Niko Siironen

WEB-SOVELLUKSEN KEHITTÄMINEN
Esimerkkinä EnergiaGuru-palvelu

Opinnäytetyö
CENTRIA AMMATTIKORKEAKOULU
Mediatekniikan koulutusohjelma
Lokakuu 2012

TIIVISTELMÄ OPINNÄYTETYÖSTÄ

Yksikkö Ylivieskan yksikkö	**Aika** Lokakuu 2012	**Tekijä/tekijät** Niko Siironen
Koulutusohjelma Mediatekniikka		
Työn nimi WEB-SOVELLUKSEN KEHITTÄMINEN.Esimerkkinä EnergiaGuru-palvelu		
Työn ohjaaja Hannu Puomio		**Sivumäärä** 28
Työelämäohjaaja Terho Kivimaa		

Opinnäytetyön tarkoituksena oli kehitellä eteenpäin Suomen Energianeuvonta Oy:n tilaamaa web-sovellusta, jonka avulla käyttäjä voisi tutkia, milloin ja millä perustein sähkösopimus olisi järkevintä kilpailuttaa.

Työn haasteina olivat järjestelmän kehittäminen hyödyntäen pohjalla olevaa Contao-nimistä julkaisujärjestelmää, kuvaajissa käytettävän datan koostaminen tietokannasta, sekä sen esittäminen selkeällä tavalla.

Opinnäytetyö käy läpi PHP:n historiaa, mitä PHP:n käyttäminen vaatii, sekä näyttävien kuvaajien tekemistä FusionChartsin avulla.

Asiasanat
EnergiaGuru, FusionCharts, PHP, web-ohjelmointi

Centria
UNIVERSITY OF APPLIED SCIENCES

ABSTRACT

CENTRIA UNIVERSITY OF APPLIED SCIENCES	Date October 2012	Author Niko Siironen
Degree programme Media Technology		
Name of thesis DEVELOPING A WEB APPLICATION.EnergiaGuru-service as an example		
Instructor Hannu Puomio		**Pages** 28
Supervisor Terho Kivimaa		

The purpose of this study was to develop further a web application commissioned by Finnish Energy Consulting Ltd. The application would allow the user to explore when and on what grounds it would be sensible to invite tenders for an electricity contract.

Practical aspects handled in the thesis included developing the system with the content management system called Contao, putting together the data used in the graphs using the database, as well as presenting the data in a clear way.

The thesis also included going through the history of PHP, what is required when using PHP, as well as creating impressive graphs with FusionCharts.

Key words
EnergiaGuru, FusionCharts, PHP, web-programming

ESIPUHE

Opinnäytetyön tekeminen oli omalla kohdalla erittäin raskas prosessi. Motivaatio työn tekemiselle oli kauan hukassa, eikä valmista meinannut tulla millään. Syynä ei ollut epämieluinen työ tai kannustuksen puute. Syy oli jossain syvällä päässä ja sen päihittäminen ei ollut aivan helpoin prosessi.

Haluan esittää kiitokset Terho Kivimaalle ja opinnäytetyöohjaajalleni Hannu Puomiolle pitkästäpinnaisuudesta ja kannustuksesta opinnäytetyön loppuun saattamiseksi. Lisäksi haluan kiittää läheisiäni ja ystäviäni tuesta ja kannustuksesta hetkinä, jolloin tuntui, ettei opinnäytetyö tulisi koskaan valmiiksi.

1 JOHDANTO

Olen ollut pienestä pojasta asti kiinnostunut ohjelmoinnista. Olen myös ollut kiinnostunut siitä, miten erilaisia web-palveluja oikein on toteutettu. Nämä kiinnostuksen kohteet olivat luonnollisia sysääjiä minulle tekemään opinnäytetyö aiheesta, jossa käsitellään ohjelmointia.

Keväällä 2011 haettiin koulussamme kesäksi harjoittelijoita kehittämään web-palvelua lähialueen yritykselle. Olin heti kiinnostunut asiasta, mutta pohdiskelin, olisiko minusta kehittämään työskentelemään moisessa projektissa, kun PHP-osaamiseni ei ollut oikein millään asteella. Luokkakaverini kannustamana lähdin kuitenkin projektiin mukaan. Kesän ajan tein perustaa pienasiakaspuolelle. Jatkoin kesän jälkeen hieman projektin parissa ja vuoden 2012 alkupuolella sain tietää, että projektista olisi mahdollista tehdä myös opinnäytetyö.

Opinnäytetyöni aiheena on Suomen Energianeuvonta Oy:n tilaaman web-sovelluksen kehittäminen käyttäen PHP-ohjelmointikieltä. Kehitystyössä on pyritty jalostamaan tuotteen tekninen puoli mahdollisimman valmiiksi. Tällöin tuote olisi teknisesti valmis, ainoastaan ulkoasu pitäisi tehdä ja tuote olisi valmis julkaistavaksi.

Alussa käyn läpi PHP:n historiaa, millainen kieli PHP on ja miten sillä kehitetään web-sovelluksia. Myöhemmin opinnäytetyössä käydään läpi kuvaajan piirtämistä käyttäen projektiin valitsemaani FusionChartsia, joka on samannimisen yrityksen tekemä Flash-pohjainen sovellus. Neljännessä luvussa esittelen EnergiaGuru-palvelua, sen toimintoja, sekä toimintoihin tekemääni kehitystyötä.

Tavoitteenani on, että opinnäytetyön luettuaan lukijalla on käsitys siitä, miten web-sovelluksia voi kehittää käyttäen PHP-ohjelmointikieltä ja FusionChartsia työkaluinaan.

2 WEB-OHJELMOINTI PHP:LLÄ

PHP: Hypertext Preprocessor (yleisesti PHP) on laajasti käytetty, avoimen lähdekoodin skriptikieli. Se on palvelinpuolen kieli eli PHP-koodi suoritetaan palvelimella ja koodin tulos lähetetään käyttäjälle selaimen näytettäväksi.

PHP:ta voi upottaa HTML-koodiin. Tämä tapahtuu käyttäen avaus- ja sulkutageja ”<?php” ja ”?>”. Web-palvelimen voi myös asettaa suorittamaan HTML-tiedostot PHP:ssa, jolloin loppukäyttäjällä ei ole mitään tietoa, millaista koodia olet luonut (The PHP Group 2012).

```
<html>
    <head><title>PHP-esimerkki</title></head>
    <body>
        <?php
        $viesti=”Terve maailma!”;
        echo ”$viesti”;
        ?>
    </body>
</html>
```

PHP vaatii käyttäjältään erittäin vähän. Tarvitaan ainoastaan palvelin, joka pystyy suorittamaan PHP-koodia ja jokin tekstin kirjoittamiseen tarkoitettu ohjelma. Esimerkiksi muistio riittää vallan mainiosti ja sitä käytetään nykyäänkin jonkin verran koodin kirjoittamiseen. Kuitenkin kannattaa ladata esimerkiksi Notepad++, jolla saa koodiin värikorostukset. Tämä auttaa koodin lukemista ja korjaamista.

Nykyään varmasti jokainen palvelimentarjoaja on asentanut palvelimelleen PHP-tuen, joten tämäkään ei ole ongelma. Jos haluat PHP-tuen omaan kotikoneeseesi, kenties helpoin tapa on asentaa XAMPP omalle koneellesi. Se on yksinkertainen paketti, jolla saat asennettua Apache Serverin, MySQL:n, PHP:n ja Perlin kotikoneeseesi. Tällä hetkellä XAMPP:issa on tuki Linuxille, Windowsille, Mac OS:lle ja Solarikselle. (Apache friends 2011.)

2.1 PHP:n historia

2.1.1 PHP:n synty ja ensimmäiset askeleet

PHP:n alku sijoittuu vuoteen 1995, jolloin Rasmus Lerdorf kehitti Perl/CGI-skriptin, joka seurasi hänen verkossa olevan ansiolistansa lukijamäärää. Vastaavanlaisia työkaluja ei tuolloin vielä ollut, mikä sai aikaan useita työkaluja koskevia sähköpostitiedusteluja. Lerdorf alkoi jakaa työkalujaan Personal Home Page -nimikkeellä. (Gilmore 2005, 1.)

