TAPAHTUMANTEKIJÄN TYÖKALUPAKKI

Case: Ice Event

LAHDEN AMMATTIKORKEAKOULU
Tekniikan ala
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka
Opinnäytetyö
Kevät 2011
Tuomas Ruuskanen

Lahden ammattikorkeakoulu
Tietotekniikan koulutusohjelma

RUUSKANEN, TUOMAS:

Tapahtumantekijän työkalupakki
Case: Ice Event

Ohjelmistotekniikan opinnäytetyö, 41 sivua

Kevät 2011

TIIVISTELMÄ

Opinnäytetyön tarkoituksena oli valmistaa sivusto, jossa esitettäisiin kootusti matkailun laitoksen Ice Event -projektissa kerättyä aineistoa. Sivustosta tuli tehdä helppokäyttöinen sekä näyttävä.

Työssä käytettiin Ajax-tekniikkaa, jolla sivuston käytöstä tulisi vaivatonta ylimääräisten sivulatausten hävitessä. Haasteena oli myös dynaamisen valikon rakentaminen, jonka sisältö haettiin suoraan tietokannasta. Valikon nappien painaminen haki sivulle sisältöä tietokannasta, ilman sivulatausta.

Tarkoituksena oli myös erottaa ohjelmoijan sekä graafikon työt erikseen, joten yhtäaikainen työskentely olisi mahdollista, ilman päällekäin tehtyjä muutoksia. Tätä varten käytettiin erillistä CSS-tiedostoa, ja sivuston rakenne sovittiin jo projektin alussa. Näin graafikko pystyi muokkaamaan tyylejä ohjelmoijan työstäessä sisältöä.

Työn aikana havaittiin Ajaxin sopivan hyvin tällaisen sivuston toteutukseen. Se ei vaatinut uusien ohjelmointikielien opiskelua, perinteisten kielien tuntemuksen lisäksi. Sivustosta tuli halutunlainen ja asiakkaat olivat erityisen tyytyväisiä ulkoasuun.

Avainsanat: Ajax, HTML, JavaScript, PHP, DOM, CSS, GET, POST

Lahti University of Applied Sciences
Degree Programme in Information Technology

RUUSKANEN, TUOMAS: Toolbox for Event Organizers
Case: Ice Event

Bachelor's Thesis in Software Engineering, 41 pages

Spring 2011

ABSTRACT

The goal of this Thesis was to build a site where it is possible to present information that was gathered in the Ice Event project by Lahti University of Applied Sciences Faculty of Tourism and Hospitality. The site was meant to be easy to use and have a pleasant appearance.

The site was made by using the Ajax technique, which made the site effortless to use, because there would not be any page refreshes after entering the website. One challenge was to create a dynamic menu, that gathers the content straight from the database. Pressing the menu buttons made the site retrieve content to the site without refreshing the site itself.

Another purpose was to separate the work of the programmer from the work of the graphic artist, so simultaneous working would be possible, without changing the structure of the website at the same time. A separate CSS file was used, and the structure of the site was agreed in the beginning of the project. With these measures the graphic artist was able to work on the styles while the programmer was focusing on the content.

Ajax was discovered to be suitable for implementing a site like this. It did not require learning of new programming languages, if traditional languages were familiar already. The site met the requirements and the clients were especially satisfied with the appearance.

Key words: Ajax, HTML, JavaScript, PHP, DOM, CSS, GET, POST

SISÄLLYS

1 JOHDANTO

Lahden ammattikorkeakoulun matkailun laitoksella oli ollut kaksivuotinen Ice Event -projekti, joka pyrkii helpottamaan erilaisten tapahtumien valmistelua valmiin pohjan mukaisesti. Lisäksi sitä voisi uudelleenkäyttää erilaisilla tapahtuma-alustoilla. Mallin käyttäjäkunta ei rajoittunut pelkästään tuoreisiin tapahtumanjärjestäjiin, vaan sen haluttiin helpottavan myös kokeneempienkin järjestäjien taakkaa huomioimalla tapahtumien eri osa-alueet monipuolisesti.

Tämän opinnäytetyön osuus koko Ice Event –projektissa oli kaiken kerätyn informaation esittäminen internetissä sivustolla. Sivuston nimeksi tuli Tapahtumanjärjestäjän työkalupakki, joka kuvaa käyttötarkoitusta varsin hyvin. Sivuston ulkoasuun tuli myös kiinnittää huomiota, jotta sivustosta ei tulisi luotaantyöntävä, jota se pelkän tekstimateriaalin takia olisi voinut olla.

Tutkimusongelmana työssä nähtiin Ajax-tekniikan käyttämän asynkroonisen tiedonsiirron soveltuminen www-palveluihin, ja saada aikaan lähes viiveetön sivusto, jolla ei ole turhia sivulatauksia. Toisena ongelmana oli miten erotella graafikon ja ohjelmoijan työt, jotta kumpaakin osa-aluetta voitaisiin työstää yhtäaikaa toisiaan häiritsemättä.

Sivuston rakentamisen aikana yhteyshenkilöinä matkailun laitokselta olivat Ice Event -projektipäälliköt. Työhön osallistuivat tekniikan laitokselta ohjelmistotekniikan yliopettaja ohjaamaan teknistä toteutusta. Mediatekniikan puolelta oli muutama opiskelija opettajansa ohjaamana vastaamassa sivuston ulkoasusta sekä käyttöliittymästä.

Tässä työssä keskityttiin käyttämään pelkästään avoimen lähdekoodin ohjelmistoja ja tekniikoita, jotta mahdollinen jatkokehitys säilyisi ilmaisena. Kappaleessa kaksi keskitytään tehtävänantoon sekä sisältöön. Kappaleessa kolme paneudutaan perinteisen sivuston ja Ajax-sivuston eroihin. Neljännessä kappaleessa kerrotaan selainohjelmoinnista, viidennen kappaleen keskittyessä palvelinohjelmointiin. Kuudennessa kappaleessa käydään läpi Ice Event –sivuston toteutusta ja seitsemäs on yhteenveto.

2 TEHTÄVÄNANTO

2.1 Asiakasvaatimukset

Sivuston suunnittelussa lähtökohtana oli ylläpidettävyys, eli sivuston sisältöä piti pystyä muokkaamaan helposti ilman ohjelmointitaitoja. Tästä syystä sivustolle tuli rakentaa hallintasivusto, jonka avulla asiakas pystyy tekemään muutoksia sisältöön.

Sivuston ulkoasu nousi varsin merkittävään rooliin, koska sivustolla esitettävä aineisto oli pääosin tekstiä, joten ulkoasulla haluttiin saada sivustosta elävämpi sekä monipuolisempi. Kuvakkeiden ja nappien suunnitteluun haluttiin panostaa, jotta ne olisivat teemaan liittyviä ja kaikki taustakuvat oli otettu tapahtumapaikoilta, kun tapahtumia valmisteltiin.

Jo heti alussa asiakkaat esittivät oman näkemyksensä sivuston rungosta, jonka pohjalta visuaalinen sekä toiminnallinen kehitys aloitettiin. Asiakkaat olivat hyvin kiinnostuneita ulkoasun onnistumisesta, joten heitä konsultoitiin tiheästi, varsinkin projektin alkuvaiheessa.

2.2 Sisällön rakenne

Asiakkailla oli materiaalia sivuston sisällöksi, jonka he syöttivät sivustolle, joten hallintasivuston tuli olla toiminnassa jo varsin varhaisessa vaiheessa. Ennen tätä heidän tuli lajitella aineisto aiheittain ja laatia järjestys, minkä pohjalta sivuston rakennetta pääsi suunnittelemaan.

Asiakkaat olivat laatineet rungon, jonka mukaan aineisto laitettaisiin sivustolle. Tuota runkoa hyväksikäyttäen sivuston eri osat hahmoteltiin ja saatiin aikaiseksi järjestys, jota pystyttiin käyttämään hyväksi tietokantaa suunniteltaessa.

3 PERINTEINEN VERKKOSIVU VS. AJAX-SIVU

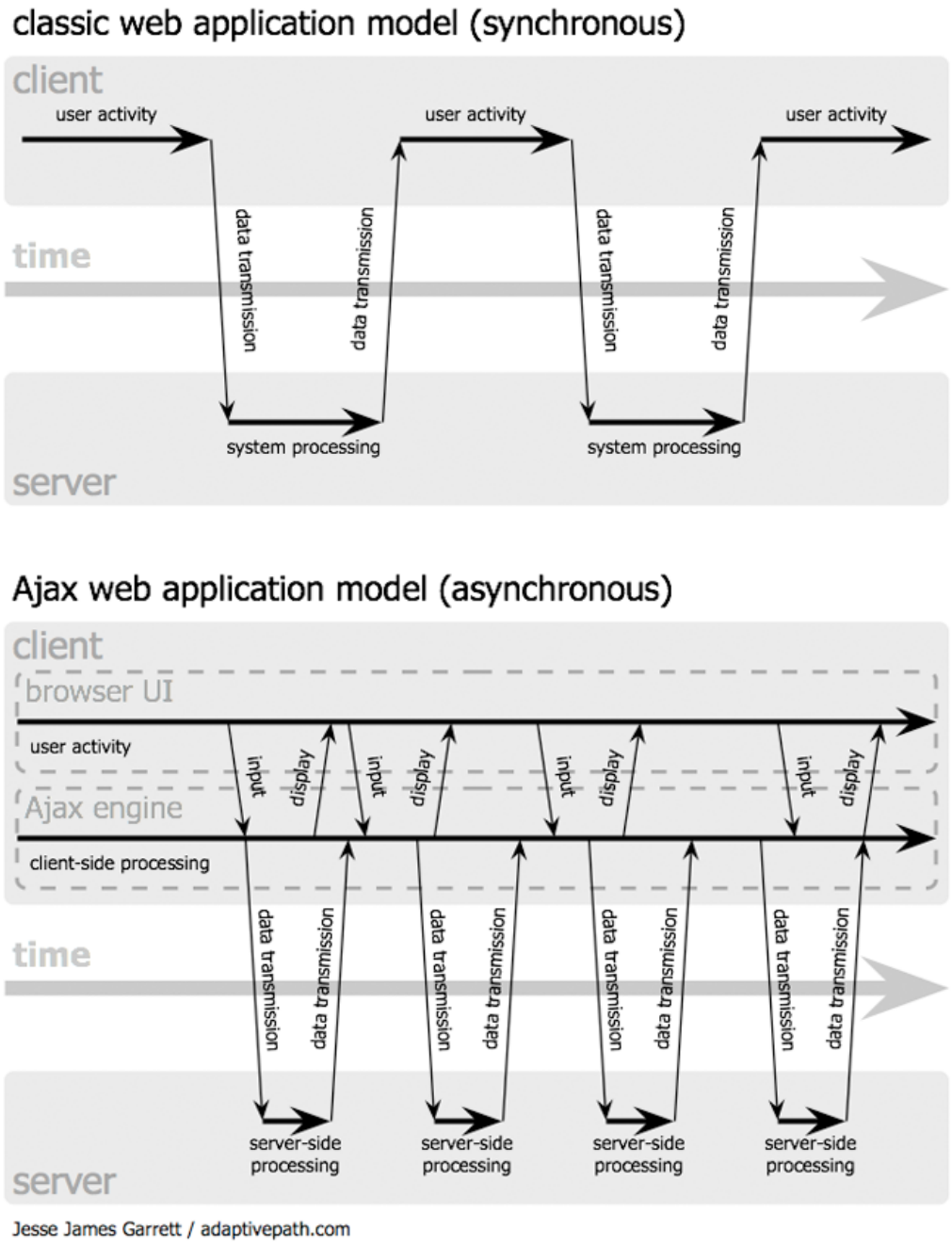
3.1 Perinteinen toteutus

Yksinkertaisimmillaan verkkosivu voi olla yksi HTML-dokumentti, mutta jos sivun ulkoasuun halutaan panostaa, on otettava tyylit käyttöön. Tyylienkin kanssa pelkkä HTML-sivu on hyvin yksinkertainen. Luultavasti sivu halutaan laajentaa sivustoksi linkittämällä useita sivuja yhteen. Monesti HTML:n ominaisuudet loppuvat kesken ja halutaan interaktiivinen sivusto, joten käyttöön otetaan JavaScript, jolla päästään ohjelmoimaan selaimen toimintaa ja luomaan interaktiivisuutta..

