Metropolia Ammattikorkeakoulu
Tietotekniikan koulutusohjelma

Mikko Yltävä

Tarjouspyyntöpalvelun toteutus verkkosovelluksena

Insinöörityö 13.12.2009

Ohjaaja: toimitusjohtaja Ville Tomperi
Ohjaava opettaja: yliopettaja Jarkko Vuori

Metropolia Ammattikorkeakoulu Insinöörityön tiivistelmä

Tekijä	Mikko Yltävä
Otsikko	Tarjouspyyntöpalvelun toteutus verkkosovelluksena
Sivumäärä	39 sivua
Aika	13.12.2009
Koulutusohjelma	tietotekniikka
Tutkinto	insinööri (AMK)
Ohjaaja	toimitusjohtaja Ville Tomperi
Ohjaava opettaja	yliopettaja Jarkko Vuori

Opinnäytetyö tehtiin projektityönä ilmastointi- ja ilmankäsittelylaitteita maahantuovalle Ventur Oy:lle, jolla oli tarve tehostaa ja nopeuttaa teollisuuspuhaltimien tarjouspyyntöprosessia. Tällä hetkellä jälleenmyyjät tekevät yritykselle tarjouspyyntöjä puhelimitse eivätkä pyynnöt aina sisällä riittävästi tietoa sopivan puhaltimen valintaa ajatellen. Ratkaisuna tähän ongelmaan suunniteltiin verkkopalvelu, jossa jälleenmyyjät voivat valita, täyttää ja lähettää tarjouspyyntölomakkeita. Lomakkeiden tulisi kysyä käyttäjältä tarjouksen kannalta oleellisia kysymyksiä ja myös tarjota vastaamiseen tarvittavaa ammattitietoutta. Lisäksi tarvittaisiin työkalu, jolla yrityksen vastuuhenkilö voisi tulevaisuudessa helposti muokata lomakkeen rakennetta ja kysymyksiä halutunlaisiksi sekä hallinnoida palvelun yleisiä asetuksia.

Työ aloitettiin tutustumalla ratkaisuvaihtoehtoihin ja rajaamalla niistä vaatimusten ja aikataulun kannalta sopimattomat vaihtoehdot pois. Pohdinnan tuloksena päädyttiin käyttämään Ajax-tekniikkaa, jolla muokkaustyökalu saataisiin käytettävyydeltään riittävän hyväksi. Tämän jälkeen perehdyttiin tarkemmin Ajax-arkkitehtuuriin sekä muutamaan muuhun verkko-ohjelmoinnissa yleisesti käytettyyn tekniikkaan.

Aluksi kehitettiin demonstraatioversio, jolla voitiin kokeilla erilaisten ratkaisujen toimivuutta verkkoympäristössä ja niiden vaikutusta käytettävyyteen. Pian tämän jälkeen kävi ilmi, että lomakkeita kuvaavasta tietorakenteesta joudutaan tekemään suoraviivaisempi ja yksinkertaisempi kuin aiemmin oli suunniteltu. Nyt voitiin asettaa projektille tarkka aikataulu ja siirtyä määrittelemään lomakepalvelulle ja muokkaimelle asetettavia vaatimuksia.

Seuraavassa vaiheessa määriteltiin edellisten vaiheiden pohjalta tarvittava toiminnallisuus ja lomakkeiden kuvaamiseen vaadittava tietorakenne. Lopuksi toteutettiin molemmat verkkopalvelun osat, käyttöohje muokkaimelle ja palvelu siirrettiin tuotantopalvelimelle käyttöönottoa odottamaan. Työn tuloksena asiakasyritys sai käyttöönsä työkalun, jonka käytön odotetaan nopeuttavan ja tehostavan tarjouspyyntöprosessia lähitulevaisuudessa.

Hakusanat	verkkosovellus, verkkopalvelu, Ajax, PHP, XML, XSLT

Helsinki Metropolia University of Applied Sciences Abstract

Author	Mikko Yltävä
Title	Implementing a call-for-bidding-service as a web application
Number of Pages	39
Date	13 December 2009
Degree Programme	Information Technology
Degree	Bachelor of Engineering
Instructor	Ville Tomperi, CEO
Supervisor	Jarkko Vuori, Principal Lecturer

This thesis was written within the context of a project assignment for Ventur Oy Ab. The company is an importer of air processing equipment and specializes in industrial fans. The call-for-bids-process currently in use in the company could be described as slow and inefficient. The resellers make contact by phone and often lack the specific technical information needed to choose a proper product for their needs.

A solution for the problem would be a web-based service to handle the call for bids process. The service would host a collection of forms for different needs to be filled in and sent online. The forms would contain the most relevant questions and also provide information concerning the important technical details. In addition an editing tool would be needed for modifying the form contents and configuring the service.

The project began by discarding the unfit solutions with regard to schedule and requirements from the employer. A decicision was made to utilize Ajax technology to acquire the needed usability. After this further research was undertaken into Ajax and the usual accompanying web technologies.

A demonstration version was hastily developed for experimentation with different available options on browser-based user interface. Soon afterwards it was observed that the data structure beneath the forms had to be streamlined and simplified. A final schedule was set and the requirement specification phase was about to begin.

In the next phase the necessary functionality and the data structure for storing the forms was defined. Both interfaces of the service were implemented fully, a user manual was written and the application was transferred to a production server to await future deployment. As a result of the project, the customer was provided with a tool, expected to enhance and hasten the call-for-bids process.

Keywords	web application, online service, Ajax, PHP, XML, XSLT

Sisällys

1 Johdanto

Ventur Oy Ab on espoolainen ilmastointi- ja ilmankäsittelylaitteita maahantuova yritys, joka kuuluu Venture Industries -konserniin.Yrityksen erikoisalaa ovat teollisuuden kohdepuhaltimet, joiden kirjo on huomattavan suuri, kuten kuvasta 1 voidaan nähdä. Valittaessa käyttötarpeeseen sopivaa puhallinta täytyykin ottaa huomioon lukuisia eri seikkoja liittyen paitsi puhaltimen ominaisuuksiin myös sen tulevaan käyttöympäristöön.

Yrityksessä tällä hetkellä käytössä oleva tarjouspyyntöprosessi on aikaa vievä ja tehoton; tarjouspyyntöjä tekevien jälleenmyyjien koulutus ja ammattitieto vaihtelevat suuresti ja puhelimitse tehtäviin tiedusteluihin joudutaan usein vastaamaan tarkentavilla lisäkysymyksillä, joihin tarjouspyynnön tehnyt henkilö puolestaan osaa harvoin kertoa vastausta välittömästi. Tarjouspyyntöjen tekeminen hidastuu ja vaikeutuu tai on pahimmassa tapauksessa jopa vaarassa keskeytyä.

Kuva 1: Ventur Oy Ab:n tuotevalikoimasta löytyviä puhaltimia [1].

Pekka Huhtaniemen insinöörityön [2] yhtenä tavoitteena oli tehostaa tarjouspyyntöprosessia luomalla työkalu, joka auttaisi jälleenmyyjää määrittelemään sopivan puhaltimen valintaa varten oleellisimmat tekniset tiedot ja parantaisi käytettyä ammattikieltä. Työssä päädyttiin ratkaisemaan ongelma rakentamalla verkkopalvelu, jossa jälleenmyyjät voivat tehdä tarjouspyyntöjä. Palveluun kirjauduttaisiin yksilöllisillä tunnuksilla ja vastattaisiin esitettyihin kysymyksiin koskien tarvittavan puhaltimen ominaisuuksia. Tämän jälkeen vastaukset arkistoitaisiin ja välitettäisiin

asiakasyritykselle, joka voisi edelleen ottaa yhteyttä jälleenmyyjään. Tarjouspyyntöjen vastaanottamisen ohella palvelun tulisi opastaa käyttäjiään tunnistamaan puhaltimen valinnan kannalta kriittiset arvot ja näin osaltaan parantaa jälleenmyyjien ammattitietoutta.

Ratkaisua voitaisiin pitää onnistuneena, jos se nopeuttaisi ja parantaisi tarjouspyyntöprosessia. Tämä näkyisi turhien tarkistusyhteydenottojen vähentymisenä, tarkentuneina kysymyksinä ja jälleenmyyjien lisääntyneenä ammatillisena tietämyksenä. [2, s. 7–14.]

Tämän työn aiheena oli verkkopalvelun suunnittelu ja toteuttaminen projektiluontoisena työnä. Varsinaisen jälleenmyyjille suunnatun tarjouspyyntöpalvelun lisäksi työssä kehitettiin Ventur Oy Ab:n omaan käyttöön tuleva muokkain, jolla yrityksen vastuuhenkilö voi hallinnoida palvelua. Tekijälle aihe tarjosi tilaisuuden perehtyä Ajax-arkkitehtuuriin sekä syventää aiempaa osaamista verkko-ohjelmoinnissa yleisesti käytettyjen teknologioiden osalta. Lisähaastetta työhön toi käyttövalmiin verkkopalvelun pystyttäminen tiukahkossa aikataulussa sekä tasapainottelu käytettävyyden, toimintavarmuuden ja laajennettavuuden välillä.

Toteutusta edeltäneessä suunnitteluvaiheessa tutustuttiin aihealueeseen ja rakennettiin nopeassa tahdissa tarjouspyyntöpalvelusta demonstraatioversio, jolla voitiin testata erilaisten tekniikoiden ja ulkoasujen soveltuvuutta projektiin. Työn seuraavassa osassa käydään läpi tämä työvaihe; esitellään keskeiset käsitteet ja lopputuloksena syntyneet valinnat. Kolmannessa osassa on dokumentoitu näiden päätösten pohjalta valmistuneen ohjelmiston arkkitehtuuri ja tekniset yksityiskohdat. Neljännessä osassa tehdään yhteenveto projektin tuloksista sekä pohditaan palvelun mahdollista jatkokehitystä ja työn tarjoamia opetuksia.

2 Teoriaa ja käsitteitä

2.1 Toteutustavan valinta

Asiakasyritys ja toisaalta palvelun käyttäjät asettivat projektin toteutukselle hyvin selkeät rajat. Yrityksen käytössä ei ollut toiminnanohjaus- tai asiakkuudenhallintajärjestelmiä, joihin tarjouspyyntöpalvelu olisi pitänyt integroida, eikä millään käyttäjäryhmällä ollut erityistä tietoteknistä osaamista. Lisäksi palvelun tulisi olla käyttövalmiina melko lyhyessä ajassa eikä mahdollisesta jatkokehityksestä ollut mitään takeita. Toisaalta käyttäjien tai palvelun läpi kulkevien tarjouspyyntöjen määrä ei myöskään tulisi vaatimaan erityistä suorituskykyä tai skaalautuvuutta.

Näistä lähtökohdista katsoen työssä oltiin valmiita uhraamaan laajennettavuutta, suorituskykyä ja jossain määrin myös käytettävyyttä toimintavarmuuden ja nopean toteutuksen hyväksi. Projekti voitiin katsoa onnistuneeksi vain, kun palvelu on täydellisesti toimintakunnossa. Päätökset arkkitehtuurista ja tietomalleista pyrittiin lyömään lukkoon aikaisimmassa mahdollisessa vaiheessa ja toiminnallisuudessa tyydyttiin ensimmäisenä löytyneeseen toimivaan vaihtoehtoon.

Ideointivaiheessa mietittiin erilaisten tietoteknisten ratkaisujen soveltuvuutta tarjouspyyntöprosessin tehostamiseen. Perinteisen työpöytäsovelluksen kehitystyö on yleensä suoraviivaista ja monimutkaistenkin käyttöliittymien suunnittelu on kehitysympäristön avulla verrattain helppoa. Etenkin muokkaimen kohdalla korkean tason ohjelmointikielellä toteutettava paikallinen sovellus vaikutti varteenotettavalta vaihtoehdolta. Varsinaisen tarjouspyyntöpalvelun toteutuksen kannalta taas ohjelmiston ja päivitysten jakelu olisi ollut työlästä ja alustariippuvuus olisi saattanut aiheuttaa ongelmia jatkossa.

