Laurea-ammattikorkeakoulu
Laurea Leppävaara

Tehokkaan Ajax-pohjaisen tarjoustyökalun toteuttaminen

Mikko Salo
Tietojenkäsittelyn koulutusohjelma
Opinnäytetyö
Lokakuu, 2008

Laurea-ammattikorkeakoulu
Laurea Leppävaara
Tietojenkäsittelyn koulutusohjelma
Tietojärjestelmien kehittäminen

TIIVISTELMÄ

Mikko Salo

Tehokkaan Ajax-pohjaisen tarjoustyökalun toteuttaminen

Vuosi 2008 Sivumäärä 47

Opinnäytetyössä tutkitaan Ajax-tekniikan käyttöä tehokkaan web-sovelluksen näkökulmasta. Tutkittua tietoa hyödynnetään tarjoustyökalun kehittämisessä. Tarjoustyökalu toteutettiin projektina Isolta Oy nimiselle yritykselle. Projektin tavoitteena oli kehittää kiinteistöalalla toimivalle asiakkaalle järjestelmä, jonka avulla he voivat tehdä tarjouksia vuokrattavista ja myytävistä toimitiloista ja ylläpitää toimitilarekisteriään, sekä välittää toimitiloja erilaisiin ulkopuolisiin palveluihin.

Projekti alkoi vuoden 2008 tammikuussa, jolloin järjestelmää alettiin suunnitella. Erityisehtona järjestelmässä oli se, että sen pitää toimia kiinteistöalalla toimivan asiakkaan käyttämän web-pohjaisen CRM-järjestelmän rinnalla. Tämä ei heidän vanhassa järjestelmässään ollut mahdollista. Järjestelmästä toivottiin myös web-pohjaista ja mahdollisimman tehokasta. Tällöin ajateltiin, että paljon puhuttu Ajax-tekniikka voisi tuoda asiakkaan tarvitsemaa tehokkuutta järjestelmän käyttöön. Tekniikka kun mahdollistaa työpöytäohjelmista tuttujen elementtien kehittämisen web-ympäristöön. Ajax-tekniikan toivottiin tuovan myös lisää nopeutta järjestelmän toimintaan.

Opinnäytetyössä käytettiin konstruktiivista tutkimusmenetelmää. Työssä tutkittiin Ajax-tekniikkaa käsittelevää kirjallisuutta, jonka avulla kerättiin tietoa siitä, mitä tekniikan avulla on aikaisemmin kehitetty ja mitä sillä kannattaa ja voi kehittää. Kerättyä tietoa hyödynnettiin tarjoustyökalun kehittämiseen.

Toteutuksen onnistumista mitattiin tekemällä valmiille sovellukselle nopeusmittaus. Nopeusmittauksen avulla tarkasteltiin sitä, kuinka paljon Ajax-tekniikan käyttö vaikuttaa järjestelmän nopeuteen verrattuna perinteisin menetelmin toteutettua web-järjestelmään. Samasta sovelluksesta tehtiin siis Ajax-toteutuksen lisäksi perinteisellä tavalla toimiva web-sovellus ja näiden suorituskyvyistä tehtiin vertailu. Lisäksi pohdittiin mitä taitoja Ajax-osaajalta vaaditaan, jotta tekniikkaa pystyisi hyödyntämään tehokkaasti.

Mittauksen tulokset osoittivat, että Ajax-pohjaisella tekniikalla saatiin huomattavia parannuksia järjestelmän suorituskykyyn ja nopeuteen. Tämä tarkoitti sitä, että sovelluksen latausajat lyhenivät, palvelinkuorma vähentyi ja sovelluksesta tuli nopeampi ja tehokkaampi käyttää.

Asiasanat Ajax, Web-sovellus, Tietokantasovellus, JavaScript

Laurea University of applied sciences
Laurea Leppävaara
Information Technology Programme
System Design and Programming

ABSTRACT

Mikko Salo

Implementation of Efficient Ajax-based offer tool

Year 2008 Pages 47

This thesis is a research paper on how to use the Ajax technique in an efficient web application. The research material was used for developing an offer tool. The offer tool was made as a project for a company named Isolta Ltd., based on a customer request. The purpose of this project was to develop for the customer in the real estate business a program which they could use when making offers for premises. They also needed a program that would maintain their register of premises and provide premises to different external services.

The planning of the system started in January 2008. The system had a special requirement: it had to work based on the web based CRM system that the customer in the real estate business uses, which was not possible with their old system. Another requirement was that it should be web based and as efficient as possible. It was decided to use the Ajax technique to bring the customer the efficiency they wanted. The technique could make it possible to use familiar elements from the desktop software for developing the web environment. The Ajax technique would also make the functionality of the program faster.

A structural method was used in the thesis work. Literature about the Ajax technique was examined and information was collected to understand what has been done before with this technique and what is worth doing, and how to develop it. A deliberation on the basis of the above was made, as well as an analysis of the skills an Ajax expert needs to be able to use the technique efficiently.

The success of the execution was measured by carrying out a speed measurement of the ready application. The speed measurement checked to what extent the Ajax technique affects the speed of the system compared to the traditional web system based methods. In addition to the Ajax application, a traditional web-system application was made and the performance capabilities of these two were compared. The results of the measurement showed that considerable improvements in the system performance and speed were made with the Ajax based technique. This means that application load times became shorter, the server load was reduced and the application became faster and more efficient to use.

Sisällys

1 JOHDANTO

Isolta Oy on vuonna 2003 perustettu tietotekniikka-alan yritys, jonka ydinliiketoimintaa ovat tietokantapohjaisten ratkaisujen kehittäminen ja Internet-palvelut. Työntekijöitä yrityksessä on 5 ja toiminnan tukena on myös sijoitusyritys, joka omistaa osan Isolta Oy:n osakekannasta.

Isolta Oy sai vuoden 2008 alussa yhteistyökumppanin kautta projektikseen suunnitella ja toteuttaa kiinteistöalalle soveltuvan tarjoustyökalun. Järjestelmä oli tarkoituksena liittää yhteistyökumppanin CRM-järjestelmään. Tarjoustyökalun on tarkoitus tuoda helpotusta kiinteistöalalla toimivan asiakkaan toimintoihin auttamalla heitä kiinteistöjen hallinnassa ja niiden tarjoamisessa eteenpäin. Järjestelmän on tarkoitus korvata heidän nykyinen kiinteistöhallintajärjestelmä tuomalla lukuisia uusia ominaisuuksia, minkä lisäksi sen täytyy toimia heidän käyttämän CRM-järjestelmän rinnalla. Tämä ei onnistunut vanhassa järjestelmässä.

Asiakas näki uuden järjestelmän tarpeelliseksi, koska he tarvitsivat työkalun jolla tehdä tarjouksia ja joka toimii heidän käyttämänsä CRM-järjestelmän yhteydessä. Samalla uusittiin myös heidän kiinteistörekisterinsä, koska se kuuluu selvästi osaksi tarjousjärjestelmää. Kuten CRM-järjestelmäkin, on uuden työkalun tarkoitus toimia Web-ympäristössä, jolloin sitä pääsee käyttämään mistä tahansa.

Tarjousjärjestelmän toiminta kuvailtiin heti alusta asti hyvin samanlaiseksi kuin perinteisten verkkokauppa-ohjelmistojen. Myyjä kokoaa tarjouksia poimimalla kiinteistöjä ikään kuin ”ostoskoreihin” ja näistä koreista muodostetaan sitten tarjouksia.

Heti alusta asti pidettiin tärkeänä, että järjestelmän käyttö pitää olla sujuvaa ja helppoa. Vaikka järjestelmä toimiikin web-ympäristössä, haluttiin, että se toimisi kuten työpöytäohjelmistot. Näin päätettiin, että järjestelmän käyttöliittymä rakennetaan nykyaikaisia menetelmiä hyväksikäyttäen. Tämän takia ajateltiin, että paljon puhuttu Ajax-tekniikka voisi tuoda apua järjestelmän käyttöliittymän rakentamiseen, jotta siitä saataisiin nopea ja tehokas.

Isoltan ajatus projektiin lähdettäessä oli toteuttaa mahdollisimman dynaaminen järjestelmä, jota voisi myöhemmin käyttää erilaisissa verkkokauppa-/tarjoustyökalutoteutuksissa. Järjestelmä suunniteltiin siis mahdollisimman helposti muunneltavaksi, eikä sitä suunniteltu pelkästään kiinteistöalalle. Tämä toi projektiin omat haasteensa. Järjestelmän muunneltavuuden on myös tarkoitus helpottaa tulevaisuuden muutoksien tekemistä myös kiinteistöalalla toimivalle asiakkaalle. Tämän opinnäytetyön tarkoituksena on tehokkaan Ajax-pohjaisen tarjoustyökalun toteuttaminen.

1.1 Taustaa ja työn lähtökohdat

Tarjoustyökalu toteutetaan Isolta Oy:n kehittämän ohjelmistoalustan päälle. Alustan päälle on aikaisemmin toteutettu muun muassa sivuston hallinta- ja julkaisujärjestelmä ja monia siihen liittyviä moduuleja. Järjestelmän päälle on myös kehitetty yksinkertainen sovellus, jonka avulla voi tehdä tarjouspyyntöjä, mutta tämän käyttö uudessa sovelluksessa jää varsin rajalliseksi, koska uudesta järjestelmästä oli tarkoitus tehdä huomattavasti helpommin muokattava erilaisiin käyttökohteisiin. Kehitysalusta on modulaarinen ympäristö, joka helpottaa ohjelmiston kehittämistä valmiilla toiminnoillaan. Järjestelmästä on tarkoitus muodostaa uusi moduuli kyseiseen järjestelmään. Moduulista tuli kuitenkin huomattavasti laajempi, kuin aikaisemmat sen päälle rakennetut, joten tämä toi omat haasteensa järjestelmän kehittämiseen. Lisäksi Ajax-tekniikka toi omat haasteensa, koska aikaisemmin ei järjestelmän päälle ollut rakennettu tätä hyväksikäyttäviä sovelluksia.

Järjestelmän kehittämiseen Isoltalta osallistui kolme eri henkilöä, joilla kaikilla on omat tehtävänsä. Järjestelmän suunnitteluun osallistui henkilöitä myös asiakasyrityksestä, heidän rooli oli kuitenkin enemmän kertoa, miten järjestelmän tuli toimia. Järjestelmän varsinainen kehitystyö aloitettiin jo keväällä 2008, mutta vasta kesän aikana järjestelmään päästiin kunnolla tekemään Ajax-toteutuksia.

1.2 Tavoitteet

Opinnäytetyön tavoitteena on tutkia Ajax-tekniikan käyttöä ja löytää ratkaisut siihen miten Ajax-tekniikkaa voisi hyödyntää tarjousjärjestelmässä. Opinnäytetyössä keskitytään siis Ajax-tekniikkaan ja siihen, miten tätä voidaan käyttää apuna nykyaikaisissa web-sovelluksissa.

Ajax-toteutuksen onnistumisen mittarina toimii sovelluksen nopeus, jota mitataan vertaamalla Ajax-pohjaista toteutusta ja perinteisellä tavalla toteutettua versiota web-sovelluksesta. Mittausta pidetään tärkeänä, koska sen avulla voidaan päätellä, onko Ajax-toteutus onnistunut toivotulla tavalla ja nopeuttaako se todella sovelluksen toimintaa ja sen käyttöä. Nopeuden lisäksi Ajax-tekniikan on tarkoitus tuoda kehitystä sovelluksen käyttöliittymään ja tuoda siihen työpöytäsovelluksista tuttuja piirteitä. Tavoitteena on myös selvittää, mitä taitoja vaaditaan Ajax-asiantuntijalta. Tämän tiedon avulla voidaan kehittää ammatillista osaamista. Näitä pohditaan enemmän työn lopussa.

1.3 Tutkimusmenetelmä

Opinnäytetyössä on käytetty tutkimusmenetelmänä konstruktiivista menetelmää, eli tutkimus on suunnittelutieteellinen. Tästä tutkimusmenetelmästä käytetään myös nimeä soveltava tutkimus. Tarkoituksena on luoda tietämystä suunnittelua ja toteutusta varten, eli konstruktio-ongelmien ratkaisemista varten. Konstruktiivisessa tutkimuksessa tavoitteena on käytännön ongelman ratkaisu luomalla uusi konstruktio. (Järvinen & Järvinen 2004, 103.)

Konstruktiivinen tutkimusmenetelmä soveltuu uuden todellisuuden, tässä tapauksessa Ajax-tekniikan käyttöön ja kehittämiseen olemassa olevien tietojen pohjalta. Konstruktiivisessa tutkimuksessa on ratkaistava, millaista todellisuutta halutaan rakentaa. Rakentajien ja päätöksentekijöiden arvoista riippuu, millaiseksi tavoitetila määritetään. Konstruktiivinen tutkimus ja sen metodit pyrkivät vastaamaan kysymyksiin, jotka koskevat rakentamista, parantamista, tarkoituksellista muuttamista, käyttöönottoa, vahvistamista, sovittamista, korjaamista sekä muutostoimenpiteiden arviointia. (Järvinen & Järvinen 2004, 103, 109.)

1.4 Rajaukset

Järjestelmä on laaja kokonaisuus ja sisältää monia eri osa-alueita. Toteutus onkin tarkoitus rajata tarjousten luomiseen tarkoitettuihin osioihin ja siinä vielä Ajax-tekniikka hyödyntäviin osiin sovellusta. Rajauksella tarkoitetaan sitä, että opinnäytetyön puitteissa järjestelmään kehitetään tarjousten luomiseen tarkoitettuja osioita sekä palvelin- että selainpäähän. Tämä tarkoittaa sitä, että kiinteistöjen hallintaan liittyvät osiot jätetään työn ulkopuolelle, vaikka näissäkin toiminnoissa Ajaxilla on tärkeä rooli.

2 KÄSITTEET

Työssäni käsitellään useita eri termejä, jotka kaikki liittyvä osaltaan Ajax-tekniikkaan ja järjestelmän toteuttamiseen.

AJAX

Ajax (Asynchronous JavaScript And XML) on tekniikka, jota käytetään vuorovaikutteisten web-sovellusten luomiseen. Ajaxin avulla selain keskustelee palvelimen kanssa siten, että koko web-sivua ei tarvitse ladata uudelleen, vaan siitä voidaan päivittää vain osia. Näitä osia voidaan päivittää vielä asynkronisesti selaimen ja palvelimen välillä. (Garrett 2005.) Työässäni Ajax-tekniikan rooli on järjestelmän käyttöliittymän toteutuksessa.

JavaScript

JavaScript on oliopohjainen komentosarjakieli ja sitä käytetään erittäin laajasti web-ympäristössä. JavaScript toimii selainpäässä. JavaScript on tärkeä osa Ajaxin toiminnallisuutta ja tämän takia sitä käytetään työssäni käyttöliittymän toteuttamiseen ja Ajax-toiminnallisuuksien luomiseen.

HTML/XHTML

HTML (Hypertext Markup Language) on W3C:n standardoima kuvauskieli, jonka avulla rakennetaan yleisesti Internet-sivuja. XHTML on tästä jatkettu versio, joka täyttää XML:n muotoiluvaatimukset. Sovelluksessa HTML-kuvauskieltä käytetään käyttöliittymän toteuttamiseen.

CSS

CSS (Cascading Style Sheets) on erityisesti Internet-sivuilla käytettävä tyyliohje, jonka avulla muotoillaan esimerkiksi HTML:n tulostusasuja. Työssäni CSS-määrittelyjä käytetään käyttöliittymän ulkoasun muotoiluun.

XML

XML (Extensible Markup Language) on rakenteellinen kuvauskieli, jolla tiedon merkitys on kuvattavissa tiedon sekaan. XML-kieltä käytetään tiedonvälitykseen järjestelmien välillä ja dokumenttien tallentamiseen. Se on rakenteellinen kuvauskieli, joka auttaa jäsentämään laajoja tietomassoja selkeämmin. XML:n kehittäjä on W3C. XML:n muotoilu muistuttaa HTML-kieltä, mutta se ei sitä ole. HTML-kuvauskielessä käytetään ennalta määrättyjä merkkejä kuvaamaan rakenteita, mutta XML-kielessä näin ei ole, vaan rakenteet voidaan määrittää itse. XML-kielen avulla voidaan välittää tietoa minkälaisten sovellusten kanssa tahansa alustariippumattomasti. (Walsh 1998, IBM)

Työssäni XML-kielen avulla tehdään erilaisia määrityksiä muun muassa integraatioihin liittyen ja sitä käytetään Ajax-kutsuissa.

