Lappeenrannan teknillinen yliopisto
Teknillistaloudellinen tiedekunta
Tietotekniikan osasto

Nettipohjainen voimalaitosratkaisujen hinta-arviojärjestelmä

Työn tarkastajana toimii prof. Jari Porras

Kandidaatintyön aihe on hyväksytty 10.10.2007

3.12.2007

Pasi Huttunen
Relanderinkatu 87 as 7
78210 Varkaus
050 - 5722 979

TIIVISTELMÄ

Lappeenrannan teknillinen yliopisto
Teknistaloudellinen tiedekunta
Tietotekniikan osasto

Pasi Huttunen

Nettipohjainen voimalaitosratkaisujen hinta-arviojärjestelmä

Kandidaatintyö

2007

37 sivua, 3 kuvaa ja 2 liitettä

Tarkastaja: Professori Jari Porras

Hakusanat: Internet-pohjaisen järjestelmän suunnittelu ja toteutus, voimalaitosratkaisujen hinta-arviolaskuri, OOXML

Keywords: Internet-based system design and implementation, cost estimate calculator for power plant solutions, OOXML

Työn tarkoitus on suunnitella ja toteuttaa nettipohjainen voimalaitosratkaisujen hinta-arviojärjestelmä Savonia Power Oy:n käyttöön. Järjestelmän tarkoitus on automatisoida voimalaitosratkaisujen tunnuslukujen laskeminen asiakkaan antamien alkuarvojen pohjalta ja tallentaa mahdollinen yhteydenottopyyntö. Järjestelmän vaatimuksina ovat laskentakaavojen helppo päivitettävyys, kaavojen automaattinen hakeminen Excel 2007–muotoisesta tiedostosta ja asiakasrajapinnan nettipohjaisuus.

Työ jakaantuu kahteen osaan. Teoriaosassa selvitetään työssä käytettyjen tekniikoiden taustaa ja selvitetään Microsoftin OOXML-tiedostomuodon rakenne työssä vaadittavin osin. Käytännön osassa suunnitellaan ja osin toteutetaan valmis järjestelmä käyttäen PHP-kieltä, XML-määrittelykieltä ja MySQL-tietokantaa. Suurimmat haasteet järjestelmän toteutuksessa ovat laskentakaavojen parsiminen Excel-tiedostosta ilman sen sisällön tiukkaa rajoittamista tiettyihin raameihin ja järjestelmän helppo päivitys saaduilla laskentakaavoilla.

Työn lopputuloksena on toimiva, muttei viimeistelty järjestelmä sekä tämä dokumentti. Työn suurin merkitys tulee olemaan edellä mainittujen suunnitteluhaasteiden selvittäminen, sekä valmis ohjelmarunko yleiseen käyttöön otetulle järjestelmälle.

ABSTRACT

Lappeenranta University of Technology
Faculty of Technology Management
Department of Information Technology

Pasi Huttunen

Internet-based Cost Estimate Calculator for Power Plant Solutions

Thesis for the Degree of Bachelor of Science in Technology

2007

37 pages, 3 figures and 2 appendices

Examiner: Professor Jari Porras

Keywords: Internet-based system design and implementation, cost estimate calculator for power plant solutions, OOXML

The goal of this work is to design and implement an Internet-based cost estimate calculator for power plant solutions for Savonia Power Oy. The function of this system is to automate the process of calculating estimate of cost of power plant solutions from values which are given by customer and save possible given request for contact. The requirements for this system are easy way to update formulas, automatic fetching of formulas from file written with Excel 2007 and Internet-based user interface for customer.

This thesis is divided to two parts. The backgrounds of technologies which are used in this thesis are researched in the theoretical part. Also the needed structure of OOXML file format is solved in this work. Working system is designed and implemented using PHP-language, XML-marking language and MySQL-database. The biggest design challenges are parsing of calculation formulas from Excel file without limiting the content of file to some strict frame and how to easily update system with newest calculation formulas.

The result of this is work is working but not necessarily fully finished system and this document. The most important outcome of this work will be solved design challenges and ready prototype code for fully finished and published system.

SISÄLLYSLUETTELO

1. JOHDANTO

1.1. Tausta

Tässä työssä on tarkoitus suunnitella järjestelmä, joka laskee automaattisesti asiakkaalle tarjotun voimalaitosratkaisun tietoja annettujen alkuarvojen perusteella. Järjestelmä pyytää asiakkaalta tietyt alkuarvot, laskee lopputuloksen käyttäen sisäisiä usein muuttuvia kaavoja ja lopulta näyttää asiakkaalle voimalaitosratkaisun hintatiedot ja esim. sähkön- ja lämmöntuottoon liittyviä tietoja.

Savonia Power Oy, joka on pieni voimalaitosratkaisuja suunnitteleva yritys, tarjoaa omia ratkaisujaan useille eri tahoille. Syy järjestelmän toteuttamiselle on se, että yrityksen markkinointihenkilö kuormittuu liikaa yhteydenottopyynnöistä ja tarkoitus olisi saada järjestelmä antamaan jo osan asiakkaan haluamista tiedoista. Järjestelmä toisi myös asiakkaalle lisäarvoa ja lisäksi myös kertoisi yrityksellemme mahdollisesti kiinnostuneesta asiakkaasta.

Järjestelmän teknillistä toteutusta määrittävät muutamat perusasiat. Järjestelmän pitää olla nettipohjainen, jotta mahdollisimman usea voisi sitä käyttää ja ettei asiakkaan tarvitse erikseen ottaa ensin yhteyttä yritykseemme laitoksen hintatietojen laskentaa varten. Järjestelmän käyttämät laskentakaavat muuttuvat usein ja tällä hetkellä ne on laadittu Excel 2007:lla luotuun tiedostoon. Helpon päivitettävyyden vuoksi kaavat tulevat pysymään kyseisessä tiedostomuodossa. Kaavat sisältävä tiedosto lähetetään tietyn väliajoin järjestelmälle, jonka työ on hakea tiedostosta esille kaavat, joita se tarvitsee laskentaan. Lisäksi tulos laskennasta pitää tallentua johonkin tiedonhallintajärjestelmään yhteydenottoa ja myöhempää tarkastelua varten.

1.2. Tavoitteet ja rajaukset

Työ koostuu teoriaosasta ja käytännön osasta. Teoriaosassa selvitetään Microsoft Office 2007:n OOXML (Office Open XML) –tiedostomuodon rakennetta etenkin SpreadsheetML:ään liittyen, XML (Extensible Markup Language) -muotoisen datan

lukemista PHP:n (PHP: Hypertext Preprocessor) avulla ja lyhyesti eri tekniikoita tiedon tallentamiseen.

Käytännön työn tavoitteena on etsiä ratkaisu kolmeen järjestelmän suunnittelussa olevaan keskeiseen ongelmaan: miten laskentakaavat saadaan kopioitua Excel-tiedostosta järjestelmän ymmärtämään muotoon, miten järjestelmä päivitetään näillä laskentakaavoilla ja miten järjestelmä tallentaa asiakkaiden tekemistä laskennoista tulevan tiedon.

Työn ulkopuolelle jää tilaustietojen raportointi järjestelmästä, kuten myös itse järjestelmän ulkoasun ja toiminnan viimeistely. Työssä ei ole niinkään tarkoitus tehdä täysin valmista toimivaa järjestelmää, vaan ratkaista ohjelmistoon liittyvät suunnitteluongelmat ja todeta valittujen ratkaisujen toimivuus käytännössä. Ohjelmiston toteutuksessa käytetään myös mahdollisuuksien rajoissa valmiita kirjastoja, eikä kaikkea pyritä tekemään itse.

1.3. Työn rakenne

Työn rakenne tulee noudattamaan alla olevaa listaa. Dokumentointia ajatellen työ jakaantuu neljään osaan. Johdannossa esitetään tutkimusongelma, teoriaosassa tarkastellaan käytettyjen tekniikoiden taustaa ja toimintaa, käytännön osassa suunnitellaan ja toteutetaan järjestelmä ja lopuksi pohditaan saatuja tuloksia.

1. Johdanto
2. Taustatietoa käytetyistä tekniikoista ja tiedon varastoinnista
3. SpreadsheetML:n rakenne
4. XML:n lukeminen PHP:n avulla
5. Järjestelmän suunnitteluongelmien ratkaisu
6. Järjestelmän suunnittelu ja määrittely
7. Järjestelmän toteutus
8. Johtopäätökset ja tulevaisuus
9. Yhteenveto

Johdannossa käsitellään yleisesti työn taustaa, syitä ja rajausta. Teoriaosassa, joka sisältää kappaleet 2-4, käydään ensin yleisemmin läpi työssä käytettyjä tekniikoita. Kappaleessa 3

paneudutaan OOXML:n taulukkolaskentaosuuteen ja selvitetään tätä aihetta tarkemmin niiltä osin, jotka työtäni koskettavat. Kappaleessa selvitetään SpreadsheetML:n toiminta siten, että tietojen perusteella on luettavissa samat asiat mitä käytän itse työssäni. Neljännessä kappaleessa selvitetään miten PHP:n avulla voidaan lukea XML-muotoista tietoa.

Työn käytännön osuus on jaettu kolmeen osioon. Ensimmäisessä osiossa, eli kappaleessa 5, ratkaistaan järjestelmän perusongelmat, eli kaavojen noutaminen ja muodostaminen Excel-tiedostosta, järjestelmän päivitys uusilla luoduilla kaavoilla ja tulosten tallennus. Kun näihin ongelmiin on löydetty ratkaisutavat, suunnitellaan itse järjestelmä kappaleessa kuusi. Lopulta kappaleessa seitsemän dokumentoidaan järjestelmän toteutus.

Viimeisessä osiossa kappaleessa 8 pohditaan käytettyjä tekniikoita, suunnitteluratkaisuja ja vastaan tulleita ongelmia ja niiden ratkaisuja, sekä mietitään miten tulevaisuudessa järjestelmää kehitetään eteenpäin. Kappale 9 sisältää yhteenvedon työstä.