Yhtenä mahdollisuutena pohdittiin myös palvelun toteuttamista jonkin valmiin työpöytäsovelluksen, kuten Microsoft Office Excelin, laajennuksena. Hyödyntämällä laajalti käytössä olevaa toimisto-ohjelmistoa olisi saavutettavuusongelmista päästy ainakin jossain määrin eroon ja ympäristö olisi ollut käyttäjille ennestään tuttu. Tosin tässäkin tapauksessa palvelun päivitys olisi vaatinut toimenpiteitä käyttäjiltä.

Lopulta päädyttiin kuitenkin toteuttamaan sovelluksen molemmat osat *verkkopalveluna*. Tällöin käyttäjiltä vaadittaisiin vain suhteellisen uusi Internet-selain ja päivitykset voitaisiin hoitaa keskitetysti yhdelle palvelimelle. Mahdollisia riskejä tässä lähestymistavassa ovat pienet eroavaisuudet selainten toiminnassa ja ongelmat tietoliikenneyhteyksissä, jotka saattavat tehdä palvelun käyttökelvottomaksi. Vaikka verkkopalvelusta ei usein saada käytettävyydeltään aivan perinteisen työpöytäsovelluksen veroista, työssä vaadittava taso voitaisiin saavuttaa asetetun aikataulun puitteissa. Tämän mahdollistaisi Ajax-arkkitehtuuri, johon perehdytään tarkemmin seuraavissa luvuissa.

2.2 Verkkopalvelun toteutustekniikoita

Verkkopalvelu voidaan määritellä joukoksi sähköisessä verkossa olevia aktiviteetteja, jotka tuovat lisäarvoa käyttäjälle [3, s. 25] tai vielä yksinkertaisemmin palveluksi, joka tarjotaan www-sivuston kautta [4]. *Verkkosovellus* puolestaan on verkkopalvelun erikoistapaus, joka on sisällöltään toiminnallinen ja staattisia verkkosivustoja monimutkaisempi ja vuorovaikutteisempi [3, s. 26].

Työtä tarkasteltaessa voitaisiin sanoa jälleenmyyjille tarjotun lomakepalvelun edustavan verkkosivustoa, koska käyttäjän mahdollisuudet vaikuttaa tarjottuun sisältöön ovat lopulta hyvin rajoitetut. Palvelun hallinnointiin tarkoitettu muokkain olisi sen sijaan selkeästi verkkosovellus. Tässä työssä palvelulla viitataan joko tarjouspyyntöpalveluun kokonaisuutena tai jälleenmyyjien käyttöön tulevaan osuuteen sovelluksesta.

Perinteisesti verkkosovellusten toteutuksessa on käytetty palvelinkeskeistä ratkaisutapaa, jossa asiakasohjelmasta käytetään nimeä *ohut asiakas*. Tässä vaihtoehdossa liiketoimintalogiikka ohjelmoidaan palvelimelle ja selaimen osaksi jää vain palvelimen tuottamien HTML-sivujen esittäminen sekä käyttäjän tekemistä toiminnoista viestiminen. Rooli on samankaltainen keskusyksikköön kiinnittyvän “tyhmän päätteen” kanssa ja Internet-selainta voisikin luonnehtia alustariippumattomaksi pääteohjelmaksi.

Esimerkiksi Internetin alkuaikoina yleistyneet CGI-ohjelmat kuuluvat tähän kategoriaan siinä missä esimerkiksi YTV:n Reittiopas. Yksinkertaisuuden vastapainona ohut asiakas

-tyyppisten verkkosovellusten käytettävyys ja käyttäjäinteraktio saattavat kärsiä: verkkosivuilla käytettävät tietokentät ja painikkeet soveltuvat hyvin suoraviivaiseen palauteviestintään, mutta pitkäkestoisessa muokkaustyössä niillä rakennettava käyttöliittymä vaatisi käyttäjää suorittamaan huomattavan paljon tarpeettomia toimintoja.

Rikkailla Internet-sovelluksilla (Rich Internet Application, RIA) tarkoitetaan verkkosovelluksia, jotka pyrkivät puuttumaan ilmenneisiin ongelmiin. Tämä tapahtuu siirtämällä toiminnallisuutta, kuten käyttäjäinteraktion hallinta, palvelimelta selaimen suoritettavaksi. Rikkaiden Internet-sovellusten toteutuksessa voidaan hyödyntää selaimeen asennettavia liitännäisiä (plugin) tai laajempia ohjelmistokehyksiä ja sovellusalustoja. Yksi varsin suosittu liitännäinen on Adobe Flash Player ja jälkimmäiseen kategoriaan kuuluvat esimerkiksi Adobe Integrated Runtime (AIR) ja Microsoft Silverlight. [5, s.5–12; 6.]

Kilpailevien alustojen joukossa uusi tulokas on JavaFX, jolla voidaan toteuttaa Java-virtuaalikoneella ajettavia rikkaita sovelluksia, jotka ovat käytettävissä työasemien lisäksi esimerkiksi mobiili- tai viihdelaitteilla. JavaFX-sovelluksissa käyttöliittymä määritellään kuvailevalla (deklaratiivisella) kielellä ja sen elementtejä voidaan sitoa helposti niitä vastaavaan tietomalliin. Tapahtumankäsittelyä helpottavana ominaisuutena muuttujille voidaan asettaa liipaisin eli heräte (trigger), joka suoritetaan muuttujan arvon muuttuessa. Vaikka kieli ei ole puhtaasti funktionaalinen ohjelmointikieli, se sisältää funktionaalisia piirteitä: funktiota voidaan käyttää muuttujan arvona, paluuarvona tai parametrina. JavaFX-kielisen sovelluksen sisällä voidaan myös käyttää perinteisiä Java-luokkia. [32, s. 1, 44; 10, s. 13–14.]

Työlle asetettujen tavoitteiden saavuttamista silmällä pitäen ohut asiakas -tyyppinen ratkaisu ei tarjoaisi riittävästi käytettävyyttä. Jo tässä vaiheessa oli selvää, että palvelun asetuksia muokattaessa tultaisiin tekemään useaan eri paikkaan paljon pieniä muutoksia, joiden vaikutusten tulisi näkyä käyttäjälle välittömästi. Perinteisillä lomakkeen lähettämiseen tarkoitetuilla HTML-elementeillä ja GET/POST -menetelmillä muokkaus olisi toki mahdollista mutta hidasta ja vaivalloista, koska pieninkin käyttäjän tekemä toiminto aiheuttaisi koko sivun päivittämisen. Aiemmat kokemukseni hieman samankaltaisen käyttöliittymän suunnittelusta puolsivat vahvasti vaihtoehtoisia ratkaisutapoja.

Jonkin liitännäisen tai ohjelmistokehyksen hyödyntäminen korjaisi edellä mainitut puutteet mutta saattaisi tuoda mukanaan uusia ongelmia. Liitännäisten ohjelmoinnissa käytettävät kielet olivat itselleni tuntemattomia ja niiden soveltuvuudesta lomakepalvelun varastoiman rakenteellisen tiedon käsittelyyn ei ollut käsitystä. Raskaammissa kehyksissä ja alustoissa tämä puoli olisi varmasti kunnossa, mutta tällöin asiakasyrityksen käyttämille tietokoneille jouduttaisiin asentamaan erillinen sovellus ja verkkopalveluille tyypillinen saavutettavuus menetettäisiin. Kokonaiseen sovellusalustaan perehtyminen ja uuden ohjelmointikielen opiskelu veisivät myös huomattavasti aikaa, jota ei ollut muutenkaan käytettävissä liiaksi asti.

Näiden kahden mallin väliltä löytyy kuitenkin myös kolmas vaihtoehto; rikas Internet-sovellus voidaan toteuttaa myös käyttämällä selainten mukana toimitettuja tekniikoita. Tähän menetelmään viitataan yleensä nimellä *Ajax*, ja siitä kerrotaan tarkemmin seuraavassa luvussa. [5, s. 17–22.]

2.3 Ajax

Helmikuussa 2005 julkaistussa artikkelissa Jesse James Garrett määritteli ensimmäisenä käsitteen Ajax, joka on lyhenne sanoista Asynchronous JavaScript + XML. Garrettin mukaan kyseessä on lähestymistapa, jossa yhdistellään useita toimivia teknologioita uudella ja ennen näkemättömällä tavalla. Perinteisessä mallissa käyttäjän tekemä toiminto lähettää HTTP-pyynnön palvelimelle, joka puolestaan palauttaa uuden kokonaisen sivun näytettäväksi. Ajax-mallissa uutta on selaimen ja palvelimen väliin lisättävä kerros, joka reagoi käyttäjän toimiin ja huolehtii palvelimen kanssa tehdyistä tiedonsiirroista sekä selaimen päivittämisestä. [7.]

Koska osa sovelluslogiikasta voidaan tällä tavalla siirtää selaimessa ajettavaksi, palvelin voi keskittyä tarjoamaan tietoa valmiiksi muotoillun sisällön sijaan. Vaikka liikennöintitiheys selaimen ja palvelimen välillä luultavasti kasvaa sujuvamman käyttöliittymän myötä, liikenteen määrä itse asiassa pienenee, koska päivitykset sisältävät vain käyttäjän suorittaman toiminnon kannalta oleellista tietoa. [8, s. 17–20.]

Asiaa käytettävyysnäkökulmasta katsoen voidaan sanoa, että Ajax-sovellus tarjoaa monipuolisen ja jouhean käyttäjäkokemuksen rakentuen standardien web-tekniikoiden varaan. Ajax siis mahdollistaa sellaisten verkkosovellusten tuottamisen, jotka voivat kilpailla käyttömukavuudessa perinteisten työpöytäsovellusten kanssa, mutta vaativat ajoalustakseen vain selaimen. [5, s. 6–10.] Lisäksi tämän kaltaisissa sovelluksissa voidaan hyödyntää tietoverkon tarjoamia mahdollisuuksia jakaa tiedostoja ja kommunikoida muiden käyttäjien kanssa. Esimerkkejä tällaisista palveluista ovat esimerkiksi Google Docs (tekstinkäsittelyohjelma, jolla tämä raportti on kirjoitettu) ja 280 Slides (opinnäytetyöseminaarin diaesityksen tekoon käytetty sovellus).

Työn toteutusvaihtoehtoja mietittäessä itselläni ei ollut aiempaa kokemusta Ajax-sovellusten toteuttamisesta. Muutamaan luettuun lehtiartikkeliin ja verkkopalvelujen käyttökokemuksiin perustuen arvioitiin kuitenkin, että työn vaatima toiminnallisuus olisi mahdollista toteuttaa saavuttaen samalla riittävä käytettävyyden taso. Työ aloitettiinkin intensiivisellä tutustumisella aiheeseen.

Cranen mukaan Ajax-mallin toteuttamiseen tarvitaan neljä avainteknologiaa: JavaScript-kieli, Document Object Model ja XMLHttpRequest -rajapinnat sekä CSS-tyylikieli, jotka kaikki löytyvät useimmista selaimista esiasennettuna. [8, s. 32–34.] Seuraavaksi käydään lyhyesti läpi näiden roolit tyypillisen Ajax-verkkosovelluksen toteutuksessa.