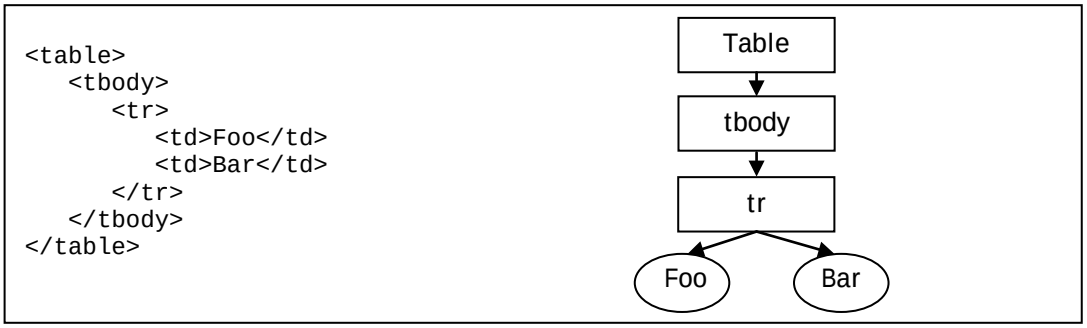
XSLT

XSLT (Extensible Stylesheet Language Transformations) on XML-pohjainen merkintäkieli XML-tiedostojen muunnoksiin. Kielen ajatuksena on, että alkuperäistä XML-dokumenttia ei muuteta, vaan sitä tyylitellään erillisen XSLT-tiedoston avulla, kuten CSS tekee HTML-dokumenteille. XSLT on yksi kolmesta XSL-kielen osasta. Muita XSL-kielen osia ovat XPath,

joka mahdollistaa XML-dokumentin elementeissä ja attribuuteissa liikkumisen (tätä käytetään myös XSLT-muunnoksissa apuna) ja XSL-FO, jonka avulla XML-dokumenttia voidaan käsitellä. XSLT on W3C:n määrittämä web-standardi. (Floyd, Gulbranssen & Hinder 2002, 10-11.) XSLT:n rooli työssä on toimia XML-tuloksien muotoilijana.

DOM (Document Object Model)

DOM-malli on W3C:n alusta- ja kieliriippumaton rajapinta dokumentin sisällön ja rakenteen muokkaamiseen. Kyseessä on yleinen tapa muokata ja esittää HTML- tai XML-dokumentteja. Sen voi ajatella puumaiseksi malliksi esittää tietoa ja tiedon rakennetta (Esimerkki DOM-mallin puumaisuudesta taulukossa 1). Se tarjoaa standardoidun tavan käsitellä dokumentteja. (Asleson & Schutta 2006, 39.)



Kuvio 1: Yksinkertainen esimerkki DOM-mallista

Työssäni DOM-malli kuuluu osaksi Ajax- ja JavaScript toteutuksia, koska se mahdollistaa dokumentin muokkaamisen selaimessa.

PHP

PHP on ohjelmointikieli, joka on erittäin laajasti käytössä web-ympäristöissä. Se on komentosarjakieli eli se tulkitaan vasta ohjelman suoritusvaiheessa. Työssäni PHP:n rooli on hoitaa palvelinpuolen toimintoja.

MySql

MySql on avoimen lähdekoodin SQL-tietokanta. Työssäni MySql toimii järjestelmän tietokantapohjana.

Asynkroninen

Käsitteellä asynkroninen tarkoitetaan Ajax-tekniikan yhteydessä sen tapaa lähettää kutsuja palvelimelle ei-reaaliaikaisesti. Tällä tarkoitetaan sitä, että kutsut eivät ole ajallisesti toisistaan riippuvaisia. Kahta eri kutsua lähetettäessä ei voida sanoa, että ensimmäinen palautuu välttämättä selaimeen ennen toista kutsua. Kutsut ovat siis riippumattomia toisistaan. Perinteisellä tavalla toteutetussa web-sovelluksessa palautetaan aina koko sivu palvelinkutsun yhteydessä (synkroninen), kun taas Ajax-pohjaisessa sovelluksessa voidaan tehdä sivuja täydentäviä kutsuja. (Lauria 2008, 3.)

3 AIKAISEMMAT TUTKIMUKSET: AJAX

Vaikka Ajax on tekniikkana suhteellisen uusi, on siitä ehditty tehdä jo useita kirjoja. Tässä luvussa tarkastelen Ajax-tekniikasta tehtyjä aikaisempia tutkimuksia. Kerättyä tietoa hyödynnetään itse toteutusvaiheessa sovelluksen kehittämiseen. Sitä käytetään apuna myös lopun pohdinnoissa siitä, mitä taitoja ja tietoja Ajax-osaajalta vaaditaan.

3.1 Mikä on Ajax?

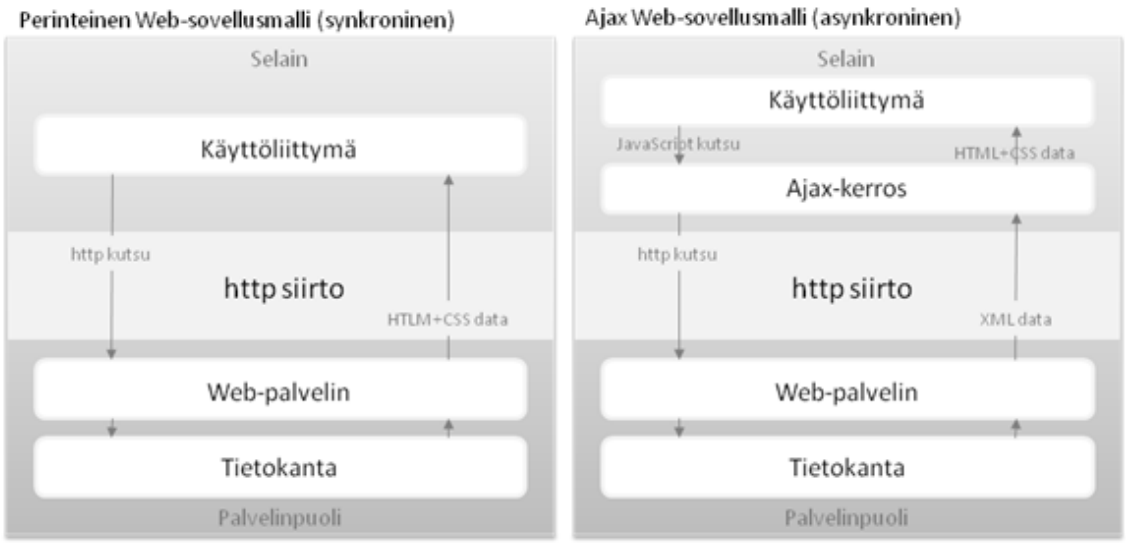
Ajax on yhdistelmä sanoista Asynchronous JavaScript and XML. Se ei ole yksittäinen tekniikka, vaan kokoelma jo pitkään käytössä olleita tekniikoita. Nämä tekniikat on yhdistetty uudenlaisella tehokkaalla tavalla. Ajax koostuu XHTML:n, CSS:n, DOM:n, XML:n, XSLT:n ja JavaScriptin yhdistelmästä. Ajaxista on nopeasti tullut lähes *de facto* – työkalu websovellusten käyttöliittymän luomiseen. Ajaxin avulla voidaan tehdä web-sovelluksia, jotka muistuttavat entistä enemmän tavallisia työpöytäohjelmistoja. Sivuja muokataan tällöin dynaamisesti ja uusia sivulatauksia ei tehdä käyttäjän tekemien toimintojen yhteydessä, kuten perinteisissä web-sovelluksissa. (Asleson & Schutta 2006, 15 ja 25.)

Tutkitaan miten perinteinen web-palvelu ja Ajax-tekniikan avulla toteutettu web-palvelu eroaa toisistaan. Perinteisessä web-palvelussa suurin osa (ellei suoriteta esimerkiksi JavaScript toimintoa) selaimessa käynnistettävistä toiminnoista käynnistää HTTP-kutsun, joka välitetään palvelimelle. Palvelin ottaa kutsun vastaan ja käsittelee sen. Käsittelyn jälkeen palvelin palauttaa uuden HTML-dokumentin selaimen ruutuun. Joka kutsun yhteydessä sivu ladataan uudelleen selaimeen ja käyttäjä joutuu odottamaan. (Garrett 2005.)

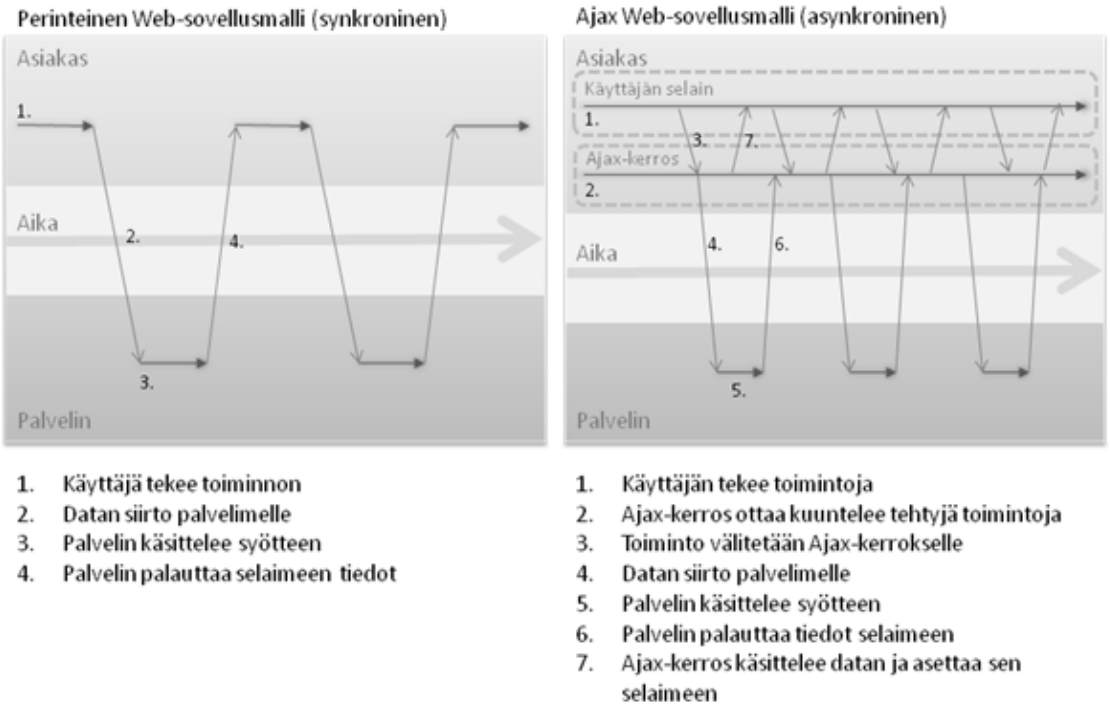
Ajax-tekniikka erottuu perinteisestä web-tekniikasta selvästi. Ajax-tekniikkaa käytettäessä palvelimen ja selaimen välille lisätään kerros, jonka avulla selain keskustelee palvelimen kanssa. Sivun uudelleen latauksen sijasta selain lataa muistiin JavaScriptillä kirjoitetun kerroksen. Tämä kerros ottaa vastaan käyttöliittymästä tulleet pyynnöt ja välittää ne eteenpäin

palvelimelle muodostaen uuden HTTP-kutsun. Palvelin käsittelee kutsun ja palauttaa vastauksen JavaScript kerrokselle, joka päivittää tiedot käyttäjän selaimeen. Perinteisellä web-tekniikalla ja Ajax-tekniikalla toteutettujen sovellusten eroja selvennetään kuvissa 1 ja 2. (Garrett 2005.)

Ajax-kerros mahdollistaa web-palveluiden perinteisen synkronisen mallin muuttamisen asynkroniseksi. Tällä tarkoitetaan sitä, että kun perinteisessä web-palvelussa sivu ladataan aina uudelleen, ladataan Ajax-pohjaisessa web-palvelussa dynaamisesti esimerkiksi vain tietty osa sivusta. Käyttäjä ei siis näe välillä tyhjää selainruutua tai tiimalasia sivun lataamisen merkiksi. Tämä taas mahdollistaa aivan uuden tyyppisten web-sovellusten kehittämisen. (Garrett 2005.)



Kuva 1: Ajax- ja perinteisen web-sovellusmallin erot (Garrett 2005.)



Kuva 2: Ajax- ja perinteisen web-sovellusmallin toimintatapa (Garrett 2005.)

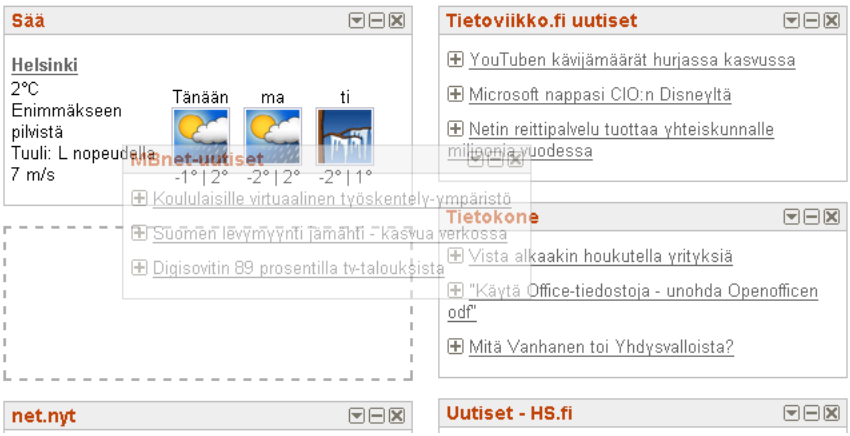
3.1.1 Ajaxin käyttökohteet ja käytön esimerkkejä

Lähes jokainen verkkosovellus voi hyötyä Ajaxista, mutta on myös tärkeää selvittää, milloin sitä kannattaa ja milloin sitä ei kannata käyttää. Verkkosovellusta kehitettäessä on tärkeä ottaa huomioon selainten poikkeavuudet, jos sovelluksen käyttäjät käyttävät esimerkiksi paljon vanhempia selaimia. On myös huomioitava millä alustalla verkkosovellusta käytetään. Jos ollaan luomassa sovellusta, jonka käyttäjiin lukeutuu myös mobiili-käyttäjiä, ei Ajax varmasti ole paras mahdollinen valinta. Hyviä Ajax-tekniikan käyttökohteita ovat esimerkiksi automaattisesti päivittyvät sivut, työkaluvihjeet, automaattisesti täydentyvät lomakkeet, edistykselliset käyttöliittymäkomponentit ja virheiden tarkistus dynaamisesti. (Asleson & Schutta 2006, 20-21.)

Hyviä esimerkkejä Ajaxin käytöstä löytyy paljon. Esimerkiksi Googlen muokattava etusivu, Google Suggest ja Gmail. Google on muutenkin panostanut paljon Ajax tekniikkaan ja sen hyödyntämiseen verkkosovelluksissaan. Myös erilaiset yhteisöpalvelut, kuten Facebook hyödyntää paljon Ajax-tekniikkaa.

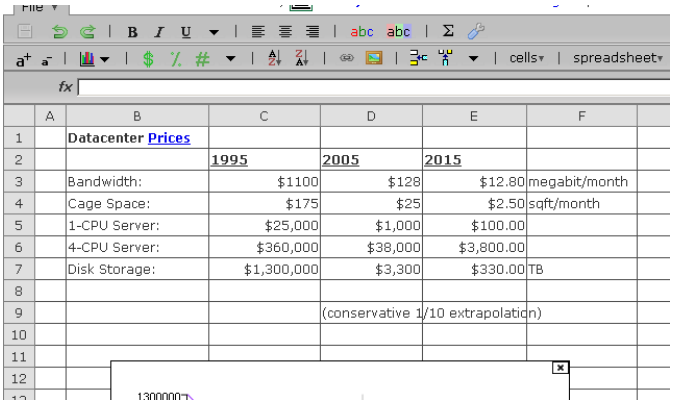
Ajaxin käytöllä voidaan tosiaan kehittää web-sovelluksia uudenlaisella tavalla. Se mahdollistaa paljon työpöytäsovelluksista tuttuja ominaisuuksia käytettäväksi web-ympäristössä. Tässä muutama esimerkki siitä, mitä sen avulla pystyy toteuttamaan.

Googlen personoitava kotisivu on hyvä esimerkki ”Drag and Drop” -toiminnon käyttämisestä. Sivulla pystyy järjestelmään eri sivun osiot haluamaansa järjestykseen vain hiirellä osioita siirtämällä.