Jossain vaiheessa voi tulla tarvetta käsitellä tietoja palvelimella, jolloin PHP on oiva ratkaisu. PHP:llä saadaan sivuille dynaamisuutta, käsiteltyä tietokantoja sekä muita palvelimella tapahtuvia toimenpiteitä. Hyvä esimerkki palvelimen tarpeellisuudesta on sivusto, joka vaatii kirjautumisen. Tämä on aina toteutettava palvelimella.

Perinteisten sivustojen ongelma on jatkuvat sivulataukset, esimerkiksi sivuston sisällä jokainen linkki vie omalle sivulleen, mikä tarkoittaa uutta sivulatausta jokaisen siirtymisen välillä. Kuvion 1 yläosasta näkyy, että tietoja lähetetään palvelimelle ainostaan käyttäjän tehdessä jotain. Tästä ei tietenkään kaikilla sivuilla ole haittaa, mutta käyttäjälle pienten muutoksien teko on raskasta, jos sivu pitää ladata joka kerta uudelleen. Monesti sivulatausta tehdessä joudutaan lataamaan paljon turhaa tietoa, kuten samaa mitä edellisellä sivulla oli esillä. Esimerkiksi lomakkeen täyttäminen perinteisellä menetelmällä vaatii kaikkien kenttien täyttämisen, lähetä-napin painamisen, minkä jälkeen odotetaan, tarvitseeko tietoja muuttaa. Muutos tapahtuu yleensä samalla lomakkeella, ainoana muutoksena on jokin merkki tai värisymboli muutettavan kohteen merkkaamiseksi. Asynkronisessa sovelluksessa lomake voidaan tarkastaa kirjoittamisen yhteydessä, joten sivulta ei tarvitse poistua virheen löytämiseksi.

3.2 Ajax-toteutus

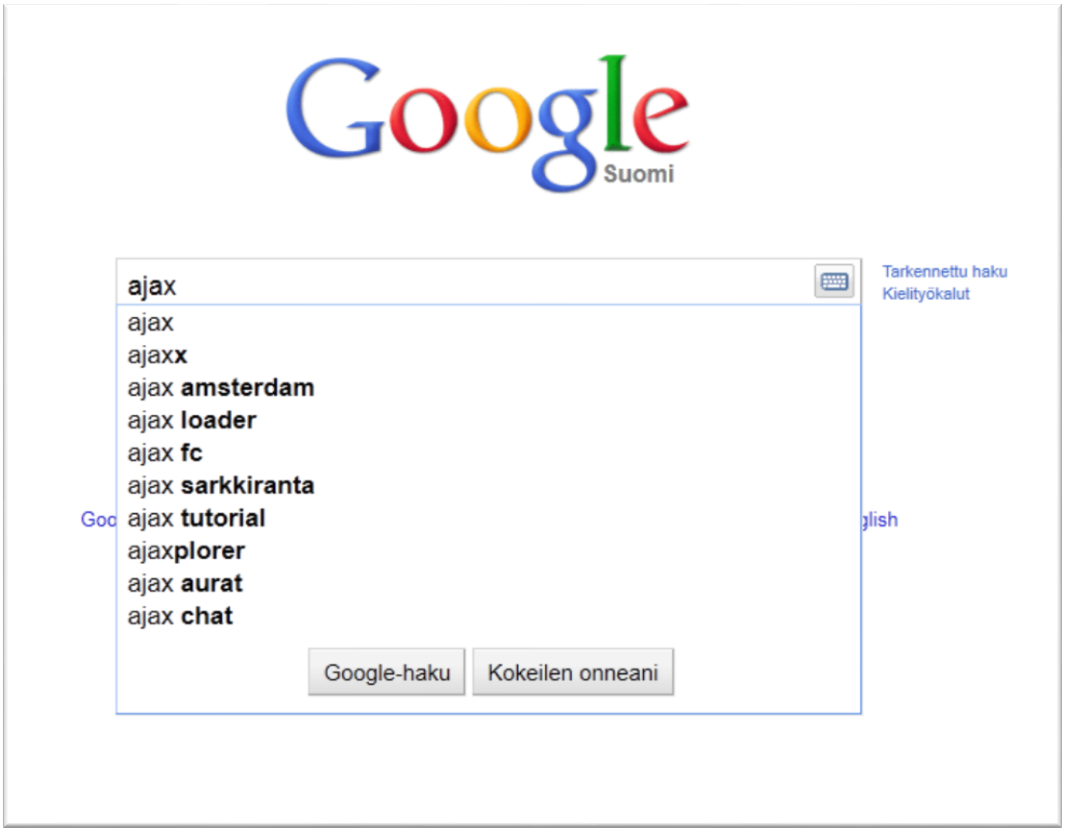


KUVIO 1. Perinteinen sivusto vs. Ajax-sivusto (Garrett 2005)

Ajax-tekniikalla toteutetut sivustot poistavat jatkuvien sivulatausten ongelman. Tämä tekniikka mahdollistaa selaimen keskustelevan palvelimen kanssa taustalla ilman turhia sivulatauksia, jolloin myös palvelimesta saadaan enemmän

kapasiteettiä käyttöön perinteiseen sivustoon verrattuna. Kuvion 1 alaosassa näkyy Ajax-tekniikan tapa kommunikoida palvelimen kanssa jatkuvasti.

Sivuston käyttäjälle tämä merkitsee vähemmän sivulatauksia, jolloin selaaminen nopeutuu sekä monipuolistuu. Ajax-tekniikan avulla voidaan tehdä dynaamisempia ja interaktiivisempia sivustoja kuin aikaisemmin. Ajaxin käytöstä hyödytään koko sivulla, mutta myös yksittäisissä osioissa, kuten hakupalkeissa tai tietojenkeruulomakkeilla. Hyvänä esimerkkinä toimivasta Ajax-toteutuksesta on Googlen hakusivusto, joka osaa ehdottaa haettavaa sanaa jokaisen syötetyn kirjaimen jälkeen kuvion 2 mukaisesti. Myös sivulla, jossa näytetään hakutuloksia, hakukone päivittää sivua Ajaxin avulla.



KUVIO 2. Googlen haku

4 SELAINOHJELMOINTI

4.1 HTML

4.1.1 Mitä on HTML?

Internet-selaimet muotoilevat WWW-sivut HTML-dokumenttien mukaisesti (Linjama 2001, 12). HTML-dokumentti kertoo selaimelle standardoidulla tavalla miten sivu tulisi esittää rakenteen ja sisällön suhteen, se ei ota kantaa ulkoasuun. HTML ei ole ohjelmointikieli, vaan se on kuvauskieli. Erilaisilla tageilla selaimelle kerrotaan standardoidusti sivuston rakenne.

JavaScriptillä pystytään määrittelemään sivuston käyttäytymistä, kun taas HTML sisältää vain sisällön ja rakenteen (Smith & Negrino 2007, 17). Näin huomataan, että moderni sivusto monesti sisältää huomattavasti enemmän kuin pelkkää HTML-koodia.

4.1.2 Selaimet

Selaimia on nykyaikana lukuisia, ja myös laitteistot ovat muuttuneet ajan myötä. Ohjelmoijat joutuvat huomioimaan isot näytöt, kuin myös hyvin pienet, esimerkiksi matkapuhelinten näytöt. Myös käytettävyyden kannalta on tarpeen huomioida kosketusnäytöt, joita löytyy tablet-tietokoneista sekä matkapuhelimista. Kosketusnäytöt on huomioitava sivuston suunnittelussa, jotta kaikki sivun osat ovat toimivia ilman hiiren käyttöä.

2011	Internet Explorer	Firefox	Chrome	Safari	Opera
February	26.5 %	42.4%	24.1%	4.1%	2.5%
January	26.6 %	42.8%	23.8%	4.0%	2.5%

KUVIO 3. Eri selainten käyttöasteet (w3schools.com 2011a)

Suosituimpia selaimia nykyään ovat Microsoft Internet Explorer, Mozilla Firefox, Apple Safari sekä Google Chrome. Selainten käyttöasteet selviävät kuviosta 3. Näistä selaimista on lisäksi useita eri versioita käytössä, joka osaltana hankaloittaa ohjelmoijien työtä. Varsinkin vanhempien selainten säilyminen käytössä niiden puutteista huolimatta. Parhaana esimerkkinä voidaan pitää Internet Explorer 6 –selainta, joka on edelleen laajasti käytössä, vaikka käytössä on jo Internet Explorer 9, kuten kuvio 4 osoittaa.

2011	Total	IE 9	IE 8	IE 7	IE 6
February	26.5 %	0.6 %	16.7 %	5.7 %	3.5 %
January	26.6 %	0.5 %	16.6 %	5.7 %	3.8 %

KUVIO 4. Microsoft Internet Explorer –selain (w3schools.com 2011b)

4.2 CSS

CSS eli Cascading Style Sheet on keino määritellä ulkoasuja sivustoille. Tyylit määritellään jokaiselle elementille erikseen erilliseen tyyli-tiedostoon, joten verkkosuunnittelijat voivat työstää yhtäaikaa HTML:ää sekä CSS-tyylimäärityksiä. Ulkoasun vaihto tai korjaaminen onnistuu myös helposti erillisen tyylitiedoston ansiosta. (Smith & Negrino 2007, 263.)

CSS-tyylitiedostolla voidaan vaikuttaa sivun ulkoasuun huomattavasti monipuolisemmin, kuin HTML-koodin sisään kirjoitetuilla tyyleillä. Näin päästään myös kehittämään sivustoa yhtäaikaa tyylillisesti sekä sisällöllisesti (Linjama 2001, 507.)

Tyylien kirjoitus HTML-koodiin voi toimia joillain pienemmillä sivustoilla, jossa ei ole tarpeen tehdä muutoksia, tai tyylimäärityksiä on vain hyvin vähäinen määrä.

Ulkoisia tyylitiedostoja voidaan myös kierrättää ja käyttää samaa tiedostoa useilla sivuilla tai sivustoilla. (Linjama 2001, 507.)

4.3 JavaScript

4.3.1 Mitä on JavaScript?

JavaScript on ohjelmointikieli, jolla voidaan luoda interaktiivisuutta verkkosivuille verrattuna pelkästään HTML:llä toteutettuihin sivuihin. JavaScript-koodia voidaan tuottaa HTML-koodin sisään tai erilliseen JavaScript-tiedostoon, joka on nykyisin suositellumpi tapa. (Smith & Negrino 2007, 2.)

Varsinkin dynaamisissa sivustoissa JavaScriptin käyttö on hyödyllistä ja osissa ratkaisuista suorastaan pakollista. Microsoftilla on JavaScriptille kilpaileva tuote VBScript, jolla on oma käyttäjäkuntansa yrityksissä, jotka käyttävät pääsääntöisesti Microsoftin työkaluja verkkosivustojen tekoon.