JavaScript

JavaScript on alun perin Netscapen Navigator 2.0 -selaintaan varten kehittämä komentosarjakieli, jonka pohjalta Microsoft ja myöhemmin European Computer Manufacturer's Association julkaisivat omat versionsa nimeltään JScript ja ECMAScript. Jälkimmäisestä tuli myöhemmin kansainvälinen standardi (ISO/IEC 16262), jonka yhdeksi kaupalliseksi toteutukseksi JavaScript katsotaan. Varsinaisen Java-ohjelmointikielen kanssa JavaScriptillä on yhteistä vain markkinointisyistä valittu nimi. [9, s. 295–296.]

JavaScript-kielessä käytetään dynaamista tyypitystä, minkä vuoksi muuttujien tyyppi tarkistetaan käännöksen sijaan vasta ohjelmaa ajettaessa. Kielen tarjoamat valmiit oliot, kuten esimerkiksi verkkosivuilla käytettävää lomaketta kuvaava `Form` tai sen sisällä olevia kontrollielementtejä kuvaavat `Button` ja `Checkbox`, helpottavat sivujen käsittelyä. [10, s. 296–299.] Yleisesti ottaen JavaScript mahdollistaa toiminnallisuuden rakentamisen selaimeen.

DOM

DOM (Document Object Model) on XML- tai HTML-dokumenttien käsittelyyn tarkoitettu ohjelmointirajapinta, jonka avulla selaimessa ajettavalla ohjelmointikielellä voidaan liikkua dokumentin sisällä ja tehdä sen rakenteeseen muutoksia. DOM-olio voidaan esittää puumaisena mallina. [11.] Netscape kehitti rajapinnan ensimmäisen version (DOM Level 0) samanaikaisesti JavaScriptin kanssa, mutta sittemmin siitä on tullut W3C:n ylläpitämä suositus [12].

Kutsumalla DOM-rajapinnan funktioita voidaan selainnäkymää muuttaa ohjelmallisesti muodostamatta yhteyttä palvelimeen.

XMLHttpRequest

XMLHttpRequest (XHR) on ohjelmointirajapinta selaimessa ajettavalle ohjelmointikielelle, jolla voidaan siirtää tietoa selaimen ja palvelimen välillä [13]. Se kehitettiin alun perin Microsoft Exchange Server 2000 -palvelimen verkkokäyttöliittymää, Outlook Web Accessia, varten [14]. Vaihtoehtoisesti tiedonsiirtoon voidaan käyttää myös vanhempaa `<iframe>` -elementtiä [8, s. 54–56].

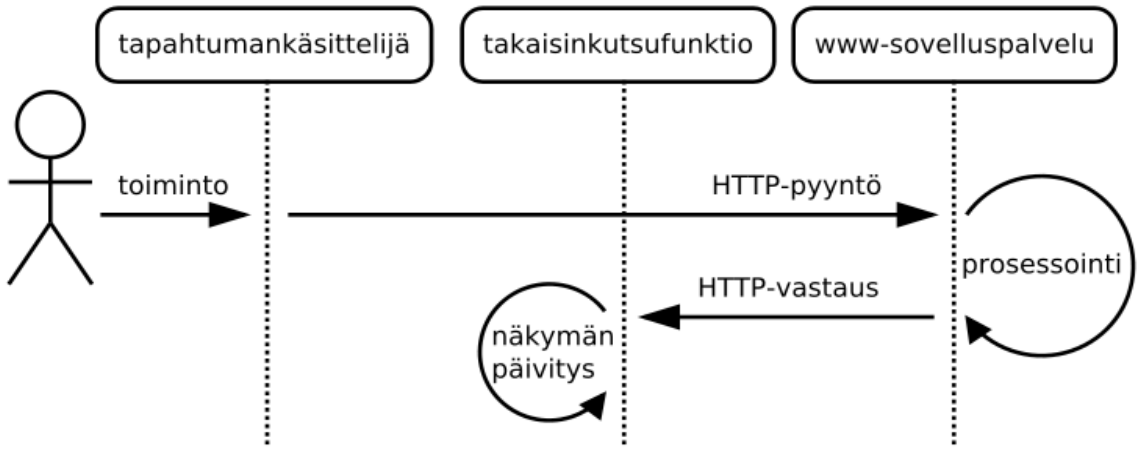
XHR-olion avulla JavaScript-kielinen ohjelma voi kommunikoida palvelimen kanssa.

CSS

CSS (Cascading Style Sheets) on tyylikieli, jolla voidaan määritellä XML- tai HTML-dokumenttien ulkoasu erillään niiden tietosisällöstä [15]. CSS:n käyttäminen ei ole välttämätöntä sovelluksen toiminnan kannalta, mutta sen tarjoamilla visuaalisilla tehosteilla voidaan parantaa sovelluksen käytettävyyttä.

Ajax-sovelluksen toiminta

Tyypillisen Ajax-sovelluksen alustaminen tapahtuu lataamalla selaimeen jokin sivu, jolloin yleensä myös asetetaan sovelluksen kontrollielementeille JavaScript-kieliset tapahtumankäsittelijät. Käyttäjän suorittaessa jonkin toiminnon tapahtumankäsittelijä lähettää palvelimelle XHR-oliota käyttäen pyynnön, jonka mukana kuljetetaan myös takaisinkutsufunktion (callback function) nimi. Käsiteltyään pyynnön palvelin lähettää takaisin vastauksen, joka puolestaan käynnistää takaisinkutsufunktion suorittamisen (kaavio 1). [16, s.15–16.]



Kaavio 1: Tapahtumien kulku tyypillisessä Ajax-sovelluksessa [16, s.16].

Mikäli XHR-olion suorittama pyyntö määritellään tapahtumaan asynkronisesti, tapahtumankäsittelijä ei jää odottamaan vastausta palvelimelta. Käyttäjälle voidaan tässä vaiheessa antaa jokin visuaalinen vihje siitä, että toiminnon suoritus on vielä kesken. XHR-olion tilasta pitää kirjaa `readyState` -jäsenmuuttuja, jossa tapahtuva muutos laukaisee tapahtuman. Tälle asetetun tapahtumankäsittelijän sisällä voidaan edelleen käsitellä palvelimelta saatu vastaus. Tämä antaa käyttäjälle mahdollisuuden jatkaa työskentelyä välittömästi suoritettuaan jonkin toiminnon, mutta saattaa aiheuttaa myös ristiriitaisia tilanteita. [16, s. 94–95.]

Selaimessa ajettavan käyttöliittymän lisäksi myös palvelimelle pitää luoda sovellus, joka vastaa sille lähetettyihin pyyntöihin.

2.4 Palvelimen ohjelmointi

PHP

PHP (PHP: Hypertext Preprocessor) sai alkunsa vuonna 1995 erään ohjelmistokehittäjän Perl-kielellä kehittämistä työkaluista, jotka oli alun perin tarkoitettu verkossa olevan ansioluettelon lukijamäärien tarkkailuun. Seuraavissa versioissa toteutuskieleksi vaihdettiin C ja innokkaan käyttäjäkunnan avustamana ohjelmiston suorituskykyä sekä laajennettavuutta parannettiin. Viidentoista vuoden kehitystyön tuloksena PHP:stä kehittyi verkkosivujen kehitystyössä äärimmäisen suosittu komentosarjakieli, joka on Netcraftin tutkimuksen mukaan asennettu jo yli 20 miljoonalle verkossa olevalle koneelle. PHP on minimalistinen ja käytännöllinen kieli, jonka luokkakirjastosta löytyy tuki suurelle joukolle erilaisia teknologioita. [17, s. 1–7; 18; 19.]

Aiemmassa verkkosivujen kehitystyössä saatujen kokemusten perusteella PHP-kielen valinta työn toteutukseen oli lähes itsestäänselvyys. PHP-tuki löytyy käytännössä jokaisen Internet-palveluntarjoajan valikoimista ilman erillistä korvausta ja kehitystyö on nopeaa. Eritoten projektin valmistumista helpottivat käyttövalmiit istunnonhallinta- ja SMTP-ominaisuudet sekä XML-dokumenttien käsittelyyn tarkoitetut kirjastot.

Ajax-sovelluskehykset

Ajax-arkkitehtuurin nopean yleistymisen myötä ajankohtaiseksi ovat tulleet perinteisessä sovelluskehityksessä käytetyt käsitteet, suunnittelumallit ja ohjelmistokehykset. Suunnittelumallilla (design pattern) tarkoitetaan kuvausta jostain toistettavissa olevasta tavasta tietyn ongelman ratkaisemiseksi ja ohjelmistokehyksellä (software framework) puolestaan uudelleenkäytettävää ohjelmarunkoa, josta voidaan tehdä valmis ohjelmisto täydentämällä tai korvaamalla halutut osat. [16, s. 71–72; 20.]

Yksi oleellisimmista Ajax-ohjelmistokehyksen tarjoamista ominaisuuksista on tehdasmetodi (factory method), joka abstrahoi XHR-olion käsittelyn. Tämä on tärkeää, koska olion luominen tapahtuu eri selainversioissa hieman eri tavoilla [5, s. 284–285]. Monet ohjelmistokehyksistä keskittyvät pääasiassa näyttävyyden ja käytettävyyden

luomiseen JavaScript-kielen keinoin. Tutkimuksissa suosituimpia kehyksiä ovat olleet Prototype, script.acu.lous, Dojo Toolkit ja jQuery. Palvelimen ohjelmoinnissa käytetyimpiä ympäristöjä ovat PHP, Java ja .NET. [21; 22.]

Työssä käytettäväksi valittiin avoimen lähdekoodin xajax-luokkakirjasto, jolla voidaan linkittää sivustolla olevat JavaScript-kieliset tapahtumankäsittelijät suoraan palvelimella sijaitseviin PHP-funktioihin. Tapahtuman sattuessa luodaan automaattisesti `xajaxResponse` -objekti, jonka jäsenfunktioilla voidaan esimerkiksi muokata halutun elementin sisältöä viittaamalla sen id-määritteeseen. Jäsenfunktiolla (myös *viesti* tai *metodi*) tarkoitetaan oliokielissä luokan funktiota, jolla pyydetään luokan ilmentymää suorittamaan jokin sille ominainen operaatio [10, s. 190]. Lisäksi kirjasto tarjoaa helpottavia toimintoja esimerkiksi ponnahdusikkunalla tehtävään varmennukseen tai HTML-lomakkeella annettujen syötteiden lähettämiseen. [23.]

Kirjastoa käyttämällä voitiin minimoida mahdollisesti hyvinkin työläs ohjelmointityö ennestään melko tuntemattomalla JavaScript-kielellä ja keskittyä ydintoiminnallisuuden toteuttamiseen palvelinpäähän. Tulevaisuudessa voitaisiin xajax-kirjaston rinnalle ottaa käyttöön jokin käytettävyyttä lisäävä ohjelmistokehys, jolla voitaisiin luoda esimerkiksi raahaustoiminto muokkaimeen.

Esimerkissä 1 on toteutettu yksinkertainen xajax-sovellus.

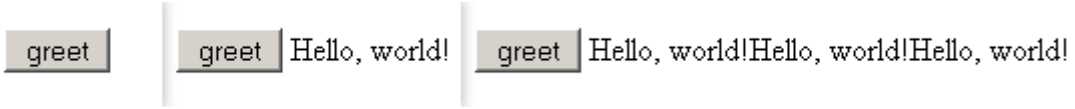
```
 1 <?php require_once("xajax_core/xajax.inc.php");
 2 $xajax = new xajax();
 3 $xajax->registerFunction('greet');
 4 $xajax->processRequest();
 5 function greet($args) {
 6   $response = new xajaxResponse();
 7   $response->append('results', 'innerHTML', $args);
 8   return $response;
 9 }
10 echo "<?xml version=\"1.0\"?>\n"; ?>
11 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"⏎
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
12 <html xmlns="http://www.w3.org/1999/xhtml">
13   <head>
```

```
14      <title/>
15      <meta http-equiv="Content-Type"⏎
          content="application/xhtml+xml; charset=utf-8" />
16      <?php $xajax->printJavascript(); ?>
17    </head>
18    <body>
19      <form action="">
20        <div id="results">
21          <input type="button" value="greet"⏎
              onclick="xajax_greet('Hello, world!');"/>
22        </div>
23      </form>
24    </body>
25 </html>
```

Esimerkki 1: Hei, maailma -ohjelman toteutus xajax-luokkakirjastoa apuna käyttäen.