Kuva 3: Ajax-esimerkkejä, Googlen muokattava etusivu

Toinen hyvä esimerkki on http://numsum.com/ osoitteesta löytyvä taulukko-laskentasovellus. Tämän avulla voi päätellä mitä kaikkea voidaan Ajaxin avulla toteuttaa Internet-ympäristöön.



Kuva 4: Ajax-esimerkkejä: Numsun-taulukkolaskentasovellus web-ympäristössä

3.1.2 XMLHttpRequest-olio

Ajaxin toiminnallisuuden lähtökohtana on JavaScriptin XMLHttpRequest-olio, joka sisällytettiin ensimmäisen kerran jo Internet Explorer selaimen versioon 5. Tekniikka on ollut olemassa jo pitkään, mutta vasta muiden selainten toteuttaessa XMLHttpRequest-olio, sen käyttö alkoi laajeta web-kehittäjien keskuudessa. XMLHttpRequest-olion avulla selain pystyy lähettämään

ja vastaanottamaan dataa palvelimen ja selaimen välillä. Palvelimelle voidaan tehdä olion avulla asynkronisia palvelupyyntöjä. Nämä palvelinpyynnöt luodaan käyttämällä JavaScriptin tarjoamaa XMLHttpRequest-oliota, jonka avulla pystytään kutsumaan palvelinta itsenäisesti. Kutsu palvelimelle muodostetaan normaalin HTTP-kutsun tavoin käyttäen joko GET tai POST metodia. Yleisesti ottaen kutsun muodostaminen hyväksytään vain sille palvelimelle, jolta sivu on ladattu. Vastaus kutsuun palaa JavaScript-oliolle ja tämän avulla se voidaan asettaa jonkin HTML-elementin sisällöksi tai se voidaan asettaa JavaScript muuttujaan jatkokäsittelyä varten. XMLHttpRequest:n sisällön ei välttämättä tarvitse olla XML-muotoista dataa nimestään huolimatta, vaan se voi olla esimerkiksi tekstiä. XMLHttpRequest sisältää siis useita metodeita ja ominaisuuksia, joilla voidaan vaikuttaa yhteyteen joka palvelimelle muodostetaan, tai joilla palvelimelta tulleet vastaukset otetaan vastaan. (Ullman & Dykes 2007, 14-15; Zakas, McPeak & Fawcett 2007, 25-31.)

Valitettavasti XMLHttpRequest-olio ei ainakaan vielä ole standardoitu, joten sen käyttö pitää huomioida eri selaimissa. Ongelma koskee erityisesti Microsoftin Explorer selainta, joka toteuttaa XMLHttpRequest-olion eri tavalla (ActiveX), kuin muut selaimet. Mahdollista on, että XMLHttpRequest-olio tullaan standardoimaan tulevaisuudessa W3:n level 3 of the DOM määrittelyssä. Taulukossa 2 on yksinkertainen esimerkki XMLHttpRequest-olion muodostamisesta JavaScriptillä. Huomioitavaa on selaintunnistuksessa, jossa tarkastetaan, onko kysymyksessä Microsoftin Explorer selain. (Ullman & Dykes 2007, 14-15.)

```
var xHRObject = false;
if (window.XMLHttpRequest){
   xHRObject = new XMLHttpRequest();
}
else if (window.ActiveXObject) {
   xHRObject = new ActiveXObject("Microsoft.XMLHTTP");
}
else {
  // Tee jotain selaimessa, joka ei tue XMLHttpRequest-oliota
}
```

Kuvio 2: XMLHttpRequest-olion muodostaminen

3.2 Ajax ja JavaScript

JavaScriptin avulla voidaan toteuttaa ohjelmia, jotka toimivat suoraan selaimessa. Se on skriptikieli eli tulkattava kieli, jonka esimerkiksi selain suorittaa. Sen avulla voidaan laajentaa selaimeen ladattavien dokumenttien toimintoja ja tehdä niistä dynaamisia. JavaScript kielen syntaksi pohjautuu osittain Javaan, mutta niillä ei silti ole kovin paljon yhteistä. JavaScriptin

ensimmäinen versio on kehitetty vuonna 1995 Netscape-navigator selaimen myötä. JavaScript tunnetaan myös nimellä EcmaScript. (Peltomäki 2001, 2-12.)

Useat kehittäjät karttavat JavaScript-kielen käyttöä, koska virheiden jäljittäminen ja koodin testaaminen on liian hankalaa. Ongelmana on myös se, että se toimii eri selaimissa vaihtelevalla tavalla ja se hankaloittaa koodin kirjoittamista. JavaScript kehittämiseen on kuitenkin viimevuosina kehitetty paljon erilaisia apuvälineitä, joita ovat muun muassa Mozilla Firefox-selaimeen asennettavat laajennukset, joilla onnistuvat esimerkiksi koodin siistiminen ja sen virheenjäljitys. Erilaisten sovellusten avulla voidaan JavaScript ja Ajax-kehitys prosesseista tehdä myös huomattavasti aiempaa helpompia. (Asleson & Schutta 2006, 6, 150 ja 160–161.)

Ajax-tekniikan yksi tärkeimmistä osa-alueista on JavaScript. JavaScript on erityisen tärkeässä asemassa juuri HTTP-kutsuja luodessa ja niitä käsitellessä. JavaScript on myös ainoa yleisesti tuettu tekniikka, jolla voidaan vaikuttaa selaimen sisältöön dynaamisesti (DOM). Kun HTTP-kutsu on suoritettu, voidaan JavaScriptin avulla päivittää selaimen ruutuun tietoja. Voidaankin sanoa, että JavaScript on tekniikka, joka tekee Ajaxista toimivan ja mahdollisen (Holzner 2007, 2,1).

3.3 Tiedon kuljettaminen selaimeen

Ajax-tekniikassa XMLHttpRequest-olio tarjoaa kaksi tapaa päästä käsiksi palvelimen vastaukseen. Näitä ovat ”responseText” ja ”responseXML”. Molemmat tavat ovat XMLHttpRequest-olion ominaisuuksia. (Ullman & Dykes 2007, 70.)

ResponseText on yksinkertaisin ja yleisin tapa palauttaa vastaus palvelimelta selaimeen jolloin vastaus palautetaan merkkijonona. Merkkijono voi tällöin olla esimerkiksi HTML-muotoiltua dataa, tai pelkkää tekstiä. ResponseText on helppo lisätä suoraan selaimeen HTTP-kutsun jälkeen, mutta sitä on hankala käsitellä edelleen JavaScriptin avulla, koska siinä ei ole elementtejä tai tietotyyppejä. Poikkeuksena tähän on kuitenkin JSON-muotoiltu data, josta kerrotaan jäljempänä. (Ullman & Dykes 2007, 70 ja 137.)

ResponseXML on taas juuri XML-datan kuljetukseen tarkoitettu, joten palvelin palauttaa vastauksen XML-oliona. Vaikka ResponseText:n avulla voidaan myös siirtää XML-dataa, tarjoaa responseXML huomattavasti etuja XML-tiedon käsittelyyn. Kun tiedonsiirto tehdään palvelimen ja selaimen välillä, kannattaa ottaa huomioon, miten dynaamista datan pitää selaimessa olla. Jos data on tekstiä, on sitä selainpäässä huomattavasti vaikeampi käsitellä ja tällöin dynaamisen sisällön luominen hankaloituu. XML-muotoisen datan käyttäminen helpottaa dynaamisen sovelluksen kehittämistä, koska tällöin dataa voidaan esimerkiksi lajitella tai helpommin käsitellä selainpäässä. (Asleson & Schutta 2006, 11; Ullman & Dykes 2007, 70-71.)

Seuraavaksi käsitellään tärkeimmät tiedon jäsentämistavat dynaamisessa Ajax-sovelluksessa. Näitä ovat: 1. JSON ja 2. XML.

1. JSON (JavaScript Object Notation)

JSON on kevyt tapa siirtää tietoa palvelimelta selaimeen. Se perustuu suoraan JavaScript-kielen taulukoihin (array) ja siksi sitä on nopea käsitellä selaimessa JavaScriptin avulla, ilman raskasta tiedon parsimista. JSON-muotoa käytettäessä on tärkeää ymmärtää JavaScriptin taulukoiden ja olioiden käsittely. Sitä käytettäessä käytetään XMLHttpRequest-olion ResponseText-ominaisuutta. JSON muotoillun datan etuja ovat sen keveys ja käsiteltävyys. Verrattuna XML-muotoiseen dataan, on sitä huomattavasti helpompi ja nopeampi käsitellä. (Zakas, McPeak & Fawcett 2007, 237; Ullman & Dykes 2007, 351)

2. XML (eXtensible Markup Language)

XML-dataa käytetään Ajax-sovelluksissa XMLHttpRequest-olion ResponseXML-ominaisuuden kanssa, jolloin data palautetaan palvelimelta selaimeen XML-muodossa. XML-data pitää siten käsitellä selaimessa JavaScriptin avulla. Apuna tässä voidaan käyttää myös XSLT-muunnosta, josta kerrotaan lisää myöhemmin. XML-muotoisen datan vahvuus Ajax-sovelluksessa on sen rakenteellisuus. XML-dataa käytettäessä voidaan kuvata tarvittaessa datan erilaiset tietotyypit ja muotoilut. Tällöin sitä on selkeämpi käsitellä, kuin pelkkää tekstiä. XML-datan käsittelyyn löytyy JavaScriptistä paljon valmiita käsittelijöitä, joilla voidaan käydä läpi XML-datan solmuja. Kaikki nykyaikaiset selaimet tukevat XML-datan käsittelyä JavaScriptin avulla. XML-datan käsittely liittyy W3C:n DOM-malliin. (Ullman & Dykes 2007, 137–138; Holzner 2007, 238–242.)

3.4 Tiedon esittäminen ja sen muotoilu

Kun tieto on haettu palvelimelta XMLHttpRequest:n avulla, on se muotoiltava selaimen sisältöön, jotta se näkyisi käyttäjälle. Tiedon esittäminen Ajax-sovelluksessa toteutetaan W3C DOM-mallin ja JavaScript avulla. Tiedon muotoilua voidaan tehdä myös esimerkiksi XSLT-muunnoksien avulla.

W3C DOM-malli on alusta- ja kieliriippumaton rajapinta, jonka avulla ohjelmat ja skriptit voivat käsitellä ja päivittää dokumentin sisältöä, rakennetta ja tyyliä dynaamisesti. W3C DOM-malli antaa siis mahdollisuuden sivun sisällön dynaamiseen muokkaamiseen. Sen avulla voidaan liikkua dokumenttipuussa ja muokata sitä. Se tarjoaa sekä joukon ominaisuuksia, joilla voidaan muokata dokumentin rakennetta, että metodit joiden avulla päästään käsiksi olioiden

sisältöön. Mallia käyttämällä voidaan käsitellä mitä tahansa elementtiä XHTML- tai XML-dokumentissa JavaScriptin avulla ja muuttaa esimerkiksi elementtien sisältöä, ominaisuuksia, sijaintia dokumentissa, tyylimäärittelyjä tai lisätä uutta sisältöä. Jotta DOM-mallia voidaan tehokkaasti käsitellä, on ymmärrettävä XHTML- tai XML-dokumentin rakenne ja niiden puumaisuus. (Ullman & Dykes 2007, 47-48; W3C DOM-malli; Asleson & Schutta 2006, 50.)

DOM-malli mahdollistaa dokumentin muokkaamisen ja käsittelyn JavaScriptin avulla. Dokumentin muokkaaminen DOM-mallin ansiosta on kohtalaisen helppoa sen tarjoaman rajapinnan takia. Esimerkkinä voidaan mainita getElementById-metodi, jonka avulla voidaan XHTML- tai XML-dokumentista palauttaa elementti, jolla on uniikki ID-arvo. JavaScript sisältää paljon metodeita, joiden avulla dokumentin muokkaaminen ja käsittely onnistuu dynaamisesti. (Asleson & Schutta 2006, 45–46.)

XSLT mahdollistaa XML-dokumentin konvertoinnin tai kääntämisen erilaiseen muotoon. Sen avulla XML-dokumentin voi kääntää esimerkiksi toiseen XML-muotoon tai vaikka HTML-muotoon, joka sitten esitetään web-sivulla. Sitä käytetään yleisesti HTML-sivun generointiin XML-dokumentista. (Floyd, Gulbranssen & Hinder 2002 10; W3C - Quin.)

XSLT:n avulla voidaan muuntaa XML-dokumentti toiseen muotoon. Tämä mahdollistaa web-järjestelmässä käyttöliittymän ja järjestelmän logiikan erottamisen toisistaan. Esimerkiksi palvelin tekee tietokannasta haun ja käsittelee sen XML-muotoon. Tämän jälkeen se käännetään XSLT:n avulla HTML-muotoon. Sama XML-data voidaan käsitellä tällöin esimerkiksi toiseen XML-muotoon, tai erilaiseen HTML-muotoon, jota käytetään vaikka PDA-laitteessa. (Floyd, Gulbranssen & Hinder 2002, 10-11.)

XSLT-käännökset voidaan tehdä sekä palvelimella, että selaimessa. Esimerkiksi PHP-kielessä XSLTProcessor-luokan avulla (http://fi.php.net/manual/en/class.xsltprocessor.php). Ajax-pohjaisessa sovelluksessa muunnos voidaan tehdä myös selaimessa. Tällöin voidaan käyttää XMLHttpRequest-olion responseXML-ominaisuutta, jonka avulla tieto haetaan palvelimelta XML-muotoisena. Muunnos tehdään selaimessa JavaScriptin avulla, jolla ladataan XSLT-tiedosto ja muunnetaan XML-data sen avulla HTML-muotoon. Kun muunnos on tehty, asetetaan muunnettu tieto DOM-mallin rajapinnan avulla sivulle. Valitettavasti Applen Safari selain ei ainoana nykyisistä selaimista tue JavaScriptin avulla tehtävää XSLT-muunnosta, joten se rajoittaa sen käyttömahdollisuuksia. (Zakas, McPeak & Fawcett 2007, 179-192.)

3.5 Ajax ja verkkosovellusten nopeus

Kuten Ajaxin esittelyssä oli jo puhe, niin Ajaxin käyttö vaikuttaa ehdottomasti web-sovellusten nopeuteen. Kun perinteisessä web-sovelluksessa painetaan jotain nappia, koko

sivu ladataan uudelleen, kun taas Ajax-sovelluksessa päivitetään vain jokin tietty osa sivusta. Se ei kuitenkaan tarkoita, että Ajaxilla toteutettu verkkopalvelu olisi välttämättä nopeampi kuin perinteisellä tavalla toteutettu. Nopeus riippuu siitä, minkä tyylinen sovellus on kysymyksessä. Jos kyse on esimerkiksi yrityksen web-sivustosta, ei Ajaxin käyttö välttämättä nopeuta sivuston toimintaa. Yleensä Ajax-sivustoon liittyy nimittäin useita JavaScript-tiedostoja ja paljon muita muistiin ladattavia komponentteja. Yleensä tehdään myös useita hakuja palvelimelle ladattaessa sivua ensimmäistä kertaa. Sivusto käyttää tällöin kohtalaisen paljon muistia, mutta tästä ei ole hyötyä käyttäjille ja siirrosta tulee huomattavasti raskaampi. (Lauria 2008, 11-13.)

Verrattuna perinteisellä tavalla toteutettuun web-sovellukseen, Ajaxin myötä selaimen ja palvelimen välisten pyyntöjen keskimääräiset tietomäärät monesti pienenevät, mutta vastaavasti pyyntöjen määrä kasvaa radikaalisti, mikä aiheuttaa omat vaatimuksensa palvelintoteutuksen laadulle ja kuorman kestolle. Tämä tarkoittaa sitä, että web-sovellusten luominen vaatii entistä enemmän testaamista ja suunnitelmallisuutta. (http://www.samcom.fi/Miten_AJAXia_voi_hyodyntaa_verkkopalveluissa.pdf)

3.6 Vaihtoehtoiset tekniikat

Ajaxin käytöllä on myös vaihtoehtoja. Samantapaisia dynaamisia toiminnallisuuksia saadaan aikaiseksi esimerkiksi käyttämällä Adoben Flash-tekniikkaa, tai Javalla toteutettavien sovelmien avulla (Java-applet). Varsinkin Java-tekniikkaan pohjautuvat ratkaisut voivat olla todella raskaita ja ne vaativat lisäksi oman Java Virtuaali koneen asennuksen sitä käyttävälle koneelle. Myös Flash-tekniikka vaatii selaimeen asennettavan lisäosan.