2. TAUSTATIETOA KÄYTETYISTÄ TEKNIIKOISTA JA TIEDON VARASTOINNISTA

2.1. XML

SGML:stä (Standard Generalized Markup Language) johdettu XML on alun perin tarkoitettu ratkaisemaan elektronisen julkaisun ongelmia. XML on kuitenkin yleistynyt järjestelmien välisessä tiedonsiirrossa ja tiedon tallentamisessa ja XML:n pohjalta on luotu useita sovelluksia, esimerkiksi MathML (Mathematical Markup Language) matemaattisen tiedon kuvaamiseen, CML (Chemical Markup Language) kemiallisen tiedon kuvaamiseen ja SVG (Scalable Vector Graphics) vektorigrafiikan kuvaamiseen. [7, 12]

XML-muotoinen tieto kuvataan elementeillä ja niiden attribuuteilla. Elementit voivat olla sisäkkäisiä, jolloin muodostuu elementtihierarkia lapsi- ja isäelementteineen. Jokaiselle elementille voi myös halutessaan määrittää attribuutteja, jotka kuvaavat juuri tätä elementtiä. Koska elementtejä tai niiden attribuutteja ei ole itse XML:n määrityksissä mitenkään rajoitettu, voi käyttäjä luoda sellaisia rakenteita mitkä sopivat hänelle parhaiten. Yksi syy XML:n suosioon onkin sen täysi muunneltavuus. Käyttäjä itse määrää miten kieltä käyttää ja itse rajoittaa, mitä arvoja mihinkin kohtiin voidaan hyväksyä. XML onkin käytännössä tapa luoda uusia merkkauskieliä, eikä niinkään itsessään tiedon esitystapa. Tämän takia XML:ää kutsutaankin metakieleksi, eli kieleksi, jolla ilmaistaan tietoa tiedosta. [7, 13]

Vaikka XML:ää onkin nopea prosessoida koneellisesti, se on loppujen lopuksi kuitenkin selväkielistä tekstiä ja on aina luettavissa ja muokattavissa myös millä tahansa tekstieditorilla. Tämä luo lisää avoimuutta XML-tiedostoihin, kun kuka tahansa näkee käytetyn rakenteen ja tästä syystä tietojen tallentaminen XML-muodossa onkin järkevää tulevaisuutta ajatellen. Vaikka tietojen luomiseen käytetty ohjelma vanhentuisi ja unohtuisi, XML-pohjaiset tiedostot, etenkin jos ne ovat muodostettu hyvin, ovat vielä luettavissa ja ymmärrettävissä, toisinkuin suljettujen tiedostoformaattien kohdalla on. XML:n suosiota lisää myös se, että uusimmat PHP-kielen versiot sisältävät XML-jäsentimen, jonka avulla XML-muotoisen tiedon käsittely on helppoa. [7, 11]

XML-dokumentin oikeellisuutta voidaan tarkastella jäsentimen avulla kahdella tapaa. Dokumentin oikeamuotoisuutta, eli XML-kielen syntaksin täyttävyyttä tutkimalla saadaan selville vain se, onko dokumentin muotoilu sopivaa. XML-tiedoston sisältämän tiedon oikeellisuuden tarkistaminen vaatii sitten jo ulkopuolista määrittelyä sallitusta tiedosta, jota XML-dokumentit eivät itsessään vielä sisällä. Tätä hyväksytyn sisällön määrittämistä varten luotiin DTD (Document Type Definition). DTD:n avulla voidaan kuvata XML-tiedoston hyväksyttyä rakennetta. DTD:n ongelma on se, että sitä pidettiin sekä liian hankalana että se ei ole itsessään XML-muotoista dataa. Tämän takia W3C (World Wide Web Consortium) kehitti XML-skeemat. Näillä voi määritellä hyväksytyn rakenteen kuten DTD:lläkin, mutta XML-skeemat ovat itsessään XML-muotoista tietoa. [7, 14]

2.2. OOXML

Microsoft Officen vanhempien versioiden käyttämät tiedontallennusmuodot olivat usein suljettuja. Tämä aiheutti sen, että niiden lukeminen muilla ohjelmilla ei onnistunut ja jopa Officen uusimmat versiot eivät aina ymmärtäneet täysin vanhimmilla versioilla tallennettuja dokumentteja. Officen uusimman version, Office 2007:n, mukana kuitenkin tuli uusi avoin XML-pohjainen OOXML-tiedostomuoto.

Officen OOXML-muodossa tallentamat tiedostot ovat itse asiassa ZIP-muodossa pakattuja tiedostoja, jotka sisältävät XML-muotoiset määrittelyt tiedosta, tyylistä ja esitystavasta sekä relaatiot tiedostojen välillä ja mahdolliset liitetyt binääriobjektit. Koska OOXML:sta pyritään saamaan nopeutetulla tahdilla ISO-standardia ja standardointi vaatii tiedostomuotospesifikaatiolta tarvittavaa avoimuutta ja dokumentoinnin tasoa, OOXML:sta onkin olemassa standardia varten tehty ohjeistus. Spesifikaation avulla OOXML-tiedostojen lukeminen ja kirjoittaminen ovat verrattain helposti itse toteutettavissa, etenkin jos sitä verrataan aikaan, jolloin tiedostoja lukeakseen ja kirjoittaakseen piti joko tuntea suljetun tiedostoformaatin määrittely tai käyttää automaation kautta jotain Officen ohjelmaa tiedon käsittelyyn. Automaatiossa ei tietoa luettu suoraan dokumenteista, vaan jonkun ohjelmointikielen avulla käynnistettiin instanssi halutusta Officen toimistosovelluksesta ja luettiin data tämän ohjelman kautta. [2, 4]

OOXML:n taulukkolaskentaan liittyvän osa on nimeltään SpreadsheetML ja tämä määrittely kertoo missä muodossa Excel 2007 tallentaa taulukkolaskentatiedostot. Xlsx-tiedosto (eli Excel 2007:n uudessa muodossa tallennettu taulukkolaskentatyökirja) koostuu tiedostotyypin ilmaisevasta tiedostosta, relaatioista, dokumentin perusmäärittelystä ja itse taulukkolaskennan tiedoista. Yksinkertaisimmillaan tallennetaan työkirjasta yksi tiedosto, sen tyylistä, solujen teksteistä ja laskentaketjuista yhdet. Lisäksi tallennetaan yksi tiedosto per jokainen käytetty teema ja työkirjan sivu, sekä kaikki yhdistävä relaatiotiedosto. SpreadsheetML määrittelee tiedostojen hyväksytyn sisällön. [3]

2.3. PHP

PHP oli alun perin vain kokoelma Rasmus Lerdorfin luomia Perl-skriptejä. PHP:n ensimmäinen versio, silloin vielä nimeltään PHP/FI (Personal Home Page / Forms Interpreter) julkaistiin vuonna 1995. Kovan suosion myötä hän jatkoi työkalujen kehittämistä ja julkaisi PHP-kielen kakkosversion vuonna 1997. Kolmanteen versioon uusittiin kielen ydin ja muutettiin nimi PHP/FI:stä PHP:ksi. Tässä vaiheessa PHP ei ollut enää pelkän Rasmuksen projekti ja käyttäjämäärät olivat jo kasvaneet huomattaviksi. Tällä hetkellä uusin versio on viides versio ja kuudes on tulossa pian. [5, 10]

PHP on nykyään myös olioita tukeva ohjelmointikieli, jonka tärkeimpiä ominaisuuksia ovat yksinkertaisuus (toimivaa koodia voi luoda yhdellä rivillä), koodin tulkitseminen vasta ohjelman suoritusvaiheessa, heikosti tyypitetyt muuttujat, jonka takia muuttujat saavat tyypin automaattisesti arvon mukaan kontekstista riippuen, hyvä integrointi HTML-dokumentteihin ja mahdollisuus ajaa PHP palvelinohjelmiston moduulina, jolloin ei tarvitse erikseen käynnistää tulkkia suorittaakseen PHP-koodia. [5]

PHP:n suosion takia sille on myös luotu suuri määrä valmiita kirjastoja. PHP:n mukana tulee mukana paljon kirjastoja, joista yleisessä käytössä ovat etenkin MySQL-funktiot ja XML-jäsennin. Näiden takia PHP onkin niin suosittu juuri dynaamisten websivujen luonnissa, koska sivun dynamisointi ja tietokantayhteydet ovat hyvin yksinkertaiset luoda. Lisäksi etenkin kotikäyttäjiä miellyttää se, että PHP (ja MySQL) ovat ilmaisia, joten käyttö ei vaadi rahallista panostusta. [5]

Tietoturvaa ajatellen PHP on varsin turvallinen, etenkin jos se ja palvelinohjelmisto ovat asennusvaiheessa säädetty oikein ja mietitty mitä oikeuksia käyttäjät tarvitsevat. PHP mahdollistaa palvelimen tiedostojärjestelmän lukemisen ja muuttamisen, joten virheellinen koodi, suunnittelulapsukset tai liian laajojen oikeuksien antaminen väärille käyttäjäryhmille voivat antaa PHP-ohjelman käyttäjälle pääsyn palvelimen tiedostojärjestelmään, vaikkei tämä olisikaan tarkoitus. Tämän takia PHP:ssa onkin paljon valmiita käskyjä datan tarkistukselle, merkkijonojen siistimiselle mahdollisesta haitallisesta aineistosta ja muille tavoille, joilla varmistetaan, että käyttäjän antama syöte on turvallista. Tietokantakäytössä turvallisuutta voi lisätä sillä, että tietokantaan annetaan järkevät oikeudet eri käyttäjäryhmille, eikä kaikille täysoikeuksia. [5, 6]

2.4. Tiedon varastointi

Tiedon varastointiin on useitakin vaihtoehtoja. Yksinkertaisimmillaan tieto voidaan tallentaa teksti- tai binääritiedostoon ja rakentaa ohjelmaan sisäisesti tiedoston käsittelyyn vaadittava logiikka. Parannettu versio pelkästä tekstitiedostosta on XML-tiedosto, jolle löytyy etenkin PHP:sta jo valmiita funktioita käsittelyyn. Kun tarvitaan tehokasta tiedon prosessointia ja käyttäjäkohtaisia oikeuksia, XML:n rajat alkavat tulla vastaan ja johonkin tietokantasovellukseen siirtyminen on perusteltua. [1]