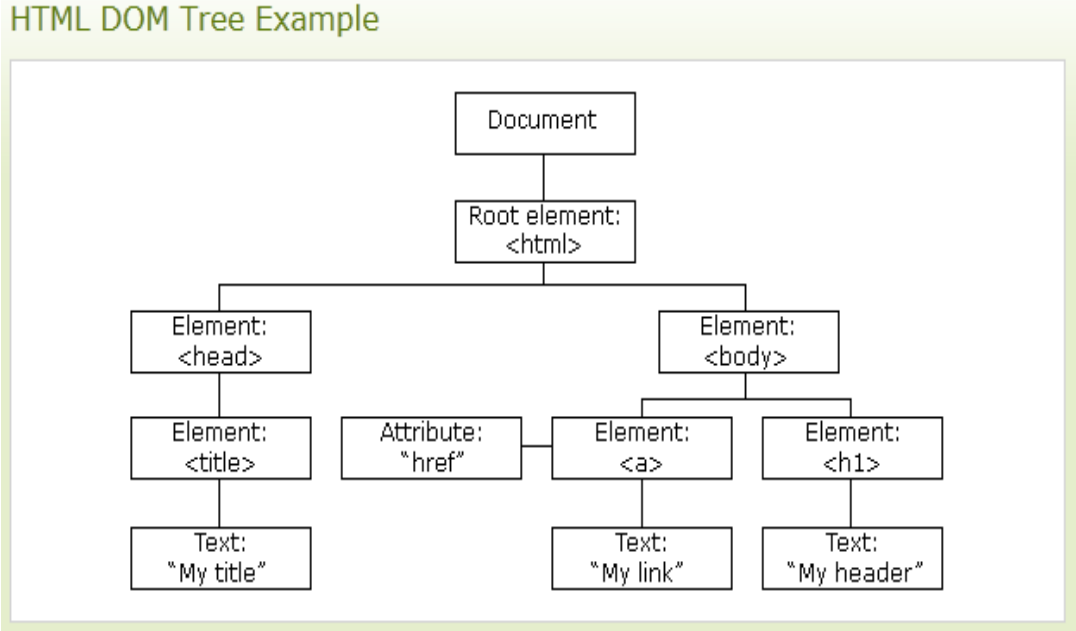
4.3.2 Mihin JavaScriptiä käytetään?

Yksinkertaisimmillaan JavaScriptillä voidaan vaihtaa sivulla olevan napin väriä, kun käyttäjä vie hiirensä osoittimen napin päälle. Toinen tyypillinen esimerkki on lomakkeiden tietojen tarkistus: onko käyttäjä syöttänyt tiedot oikeassa muodossa ja onko kaikki kentät täytetty? (Smith & Negrino 2007, 6.)

Pelkällä HTML:llä lomaketietojen tarkastus ei onnistu, joten on turvauduttava Javascriptiin, ellei halua lomakkeita, jotka hyväksyvät kaiken kirjoitetun sisällön. Esimerkiksi sähköpostiosoitteen oikea muoto on helppo tarkastaa JavaScriptillä. Tämä taas helpottaa työmäärää palvelimella huomattavasti, varsinkin jos kyseessä on virallisia sopimuksia, joissa tietojen oikeellisuus on välttämätöntä.

4.4 DOM

4.4.1 Document Object Model



KUVIO 5. HTML DOM rakenne-esimerkki (w3schools.com 2011c)

DOM eli Document Object Model on rajapinta, jonka avulla päästään muokkaamaan HTML- sekä XML-dokumentteja. Se määrittelee loogisen rakenteen näille dokumenteille ja tavan, jolla niitä pääsee muokkaamaan. (Robie 1998.)

Yleisesti HTML-dokumentti esitetään puumaisena rakenteena kuvion 5 mukaisesti ja DOM:n hienous piileekin standardoidussa tavassa käsitellä dokumentteja, joten kaikkien selainten on noudatettava samoja sääntöjä DOM:n kohdalla.

4.4.2 DOM-käsittely

DOM-sisällön käsittelyyn käytetään yleisesti JavaScriptiä, jossa on suuri määrä ominaisuuksia DOM-elementtien käsittelyyn. Elementtejä voidaan käsitellä yksitellen manuaalisesti tai yksinkertaisesti innerHTML-ominaisuuden avulla. (Smith & Negrino 2007, 284.)

```
<script type="text/javascript">
function notEmpty(){
    var myTextField = document.getElementById('myText');
    if(myTextField.value != "")
        alert("You entered: " + myTextField.value)
    else
        alert("Would you please enter some text?")
}
</script>
<input type='text' id='myText' />
<input type='button' onclick='notEmpty()' value='Form Checker' />
```

KUVIO 6. HTML DOM -koodiesimerkki

JavaScriptillä päästään käsittelemään HTML:n elementtejä monilla tapaa. getElementById on yksi käyttökelpoinen ominaisuus monissa eri tapauksissa. Kuviossa 6 sillä haetaan input-kenttään syötetty arvo. Tämän jälkeen tutkitaan, onko kenttään syötetty mitään. Tämä tietenkin vaatii elementtien nimeämistä tunnisteilla eli id:llä. Id:n hyvä puoli on sen yksilöllisyys, joten tiettyä tunnistetta ei ole kuin yksi, ja näin ollen sen löytäminen DOM-puusta on helppoa. Toinen käyttökelpoinen ominaisuus on getElementByName, joka palauttaa nykyisen elementin lapsisolmut, joilla on tietty nimi.

4.5 Ajax

4.5.1 Mitä on Ajax?

Ajaxin paras puoli on asynkronisuudessa. Käyttäjälle se merkitsee vähemmän sivulatauksia – sivua ladataan vain tarvittaessa – sivusto työskentelee taustalla, ja käyttäjä pystyy työskentelemään odottamatta sivun latautumista jokaisen hiirenpainalluksen välissä. (Riordan 2008, 7.)

Ajax on tällä hetkellä varteenotettava tekniikka toteuttaa dynaamisia verkkosivuja. Se ei ole uusi teknologia vaan enemmänkin tekniikka, jossa JavaScript on vahvasti osallisena. Tästä johtuen sen käyttöönotto ei vaadi ohjelmoijilta uuden kielen opettelemista, jos JavaScript on ennestään tuttu. (Asleson & Schutta 2007, 14.)

4.5.2 Milloin käyttää Ajaxia?

Ajaxin käytölle on monia hyviä syitä, mutta asiaa on aina mietittävä sivustokohtaisesti. Onko järkeä sisällyttää sivulle Ajaxia vain sen itsensä takia vai saavutetaanko sillä jotain? (Asleson & Schutta 2007, 20.)

Esimerkiksi lomakkeiden tietojen tarkastus on helppo tehdä Ajaxilla, jolloin tarkastus voidaan tehdä samaan aikaan, kun käyttäjä syöttää tietoja ja antaa palautteen välittömästi. Perinteiseen tapaan tarkastus tehdään vasta, kun käyttäjä lähettää lomakkeen tiedot palvelimelle, joten ainut ero on tarkastuksen reaaliaikaisuus.

Ajaxin käytölle kannattaa olla jokin selvä tarkoitus, josta kaikki hyötyvät ja se ei monimutkaista sivuston rakennetta. Esimerkiksi pienelle sivustolle, jossa on hakumahdollisuus sivuston sisältä, ei välttämättä kannata laittaa Googlen hakukoneen tapaista sanojen ehdottelua. Tällaista voisi ajatella sivustolle, jossa on jatkuvasti kasvava tietomäärä, jolloin ehdotuksista olisi apua.

4.5.3 XMLHttpRequest-objekti

Tärkein Ajaxin osista on XMLHttpRequest-objekti, joka tuli tuetuksi Internet Explorer 5 –selaimeen vuonna 1999 ja on saanut tuen myös muihin selaimiin hetkeä myöhemmin. Nykyään tuettuina ovat kaikki yleisimmät selaimet, vaikkei XMLHttpRequest-objekti olekaan W3C-standardi. Objektin muodostaminen vaatii selaimen tunnistamisen, koska Internet Explorer on ainut selain, joka toteuttaa objektin ActiveX-komponenttina muiden selainten toteuttaessa JavaScript-oliona. (Asleson & Schutta 2007, 25.)

Tämä tarkoittaa tunnistuksen rakentamista esimerkiksi kuvio 7 mukaisesti. Rivillä 107 tutkitaan, onko selaimena jokin muu kuin Internet Explorer. Rivit 111 ja 115 tutkivat Intenet Explorerin eri versiot. Lopputulos on joka tapauksessa sama - XMLHttpRequest-objekti on luotu. Objektin avulla pystytään lähettämään pyyntöjä palvelimelle ja käsittelemään vastauksia (Asleson & Schutta 2007, 25).

```
104 function GetXmlHttpObject(){
105     var request;
106     try {
107         request = new XMLHttpRequest();
108     }
109     catch(tryMS){
110         try {
111             request = new ActiveXObject("Msxml2.XMLHTTP");
112         }
113         catch(otherMS){
114             try {
115                 request = new ActiveXObject("Microsoft.XMLHTTP");
116             }
117         catch(failed){
118             request = null;
119       }
120      }
121     }
122     return request;
123 }
```

KUVIO 7. XMLHttpRequest-objektin luonti

4.5.4 Pyyntöjen tilat

Ajax pyrkii olemaan reaaliaikainen tekniikka, mutta siinä voi myös esiintyä viivettä. Esimerkiksi verkko ei ole riittävän nopea tai palvelimella on ruuhkaa. Näissä tapauksissa JavaScript-koodin on osattava toimia palvelimelta tulevien tilakoodien mukaan. Tämä tapahtuu tutkimalla XMLHttpRequest-objektin tilaa, joka ilmenee readyState-ominaisuudesta numeroilla 0-4. (Asleson & Schutta 2007, 29.)

readyState	Holds the status of the XMLHttpRequest. Changes from 0 to 4: 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready

KUVIO 8. readyState (w3schools.com 2011d)

XMLHttpRequest-objektin tilat näkyvät kuviossa 8. Nolla on lähtötilanne, jolloin pyyntöä ei ole vielä alustettu. Tila numero yksi on merkkinä palvelinyhteyden muodostamisesta, jota seuraa tila numero kaksi, jolloin pyyntö on vastaanotettu. Kolmas tila tarkoittaa pyynnön käsittelyä palvelimella ja tästä seuraa neljäs tila, joka ilmoittaa vastauksen olevan valmiina. Tämän jälkeen vastaus voidaan syöttää sivustolle.

4.5.5 innerHTML ja responseText

innerHTML on JavaScriptin ominaisuus DOM-puun käsittelyssä. Sillä päästään käsiksi tagien välissä olevaan sisältöön suoraan ilman kajoamista itse tageihin. Tämä mahdollistaa tagien sisällön helpon muokkaamisen ja esimerkiksi tekstejä ja kuvia voidaan vaihtaa koskematta rakenteeseen. (Asleson & Schutta 2007, 43.)

responseText on XMLHttpRequest-objektin ominaisuus, jolla palvelimelta tuleva vastaus saadaan tekstimuodossa. Toinen vaihtoehto on responseXML, joka palauttaa vastauksen XML-oliona. (Asleson & Schutta 2007, 42.)

```
xmlhttp.onreadystatechange=function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
    document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
    }
  }
xmlhttp.open("GET","ajax_info.txt",true);
xmlhttp.send();
```

KUVIO 9. innerHTML:n käyttö responseText:n kanssa

Yhdistettäessä responseText sekä innerHTML, saadaan käyttökelpoinen tekniikka palvelimelta tulevan tiedon liittämiseen HTML-sisältöön. Palvelimelta tuleva vastaus syötetään myDiv-elementin sisään kuviossa 9. Tämä onkin yksi Ajaxin keskeisistä asioista, koska sen avulla voidaan muokata sivuston sisältöä.

4.5.6 Takaisinkutsu

Onreadystatechange-funktio asetetaan odottamaan muutoksia kutsun tilassa. Jos tila muuttuu, kutsutaan takaisinkutsufunktiota. Kuviossa 9 takaisinkutsufunktio function() pitää sisällään ne toimenpiteet, mitä suoritetaan tilan muuttuessa. Kuvion 9 esimerkissä ainoastaan tilalle neljä on asetettu toiminnallisuuksia. Viimeinen toimenpide on kutsun lähettäminen send()-funktiolla, jonka jälkeen takaisinkutsufunktio alkaa saamaan tilamuutoksia vastauksen saapuessa. (Shannon 2011.)