Flash-tekniikat ovat nykyään varsin suosittuja. Tekniikka on ensisijaisesti kehitetty animaatioiden toteuttamiseen, mutta nykyisin sillä voidaan toteuttaa myös varsin monipuolisia websovelluksia. Sovellusten kehittämiseen käytetään tässä tekniikassa ActionScript-kieltä. ActionScript kielestä on hetki sitten ilmestynyt versio 3.0, joka on laajentanut kieltä huomattavasti. ActionScript 3.0 perustuu ECMAScriptiin, joka on juonnoissa käytettävä kansainvälinen standardoitu ohjelmointikieli. ECMAScriptiin perustuu myös JavaScript kieli, joten kielet ovat toistensa kaltaisia. (Korpela 2007, Adobe 2008.)

Toinen vaihtoehtoinen tekniikka on Java ja sen sovelmat (applet). Java-sovelmien avulla voidaan toteuttaa todella monipuolisia vuorovaikutteisia verkkosovelluksia. Sovelma liitetään verkkosivuun luomalla dokumenttiin applet-elemntti, joka viittaa palvelimella sijaitsevaan sovelman luokkatiedostoon. Java-sovelma ei välttämättä vaadi toimiakseen jatkuvaa Internetyhteyttä. Java-sovelmien huono puoli on se, että kaikki selaimet eivät niitä tue ja ne vaativat selaimeen liitettävän ajoympäristön. (Korpela 2007.)

3.7 Ajax ja valmiit sovelluskehykset (Framework)

Ajaxista puhuttaessa kuulee usein myös Ajax-sovelluskehyksistä (Framework), tai pikemminkin JavaScript-sovelluskehyksistä. Lähes kaikessa Ajaxiin liittyvässä kirjallisuudessa puhutaan myös tästä. On siis syytä pohtia mitä nämä ovat ja kannattaako sellaisia käyttää ja mihin niitä kannattaa hyödyntää? Kun Internetistä tutkii asiaa huomaa, että sovelluskehyksiä on lukemattomia ja uusia ilmestyy kokoajan. Ajaxin ansiosta JavaScript kasvattaa suosiotaan jatkuvasti.

Muutamista sovelluskehyksistä ja niiden käytöstä on kirjoitettu myös kirja. Tämä tarkoittaa sitä, että kehyksiä todella käytetään ja että ovat suosittuja kehittäjien keskuudessa. Kirjoja on kirjoitettu esimerkiksi Prototype- ja Mootools- nimisistä kehyksistä. Esimerkiksi Prototypestä kirjoitettu kirja on nimeltään Prototype and script.aculo.us: You never knew JavaScript could do this!, sen on kirjoittanut Christophe Porteneuve ja julkaissut The Pramatic Programmers. Mootools-sovelluskehyksestä kirjoitettu kirja on taas nimeltään: MooTools Essentials: The Official MooTools Reference for JavaScript™ and Ajax Development,sen on kirjoittanut Aaron Newton ja julkaissut Apress.

Sovelluskehyksellä tarkoitetaan Ajaxin yhteydessä yleensä erilaisia JavaScript kirjastoja, joissa on valmiudet Ajax-toteutuksiin. Puhutaan siis selainpäässä toimivasta kehyksestä. Tällaisten kirjastojen käytössä on se hyvä puoli, että niiden käyttö on valmiiksi testattua ja ne toimivat hyvin eri selaimilla. Tämä onkin tärkeää kehitettäessä Ajax-pohjaisia sovelluksia. (Holzner 2007, 5.)

Sovelluskehystä käyttämällä ei voi välttyä itse JavaScript-koodin kirjoittamiselta, mutta ne helpottavat urakkaa huomattavasti. Kehittäjän ei tarvitse ottaa samalla tavoin huomioon esimerkiksi selain-poikkeavuuksia JavaScriptin suorittamisessa, koska ne on jo otettu kehyksessä huomioon. Kehittäjä pystyy tällöin paremmin keskittymään itse sovelluksen kehittämiseen, kun JavaScript ongelmien ratkomiseen. Valmiit kehykset sisältävät myös paljon valmiita ominaisuuksia joita on helppo ottaa käyttöön rakennettavassa sovelluksessa. Suurin osa kehyksistä on myös ilmaisia, mutta toisaalta kaikki eivät ole kirjoitettu parhaalla mahdollisella tavalla, koska ovat yleensä harrastelijoiden luomuksia. Ennen kehyksen valintaa kannattaakin tutustua valikoimaan ja valita näiden joukosta itselleen sopivin. (Holzner 2007, 5.)

JavaScript-sovelluskehykset sisältävät usein myös erilaisia efektikirjastoja. Näiden avulla voidaan piristää tai selkiyttää käyttöliittymää erilaisin JavaScriptillä toteutettujen efektien- ja käyttöliittymäkomponenttien avulla. (Ekonoja & Lahtonen & Mäntyala 2008.)

3.8 Ajax ja käytettävyys

Käytettävyyden suhteen Ajax-tekniikka on ollut paljon esillä eri tilanteissa. Tämän huomaa kun tutkii aihetta Internetin hakukoneiden avulla. Tuloksia löytyy paljon ja kannanotot ovat usein jyrkkiä, kuten esimerkiksi Jakob Nielsenillä artikkelissaan: Why Ajax Sucks (Most of the Time).

Ajax tekniikan käyttö ei aina ole suositeltavaa. Parhaiten se soveltuu juuri erilaisiin Internet-sovelluksiin ja käyttöliittymiin. Jakob Nielsen on kirjoittanut tästä artikkelin: Why Ajax Sucks (Most of the Time). Toisaalta Nielsen huomauttaa artikkelissa, että "käyttäjät, jotka todella tietävät mitä ovat tekemässä, voivat tehdä Ajaxilla toimivia ratkaisuja". (Nielsen 2005.)

On tärkeää, että käyttäjä tietää mistä on kysymys. Käyttöliittymä ei saa tuoda käyttäjille yllätyksiä, koska ne hämmentävät. Käyttäjälle pitää tuoda selkeästi esille, jos käyttöliittymään on tuotu jotain uutta, tai ohjelman suorituksessa on tapahtunut virhe. Staattisen web-sovelluksen käyttäjät ovat tottuneet, että joka latauksen yhteydessä selain osoittaa lataavansa uutta sivua (selaimen oikean yläkulman logo pyörii) ja sivu latautuu uudelleen. Ajax sovelluksissa ei tällaista tapahdu, koska HTTP-kutsuhan toteutetaan perinteisestä poiketen JavaScriptin avulla taustalla. Tällöin pitää huomioida se, että lataamistoimenpiteet esitetään muulla tapaa, etteivät käyttäjät hämäänny. Pienemmissä latauksissa yleisesti käytetty visuaalinen vinkki voi olla niin kutsuttu Thobbler-kuva, joka esittää, että tiettyä kohtaa sivusta ladataan. Tällä tarkoitetaan pientä animoitua kuvaa, joka esittää latausprosessia ja saa käyttäjän ajattelemaan, että jotain tapahtuu, eikä suoritus ole jumiutunut. Suuremmissa latauksissa voidaan käyttää edistymispalkkia, jonka toteutus on kuitenkin hiukan hankalampaa, koska latausaikoja voi olla vaikea arvioida. (Lauria 2008, 19-20.)

Ajax-tekniikasta puhuttaessa nousee esiin myös erilaisia käytettävyysongelmia. Yksi Ajaxin ongelma on, että se tekee selaimen palaa-napista hyödyttömän. Nielsenin mukaan tämä on selainten toiseksi käytetyin ominaisuus. Tätä voidaan verrata esimerkiksi aikaisemmin suosiossa olleeseen kehys-tekniikkaan (frames). Tämä kannattaa huomioida sovellusta kehitettäessä. Ongelman voi tosin kiertää, mutta se ei ole kovin helppoa. Toisaalta joskus palaa-napin käyttö varsinkin Internet-sovelluksista puhuttaessa voi aiheuttaa ongelmia tavalla tai toisella (esimerkiksi pankkien sovellukset). Palaa-napin käyttöä tarvittaessa kyllä pystyy hallitsemaan myös Ajax-sovelluksissa. Tästä esimerkkinä on Googlen Gmail-sähköpostisovellus. Joissakin JavaScript-sovelluskehyksissä on huomioitu palaa-napin käyttö. (Ullman & Dykes 2007, 19.)

Muita ongelmia tekniikan käytössä ovat muun muassa selainten kirjanmerkkien käyttö, URL-osoite kun ei vaihdu toimintoja tehtäessä. Tähänkin on toisaalta kehitetty erilaisia tekniikoita, joilla asiaa pystytään hallitsemaan. Tämä tarkoittaa kuitenkin sitä, että selaimen URL-

osoitteita on manipuloitava JavaScriptin avulla. Samalla Ajax-tekniikan käyttö hankaloittaa myös hakukoneiden toimintaa, koska ne eivät voi kunnolla indeksoida Ajax-tekniikalla toteutettuja sivustoja. Verkkosovelluksissa tämä ei ole kuitenkaan ongelma, koska niitä ei edes haluta indeksoida hakukoneissa, eikä myöskään huomioida kirjanmerkeissä. (Ullman & Dykes 2007, 19–20.)

Ongelmana on myös selaimien soveltuvuus Ajax-pohjaisille web-sovelluksille. Järjestelmää kehitettäessä kannattaa huomioida sovelluksen kohderyhmä. Jos käyttäjät käyttävät vanhoja selaimia, tai esimerkiksi mobiililaitteita, kannattaa tekniikan käyttöä tosiaan harkita. Toisaalta tämä ongelma on pikkuhiljaa katoamassa, kun käyttäjät ottavat käyttöön uudempia selaimia ja mobiililaitteetkin kehittyvät. (Nielsen 2005)

Yhtenä käytettävyysongelmana Ajaxin suhteen voidaan pitää myös sitä, ettei Ajax-pohjaista sivustoa voi selata ”Offline-tilassa”. Tämä tarkoittaa sitä, että sivusto ei toimi, jos pienikin verkkokatkos yllättää. Tosin jos kysymys on web- ja tietokantapohjaisesta sovelluksesta, eikä Internet-sivustosta on selvää, että yhteys pitää olla muodostetuttu sitä käytettäessä. Tässä vaiheessa on myös hyvä huomioida, että Ajax-pohjaista sovellusta käytettäessä on varottava, ettei selaimen välimuisti (cache) sotke sovelluksen toimintalogiikkaa jättämällä välimuistiin tiettyjä sivuja, tai sovelluksen osia. (Holzner 2006, 328–329.)

Ajax-kehityksessä merkittävä asia on se, että mitä kehitetään. Jos tosiaan ollaan kehittämässä ihan tavallista Internet-sivustoa tai esimerkiksi yrityksen esittely sivustoa on mietittävä kannattaako Ajaxia käyttää. Pahimpia ongelmia ovat tässä juuri hakukoneiden indeksointeihin liittyvä kysymykset. Yleensä kuitenkin halutaan, että hakukoneet löytävät yrityksesi sivut. (Nielsen 2005.)

4 TARJOUSJÄRJESTELMÄ JA SEN TOIMINNAN KUVAUS

Jotta lukija saisi jonkinlaisen käsityksen järjestelmästä, on syytä esitellä sitä hiukan ennen kuin päästään käsiksi itse Ajax-toteutukseen. Järjestelmää suunniteltaessa otettiin huomioon, että järjestelmän käyttöliittymään tullaan hyödyntämään Ajax-tekniikkaa. Järjestelmän voi jakaa kahteen eri pääosioon joita ovat:

1. Tuotehallinta (tässä tapauksessa kiinteistöjen tai toimitilojen hallinta)
2. Tarjoustyökalu

Tuotehallinnan rooli sovelluksessa on tärkeä, mutta työn kannalta ei niinkään merkittävä, koska työ on rajattu ainoastaan tarjoustyökalun Ajax-toimintojen kehittämiseen. Tuotehallin-

nasta on kuitenkin hyvä nostaa esiin tuotetyypit, koska tämän avulla vaikutetaan siihen, mitä tarjoustyökalun puolella tulostetaan.

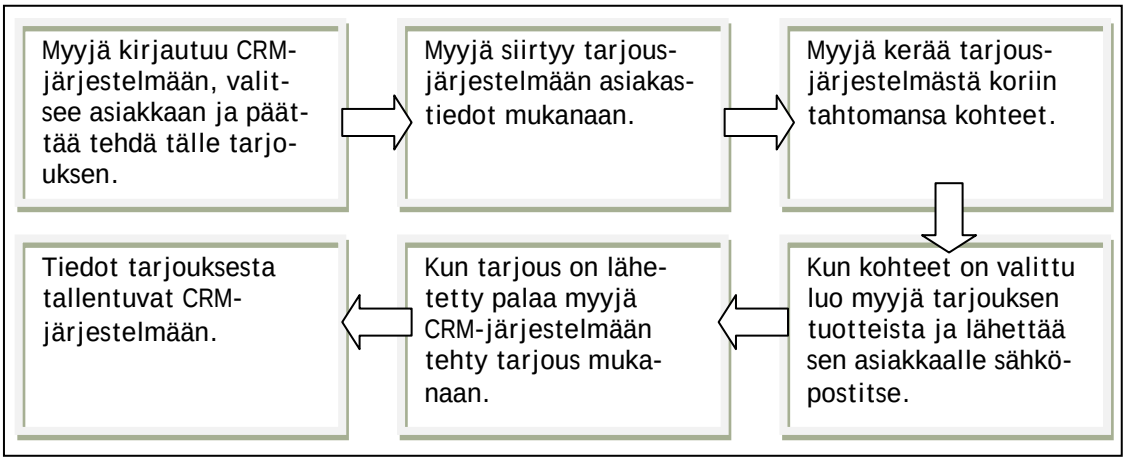
4.1 Tuotetyypit

Koska järjestelmän pitää olla mahdollisimman dynaaminen, on tärkeää, että tuoterekisteri voidaan muokata juuri sellaiseksi kuin tarve vaatii. Tähän järjestelmässä käytetään tuotetyyppejä. Tuotetyyppien avulla järjestelmässä määritetään minkälaisia tuotteita tuoterekisteri pitää sisällään. Tuotetyyppien avulla määritetään myös ulkoasut esimerkiksi tuotekorteille jotka tarjoustyökalussa näkyvät ja määritetään hakulomakkeet.

Tuotetyypin avulla luodaan pohja tuotekortille ja kerrotaan miten se tulostuu erilaisissa näkymissä. Tuotetyypit toimivat siis koko tuoterekisterin pohjana. Tuotetyyppiä voidaan myös muokata ”lennosta”, jolloin esimerkiksi puuttuvat kentät voidaan lisätä jälkikäteen järjestelmän, tarvitsematta päivittää tietokantoja. Tuotetyyppien avulla voidaan myös määrittää hakuja, joita tietokannasta voidaan tehdä. Tuotetyypit voivat olla toisilleen myös alisteisia. Kyseisessä tarjousjärjestelmässä tuotetyyppeinä ovat kiinteistöt ja niissä olevat toimitilat.

4.2 Tarjoustyökalu

Kuten aikaisemmin jo mainittiinkin, on Ajax-toteutuksen kohteena tarjousten tekemiseen ja niiden lähettämiseen tarkoitettu verkkosovellus, joka liitetään yhteistyökumppanin kehittämään CRM-järjestelmään. Liitos CRM-järjestelmään on tärkeää, koska myyjät siirtyvät CMR-järjestelmästä tarjoustyökaluun, tämä käy ilmi taulukosta 3. Tällöin myyjän ja tarjouskohteen tiedot siirtyvät järjestelmien välillä. Tarjouksen tehtyään myyjä siirtyy takaisin CRM-järjestelmään. Järjestelmät luovat myyjälle kokonaisuuden, jolla hänen on helppo seurata mitä kullekin asiakkaalle on tarjottu.