PHP:n saama suosio sai Lerdorfin kehittämään työkalusarjaansa edelleen ja hän lisäsi siihen mm. HTML-lomakkeelle syötetyn datan konvertoinnin symbolisiksi muuttujiksi. Tämän avulla käyttäjät pystyivät viemään datan edelleen muihin järjestelmiin. Kehitystyö huipentui vuonna 1997, jolloin julkaistiin Personal Home Page – Form Interpreter (PHP-FI). Kun PHP:n suosio jatkoi kasvuaan, syntyi lisää laajennuksia ja parannuksia ohjelmoijilta ympäri maailmaa. (Gilmore 2005, 1.)

2.1.2 PHP versio 3, 4 ja 5

PHP 3.0 oli ensimmäinen versio, joka muistutti nykyistä PHP:tä. Vuonna 1997 Andi Gutman ja Zeev Suraski aloittivat kääntäjän uudelleenohjelmoimisen. He ottivat yhteyttä Lerdorfiin ja päätyivät lopulta yhdessä kehittämään itsenäistä ohjelmointikieltä. Samalla hylättiin PHP-FI –nimike ja PHP:stä tuli lyhenne uudelle nimelle, PHP: Hypertext Preprocessor. (The PHP Group 2012.)

Kun PHP 3.0 ilmestyi kesäkuussa 1998, yli 50 000 käyttäjää käyttivät PHP:tä web-sivustojensa laajentamiseen (kirja). Varmaankin suurin syy, miksi PHP:stä tuli niin suosittu, oli PHP 3.0:n helppo laajentaminen. Lisäksi PHP 3.0 tarjosi ensimmäistä kertaa tuen olio-ohjelmoinnille ja tehokkaamman ja yhdenmukaisemman syntaksin. (The PHP Group 2012.)

Talvella 1998 Gutman ja Suraski olivat aloittaneet uuden PHP-ytimen kirjoittamisen. Tarkoituksena oli luoda ydin, joka voisi käsitellä paremmin monimutkaisia sovelluksia. PHP 3.0:n ominaisuudet ja laaja tuki kolmannen osapuolen tietokannoille, sekä laaja API-

tuki mahdollistivat erittäin monimutkaisten sovellusten tekemisen, mutta sitä ei suunniteltu sellaisten sovellusten käsittelemiseen. (The PHP Group 2012.)

Ensiksi julkaistiin uudistettu Zend-niminen moottori vuoden 1999 keskivaiheilla ja toukokuussa 2000 julkaistu virallinen PHP 4.0, joka sisälsi aikaisemmin julkaistun moottorin lisäksi todella laajan kirjaston uusia ominaisuuksia. Suuren suorituskyvyn parannuksen lisäksi PHP 4.0:ssa julkaistiin mm. tuki monille uusille palvelinalustoille, HTTP-istunnot ja turvallisempia keinoja käsitellä käyttäjän syöttämää dataa. (The PHP Group 2012.)

Vuonna 2004 julkaistiin PHP 5.0 pitkällisen kehittämisprosessin ja useiden esijulkaisujen jälkeen. Siinä julkaistiin uusi versio Zend-moottorista, uusi oliopohjainen arkkitehtuuri ja tusinoittain muita ominaisuuksia. Siihen lisättiin myös esimerkiksi *try-catch*-poikkeuksien hallinta. (Gilmore 2005, 4.)

Oliopohjaisen arkkitehtuurin muutokset mahdollistivat esimerkiksi eksplisiittiset muodostin- ja tuhoajafunktiot, olioiden kloonauksen ja luokkien abstraktion (Gilmore 2005, 4). *Try-catch*-poikkeuksien hallinnan avulla voi ohjelmoija ohjata virheilmoitusten raportointia. *Try-catch* on ollut kauan osa useimpia kieliä, kuten C/C++ ja Java.

Uusimmat PHP:n vakaat versiot ovat 5.4.7 ja 5.3.17. Nämä versiot ovat korjanneet yli 20 aikaisemmissa versioissa ilmennyttä vikaa ja ne on julkaistu 13.9.2012. (The PHP Group 2012.)

2.2 Miksi valittiin PHP?

Jokaisella ohjelmoijalla on omat syynsä PHP:n käyttämiselle. On kuitenkin muutamia peruspiirteitä, miksi useat valitsevat PHP:n kielekseen kehittäessään web-sovellusta.

Ensinnäkin, PHP on helppo kieli aloittelijalle. Se ei vaadi muuttujien tyypittämistä, vaan kieli osaa itse arvata, mitä tyyppiä mikäkin muuttuja on. Muuttujia ei niin ikään tarvitse esitellä muutenkaan etukäteen, vaan PHP luo muuttujat lennosta sitä mukaa, kun niitä skriptissä kutsutaan. Lisäksi taulujen kokoa ei tarvitse tietää, eikä osoittimia tarvitse hallita,

kuten esimerkiksi C/C++:ssa. PHP ei myöskään vaadi toimiakseen mitään ulkoisia kirjastoja, jotka ovat tuttuja useista muista kielistä.

PHP tarjoaa myös todella monenlaisia mahdollisuuksia. Se sisältää useita eri tietokantatukia, joista voi valita mieluisensa. Sillä voi tehdä dynaamista sisältöä, käsitellä kaavakkeiden tietoja, luoda useita eri tiedostomuotoja ja niin edelleen. Lisäksi PHP tarjoaa kattavan tuen niin funktionaaliselle ohjelmoinnille kuin olioperäiselle ohjelmoinnille.

PHP on myös ilmainen. Se ei sisällä minkäänlaisia käyttö-, muokkaus- tai jakelurajoituksia. PHP:tä voi käyttää ilman minkäänlaisia lisenssirajoituksia, joita kaupallisiin tuotteisiin usein sisältyy. (Gilmore 2005, 7.) Sille löytyy myös hyvä tuki niin kaupallisista, kuin ilmaisistakin sovelluksista, mikä helpottaa PHP:n käyttöönottoa ja opettelemista. PHP:tä voi myös laajentaa erilaisilla moduuleilla ja laajennoksilla, jotka ovat useimmiten ilmaisia (Gilmore 2005).

PHP on myös nopea kieli. Skriptin suorittaminen on nopeaa, eikä PHP vaadi laitteistolta paljoa resursseja, joten sitä on helppo suorittaa myös vanhoissa laitteissa. Varsinkin Unix/Linux-ympäristöissä PHP toimii varsin moitteettomasti. (2kmediat.com 2012.)

3 GRAFIIKANPIIRTO-OHJELMA FUSIONCHARTS

FusionCharts on samannimisen yrityksen tuottama paketti, joka sisältää useiden erilaisten kuvaajien piirtämisen joko Flashilla tai JavaScriptillä. Yritys perustettiin vuonna 2002 Pallav Nadhanin toimesta hänen ollessa yliopistossa. Tuohon aikaan kuvaajat olivat tylsiä ja FusionCharts lähti kehittämään Flash-kuvaajia, jotka olisivat animoituja ja näyttäviä. (FusionCharts 2012.)

FusionChartsilla on yli 20000 asiakasta yli 100:ssa eri maassa. Esimerkiksi Google ja Facebook käyttävät datan esittämiseen FusionChartsia. FusionCharts on siis erittäin suosittu ja suurien yritysten käyttämä. (FusionCharts 2012.)

3.1 FusionCharts Free ja FusionCharts XT

FusionChartsin laitteistovaatimukset ovat erittäin pienet. JavaScript-versiota käytettäessä riittää, että selaimesta on JavaScript sallittuna ja tuettuja selaimia ovat Internet Explorer 6 ja uudemmat, Firefox 2 ja uudemmat, Safari 5.0 ja 5.1 (sisältää myös Safari for iOS devices), Opera 10 ja 11, sekä Chrome. Jos puolestaan käyttää Flash-versiota, tarvitsee Adobe Flash Playerin version 8 tai uudemman. FusionChartsilla on siis erittäin monipuolinen tuki, mikä lisää käyttöönoton mielekkyyttä. (FusionCharts 2012.)

FusionChartsista on olemassa kaksi versiota: FusionCharts Free ja FusionCharts XT. FusionCharts Freessä on ”vain” 22 erilaista kuvaajatyyppiä. Freestä on riisuttu pois joitakin muokkausmahdollisuuksia, esim. animaatioita on rajoitettu. Freestä on ainoastaan Flash-versio saatavilla. (FusionCharts 2012.) Free hyväksyy tiedon ainoastaan XML-muodossa. (FusionCharts 2012.) Free siis sopii peruskuvaajien tekemiseen, mutta melko äkkiä tulee eteen tilanne, jossa tarvitsee FusionChartsin XT-versiota.