Kuviossa 9 if-lauseen ehdoiksi on määrätty xmlhttp-objektin readyState, jonka on oltava neljä, sekä saman objektin status, joka kertoo palvelimen tilasta. Status 200 on tila, jolloin palvelin on toiminnassa normaalisti.

4.5.7 Ulkoasun erottaminen muusta koodista

Tärkeintä ohjelmoijan sekä graafikon töiden erottamisessa on kaiken ulkoasuun liittyvän sisällön kirjoittaminen erilliseen tyylitiedostoon. Tällöin muutoksia ulkoasuun voi tehdä häiritsemättä sivuston logiikkaa, kunhan muistaa käyttää samoja luokkia ja muuttujanimiä. Esimerkiksi taustakuvan vaihtoon voi kirjoittaa JavaScript-koodin, jolla muutetaan taustakuvaa tietyin väliajoin. Tässä kannattaa viitata CSS:n sisälle luokkiin eikä yksittäisiin kuviin. (Riordan 2008, 84.)

Ulkoasun erottaminen helpottaa työskentelyä, varsinkin, jos kyseessä on iso projekti. Kommunikaatio graafikon ja ohjelmoijan välillä on eräs tärkeimmistä asioista onnistuneen projektin toteuttamiseen. Osioiden erottamisella päästää hyviin tuloksiin, mutta se vaatii neuvotteluita ja sopimisia, millaista rakennetta käytetään, ja mitkä ovat luokkien ja elementtien nimet. Tämä vaatii kurinalaisuutta, jotta pysytään sovituissa raameissa projektin aikana. Muutosten tekeminen konsultoimatta toista osapuolta voi johtaa suuriinkiin muutoksiin muissa osioissa.

Ulkoasuihin voi kohdistua paljon muutoksia varsin nopeaan tahtiin, joten sitä on voitava hallita ilman pelkoa toiminnallisuuksien menettämisestä. Tässä avainasemassa on sovituissa määreissä pysyminen.

5 PALVELINOHJELMOINTI

5.1 PHP

5.1.1 Mitä on PHP?

PHP on palvelinpään skriptauskieli, joten PHP-sisältö ei ole näkyvissä selaimessa. Se on helpompaa kirjoittaa kuin kilpailijansa, ja se kehittyy jokaisen julkaisun yhteydessä (Appu 2002, 12). Osakseen näistä syistä PHP on levinnyt käytetyimpien palvelinkielien. Varsinkin yhdistettynä MySQL-tietokantaan PHP:stä tulee hyvin monipuolinen ja sopii käytettäväksi monille erilaisille sivustoille.

PHP:n perusominaisuuksiin kuuluu myös: tiedostojen luku ja kirjoitus, sähköpostin lähetys sekä sitä voi sisällyttää osaksi HTML-koodia, jolloin sivustolle saadaan dynaamisuutta (Appu 2002, 12). Hyvin suosittua on HTML-koodin sisään kirjoittaminen, esimerkiksi HTML:llä voidaan luoda taulu, jonka sisältö haetaan PHP:llä MySQL-tietokannasta. Tällaisen ratkaisun etuna on tietenkin kirjoitettavan koodin vähäinen määrä ja muutosten tekemisen helppous. Enää ei tarvitse syöttää kaikkia taulun alkioita käsin HTML:ään, vaan ne haetaan suoraan tietokannasta, joten tietokantaan tehdyt muutokset päivittyvät suoraan HTML-tauluun. Tämä helpottaa ylläpitoa, koska samaa tietokantaa voidaan käyttää myös muilla sivuilla.

PHP:llä päästään HTML:ää vaativampiin toimenpiteisiin. Enää ei rajoituta yksittäisen sivun näyttämiseen vaan voidaan hyödyntää kaikki palvelinresurssit ja tehdä huomattavasti monipuolisempia asioita pelkkään HTMLään verrattuna.

5.1.2 GET ja POST

GET- ja POST-metodit ovat HTML:n metodeja, joilla lähetetään tietoja lomakkeilta palvelimelle. Näiden käytössä tulee olla kuitenkin tarkkana, koska ne eroavat toisistaan huomattavasti ja sopivat erilaisiin kohteisiin.

Näiden metodien käsittelyssä on eroja, joten ne on otettava huomioon sivustoa suunniteltaessa. GET-metodin syötteet luetaan aina URL:stä, esimerkkinä Facebook-sivuston Etsi-kenttä kuviossa 10. Hakusana tässä on haku, joka on muuttujan q arvona. Muuttuja q on erotettu muusta osoitteesta kysymysmerkillä, joten PHP:llä etsitään muuttujaa q, joka on kysymysmerkin jäljessä. Muut muuttujat erotetaan &-merkillä.

KUVIO 10. Facebook-haku

GET-metodi kirjoittaa siirrettävän datan suoraan URL:in perään, joten osoiterivillä näkyy lomakkeen sisältö. Tästä syystä GET-metodia ei voida käyttää salasanojen ja muiden luottamuksellisten tietojen välitykseen. Se ei myöskään sovellu isojen tietomäärien lähetykseen, koska jotkut selaimet eivät sallia pitkiä URL:iä. (Appu 2002, 143-144.)

Get-metodin hyvänä puolena on näkyvyys, jolloin samaa osoitetta voidaan käyttää uudelleen. Esimerkiksi haluttaessa hakea samalla hakusanalla GET-metodia käyttävällä hakukoneella, voidaan käyttää samaa kirjanmerkkiä kerrasta toiseen.

POST-metodi välittää tiedot eri tavalla. Se ei syötä mitään URL:n joukkoon, vaan lähettää tiedot kutsun osana. Näin osoitetta ei voida hyödyntää uudelleen saman kutsun lähettämiseen. Monesti siitä ei ole haittaa, koska palvelimen tilaa ei haluta

muuttaa uudelleen. POST-metodi sopii hyvin arkaluontoisen tiedon lähettämiseen sekä pitkien tekstien lähettämiseen. (Appu 2002, 144.)

POST-metodia kannattaa käyttää tilanteissa, jolloin halutaan palvelimella tapahtuvan muutoksia tai lähetyksen pituus on suuri. Esimerkkinä hyvästä käyttöpaikasta voisi olla blogi, koska GET-metodilla pidempi teksti ei välittyisi oikein palvelimelle eikä myöskään samaa tekstiä tule lähetettyä vahingossa moneen kertaan.

PHP:llä GET:n ja POST:n arvojen luku tapahtuu $_GET- ja $_POST-funktioilla. Erona näissä on paikka mistä arvoja etsitään: $_GET-funktio etsii URL:n sisältä muuttujaa, kuten kuviossa 10, ja $_POST-funktio etsii arvoa kutsusta itsestään input-kentän name-attribuutin nimellä.

5.1.3 PHP ja Ajax

Eräänä esimerkkinä Ajaxin sekä PHP:n yhdistämisestä voisi olla käyttäjätunnuksen ja salasanan tarkistaminen. JavaScriptillä luetaan käyttäjän kirjoittamat tiedot, minkä jälkeen JavaScript lähettää ne palvelimelle PHP:n käsiteltäväksi. PHP:llä voidaan ottaa yhteyden tietokantaan ja tarkastaa, onko käyttäjätunnus sekä salasana oikein. Tämän jälkeen PHP:n tulee lähettää vastaus, kyllä tai ei, jonka JavaScript lopulta kertoo käyttäjälle. Ajax-tekniikan suosiminen päästää käyttäjän näkemään vastauksen ilman sivulatausta, jolloin lomakkeen tiedot tulevat kerralla oikein. (Meloni 2003, 192.)

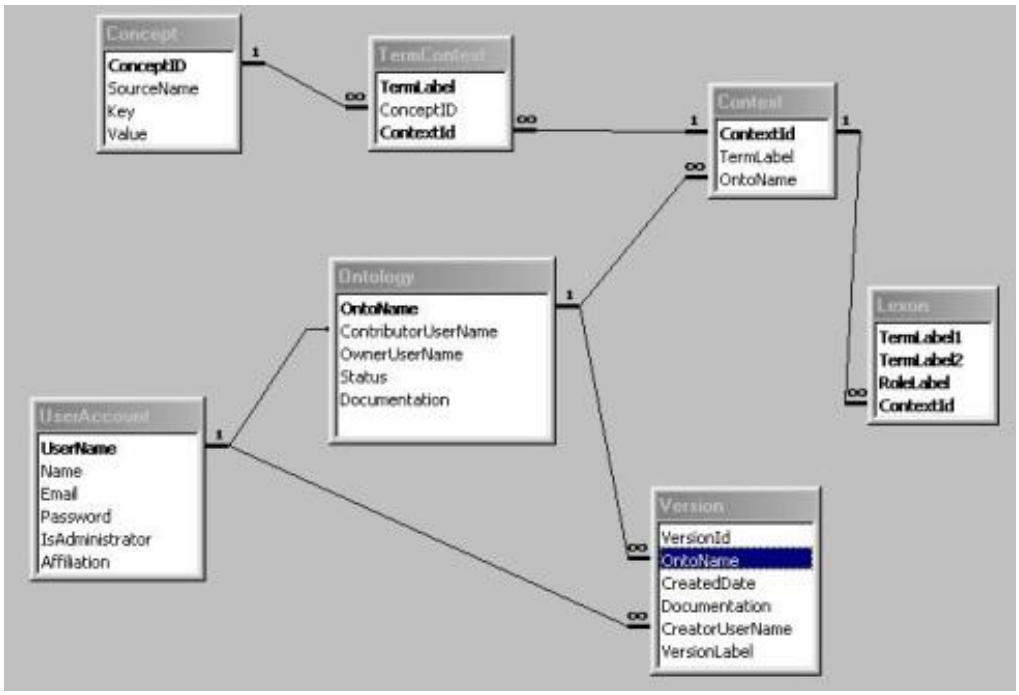
Tähän voidaan liittää myös käyttäjätunnuksen ja salasanan kryptaus, joten ulkopuoliset eivät pääse käsiksi näihin. Näin voidaan tietenkin tarkastaa kaikki muutkin tiedot, mitä lomakkeille halutaan. Tällä saadaan etua, koska esimerkiksi käyttäjätunnuksen vapaanaolo voidaan tarkistaa ilman sivulatausta. Varsinkin isoilla kansainvälisillä sivustoilla voi olla ongelmana käyttäjätunnusten luonti, koska omat suosikit voivat olla varattuina.

5.2 Tietokannat

5.2.1 Relaatiotietokannat

Yksinkertaistettuna relaatiotietokannoilla tarkoitetaan yhteen liitettyjä tauluja, jotka koostuvat sarakkeista ja riveistä. Nämä ovat yhteydessä toisiinsa tiettyjen arvojen pohjalta. Esimerkiksi verkkokaupan koko katalogi voi olla relaatiotietokannassa, kaikki tuotteet ovat jotenkin yhteydessä toisiinsa, koska ne ovat yleensä lajiteltua ryhmittäin. (Meloni 2003, 8.)

Tämän lisäksi eri tuotteilla voi olla muitakin yhteyksiä kuin pelkkä tuoteryhmä. Monissa verkkokaupoissa ostaja voi myös nähdä, mitä muut saman tuotteen ostaneet ovat ostaneet, tai verkkokauppa voi näyttää vaihtoehtoisia tuotteita tai palveluita. Kuviossa 11 on esimerkki relaatiotietokannasta, jossa havainnollistetaan eri tauluissa olevien arvojen yhteyttä toisiinsa. Taulussa UserAccount olevaa UserName-kenttää käytetään myös tauluissa Ontology ja Version, mutta näissä on eri kentän nimi.