Kuvio 3: Tarjoustyökalun toimintamalli

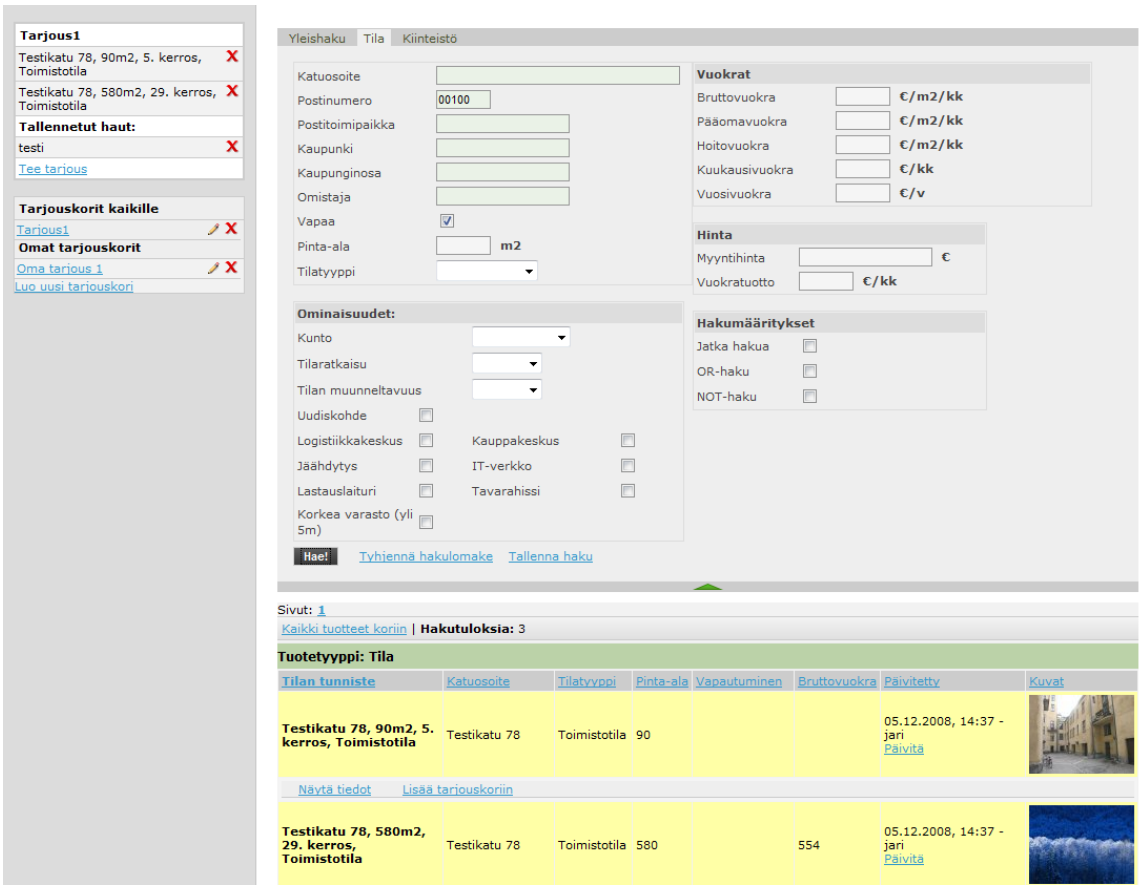
Tarjoustyökalu koostuu kahdesta eri osiosta, joita ovat:

1. Tarjousten koostaminen
2. Tarjouksen tekeminen ja lähettäminen

Tarjouskorien hallinta on olennainen osa tarjoustyökalun toimintaa. Tarjouskoriin kerätään tuoterekisteristä tuotteita ja lopuksi tästä koostetaan tarjous. Järjestelmässä koreja voi olla useita ja käyttäjä voi niitä itse määrittää, joten kaikilla käyttäjillä on omat korinsa. Tarjouskorien toteuttamisessa Ajax-tekniikan rooli on luultavasti merkittävä, koska tämän avulla voidaan toteuttaa ne todella nopeiksi.

Tuotehaku on myös merkittävässä osassa järjestelmää. Tuotehaun toimivuus on tärkeää, koska järjestelmä voi sisältää kymmeniätuhansia tuotteita. Tarjouskoreihin voidaan tallentaa myös hakuja, jotta yleisimmät haut tapahtuisivat nopeasti.

Tarjoustyökalun päänäkymä on seuraavassa kuvassa (kuva 5) ja se koostuu viidestä osasta.



Kuva 5: Tarjousjärjestelmän päänäkymä

Tarjousjärjestelmän päänäkymä koostuu viidestä eri osasta. Kaikki tulostuksessa näkyvät osiot määritetään tuotetyyppien avulla, eli tuotetyypeissä kerrotaan miten ne tulostetaan ruudulle:

1) Aktiivinen tarjouskori, johon listalta koriin siirretyt tuotteet tulevat. Tarjouskoriin voi myös tallentaa hakutuloksia, jolloin tietyt haut voidaan suorittaa todella nopeasti. Tallennettuja hakuja voi olla useita ja yhtä koria voidaan esimerkiksi käyttää vain hakujen tallentamiseen.

2) Tarjouskorien hallinta, jonka avulla luodaan uusia, aktivoidaan ja poistetaan tarjouskoreja.

3) Haku, jonka avulla rekisteristä etsitään tuotteita.

4) Tuotelistaus, johon hakutulokset listautuvat. Hakutuloksissa jo tarjouskorissa olevat korostetaan.

5) Tuotekortti, joka aukeaa kun listalta painetaan ”Näytä tiedot” -linkkiä.

5 TOTEUTUS

Tässä osassa on tarkoitus käydä läpi sovelluksen toteutukseen liittyviä kysymyksiä ja kerrotaan itse Ajax-toteutuksesta. Aluksi tutkitaan lyhyesti palvelinpuolen sovellusympäristöä ja valitaan sopiva sovelluskehys JavaScript alustaksi.

5.1 Palvelinpuolen sovellusympäristö

Sovellusympäristö on tärkeä osa sovellusta. Järjestelmän pohjana käytetään Isoltan kehittämää ohjelmointikehystä, joka tarjoaa valmiit puitteet ohjelmiston kehittämiseen. Tärkeimmät edut ovat muun muassa tietokantojen käsittelyyn ja käyttäjien hallintaan rakennetut valmiit toiminnot. Sovellusympäristön päälle on aikaisemmin kehitetty muun muassa sivustonhallintatyökalu ja monia tähän liittyviä moduuleita, kuten uutistyökalu, kalenteri ja kuvagalleria. Tähän ympäristöön tarjousjärjestelmä liitetään uutena laajana moduulina.

Ajax-toteutusta tehdessä jouduttiin järjestelmän pohjaa kuitenkin hieman muokkaamaan, jotta Ajaxin avulla toteutetut HTTP-kutsut voitaisiin käsitellä oikealla tavalla. Tämä tarkoittaa sitä, että Ajax-pyyntöjen yhteydessä ei voitu palauttaa koko HTML-sivun sisältöä, kuten ennen, vaan selaimeen palautetaan ainoastaan tietyn kohdan sisältö GET- tai POST-parametrien mukaan.

5.2 Selainpuolen sovelluskehykset

Koska JavaScript on avainasemassa Ajax-pohjaisen sovelluksen toteutuksessa, tulee mieleen kannattaako kaikkea JavaScript-koodia kirjoittaa itse vai kannattaako turvautua valmiisiin sovelluskehyksiin. Kuten aikaisemmin mainitsin, löytyy Internetistä useita sovelluskehyksiä, jotka ovat tutustumisen arvoisia.

Tutkittuani asiaa valitsin käyttööni Prototype-nimisen JavaScript-sovelluskehyksen. Tämän ympäristön valitsin aikaisemman kokemuksen perusteella, koska olin tähän törmännyt aikaisemmissa projekteissa ja se on laajasti käytössä (Käytössä esimerkiksi Applen: www.apple.com ja Ikean: www.ikea.fi, Internet-palveluissa). Tämän pystyy selvittämään katsomalla palveluiden lähdekoodia selaimesta). JavaScript-toteutukset voivat olla todella haastavia ja niiden kirjoittamiseen kuluu helposti paljon aikaa. Koodia kirjoittaessa on huomioitava muun muassa eri selainten toiminnallisuudet ja tavat toteuttaa jokin tietty asia. Kun on pohja mikä on jo valmiiksi mietitty, koodin kirjoittaminen nopeutuu ja helpottuu huomattavasti. Prototype on lisäksi hyvin dokumentoitu ympäristö ja sillä on paljon käyttäjiä. Valmiita koodiesimerkkejä löytyy siis paljon. Kehys löytyy osoitteesta: http://www.prototypejs.org. Kuten jo aikaisemmin mainitsin, kyseisen sovelluskehyksen käytöstä on kirjoitettu myös kirja, joka sisältää paljon esimerkkejä ja muuta materiaali ympäristön käytöstä**.**

Prototypen rinnalle valitsin script.aculo.us- nimisen efekti- ja käyttöliittymäelementti kirjaston. Script.aculo.us toimii yhdessä Prototypen kanssa hyödyntäen sen kirjastoja efekteissään ja käyttöliittymäelementeissään. Tämän avulla voidaan toteuttaa helposti esimerkiksi raahaa ja pudota toimintoja (Drag and Drop). Script.aculo.us löytyy osoitteesta: http://script.aculo.us/.

5.3 Ohjelmointityö: JavaScript ja PHP

Ohjelmoinnin toteutus voidaan tässä projektissa jakaa kahteen eri osaan: Palvelin ohjelmointi, jossa käytössä on PHP ja selainpään ohjelmointi, jossa käytetään JavaScript-kieltä. Ajax-pohjaisessa järjestelmässä molemmat ovat tärkeässä roolissa: JavaScriptin avulla muodostetaan yhteys palvelimelle, palvelimella tieto prosessoidaan ja se palautetaan selaimeen. Lopuksi JavaScriptin avulla tieto esitetään selaimen DOM-puussa.

JavaScriptin rooli Ajax-pohjaisessa sovelluksessa on merkittävä. Sen avulla rakennetaan oikeastaan koko järjestelmän toiminnallisuus selainpäähän, tai vähintäänkin Ajax-tekniikkaan liittyvät osiot. Ohjelmointityössä käytettiin apuna Prototypeä, joka osaltaan helpottaa koodin kirjoittamista ja vähentää kirjoitettavan koodin määrää, mutta toki koodia pitää silti itse kirjoittaa.

JavaScript koodin toteuttaminen aloitettiin luomalla Ajax-toiminnallisuudet, joka käytännössä tarkoittaa XMLHttpRequest-olion muodostamista ja sen käsittelyä. Tämä mahdollistaa palvelimelle tehtävät asynkroniset haut. Seuraavaksi toteutettiin niin kutsuttu init-funktio, joka suoritetaan aina, kun järjestelmä avataan tai selain päivitetään (onLoad). Tämän avulla käynnistettiin siis ensimmäiset Ajax-haut ja näin selaimen ruutuun ilmestyvät esimerkiksi korit. Kyseisessä tarjousjärjestelmässä JavaScriptin rooli on toimia ikään kuin toimintojen ohjaajana. JavaScript toimii kerroksena käyttöliittymän ja palvelimen välissä. Kun perinteisessä web-sovelluksessa klikataan linkkiä, kutsuu se palvelimelta tiettyä HTML-sivua, tai vaikka PHP-tiedostoa, jota ohjataan GET-parametrien avulla. Tarjousjärjestelmässä linkin painaminen suorittaakin JavaScript funktion, joka sitten käynnistään jonkin Ajax-toiminnallisuuden.

JavaScript kehitystyön ohella pitää huomioida myös palvelin-puolen ohjelmointi. Kun jokin Ajax-toiminto kirjoitetaan JavaScriptiin, pitää se huomioida luonnollisesti myös palvelinpäässä. Esimerkiksi sovelluksen käynnistyessä ladataan ostoskori seuraavalla tavalla:

1. Init-skripti suoritetaan jäjestelmän käynnistyessä (onLoad).
2. Skriptissä kutsutaan JavaScript-funktiota, johon on määritetty mitä palvelimelta ladataan.
3. XMLHttpRequest-olio muodostetaan ja tehdään palvelinkutsu.
4. Palvelin ottaa kutsun vastaan, aivan kuten periteisellä tavalla toteutetussa web-palvelussa.
5. Palvelin ohjaa kutsun oikeaan paikkaan GET-parametrien avulla.
6. Palvelimella suoritetaan funktio, joka muodostaa ostoskorin ja palauttaa sen selaimelle.
7. XMLHttpRequest-olio ottaa tiedon vastaan.
8. Data tulostetaan selaimeen määritetyn HTML-tagin sisälle esimerkiksi käyttämällä JavaScript:n innerHTML-ominaisuutta.

Tarjousjärjestelmää kehitettäessä piti siis ottaa huomioon sekä JavaScript, että PHP-koodi. Kun jokin Ajax-toiminnallisuus tehtiin selaimeen, tehtiin se myös PHP-koodiin. Oikeastaan järjestelmää kehittäessä tämä meni toisin päin. Ensin toiminto toteutettiin palvelimelle. Vasta tämän jälkeen tehtiin lisäykset JavaScript-koodiin, joka suoritti toiminnon selaimessa. Tämä tehtiin vasta, kun koodia piti päästä testaamaan.

5.4 Ajax-tekniikan käyttökohteet toteutuksessa

Tässä kappaleessa kerrotaan missä osa-alueissa Ajax-tekniikkaa käytettiin sovellusta kehitettäessä ja mitä etuja ratkaisut toivat sovelluksen toimintaan.

Ajax tekniikkaa hyödynnettäessä on mietittävä, miten se hyödyttää sovellusta ja sen käytettävyyttä tai nopeuttaako se järjestelmän toimintaa. On tärkeää tietää, milloin sitä voi käyttää ja milloin ei. Ajax-tekniikan käytön vain sen itsensä vuoksi ei saisi olla käytön päällimmäisenä motiivina, vaan sen on tuotava oikeasti hyötyä. (Asleson & Schutta 2006, 20)

5.4.1 Tarjouskorit

Tarjouskorit ja niiden hallinta on ihanteellinen kohde Ajax-pohjaiselle käyttöliittymälle, koska siltä vaaditaan nopeutta ja tehokkuutta. Kun tuote lisätään koriin, on turha ladata koko sivua uudelleen, se vain hidastaisi järjestelmän käyttöä. Parempi tapa on päivittää vain tietty osa sivusta, eli tässä tapauksessa tarjouskori. Data siirtyy taustalla asynkronisesti häiritsemättä sivun muuta toimintaa. Tuotetta lisättäessä koriin on tieto tallennettava tietokantaan varmuuden ja turvallisuuden säilyttämiseksi. Tarjouskorit jakautuvat kahteen osaan, joita ovat: aktiivinen kori ja korien hallinta. Koreihin liittyvät toiminnot ovat keskenään hyvin samantapaisia ja seuraavaksi tarkastellaan lähemmin, sitä miten tuote lisätään tarjouskoriin.

Kuva 6: Tarjouskorien hallinta

Tuote lisästään koriin painamalla linkkiä, joka suorittaa JavaScript-komennon. Komento suorittaa JavaScript funktion, joka puolestaan muodostaa Ajax-kerroksen, jonka avulla otetaan yhteys palvelimeen (XMLHttpRequest-olio). Tärkein tieto Ajax-kutsulle on url-osoite, johon kutsu lähetetään. URL-osoitteeseen liitetään kaikki tiedot jotka tarvitaan, jotta se osataan palvelinpäässä ohjata oikeaan toimintoon. Kuten aikaisemmin mainittiin, kutsun URL-osoite on samanlainen kuin normaalissakin palvelinkutsussa. Tarjouskori tapauksessa käytetään GET-metodia kutsun lähettämiseen, koska dataa ei lähetetä paljon vaan ainoastaan tieto siitä, mikä tuote koriin lisätään. Kutsun URL-osoite voi näyttää seuraavanlaiselta:

index.php?module_id=3&get_page_html=1&load=add_product_to_cart &product_id=12345;

Palvelin vastaanottaa kutsun ja suorittaa kutsussa määritetyn toiminnon. Kutsussa näkyy load arvona add_product_to_cart, jolloin se pystytään ohjaamaan oikeaan toimintoon, eli tuotteen lisäämiseen. Product_id parametrilla kerrotaan taas tuotteen id. Get_page_html kertoo järjestelmässä, että kysymyksessä on ajax-kutsu jolloin selaimeen palautetaan ainoastaan tarvittava sisältö eikä koko sivun HTML-sisältöä, kuten normaalissa palvelinkutsussa. Module_id tarkoittaa järjestelmän sisäistä ohjausta siitä, mihin järjestelmän moduuliin kutsu ohjataan.

Kun kutsu on ohjattu oikeaan paikkaan, suorittaa PHP-koodi tarvittavat toiminnot, jotta tuote saadaan tallennettua ostoskori tietokantatauluun. Tallennuksen jälkeen palauttaa palvelin selaimeen päivitetyn ostoskorin kokonaisuudessaan. Palautus selaimelle tapahtuu XML-muodossa ja se käsitellään selaimessa JavaScriptin avulla käyttämällä XSL-tyylitiedostoa (tästä lisää myöhemmin). Selain ottaa siis tiedon vastaan ja JavaScriptin avulla päivitetään se sitten käyttöliittymään. Näin korissa on lopulta uusi tuote, joka näkyy käyttäjälle.