MySQL on ilmaisena ja hyvin PHP:n kautta tuettuna yleisin valinta avoimen lähdekoodin tietokantajärjestelmäksi, etenkin kooltaan pienemmissä järjestelmissä. MySQL venyy myös suurempiin järjestelmiin ja MySQL onkin saanut jo suurasiakkaita, esimerkkeinä NASA ja Yahoo! [5, 9]

MySQL on tietokantana varsin tyypillinen, eli relaatiotietokanta, joka täyttää pääosin SQL-standardin (Structured Query Language) käskyt, mutta lisää myös jotain omia laajennuksia. Relaatiotietokannalla tarkoitetaan tietokantaa, joka koostuu yleensä tauluista (eli relaatioista) ja taulut riveistä (tietueista) ja sarakkeista (kentistä). Sarakkeille määritetään etukäteen tietotyyppi ja sallitut arvot. SQL taas tarkoittaa kysely- ja määrittelykieltä, joka on tarkoitettu relaatiotietokannoille. Vaikka suurin osa relaatiotietokannoista käyttääkin hieman erilaista syntaksia ja laajentaa hieman SQL-määrityksiä, niin osaamalla käyttää yhtä tietokantaa osaa yleensä pienellä vaivalla myös käyttää toista. [1, 8]

Tiedon varastoinnissa (relaatio)tietokanta on yleensä kätevä tiedonhakua ajatellen. SQL-kielen avulla tietokannalle voi tehdä kyselyitä, joista palautuu kyselyn ehdot täyttäviä tuloksia. Itse ei tarvitse huolehtia tietovaraston lukemisesta ja oikeiden vastausten etsimisestä, vaan tietokantajärjestelmä tekee tämän automaattisesti ja käyttäjälle jää vain oikeiden kysymysten esittämisen vastuu. [1]

3. SPREADSHEETML:N RAKENNE

Tässä kappaleessa käsitellään OOXML:n taulukkolaskentaosaa, eli SpreadsheetML:ää. Koska pelkkä taulukkolaskentaosion spesifikaatio on jo itsessään yli 1100 sivua pitkä, käsitellään tässä vain työssäni vaadittavat asiat. Käsittely tapahtuu xlsx-tiedoston sisäinen tiedosto kerrallaan.

3.1. Workbook.xml

Kyseisessä tiedostossa määritellään työkirjan perusominaisuudet. Yksinkertaisin ja minimissään vaadittu muoto tiedoston sisällöstä on esitetty alla. Elementti *workbook* kertoo työkirjan alkaneen ja elementin *sheets* sisään tulee työkirjan kaikki laskentataulukot. Elementti *sheet* aloittaa yhden laskentataulukon määrittelyn ja sen attribuutteina annetaan laskentataulukon nimi, sisäinen indeksi ja indeksinumero, jolla viitataan laskentataulukon omaan xml-tiedostoon.

```
<workbook>
  <sheets>
    <sheet name=”Sheet1” sheetId=”1” r:id=”rId1”/>
  </sheets>
</workbook>
```

3.2. SheetX.xml

Jokaiselle tiedostossa workbook.xml määritetylle työkirjan laskentataulukolle on oma XML-tiedostonsa, jonka nimen loppuindeksi on annettu attribuutilla *r:id*. Nämä XML-tiedostot sisältävät jokaisen laskentataulukon sisältämät tiedot sekä yleisellä tasolla että myös kolumni-, rivi- ja solutasolla.

Jättämällä pois kaikki tämän työn näkökulmasta turhat elementit, saadaan yksinkertaistettu rakenne laskentataulukon sisällöstä. Elementti *worksheet* aloittaa laskentataulukon määrittelyn ja elementti *sheetData* ilmoittaa taulukon määrittelyn alkaneen. Data ilmaistaan rivejä apuna käyttäen. Elementti *row* aloittaa rivin määrittelyn ja sen attribuutti *r* kertoo rivinumeron. Lapsielementteinä ovat jokainen kyseisellä rivillä oleva solu, joka ei

ole tyhjä. Rivielementit ovat alakkain ja kaikki ne rivit, joissa on jotain sisältöä, ovat ilmoitettu.

```
<worksheet>
  <sheetData>
    <row r=”1”>
      <c …> … </c>
      …
    </row>
    …
  </sheetData>
</worksheet>
```

Rivielementin sisällä ilmoitetaan jokaisen kyseisen rivin sisältämän ei-tyhjän solun tiedot. Alla on esimerkki yhden rivin määrittelystä, joka sisältää kaksi solua, jaetun merkkijonon sisältävän solun ja kaavasolun. Solu alkaa elementillä *c* ja attribuutteina annetaan koordinaatin kertova *r*, tyyli-indeksin kertova *s* ja mahdollisesti annettu solun datatyypin ilmaiseva attribuutti *t*. Solun lapsielementteinä annetaan tarvittaessa elementti *v* solun arvoa varten, elementti *f* kaavaa varten ja elementti *is* paikallista merkkijonoa varten.

Esimerkin solu A1 sisältää tyyliviittauksen indeksille 1 ja kentän tyyppi on **s**, eli jaettu merkkijonokenttä. Solun arvona annetaan indeksi jaettujen merkkijonojen tiedostossa. Solu B1 sisältää kaavan elementissä *f*, jonka attribuutti *ca* tarkoittaa sitä, että kaava pitää laskea aina uudestaan sitä käsitellessä. Elementin *v* sisällä on kaavan viimeisin laskettu arvo.

```
<row …>
  <c r=”A1” s=”1” t=”s”>
    <v>0</v>
  </c>
  <c r=”B1”>
    <f ca=”1”>Taul2!B1 * Taul2!B2</f>
    <v>500000</v>
  </c>
</row>
…
```

3.3. Sharedstrings.xml

Excel ei kirjoita jokaisen tekstimuotoisen solun sisältöä suoraan solun arvoksi, vaan se kerää kaikki merkkijonot yhteen listaan siten, että lista sisältää vain uniikkeja merkkijonoja. Näihin merkkijonoihin sitten viitataan solun arvokentästä *v* indeksinumerolla. Tällä tavoin säästetään tilaa, kun samoja tekstiarvoja ei tarvitse kirjoittaa useasti tiedostoon.

Elementti *sst* aloittaa jaettujen merkkijonojen listan. Attribuutti *count* kertoo montako merkkijonoa, lukuun ottamatta suoraan solun arvokenttään kirjoitetut, Excel-tiedostossa oli. Attribuutti *uniqueCount* taas kertoo, montako uniikkia merkkijonoa oli ja tämä määrä vastaa listalla olevien merkkijonojen määrää. Yksittäisen merkkijonon määrittely alkaa elementillä *si* ja merkkijonon tekstiosio on elementissä *t*, lisäksi voisi olla tekstin ulkoasun määrittelyä omien elementtiensä sisällä.

Kappaleen 3.2. esimerkissä solussa A1 oli viittaus jaettuun merkkijonoon indeksillä 0. Jos jaettujen merkkijonojen XML-tiedosto olisi kuten alla oleva, olisi solun A1 arvo **Foobar.**

```
<sst count=”1” uniqueCount=”1”>
  <si>
    <t>Foobar</t>
  </si>
  …
</sst>
```

3.4. Styles.xml

Tässä tiedostossa määritellään Excel-tiedoston tyylit. Koska suurin osa tyylimäärittelyistä on työtäni ajatellen turhia, keskityn vain tarvitsemaani tyylimäärittelyyn, eli solun taustaväriin ja sen lukemiseen. Yksinkertaistettu ylätason kuvaus tiedostosta on alla, tarkemmin alemman tason elementeistä jäljempänä. Tyylimäärittely alkaa elementillä *styleSheet*, joka sisältää mm. elementit *fills*, *fonts*, *borders*, *cellStyleXfs*, *cellXfs* ja *cellStyles*. Näistä *fills*:n lisäksi kiinnostavia ovat kolme viimeistä.

```
<styleSheet>
  <fills count=N> … </fills>
  <fonts count=N> … </fonts>
  <borders count=N> … </borders>
  <cellStyleXfs count=N> … </cellStyleXfs>
  <cellXfs count=N> … </cellXfs>
  <cellStyles count=N> … </cellStyles>
</styleSheet>
```

Solun tyylin määrittäminen alkaa sillä, että solun *c* attribuutin *s* indeksiarvo etsitään *cellXfs* elementin sisältämistä lapsielementeistä. Indeksiarvo on nollasta lähtevä arvo, joka kertoo *cellXfs* elementin sisällä olevan *xf*-elementin järjestysnumeron. Esimerkissä solun *s* arvo oli 1, eli koska indeksi alkaa nollasta, olisi kyseessä toinen *xf*-rivi. Rivi sisältää yleensä attribuutteina numerotiedon esitystavan, fontin, täytön, kehystyksen ja *cellStyleXfs*:n indeksit. Muiden attribuuttien lisäksi yksi tärkeä attribuutti on *applyFill*, jonka täytyy olla 1, jotta täyttöväri otettaisiin käyttöön. Alla on esimerkki *cellXfs*-elementin sisällöstä. Ensimmäinen *xf*-rivi on oletustyyliä varten ja se ei muuta solun tyyliä mitenkään, koska se ei sisällä yhtään apply-kenttää. Toinen *xf*-rivi kertoo, että solun taustavärin tyyli-indeksi on 1 ja että taustavärin tyyliasetukset otetaan käyttöön.

```
<cellXfs count=”2”>
  <xf numFmtId=”0” fontId=”0” fillId=”0” borderId=”0” xfId=”0”/>
  <xf numFmtId=”0” fontId=”0” fillId=”1” bordered=”0” xfId=”0”
applyFill=”1”/>
</cellXfs>
```