KUVIO 11. Esimerkki relaatiotietokannasta (Velasquez 2010)

Standardikieli tietokannoissa on SQL, eli Structured Query Language. Sen olemassaolo mahdollistaa tietokantajärjestelmien vaihdon kesken kaiken ja helpottaa siirtymistä järjestelmästä toiseen, jos tarvetta vaihdokseen esiintyy. Tällöin kaikkia järjestelmiä ei tarvitse rakentaa uudelleen, vaan selvitään pienillä muutoksilla. (Meloni 2003, 10.)

5.2.2 MySQL

MySQL on tietokantojen hallintajärjestelmä, joka on levinnyt laajalti yritys- sekä kotikäyttöön. Eräänä etuna on myös sen sopivuus lähes kaikkien ohjelmointikielien kanssa unohtamatta helppoa siirreltävyyttä. (Meloni 2003, 10-11.)

Varsinkin alhainen tai ilmainen hankintahinta sekä laaja tuki eri kielille ovat saaneet MySQL:n suureen suosioon, eikä se aseta ehtoja käyttöjärjestelmille vaan on täysin siirreltävissä.

Tietokannan käyttö perustuu SQL-käskyihin, joita on lukuisia. Peruskomentoina mainittakoon taulun tietojen lukeminen SELECT-komennolla sekä tauluun lisääminen INSERT INTO –komennolla. Käskyissä voidaan määritellä hakuparametrejä, hakea useasta taulusta kerralla ja tehdä muita tarvittavia toimenpiteitä.

MySQL:n käyttö onnistuu monin keinoin. Tietokannan hallintaan liittyviä toimenpiteitä voidaan suorittaa komentoriviltä tai erilaisilla graafisilla käyttöliittymillä varustetuilla ohjelmilla. Eräs suosittu vaihtoehto on phpMyAdmin, joka on graafinen ohjelma MySQL:n käyttöön. Vaatimuksena phpMyAdminin käytölle on PHP:n asennus sekä verkkopalvelin. (Meloni 2003, 48.)

PhpMyAdminin avulla tietokantojen luonti onnistuu helposti graafisuuden ansiosta. Sillä onnistuvat kaikki MySQL:n toiminnot, ja käyttö tapahtuu selaimella. Komentojen suorittamiseen on kaksi tapaa: lomakenäkymä tai käskyjen manuaalinen kirjoittaminen.

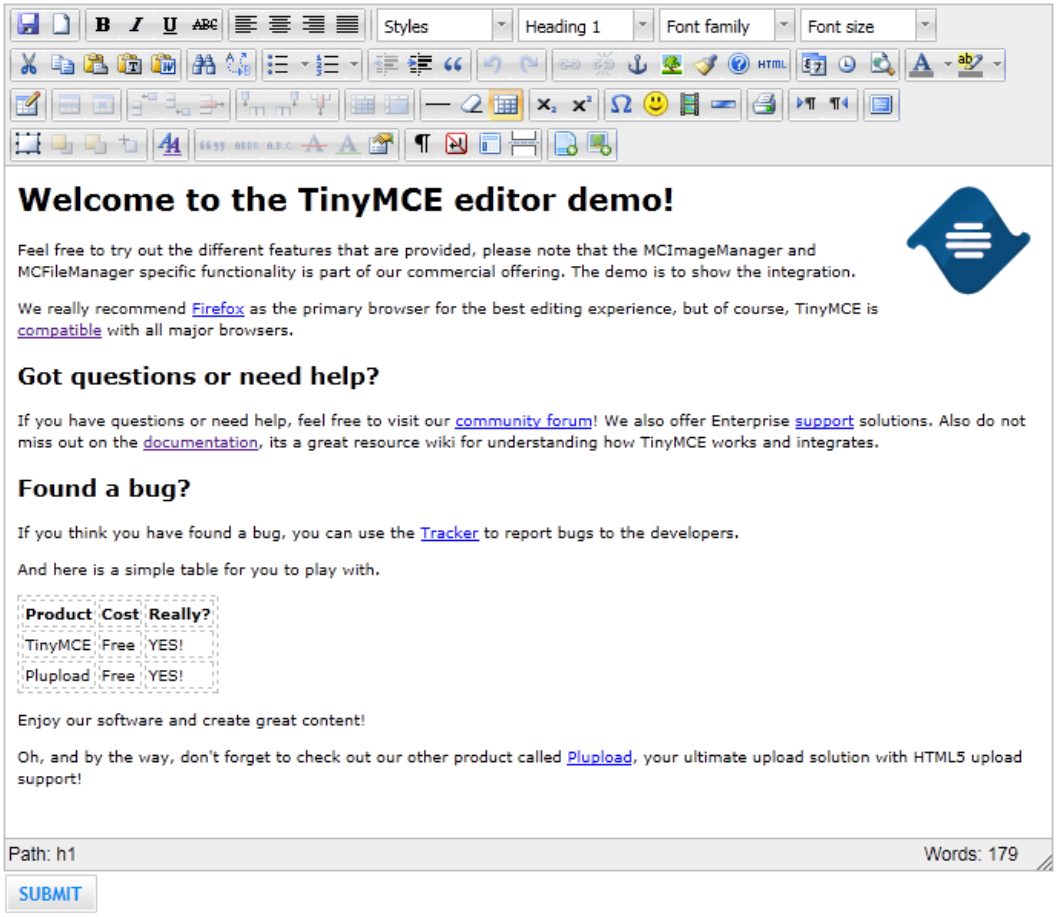
PHP on yksi ohjelmointikielistä, joihon on tehty valmiita funktioita MySQL-tietokantaan kytkeytymistä varten. Kyselyiden ja vastausten käsittely PHP:lla on helppoa, ja tästä syystä PHP ja MySQL ovat monissa verkkosovelluksissa yhdessä käytettyjä. (Meloni 2003, 298.)

5.3 TinyMCE

5.3.1 TinyMCE-tekstieditori

TinyMCE on Moxiecode Systemsin kehittämä tekstieditori HTML-sisällön muokkaamiseen käyttäjäystävällisesti. TinyMCE tarjoaa käyttäjälle tavallisen tekstieditorin, joka muuttaa kirjoitetun sisällön HTML-muotoon. (Moxiecode Systems AB, 2011.)

Se sopii myös moniin erilaisiin sovelluksiin, kuten yritysten intraneteistä ja sisällönhallintajärjestelmiin. TinyMCE on lisensoitu GNU Lesser General Public License alle, joten sen käyttäminen on vapaata ilman maksuja.



KUVIO 12. TinyMCE-tekstieditori (Moxiecode Systems AB 2011)

TinyMCE on JavaScript-pohjainen editori, joten palvelimeen ei tarvitse tehdä muutoksia muutamia pieniä poikkeuksia lukuunottamatta. Yksinkertaisesti ladataan paketti palvelimelle, puretaan paketti oikeaan hakemistoon ja aletaan muokata editoria omiin tarpeisiin sopivaksi. Muokkausmahdollisuuksia editorissa on paljon: varsinkin ulkoasuun ja käytettäviin toimintoihin on valmiita malleja sekä elementtejä. (Moxiecode Systems AB, 2011.)

Käytettävien osien valitseminen tapahtuu määrittämällä koodilla, mitä näppäimiä halutaan käyttää, kun editori avataan. Editoriin on saatavilla erilaisia valmiita teemoja, lisäosia sekä työkalupalkkeja työskentelyä helpottamaan, kuten kuviosta 12 näkyy. Kuvion editorissa on aktiivisena kaikki valmiina saatavat muotoilutyökalut.

Browser	Windows XP	Mac OS X	GNU/Linux
IE 9	OK	N/A	N/A
IE 8	OK	N/A	N/A
IE 7	OK	N/A	N/A
IE 6	OK	N/A	N/A
Firefox 2.x	OK	OK	OK
Firefox 3.x	OK	OK	OK
Firefox 4.x	OK	OK	OK
Chrome	OK	OK	OK
Safari 3.x	OK	OK	N/A
Safari 4.x	OK	OK	N/A
Safari 5.x	OK	OK	N/A
Opera	OK	OK	OK
Konqueror	N/A	N/A	No
KonquerorKDE4	N/A	N/A	OK[2]

Notes

1. Safari 2 is no longer supported since Safari 3 is a priority upgrade for Tiger and Leopard so upgrade your OS instead.
2. Partially working.

KUVIO 13. Selainyhteensopivuus (Moxiecode Systems AB 2011)

Editori tukee kaikkia yleisimpiä selaimia kuvion 13 mukaisesti pääpainon ollessa Microsoft Internet Explorer-, Mozilla Firefox- ja Apple Safari-selaimissa. Vanhemmat selaimet ovat tuettuina myös. (Moxiecode Systems AB, 2011.)

5.3.2 iBrowser

iBrowser on lisäosa moniin verkkopohjaisiin tekstieditoreihin kuten TinyMCE, SPAW sekä htmlAREA. Ohjelman kehittäjä on net4visions.com. Tällä lisäosalla tekstieditoreihin saadaan kuvienlisäysmahdollisuus ilmaiseksi, esimerkiksi TinyMCE:n omasta kuvapalvelusta joutuu maksamaan. (net4visions.com, 2011.)

Lisäosan toiminta on vahvistettu seuraavilla selaimilla: Microsoft Internet Explorer, Mozilla Firefox ja Apple Safari. Perusominaisuudet ovat kuvien

lataaminen palvelimelle, kuvien selaaminen palvelimella, kuvakansioiden muokkaus sekä kuvien koon muokkaaminen. iBrowser on PHP- sekä JavaScript-pohjainen tuote ja sen asennus vaatii palvelimella oikeuksien määrittämistä iBrowser-kansioihin sekä kuvakansioiden polkujen syöttämistä iBrowserin tietoihin. (net4visions.com, 2011.)

6 ICE EVENT -SIVUSTO

6.1 Ulkoasu

6.1.1 Graafinen ilme

Ulkoasuun panostettiin paljon resursseja, jotta asiakas olisi tyytyväinen lopputulokseen. Ulkoasun suunnittelu olikin yksi lähtökohdista sivuston suunnittelussa ja sen piti keventää varsin raskasta sisältöä, joka oli enimmäkseen tekstiä. Varsinkin taustakuviin asiakkailla oli paljon ideoita.

6.1.2 Taustakuvien vaihto

Sivustolle toteutettiin vaihtuvat taustakuvat, kun päävalikossa siirryttiin eri osioihin. Tämä tapahtui pelkästään JavaScriptillä Ajax-tekniikkaa hyväksikäyttäen. Kun käyttäjä valitsi päävalikosta jonkun osion, tämä käynnisti JavaScriptin onClick-tapahtuman, joka kutsui funktiota, jolla uusi taustakuva haettiin palvelimelta. Tässä ei nähty tarvetta mahdollisuudelle vaihtaa taustakuvia jatkuvasti, joten sivuston teemaan sopivista kuvista valittiin parhaat. Kuviossa 14 näkyy changeImage-funktion, jonka tarkoituksena on vaihtaa taustakuvia. Funktiota kutsuttiin päävalikon nappia painettaessa, jolloin taustakuva vaihdettiin id:n mukaiseksi.

```
function changeImage(img){
    if(img == 25){
        document.getElementById("logo1").style.background=
        "url('kuvat/taustat/tausta1_1.jpg') no-repeat";
        document.getElementById("logo2").style.background=
        "url('kuvat/taustat/tausta1_2.jpg') no-repeat";
    }
    if(img == 26){
        document.getElementById("logo1").style.background=
        "url('kuvat/taustat/tausta2_1.jpg') no-repeat";
        document.getElementById("logo2").style.background=
        "url('kuvat/taustat/tausta2_2.jpg') no-repeat";
    }
    if(img == 27){
        document.getElementById("logo1").style.background=
        "url('kuvat/taustat/tausta3_1.jpg') no-repeat";
        document.getElementById("logo2").style.background=
        "url('kuvat/taustat/tausta3_2.jpg') no-repeat";
    }
    if(img == 28){
        document.getElementById("logo1").style.background=
        "url('kuvat/taustat/tausta4_1.jpg') no-repeat";
        document.getElementById("logo2").style.background=
        "url('kuvat/taustat/tausta4_2.jpg') no-repeat";
    }
    if(img == 29){
        document.getElementById("logo1").style.background=
        "url('kuvat/taustat/tausta5_1.jpg') no-repeat";
        document.getElementById("logo2").style.background=
        "url('kuvat/taustat/tausta5_2.jpg') no-repeat";
    }
    if(img == 30){
        document.getElementById("logo1").style.background=
        "url('kuvat/taustat/tausta6_1.jpg') no-repeat";
        document.getElementById("logo2").style.background=
        "url('kuvat/taustat/tausta6_2.jpg') no-repeat";
    }
}
```

KUVIO 14. changeImage-funktio

Graafikko sai muokattua kuvista pienikokoisia, joten taustakuvien lataaminen ei ollut raskasta hitaammillakaan yhteyksillä. Työssä todettiin tarvittavan vain muutama taustakuva.