Jotta käyttäjä tietää, että jotain on tapahtunut, tehostetaan JavaScriptin avulla korostamalla koriin luotu tuoterivin tausta keltaisella värillä, joka liukuu takaisin pohjaväriin. Tieto näkyy ostoskorin lisäksi myös tuotelistauksesta ja tuoterivin tausta värjätään keltaisella. Näin käyttäjä näkee jo selaillessaan, mitkä kiinteistöt ovat jo korissa. Tämä tehdään dynaamisesti JavaScriptin avulla. Selaimen sivua ei siis missään välissä päivitetä uudelleen, vaan ainoastaan tietty alue.

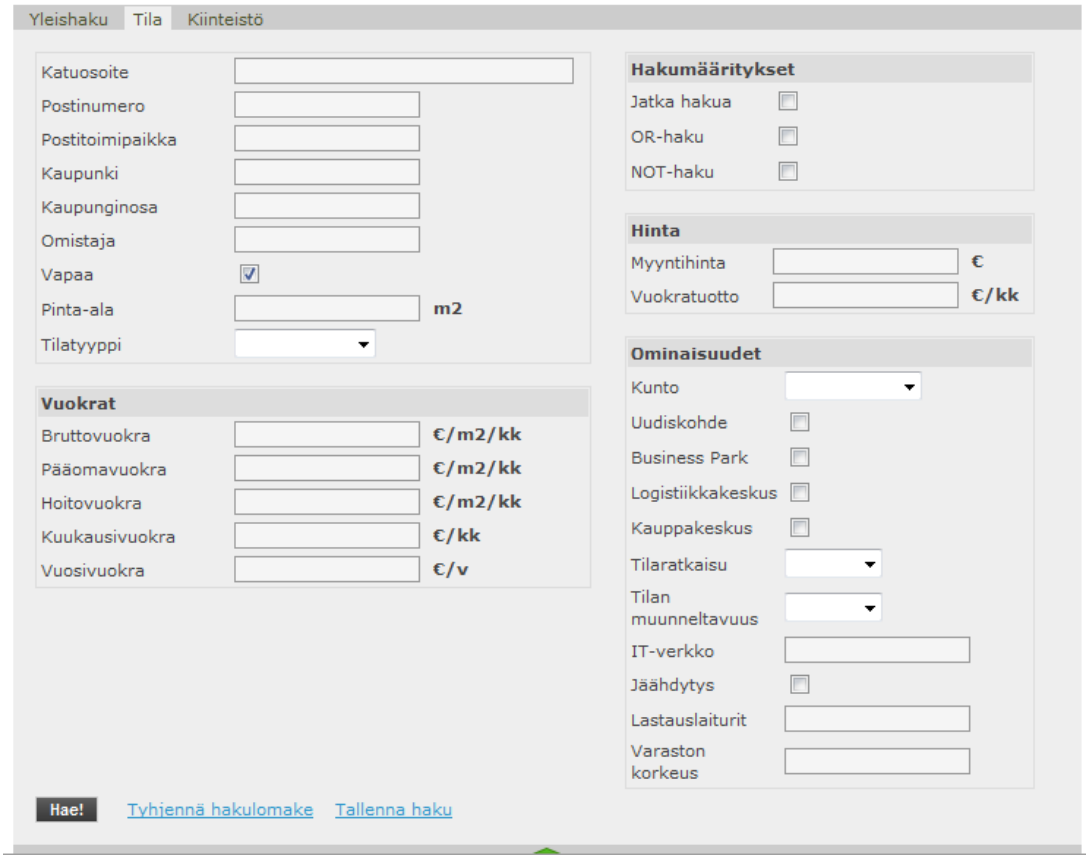
5.4.2 Tuotehaku ja tuotteiden listaus

Tuotehaku on tärkeä osa järjestelmää. Sen avulla etsitään tietokannasta erilaisia kohteita. Järjestelmän dynaamisuuden kannalta hakulomakkeen määrittäminen tehdään jokaiselle tuotetyypille erikseen (katso kohta Tarjousjärjestelmä ja sen toiminnan kuvaus). Hakulomakkeet voivat siis näyttää miltä tahansa ja ne määritellään tuotetyyppiin määritettyjen kenttien perusteella.

Hakulomakkeessa Ajax-tekniikkaa hyödynnetään useammassakin kohdassa. Kohteita ovat itse haun toteuttaminen ja hakutuloksen esittäminen ja hakulomakkeella olevat kentät, joissa käytetään automaattitäydennystä. Ajaxin avulla voidaan myös tallentaa haku, jolloin yleisimpiä hakuja voidaan tehdä nopeasti.

Haku suoritetaan myös Ajax-kutsuna, mutta kun tietoa palvelimelle lähetetään enemmän, käytetään GET-metodin sijaan POST-metodia. Prototype tuo mukanaan kätevän tavan käsitellä lomakkeita JavaScriptin avulla, joten sen käsittely on helppoa. Painettaessa Hae nappia, käynnistetään JavaScript toiminto, joka hakee funktion parametrina mukana tulleen lomakkeen se id:n perusteella. JavaScript hakee lomakkeen kaikki kentät ja se muutetaan Prototy-

pen serialize-funktion muotoon, joka voidaan lähettää suoraan Ajax-kutsun mukana palvelimelle.



Kuva 7: Tarjousjärjestelmän hakulomake

Automaattitäydennykset on toinen kohta, johon Ajax-tekniikkaa hyödynnetään. Tätä toiminnallisuutta ei kuitenkaan vielä tässä vaiheessa rakennettu, vaan se jää myöhemmäksi. Automaattitäydennyksen hyötynä on, kun käyttäjä kirjaa hakulomakkeeseen esimerkiksi kaupunkia, hän voi valita aukeavalta popup-listalta oikean vaihtoehdon. Kun käyttäjä on valinnut kaupungin, tulisi järjestelmän ehdottaa kaupunginosaa kyseiselle kaupungille. Tämä toiminnallisuus rakennetaan kuitenkin vasta syksyn 2008-aikana.

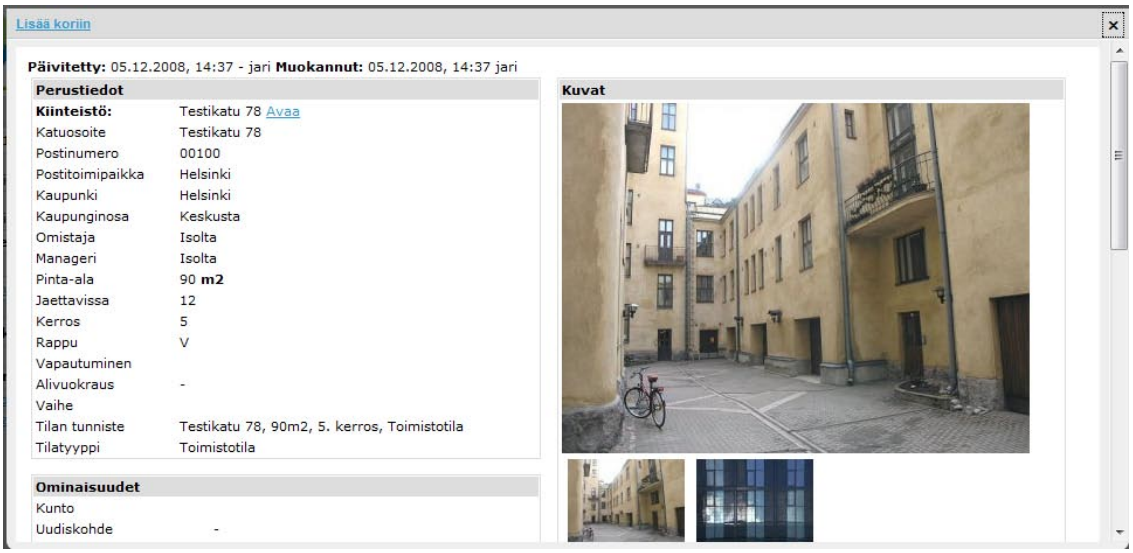
5.4.3 Tuotekortit

Tuotekorttien avulla myyjä voi tarkastella tuotteen tietoja listausta tarkemmin. Tuotekortissa Ajaxin rooli on toisenlainen kuin ostoskorissa, koska tietoja selatessa ei tietokantaan tarvitse lisätä mitään, vaan ainoastaan hakea.

Tuotekorteissa myös dynaamisuus on tärkeää. Tuotekorteissa on saatavilla useita eri näkymiä työkalussa, jotka määritetään tuotetyyppeihin. Myyjällä on oma näkymänsä, jonka kautta

myyjä saa kaiken mahdollisen informaation tuotteesta. Tarjousta tehtäessä myyjä lähettää sähköpostitse asiakkaalle tarjouksen, jonka kautta asiakas pääsee myös katselemaan tuotekortteja pelkistettyyn näkymään.

Tuotekortteihin tehtiin Ajax-toteutus, koska niitä haluttiin selata nopeasti. Kortin avautuminen ei saa kestää pitkään ja samalla haluttiin, että hakulistaus säilyy taustalla, koska haku voi olla hyvinkin monimutkainen ja sen uudelleensuorittaminen kuluttaisi turhaan aikaa ja kuormittaisi palvelinta.



Kuva 8: Tarjousjärjestelmän tuotekortti

Tuotekortti avautuu JavaScriptin ja css:n avulla muodostettuun ikkunaan. Ikkuna ei kuitenkaan ole normaali popup, vaan ikään kuin kerros, joka muodostuu selaimeen. Avautuvaan ikkunaan tulostetaan sitten Ajaxin avulla palvelimelta haettu tuotekortti. Tuotekortin voi tällöin sulkea nopeasti ja käyttäjä voi jatkaa listan selailua täysin samoista asetuksista kuin aikaisemminkin.

5.5 XML:n ja XSLT:n rooli järjestelmässä

Järjestelmää rakennettaessa eteen tuli ongelma miten saadaan tarjousjärjestelmästä mahdollisimman dynaaminen eri käyttötarkoituksia varten. Ajatellaan vaikka ostoskoria kiinteistöalalle tarkoitettuun järjestelmään ja esimerkiksi varaosakauppaan erikoistuneen yrityksen järjestelmänä. Luultavasti korista tarvitsee nähdä erilaisia tietoja. Kiinteistöalan järjestelmässä ei varmasti tarvita tietoa siitä, montako kertaa tuote (tässä tapauksessa kiinteistö) on koriin laitettu, koska kyseistä kiinteistöä on vain yksi kappale. Varaosia taas voi olla useita jolloin on tärkeää, että niitä voidaan tarjota asiakkaalle useita kappaleita. Korissa pitää täl-

löin näkyä tieto siitä, monta kappaletta kyseistä tuotetta on koriin laitettu. XSLT:n avulla voidaan päättää miten palvelimelta saatu XML-data muotoillaan ja mitä siitä selaimessa näytetään.

Tätä muunneltavuutta varten tarjousjärjestelmässä alettiin miettiä XSLT:n käyttöä Ajax-tulostuksissa. XSLT:n käyttö on helppo perustella, koska sen käyttö on helppoa ja nopeaa, jolloin sitä voidaan käyttää erilaisiin yksilöllisiin ratkaisuihin. Lisäksi se on W3:n standardoima ja selaimista löytyy hyvin tuki sen käyttöön (lukuun ottamatta Apple:n Safari selainta, joka ei tue JavaScriptillä ladattuja XSLT-tiedostoja). XSLT:n etuna on se, ettei palvelimen tarvitse välttämättä tietää, mitä selainpäässä tulostetaan. Lähetetään vain data XML-muotoisena pakettina selaimeen ja selaimessa käsitellään se XSLT:n avulla oikeaan muotoon ja päätetään vasta silloin, mitä tulostetaan. Esimerkkinä tästä voidaan käyttää tarjouskoria, jonka tulostuksessa mietittiin ensimmäisen kerran XSLT:n käyttöä.

Tarjouskoria ladattaessa suoritetaan Ajax-kutsu palvelimelle, eli muodostetaan XMLHttpRequest-olio, jonka tarkoitus on lähettää HTTP-kutsu palvelimelle. Kutsu lähetetään ja käsitellään palvelimella. Palvelin palauttaa selaimelle tarjouskorin tietoineen XML-jäsennettynä datana. Tarjouskorin XML-data voi näyttää esimerkiksi seuraavanlaiselta:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<cart id="1">
     <name>Testi xml cart</name>
     <make_offer_url>http://www.isolta.fi</make_offer_url>
     <delete_product_url>http://www.isolta.fi</delete_product_url>
     <products>
          <product id="12345">
               <paroduct_name>Testituote</paroduct_name>
               <quantity>1</quontity>
          </product>
          <product id="54321">
               <paroduct_name>Testituote</paroduct_name>
               <quantity>3</quontity>
          </product>
     </products>
</cart>
```

Kuvio 4: Tarjouskori XML-muodossa

Kun selain on vastaanottanut XML-datan, muunnetaan se HTML-muotoon XSLT:n avulla, taulukossa 5 esitetyllä tavalla.

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" encoding="iso-8859-1" indent="yes" />
     <!-- Tulostetaan ostoskorin nimi -->
     <xsl:template match="cart">
          <div id="collection" style="position:relative;">
               <div id="collections-title">
                    <xsl:value-of select="name" />
               </div>
               <xsl:apply-templates select="products" />
          </div>
     </xsl:template>
     <!-- Tulostetaan tuoterivit -->
     <xsl:template match="product">
          <div class="product-row" style="float:left; width:97%;">
               <div class="name">
                    <xsl:value-of select="pname" /></div>
               </div>
     </xsl:template>
</xsl:stylesheet>
```

Kuvio 5: XSLT-tyylitiedosto XML-dokumentin muotoiluun

Tyylitiedosto ladataan JavaScriptin avulla taulukon 6 mukaisella tavalla. Sen avulla käsitellään palvelimelta saatu XML-data ja muunnettu tieto tulostetaan selaimeen. Kuvassa 9 on esimerkkien mukainen ostoskori.

```
var xml = (new DOMParser()).parseFromString(responseXML, "text/xml");
xsl=loadXMLDoc("cart.xsl");
xsltProcessor=new XSLTProcessor();
xsltProcessor.importStylesheet(xsl);
resultDocument = xsltProcessor.transformToFragment(xml,document);
document.getElementById("cart").appendChild(resultDocument);
```

Kuvio 6: Esimerkki XSLT-muunnoksen tekemisestä JavaScriptin avulla

Kuva 9: XSLT:n avulla muodostettu tarjouskori

Samantapaista ratkaisua voidaan hyödyntää useissa eri tilanteissa määrittämällä XSLT-tiedosto palvelimelta saatavan XML-datan mukaisesti. Tällaisessa järjestelyssä on etuna se, että palvelimen ei tarvitse tietää millaisena selaimeen palautettava tieto esitetään selaimessa, vaan se voidaan tehdä tapauskohtaisesti.

XSLT-muunnoksia voidaan tehdä myös PHP:n avulla ennen kuin tieto palautetaan selaimeen palvelimelta. Tällöin muunnoksessa käytetään apuna PHP:n XSL-kirjastoja. Jos muunnos tehdään jo palvelimella, voidaan data palauttaa käyttöliittymään suoraan HTML-muodossa. Hankaluutena on tällöin se, ettei dataa voida muokata kovinkaan tehokkaasti enää selaimessa ja se ei ole enää niin dynaamista, kuin voisi olla.

5.6 Ajax-toteutuksessa vastaan tulleita ongelmia

Ajax-toteutuksen yhteydessä törmättiin useisiin ongelmiin, mutta kaikkiin löydettiin jonkinlainen ratkaisu. Tässä muutama merkittävä ongelma, joita helposti tulee vastaan nimenomaan Ajax-sovellusta kehitettäessä.

Ensimmäinen ongelma oli selaimen välimuisti. Ladatessaan erilaisia sivustoja, selain tallentaa tiedot välimuistiin, jolloin esimerkiksi sivun valikon linkkiä painettaessa selain ei haekaan kaikkia tiedostoja (CSS, tai JavaScript) uudelleen palvelimelta, vaan tallentaa ne välimuistiinsa. Välimuisti nopeuttaa huomattavasti sivuston käyttöä, koska joka kerta ei tarvitse ladata uudellaan staattisia tiedostoja palvelimelta. Se myös vähentää liikennekaistan kulutusta. Ajax-tekniikkaa käytettäessä selaimen välimuistiin jäävä sivun osa voi aiheuttaa sen, että sivun osa ei näytä latautuvan ensimmäisen latauksen jälkeen ollenkaan. Tämän voi estää käyttämällä PHP:n header-funktiota, jossa pakotetaan selain lataamaan sisältö aina uudelleen, eikä sitä aseteta sen välimuistiin: header("Cache-Control: no-cache, must-revalidate");

Merkistökoodaukset olivat toinen ongelma sovellusta kehitettäessä. Merkistökoodaus ongelmat johtuivat pitkälti käytetystä Prototype-nimisestä JavaScript-sovelluskehyksestä. Ongelmana on, että sovellus on toteutettu käyttäen ISO-8859-1 merkistökoodausta, kun taas Prototype käyttää UTF-8 merkistökoodausta. Ongelma ilmeni Ajax-hakujen yhteydessä siinä, että esimerkiksi ääkköset korruptoituivat haettaessa dataa XMLHttpRequestin avulla palvelimelta tai viedessä tietoja palvelimelle. Ongelma kierrettiin käyttämällä PHP:n iconv-funktiota, jolla käytetty merkistö voidaan koodata uudelleen. Lisäksi käytettiin PHP:n htmlentities-funktiota, joka koodaa erikoismerkit HTML-entiteeteiksi.