Solun ulkonäköön voi vaikuttaa pelkän solutyylimäärittelyn lisäksi myös yleinen nimetty tyylimäärittely. Elementin *cellXfs* lapsielementin *xf* attribuutti *xfId* kertookin indeksin elementin *cellStyleXfs* lapsielementille. Tämä tyylimäärittely toimii kuten edeltäväkin, eli mahdolliset tyylimäärittelyt annetaan indekseinä ja tyylimäärittelyt otetaan käyttöön vain, jos sitä vastaava apply-attribuutin arvo on 1. Indeksin 0 rivi on tarkoitettu oletukseksi ja koska se ei sisällä mitään apply-attribuutteja, ei solun tyyliä päällekirjoiteta täällä ollenkaan. Esimerkissä *cellXfs*:n *xf*-elementin attribuutti *xfId* = 0, eli lukisimme *cellStyleXfs*:stä ensimmäisen rivin ja huomaisimme, ettei tyyli muutu mitenkään. Jos *xfID* olisi 1, tyyli pysyisi muuten samana, mutta fontti muuttuisi.

```
<cellStyleXfs count=”2”>
  <xf numFmtId=”0” fontId=”0” fillId=”0” borderId=”0”/>
  <xf numFmtId=”0” fontId=”1” fillId=”0” borderId=”0” applyFont=1/>
</cellStyleXfs>
```

Lisäksi on vielä *cellStyles* elementti, jonka lapsielementit *cellStyle* kertovat valmiit nimetyt tyylit, esimerkkinä Normal, Heading 1, jne. Elementin attribuutti *name* kertoo tyylin sisäisen nimen, attribuutti *xfId* osoittaa *cellStylesXfs* elementin lapsielementteihin ja *builtinId* osoittaa, mikä Excelin sisäinen tyyli on kyseessä, *name*-attribuutti kun on enemmän vain kuvaus, kuin oikeasti identifioiva kenttä.

```
<cellStyles count=”1”>
  <cellStyle name=”Normal” xfId=”0” builtinId=”0”/>
</cellStyles>
```

Kun nämä yllä olevat tyylimäärittelyt on käyty läpi, on selvillä eri tyyliseikkoihin vaikuttavien asioiden indeksit. Fontin, kehysteen tai numeron esitystavalla ei ole työssä merkitystä, joten tarkastelen vain solun taustaväriä. Taustavärit on taas oma indeksoitu listansa, pääelementtinä *fills* ja lapsielementteinä *fill*. Jokainen näistä lapsielementeistä sisältää yhden uniikin solun taustaväritystavan. Elementti *fill* sisältää lapsielementtinä elementin *patternFill* tai *gradientFill*. Normaalit yksiväriset taustat tulevat *patternFill* elementin kautta, jolloin sen attribuutti *patternType* kirjoitetaan arvoksi **solid**. Itse väri lopulta määräytyy *patternFill* elementin lapsielementeistä *bgColor* ja *fgColor*, joista tasaisessa värissä vain *fgColor* merkitsee. Muu kuin oletusväri annetaan attribuutilla *rgb* muodossa ARGB tai attribuutilla *indexed*, jolloin väri annetaan väri-indeksin arvolla. Jos peruspalettia ei ole ylikirjoitettu, väri-indeksin arvot osoittavat oletuspalettiin, muussa tapauksessa muokattu paletti kirjoitetaan styles.xml-tiedostoon.

```
<fills count=”2”>
  <fill>
    <patternFill patternType=”none”/>
  </fill>
  <fill>
    <patternFill patternType=”solid”>
      <fgColor indexed=”13”/>
```

```
      <bgColor indexed="64"/>
    </patternFill>
  </fill>
</fills>
```

Esimerkissä saimme siis solun elementistä *c* attribuutin *s*, joka osoitti indeksin elementin *cellXfs* lapsielementeissä *xf*, jonka attribuuteista saimme tietää solun perustyyli-indeksit. Seuraavaksi tutkittiin attribuuttia *xfId* hyväksikäyttäen *cellStyleXfs* elementti saadakseen selville, halutaanko siellä päällekirjoittaa tyyli-indeksejä. Koska ei haluttu, jäi *fillId*:ksi 1, joka osoitti elementin *fills* toiseen lapsielementtiin *fill*, josta luimme värjäystavan olevan yksivärinen ja elementti *fgColor* kertoi värin indeksiarvoksi 13, joka tarkoittaa keltaista väriä.

Sekavaisuudesta ja monimutkaisuudesta huolimatta värin ja tyylitietojen lukeminen ylipäätään on kuitenkin yksiselitteistä ja spesifikaatiossa hyvin dokumentoitu. Saman lopputuloksen tyylillisesti voi saada aikaan kuitenkin hyvin erilaisin määrittelyin ja esimerkiksi Excel 2007 tallentaa tyylimäärittelyt varsin eritavoin kuin Excel 2003, johon on asennettu lisäosa, joka osaa muuttaa tiedostot OOXML-muotoon.

4. XML:N LUKEMINEN PHP:N AVULLA

Jotta PHP-kielessä voitaisiin käyttää XML:n käsittelyyn tarkoitettuja valmiita luokkia, pitää PHP olla käännetty XML:n sallivan vivun kanssa. XML on uusimmissa PHP:n versioissa kuitenkin jo sallittu automaattisesti, joten sen käyttöönotto ei vaadi käyttäjältä mitään toimia. XML-tiedostojen lukemisen voi tehdä usealla tavalla, mutta PHP sisältää jäsentimen, jonka avulla XML-tiedostojen käsittely helpottuu. PHP:sta löytyy useampikin funktioryhmä XML-tiedon lukemiseen ja käsittelyyn ja muiden tekemiä luokkakirjastoja on lukematon määrä lisää. [7]

Työ aloitetaan luomalla jäsenninobjekti. Koska jäsennin ei sisällä itsessään logiikkaa sille, mitä luetuille elementeille tehdään, pitää itse toteuttaa funktio alkavan elementin ja loppuvan elementin käsittelyyn. Näiden funktioiden nimet annetaan jäsenobjektille metodin kautta, jolloin se osaa kutsua kyseisiä funktioita aina kun elementti alkaa tai loppuu. [7]

Jos XML-tieto on tiedostossa, seuraava vaihe on toteuttaa tiedostoa lukeva funktio, joka lukee tiedostoa tekstimuodossa ja antaa jäsentimelle luetun tiedon. Tämä tapahtuu yleensä tietyn tavumäärän kokoisina paloina, mutta mikään ei estä lukemasta kerralla koko tiedostoa merkkijonoon ja antamasta sitä jäsentimelle. Jäsentimen lukumetodi kutsuu käyttäjän toteuttamia funktioita aina tarvittaessa ja mahdollisen virheen sattuessa palauttaa arvon epätosi ja kirjoittaa jäsenninobjektiin sisäisesti tiedon tapahtuneesta virheestä. Tämä tieto on luettavissa oman metodinsa kautta ja on tarjolla virhekoodina tai selväkielisenä virheilmoituksena. [7]

Yksinkertaisen XML-tiedoston lukemiseen ei enempää tarvita. XML-laajennusosa sisältää kuitenkin lisäksi metodit XML-jäsentimen asetusten muokkaamiseen, sen hetkisen rivin, kolumnin tai tavun numeron lukemiseen ja UTF-8 enokoodaukseen ja dekoodaukseen. Lisäksi omia callback-funktioita voi määrittää itse dokumentin tiedon käsittelyyn, nimiavaruuksien alkamisen ja loppumisen käsittelyyn, notaation määrittelylle ja käsittelyohjeille. Näiden omien callback-funktioiden kautta XML-jäsentimen toiminnan voi mukauttaa hyvin pitkälle ja valita itse, mitä jäsennin lopulta lukemalleen tiedolle tekee. [7]

5. JÄRJESTELMÄN SUUNNITTELUONGELMIEN RATKAISU

Kolme suurinta ongelmaa järjestelmään liittyen ovat miten Excel-tiedoston laskentakaavat saadaan kopioitua järjestelmän ymmärtämään muotoon, miten järjestelmä päivitetään näillä uusilla kaavoilla ja miten järjestelmä tallentaa ne tiedot, joita asiakkaan tekemistä kyselyistä tulee.

5.1. Kaavojen hakeminen Excel-tiedostosta

Koska Excel 2007:n tiedostot ovat ZIP-pakattu paketti XML-tiedostoja, relaatioita ja binääriobjekteja ja koska tiedostomuodosta on olemassa julkisesti saatava tarkka spesifikaatio, on selvää, ettei kaavojen lukeminen voi olla ylivoimainen tehtävä. Valitsin ohjelmointikieleksi PHP:n osittain siksi, että se sisältää XML-jäsentimen (ja hyvät MySQL-funktiot) ja osittain siksi, että kyseinen ohjelmointikieli on minulle jo ennestään tuttu.

Vaikka PHP:n XML-tuki onkin hyvä ja vaikka xlsx-tiedostomuodosta on tarkat tiedot olemassa, on XML-tiedostojen lukeminen ja etenkin käsittely aika työlästä. Etsin netistä valmiin luokkakirjaston, jonka avulla Excel-tiedostojen lukeminen yksinkertaistuu. PHPExcel-kirjasto olikin täydellinen tähän tarkoitukseen ja sen avulla pääsi eroon mekaanisesta XML-tietojen lukemisesta ja pääsi lukemaan suoraan solujen sisältöä, tyylejä ja muita työssä tarvittavia kohtia.

Kun PHPExcel-kirjastosta tuli tarpeeksi edistynyt versio julki ja kun olin prototyypillä todistanut, että solujen värin, kaavan ja arvon lukeminen on mahdollista, olin todistanut koko työn tekemisen mahdolliseksi.

5.2. Järjestelmän päivitys uusilla kaavoilla

Järjestelmän on tarkoitus esittää asiakkaalle lomake, jossa hän syöttää tiettyjä alkuarvoja ja näiden perusteella hän saa tietää loppuarvot. Matemaattiset kaavat on tarkoitus säilyttää jossain tiedonhallintajärjestelmässä ja ne on tarkoitus pystyä päivittämään helposti

ylläpitäjän toimesta. Koska järjestelmä sijaitsee nettipalvelimella ja koska ylläpitäjän työn pitää olla päivitystilanteessa mahdollisimman vähäistä, päätin päivityksen tapahtuvan viemällä palvelimelle jossain muodossa oleva kaavat sisältävä tiedosto. Järjestelmä päivittäisi nämä uudet kaavat tiedonhallintajärjestelmäänsä ja käyttäisi tästä lähtien uusimpia kaavoja laskuissaan.