FusionCharts XT:stä löytyy Flash-version lisäksi myös JavaScript-versio. JavaScript-versio on sikäli hyödyllinen, että esimerkiksi iPhone-laitteet eivät tue Flashia, joten kuvaajat saadaan näissä laitteissa näkymään JavaScriptin avulla. Eri kuvaajatyyppejä on XT:ssä kymmeniä ja ne ovat todella hyvin muokattavissa (FusionCharts 2012.) Voit tehdä

näyttävämpiä animaatioita, antaa datan XML:n lisäksi myös JSON-tiedostona ja vaikkapa tallentaa kuvaajat JPEG-, PNG- tai PDF-tiedostoksi ja datan CSV-muotoon. (FusionCharts 2012.)

3.2 Datatiedoston rakenne

FusionChartsin datatiedoston rakenne XML-tiedoston tapauksessa on melko yksinkertainen. Monen kuvion kuvaajan tapauksessa datatiedoston rakenne on kutakuinkin seuraavanlainen (mukaelma FusionCharts 2012):

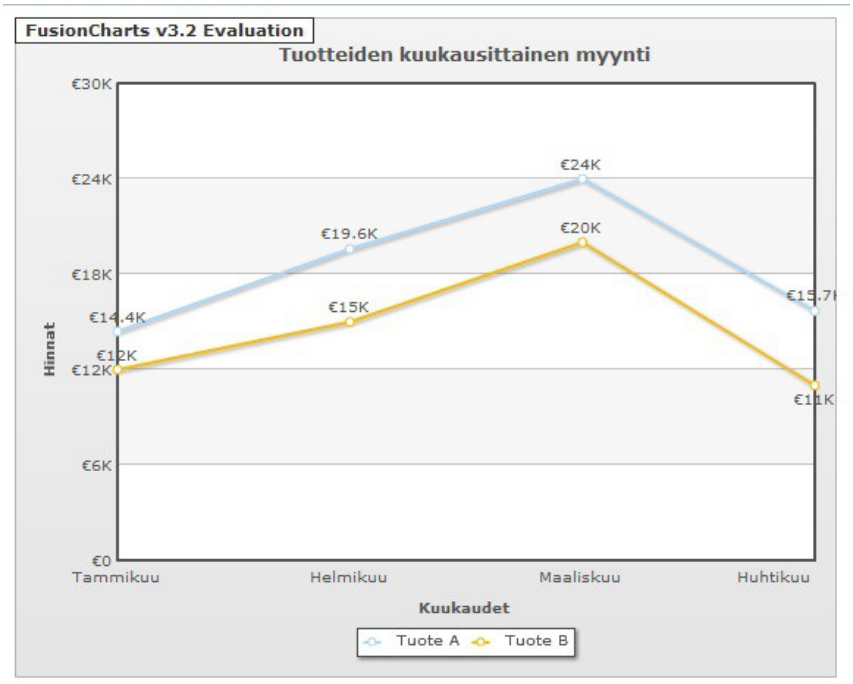
```
<chart caption='Tuotteiden kuukausittainen myynti'
xAxisName='Kuukaudet' yAxisName='Hinnat' numberPrefix='€'>

   <categories>
      <category Label="Tammikuu"/>
      <category Label="Helmikuu"/>
      <category Label="Maaliskuu"/>
      <category Label="Huhtikuu"/>
   </categories>

   <dataset seriesName="Tuote A">
       <set  value='14400' />
       <set  value='19600' />
       <set  value='24000' />
       <set  value='15700' />
   </dataset>
   <dataset seriesName="Tuote B">
       <set  value='12000' />
       <set  value='15000' />
       <set  value='20000' />
       <set  value='11000' />
   </dataset>
</chart>
```

`<chart>`-tagi aloittaa koko datatiedoston. Siinä kerrotaan caption-attribuutilla kuvaajan otsikko, xAxisName-atribuutilla x-akselin otsikon nimi ja yAxisName-attribuutilla y-akselin otsikko. numberPrefix-attribuutti on valinnainen, jolla voidaan määrittää

FusionCharts asettamaan jokaisen y-akselin arvon eteen attribuutissa määritetty merkkijono. Tämä attribuutti on hyödyllinen tilanteessa, jossa kuvaajassa esiintyy esimerkiksi valuuttoja. Valuuttamerkkiä ei voi sijoittaa arvojen syöttöön, sillä arvojen pitää olla paljaita numeroita.

`<categories>`-tagi kertoo FusionChartsille kuvaajan kategoriat. Esimerkiksi, jos halutaan esittää kuukausittainen tuotemyynti, tämän tagin sisälle syötetään `<category>`-tagin Label-attribuutille arvoksi kuukausien nimet. Nämä arvot näkyvät kuvaajan x-akselilla.

`<dataset>`-tagi tarkoittaa eri kuvaajia ja sen seriesName-attribuutti kertoo esimerkiksi pylvään tai viivan otsikoinnin, esimerkkitapauksessa tuotteiden nimet. `<set>`-tagin value-attribuutilla kerrotaan arvo. `<set>`-tageja tulee olla yhtä monta, kuin `<category>`-tageja ja ne tulevat kuvaajaan annetussa järjestyksessä. Näin ollen esimerkissä tammikuun kohdalle tulisi arvo 14400.



KUVIO 1. Esimerkkikoodin mukainen kuvaaja

Perustasolla xml-tiedoston rakenne on siis varsin yksinkertainen. Rakenteesta voidaan myös tehdä monimutkaisempi erilaisten trendiviivojen jne. avulla, mutta niitä ei käsitellä tässä opinnäytetyössä.

4 ENERGIAGURU

EnergiaGuru-palvelun ideana on, että asiakas voi seurata oman sähkösopimuksensa hintakehitystä suhteessa muihin sähkösopimuksiin. Palvelussa on kaksi käyttäjäryhmää: pienasiakas eli kotitaloudet, sekä yritykset. Molemmille ryhmille on sekä ilmaista, että maksullista sisältöä.

Pienasiakaspuolen palveluita olivat tätä opinnäytetyötä kirjoittaessa ilmainen sähkösääennuste, sekä maksullinen sähkönhintaennuste. Yrityspuolen palveluita ovat ilmainen sähkösääennuste, sekä maksullinen sähkönsuojausennuste. Esittelen näitä palveluita seuraavissa luvuissa tarkemmin, sekä niihin tekemäni kehitystyön. Koska koodi on salaista, esittelen mekaniikan ja annan esimerkkejä ratkaisumahdollisuuksista.

4.1 Pienasiakaspuoli

4.1.1 Sähkösääennuste

Pienasiakaspuolella oli alkujaan kolme palvelua: sähkönhintaennuste, sähkösääennuste ja sähkön hintahistoria. Näistä sähkösääennuste oli ilmainen ja sähkönhintaennuste ja sähkön hintahistoria olivat maksullisia.

Sähkön hintahistoria ulottui kuuden vuoden taakse ja esitteli korosteiden avulla asiakkaalle, mitkä tekijät ovat vaikuttaneet hintojen kehittymiseen kuvaajan osoittamalla tavalla. Kuvaajassa oli merkittävissä ankkuripisteissä kehystetty viisikulmio, joka osoitti, että kohdassa oli tapahtunut jokin merkittävä asia. Viemällä hiiren korostettuun ankkuripisteeseen, käyttäjä näki selitteen, joka kertoi hintakehityksen kannalta merkittävistä tapahtumista.

Sähkösääennusteessa oli tuolloin hintahistoria kuuden kuukauden ajalta ja sen lisäksi ennuste siitä, mihin suuntaan eri tuoteryhmien hinnat tulisivat kehittymään ja mitkä olisivat hintakehitykseen vaikuttavat tekijät. Lisäksi kuvaajan yhteydessä oli yleisesti markkinatilannetta kuvaava kommentti, jossa annettiin seikkaperäisempi arvio siitä, mihin suuntaan hinnat voisivat kehittyä tulevaisuudessa.

Asiakas päätyi siihen, että sähkösääennuste ja sähkön hintahistoria tulisi yhdistää. Vaatimuksena oli, että asiakas näkisi hintakehityksen kolmen vuoden ajalta ja hintojen kehityksessä muut tuotteet (toistaiseksi voimassaolevat tarjoussopimukset, toimitusvelvolliset sopimukset, yksi- ja kaksivuotiset tarjoussopimukset) näkyisivät tarkalla kehityksellä, mutta pörssihinnasta otettaisiin kuukausikohtainen keskiarvo. Tämä siksi, että pörssihinnat heilahtelevat toisinaan erittäinkin voimakkaasti ja kuvaajaan tulisi turhaa heilahtelua ja näin ollen haittaisi merkittävästi kuvaajan lukemista. Keskiarvot vähentäisivät tätä heilahtelua ja kuvaajan luettavuus paranisi.