Riveillä 2-4 luodaan `xajax` -olio, rekisteröidään sen tietoon käytettävät PHP-funktiot sekä käsitellään vastaanotettu pyyntö. Riveillä 5-9 määritellyn funktion `greet` sisällä luodaan `xajaxResponse` -tyyppinen vastausolio, jonka avulla selainnäkymää voidaan käsitellä. Käyttämällä olion jäsenfunktiota `append` lisätään tunnisteella `results` yksilöidyn elementin sisällön jatkeeksi parametrina annettu merkkijono.

Rivillä 16 luodaan dokumentin otsikkoelementin sisälle tarvittava JavaScript-kielinen apufunktio, jonka nimen etuliitteeksi tulee "`xajax_`". Lopulta rivillä 21 lisätään painikkeelle tapahtumankäsittelijä, joka kutsuu edellä mainittua apufunktiota antaen parametrinä tervehdystekstin. Kuvassa 2 on esitetty sovelluksen toiminta.



Kuva 2: Esimerkkisovelluksen piirtämä näkymä alustettuna sekä yhden ja kolmen painalluksen jälkeen.

Seuraavaksi käydään läpi muutamia verkkopalvelujen ohjelmoinnissa standardin asemaan nousseita teknologioita, joita tässäkin työssä hyödynnettiin.

2.5 Verkko-ohjelmoinnin standardeja ja suosituksia

XML

XML (Extensible Markup Language) on SGML-kielen (Standard Generalized Markup Language) rajoitetumpi muoto, joka koostuu merkkitiedosta (character data) ja sen jäsentelyyn käytetystä merkkauksesta (markup). XML-kielen kehitti vuonna 1996 XML Working Group, ja nykyisin kielen määritelmää ylläpitää W3C. [24] Kielen avulla voidaan helposti luoda uusia tai laajentaa jo olemassa olevia merkintäkieliä vastaamaan omia tarpeita. [25.]

Opinnäytetyössä XML 1.0 katsottiin sopivaksi työkaluksi tiedon varastoimiseen, koska se tarjoaa alustariippumattoman tavan kuvailla ja käsitellä rakenteellista tietoa. Tarjouspyyntöpalvelussa käytettävästä tietorakenteesta ei tässä vaiheessa ollut mitään tarkkoja suunnitelmia, mutta sen tulisi mahdollistaa esimerkiksi kysymysten jakaminen pienempiin elementteihin ja toisaalta kysymysten jäsentely suuremmiksi kokonaisuuksiksi.

Toisaalta XML-kuvauskielen avulla on helppo sarjallistaa monimutkainenkin puurakenne levylle, ja palvelun asetukset päätettiinkin alustavasti tallentaa palvelimella sijaitseviin tiedostoihin. Sekä tallennettava tietomäärä että luku- ja kirjoitusoperaatioiden määrä tulisivat olemaan niin pieniä, ettei relaatiotietokannan tyyppisellä järeällä ratkaisulla saavutettaisi mitään merkittävää etua. Tehtävää käyttöliittymää silmällä pitäen XML-perustaisella kielellä määritellystä tiedosta olisi myös helppo luoda näkymiä XSLT-muunnoskielen avulla. Lisäksi kieli oli entuudestaan itselleni tuttu.

XML-sovelluksen määrittelyssä voidaan käyttää joko rakennemäärittelyä (Document Type Definition, DTD) tai skeemaa. Näistä ensimmäinen keskittyy dokumentin kielioppiin ja jälkimmäisellä taas voidaan lisäksi määritellä elementtien välissä olevan tiedon tyyppi ja sisältö. [14; 25, s.195.] Työssä tyydyttiin tekemään rakennemäärittelyt palvelun tietojen varastointiin käytettäville XML-dokumenteille. Esimerkissä 2 on esitetty palvelun mittayksiköt sisältävä dokumentti.

```
 1 <?xml version="1.0" encoding="UTF-8"?>
 2 <!DOCTYPE units SYSTEM "http://ventur.info/dtd/units.dtd">
 3 <units>
 4   <unit id="0" name="lämpötila" symbol="ºC">
 5     <secondary id="0" symbol="ºF" conversion="(VALUE-32)*5/9"/>
 6   </unit>
 7   <unit id="1" name="tilavuusvirta" symbol="m³/h">
 8     <secondary id="0" symbol="m³/s" conversion="VALUE/3600"/>
 9     <secondary id="1" symbol="dm³/min" conversion="VALUE/1000/60"/>
10     <secondary id="2" symbol="dm³/s" conversion="VALUE/1000/3600"/>
11   </unit>
12   <unit id="2" name="massavirta" symbol="kg/s">
13     <secondary id="0" symbol="kg/h" conversion="VALUE*3600"/>
14   </unit>
15   <unit id="3" name="paine" symbol="Pa">
16     <secondary id="0" symbol="kPa" conversion="VALUE*1000"/>
17     <secondary id="1" symbol="mm H₂O"⏎
         conversion="VALUE/0.101971621298"/>
18     <secondary id="2" symbol="mbar" conversion="VALUE*100"/>
19   </unit>
20 </units>
```

Esimerkki 2: XML-kielinen dokumentti, joka määrittelee tarjouspyyntöpalvelussa käytettävät mittayksikköt.

XHTML

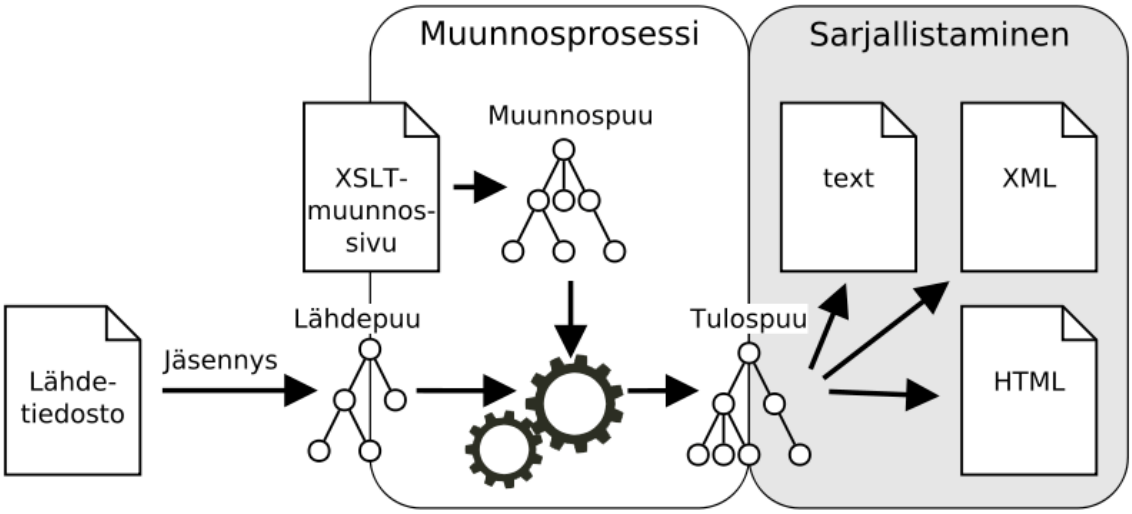
XHTML (Extensible HyperText Markup Language) on merkintäkieli, joka toteuttaa aiemman HTML 4 -kielen XML-sovelluksena [26]. Työssä valittiin käyttöliittymien esittämiseen XHTML 1.0 Strict -versio, josta on karsittu pois kaikki sivun ulkoasun muotoiluun liittyvät elementit.

XSL

XSL (eXtensible Stylesheet Language) on W3C:n suositusten muodostama kokonaisuus, joka koostuu kolmesta XML-dokumenttien muuntamiseen ja esitykseen tarkoitetusta osasta: XSLT (XSL Transformations), XPath (XML Path Language) sekä XSL-FO (XSL Formatting Objects) [27].

Aivan kuten XML on seuraaja SGML-kielelle, on jälkimmäisen muotoiluun tarkoitettu DSSSL (Document Style Semantics and Specification Language) toiminut edeltäjänä XSLT-kielelle [19, s. 31]. XSLT voidaan Kayn mukaan yleisesti määritellä kielenä, jolla voidaan muuntaa XML-kielisen dokumentin rakennetta. XML-kielen yleistymisen myötä kasvaa myös tarve muuntaa sillä esitettyä tietoa ihmiselle havainnollisempaan muotoon, kuten PDF- tai HTML-dokumentiksi. Toisaalta sovellusten välisessä kommunikaatiossa, johon XML parhaiten soveltuu, haluttu muoto voi olla esimerkiksi CSV, SQL, HTTP-viesti tai jokin toinen XML-pohjainen kieli. [28, s. 12-13.]

XSLT-muunnoksen vaiheet on esitetty kaaviossa 2.



Kaavio 2: XSLT-muunnos. Määritelmän mukaan XSLT-muuntimen täytyy suoriutua vain vaalean alueen sisällä olevista toiminnoista; muuntimen täytyy pystyä lukemaan muunnossivu ja muuntamaan lähdepuu tulospuuksi. Myös lähdetiedoston jäsentäminen ja tulospuun sarjallistaminen löytyvät silti useimmista valmiista tuotteista ja jälkimmäisen määrittelyyn löytyy XSLT-kielestä oma elementti, `<xsl:output>`. [28, s. 53-55].

Kay määrittelee XSLT-kielen määräävimmiksi ominaisuuksiksi XML-pohjaisuuden, sivuvaikutuksettomuuden ja rakenteen, jossa käytetään hahmohakuun (pattern matching) perustuvia malleja (template). Keskimmäisellä viitataan siihen, että kielellä ohjelmoidut funktiot eivät suoritettaessa tee muutoksia ympäristöönsä; ne voidaan ajaa useita kertoja eri järjestyksessä ja lopputulos on aina sama. Tästä ominaisuudesta johtuen XSLT-kielessä muuttujien arvoa ei voi asetuksen jälkeen muuttaa.

Kolmas kohta tarkoittaa, että muunnossivun mallit voivat olla lähdedokumentista riippumattomassa järjestyksessä ja yksittäisellä mallilla kuvataan jollekin lähdedokumentin elementille tehtävät toimenpiteet. [28, s. 36–39.] Kohteen määrittelevä sääntö ilmaistaan hahmona (pattern), joka on XPath-lausekkeen rajoitetumpi muoto [9, s. 429–431].

XPath kehitettiin osana XSLT-kieltä, mutta sitä voidaan tietyin rajoituksin käyttää myös itsenäisesti [9, s. 26]. Tärkein XPath-lausekkeen tyyppi on polkulauseke, josta kieli on saanut nimensä. Polkulauseke muodostetaan vinoviivoilla erotelluista askelista, jotka koostuvat akselimäärittelystä, solmutestistä ja mahdollisesta predikaatista. Akselimäärittelyn ja solmutestin välissä käytetään kahta kaksoispistettä ja predikaatin ympärillä hakasulkeita. Polkulauseke palauttaa aina solmujoukon, joka saattaa olla myös tyhjä. [28, s. 84–92.]