6.1.3 Napit

Päävalikon nappien piti olla havainnollisia, joten kuvituksen tuli liittyä aiheeseen. Nappia painaessa kutsuttiin JavaScript-funktiota showContent, joka välitti kyseisen napin id:n palvelimelle getContent.php-tiedostoon, josta tullut vastaus vaihtoi sivun sisällön kyseistä nappia vastaavaksi. Kuviossa 15 on kuvattuna showContent-funktio, jossa käytetään GET-metodia tiedon välitykseen. Nappeja tuli kuusi kappaletta, jotka ovat kuvion 15 mukaisessa järjestyksessä etusivu, tapahtuman tekeminen, linkit, yhteystiedot, tekijät ja sivukartta. Muutama lisänappi tuli myös valmiiksi, joten päätasoja pystyi tarvittaessa muuttamaan.

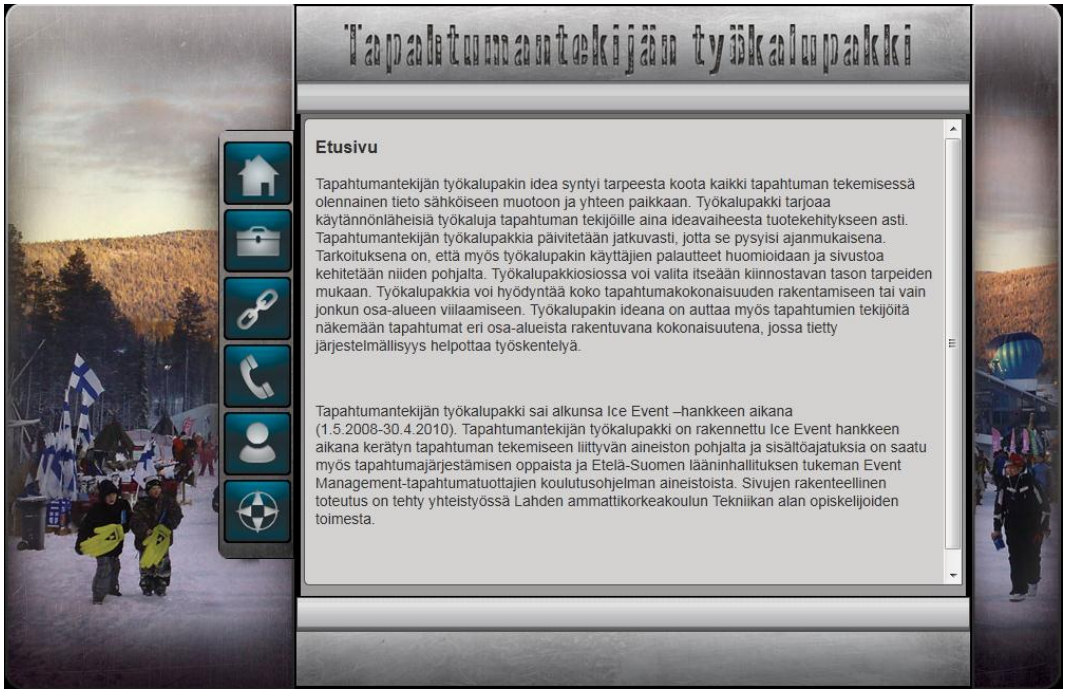
```
function showContent(str)
{

    xmlhttp=GetXmlHttpObject();
        if (xmlhttp==null)
                {
                alert ("Browser does not support HTTP Request");
                return;
                }
    var url="getContent.php";
    url=url+"?q="+str;
    url=url+"&sid="+Math.random();


    xmlhttp.onreadystatechange=stateChanged;

    xmlhttp.open("GET",url,true);
    xmlhttp.send(null);
}
```

KUVIO 15. showContent-funktio

6.1.4 Valikkorakenne

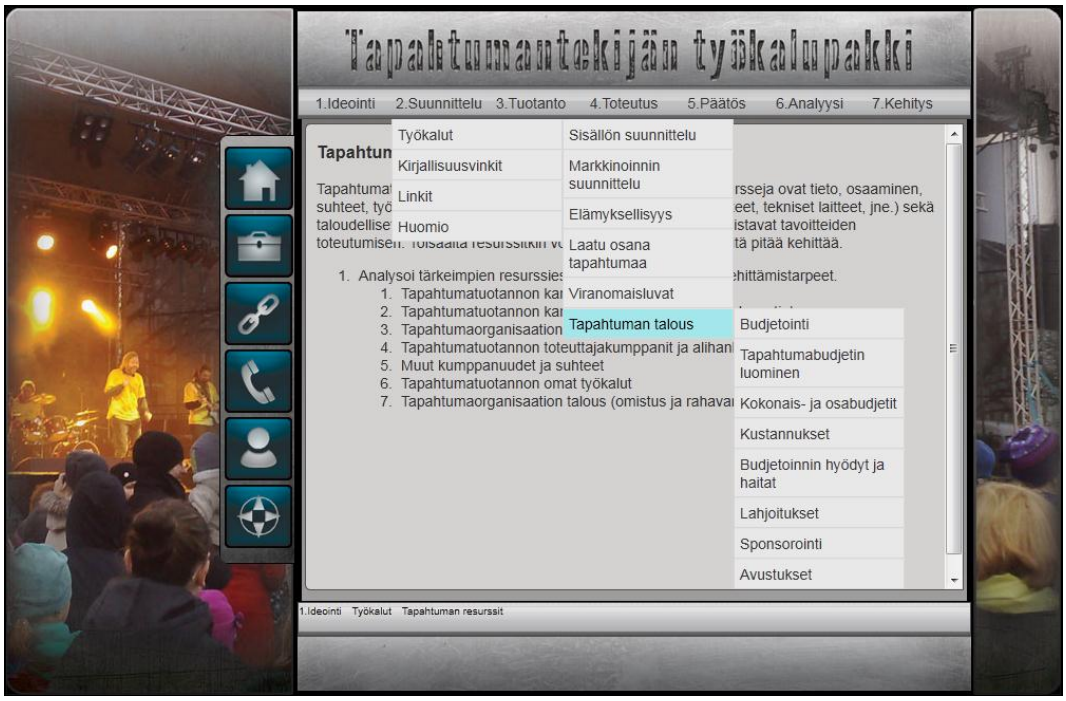
Sivustolle tuli kaksi valikkoa, joista päävalikko sijaitsi vasemmassa reunassa pystyssä ja sillä siirryttiin pääosioiden välillä. Päävalikkoon tuli kuusi nappia, jotka veivät sivuston eri osioihin. Pääosioiden vaihtuessa myös taustakuva

vaihtuu, joten tarvittiin erilliset taustakuvat jokaiselle osiolle. Kuviossa 16 näkyy sivuston etusivu sekä yksi kuudesta taustakuvasta.



KUVIO 16. Sivuston etusivu

Haastavampi valikko löytyy Tapahtuman tekeminen -pääosion alta. Tämä valikko oli sivuston suunnittelun lähtökohtana, joten valikon suunnittelu vei paljon aikaa. Valikosta tuli dynaaminen sekä vaakatasossaoleva. Tällä valikolla päästiin tutkimaan itse aineistoa, päävalikon ollessa vain päätasoilla liikkumista varten. Hiiren vieminen valikon osien päälle avasi uusia tasoja, jos niitä oli tarjolla. Alivalikoiden määräksi päätettiin neljä, koska muuten alivalikot veisivät liikaa tilaa, eikä asiakkaatkaan nähneet tarvetta suuremmalle määrälle. Myöskin valikon tekstien pituutta jouduttiin rajoittamaan, koska liian pitkät tekstit eivät sopineet sivulle. Kuviossa 17 näkyy valikon rakenne, jossa myös kaksiriviset kentät ovat mahdollisia.



KUVIO 17. Sivuston dynaaminen valikko

Valikon muotoilu tapahtui pelkästään CSS-koodilla. Näin valikko saatiin pidettyä yksinkertaisena. Tässä tapauksessa muotoilu jäi graafikon tehtäväksi käyttöliittymää suunniteltaessa. Sivun alareunaan tuli myös linkit nykyiselle sivulle ja edellisille tasoille.

6.1.5 Eri selainten huomiointi

Tarkoituksena oli tehdä sivusto yhteensopivaksi yleisimpiin selaimiin, poislukien hyvin vanhat versiot. Esimerkiksi Microsoft Internet Explorerissa toimivuus varmistettiin versioon 6 asti, koska vanhempien versioiden käyttäjiä sivustolla ei katsottu olevan.

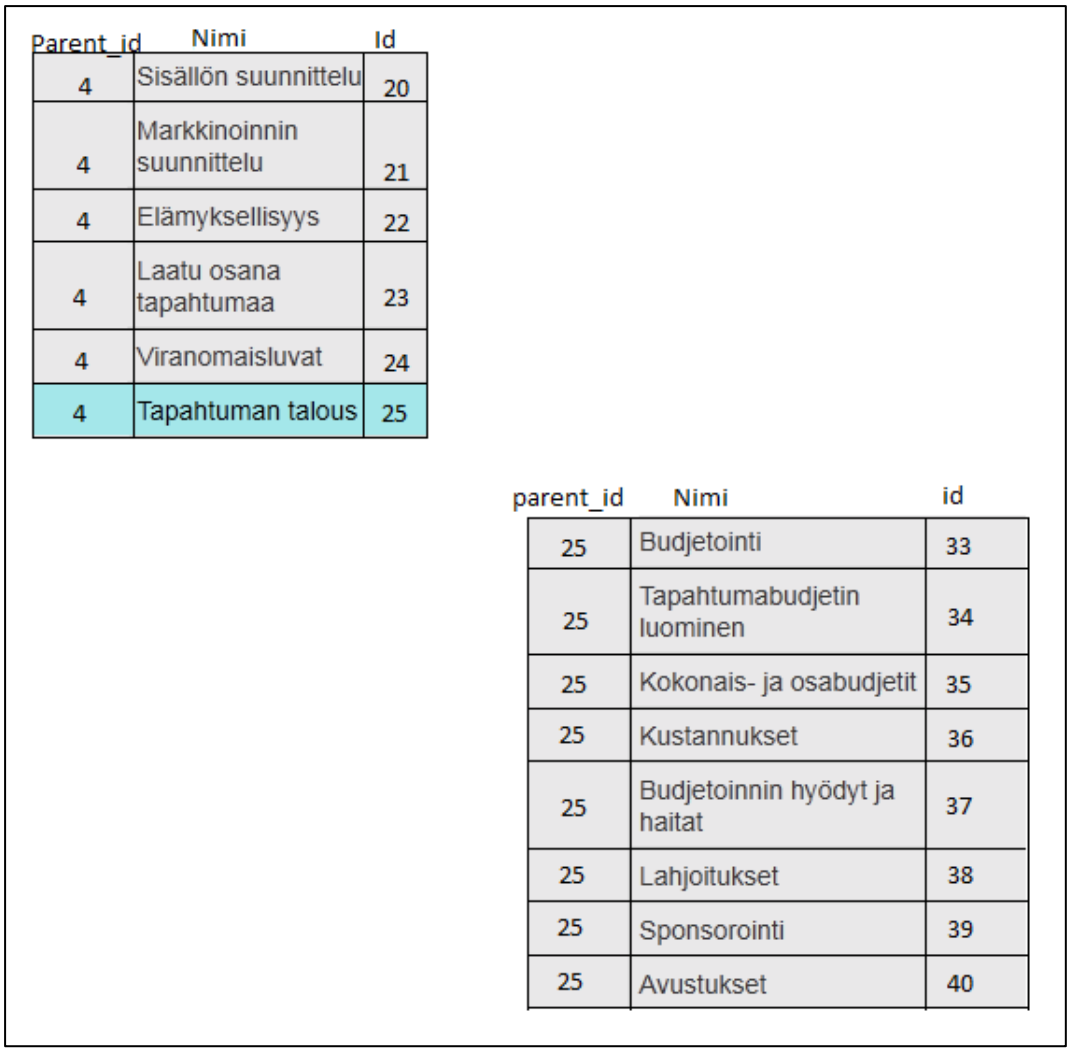
Tyylien tekeminen toimi varsin hyvin muille selaimille paitsi Internet Explorerille, jolle tarvitsi tehdä muutama poikkeussääntö pääsivun koodiin, koska samat muotoilusäännöt eivät päteneet. Muita selaimia ongelmat eivät koskeneet, ja ne toimivat testeissä hyvin.