Myöhemmässä vaiheessa tultua ilmi, että ohjelma jakautuu tietoturvan ja kaavankääntämisen viemän prosessoritehon takia kahteen osaan, jossa yrityksen sisäverkossa paikallisesti pyörivällä palvelimella muutetaan Excel-tiedosto kaavoiksi ja tällä päivitetään julkisesti netissä olevan järjestelmän osan tiedot, piti kaavojen siirtoa varten niille kehittää joku kätevä tiedostomuoto. XML oli selkeä valinta helpon käytettävyyden, muuntelun ja PHP-tuen takia.

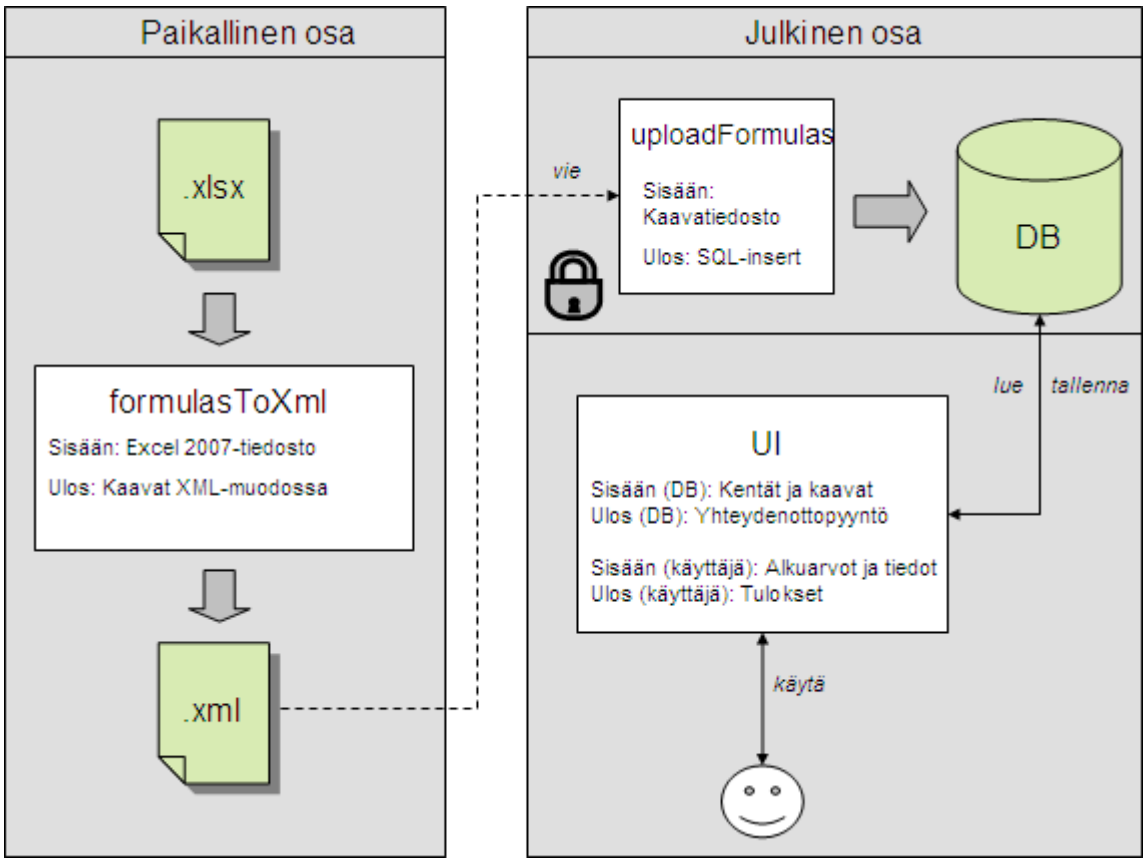
Tarkoitus olisi siis muodostaa Excel-tiedoston kaavoista XML-tiedosto, joka viedään helpon käyttöliittymän kautta palvelimelle, joka päivittää kyseisellä tiedostolla oman tietokantansa uusimpiin kaavoihin. Tiedonhallintajärjestelmäksi valitsin MySQL:n pääosin ilmaisuuden, hyvän PHP-tuen ja käyttäjäkohtaisten käyttöoikeuksien takia. Kaksi ensimmäistä syytä vaikuttivat siihen, minkä tietokantajärjestelmän valitsen ja viimeinen syy siihen, että ylipäätään valitsin tietokannan, enkä tallentanut tietoja XML-muodossa.

5.3. Tulosten tallentaminen

Koska MySQL-tietokannassa säilytetään laskennassa käytettäviä kaavoja, tätä samaa tietokantaa ajattelin käyttää myös tulosten tallentamiseen. Asiakas voisi halutun lopputuloksen saatuaan antaa omat yhteystietonsa ja lähettää nappia painamalla tarjous/lisätietopyynnön, jolloin tietokantaan tallentuisi annetut alkuarvot, laskennan tulos, käytetyn kaavakokoelman versionumero ja asiakkaan omat tiedot.

6. JÄRJESTELMÄN SUUNNITTELU JA MÄÄRITTELY

Kuvassa 1 näkyy karkeasti järjestelmän osat ja toiminta. Järjestelmä on jaettu kahtia, joista toinen osa pyörii yrityksen sisäverkossa jollain palvelimella ja tekee Excel-tiedoston kaavoista XML-tiedoston. Järjestelmän julkinen osa sijaitsee jollain webpalvelimella, on päivitettävissä kaavat sisältävällä XML-tiedostolla ja huolehtii asiakasrajapinnasta, eli kyselylomakkeen piirtämisestä, lopputulosten laskemisesta kaavojen perusteella ja mahdollisen tarjous- tai yhteydenottopyynnön tallentamisesta tietokantaan.



Kuva 1: Järjestelmän osajako ja yleinen kuvaus

6.1. Excel-tiedoston muoto

Koko työn alkuperäinen idea oli siinä, että laskentakaavat sijaitsisivat Excel-tiedostossa. Tämä siksi, että kyseistä tiedostoa päivittävät tahot eivät omista ohjelmointikokemusta, eivätkä myöskään halua syöttää kaavoja johonkin erilliseen järjestelmään, mutta osaavat käyttää Excelin funktioita. Tarkoitus ei myöskään ollut määrittää tiukasti Excel-tiedoston sisältöä, käytettyjä sarakkeita ja muuta sellaista, vaan pitää Excel-tiedoston sisältö

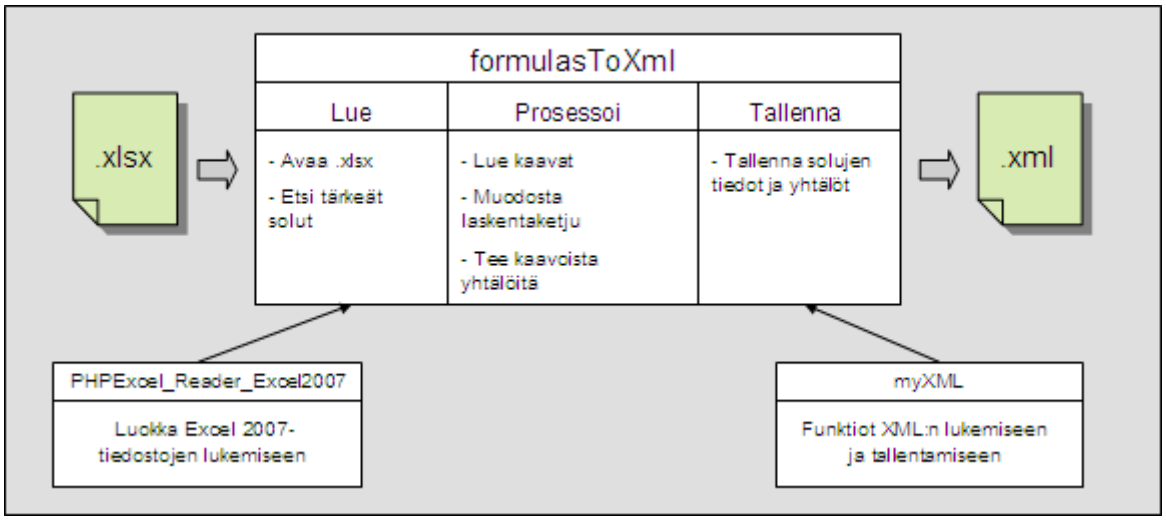
mahdollisimman rajoittamattomana. Ainoa järjestelmän kannalta tärkeä asia on se, että tiedostossa on jollain värillä erotettu alkuarvokentät ja loppuarvokentät.

Koko järjestelmän kannalta Excel-tiedostossa on kahdenlaisia merkitseviä kenttiä, alkuarvokenttiä ja loppuarvokenttiä. Alkuarvokentät ovat niitä, joihin odotetaan asiakkaalta syötettä ja loppuarvokentät ovat niitä, jotka näytetään asiakkaalle sen jälkeen kun hän on syöttänyt alkuarvot ja painanut laskennan aloittavaa nappia. Yksinkertaista esimerkkiä käyttäen alkuarvokenttä voisi olla vaikka halutun laitoksen megawattiteho ja loppuarvokenttänä laitoksen hinta, joka laskettaisiin kaavalla megawattiteho * hinta per megawatti. Excel-tiedostossa tämä olisi toteutettu siten, että soluun A1 laitettaisiin haluttu teho (alkuarvokenttä), solussa A2 olisi hinta per megawatti ja solussa A3 olisi kaava A1 * A2 (loppuarvokenttä). Tiedosto sisältää myös paljon muita kenttiä, muttei näillä ole merkitystä lopputulosta ajatellen, jos ne eivät sisälly laskentaketjuun. Järjestelmän toiminnan kannalta ei ole merkitystä vaikka Excel-tiedosto sisältäisi paljon vanhoja kaavoja, joihin ei enää viitata tai vaikka havainnollistamista varten kokonaisia välilehtiä, joiden arvot eivät tule laskentaketjuihin.