6 SOVELLUKSEN MITTAAMINEN JA TULOKSET

Tässä osassa käydään läpi opinnäytetyön mittareihin liittyvät tulokset, analysoidaan syntyneitä tuloksia ja kerrotaan itse mittaustoimenpiteistä.

6.1 Sovelluksen nopeuden mittaaminen

Sovelluksen nopeuden mittaamisen tarkoituksena on verrata ja tutkia nopeuttaako Ajax-tekniikan käyttö sovellusta verrattuna perinteisellä tavalla toteutettuun web-sovellukseen. Ajax-tekniikan yksi päätavoite käyttöliittymän kehittämisen lisäksi on juuri latausaikojen parantaminen. Mittaustulos on erittäin mielenkiintoinen, koska jos järjestelmä on suunniteltu järkevästi, sen pitäisi vähentää sovelluksen aiheuttamaa kuormaa ja samalla tietysti nopeuttaa sitä. Tämä kerrotaan olevan juuri Ajax-tekniikan vahvuus.

Sovellusta kehitettäessä ajateltiin, että järjestelmä on hyvä tehdä sellaiseksi, että sitä pystyy käyttämään sekä perinteisellä tavalla että Ajax-tekniikkaa hyödyntäen. Mittaus oli siis kohtalaisen helppo toteuttaa. Mittaukset tehtiin käyttämällä peräjälkeen perinteisellä tavalla toteutettua järjestelmää ja Ajax-pohjaisesti toteutettua järjestelmää, joilla molemmilla tehtiin täysin samat toimenpiteet.

Sovelluksen nopeutta mitattiin käyttämällä Proxy Sniffer -nimistä sovellusta, joka on erityisesti suunniteltu web-sovelluksen/sivuston nopeuden ja kuorman kestokyvyn testaukseen. Ohjelmistosta löytyy ilmainen perusversio ja monipuolisempi maksullinen versio. Ohjelmiston toimintaidea on luoda koneelle johon se asennetaan, välityspalvelin (Proxy), jota käytetään web-liikenteen seuraamiseen ja mittaamiseen. Tällöin käytössä olevaan selaimeen asetetaan kyseinen sovellus välityspalvelimeksi, jonka kautta kaikki web-liikenne kyseisessä selaimessa kulkee. Ohjelmisto on Java-pohjainen ja sitä käytetään selaimen avulla. Ohjelmiston avulla voidaan mitata kaikkea web-liikennettä, joka selaimen kautta kulkee, myös XMLHttpRequest:n (AJAX) avulla toteutettuja. Tämä onkin oleellista mittausta tehdessä. Kukin mittaussessio nauhoitetaan ja ohjelmisto näyttää tulokset erilaisina kaavioina. Ohjelmisto löytyy osoitteesta: http://www.proxy-sniffer.com/.

6.1.1 Mittauksen toteuttaminen

Mittauksessa käytettiin hyväksi rekisteriä, joka kattaa yli 1000 kiinteistöä ja huomattavasti enemmän kiinteistöihin liittyviä toimitiloja.

Mittaus tehtiin testaamalla aluksi, että järjestelmä toimii molemmilla tavoilla. Tämän lisäksi mittauksen eri toiminnot käytiin läpi. Ennen kumpaakin mittausta tyhjennettiin selaimen välimuisti, jotta se ei sotkisi mittaustuloksia. Mittauksesta tehtiin suunnitelma, jonka pohjalta se toteutettiin. Mittaus suunniteltiin siten, että tarjousjärjestelmän kaikki tärkeimmät Ajax-toteutukset saadaan testatuksi ja niitä verrataan vastaaviin toimintoihin perinteisellä tavalla toteutettuun web-järjestelmään. Mittauksessa tehtiin seuraavat toiminnot, joista perinteisel-

lä tavalla toteutetussa järjestelmässä jokainen toiminto tekee uuden sivulatauksen, kun taas Ajax-tekniikan avulla toteutettu tekee XMLHttpRequest-kutsun palvelimelle.

1. Järjestelmään kirjaudutaan ja sen pääsivu avautuu
2. Luodaan uusi tarjouskori
3. Aktivoidaan juuri luotu tarjouskori
4. Suoritetaan haku syöttämällä tilahaun postinumeroksi 00100 (hakutuloksena 80-toimitilaa, jotka ovat sivutettu kahdelle eri sivulle, 50-per sivu)
5. Avataan toimitilan tuotekortti (tilan id=13606)
6. Palataan takaisin hakuun
7. Siirrytään hakutuloksien sivulle 2
8. Lisätään koriin toimitila (tilan id=16787)
9. Poistetaan koriin lisätty toimitila
10. Poistetaan lisätty kori

Tulokset analysoidaan käymällä yläpuolella oleva listaus läpi kohta kohdalta ja tekemällä lopuksi yhteenveto.

Mittauksen tulokset jakautuvat kahteen eri kaavioon: Perinteinen Web-järjestelmä ja Ajax-pohjainen Web-järjestelmä. Kaaviot löytyvät liitteestä 1.

Kaavioista käy ilmi seuraavat asiat:

- Toimintojen tekojärjestys
- Latausajat millisekunteina
- Latausmäärät tavuina
- Suoritettujen HTTP-kutsujen määrä ja tieto siitä, mihin HTTP-kutsu kohdistuu
- Metodi, jolla pyyntö on tehty (GET vai POST)

6.1.2 Tulosten analysointi

Tulokset analysoidaan kohta kohdalta aikaisemmin mainitun listauksen mukaisesti. Jokainen toiminto käydään läpi ja toiminnoissa verrataan mitä tapahtuu perinteisellä tavalla toteutetussa web-järjestelmässä ja Ajax-pohjaisessa järjestelmässä.

Perinteisellä web-järjestelmällä tarkoitetaan perinteisellä tavalla toteutettua web-järjestelmää, eli sivu ladataan tällöin uudelleen joka toiminnon yhteydessä.

1. Järjestelmään kirjautuminen

Perinteinen web-järjestelmä lähettää kirjautumislomakkeen tiedot palvelimelle ja lataa selaimen muistiin käytössä olevat JavaScript- ja CSS-dokumentit. Järjestelmä lataa kaikki tarpeelliset elementit kerralla.

Ajax-pohjainen järjestelmä lähettää myös kirjautumislomakkeen tiedot palvelimelle ja lataa selaimen muistiin käytössä olevat JavaScript- ja CSS-dokumentit. Näiden lisäksi järjestelmä suorittaa kolme XMLHttpRequest-kutsua palvelimelle. Näitä ovat aktiivisen tarjouskorin lautaus (load=cart), tarjouskorien hallinnan lautaus (load=show_collections) ja tuotlistauksen lataus (load=product_list). Tuotelistaus on kuitenkin tyhjä (järjestelmää voitaisiin optimoida). Kutsujen vastaukset asetetaan käyttöliittymään JavaScriptin avulla niille määritetyille paikoille.

	Hakujen määrä (kpl)	Siirretty data (tavua)	Kulunut aika (ms)
Perinteinen web-järjestelmä	7	58343	1109
Ajax-pohjainen järjestelmä	10	58619	1642

2. Luodaan uusi kori

Perinteinen web-järjestelmä lataa koko sivun uudelleen ladaten ruutuun lomakkeen. Tarjouskorille annetaan nimi ja se lähetetään käyttäen POST-metodia palvelimelle ja se tallennetaan. Tehdään siis kaksi kutsua palvelimelle sekä lomakkeen lataus ja sen tallennus, jotka molemmat aiheuttavat sivulatauksen.

Ajax-pohjainen järjestelmä suorittaa kaksi XMLHttpRequest-kutsua palvelimelle. Ensimmäinen lataa lomakkeen (load=new_collection_form) ja toinen tallentaa sen sisällön (load=create_new_collection&name=Testikori2). Molemmat tehdään käyttäen GET-metodia.

	Siirretty data (tavua)	Kulunut aika (ms)
Perinteinen web-järjestelmä	4030	687
Ajax-pohjainen järjestelmä	925	360

3. Aktivoidaan luotu kori

Järjestelmät lähettävät GET-parametrinä palvelimelle aktivoitavan korin id:n ja se tallennetaan tietokantaan. Perinteisellä tavalla toteutettu web-järjestelmä tekee jälleen uuden sivulatauksen, kun Ajax-pohjainen järjestelmä muodostaa HTTP-kutsun palvelimelle ja aktivoi valitun korin (collection_id=68).

	Siirretty data (tavua)	Kulunut aika (ms)
Perinteinen web-järjestelmä	3037	547
Ajax-pohjainen järjestelmä	256	156

4. Haun suorittaminen

Haku suoritetaan molemmissa tapauksissa käyttäen POST-metodia. POST-metodin avullla lähetetään hakulomakkeen sisältö palvelimelle. Perinteisellä tavalla toteutettu web-järjestelmä lähettää lomakkeen sisällön palvelimelle ja uusi sivu hakutuloksineen avautuu. Ajax-pohjainen järjestelmä kerää lomakkeen tiedot JavaScriptin avulla ja lähettää ne XMLHttpReuquest:n avulla palvelimelle. Palvelin palauttaa selaimeen ainoastaan hakutulokset, joka päivitetään käyttöliittymään.

	Siirretty data (tavua)	Kulunut aika (ms)
Perinteinen web-järjestelmä	6183	1156
Ajax-pohjainen järjestelmä	3289	422

5. Avataan tuotekortti

Molemmissa tapauksissa pyyntö suoritetaan GET-metodin avulla. Palvelimelle lähetetään valitun tuotteen id-arvo ja palvelin palauttaa tuotekortin sisältöineen. Ajax-sovellukselle palautetaan ainoastaan tuotekortin sisältö, kun taas toisessa tapauksessa sivu ladataan uudelleen kokonaisuudessaan.

	Siirretty data (tavua)	Kulunut aika (ms)
Perinteinen web-järjestelmä	2579	515
Ajax-pohjainen järjestelmä	1418	532

6. Palataan hakuun

Tuotekortilta hakuun palataan perinteisessä sovelluksessa painamalla back-nappia, joten toiminto ei aiheuta verkkoliikennettä. Myöskään Ajax-pohjaisessa järjestelmässä tuotekortin sulkeminen ei aiheuta verkkoliikennettä johtuen tavasta jolla se on muodostettu.

7. Siirrytään hakutuloksissa seuraavalle sivulle

Palvelimelle lähetetään GET-parametrina valitun sivun numero jolloin palvelin palauttaa kyseisen hakujoukon. Ajax-sovellukselle palautetaan ainoastaan listaus kohteista, kun taas toisessa tapauksessa ladataan koko sivu uudelleen.

	Siirretty data (tavua)	Kulunut aika (ms)
Perinteinen web-järjestelmä	5232	750
Ajax-pohjainen järjestelmä	2394	375

8. Lisätään kiinteistö koriin

Molemmat järjestelmät lähettävät palvelimelle GET-parametrina tiedon koriin asetettavasta kiinteistöstä ja palvelin tallentaa sen tietokantaan. Ajax-pohjainen järjestelmä päivittää ainoastaan aktiivisen korin, kun taas toinen järjestelmä tekee uuden sivulatauksen.

	Siirretty data (tavua)	Kulunut aika (ms)
Perinteinen web-järjestelmä	3166	593
Ajax-pohjainen järjestelmä	437	188

9. Poistetaan kiinteistö korista

Molemmat järjestelmät lähettävät palvelimelle GET-parametrina tiedon poistettavasta kohteesta sen id:n perusteella ja palvelin poistaa sen tietokannan taulusta, jossa ylläpidetään korin tietoja. Ajax-pohjainen järjestelmä päivittää jälleen korin, kun taas toinen järjestelmä tekee uuden sivulatauksen.

	Siirretty data (tavua)	Kulunut aika (ms)
Perinteinen web-järjestelmä	3063	578
Ajax-pohjainen järjestelmä	256	156

10. Poistetaan kori

Molemmat järjestelmät lähettävät palvelimelle GET-parametrina tiedon poistettavasta korista sen id:n perusteella ja se poistetaan tietokannasta. Lisäksi palvelimella vaihdetaan aktiivinen kori toiseksi, koska poistettava kori oli aktiivinen. Ajax-pohjainen järjestelmä päivittää korin, ja listauksen koreista, kun taas toinen järjestelmä tekee uuden sivulatauksen.

	Siirretty data (tavua)	Kulunut aika (ms)
Perinteinen web-järjestelmä	2928	546
Ajax-pohjainen järjestelmä	462	171

6.1.3 Johtopäätökset mittauksesta

Mittauksen perusteella voidaan sanoa, että Ajax-tekniikan käyttö nopeuttaa sovelluksen käyttöä ja vähentää palvelimen ja selaimen välillä kulkevaa liikennettä. Pidemmällä aikavälillä Ajax-pohjainen järjestelmä säästää kaistaa ja kuormittaa siten palvelinta vähemmän. Se tekee myös sovelluksen käyttökokemuksesta mukavamman, koska käyttäjän ei eri toimintojen välillä tarvitse odotella ruudun päivittymistä.

	Ajax-pohjainen järjestelmä	Perinteinen järjestelmä
Lataukset yhteensä	68 056 tavua	88 561 tavua
HTTP-kutsujen määrä	19	16
Toimintojen suoritusaika yhteensä	4002 ms	6481 ms

Kuvio 7: Testitapaukset 1-10 yhteenvetona

Testin yhteenvetona Ajax-pohjainen järjestelmä latasi palvelimelta 68 056-tavua ja perinteisellä tavalla toteutettu 88 561-tavua. Palvelimelta haetun tiedon määrä on lähes ¼ vähemmän Ajax-pohjaisessa, kuin perinteisellä tavalla toteutetussa järjestelmässä. Hakuja Ajax-pohjaisessa järjestelmässä oli enemmän (19 HTTP-kutsua), kuin perinteisellä tavalla toteutetussa web-sovelluksessa (16-kutsua), mutta kutsujen tiedonsiirto oli vähäisempää. Tämä on Ajax-pohjaisessa järjestelmässä luonnollista. Testitapauksessa (Testin kohta 1) järjestelmään kirjauduttaessa ladataan tarjouskorit ja niiden hallinta usealla eri HTTP-kutsulla palvelimelle, joka lisää tehtyjen palvelinkutsujen määrää.

Mittauksessa perinteisellä tavalla toteutetussa web-järjestelmässä on muutama ylimääräinen lataus, joita ei olisi tarvinnut tehdä. Nämä siis hiukan vääristävät tulosta, mutta ei merkittävästi. Näitä olivat jotkin sovelluksen käynnistyksen yhteydessä tehtävät JavaScript-tiedosto lataukset, joita käytetään vain Ajax-tekniikalla toteutetussa sovelluksessa. Nämä tiedostot ovat: modalbox.js ja basic.js. Tiedostot hoitavat pelkästään Ajax-toteutukseen liittyvä toimenpiteitä. Nämä tapaukset koskevat testin kohtaa yksi, eli tiedostot latautuvat, kun sovellus käynnistetään.

7 YHTEENVETO JA JOHTOPÄÄTÖKSET

Ajax-tekniikka sopii mielestäni kaikkiin verkkosovelluksiin, joiden käyttöliittymää halutaan ilmaisuvoimaisemmaksi, sitä halutaan rikastaa tai sovellusta halutaan muutenkin kehittää tehokkaammaksi. Ajax-tekniikkaa voidaan käyttää lähes kaikkien eri palvelinpuolentekniikoiden kanssa, joten se soveltuu useisiin eri tilanteisiin riippumatta siitä, millä tekniikalla web-sovellus on kehitetty. Ajax-tekniikkaa voidaan rakentaa myös osaksi jo olemassa olevaa web-sovellusta. Tämän todistaa se, että sitä käytettiin sovelluksessa, joka kehitettiin Isolta Oy:n kehittämän ohjelmistokehyksen päälle, vaikka tällaista ei aikaisemmin ollut kehitetty.