Esimerkiksi seuraava hahmo on kolmiaskelinen polkulauseke:

```
/child::units/child::unit[position() != last()]⏎
/child::secondary[contains(attribute::symbol, '/s')]
```

Ensimmäisen askeleen alussa oleva vinoviiva määrittää polun absoluuttiseksi, jolloin se määritellään dokumentin juuresta lähtien. Toisen askeleen predikaatti sulkee pois listan viimeisen mittayksikön ja kolmas askel ne muunnosyksiköt, joiden tunnuksessa ei esiinny sekuntia. Esimerkin 2 XML-dokumentista tähän hahmoon sopivia solmuja löytyisi siis kaksi kappaletta:

```
<secondary id="0" symbol="m³/s" conversion="VALUE/3600"/>
<secondary id="2" symbol="dm³/s" conversion="VALUE/1000/3600"/>
```

Polkulausekkeesta on olemassa myös lyhennetty ja hieman suppeammat toiminnot käsittävä syntaksi, jota käyttämällä hahmo voitaisiin esittää seuraavasti:

```
/units/unit[position() != last()]/secondary[contains(@symbol, '/s')]
```

Lisäksi XPath sisältää aritmeettisia ja merkkijonojen käsittelyyn tarkoitettuja funktioita.

Mallin sisällä kohdesolmun alta löytyvät solmut voidaan käydä paikallisesti läpi `<xsl:for-each>` ja `<xsl:value-of>` -elementtien avulla. Vaihtoehtoisesti voidaan käyttää `<xsl:apply-templates>` -elementtiä, jolloin lapsisolmut käsitellään omissa malleissaan. Tätä jakoa kutsutaan joissain yhteyksissä nimellä pull/push processing, ja molempia lähestymistapoja voidaan käyttää muunnossivuilla tilanteen mukaan. [28, s. 161.] Korkean tason ohjelmointikielien tapaan myös XSLT-muunnossivuilla voidaan ottaa käyttöön `<include>` - `<import>` -elementeillä valmiita mallikirjastoja ja yliajaa näiden sisältämiä malleja sitovuussääntöjen puitteissa [28, s. 98–105].
Seuraavassa esimerkissä haetaan esimerkin 2 dokumentista joitain haluttuja määritteitä ja tulostetaan ne muotoiltuna:

```
 1 <xsl:transform version="1.0"
 2   xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 3   <xsl:output method="text"/>
 4   <xsl:template match="/units">
 5     <xsl:for-each⏎
         select="unit[count(secondary) > 2]/secondary">
 6       <xsl:sort select="secondary/@symbol"/>
 7       <xsl:text>1</xsl:text>
 8       <xsl:value-of select="@symbol"/>
 9       <xsl:value-of select="concat(' = 1',⏎
           substring-after(@conversion, 'VALUE'),⏎
           '&#x20;', ../@symbol, '&#xA;')"/>
10     </xsl:for-each>
11   </xsl:template>
12 </xsl:transform>
```

Rivillä 1 on määritelty XSLT-muunnoksen versio ja rivillä 3 tuloksena sarjallistettavan dokumentin tyyppi. Malli rivillä 4 sopii dokumentin juurielementtiin, ja niinpä tämä malli suoritetaankin vain kerran. Rivin 5 `<for-each>` -elementin sisältö suoritetaan jokaiselle select-määritteeseen sopivalle solmulle, eli tässä tapauksessa muunnosyksiköille, joita on kolme tai useampia. Rivi 6 järjestelee ne tunnusmääritteen mukaiseen aakkosjärjestykseen. Lopulta riveillä 7–9 koostetaan merkkijono ja siirretään se tulospuuhun, josta muodostuu seuraavanlainen:

```
1m³/s = 1/3600 m³/h
1dm³/min = 1/1000/60 m³/h
1dm³/s = 1/1000/3600 m³/h
1kPa = 1*1000 Pa
1mm H₂O = 1/0.101971621298 Pa
1mbar = 1*100 Pa
```

Työssä valittiin käytettäväksi XSLT-kielestä versio 1.0, koska käytetty libxslt-kirjasto ei tue uudempaa versiota eikä muita kirjastoja saa kovin helposti otettua käyttöön PHP-ympäristöön. XPath-kielen versio määräytyy valitun XSLT-kielen version mukaiseksi. Kieliä käytettiin luomaan palvelun asetustietojen pohjalta osia selainnäkymästä.

XSL-FO on tarkoitettu muunnoksen tuloksena syntyneen dokumentin muotoiluun korkeatasoiseksi julkaisuksi näytölle, paperille tai jopa puheeksi [28, s. 25]. Tavallisen selaimella katseltavan verkkosivun viimeistelyyn se on kuitenkin tarpeettoman hienosyinen, ja siksipä sitä ei tässä työssä käsitellä tarkemmin.

UTF-8

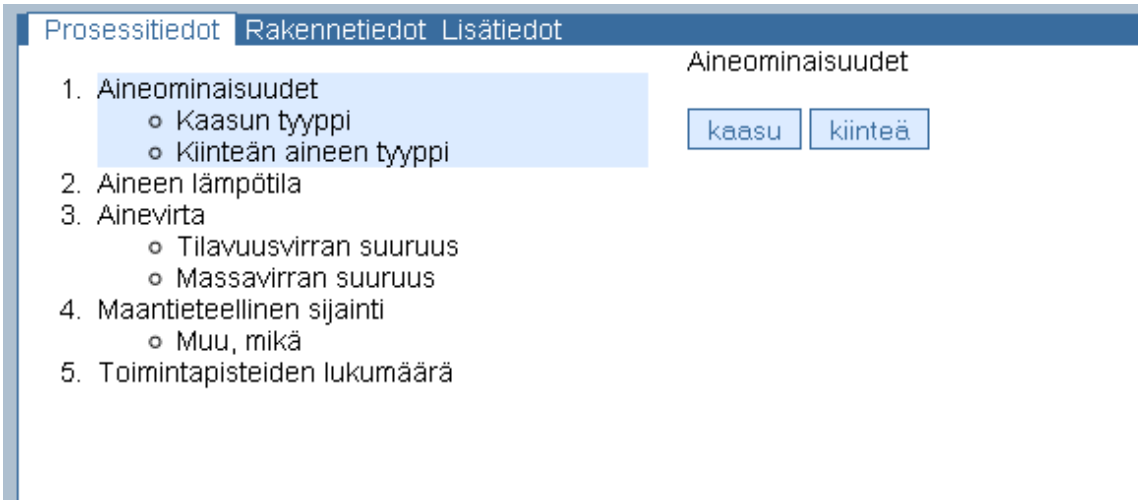
UTF-8 (Unicode Transformation Format) on koodaustapa, jolla voidaan määrittää koko Unicode-standardin määrittelemä merkistö käyttämällä 1–4 tavua yhtä merkkiä kohti. Jos tavun merkitsevin bitti on pois päältä, merkin tulkitaan olevan ASCII-koodattu. Näin ollen UTF-8 on alaspäin yhteensopiva ASCII-standardin kanssa, ja se onkin valittu XML-dokumenttien oletuskoodaustavaksi [29, s. 65]. Siitä tuli Googlen mukaan myös yleisin verkkosivustojen koodaustapa joulukuussa 2007 [30]. Työssä se valittiin palvelun tietovarastona käytettyjen XML-dokumenttien koodaustavaksi, koska skandinaavisten kirjainten lisäksi saatiin näppärästi ja melko luotettavasti selaimessa näkymään esimerkiksi mittayksiköiden yläindeksit.

2.6 Demonstraatioversion rakentaminen

Ideointivaiheessa mietittiin yhtenä ratkaisuvaihtoehtona tarjouspyyntökyselyn muodostamista dynaamisesti. Tällöin tarjouspyyntöpalveluun annetun vastauksen perusteella tehtäisiin päätös seuraavasta näytettävästä kysymyksestä. Kysymyksiä luotaisiin etukäteen riittävä valikoima ja niiden lisäksi tarvittaisiin jonkinlainen kyselyrunko, joka mahdollistaa vertailun ja ohjausrakenteet. Käyttäjälle sovellus näyttäytyisi ohjattuna toimintona, joka tekee tarkentavia lisäkysymyksiä tarvittavan puhaltimen ominaisuuksista.

Rungoksi suunniteltiin puuta, jonka solmuille voidaan vakioetenemisvaihtoehdon lisäksi määritellä vaihtoehtoisesti seuraavia solmuja, jos esimerkiksi numeerinen vastaus osuu yhdelle ennalta määritellyistä arvoalueista. Vaihtoehdon todettiin kuitenkin olevan liian monimutkainen toteuttaa, mahdollisesti vaarantavan projektin valmistumisen aikataulussa eikä se olisi välttämättä ollut edes yhtä havainnollinen palvelun käyttäjälle kuin "lineaarinen" lomake. Erityisen vaikeaa tässä lähestymistavassa olisi ollut toteuttaa muokkain, jolla kyselyn eteneminen ja rakenne olisi helppo havainnollistaa. Kyseenalaista oli myös se, tarjoaisiko sinänsä elegantti ratkaisu mitään liiketaloudellista hyötyä työn tilaajalle.

Palvelusta kuitenkin toteutettiin demonstraatioversio, jolla voidaan suorittaa alkeellisia, haarautuvia kyselyjä. Palvelun lomakenäkymä on esitetty kuvassa 3.



Kuva 3: Tarjouspyyntölomake demonstraatioversiossa. Seuraava kysymys määräytyisi käyttäjän valitseman aineen olotilan mukaan ja valinta näkyisi myös vasemmalla olevassa kysymysluettelossa.

Demonstraatioversiosta saatiin lopputuloksena hyvä pohja ulkoasulle ja tuntumaa edessä odottaviin ongelmiin. Erilaiset tavat toteuttaa käytettävyyttä JavaScript-kielen avulla näytettävien piirtotasojen (layer) ja työkaluvihjeiden (tooltip) muodossa tulivat tutuiksi. Lopullisesta versiosta poiketen tarjouspyyntöpalvelu toteutettiin Ajax-arkkitehtuuria käyttäen, mikä antoi mahdollisuudet esimerkiksi vastausten nopeaan validointiin.

Pikaisen palaveroinnin perusteella päätettiin toteuttaa palvelu valmiiksi määritellyillä lomakkeilla, joiden suunnittelusta huolehti osana omaa opinnäytetyötään Pekka Huhtaniemi. Lomakkeet jakautuvat kahteen kategoriaan, joista ensimmäistä käytetään korvattaessa vanhaa puhallinta ja siitä löytyy vain yksi lomake. Toiseen kategoriaan, joka on tarkoitettu uuden puhaltimen valintaan, sisältyy seitsemän lomaketta jaoteltuina eri puhallintyyppien mukaan sekä lisäksi yksi lomake tilanteisiin, joissa tyyppiä ei tiedetä.

3 Lopullinen toteutus

3.1 Tietorakenteen viimeistely

Lomakkeiden valmistuttua voitiin määritellä niiden kuvaukseen tarvittava tietorakenne, josta muodostui seuraavanlainen:

Lomakkeet jakautuvat kategorioihin ja niille voidaan asettaa otsikko. Kysymykset esitetään lomakkeilla numeroituna.

Jokaiselle esitettävälle kysymykselle voidaan asettaa otsikko, erillinen ohjeistusteksti sekä yksi tai useampia alikysymyksiä, jotka jälkimmäisessä tapauksessa luetteloidaan kirjaimilla. Myös alikysymyksille voidaan asettaa edellä mainitut ominaisuudet, ja niiden alle voidaan sisällyttää yksi tai useampia valintoja. Valinnat ovat toisensa poissulkevia, ja ne esitetään lomakkeella valintapainikkeilla eriteltyinä, mikäli niitä on useita.

Valintojen alle voidaan rakentaa varsinainen kysymyssisältö käyttämällä neljää erityyppistä elementtiä: merkkijonoa, tekstikenttää, tekstialuetta sekä pudotusvalikkoa.

Merkkijonoon sisältyy haluttu määrä merkkejä.

Tekstikentälle voidaan asettaa pituus sekä mittayksikkö, josta kerrotaan tarkemmin luvussa 3.2.4.