Jotta ohjelma tietää mikä on alkuarvokenttä ja mikä loppuarvokenttä, ne merkitään Excel-tiedostoon väreillä. Alkuarvokentät merkitään keltaisella taustavärillä ja loppuarvokentät punaisella taustavärillä. Tällöin ei rajoiteta niitä johonkin tiettyyn soluun tai taulukkoon, vaan ne voivat olla missä tahansa, kunhan taustaväri on kunnossa. Kenttien mahdolliset lisätiedot, eli seliteteksti ja yksikkö merkitään samalle riville kentän vasemmalle tai oikealle puolen.

6.2. Järjestelmän paikallinen osa

Kuvassa 2 on esitetty järjestelmän paikallisen osan yleinen toiminta ja sen käyttämät luokat ja funktiokirjastot. Osan toiminnasta tehdään käyttäjän kannalta todella yksinkertaista. Käyttäjä siirtää tekemänsä Excel-tiedoston tiettyyn hakemistoon, käynnistää PHP-skriptin ja tuloksena on XML-tiedosto joka sisältää kaavat tai virheilmoitus, jos jotain menee pieleen.



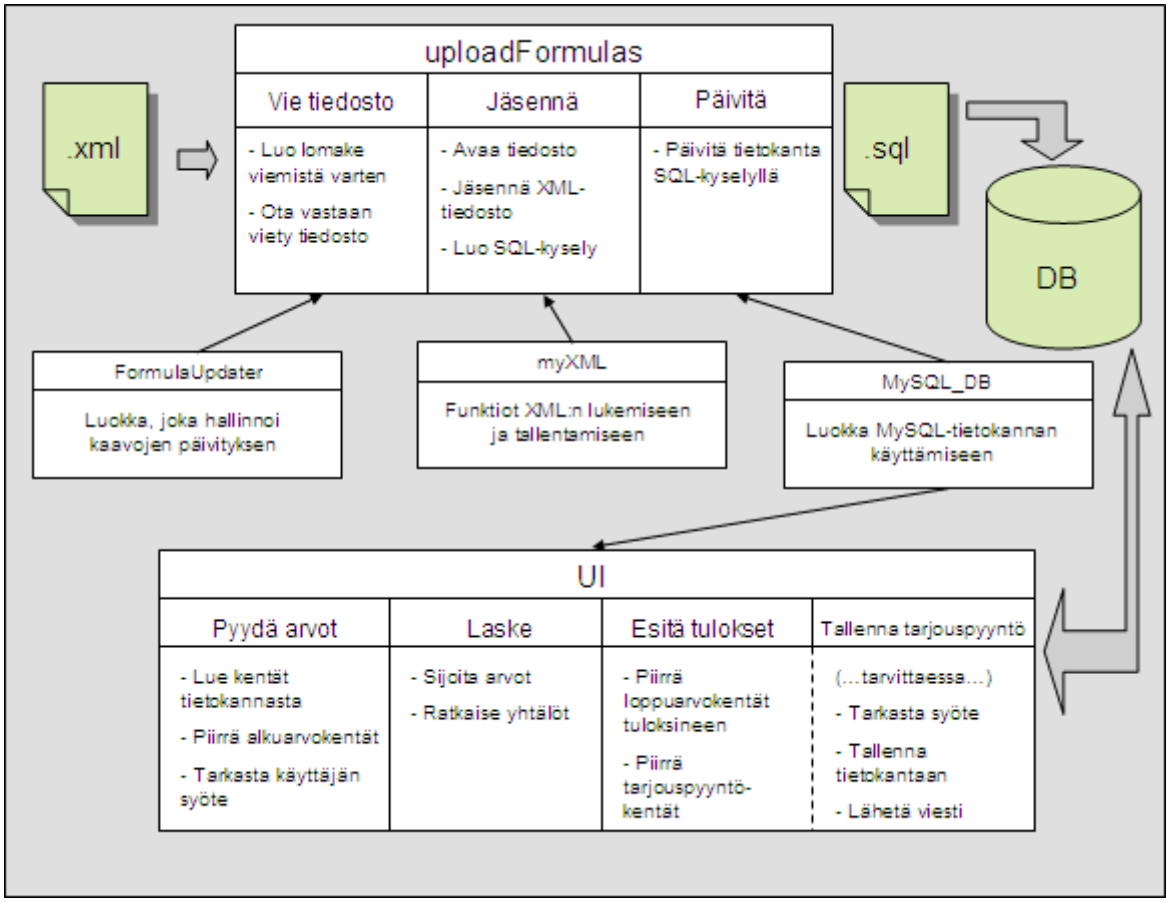
Kuva 2: Kaavamuuntimen yleinen toimintaidea

Kaavatiedosto sisältää kaiken sen infon, jonka perusteella järjestelmän yleinen osio pystyy sekä pyytämään asiakkaalta syötettä, laskemaan tulokset kuin myös esittämään ne. Kaavatiedostoon tallennetaan alkuarvosolut ja loppuarvosolut siten, että niiden sisältämät kaavaketjut on selvitetty. Ei ole syytä ylläpitää enää raskasta solujärjestelmää vastaanottavassa päässä, vaan Excel-tiedoston kaavojen soluviittauksista tehdään yhtälöitä, jotka sisältävät vain alkuarvokenttiä, vakioita tai funktioita (joiden määrä on rajoitettu pariin käytettyyn). Yhtälöiden lisäksi XML-tiedostoon talletetaan kentän perustiedot, eli kentän selite ja yksikkö.

6.3. Järjestelmän yleinen osa

Kuva 3 selvittää järjestelmän yleisen osan toimintaa. Se koostuu kahdesta käyttöliittymästä. Toinen käyttöliittymä on salasanasuojatussa hakemistossa ja liittymän kautta ylläpitäjä lähettää uudet kaavat järjestelmälle. Järjestelmä lukee saadun tiedoston, muodostaa siitä SQL-kyselyn ja päivittää tietokannan uusimpiin kaavoihin. Näitä kaavoja käytetään lennosta kun asiakas laskee laitoksen tunnuslukuja.

Toinen käyttöliittymä on asiakkaalle näkyvä pääsivu, jonka kautta alkuarvojen syöttö, loppuarvojen tarkastelu ja mahdollinen tarjouspyyntö esitetään ja tiedot talletetaan. Apuna näillä liittymillä on kuvassa näkyvät luokat ja kirjastot.



Kuva 3: Järjestelmän yleisen osan toiminta

Järjestelmän yleisen osan asiakasrajapinnan käyttöliittymä hoitaa tietojen esittämisen asiakkaalle, laskennan ja tulosten tallentamisen. Kun asiakas tulee sivuille, ruudulle tulostetaan tietokannasta alkuarvokentät selitteksteineen ja yksiköineen. Asiakas täyttää kentät ja painaa jotain nappia, jolloin järjestelmä sijoittaa tietokannassa olevien loppuarvokenttien yhtälöihin annetut arvot. Nyt kun yhtälöt eivät sisällä enää tuntemattomia muuttujia, voidaan ne ratkaista. Tulokset esitetään loppuarvokenttien muodossa taas selitteksteillä ja yksiköillä varustettuna.

Asiakas voi muuttaa alkuarvoja ja laskea lopputulokset uudelleen tai lähettää tarjous/yhteydenottopyynnön, jolloin hän syöttää tietonsa ja painaa nappia. Järjestelmän pitää tallentaa tiedot tietokantaan ja lähettää tieto yrityksellemme, että tarjouspyyntö tai pyyntö lisätiedoista tai yhteydenotosta on tehty.

7. JÄRJESTELMÄN TOTEUTUS

7.1. Järjestelmän paikallinen osa

Osan tehtävänä on etsiä Excel-tiedostosta tarvittavat kentät, ottaa niistä kaavat esille, muodostaa niistä yhtälöt ja luoda sopiva XML-tiedosto siirtoa varten. Kaavojen etsiminen ja XML-tiedoston luominen olivat helppoja asioita, mutta yhtälöiden luominen osoittautui haasteelliseksi.

Alkuarvo- ja loppuarvokenttien etsiminen toimii varsin yksinkertaisesti. Ohjelma lukee työkirjan laskentataulukoiden lukualueet läpi yksi kerrallaan ja kun löytää keltaisen tai punaisen solun, merkitsee niiden sisällön, tyypin ja koordinaatin jonoon.

Seuraavassa vaiheessa pitää muodostaa kaavoista ja soluviittauksista yhtälöt. Ohjelma alkaa käydä jonoa läpi solu kerrallaan, ottaa solun kaavan, jakaa sen palasiin ja tutkii, onko mitkään näistä palasista soluviittauksia. Jos ei ole, kaava on sellaisenaan valmis ja ohjelma merkitsee solun käsitellyksi ja siirtyy seuraavaan. Jos kaavassa on soluviittauksia, ihan ensiksi tarkastetaan, onko kyseistä solua edes jonossa. Jos kyseessä ei ole alkuarvo- tai loppuarvosolu, niin sitä ei ole alun perin lisätty jonoon ja se lisätään nyt jonon viimeiseksi. Tämä siksi, ettei turhaan selvitetä jokaisen normaalin solun laskentakaavoja, vaan vain niiden, jotka ovat laskentaketjussa. Kun solu on lisätty tai jos se oli jo ennestään olemassa, ohjelma tarkistaa, onko viitattu solu jo käsitelty. Jos kyllä, solun kaava kopioidaan ja sijoitetaan nykyisen viittauksen tilalle. Jos ei, kaavanselvitysfunktio kutsuu rekursiivisesti itseään antaen argumentiksi tämän uuden solun. Kun rekursiivinen kutsu palautuu, merkitään kyseinen viittaus käsitellyksi ja sijoitetaan saatu kaava viittauksen tilalle. Kun jokainen soluviittaus on käsitelty, on kaavakin valmis ja solu merkitään valmiiksi. Ohjelman käytyä koko jono läpi, on jokainen loppuarvosolu saatu muutettua yhtälöksi.