6.2 Dynaaminen valikko

6.2.1 Tietokantarakenne

Tietokannaksi valittiin MySQL-tietokanta. Valinta oli varsin helppo: MySQL on ilmainen, MySQL oli koulun palvelimella jo valmiina, ja se tuli myös WampServerin mukana, joka oli käytössä sivustoa tehdessä.

Tietokannan suunnittelussa pääpaino oli ongelmassa miten toteuttaa puumainen tietokanta. Tietokantaan tuli yksi taulu käyttäjille, jotka pääsivät hallintasivustolle, mutta itse päätaulu piti sisällään kaiken informaation sivustolta. Päätaulun rakenteeksi tuli edellämainittu puumainen rakenne. Rakenteellisesti siinä oli kaksi tärkeää saraketta: id sekä parent_id. Id kertoo kyseisen alkion numeron, jolla kyseinen alkio identifioitiin, parent_id kertoo, mikä on tämän kyseisen alkion vanhemman id. Näin pystyttiin tekemään taulu, jossa oli puumainen rakenne ja kaikki alkiot olivat selkeässä järjestyksessä ja luokiteltavissa halutulla tavalla. Kuviossa 18 on pyritty havainnollistamaan parent_id:n suhdetta id:en, Tapahtuman talous –alkion id on 25 ja kaikki sen lapsialkiot saavat parent_id:ksi 25. Kuvio 18 kuvaa kuvion 17 valikkoa tietokannan rakenteen osalta.



Parent_id	Nimi	Id
4	Sisällön suunnittelu	20
4	Markkinoinnin suunnittelu	21
4	Elämyksellisyys	22
4	Laatu osana tapahtumaa	23
4	Viranomaisluvat	24
4	Tapahtuman talous	25

parent_id	Nimi	id
25	Budjetointi	33
25	Tapahtumabudjetin luominen	34
25	Kokonais- ja osabudjetit	35
25	Kustannukset	36
25	Budjetoinnin hyödyt ja haitat	37
25	Lahjoitukset	38
25	Sponsorointi	39
25	Avustukset	40

KUVIO 18. Tietokanta-esimerkki

Muut sarakkeet taulussa sisälsivät alkion nimen, järjestysnumeron sekä alkion tekstisisällön. Tämä taulu oli paikka, jonne kaikki sivut tallennettiin, joten käyttäjän katsellessa valikkoa hän itse asiassa näki tietokannan alkioiden nimi-kenttien sisällöt. Valitessaan jonkun valikon osista, sivusto latasi sisältönsä kyseisen alkion tekstikentästä.

Alkioille tuli myös eräänlainen järjestysnumero, jolla samalla tasolla olevia alkioita pystyttiin järjestelemään muunkin kuin nimen tai id:n mukaan. Tekstikentän sisällöksi tallennettiin kaikki se, mitä kyseisellä sivulla haluttiin esitettävän eli teksti kuvineen. Kuvista tekstikenttään tallennettiin pelkkä osoite, ja itse kuva oli palvelimen hakemistossa.

6.2.2 Valikon toteuttaminen

Eräs haaste valikon sisäisessä rakenteessa oli dynaamisuus. Valikko haki sisältönsä MySQL-tietokannasta, johon kaikki sisältö oli tallennettu. Tämä toimenpide oli tehtävä joka kerta, kun sivustolle tultiin. Myös eräänä haasteena valikon toteuttamisessa oli Ajax-tekniikan käyttäminen. Haluttiin sisällön vaihtuvan sivulla ilman sivulatauksia, joten selaamisesta tulisi miellyttävämpää ja nopeampaa. Ajax-tekniikkaa käytettiin myös päävalikossa, joten ylimääräisiä sivulatauksia ei ollut tarpeen tehdä koko sivustolla.

```
function stateChanged()
{

if (xmlhttp.readyState==4)
    {

        if (xmlhttp.status==200){

        document.getElementById("teksti").innerHTML = xmlhttp.responseText;
        }
    }
}
```

KUVIO 19. Tekstin päivittäminen sivulle

Valikko generoitiin PHP-skriptillä, joka tulosti valikon rakenteen HTML-muotiseen listaan. Listan alkiot sisälsivät linkin halutulle sivulle sekä JavaScript-funktion, jolla sisältöä sivulle haettiin. Funktio välitti eteenpäin kyseisen linkin eli tietokannan alkion id:n, jonka perusteella erillinen PHP-skripti haki tiedot tietokannasta. Tämän jälkeen toinen JavaScript-funktio päivitti sivun sisällön innerHTML- sekä responseText-ominaisuuksia käyttäen, kuten kuviosta 19 saadaan selville. Tässä palvelimen vastaus syötetään teksti-nimisen elementin sisään.

Tekstin liittäminen sivulle ei vaatinut mitään muotoilutoimenpiteitä, koska se oli tietokannassa valmiiksi HTML-muodossa ja muotoilut, kuten sisennykset, oli tehty HTML-koodina.

6.2.3 Tietokantayhteydet

Tietokantaan yhteyden muodosti PHP-skriptit, joita tarvittiin muutamia itse pääsivustolla ja useampia hallintasivustolla. Pääsivustolla PHP:tä käytettiin osana Ajax-toteutusta; kun JavaScript kaipasi sisältöä sivulle, se kutsui palvelimelta PHP-skriptiä, joka otti yhteyden MySQL-tietokantaan ja lopulta palautti tiedot JavaScriptin esitettäviksi. Kuviossa 20 on kuvattuna yksi skripteistä, jolla haetaan sisältöä tietokannasta.

```php
<?php
$id=$_GET["q"];

$con = mysql_connect("[redacted]");
mysql_select_db("iceevent");

        $res = mysql_query("SELECT teksti FROM yksi WHERE id = '$id'");

         if (!$res) {
                    echo 'Could not run query: ' . mysql_error();
                    exit;
                }
$row = mysql_fetch_row($res);
echo $row['0'];
mysql_close($con);

?>
```

KUVIO 20. Tekstin haku tietokannasta

PHP:n ominaisuuksista löytyi varsin kattava lista MySQL-funktioita, joten tietokantaan yhdistäminen osoittautui helpoksi. Tietokannan fyysinen sijainti oli koulun pikkumyy-palvelimella, jossa myös itse sivusto sijaitsi. Tämä helpotti tietokannan käyttöä, koska yhteydet olivat paikallisia.

6.3 Ylläpito

6.3.1 Hallintasivusto

Pääsivuston hallintaa varten tuli kehittää yksinkertainen hallintasivusto, jonka avulla ohjelmointitaidoton ylläpitäjä pystyy tekemään ylläpitotoimia. Vain sivuston ylläpitäjillä oli pääsy sinne, joten ulkoasua ei tarvinnut painottaa vaatimuksissa. Hallintasivustolle ei tehty erillisiä linkkejä pääsivun yhteyteen, vaan sinne pääsi vain tietyn osoitteen kautta ja salasanaa käyttämällä.

Hallintasivustolta tuli pystyä lisäämään materiaalia sivustolle, muokkaamaan sitä sekä poistamaan sivuja. Nämä toteutettiin pääosin PHP-pohjaisella ratkaisulla ja apuna käytettiin hieman JavaScriptiä. Pääsivulla oli kolme osiota, joiden sisältö oli sama, toiminta eri. Ensimmäisessä osiossa käyttäjälle avautui kuvion 21 mukainen lista kaikista sivuston sivuista, jotka löytyivät tietokannasta, ja sivun linkkiä painamalla pääsi muokkaamaan kyseisen sivun sisältöä editorissa, minkä jälkeen se tallennettiin takaisin tietokantaan.

Nämä listat generoitiin PHP-skriptillä aina, kun sivulle saavuttiin, joten mahdolliset muutokset näkyivät käyttäjälle välittömästi. Linkkien sisennykset auttoivat havainnollistamaan tasoja, jotta muutokset tulisi tehtyä oikeaan paikkaan.

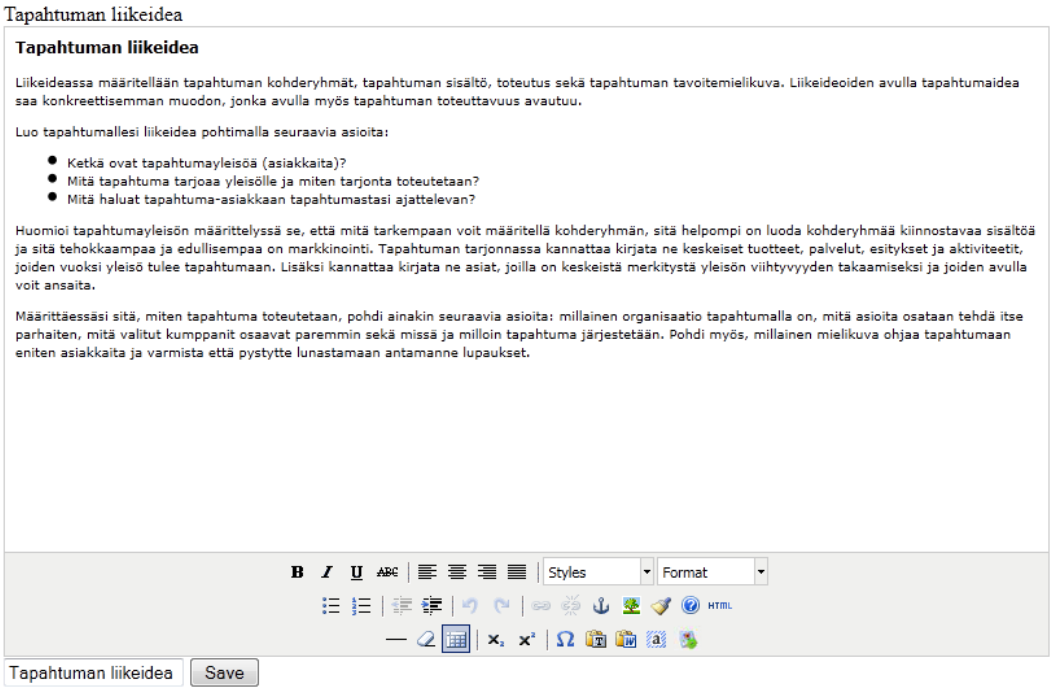
KUVIO 21. Hallintasivusto

Toinen osio oli samanlainen lista, mutta tällä kertaa käyttäjä pystyi luomaan uuden sivun kyseisen sivun kanssa samalle tasolle tai vaihtoehtoisesti alitasolle. Tämän jälkeen sivu tallennettiin tietokantaan ja seuraavalla kerralla, kun sivusto ladattiin, uusi sivu oli näkyvissä. Kolmas vaihtoehto oli sivun poistaminen samankaltaiselta listalta.