Tekstialueelle voidaan asettaa rivien määrä ja haluttaessa otsikko.

Pudotusvalikkoon voidaan sisällyttää haluttu määrä vaihtoehtoja.

Alusta lähtien oli selvää, että muokkain tulisi olemaan työn monimutkaisin ja työläin osuus, ja siksi sen kehitystyö aloitettiin pikaisesti tietorakenteen selkiydyttyä. Sovelluksen tuli esittää lomakkeet havainnollisesti ja mahdollistaa niiden muokkaus vaivattomasti. Kehitysvaiheessa mietittiin pitkään, tulisiko lomakkeet esittää muokkaustilassa lopullisessa muodossaan vai olisiko alla piilevän tietorakenteen

havainnollistaminen tärkeämpää. Lopulta päätettiin toteuttaa erillinen esikatselunäkymä, jossa lomakkeen julkaisukelpoisuus voitaisiin varmistaa ennen käyttöönottoa.

Pääpiirteissään sovellus tulisi olemaan erikoistunut XML-muokkain, jossa muutokset tietorakenteeseen tehtäisiin käsittelemällä sitä libxml2-kirjaston tarjoamilla metodeilla. Muokkaimen näkymä puolestaan tuotettaisiin kulloiseenkin tilanteeseen sopivalla XSLT-muunnoksella (tai useilla muunnoksilla), joihin käytettäisiin libxslt-kirjastoa. Seuraavissa luvuissa kerrotaan yksityiskohtaisemmin sovelluksen toteutuksesta.

3.2 Muokkaimen toiminta

3.2.1 Alustus

Muokkaimen käynnistyessä selaimeen piirretään alustussivu, jolta löytyy XHTML-dokumentin oikeellisuuden kannalta välttämätön perusrakenne. Dokumentin `<head>`-elementtiin kirjoitetaan alustuksen yhteydessä tapahtumankäsittelyn mahdollistavat JavaScript-kieliset apufunktiot. Tapahtumankäsittelijälle annetaan parametrina toiminnon tyypin kertova merkkijono, jonka perusteella valitaan sopiva apufunktio, sekä tarvittaessa toiminnon kohdetta ilmaiseva merkkijono. Joissain toiminnoissa asetetaan useita yksittäisiä arvoja kerralla, jolloin jälkimmäinen parametri on merkkijonoista koostuva taulukko. Tällainen joustava käyttö on mahdollista johtuen sekä PHP- että JavaScript -kielien heikosta tyypityksestä.

Sivun latauduttua sen `<body>` -elementissä sijaitseva automaattisesti ajettava `onLoad`-funktio suorittaa `initEditor` -tyyppisen toiminnon, jonka seurauksena istuntomuuttujiin ladataan palvelun XML-muodossa säilytettävät asetustiedostot ja tarkistetaan, onko palveluun kirjauduttu sisään. Jos näin ei ole, piirretään kirjautumista varten tarvittavat tekstikentät ja painike. Sisäänkirjautumisyrityksen yhteydessä tarkistetaan, löytyykö käyttäjätunnus-salasanaparia palvelun asetuksista. Mikäli sellainen löytyy, käyttäjän tunnus kirjoitetaan istuntomuuttujaan ja myös palvelun tilasta kirjaa pitävää tiedostoon. Tällöin voidaan varoittaa palveluun samanaikaisesti sisäänkirjautuvaa käyttäjää mahdollisesta ristiriitatilanteesta.

Onnistuneen sisäänkirjautumisen jälkeen sivun runkoon lisätään tunnisteilla yksilöidyt `<div>` -elementit, jotka on tarkoitettu muokkaimen neljälle alueelle: päävalikolle, muokkaustilalle, työkaluvalikolle sekä työkalutilalle. Tästä hetkestä uloskirjautumiseen asti sivun rakenne ei enää muutu, vaan kaikki muutokset tapahtuvat näiden elementtien sisällä. Lopuksi käynnistetään aktiivisen osan valitseva toiminto `select`, jolle annetaan parametrina tietovaraston alin taso. Tällöin lomakekategoriat piirtyvät muokkaustilaan ja muokkain on valmis käytettäväksi.

3.2.2 Muokkaimen ulkoasu ja asettelu

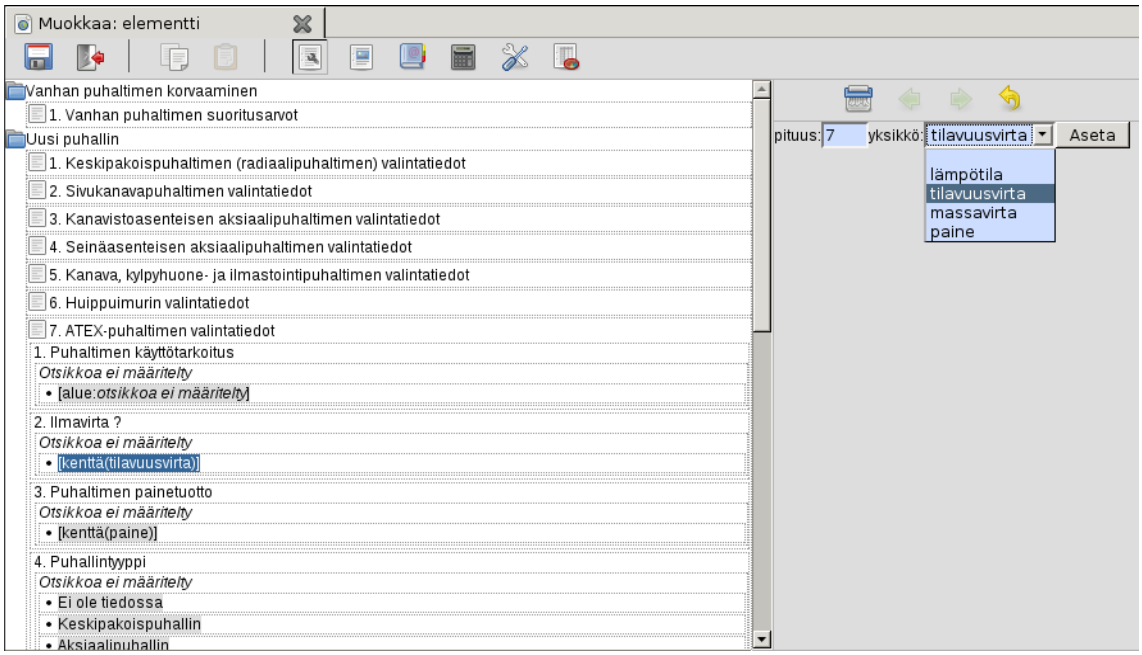
Päävalikon vasemmassa reunassa olevilla kahdella painikkeella voidaan tallentaa tehdyt muutokset palvelimelle ja kirjautua ulos muokkaimesta. Koska sovellus ei pidä kirjaa palveluun tehdyistä muutoksista, käyttäjälle esitetään ponnahdusikkunalla varmentava kysymys tallennettaessa tai poistuttaessa. Seuraavilla kahdella painikkeella voidaan kopioida lomakkeen osia ja liittää niitä toisaalle. Valikon lopusta löytyvät painikkeet ovat puolestaan näkymän valitsemista varten.

Muokkaimessa käytettävät kuvakkeet ovat peräisin Tango Desktop Project -hankkeesta, jonka tarkoituksena on yhtenäistää vapaiden ja avoimien ohjelmistojen tarjoama käyttäjäkokemus. Tyyli- ja nimeämisohjeiden sekä väripaletin lisäksi hanke on tuottanut kuvakepaketin, joka on vapaassa levityksessä (public domain). [31.]

Muokkaimeen luotiin palvelun eri osa-alueiden muokkaamista varten kuusi erilaista näkymää. Oletusvaihtoehto on lomakenäkymä, jossa tarjouspyyntölomakkeiden suunnittelu tapahtuu. Esikatselunäkymässä voidaan tarkastella aktiivisena olevaa lomaketta lopullisessa esitysmuodossa. Käyttäjänäkymä on tarjouspyyntöpalvelun käyttäjien hallinnointia varten ja yksikkönäkymässä voidaan muokata lomakkeilla käytettäviä mittayksikköjä. Asetusnäkymässä voidaan hallinnoida palvelun yleisiä asetuksia ja tarjouspyyntönäkymä on tarkoitettu palvelun arkistoimien tarjouspyyntöjen selaamiseen.

Vasemmassa laidassa sijaitsevaa muokkausaluetta käytetään useimmissa näkymissä muokattavan tiedon esittämiseen jäsennellyssä muodossa ja sen sisältä voidaan valita jokin haluttu osa aktiiviseksi. Oikean reunan työkaluvalikosta voidaan suorittaa

perustoimintoja ja sen alla olevalta työkalualueelta löytyvät kulloinkin aktiivisen osan tarkempaan muokkaamiseen tarvittavat välineet.



Kuva 4: Muokkaimen lomakenäkymä. Lomakkeen aktiivinen osa näkyy muokkaustilassa tummalla taustalla ja sen muokkausvaihtoehdot ovat esillä työkalutilassa. Muokkainsivun `<title>` -elementin sisältö asetetaan ilmoittamaan aktiivisen osan tyypin, joka tässä tapauksessa on elementti.

3.2.3 Lomakenäkymän ja esikatselunäkymän toiminnot

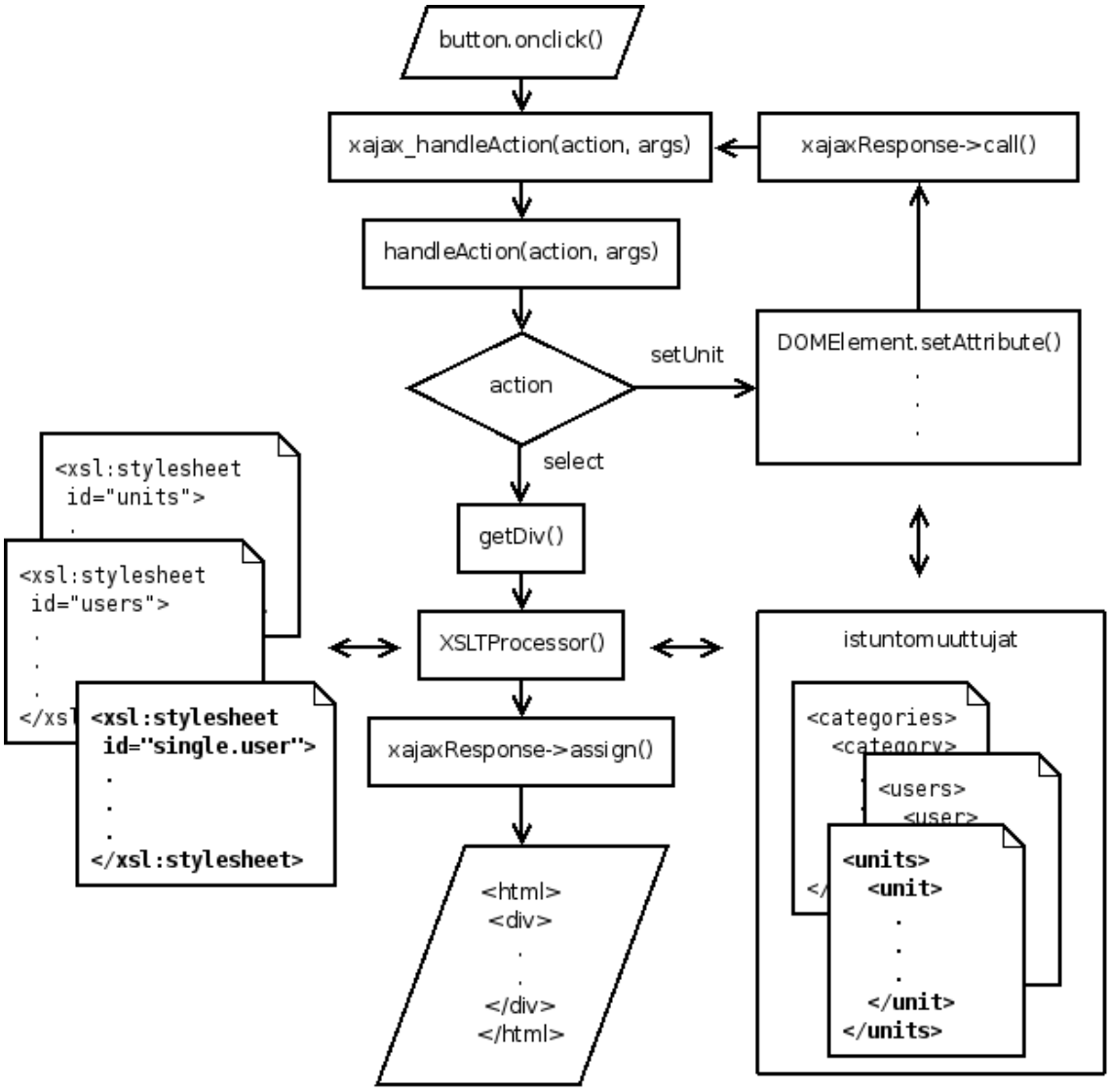
Lomakenäkymässä vain aktiivisen lomakkeen osat piirretään, muista näytetään lomakkeen nimi. Aktiiviselta lomakkeelta voidaan valita haluttu osa muokattavaksi, jolloin työkaluvalikkoon piirretään painikkeet mahdollisille perustoiminnoille. Jos jotain toimintoa ei voida tällä hetkellä suorittaa, se esitetään häivytettynä ja painikkeelle ei kytketä tapahtumankäsittelijää. Työkalutilaan piirretään aktiivisen osan muokkaamiseen tarvittavat yksilölliset hallintavälineet, esimerkiksi lomakkeen ollessa aktiivisena tilaan piirretään tekstikenttä lomakkeen nimen asetusta varten ja painike muutosten vahvistamiseksi.