Kun aloin käsitellä hintatietoja, alussa oli mietittävä, miten voisin varmistua siitä, että tietyn kuukauden hinnat varmasti tulisivat valittua. Tämä olisi helpoin ratkaista käyttämällä MySQL:ssä BETWEEN-rajausta päivämäärälle. Tällöin pitäisi määritellä jokaisen kuukauden ensimmäinen ja viimeinen päivä. Ensimmäisen päivän määrittäminen on yksinkertaista PHP:n *date()*-funktion avulla. Ensimmäisenä parametrina *date()*-funktio ottaa päivämäärän esitysmuodon ja toisena parametrina esitysmuotoon muunnettavan aikaleiman. Sitä voi käyttää esimerkiksi näin:

```
$ensimmainenPaiva=date('Y-m-01', strtotime('-36 months'));
```

Esimerkissä päivämäärän esitystavaksi asetetaan ”YYYY-MM-DD”. Tämä siksi, että tietokannan DATE-kenttä on sitä muotoa. *strtotime()*-funktio ottaa parametrina merkkijonona ajan, sekä käyttäjän valinnan mukaan aikaleiman, jota käytetään suhteellisen aikaleiman laskemisessa. Merkkijono voi olla esimerkiksi ”-36 months”, ”now” tai ”+3 years”. Eli, jos annat *strtotime()*-funktioon parametreiksi ”-4 years” ja aikaleiman, joka osoittaa neljän vuoden taakse, luo strtotime() aikaleiman, joka osoittaa kahdeksan vuoden taakse. Tämäkin voi olla kätevää joissakin tapauksissa.

BETWEEN-rajaus vaatii myös viimeisen päivän. Koska *strtotime()*-funktiolla ei voi määritellä kuukauden viimeistä päivää, viimeinen päivä täytyy määritellä toisella tapaa. Se onnistuu esimerkiksi seuraavalla tavalla:

```
$viimeinenPaiva=date('Y-m-d', mktime(0,0,0, date('m')-35,
date('01')-1, date('Y')));
```

mktime()-funktio ottaa parametreina tunnit, minuutit, sekunnit, kuukauden, päivän ja vuoden. Kuukauden viimeisen päivän määrittelyn tapauksessa tunnit, minuutit ja sekunnit eivät ole tarpeellisia, joten ne voi asettaa nollaksi. Vuodeksi kelpaa nykyinen vuosi, mutta kuukautta pitää siirtää 35 kuukautta taaksepäin. Tällöin saadaan kuukausi, joka on aikaisemmin asetettua $ensimmainenPaiva-muuttujaa seuraava kuukausi. Kun päivämääräksi asetetaan ensimmäinen päivä ja siirrytään sitä edeltäneeseen päivään, saadaan tulokseksi kuukauden viimeinen päivä. Näin saa $viimeinenPaiva arvon ”2009-10-31”.

Nyt on mahdollista rakentaa MySQL-kysely, joka hakee hinnat tiettyjen päivämäärien väliltä. Yleensä suoritettava SQL-lause rakennetaan ensiksi merkkijonoksi ja suoritetaan sitten käyttäen rakennettua merkkijonoa.

```
            $sql=”SELECT hinta, paivamaara FROM hinnat WHERE paivamaara
BETWEEN ’ ”.$ensimmainenPaiva.” ’ AND ’  ”.$viimeinanPaiva.” ’ ORDER BY
paivamaara”;
            $result=mysql_query($sql);
```

Näin saadaan valittua päivämäärät ja hinnat oikealta aikaväliltä. Lopussa oleva ORDER BY varmistaa, että päivämäärät ja hinnat tulevat oikeaan järjestykseen. Voihan nimittäin olla, että tietokantaan on syötetty hinta esimerkiksi päivämäärälle 3.3.2010 ennen kuin se on syötetty päivämäärälle 2.3.2010. Jos päivämääriä ei järjestellä oikeaan järjestykseen, päivämäärät ja hinnat menisivät väärään järjestykseen ja kuvaajassa näkyisi 3.3.2010 ja sitten vasta 2.3.2010.

Näin saatiin ratkaistua, miten ensimmäisenä kuvaajaan tulevat hintatiedot haetaan. Seuraavana ongelmana oli, että miten tämä kaikki saataisiin automatisoitua, jotta turhalta toistolta vältyttäisiin. Tähän tarkoitukseen soveltuu erinomaisesti for-silmukka.

```
            for($i=-36; $i<=0; $i++)
            {
                $j=$i+1;
                $ensimmainenPaiva=date(’Y-m-01’, strtotime(’’.$i.’
months’));
```

```
            $viimeinenPaiva=date('Y-m-d', mktime(0, 0, 0, date('m')+$j,
date('01'-1, date('Y')));
            ...
        }
```

$i-muuttujaa käytetään ensimmäisen päivämäärän selvittämiseen kunkin kuun kohdalla. Jotta voitaisiin selvittää kuukauden viimeinen päivä, käytetään apuna $j-muuttujaa, joka on yhden suurempi kuin $i. For-silmukassa otetaan talteen tuotteiden hinnat, päivämäärät, sekä selitetekstit.

Seuraavaksi täytyy selvittä, mitkä ovat hinnoissa esiintyvät pienin ja suurin arvo. Tätä tietoa tarvitaan, jotta kuvaajan ylä- ja alaraja voitaisiin myöhemmin asettaa kuvaajaan oikein. Vertailun tekeminen on helppoa foreach-silmukalla.

```
foreach($hinnat as $hinta)
{
    if($hinta<$minimi)
    {
            $minimi=$hinta;
    }
}
foreach($hinnat as $hinta)
{
    if($hinta>$maksimi)
    {
            $maksimi=$hinta;
    }
}
```

Kun kaikki tarvittava data on kerätty, on aika alkaa luoda xml-tiedostoa. Sitä varten pitää avata xml-tiedosto kirjoitustilaan.

```
$tiedosto="tiedostopolku/Datatiedosto.xml";
$fh = fopen($tiedosto, 'w') or die("can't open file");
```

Tiedostopolku on syytä laittaa muuttujaan, sillä samaa tiedostopolkua tarvitaan myöhemmin tiedoston sulkemiseen. Kun tiedostopolku on muuttujassa, polun ja/tai tiedostonimen muuttuessa ei tarvitse muuttaa kuin muuttujan tietoa, eikä *fopen()*- ja

fclose()-funktioiden parametreja. *die()*-funktio yksinkertaisesti lopettaa koodin suorittamisen, jos tiedostoa ei syystä tai toisesta voitu avata.

Seuraavaksi luodaan itse xml-tiedosto koodin avulla. Olen päätynyt ratkaisuun, jossa kukin osa-alue tallennetaan ensin merkkijonoon ja sen jälkeen kirjoitetaan tiedostoon. Eli ensin koko tiedoston aloittava `<graph>`-aloitustagi attribuutteineen sijoitetaan merkkijonomuuttujaan ja sen jälkeen kirjoitettaan tiedostoon, jonka jälkeen sijoitetaan `<categories>`-tagi, sen sisällä olevat `<category>`-tagit ja `</categories>`-lopetustagi ja kirjoitetaan ne tiedostoon. Tätä jatketaan myös `<dataset>`-tagien kanssa ja lopussa tiedoston lopettava `</graph>`-lopetustagi. `<graph>`-tagi on esimerkistä poiketen tässä sen vuoksi, että kaikkia tarvittuja asetuksia ei voi asettaa `<chart>`-tagiin, vaan niitä varten tarvitaan `<graph>`-tagi.

```
$stringData = "<graph caption=\"Hinnat\" subcaption=\"\" RotateNames=\”1\”
hovercapbg=\"FFECAA\" hovercapborder=\"F47E00\" formatNumberScale=\"0\"
decimalPrecision=\"2\" showValues=\"0\" showLimits=\"0\" adjustDiv=\"1\"
showDivLineValue=\"1\" yAxisMinValue=\"".$minimi."\" yAxisMaxValue=\"".$maksimi."\"
yAxisName=\"snt/kWh\" showLegend=\"0\" >\n";

fwrite($fh, $stringData);
```

Koodissa on pari kohtaa, jotka vaativat selventämistä: decimalPrecision-attribuutille annetaan arvoksi xml-tiedostossa olevien arvojen pyöristystarkkuus ts. desimaalien määrää. showValues-attribuutilla määritellään, näytetäänkö ankkuripisteiden arvoja vai ei. Jos arvo on nolla, arvoja ei näytetä. showLimits-attribuutilla voidaan piilottaa kuvaajan pienin ja suurin arvo. adjustDiv-attribuutilla määrätään, jakaako FusionCharts arvoasteikon automaattisesti sen mielestä sopiviin arvoväleihin. Vaihtoehtoisesti voit määrittää itse väliarvojen määrän antamalla adjustDiv-attribuutille arvoksi nolla ja käyttämällä numdivlines-attribuuttia, jonka arvo on jakoviivojen määrä. Viimeisenä käsiteltävänä attribuuttina haluan nostaa showLegend-attribuutin. Sen avulla määritellään, näytetäänkö kuvaajan alla selitteitä vai ei. Koska kuvaajan viereen tulee tässä tapauksessa ennuste, jossa tuotteiden nimet ovat viivojen värisiä, ei selitettä ole syytä näyttää.