Lisäongelmia edellä mainittuun tekniikkaan tuo funktiot. Kaava `A1 + A2 + A3` on helppo selvittää, koska kaavassa on vain yksinkertaisia soluviittauksia ja plusmerkkejä. Sen sijaan kaava `SUM(A1:A3)` on vaikeampi, vaikka laskennallisesti kyse on samasta asiasta. On vaikea automaattisesti käsitellä funktioita, jonka takia ohjelmaan onkin luotu staattisesti käsittelytavat niille funktioille, joita tulee vastaan (alkuvaiheessa vain SUM ja LN). SUM-

funktio muuttuu siis ohjelman sisällä muotoon `A1 + A2 + A3`. Toinen ongelma on muodostuvien yhtälöiden pituus. Koska kaavat sijoitetaan sellaisenaan ilman sieventämistä joka viittauskerralla aina eteenpäin, tulee pitkästä soluviittausketjusta helposti tuhansien merkkien yhtälö, joka olisi ihmisen helppo sieventää, mutta koneen automaattisesti vaikeampi. Koska nämä kilometriyhtälöt eivät kuitenkaan nykykoneita hidasta, niin ei ole syytä tässä vaiheessa niihin vielä puuttua. Järjestelmän tulevissa versioissa yhtälöiden supistamista voi yrittää tehdä.

Yhtälöiden muodostuksen jälkeen tallennetaan yhtälöt solun tietoineen XML-muodossa siirtoa varten. Liitteessä 1 on XML-skeema, joka määrittelee sallitun XML-rakenteen. Juurielementtinä on *Fields* ja jokaista kenttää varten on oma *Field*-elementtinsä. Solun tiedot tallennetaan elementteihin *FormulaID*, *Color*, *Label*, *Unit*, *Type* ja *Formula*. Ensimmäinen tieto kertoo kaavan tunnisteen yhtälösijoituksia varten. Tätä käytetään vain alkuarvokentissä. Kaavan tunnisteet ovat muotoa ”??a??”, ”??b??”, jne ja sijoitusvaiheessa yhtälöstä etsitään kyseisiä merkkijonoja ja korvataan ne asiakkaan antamilla luvuilla. Elementti *Color* sisältää tiedon onko kyseessä alkuarvo- vai loppuarvokenttä. *Label* ja *Unit* ovat solun seliteteksti ja yksikkö. *Type* kertoo onko kyseessä teksti- vai lukusolu ja *Formula* sisältää muodostetun yhtälön.

7.2. Järjestelmän yleinen osa, kaavojen päivitys

Kaavapäivitykseen tarvittava sivu sisältää yksinkertaisen tiedostovalitsimen, johon ylläpitäjä antaa halutun XML-kaavatiedoston ja painaa nappia, jolloin tiedosto siirretään webpalvelimelle. Tämän jälkeen PHP-skripti muuttaa tiedoston SQL-lausekkeiksi ja suorittaa sen, jolloin tiedot päivittyvät tietokantaan. Tietokantataulun *fields* rakenne on esitetty liitteessä 2. Taulun rakenne on muuten sama kuin XML-tiedostossa, mutta lisäksi on sisäinen tunnistekenttä ID ja kaavaversion tunnistava kenttä VERSIONID. Tälle sarakkeelle tallennetaan kaavojen versioluku, joka on yhden suurempi kuin edellinen versio. Taulussa *version* määrätään mitä versiota kaavoista käytetään. Normaalisti käytössä on aina uusin versio, mutta halutessa voidaan käyttää myös vanhempia kaavoja, jos uusissa on jotain ongelmaa.

Koska Excel-tiedosto ja sen kääntävä PHP-skripti sijaitsevat yrityksen sisäverkossa ilman yhteyttä ulkomaailmaan ja koska julkisen osan päivityssivu on salasanojen takana, ei järjestelmä ota suuremmin kantaa XML-tiedoston mahdolliseen vaarallisuuteen. Tulevaisuutta ajatellen tämä on korjattava asia, mutta työn tässä vaiheessa ei vielä niin tärkeä.

7.3. Järjestelmän yleinen osa, käyttöliittymä asiakkaalle

Yleisen osan pääsivu on tarkoitettu asiakkaalle. Järjestelmä lukee tietokannasta uusimpien käytettävien kaavojen version ja etsii sen tiedon avulla kaavataulusta ne alkuarvokentät, joiden versionumero on sama ja printtaa nämä käyttäjälle. Kun käyttäjä on syöttänyt alkuarvokenttiin tiedot, järjestelmä tarkastaa syötteen ja sen jälkeen sijoittaa käyttäjän antamat arvot yhtälöihin, ratkaisee yhtälöt ja lopulta näyttää loppuarvokentät juuri laskettuine tuloksineen käyttäjälle.

Yhtälöiden ratkaisun voisi tehdä monella tapaa. Yksi tapa olisi jakaa yhtälö pieniin palasiin ja käydä näitä palasia läpi yksitellen ja päätellä, miten PHP:ssa sama matemaattinen asia hoidetaan. Helpointa olisi kuitenkin, jos tekstimuodossa olevan yhtälön voisi ratkaista sellaisenaan. PHP:ssa tähän on mahdollista eval-käskyn kautta, joka suorittaa argumenttina annetun merkkijonon PHP-koodina. Tällöin voi antaa tekstimuotoisen kaavan eval-käskyn argumentiksi ja PHP toimii kuin se kaava olisi koodattu normaalisti PHP:lla ja osaa laskea siitä lopputuloksen. Yksinkertaisuuden varjopuoli on tietoturvaongelmat. Eval ei erikseen tarkista mitä se suorittaa, joten jos vihamielinen käyttäjä onnistuisi syöttämään vihamielistä koodia vaikka alkuarvokenttiin oikein aseteltuna, kaavasijoituksen jälkeen PHP suorittaisi nämä yhtälöt sellaisenaan ja vihamielinen koodi tulisi ajettua. Alkuarvokenttien syötteiden tarkastus, kaavapäivitysten laittaminen salasanan taakse ja kaavanluontityökalun pitäminen nettiyhteyden ulkopuolella vähentää riskiä, mutta tulevaisuutta ajatellen tämä ongelma pitää silti korjata jotenkin.

Käyttäjä voi halutessaan muuttaa alkuarvoja ja laskea lopputuloksen uudestaan tai hyväksyä laskelman, syöttää tietonsa ja lähettää tarjous/yhteydenottopyynnön. Tietokantaan menee automaattisesti tieto asiakkaan laskemasta voimalaitosratkaisusta ja hänen yhteystiedoistaan. Asiakastietokantataulun määrittely on liitteessä 2, taulussa

contact. Tietokantaan tallennetaan asiakkaan yhteystietojen ja mahdollisen lisätietotekstin lisäksi syötetyt alkuarvot, käytetty kaavojen versionumero ja jo valmiiksi lasketut loppuarvot. Näistä solujen tiedot menevät omaan tauluunsa *contactfields* ja niiden indeksiin viitataan *contact* taulusta. Valmiiksi lasketut luvut saadaan suoraan tietokannasta lukemalla, eikä niitä tarvitse laskea aina uudestaan, mutta myös toisaalta voimme halutessamme tehdä laskut uudestaan, jos asiakkaan tilaus hieman muuttuu tai tarkentuu ja alkuarvokenttiä pitää muuttaa. Käytettyjen kaavojen versio tallennetaan siksi, että voimme ajaa tuloksia alkuarvoista sekä asiakkaan käyttämällä kaavojen versiolla että myös uusimmalla kaavojen versiolla.

8. JOHTOPÄÄTÖKSET JA TULEVAISUUS

Työn tuloksista merkittävin oli tieto ylipäätään siitä, että järjestelmä on toteutettavissa valitsemillani tekniikoilla. Excel-tiedoston kaavojen muuttaminen PHP:n ymmärtämään muotoon oli jopa helpompaa kuin alun perin kuvittelin. Kokemusten myötä Excel-muotoisten tiedostojen kirjoittaminen tuntuu myös mahdolliselta ja verrattain yksinkertaiselta toimenpiteeltä, jos tähän olisi tarvetta. OOXML-tiedostomuoto on kuitenkin varsin hyvin ja kattavasti dokumentoitu ja viiden tuhannen sivun pituutta lukuun ottamatta varsin helposti ymmärrettävässä muodossa.

Ajatus Excel-tiedoston mahdollisimman vapaasta muodosta oli toimiva. Solun taustavärin perusteella tarpeellisten solujen hakeminen oli hyvin yksinkertaista hieman sekavaa tyylimäärittelyä lukuun ottamatta ja tämän ansioista sai vältettyä raskaat säännöt Excel-tiedostojen sisällölle ja sille, mihin soluihin saa laittaa mitäkin arvoja.

Vaikka työ keskittyikin enemmän teoriaan ja prototyyppiin kuin valmiiseen koodiin, niin yrityksemme kuitenkin tarvitsee edelleen kyseisen laskentajärjestelmän, joten järjestelmän kehitys jatkuu pikaisesti etenkin koodin osalta. Tarkoitus on saada nopealla aikataululla ensimmäinen asiakkaiden käytettävissä oleva versio julkaistua käyttäen samoja tekniikoita kuin tähänkin asti. Työssä ei ilmennyt mitään sellaista, minkä takia olisi ollut perusteltua vaihtaa ohjelmointikieltä, tietokantaa tai Excel-tiedostojen automaattisen käsittelyn sijaan pakottaa kaavojen luojat antamaan kaavat jossain muussa muodossa.

Työn aikana tuli myös paljon vastaan asioita, joihin ei työn aikana ennättänyt tai halunnut vielä ottaa kantaa. Tietoturva on yksi tällainen. Vaikka järjestelmä onkin siedettävän turvallinen, se sisältää muutamia yleisesti huonona tietoturvana pidettäviä asioita ja näiden korjaaminen on tärkeää. Myös optimoinnin suhteen on paljon tekemistä. Suurin osa algoritmeista on varsin yksinkertaisia silmukoita, jotka tekevät työnsä, mutta turhan suurella prosessori- ja muistikuormalla. Järjestelmän jakaminen kahteen osaan kuitenkin vähentää optimoinnin merkitystä, koska järjestelmistä hitaampi osa on käytössä vain kaavojen hakemisen aikana, eikä asiakas pääse sitä edes käyttämään. Asiakkaan käyttämä järjestelmän osa on hyvin nopea vaikka yhtälöt olisivat tuhansia merkkejä pitkiä.