Kun järjestelmää lähdettiin kehittämään, ei minulla vielä ollut juurikaan kokemusta Ajax-tekniikan käytöstä. Projektin edetessä oppi monia uusia asioita. Asian opettelu lukemalla auttoi jonkin verran, mutta tietysti parhaiten sen sisäisti vasta itse tekemällä ja kehittämällä. Aiheesta on vielä paljon opiskeltavaa ja opittavaa, mutta oma kiinnostus Ajax-kehitykseen on koko ajan kasvanut, koska ymmärtää sen tuomat mahdollisuudet web-sovelluksien kehittämiseen. Erilaisissa web-pohjaisissa järjestelmissä Ajax-tekniikan käyttömahdollisuudet ovat rajattomat. Ajax-tekniikkaa tulen varmasti hyödyntämään tulevaisuuden kehitysprojekteissa aina kun siihen on tilaisuus. Toisaalta ymmärrän senkin, ettei sitä kannata käyttää kaikissa paikoissa.

Ajax-osaajalta vaaditaan monenlaisia taitoja. Hänen täytyy ymmärtää useita eri tekniikoita ja on osattava hyödyntää niitä tehokkaasti käytäntöön. Ei pelkästään riitä, että osaa ohjelmoida, vaan pitää osata nähdä rakennettava järjestelmä suurempana kokonaisuutena. Muuten voi olla vaikeata hahmottaa, millä tavalla Ajax-tekniikka sopii kehitettävään järjestelmään. Ajax-tekniikan mahdollisuudet avaavat kehittäjälle uusia ovia rikkaampien Web-sovellusten kehittämiseen ja sen avulla Web-järjestelmät voivat todella kehittyä. On kuitenkin tiedettävä, mihin sitä kannatta käyttää ja mihin ei.

Ajax-asiantuntijan pitää siis hallita monia eri tekniikoita. JavaScriptin rooli Ajax-sovellusta kehitettäessä on suuri. JavaScriptin avulla suoritetaan palvelinkutsut ja käsitellään dataa. Palvelinpuolella pitää vähintään ymmärtää se, miten sovelluslogiikka pitää sinne rakentaa. Palvelimelta saatu data käsitellään JavaScriptin avulla ja asetetaan selaimeen DOM-mallia hyödyntäen. Vastaus palvelimelta voi tulla XML-muotoisena, joten sekin pitää hahmottaa. XML-dataa voidaan käsitellä XSLT:n tai JavaScriptin avulla muunnettaessa sitä HTML-muotoon. Selaimeen asetettava data pitää myös CSS:n avulla muotoilla, jotta se sopisi sovelluksen käyttöliittymään. Kuten huomataan, tekniikoita on paljon ja niitä pitää osata yhdistää. Ajax-osaajalla pitää olla käsitys näistä kaikista ja niitä pitää osata hyödyntää.

7.1 Tavoitteiden saavuttaminen

Opinnäytetyön tärkeimpänä tavoitteena oli Ajax-tekniikan hyödyntäminen tarjousjärjestelmässä siten, että se tuo helpotusta sovelluksen käyttöön ja nopeuttaa sen toimintaan.

Tavoitteet toteutuivat työssäni hyvin. Nopeusmittauksen perusteella voidaan sanoa, että Ajax-tekniikan käyttö tosiaan nopeuttaa järjestelmän käyttöä, pienentää huomattavasti latausaikoja ja vähentää verkkoliikennettä. Lisäksi Ajax-tekniikka tuo lukuisia uusia mahdollisuuksia kehittää uusia dynaamisia toiminnallisuuksia järjestelmään.

7.2 Jatkokehitys

Kuten alussa jo mainittiinkin, on tarjousjärjestelmän Ajax-toteutus vain pieni osa kokonaisuudesta. Tarjoustyökalua tullaan kehittämään syksyn 2008 ja talven 2009 aikana. Syksyn 2008 aikana on järjestelmälle tulossa myös uusia käyttäjiä, jotka tuovat omat vaatimuksensa järjestelmän käyttöön.

Jatkokehityksessä panostetaan entistä enemmän Ajax-pohjaiseen arkkitehtuuriin ja varsinkin palvelinpuolen järjestelmää tullaan kehittämään tätä silmälläpitäen. Perinteisellä tavalla toteutettua web-järjestelmää ei tietenkään unohdeta, vaan muutoksia tehdään siten, että järjestelmä tukee molempia tapoja. Järjestelmää on siis tarkoitus pystyä käyttämään, vaikka JavaScriptin käyttö olisi selaimessa estetty. Merkittävässä roolissa on XML-kielen käyttö, jonka avulla tieto Ajax-pohjaisessa sovelluksessa siirretään palvelimen ja selaimen välillä ja jota hyödynnetään myös perinteisellä tavalla toteutetussa web-järjestelmässä tekemällä sama toiminnallisuus mahdolliseksi palvelinpuolelle ja selaimeen.

Myös selainpuolen tekniikkaa on tarkoitus jatkokehittää siten, että siitä tulisi monikäyttöisempi. Tällä tarkoitetaan sitä, että samaa JavaScript-koodia voitaisiin käyttää suoraan erityyppisissä toteutuksissa, ilman räätälöintejä, joita se vielä toistaiseksi hiukan vaatii. Myös XSLT-muunnoksien käyttöä tullaan lisäämään järjestelmää kehitettäessä, mutta luultavasti enemmän jo palvelinpäähän Safarin ongelmien takia. Näillä muutoksiaan saadaan käyttöliittymä erotettua kokonaan omaksi osakseen eikä se vaikuta muuhun järjestelmän toimintalogiikkaan.

8 LÄHTEET

Ajax, promise or hype? 2005. Viitattu 11.2.2008.
http://www.quirksmode.org/blog/archives/2005/03/ajax_promise_or.html#more

Asleson, R. & Nathaniel T. S. 2006. Ajax - Tehokas hallinta. Kääntänyt: Kamppila, M. Jyväskylä: Readme.fi.

Adobe. 2008. Flash-sisällön ja -sovellusten esittäminen työasemissa ja laitteissa. Viitattu 25.10.2008. http://www.adobe.com/fi/products/flashplayer/productinfo/features/

Ekonoja, A., Lahtonen, T. & Mäntyala, J. Ajax– Luento 2008. Viitattu 20.9.2008.
http://appro.mit.jyu.fi/sovellukset/luennot/luento15/

Floyd, M. 2002. Special Edition Using XSLT. Indianapolis: Que.

Garrett, J. 2005. Ajax: A New Approach to Web Applications. Viitattu 2.1.2008.
http://www.adaptivepath.com/publications/essays/archives/000385.php

Holzner, S. 2006. Ajax for Dummies. Indianapolis: Wiley Publishing.

Holzner, S. 2007. Ajax Bible. Indianapolis: Wiley Publishing

IBM. New to XML. Viitattu 19.9.2008.
http://www.ibm.com/developerworks/xml/newto/index.html

Järvinen, P. & Järvinen, A. 2004. Tutkimustyön metodeista. Tampere: Tampereen Yliopistopaino.

Korpela, J. 2007. Web-julkaisemisen opas. Viitattu 28.9.2008.
http://www.cs.tut.fi/~jkorpela/webjulk/3.4.html

Lauria, S. M. 2008. Advanced Ajax : architecture and best practices. Pearson Education Inc.

Nielsen, J. 2005. Why Ajax Sucks (Most of the Time). Viitattu 2.1.2008.
http://www.usabilityviews.com/ajaxsucks.html

Peltomäki, J. 2001. JavaScript. Jyväskylä: Docendo.

Ullman, C. & Lucinda, D. 2007. Beginning Ajax. Indianapolis: Wiley Publishing

W3C - Quin, L. What is XSL? Viitattu 1.9.2008. http://www.w3.org/Style/XSL/WhatIsXSL.html

W3C. 2005. Document Object Model (DOM). Viitattu 1.9.2008. http://www.w3.org/DOM/

Walsh, N. 1998. A Technical Introduction to XML. Viitattu 21.9.2008. http://www.xml.com/pub/a/98/10/guide0.html?page=2#AEN58

Zakas, N. C., McPeak, J. & Fawcett, J. 2007. Professional Ajax 2nd Edition. Indianapolis: Wiley Publishing.

Kuvaluettelo

Kuvio- ja taulukkoluettelo

Nopeusmittaus suunintelma

1. Kirjaudutaan järjestelmään
2. Luodaan uusi kori
3. Aktivoidaan uusi kori
4. Suoritetaan haku syöttämällä tilahaun postinumeroksi 00100 (hakutuloksena 80-kiinteistöä, 50-kiinteistöä/sivu)
5. Avataan kiinteistön 13606 tuotekortti
6. Palataan hakuun
7. Siirrytään sivulle 2
8. Lisätään koriin kiinteistö 16787
9. Poistetaan kiinteistö 16787 korista
10. Poistetaan kori

Mittauksen toteuttaminen

Nopeusmittaus toteutetaan ProxySniffer nimisellä ohjelmalla. Mittaus tehdään kahtena eri sessiona siten, että molemmat sessiot nauhoitetaan. Kun nauhoitus lopetetaan muodostaa ohjelma kyseisestä sessiosta kaavion, jonka avulla sitä voidaan arvioida.

× 97		[-] Page #2: Ajax-tapa user's think time: 3 seconds ±35%			
Item	**Offset**	**Position**	**Content Size**	**Time**	**HTTP Request ← HTTP Response**
× 98	0,00 sec		2'585 bytes	563 ms	POST http://demo2.isolta.org/index.php?module_id=members ← 200 (OK) TEXT/HTML
× 99	0,59 sec		29'381 bytes	172 ms	GET http://demo2.isolta.org/js/prototype.js ← 200 (OK) APPLICATION/X-JAVASCRIPT
× 100	0,59 sec		2'068 bytes	125 ms	GET http://demo2.isolta.org/pages.css ← 200 (OK) TEXT/CSS
× 101	0,83 sec		8'988 bytes	94 ms	GET http://demo2.isolta.org/js/effects.js ← 200 (OK) APPLICATION/X-JAVASCRIPT
× 102	0,94 sec		5'090 bytes	78 ms	GET http://demo2.isolta.org/js/basic.js ← 200 (OK) APPLICATION/X-JAVASCRIPT
× 103	1,03 sec		3'202 bytes	32 ms	GET http://demo2.isolta.org/fckeditor/fckeditor.js ← 200 (OK) APPLICATION/X-JAVASCRIPT
× 104	1,08 sec		6'664 bytes	47 ms	GET http://demo2.isolta.org/js/modalbox.js ← 200 (OK) APPLICATION/X-JAVASCRIPT
× 112	1,28 sec		106 bytes	188 ms	GET http://demo2.isolta.org/index.php?module_id=3&get_page_html=1&load=cart ← 200 (OK) TEXT/HTML
× 113	1,31 sec		462 bytes	171 ms	GET http://demo2.isolta.org/index.php?module_id=3&get_page_html=1&load=show_collections ← 200 (OK) TEXT/HTML
× 114	1,33 sec		73 bytes	172 ms	GET http://demo2.isolta.org/index.php?module_id=3&get_page_html=1&load=productlist ← 200 (OK) TEXT/HTML
× 118	8,20 sec		420 bytes	188 ms	GET http://demo2.isolta.org/index.php?module_id=3&get_page_html=1&load=new_collection_form&cid=0 ← 200 (OK) TEXT/HTML
× 119	19,66 sec		505 bytes	172 ms	GET http://demo2.isolta.org/index.php?module_id=3&get_page_html=1&load=create_new_collection&name=Testikori2&type=0&cid=0 ← 200 (OK) TEXT/HTML
× 120	19,97 sec		256 bytes	156 ms	GET http://demo2.isolta.org/index.php?module_id=3&get_page_html=1&load=set_active_collection&collection_id=68 ← 200 (OK) TEXT/HTML
× 122	29,09 sec		3'289 bytes	422 ms	POST http://demo2.isolta.org/index.php?module_id=3&load=productlist&get_page_html=1&page_number=0&search_option=advanced_search&search_product_type_id=1%232 ← 200 (OK) TEXT/HTML
× 169	37,48 sec		1'418 bytes	532 ms	GET http://demo2.isolta.org/index.php?module_id=3&get_page_html=1&load=productShow&pid=13606 ← 200 (OK) TEXT/HTML
× 171	43,78 sec		2'394 bytes	375 ms	POST http://demo2.isolta.org/index.php?module_id=3&load=productlist&get_page_html=1&research=1&sort_field=&page_number=1 ← 200 (OK) TEXT/HTML
× 196	50,28 sec		437 bytes	188 ms	GET http://demo2.isolta.org/index.php?module_id=3&get_page_html=1&load=toCart&pid=16787 ← 200 (OK) TEXT/HTML
× 197	61,05 sec		256 bytes	156 ms	GET http://demo2.isolta.org/index.php?module_id=3&get_page_html=1&load=removeFromCart&pid=16787 ← 200 (OK) TEXT/HTML
× 198	67,69 sec		462 bytes	171 ms	GET http://demo2.isolta.org/index.php?module_id=3&get_page_html=1&load=remove_collection&remove_cid=68 ← 200 (OK) TEXT/HTML
Total:		**67,86 sec**	**68'056 bytes**		**19 Requests , 1,00 kbytes/sec**

× 0		[-] Page #1: perinteinen tapa user's think time: 3 seconds ±35%			
Item	**Offset**	**Position**	**Content Size**	**Time**	**HTTP Request ← HTTP Response**
× 1	0,00 sec		2'949 bytes	562 ms	POST http://demo2.isolta.org/index.php?module_id=members ← 200 (OK) TEXT/HTML
× 2	0,61 sec		2'068 bytes	94 ms	GET http://demo2.isolta.org/pages.css ← 200 (OK) TEXT/CSS
× 3	0,61 sec		29'381 bytes	188 ms	GET http://demo2.isolta.org/js/prototype.js ← 200 (OK) APPLICATION/X-JAVASCRIPT
× 4	0,89 sec		8'988 bytes	109 ms	GET http://demo2.isolta.org/js/effects.js ← 200 (OK) APPLICATION/X-JAVASCRIPT
× 5	1,05 sec		5'091 bytes	47 ms	GET http://demo2.isolta.org/js/basic.js ← 200 (OK) APPLICATION/X-JAVASCRIPT
× 6	1,11 sec		3'202 bytes	47 ms	GET http://demo2.isolta.org/fckeditor/fckeditor.js ← 200 (OK) APPLICATION/X-JAVASCRIPT
× 7	1,17 sec		6'664 bytes	62 ms	GET http://demo2.isolta.org/js/modalbox.js ← 200 (OK) APPLICATION/X-JAVASCRIPT
× 16	4,64 sec		1'064 bytes	156 ms	GET http://demo2.isolta.org/index.php?module_id=3&urlLoad=create_cart ← 200 (OK) TEXT/HTML
× 17	10,42 sec		2'966 bytes	531 ms	POST http://demo2.isolta.org/index.php?module_id=3&urlLoad=make_new_collection ← 200 (OK) TEXT/HTML
× 18	14,61 sec		3'037 bytes	547 ms	GET http://demo2.isolta.org/index.php?module_id=3&urlLoad=set_active_cart&cid=66 ← 200 (OK) TEXT/HTML
× 20	24,14 sec		6'183 bytes	1'156 ms	POST http://demo2.isolta.org/index.php?module_id=3&urlLoad=productlist&page_number=0&search_option=advanced_search&search_product_type_id=1 ← 200 (OK) TEXT/HTML
× 67	33,92 sec		2'579 bytes	515 ms	GET http://demo2.isolta.org/index.php?module_id=3&urlLoad=product_card&pid=13606 ← 200 (OK) TEXT/HTML
× 69	41,81 sec		5'232 bytes	750 ms	GET http://demo2.isolta.org/index.php?module_id=3&urlLoad=productlist&research=1&sort_field=&page_number=1 ← 200 (OK) TEXT/HTML
× 94	61,52 sec		3'166 bytes	593 ms	GET http://demo2.isolta.org/index.php?module_id=3&urlLoad=product_to_cart&pid=16787 ← 200 (OK) TEXT/HTML
× 95	77,08 sec		3'063 bytes	578 ms	GET http://demo2.isolta.org/index.php?module_id=3&urlLoad=remove_product&pid=16787 ← 200 (OK) TEXT/HTML
× 96	80,77 sec		2'928 bytes	546 ms	GET http://demo2.isolta.org/index.php?module_id=3&urlLoad=remove_cart&cid=66 ← 200 (OK) TEXT/HTML
Total:		**81,31 sec**	**88'561 bytes**		**16 Requests , 1,09 kbytes/sec**