Seuraavaksi asetetaan <categories>-tagi ja sen sisällä olevat <category>-tagit. Tässä tyydyttiin sellaiseen ratkaisuun, että näytettäisiin ainoastaan ensimmäinen ja viimeinen päivämäärä.

```
            //aloitetaan categories-tagi, antaa x-akselin otsikot
            $stringData="<categories>\n";

            $koko=sizeof($paivat);
            $viimeinen=$koko-1;
            for($i=0;$i<$koko;$i++)
            {


                if($i==0 || $i==$viimeinen)
                {
                    $aikaleima=strtotime($paivat[$i]);
                    $pv=date('j.n Y', $timestamp);
                    $stringData=$stringData."<category name=\"".$pv."\"
/>\n";
                }
                else
                {
                        $stringData=$stringData."<category name=\"\" />\n";
                }
            }
            //suljetaan categories-tagi
            $stringData=$stringData."</categories>\n";

            fwrite($fh, $stringData);
```

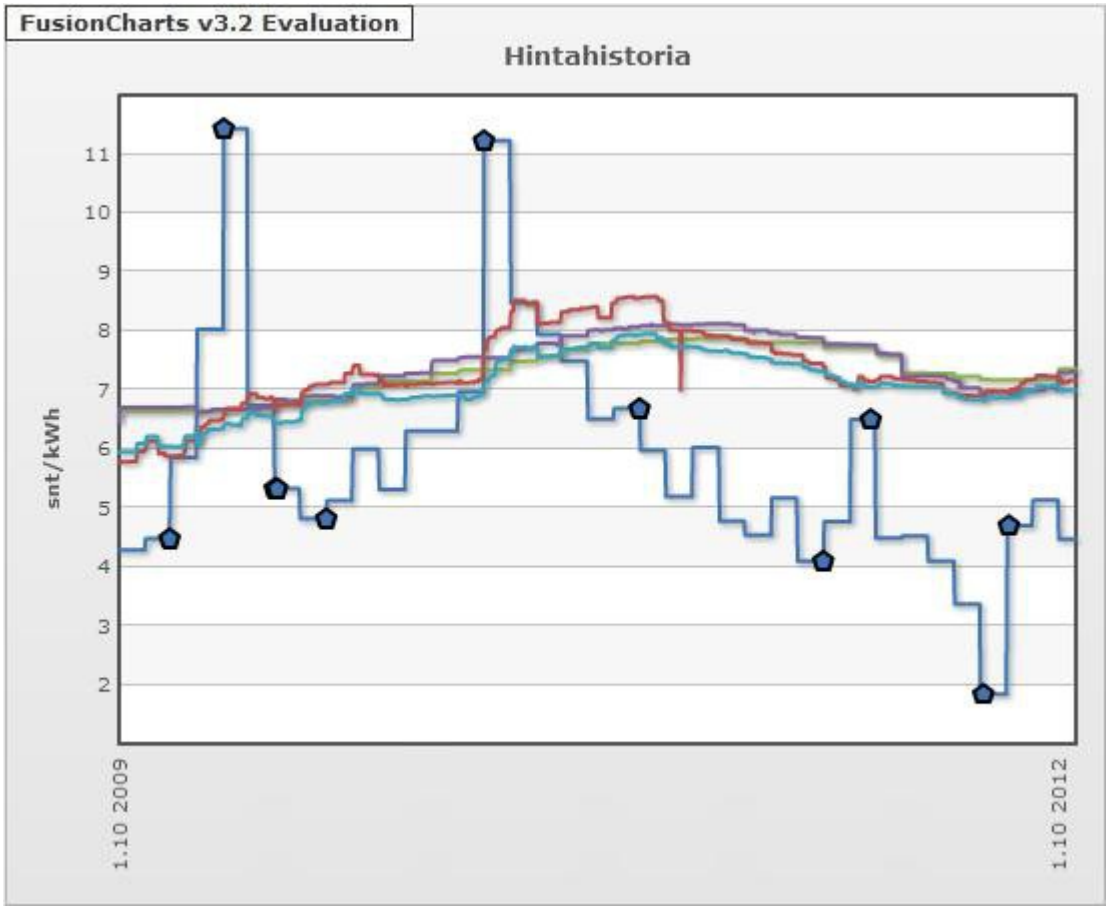
Näin saadaan asetettua päivämäärä ainoastaan ensimmäiseen ja viimeiseen <category>-tagiin ja muut jäävät tyhjäksi. Kuitenkin päivämäärät otetaan kaikki talteen taulukkoon, jotta jatkon kannalta, jos halutaan esimerkiksi tietyin väliajoin päivämäärät, saataisiin ne esiin muuttamalla ainoastaan for-silmukkaa. Lisäksi näin varmistutaan siitä, että <category>-tageja tulee yhtä monta kuin on hintatietoja haettuna tietokannasta.

Tämän jälkeen siirrytään hintatietojen kirjoittamiseen. Esimerkissä on käytetty nordpoolin hintojen kirjoittamista, koska se poikkeaa jonkin verran muista tuotteista erityistapauksissa tulevien ankkuripisteidensä vuoksi.

```
$j=0;
$stringData = "<dataset seriesName=\"Nordpool\" color=\"3F76B7
\">\n";
foreach($nordpool as $pv)
{
    if($korostus[$j]==1)
    {

            $merkkijono="<set value=\"".$pv."\"
anchorBorderThickness=\"2\" anchorBorderColor=\"000000\"
anchorRadius=\"5\" anchorSides=\"5\" anchorBgcolor=\"3F76B7\"
toolText=\"".utf8_decode($kommentti[$j])."\" />\n";
    }
    else
    {
            $merkkijono="<set value=\"".$pv."\"
anchorAlpha=\"0\" />\n";
    }
    $j+=1;
    $stringData=$stringData.$merkkijono;
}
$stringData = $stringData."</dataset>\n";
fwrite($fh, $stringData);
```

<dataset>-tagiin tulee viivan nimi ja sen väri. <set>-tagin attribuutit määritellään sen mukaan, tuleeko ankkuripisteeseen korostus vai ei. $korostus-taulukossa on tallessa tieto siitä, mihin kohtaan korostus tulee laittaa. Tämän avulla vältytään merkkijonovertailuilta. Jos ankkuripisteeseen tulee korostus, asetetaan <set>-tagissa arvo, ankkuripisteen reunaviivan paksuus pikseleinä, ankkuripisteen reunaviivan väri, ankkuripisteen säde pikseleinä, kuinka monta sivua ankkuripisteessä on, ankkuripisteen taustaväri, sekä ohjeteksti. Muutoin asetetaan arvo ja piilotetaan ankkuripiste. Koska FusionChartsilla on ongelmia utf8-koodatun tekstin näyttämisen kanssa, täytyy koodaus purkaa ennen arvon asettamista.

Muiden tuotteiden hintojen asettaminen on huomattavasti yksinkertaisempaa, sillä niihin ei korostuksia tule. Näin ollen muille tuotteille tarvitaan ainoastaan nordpoolin tapauksessa else-käsittelijän sisällä oleva koodi. Kun kaikki tuotteet on käyty läpi, kirjoitetaan vielä lopetustagi `</graph>` ja suljetaan tiedosto käyttämällä *fclose()*-funktiota, jonka parametrina käytetään tiedostonimeä.