Yksi kehitettävä alue on myös Excel-tiedoston kaavojen lukeminen. Excelin funktioiden toiminnallisuus pitäisi saada siirrettyä sellaisenaan järjestelmään, jolloin ei tarvitsisi käsin toteuttaa jokaisen funktion toimintaa uudestaan. Yhtälöiden pituus on myös tulevaisuudessa ongelma, koska jo muutaman laskentataulukon työkirjassa, jossa viittaillaan ristiin laskentataulukoiden välillä, yhtälöistä tulee tuhansia merkkiä pitkiä hirviöitä. Pituus myös helposti kertautuu eksponentiaalisesti jokaisen viittauksen myötä, kun edellisen solun yhtälö sisältyy sellaisenaan mahdollisesti useitakin kertoja uuden solun yhtälöön.

Ohjelmointikielenä PHP on hyvinkin kätevä järjestelmän julkisen osan toteuttamista varten, mutta kaavat muodostavan osan ei tarvitsisi olla PHP-muodossa ja yleensä paikallisesti pyörivät ohjelmat ovat tehty jollain muulla kielellä. En kuitenkaan välttämättä näe tässä lisäarvoa, etenkin kun Excel-tiedostojen lukemiseen tarkoitettu PHPExcel-kirjasto on sekin PHP-muodossa.

Jälkikäteen en olisi välttämättä tehnyt mitään toisin työhön liittyen. PHP oli hyvä valinta ohjelmointikieleksi, MySQL on aina turvallinen valinta tietokannaksi ja Excel 2007:n xlsx-tiedostomuoto oli ennakkoluuloja yksinkertaisempi ja selkeämpi.

Työn tekeminen sujui ilman suurempia ongelmia. Aina välillä eteneminen pysähtyi jonkun uuden ongelman tultua vastaan, mutta jokainen näistä selvisi yleensä varsin nopeasti ja kehittäminen pääsi jatkumaan. Valitsemani tekniikat olivat ensimmäisiä valintojani ja onneksi työssä ei tullut tilannetta, että umpikujan takia olisi pitänyt valita jotain toisin ja tehdä osa ohjelmakoodista uudestaan. Järjestelmä on kuitenkin vasta vielä prototyyppiasteella, jonka takia on hyvin mahdollista, että järjestelmän laajentuessa ja valmistuessa ja uusien toiminnallisuuksien lisääntyessä joutuu vielä miettimään, pitääkö järjestelmän suhteen tehdä jotain suuria periaatteellisia muutoksia.

9. YHTEENVETO

Työn tarkoitus oli suunnitella ja toteuttaa nettipohjainen järjestelmä voimalaitosratkaisujen tunnuslukujen, etenkin hinta-arvion laskemiseen. Järjestelmän erikoispiirteenä oli se, että laskennan kaavat ja asiakkaalta kysyttävät tiedot sekä niiden esittämistapa luettiin automaattisesti Excel 2007-muodossa tallennetusta tiedostosta. Lisämääritteinä kaavojen ylläpito ja järjestelmän päivitys piti tapahtua mahdollisimman helposti ja automaattisesti ilman suurta mekaanista työtä. Excel-tiedostojen sisältöä ei myöskään saanut rajata suuremmin.

Työssä käytiin läpi XML:n, PHP:n, OOXML:n ja tiedonhallintajärjestelmien taustoja ja toimintaa. OOXML:n tiedostomuodon taulukkolaskentaosan spesifikaatiota käytiin läpi niiltä osin, kun se työssäni oli tarpeellista. Lisäksi työssä käytiin lyhyesti läpi miten PHP:lla luetaan XML-tiedostoja.

Käytännön osiossa ensin ratkaistiin suunnittelun suurimmat ongelmat, eli miten Excel-tiedostosta saadaan luettua kaavat, miten näillä kaavoilla päivitetään järjestelmää ja miten asiakkaan syöttämät tiedot tallennetaan. Tässä vaiheessa myös valittiin käytetyt kielet ja tekniikat, eli PHP ohjelmointiin, XML kaavatiedoston säilyttämiseen ja MySQL-tietokantaratkaisuksi. Pienen testailun jälkeen kaavojen lukeminen päätettiin tehdä PHPExcel-kirjastoa käyttäen ja solujen laskentakaavaketjuista oli tarkoitus tehdä yhtälöitä, jotka talletetaan XML-muotoiseen tiedostoon. Kaavojen päivitys tapahtuisi automaattisesti vain lähettämällä XML-tiedosto palvelimelle. Asiakkaan antamat tiedot talletettaisiin tietokantaan ja niistä lähetettäisiin sähköpostia yrityksellemme. Suunnitteluongelmien ratkaisun jälkeen suunniteltiin ja määriteltiin järjestelmän toimintaa. Lopuksi tämä järjestelmä toteutettiin ja käytiin läpi toteuttamisen vaiheet, ongelmat ja lopputulokset.

Työn toteuttaminen onnistui ilman suurempia ongelmia ja jatkokehitystä ajatellen järjestelmä sopii sellaisenaan viimeisteltäväksi, eikä käytettyjä tekniikoita tarvitse muuttaa.

LÄHDELUETTELO

1. Allen, C., Creary, C. ja Chatwin, S. *Introduction to Relational Databases*. McGraw-Hill Osborne Media, 2003.

2. Chappell, D. *Understanding ActiveX and OLE*. Microsoft Press, Washington, 1996.

3. Ecma International. *Standard ECMA-376*. Internet WWW-sivu, URL: http://www.ecma-international.org/publications/standards/Ecma-376.htm (15.10.2007).

4. Ecma International. *OpenXML White Paper*. Internet WWW-sivu, URL: http://www.ecma-international.org/news/TC45_current_work/OpenXML%20White%20Paper.pdf (15.10.2007).

5. Gilmore, J. *Beginning PHP and MySQL5: From Novice to Professional, Second Edition*. Apress, 2006.

6. Heinisuo, R. ja Rauta, I. *PHP ja MySQL. Tietokantapohjaiset verkkopalvelut*. Talentum Media Oy, 2007.

7. Holzner, S. *Inside XML*. New Riders Press, 2000.

8. MySQL AB. *MySQL AB :: MySQL 6.0 Reference Manual :: 1 General Information*. Internet WWW-sivu, URL: http://dev.mysql.com/doc/refman/6.0/en/introduction.html (20.11.2007).

9. MySQL AB. *MySQL AB :: MySQL Customers by Industry*. Internet WWW-sivu, URL: http://www.mysql.com/customers (20.11.2007).

10. The PHP Group. *PHP: History of PHP and related projects*. Internet WWW-sivu, URL: http://fi2.php.net/manual/en/history.php (27.11.2007).

11. The PHP Group. *PHP: XML Parser Functions – Manual*. Internet WWW-sivu, URL: http://www.php.net/xml (12.10.2007).

12. W3C. *Extensible Markup Language (XML)*. Internet WWW-sivu, URL: http://www.w3.org/XML (12.10.2007).

13. W3C. *Extensible Markup Language (XML) 1.1 (Second Edition)*. Internet WWW-sivu, URL: http://www.w3.org/TR/xml11 (29.11.2007).

14. W3C. *XML Schema Part 1: Structures Second Edition*. Internet WWW-sivu, URL: http://www.w3.org/TR/xmlschema-1 (29.11.2007).

Liite 1. Kaavatiedoston XML-skeema

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="Fields">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="Field" minOccurs="0" maxOccurs="unbounded">
     <xs:complexType>
      <xs:sequence>
       <xs:element    name="FormulaID"    minOccurs="0"    maxOccurs="1"
type="xs:string"/>
       <xs:element name="Color" minOccurs="1" maxOccurs="1">
        <xs:simpleType>
         <xs:restriction base="xs:string">
          <xs:enumeration value="input"/>
          <xs:enumeration value="output"/>
         </xs:restriction>
        </xs:simpleType>
       </xs:element>
       <xs:element      name="Label"      minOccurs="1"      maxOccurs="1"
type="xs:string"/>
       <xs:element      name="Unit"      minOccurs="0"      maxOccurs="1"
type="xs:string"/>
       <xs:element name="Type" minOccurs="1" maxOccurs="1">
        <xs:simpleType>
         <xs:restriction base="xs:string">
          <xs:enumeration value="num"/>
          <xs:enumeration value="string"/>
         </xs:restriction>
        </xs:simpleType>
       </xs:element>
       <xs:element     name="Formula"     minOccurs="1"     maxOccurs="1"
type="xs:string"/>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

Liite 2. Tietokantarakenne

Taulu: fields

Field	Type	Null	Default	Comments
ID	mediumint(8)	No		primary key, auto_increment
VERSIONID	mediumint(8)	No		
FORMULAID	varchar(8)	Yes	*NULL*	
COLOR	varchar(25)	No		
LABEL	text	No		
UNIT	varchar(25)	Yes	*NULL*	
TYPE	varchar(25)	No		
FORMULA	text	No		

Taulu: version

Field	Type	Null	Default	Comments
VERSIONID	mediumint(8)	No		

Taulu: contact

Field	Type	Null	Default	Comments
ID	mediumint(8)	No		primary key, auto_increment
VERSIONID	mediumint(8)	No		
COMPANY	varchar(100)	No		
PERSON	varchar(100)	Yes	*NULL*	
ADDRESS	text	Yes	*NULL*	
EMAIL	varchar(50)	Yes	*NULL*	
TEL	varchar(50)	Yes	*NULL*	
REQUEST	text	Yes	*NULL*	

jatkuu

Taulu: contactfields

Field	Type	Null	Default	Comments
ID	mediumint(8)	No		primary key, auto_increment
CONTACTID	mediumint(8)	No		
FIELDID	mediumint(8)	Yes	*NULL*	
COLOR	varchar(25)	No		
VALUE	text	No	*NULL*	