Aktiivisen osan valinnan lisäksi perustoiminnoista löytyvät uuden elementin luonti, poisto sekä siirto eteen- tai taaksepäin, joita voidaan käyttää sellaisenaan lähes kaikelle palvelussa käytettävälle puurakenteiselle tiedolle. Näiden lisäksi luotiin muutama erikoistuneempi toiminto esimerkiksi palvelun yleisten asetusten muokkaamista varten, jolloin asetetaan kerralla joukolle kenttiä halutut arvot.

PHP-ympäristö tarjoaa valmiit toiminnot istuntomuuttujien käsittelyyn, joissa säilytetään muokkaimen tila kokonaisuudessaan:

- palvelun asetukset XML-muotoisina
- taulukko, joka sisältää aktiivisen elementin ja sen ylätasojen tunnisteet
- aktiivisen osan tyyppi ja osaa muokkaustilassa kuvaavan XHTML-elementin tunniste
- valittuna oleva näkymä
- sisäänkirjautunut käyttäjä
- leikepöydän sisältö.

Kaaviossa 3 on kuvattu muokkaimen toiminta, kun käyttäjä tekee muutoksia palvelussa käytettävään mittayksikköön ja vahvistaa ne "aseta" -painikkeella.



Kaavio 3: Muokkaimen toiminta esimerkkitapauksessa.

Muokkaimen painikkeeseen kytketty tapahtumankäsittelijä kutsuu xajax-luokkakirjaston tuottaman apufunktion välityksellä PHP-kielistä `handleAction`-funktiota. Koska ensimmäisenä parametrina on annettu merkkijono "`setUnit`", tapahtumankäsittelijän sisällä tehdään muiden parametrien määrittämät muutokset mittayksikköjen asetukset sisältävään istuntomuuttujaan. Muuttujaan talletetun DOM-olion käsittelyyn käytetään libxml2-kirjaston tarjoamia metodeja.

Aivan toiminnon käsittelyn sisältävän lohkon lopussa päivitetään näkymä tapahtumankäsittelijän alussa luodun `xajaxResponse` -olion `call` -metodin avulla, jolla voidaan kutsua JavaScript-kielisiä funktioita. Tässä tapauksessa tapahtumankäsittelijää kutsutaan uudelleen antaen parametrina toiminto "`select`", jolloin suoritetaan PHP-

kielinen funktio `getDiv`. Funktio lataa levyltä tilanteeseen sopivan muunnossivun ja muodostaa siitä DOM-olion. Seuraavaksi luodaan libxslt-luokkakirjaston sisältämä `XSLTProcessor`-olio, jonka avulla suoritetaan muunnos käyttäen lähdepuuna jotain istuntomuuttujista löytyvistä DOM-olioista. Lisäksi muuntimelle annetaan näkymän muodostamisessa välttämättömiä parametrejä, kuten esimerkiksi aktiiviseksi valitun yksikön tunniste.

Muunnoksen tuloksena muunnin palauttaa XHTML-muotoinen katkelman, joka asetetaan aktiivista osaa muokkaimen näkymässä vastaavan elementin sisällöksi `xajaxResponse` -olion `assign` -metodia käyttäen. Toiminto toistetaan kaikille päivitystä tarvitseville muokkaimen alueille.

Suorituskykyä ajatellen tällainen järjestely saattaa vaikuttaa palvelimen tarpeettomalta kuormittamiselta, koska joissain tilanteissa pieni muutos muokkaimen tilassa voi vaatia koko näkymän päivittämistä usealla peräkkäisellä XSLT-muunnoksella. Työn toteutusvaiheessa mietittiinkin vaihtoehtoista ratkaisutapaa, jossa JavaScript-kielen avulla huolehdittaisiin näkymän päivityksestä ja käyttäjän tekemät muutokset siirrettäisiin esimerkiksi aina näkymää vaihdettaessa tai tietyin aikavälein pysyvään tietorakenteeseen.

Koska muokkainta tulisi kuitenkin tulevaisuudessa käyttämään vain yksi asiakasyrityksen henkilö kerrallaan, palvelinresurssien säästelyä ei katsottu tarpeelliseksi. Valitussa toteutuksessa on myös eräs käytettävyyden kannalta hyvä puoli: esimerkiksi tietoliikenneyhteydestä johtuvissa häiriötilanteissa muokkain saadaan palautettua toimintakuntoon painamalla kerran selaimen virkistysnäppäintä tai käyttäjän eksyessä vahingossa pois muokkainsivulta takaisin pääsee navigaatiopainikkeilla.

Esikatselunäkymässä koko muokkaimen piirtoalue päävalikkoa lukuun ottamatta käytetään esittämään lomaketta sellaisena, kuin se näyttäytyy tarjouspyyntöä lähettävälle käyttäjälle. Näkymään siirtävä painike on toiminnassa vain, jos lomake tai lomakkeen osa on valittuna aktiiviseksi.

3.2.4 Muut näkymät

Käyttäjänäkymässä muokkaustilaan piirretään listamuotoisena kaikki tarjouspyyntöpalvelun käyttäjät ja työkalutilaan painike uuden käyttäjän lisäämiseksi. Aktiiviseksi valitun käyttäjän henkilö- ja yhteystietoja voi muokata työkalutilassa tai käyttäjätunnuksen voi poistaa.

Mittayksikkönäkymässä voidaan hallinnoida tarjouspyyntöpalvelussa käytettyjä yksikkömuunnoksia. Lomakkeella oleviin tekstikenttä-tyyppisiin elementteihin voidaan liittää mittayksikköominaisuus, jolloin kentän jälkeen piirretään pudotusvalikko, josta löytyy valitulle suureelle erilaisia vaihtoehtoisia mittayksikköjä. Palvelu huolehtii annetun arvon muuntamisesta ensisijaiseksi määriteltyyn yksikköön tarjouspyyntöä lähetettäessä. Muokkausnäkymässä voidaan luoda ja poistaa mittayksikköjä sekä muokata muunnoksessa käytettäviä laskukaavoja.

Asetusnäkymässä voidaan muokata lomakkeilla näytettäviä yrityksen yhteystietoja sekä lisätä ja poistaa sähköpostiosoitteita tarjouspyyntöjen vastaanottajalistalta.

Tarjouspyyntönäkymässä tehdyt tarjouspyynnöt esitetään saapumisjärjestyksen mukaan järjestettynä listana, jolla välitöntä yhteydenottoa vaatineiden tekemät pyynnöt ovat korostettuina. Valitsemalla jokin tarjouspyyntö listalta aktiiviseksi voidaan nähdä lomakkeen kysymykset ja lähettäjän vastaukset niihin.

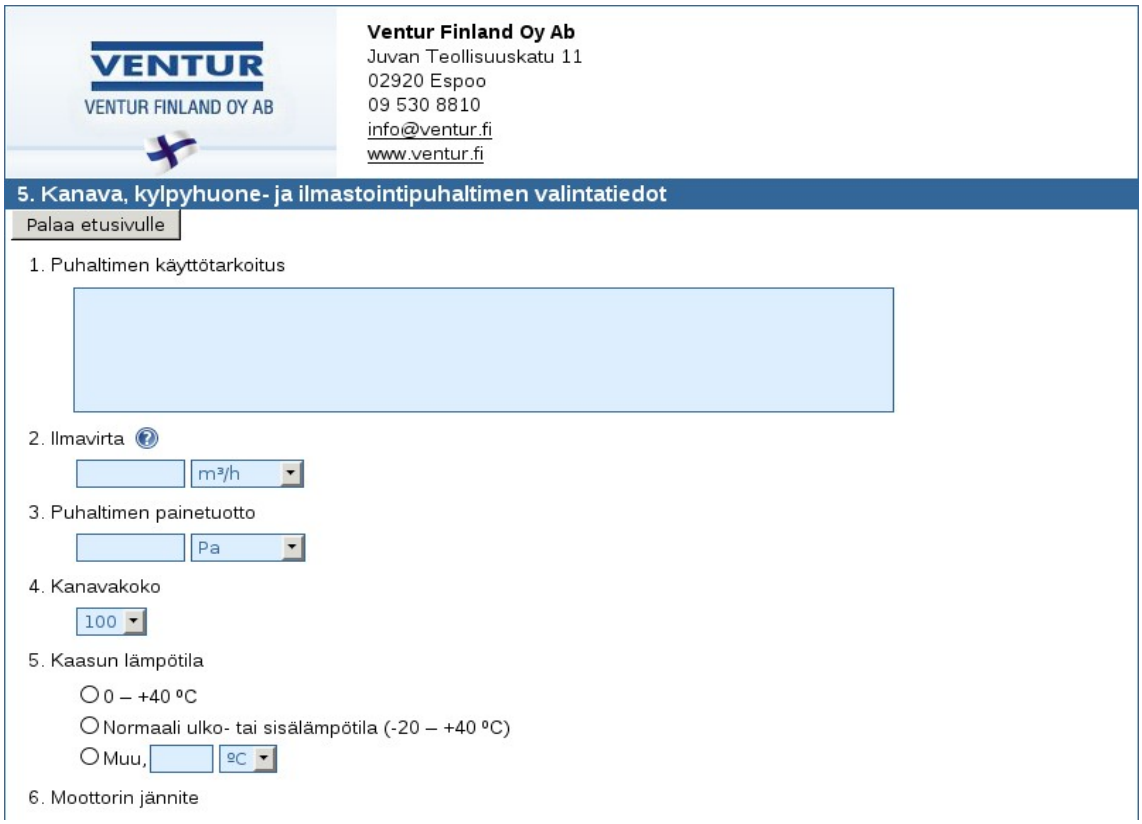
3.2.5 Viimeistely

Xajax-olio asetettiin toimimaan synkronisesti, koska muokkaimella tehtävät toiminnot muuttavat palvelun tietoja ja nopeasti perättäin tehtynä ne saattaisivat sekoittaa tapahtumankäsittelijän toiminnan. Muokkaimessa käytettävät XSLT-muunnossivut olivat kehitystyön tässä vaiheessa päässeet venähtämään melko pitkiksi ja muodostuneet rakenteeltaan vaikeasti luettaviksi. Sivut pilkottiin pienempiin, helpommin hallittaviin osiin ja samalla pyrittiin vaihtamaan pull-tyylinen solmujen käsittely push-tyyppiseksi.