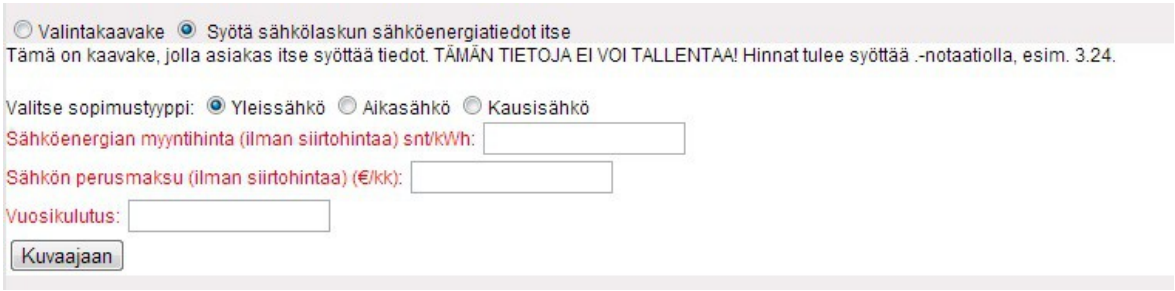
KUVIO 2. Sähkösääennuste

4.1.2 Sähkönhintaennuste

Sähkönhintaennusteessa asiakkaalla oli alkujaan mahdollisuus syöttää oman sähkösopimuksensa tiedot valintakaavaketta käyttämällä. Tämän toiminnon lisäksi tilaaja halusi, että asiakkaalla olisi myös mahdollisuus syöttää omat sähkösopimuksen tietonsa käsin. Tämä on tarpeen esimerkiksi siinä tapauksessa, ettei asiakkaan sähkösopimuksessa käytettyä myyntituotetta olisi enää tarjolla. Tein tämän siten, että lisäsin ylös kaksi radiopainiketta, joilla kontrolloidaan javascriptin avulla, kumpi kaavake näytetään.

```
function naytakaavake()
{

            if(document.getElementById("valinta").checked==true)
            {
                document.getElementById("page").style.display="block";
                document.getElementById("altpage").style.display="none";
            }

            else if(document.getElementById("syotto").checked==true)
            {
                document.getElementById("page").style.display="none";
                document.getElementById("altpage").style.display="block";
            }
}
```

Sivulla on kaksi eri -tagia, joille on annettu nimeksi ”page” ja ”altpage”. Pagen sisälle ladataan alkujaan ollut valintakaavake ja altpagen sisälle kaavake, johon voi itse syöttää sähkösopimuksensa tiedot. Javascriptissä yksinkertaisesti vain muutetaan -tagien näkyvyyttä sen mukaan, kumpi toiminto halutaan suorittaa.



KUVIO 3. Sopimustietojen syöttökaavake

Tämän lisäksi tilaaja esitti toiveen, että valintakaavakkeen tiedot olisi mahdollista tallentaa. Näin ei tarvitsisi jokainen kerta erikseen käydä valintakaavaketta läpi, vaan käyttäjä voisi jatkaa suoraan hintaennusteeseen. Tämä on toteutettu yksinkertaisesti siten, että PHP-tiedostossa kirjoitetaan MySQL:n UPDATE-komennolla käyttäjän tiedot tietokantaan. Tämän jälkeen painiketta klikkaamalla käyttäjä pääsee jatkamaan hintaennusteeseen.

Myös hintaennusteeseen tilaajalla oli joitakin toiveita. Kun tuotteita listataan, tulisi oman tuotteen kohdalla olla vertailu oman tyyppinsä tuotteiden keskiarvoiseen hintaan. Jos oma tuote olisi tuotetyyppinsä keskiarvoa halvempi, oman tuotteen suosituksissa pitäisi näkyä ”+”-merkki ja ilmoitus ”Keskivertoa halvempi, kannattava”. Muuten näytettäisiin ”-”-

merkki ja ilmoitus ”Keskivertoa kalliimpi, ei kannattava”. Lisäksi kaikkien suositeltujen tuotetyyppien nimet tulisi lihavoida, jotta ne erottuisivat selkeämmin kannattamattomista tuotetyypeistä.

Aluksi järjestelmä laatii listan kalleimmasta tuotteesta halvimpaan. Sitten järjestelmä hakee tiedon, mitkä tuotteet ovat kannattavia ja mitkä eivät, sekä tuotteiden kannattavuutta selittävän kommentin. Tämän jälkeen järjestelmä käy läpi tuotteet yksi kerrallaan.

```
if($tuotteet[$i]== "nordpool")
{
    if ($sopimustyyppi==”nordpool”)
    {
            $halvempi=1;
    }
    if($nordpoolkannattavuus==1)
    {
            echo  "<font  color=\"#".$npcolor."\"><b>Pörssihinta
(".$nphinta."€)</b></font>";
    }

    else
    {
            echo "<font color=\"#".$npcolor."\">Pörssihinta (".
$nphinta."€)</font>";
    }
}
```

Esimerkissä käydään läpi pörssisidonnaisten tuotteiden keskiarvo. $npcolor on erillisessä PHP-tiedostossa määritelty värikoodi ja $nphinta on pörssituotteille laskettu hinta €/vuosi. Siinä tapauksessa, että käyttäjän tuote on myös pörssisidonnainen, asetetaan $halvempi-muuttujan arvoksi 1. Tämä siksi, että kun käsitellään käyttäjän omaa tuotetta, tutkitaan, onko $halvempi-muuttujan arvo 1. Jos on, käyttäjän oma tuote lihavoidaan ja suosituksiin laitetaan ”+”-merkki sekä kommentti kannattavasta tuotteesta. Kuitenkaan muuttujan arvon asettamisesta ei ole haittaa, jos käyttäjän oma tuote on käsitelty jo.

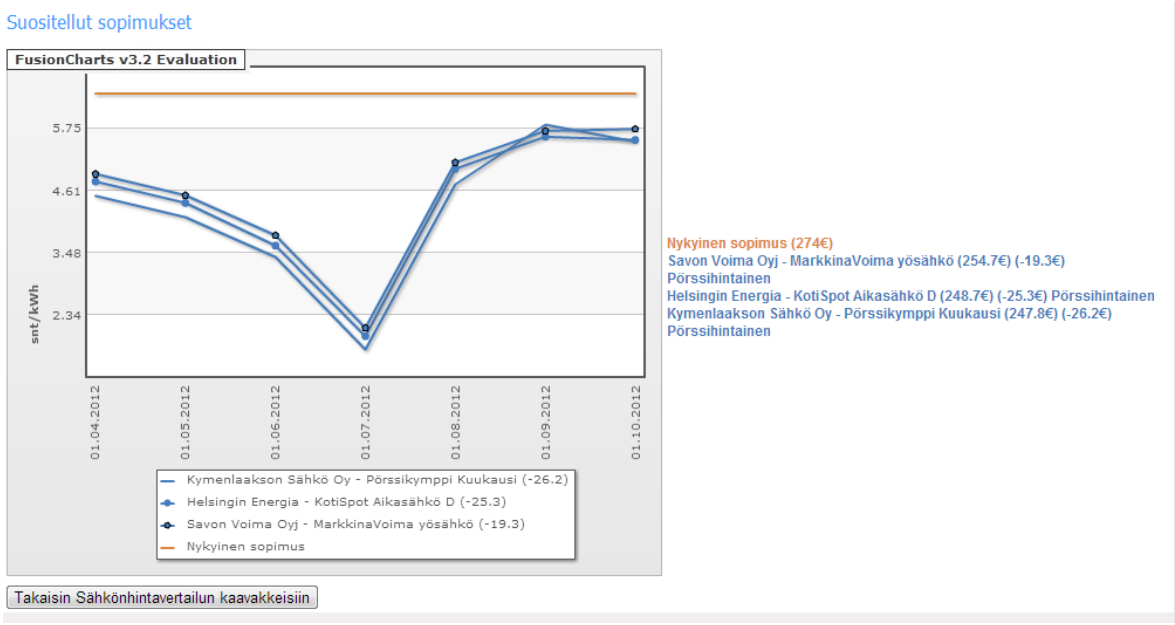
Tilaaja tahtoi myös, että hintaennusteen alapuolelle tulisi kuvaaja suositeltujen tuotetyyppien kolme halvinta tuotetta vertailtavaksi käyttäjän oman tuotteen kanssa. Näin

käyttäjä pääsisi näkemään, mitä tuotteita hänen kannattaisi harkita ja miten kunkin tuotetyypin halvimpien tuotteiden vuosihinta eroaa käyttäjän omasta tuotteesta.

```
if ($t1vkan==1)
{
    $query="SELECT yhtio, tuote FROM hinnasto WHERE
tuotetyyppi=”tarjous1v” ORDER BY hinta LIMIT 3";
    $result = mysql_query($query);
    while($row=mysql_fetch_assoc($result))
    {
            $t1yritys[]=$row['yhtio'];
            $t1tarif[]=$row['tariffi'];
    }
}
```

Tämänkaltaisen kyselyn avulla haetaan näppärästi kolme halvinta tuotetta kustakin tuoteryhmästä. Koodia on toki oikeaan versioon muokattu rajusti, jottei kantarakenne paljastuisi. Vastaavanlainen kysely tehtiin myös muille tuotteille. Koodi tarkistaa, että ovatko esimerkiksi yksivuotiset tarjoustuotteet tällä hetkellä kannattavia. Jos ovat, haetaan kannasta tieto halvimmista tuotteista, sekä yhtiöt, jotka kyseiset tuotteet tarjoavat.

