDays();

        if (newList.Count < 1)
        {
            errorMessage = "Ei valittuja päivämääriä";
        }
        foreach (DateTime dt in newList)
        {
            if (dt > maxDisabledDay)
            {
                List<NursingTime> nursingTimes = PersonService.Instance
                    .LoadNursingTimes(childInfo.SelectedChildId, null, dt.Date, dt.Date);

                if (nursingTimes.Count > 0)
                {
                    NursingTime newNursingTime;
                    newNursingTime = nursingTimes.First();
                    PersonService.Instance.DeleteNursingTime(newNursingTime);
                }
                else
                {
                    List<NursingTime> nursingTimesRange = PersonService.Instance
                        .LoadNursingTimesRange(childInfo.SelectedChildId, null, dt);

                    if (nursingTimesRange.Count > 0)
                    {
                        NursingTime newNursingTime;
                        newNursingTime = nursingTimesRange.First();
                        PersonService.Instance.DeleteNursingTime(newNursingTime);
                    }
                }
            }
        }

        list.Clear();
        Calendar1.SelectedDates.Clear();

        if (errorMessage == "")
            this.Master.ShowPopupBox(true, "Varaukset poistettu");
        else
            this.Master.ShowPopupBox(false, errorMessage);
    }
}
```

KUVA 16. DeleteDayCareTimes_OnClick-funktion toteutus

5.2.5 Hoitoaikavarausten listaaminen ja merkintä kalenterissa

Kaikki aktiivisen kuukauden hoitoaikavaraukset listataan sivun alalaitaan aikajärjestyksessä, sekä varatut päivämäärät merkitään kalenterin punaisella taustavärillä. Tämä toiminto toteutettiin kalenterikontrollin *OnDayRender*-tapahtumaan kiinnitetyssä *checkForPlacements*-funktiossa, joka suoritetaan jokaisen kalenterissa näkyvän päivämäärän osalta erikseen sivulatauksen yhteydessä. Funktiossa tarkistetaan päivän hoitoaikavaraukset ja mikäli varauksia löytyy, muutetaan päivämäärän taustaväri punaiseksi. Samalla päivämäärä sekä hoitoaikavarauksen alku- ja loppukellonajat lisätään sivulla olevaan listaan. Käyttäjälle näkyy siis kalenterissa visuaalisesti varatut hoitopäivät sekä selkeä listaus päivämääristä ja varausten kellonajoista kalenterin alapuolella (kuva 17).



KUVA 17. Kalenteriin tallennettujen hoitoaikavarausten näyttäminen

6 JATKOKEHITYSTARPEET JA -MAHDOLLISUUDET

Vaikka hoitoaikavaraustoiminto on käyttökelpoinen nykyisellä tavalla toteutettuna, on kehitysmahdollisuuksia toimintoon useita. Järjestelmän testikäyttäjiltä on saatu palautetta ja ideoita siitä, miten käytettävyyttä voitaisiin parantaa. Esimerkiksi perheissä, joissa on useampia lapsia, voisi olla hyödyllistä, että kaikille perheen lapsille voisi tallentaa hoitoaikavaraukset samanaikaisesti yhdellä klikkauksella. Tämä vähentäisi vanhempien työmäärää ja parantaisi käytettävyyttä huomattavasti. Lisäksi varaussivulle on suunnitteilla toiminto, jolla käyttäjä voi määrittää alku- ja loppupäivämäärän ajalle, jolle haluaa varauksia tallentaa. Näin ollen päiviä ei tarvitse valita kalenterista viikottain, mikä vähentää myös käyttäjältä vaadittujen klikkausten lukumäärää, eli parantaa käytettävyyttä. Kaiken kaikkiaan hoitoaikavaraustoiminto on toimiva, mutta se tulee varmasti kehittymään lopulliseen muotoonsa vasta laajemman testikäytön jälkeen. Käyttäjien palaute on toiminnon kehittämisessä erittäin tärkeää.

7 YHTEENVETO

Opinnäytetyön aiheena oli toteuttaa DaisyNet-sovellukseen hoitoaikavaraustoiminto. Toiminnon avulla käyttäjät voivat tehdä lapsille hoitoaikavarauksia päiväkoteihin, joissa Daisy-järjestelmä on käytössä. Internetissä toimiva varausjärjestelmä vähentää sekä vanhempien että päiväkodin työntekijöiden työmäärää, sillä varausten tekeminen ja muokkaaminen toimii 24 tuntia vuorokaudessa eikä paperilappujen täyttämistä ja kuljettamista päiväkotiin enää tarvita.

Hoitoaikavaraustoiminnon toteuttaminen ei osoittautunut kuviteltua vaikeammaksi, vaikka omat haasteensa projektille toikin se, että ASP.NET-framework ei ollut ennestään kovinkaan tuttu ympäristö. Toiminnosta saatiin kuitenkin rakennettua täysin toimiva ja vaatimusmäärittelyä vastaava, joten siltä osin projekti onnistui hyvin.

Käyttöliittymä hoitoaikavaraustoiminnolle oli graafikon suunnittelema ja sen toteuttaminen onnistui suhteellisen helposti, sillä HTML- ja CSS-tekniikat ovat hyvin tuttuja useamman vuoden kokemuksella. Vaikka ulkoasu muuttuikin hieman työtä tehdessä, se ei vaikeuttanut toiminnon rakentamista juurikaan. JavaScript-koodin kirjoittamista helpotti todella paljon jQuery-kirjaston käyttäminen, mikä oli ehdottomasti hyvä valinta tälle projektille.

Opinnäytetyön tekeminen opetti laajasti ASP.NET-ympäristössä työskentelyä ja Microsoft Visual Studio -ohjelman käyttöä. Työn edetessä sai perspektiiviä siihen, miten ison järjestelmän ylläpitäminen ja uusien ominaisuuksien kehittäminen hoidetaan ja miten varmistutaan siitä, että uudet ominainaisuudet eivät vaikuta vanhojen ominaisuuksien toimintaan. Testausvaiheen tärkeys nousi esille useamman kerran opinnäytetyötä tehdessä. Myös versionhallinnan käyttö tuli tutuksi ja huomasin, että on tärkeä pitää koodi ajantasalla niin versionhallinnassa kuin omalla koneellakin ongelmien välttämiseksi. Kaiken kaikkiaan projektissa konkretisoitui se, minkälaista ohjelmistokehitys voi olla tämänpäivän yrityksessä.

LÄHTEET

ASP.NET. 2013. Saatavissa: http://en.wikipedia.org/wiki/ASP.NET. Hakupäivä: 19.4.2013.

CSS. 2013. Saatavissa: http://en.wikipedia.org/wiki/Cascading_Style_Sheets. Hakupäivä: 4.4.2013.

HTML. 2013. Saatavissa: http://fi.wikipedia.org/wiki/HTML. Hakupäivä: 4.4.2013.

JavaScript. 2013. Saatavissa: http://en.wikipedia.org/wiki/JavaScript. Hakupäivä: 4.4.2013.

jQuery. 2013. Saatavissa: http://en.wikipedia.org/wiki/JQuery. Hakupäivä: 19.4.2013.

jQuery History. 2013. Saatavissa: https://jquery.org/history/. Hakupäivä: 19.4.2013.