3.3 Lomakepalvelu

Koska lomakkeen täyttäminen sekä lähetys eivät vaadi erityisen monimutkaista toiminnallisuutta, palvelu toteutettiin XHTML-sivuina ja käyttämällä GET-tyyppisiä HTTP-pyyntöjä lomakkeen valintaan sekä POST-tyyppisiä pyyntöjä sisäänkirjautumisessa ja lomakkeen lähetyksessä. Ajax-toteutuksessa käyttäjän tekemät virkistys- tai navigaatiotoiminnot saattaisivat sekoittaa kyselyn etenemisen.

Lomakepalvelun lopullinen ulkoasu on esitetty kuvassa 5.



Kuva 5: Valmiin palvelun lomakenäkymä.

Tarjouspyyntöpalvelun etusivulta kirjaudutaan sisään palveluun, jonka jälkeen käyttäjälle näytetään lista lomakkeista jaoteltuina kategorioihin. Valittu lomake piirretään näkymään, ja tarjouspyynnön tekijä voi täyttää ja lähettää sen, tyhjentää jo täytetyt kentät tai peruuttaa pyynnön ja palata takaisin etusivulle.

Lomakepalvelu muodostaa POST-pyynnön sisältämistä vastauksista lomakkeen kysymyksiin XML-puun, joka tallennetaan palvelun lokitiedostoon. Lisäksi puusta luodaan sähköposti lähetettäväksi asetuksissa määritellyille vastaanottajille tilaajayrityksen sisällä. Viesti lähetetään PHP-ympäristöstä löytyvän `mail` -funktion avulla SMTP-protokollaa käyttäen.

Palvelua viimeisteltäessä törmättiin myös yhteen PHP-luokkakirjaston ikävähköön ominaisuuteen: lähetettäessä liitetiedostoa, jonka koko ylittää palvelimen asetuksissa annetun rajan, koko POST-pyyntö jää tyhjäksi ja täytetyn lomakkeen tiedot katoavat. Tässä vaiheessa esimerkiksi JavaScript-kielellä toteutettu komponentti tiedostojen lataamiseen palvelimelle olisi ollut tarpeen, mutta aikataulussa ei enää ollut tilaa sellaisen toteuttamiseen.

4 Tulokset ja yhteenveto

Työn tuloksena syntyi jotakuinkin aikataulun puitteissa valmis verkkopalvelu, josta pahimmat ohjelmointivirheet korjattiin jälkeenpäin. Osana työtä kirjoitettiin myös lyhyt käyttöohje muokkaimelle. Palvelun siirtäminen tuotantopalvelimelle sujui pääasiassa ongelmitta, joskin postin lähettämiseen käytetty PHP-funktio käsitteli rivinvaihtoja eri tavalla kuin kehitysympäristössä, mikä aiheutti pientä päänvaivaa. Työn kirjoitusvaiheessa palvelun käyttöönotto on vielä kesken ja tilaajan palautetta odotellaan.

Tarjouspyyntöpalvelun jatkokehitystarpeita ajatellen tärkein yksittäinen asia olisi lomaketiedon tallentaminen tietokantaan. Tämä helpottaisi yhtäaikaisten kirjoitustilanteiden hallinnassa ja tekisi palvelun tuottamasta tiedosta liiketaloudellisessa mielessä käyttökelpoisempaa. Relaatiotietokannan ohella käyttökelpoinen ratkaisu saattaisi olla myös varsinainen XML-tietokanta, josta haut voisi suorittaa suoraan XPath- tai XQuery-kieltä käyttäen.

XSLT-muunnostiedostojen käyttäjälle tulostamat viestit tulisi eriyttää omiin tiedostoihin, jotta mahdollinen käännöstyö tulevaisuudessa olisi helppoa. Lisäksi sovellus tulisi osittaa helpommin ylläpidettäviksi moduuleiksi, missä oliopohjaisuus olisi varteenotettava mahdollisuus. Luonnollisesti myös tietoturvaan tulisi kiinnittää huomiota etenkin muokkaimen osalta.

Työn loppuvaiheessa harkittiin myös kaikkien tiedonkäsittelyoperaatioiden toteuttamista XSLT-muunnoksella. Tämä helpottaisi huomattavasti uusien toimintojen lisäämistä palveluun tulevaisuudessa, mutta aikataulun puitteissa menettelyn toimivuutta ei valitettavasti ehditty kokeilla.

Mahdollisesti tulevaisuudessa käyttökelpoinen ominaisuus voisi myös olla mahdollisuus tallentaa lomakkeen täytetyt kohdat palveluun. Tällöin tarjouspyynnön tekijä voisi tarvittaessa hankkia johonkin kysymykseen tarvittavaa lisätietoa ja jatkaa keskeneräisen tarjouspyyntölomakkeen täyttämistä. Harkitsemisen arvoinen voisi olla myös toiminto, jolla lomake voitaisiin merkitä passiiviseksi. Tällöin se ei näkyisi palvelussa mutta olisi käytettävissä esimerkiksi testausta varten.

Työn tiimoilta opituista asioista ylivoimaisesti tärkein oli XSLT-kieli, jonka käyttömahdollisuudet eivät rajoitu pelkästään esitysmuotoisen tiedon tuottamiseen, vaan sillä on paikkansa esimerkiksi integraatiotyössä. Kieltä opiskellessa heräsi myös kiinnostus funktionaalisen ohjelmoinnin periaatteisiin.

Ajax-tekniikkaan perehtyminen tullee myös osoittautumaan tarpeelliseksi. Verkkopalvelujen yhä yleistyessä niiden odotetaan myös tarjoavan lisää käytettävyyttä ja muistuttavan yhä enemmän työpöytäsovelluksia. Kynnys erillisen liitännäisen tai sovellusalustan asentamiseen tulee kuitenkin pysymään korkeana, koska samalla menetetään verkkopalvelun tärkein ominaisuus, saavutettavuus, lähes kaikkialla.

Arkipäiväisimmissä verkkopalveluissa Ajax-tekniikalla aikaansaatua käytettävyyttä pidettäneenkin jatkossa vaadittavana perustasona, ja ohjelmistokehityksen parissa toimivalle sen toimintaperiaatteen tuntemus on vähintäänkin hyödyllistä.

Lähteet

1 Puhaltimet. (WWW-dokumentti.) Ventur Finland Oy Ab. <www.ventur.fi/sites/fi/products/categories/36>. Luettu 7.11.2009.

2 Huhtaniemi, Pekka. Puhaltimen valintaprosessin tehostaminen tarjouspyyntövaiheessa. Insinöörityö. Metropolia Ammattikorkeakoulu, 2009.

3 Sinkkonen, Irmeli. Helppokäyttöisen verkkopalvelun suunnittelu. Helsinki: Tietosanoma, 2009.

4 Verkkopalvelu. (WWW-dokumentti.) TEPA – Sanastokeskus TSK:n termipankki. <www.tsk.fi/tepa>. Luettu 18.8.2009.

5 Poikonen, Mikko. Rikkaat Internet-sovellukset. Kandidaatintutkielma. Jyväskylän yliopisto, 2007.

6 Rapoza, Jim. RIA War Is Brewing. (WWW-dokumentti.) <etech.eweek.com/content/application_development/ria_war_is_brewing.html>. 11.5.2008. Luettu 25.10.2009

7 Garrett, Jesse James. Ajax: A New Approach to Web Applications. (WWW-dokumentti.) <www.adaptivepath.com/ideas/essays/archives/000385.php>. 18.2.2005. Luettu 19.8.2009.

8 Crane, Dave. Ajax In Action. Greenwich: Manning, cop. 2006.

9 Barron, David. World of Scripting Languages. Chichester: Wiley, 2000.

10 Harsu, Maarit. Ohjelmointikielet: periaatteet, käsitteet, valintaperusteet. Helsinki: Talentum, 2005.

11 Document Object Model (DOM) Level 2 Core Specification. (WWW-dokumentti.) W3C. <www.w3.org/TR/DOM-Level-2-Core/>. 13.11.2000. Luettu 20.8.2009.

12 Koch, Peter-Paul. The Document Object Model: an Introduction. (WWW-dokumentti.) <www.digital-web.com/articles/the_document_object_model >. 14.5.2001. Luettu 21.8.2009.

13 The XMLHttpRequest Object. (WWW-dokumentti.) W3C. <www.w3.org/TR/XMLHttpRequest/>. 15.4.2008. Luettu 19.8.2005.

14 Hopmann, Alex. The story of XMLHTTP. (WWW-dokumentti.) <www.alexhopmann.com/xmlhttp.htm>. 31.1.2007. Luettu 19.8.2009.

15 Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification. (WWW-dokumentti.) W3C. <www.w3.org/TR/CSS21/>. 23.4.2009. Luettu 20.8.2009.

16 Mahemoff, Michael. Ajax Design Patterns. Sebastopol: O'Reilly, 2006.

17 Beginning PHP and MySQL : From Novice to Professional. Gilmore, W. Jason. New York: Apress, 2008.

18 History of PHP. (WWW-dokumentti.) <www.php.net/manual/en/history.php.php>. Luettu 21.8.2009

19 Usage Stats for April 2007. (WWW-dokumentti.) <php.net/usage.php>. Luettu 20.8.2009.

20 Software framework. (WWW-dokumentti.) Wikipedia. <en.wikipedia.org/wiki/Software_framework/>. Luettu 21.8.2009.

21 2006 Survey Results. (WWW-dokumentti.) Ajaxian.com <ajaxian.com/archives/ajaxiancom-2006-survey-results/>. 23.9.2006. Luettu 21.8.2009.

22 Javascript framework usage among top websites. (WWW-dokumentti.) <royal.pingdom.com/?p=305>. 11.6.2008. Luettu 21.8.2009.

23 xajax PHP Class Library. (WWW-dokumentti.) <xajaxproject.org>. Luettu 20.8.2009.

24 Extensible Markup Language (XML) 1.0 (Fifth Edition). (WWW-dokumentti.) W3C. <www.w3.org/TR/xml/>. 26.11.2008. Luettu 22.8.2009.

25 Holzner, Steve. Real world XML. Indianapolis: New Riders, 2002.

26 XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition). (WWW-dokumentti.) W3C. <www.w3.org/TR/xhtml1/>. 1.8.2002. Luettu 22.8.2009.

27 The Extensible Stylesheet Language Family (XSL). (WWW-dokumentti.) W3C. <www.w3.org/Style/XSL>. Luettu 22.8.2009.

28 Kay, Michael. XSLT Programmer's Reference. Indianapolis: Wiley, 2003.

29 Haralambous, Yannis. Fonts & Encodings. Sebastopol: O'Reilly Media, 2007.

30 Davis, Mark. Moving to Unicode 5.1. (WWW-dokumentti.) <googleblog.blogspot.com/2008/05/moving-to-unicode-51.html> 5.5.2008. Luettu 19.8.2009.

31 Tango Desktop Project. (WWW-dokumentti.) <tango.freedesktop.org>. Luettu 7.11.2009.

32 Weaver James L., Gao Weiqi, Chin Stephen, Iverson Dean. Pro JavaFX Platform: Script, Desktop and Mobile RIA with Java Technology. Apress, 2009.