Haettujen tietojen pohjalta haetaan tuotteiden hintatiedot ja luodaan XML-tiedosto. Oikeastaan ainoa poikkeus on se, että hakuehtoina käytetään tuotetyypin sijaan yhtiöitä ja tariffeja. Koska yhdessä tuoteryhmässä on aina kolme tuotetta, päädyin ratkaisemaan tilanteen siten, että tuoteryhmän tuotteiden hintakehitys piirretään kaikki samalla värillä, mutta halvimmassa tuotteessa ei näytetä ankkuripisteitä, keskimmäisessä näytetään ilman reunaviivaa ja kalleimmassa tuotteessa ankkuripisteillä on myös reunaviiva. Näin jälkeenpäin ajateltuna kyseinen ratkaisumalli tuntuu typerältä, sillä viiva, jonka ankkuripisteillä on reunaviiva, nousee kaikista näkyvimmäksi ja ilman ankkuripisteitä oleva viiva jää kaikista helpoiten piiloon. Tämän vuoksi olisi ehkä syytä muuttaa viivojen ulkoasua toisenlaiseksi.



KUVIO 4. Suositellut sopimukset

4.2 Yrityspuoli

4.2.1 Sähkösuojausennuste

Alkujaan asiakkaan täyttämässä kaavakkeessa tuotteet oli kirjattu käsin, mikä on varsin huono ratkaisu, kun ajattelee, että joka kuukausi listasta jää joitakin tuotteita pois ja uusia tulee tilalle. Jonkinlaisen automaattisen rakenteen rakentaminen oli siis erittäin tärkeää. Tilaajan mukaan listassa tulisi olla kuusi kuukausituotetta, 12 vuosineljännestuotetta, sekä seitsemän vuosituotetta ja jokaisesta listan tuotteesta myös versio, jossa on lisättynä aluehinta.

```
for($i=1;$i<7;$i++)
{
    $kk=date('n', strtotime('+'.$i.' months'));
    $kkvuosi=date('y', strtotime('+'.$i.' months'));
    if($kk==1)
    {
            $kknimi="JAN";
    }
    if($kk==2)
    {
```

```
            $kknimi="FEB";
    }
    .
    .
    .

    $arvo=100+$i;
    $tuote[$i]="ENOM".$kknimi."-".$kkvuosi;
    echo "<option value=\"".$arvo."\">".$tuote[$i]."</option>";
}

for($i=1;$i<13;$i++)
{
    //koska kvartaali käsittää 3kk (esim. Tammi-Maalis on
kvartaali 1), täytyy aikaa siirtää 3kk eteenpäin jokaisen
kvartaalin kohdalla

    $siirto=$i*3;
    $qkk=date('n', strtotime('+'.$siirto.' months'));
    $qvuosi=date('y', strtotime('+'.$siirto.' months'));

    if($qkk<=3)
    {
            $kvartaali=1;
    }
    elseif($qkk>3 && $qkk<7)
    {
            $kvartaali=2;
    }
    elseif($qkk>6 && $qkk<10)
    {
            $kvartaali=3;
    }
    elseif($qkk>9)
    {
            $kvartaali=4;
    }
    $qtuote[$i]="ENOQ".$kvartaali."-".$qvuosi;
    $arvo=200+$i;
```

```
    echo "<option value=\"".$arvo."\">".
$qtuote[$i]."</option>";
}

for($i=1;$i<8;$i++)
{
    $vuosi=date('y', strtotime('+'.$i.' years'));
    $vuosituote[$i]="YR-".$vuosi;
    $arvo=300+$i;
    echo "<option value=\"".$arvo."\">".
$vuosituote[$i]."</option>";
}
```

Samankaltaisella rakenteella tehtiin valikkoon tuotteet, joissa aluehintalisäys on mukana. $arvo-muuttujan arvo on määritelty suureksi, jotta tuotteen nimen päätteleminen olisi helppoa. Näin ei tarvitse välittää tuotteen nimeä, vaan arvo, jonka perusteella tuotteen nimi rakennetaan muualla sovelluksessa. Kuukausituotteet ovat 101-106, kvartaalituotteet 201-212, vuosituotteet 301-307, kuukausituotteet aluehintalisäyksellä 401-406, kvartaalituotteet aluelisäyksellä 501-512 ja vuosituotteet aluelisäyksellä 601-607.

Sähkösuojausennusteessa käyttäjä syöttää tietoihin mm. suojaushintapyynnön. Tämä on arvo, jonka käyttäjä haluaisi itselleen suojaushinnaksi. Lisäksi analyytikko syöttää järjestelmään ennusteen suojaushinnan kehittymisestä. Analyytikko syöttää järjestelmään sillä hetkellä todennäköisimmän toteutumishinnan, jonka todennäköisyys on 50%, sekä todennäköisimmän hinnan ala- ja yläpuolelta sellaiset hinnat, joiden toteutumisen todennäköisyys on 10%. Järjestelmä vertaa käyttäjän antamaa hintapyyntö näihin arvoihin ja laskee sen mukaan, mikä on todennäköisyys suojaushintapyynnön toteutumiselle.

Ensiksi tutkitaan, onko käyttäjän antamalle tuotteelle tallennettu ennustetta hinnan kehittymiselle. Jos ei ole, tulostetaan ruudulle ilmoitus, ettei tuotteelle ole vielä tällä hetkellä ennustetta. Jos ennuste on jo syötetty, aletaan selvittää, mihin todennäköisyysvälille käyttäjän suojaushintapyyntö sijoittuu.

```
if($suojhintapyynto<$ennuste)
{
    //lasketaan ala- ja ylärajan välinen erotus, sekä yhden
```

```
    //portaan rahallinen vastine
    $rahaporras=($ennuste-$alas10)/$portaat;

    for($i=0; $i<=$portaat;$i++)
    {  //luodaan taulukko vertailuhinnoista
        if ($i==$portaat)
        {   //jos on viimeinen hinta, sijoitetaan todennäköisin
            $alavertailuhinta[$i]=$ennuste;
        }
        else
        {
            $alavertailuhinta[$i]=$alas10+$i*$rahaporras;
        }
    } //taulukon luonnin loppu

    for($i=0; $i<=$portaat; $i++)
    {
        if($i==0)
        {  //jos on ensimmäinen porras
            if($suojhintapyynto<$alas10)
            {
                echo "Toteutumistodennäköisyys alle 10%.";
            }
            elseif($suojhintapyynto==$alas10)
            {
                echo "Toteutumistodennäköisyys on 10%.";
            }
        }
        else
        {
            //apumuuttuja edellisen portaan numeron
            //selvittämiseen
            $j=$i-1;

            $alaprosentti=$pohjaprosentti+$j*$prosentti;
            $ylaprosentti=$pohjaprosentti+$i*$prosentti;
            if($suojhintapyynto>$alavertailuhinta[$j] &&
$suojhintapyynto<$alavertailuhinta[$i])
            {
                echo "Toteutumistodennäköisyys on ".
$alaprosentti."-".$ylaprosentti."%.";
```

```
                }
                if($suojhintapyynto==$alavertailuhinta[$i])
                {
                    echo "Toteutumistodennäköisyys on ".
$ylaprosentti."%.";
                }
            } //else
        } //for
} //if suojhintapyynto
```

Koodi tutkii tilanteen, jossa käyttäjän antama suojaushintapyyntö on pienempi kuin todennäköisimmin toteutuva suojaushinta. Koodissa $ennuste vastaa todennäköisimmin toteutuvan suojaushinnan arvoa. $alas10 vastaa todennäköisimmin toteutuvan suojaushinnan alapuolella olevaa 10%:in todennäköisyydellä toteutuvan hinnan arvoa. Asiakas halusi, että todennäköisyysväli olisi 5%, joten portaita olisi kahdeksan. Tämä luku on tallennettu $portaat-muuttujaan. $j on apumuuttuja, jolla määritellään edellisen portaan tietoja. Jos käyttäjän syöttämä suojaushintapyyntö osuu edellisen portaan arvon $alavertailuhinta[$j] ja nykyisen portaan $alavertailuhinta[$i] välille, on suojaushintapyynnön toteutumisen todennäköisyys näiden portaiden hintojen toteutumisen todennäköisyyksien välillä. Vastaavanlaisella rakenteella on tehty myös vertailu tilanteelle, jossa käyttäjän antama suojaushintapyyntö on suurempi, kuin todennäköisimmin toteutuva suojaushinta.