6.3.2 TinyMCE-editori

Hallintasivustolla olevien uuden sivun luonti sekä vanhan muokkaus vaativat editorin käyttämistä. Tähän valittiin TinyMCE, joka ominaisuuksiltaan sopi tähän

hyvin. Kyseessä on verkkopohjainen tekstinkäsittelyohjelma, jolla erilaiset muotoilut, kappalejaot, sisennykset ja muut vastaavat saatiin tehtyä hyvinkin helposti, kuten kuviosta 22 nähdään. Perinteisenä vaihtoehtona olisi ollut HTML-tekstikentän käyttäminen, mutta silloin mitään muotoiluja ei olisi ollut mahdollista käyttää, ja pitkien tekstien ulkoasu sekä luettavuus olisi ollut kehno. Hyvänä puolena tässä editorissa oli sen kyky tuottaa kaikki teksti suoraan HTML-muodossa, joten tietokantaan voitiin suoraan tallentaa HTML-sisältöä ilman muunnoksia kooditasolla. Tämä helpotti myös sisällön hakemista tietokannasta, koska se oli valmiiksi halutussa muodossa sivulle liittämistä varten.

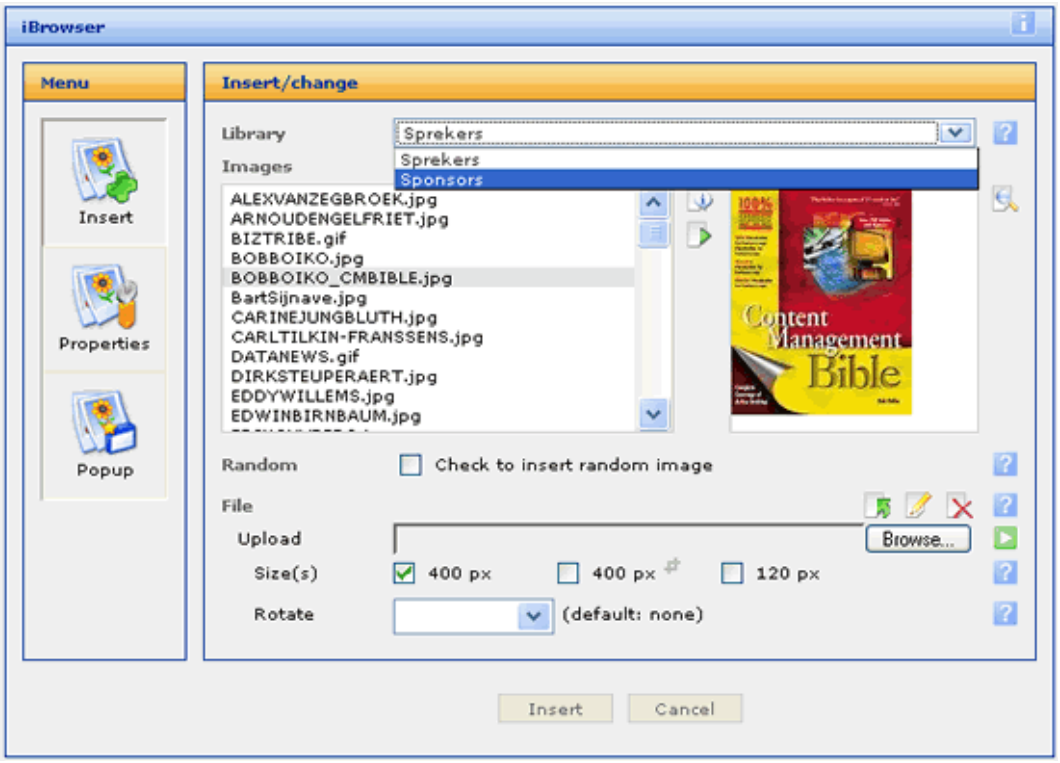


KUVIO 22. TinyMCE

Eräs ongelma havaittiin käyttöönoton jälkeen, käyttäjien liittäessä tekstiä suoraan Microsoft Word –tiedostosta. TinyMCE teki huomattavan määrän ylimääräistä HTML-koodia, joka myöhemmässä vaiheessa paljastui Word-muotoiluiksi, jotka tulivat liitettäessä mukaan, vaikka eivät näkyneetkään ennenkuin editorin näkymän vaihtoi HTML-muotoon. Onneksi editoriin löytyi lisäosa, jolloin se poisti automaattisesti kaikki turhat rivit.

6.3.3 Kuvien lisääminen

Kuvia varten aluksi oli tarkoitus pyhittää sivuston toinen reuna, mutta todettiin sivuston käyvän turhan ahtaaksi tällaisen tekemiseen. Toinen vaihtoehto oli tehdä erillinen kuvagalleria, mutta sekään ei tähän sopinut. Lopulta päädyttiin kuvien lisäämiseen tekstin joukkoon ja tätä varten löytyi TinyMCE-editoriin iBrowser-lisäosa, jolla kuvia pystyi siirtämään palvelimelle ja sitten upottamaan tekstin joukkoon. Kuviossa 23 näkyy iBrowserin päänäkymä, jolta kuvien siirto palvelimelle tapahtuu. Samaa näkymää käytetään myös kuvien valintaan, jolloin iBrowser välittää kuvan TinyMCE:lle.



KUVIO 23. iBrowser

iBrowserin käyttöönotto tuotti ongelmia kansioiden käyttöoikeuksien kanssa, joten aivan ohjetta noudattaen ei voitu edetä. Ongelmaa oli myös kansioiden polkujen syöttämisessä iBrowserin asetuksiin.

6.4 Toimitus

Työ oli määrä pitää Pikkumyy-palvelimella, mutta sen piti olla siirrettävissä muualle. Tätä varten tuli tuottaa cd-levy, jossa kaikki koodit olivat sekä ohjeet siirtoa varten. Ohjeissa otettiin kantaa laitteistovaatimuksiin ja muutoksiin, joita koodiin tarvitsee tehdä siirron yhteydessä. Mukaan tuli myös tietokannasta kopio, mutta se oli kopiointipäivänmäärän mukainen, joten uusi kopio tuli ottaa, jos halusi kaiken sisällön mukaan, mitä sivustolle oli lisätty. Varmuuskopiona sen hetkinen tietokanta toimi jokatapauksessa.

7 YHTEENVETO

Työn tarkoituksena oli valmistaa sivusto Lahden ammattikorkeakoulun matkailun laitoksen Ice Event –projektiin. Sivuston ulkoasusta piti tulla näyttävä, jotta se tasapainottaisi tekstipitoista sisältöä. Sivustosta oli saatava interaktiivinen ja miellyttävä käyttää.

Sivustolla päätettiin käyttää Ajax-tekniikkaa, jotta käytettävyys paranisi verrattuna perinteiseen ratkaisuun. Toisena haasteena oli ohjelmoijan sekä graafikon töiden erottelu, jotta työskentely olisi mahdollista yhtä aikaa, toisiaan häiritsemättä.Ajax-tekniikka toi sivustolle haluttua käytettävyyttä sekä nopeutta, koska ylimääräiset sivulataukset jäivät pois. Sivulatauksia ei tullut, kuin sivun lataaminen sinne siirryttäessä, joten sivulatausten määrän karsinnassa onnistuttiin hyvin.

Graafikon ja ohjelmoijan töiden erottaminen toimi hyvin. Alussa sovitut määreet pitivät, ja muutoksia toisen työhön ei tarvinnut tehdä työskentelyn aikana. Erillinen CSS-tiedosto oli ainoastaan graafikon käytettävissä työn aikana, joten ohjelmoinnissa ei tarvinnut tyyleihin puuttua, kunhan itse sivuston rakenne pysyn muuttumattomana.

Ice Event –sivusto valmistui hyvin ja toimi, kuten sen haluttiin toimivan. Viiveitä sisällön päivittämisessä ei juurikaan syntynyt, kuten olettamus alussa jo olikin, joten tässä onnistuttiin hyvin. Ajax-tekniikka lunasti sille esitetyt odotukset ja oli hyvin käytettävä sivuston rakentamisessa.

Jatkokehityssuunnitelmaan tulisi laittaa hallintosivuston ulkoasun muokkaaminen käyttäjäystävällisempään muotoon. Pelkällä CSS-tiedostolla voisi saada kauniimman hallintasivuston aikaiseksi.

Sivuston siirtäminen sisällönhallintajärjestelmään voisi olla tulevaisuuden kannalta kannattavaa, mikäli aineiston määrä kasvaisi entisestään. Tällöin hallinta olisi tehtävissä työkaluilla, joita käytetään useilla eri sivustoilla, esimerkiksi Yle

käyttää Drupal-sisällönhallintajärjestelmää. Tähän ei kuitenkaan nähty tarvetta tässä vaiheessa, ja asiakkaat olivat tyytyväisiä sivustoon tällaisenaan.

LÄHTEET

Appu A. 2002, Making use of PHP. Wiley Publishing Inc.

Asleson R. & Scutta N. T. 2007, Ajax – Tehokas hallinta. Jyväskylä: Gummerus Kirjapaino Oy

Garrett J. J. 2005, Ajax: A New Approach to Web Applications [Viitattu 29.3.2011]. Saatavissa: http://www.adaptivepath.com/ideas/e000385

Linjama T. 2001, XHTML. Gummerus Kirjapaino Oy

Meloni J. 2003, MySQL Trainer Kit. Edita Publishing Oy

Moxiecode Systems AB. 2011, TinyMCE [Viitattu 29.3.2011]. Saatavissa: http://tinymce.moxiecode.com/wiki.php/TinyMCE

net4visions.com. 2011, iBrowser [Viitattu 30.3.2011]. Saatavissa: http://www.net4visions.com/ibrowser.html

Riordan R. M. 2008, Head First Ajax. O'Reilly Media

Robie J. 1998, What is the Document Object Model? [Viitattu 30.3.2011]. Saatavissa: http://www.w3.org/TR/DOM-Level-1/introduction.html

Shannon R. 2011, Ajax [Viitattu 10.4.2011]. Saatavissa: http://www.yourhtmlsource.com/javascript/ajax.html

Smith D. & Negrino T. 2007, JavaScript – Tehokas hallinta. Jyväskylä: Gummerus Kirjapaino Oy

Velasquez T. 2010, Relational database. [Viitattu 10.4.2011]. Saatavilla: http://tiffanyvelazquez.blogspot.com/2010/11/relational-database.html

w3schools.com 2011, AJAX - The onreadychange Event. [Viitattu 10.4.2011]. Saatavissa: http://www.w3schools.com/ajax/ajax_xmlhttprequest_onreadystatechange.asp

w3schools.com 2011, Browser Statistics. [Viitattu 10.4.2011]. Saatavissa: http://www.w3schools.com/browsers/browsers_stats.asp

w3schools.com 2011, HTML DOM Tutorial. [Viitattu 10.4.2011]. Saatavissa: http://www.w3schools.com/htmldom/default.asp

w3schools.com 2011, The Internet Explorer Browser. [Viitattu 10.4.2011]. Saatavissa: http://www.w3schools.com/browsers/browsers_explorer.asp