5 TULOKSET JA POHDINTA

PHP:llä on helppo luoda web-ohjelmistoja. Se ei vaadi mitään erityisiä ohjelmia, vaan skriptiä voi kirjoittaa vaikkapa ihan vain muistiolla. Tällöin koodin kirjoittaminen ei ole mielekästä korosteiden puuttumisen vuoksi ja koodi on erittäin virhealtista. Onneksi on olemassa useita erilaisia ilmaisia ja maksullisia ohjelmia, joilla on tuki PHP:lle. Ilmaisohjelmista esimerkkinä itselläni käytössä oleva Notepad++. Maksullisista ohjelmista mainittakoon Dreamweaver, joka on osa Adoben CS-pakettia. Dreamweaverin eduksi voidaan laskea se, että siinä sai sekä koodin, että design-osan – joka esittää sivun ulkoasun – näkymään ruudulle, jolloin oli helpompi ajatella, mihin osaan sivustoa koodimuutokset todellisuudessa kohdistuivat.

Opinnäytetyö syvensi tietämystäni PHP:n osalta ja FusionChartsin käyttötavoista. Opinnäytetyö opetti myös, että olisi syytä pohtia ja suunnitella koodia etukäteen, eikä rynnätä suorinta tietä muuttamaan koodia tarpeen mukaan. Tällöin välttyy monilta murheilta. Jatkossa pyrin suunnittelemaan paremmin koodin rakenteen ennen sen varsinaista kirjoittamista.

Kehitettävistä asioista ensimmäisenä pitäisi tehdä projektille järkevämpi tietokanta. Tällä hetkellä tietokanta on hirvittävä sekamelska. Sen yksinkertaistaminen ja turhan toiston poisjättäminen on vähintä, mitä kannalle voisi tehdä. Lisäksi joitakin tauluja on käytetty niin, että samaan tauluun on laitettu tietoa monesta paikkaa, mikä tekee taulusta vaikean tulkita ja kenttien merkityksen selvittämiseksi joutuu käymään koodia turhaan läpi.

Tästä ongelmasta päästäänkin toiseen kehitystarpeeseen. Projektista ei ollut, eikä ole vieläkään minkäänlaista dokumentaatiota. Olisi hyvä tehdä ainakin jonkinlainen dokumentaatio, jossa kerrottaisiin ainakin, mitä tietoa löytyy mistäkin tietokannan taulusta ja mitä missäkin php-tiedostossa on. Tällöin olisi uuden tekijän helpompi ryhtyä projektiin, kun hän voisi katsoa dokumentaatiosta, mitä missäkin on. Tällöin ei tarvitse selata hirvittävää määrää koodia läpi, että saisi kuvan järjestelmästä.

Tällä hetkellä projekti ei hirvittävästi hyödynnä alustana olevaa Contao-nimistä julkaisujärjestelmää. Olisi järkevää, jos jokaiselle kaavakkeelle ja kuvaajalle tehtäisiin oma

artikkeli. Artikkeleille annetaan aliakset ja niiden avulla artikkelia voi kutsua koodissa. Tällöin ei tarvitse muistaa polkuja muuten kuin contaon artikkelien luonnissa. Aliaksella voidaan kutsua sitten itse artikkelia. Tämä myös helpottaa tilannetta, jossa artikkeliin liitetyn PHP-tiedoston sijainti muuttuu. Tällöin ei tarvitse kahlata kaikkia PHP-tiedostoa käyttäviä tiedostoja läpi, vaan sijaintimuutos tehdään artikkeliin ja tämän jälkeen järjestelmä toimii jälleen oikein.

Olen törmännyt internetissä PHP template engine -ratkaisuihin. Niillä on tarkoitus eritellä koodi ja ulkoasu toisistaan. Tällöin koodi pysyy siistinä, eikä koodin sisällä tarvitse tehdä erillisiä ulkoasumääritteitä. Se helpottaa niin koodin luettavuutta kuin ulkoasun tarkastelua ja muuttamistakin. Varmaankin suosituin template engine tällä hetkellä on Smarty. Se on ilmainen ohjelma, jota monet yritykset käyttävät työssään.

Perinteisen javascriptin voisi kokonaan tai ainakin osittain korvata jQuerylla, joka on javascriptin johdannainen. Sen avulla voi tehdä helposti näyttäviä ratkaisuja ja esimerkiksi drop-down-valikoita navigaatiopalkkiin, näyttävästi esiin tulevat help-ruudut ja niin edelleen. jQuery on niin ikään ilmaisesti ladattavissa ja käyttö on ilmaista.

Lisäksi saattaisi olla aiheellista harkita, josko Flash-ratkaisun voisi hylätä ja siirtyä javascript-versioon. Flash alkaa olla tiensä päässä ja sen tukemisen tarpeellisuutta on alettu pohtia erittäin tarkasti. Lisäksi Flash on ainakin minun kohdallani alkanut temppuilla uusimpien Firefoxin versioiden kanssa. Firefox on kuitenkin selaimena sen verran merkittävä, että sen yhteensopivuus järjestelmän kanssa olisi erittäin merkittävä.

LÄHTEET

2kmediat.com. 2012. PHP-opas: Miksi PHP? Www-dokumentti. Saatavissa: http://www.2kmediat.com/php/johdanto2.asp. Luettu: 9.10.2012.

Apache friends. 2011. XAMPP. Www-dokumentti. Saatavissa: http://www.apachefriends.org/en/xampp.html. Muokattu: 27.1.2011. Luettu: 19.10.2012.

FusionCharts. 2012. 20,000 customers. 450,000 users. 118 countries. Www-documentti. Saatavissa: http://www.fusioncharts.com/success-stories/customers/. Luettu: 9.10.2012.

FusionCharts. 2012. Breaking the ice. Www-dokumentti. Saatavissa: http://www.fusioncharts.com/company/breaking-the-ice/. Luettu: 9.10.2012.

FusionCharts. 2012. Compare Free vs. Paid. Www-dokumentti. Saatavissa: http://www.fusioncharts.com/goodies/fusioncharts-free/compare/. Luettu: 10.10.2012.

FusionCharts. 2012. Creating your First Multi-Series & Stacked Charts. Www-dokumentti. Saatavissa: http://docs.fusioncharts.com/charts/contents/FirstChart/FirstMultiSeriesChart.html. Luettu: 10.10.2012.

FusionCharts. 2012. FusionCharts Free. Www-dokumentti. Saatavissa: http://www.fusioncharts.com/goodies/fusioncharts-free/. Luettu: 10.10.2012.

FusionCharts. 2012. Tech Specs of FusionCharts Suite XT. Www-dokumentti. Saatavissa: http://www.fusioncharts.com/products/suite/tech-specs/. Luettu: 10.10.2012.

Gilmore, W. 2005. PHP & MySQL – Tehokas hallinta. Jyväskylä: Gummerus Kirjapaino Oy.

The PHP Group 2012. PHP: History of PHP. Www-dokumentti. Saatavissa: http://www.php.net/manual/en/history.php.php. Muutettu: 2012. Luettu: 8.10.2012.

The PHP Group 2012. PHP News Archive – 2012. Www-dokumentti. Saatavissa: http://www.php.net/archive/2012.php#id2012-09-13-1. Luettu: 8.10.2012.

The PHP Group. 2012. PHP: What is PHP? Www-dokumentti. Saatavissa: http://www.php.net/manual/en/intro-whatis.php. Muutettu: 2012. Luettu: 8.10.2012.