Verkkoautomessujen tekninen toteutus

Mikko Kärkelä

TAMPEREEN AMMATTIKORKEAKOULU

Tampere University of Applied Sciences

TIIVISTELMÄ

Tampereen ammattikorkeakoulu

Tietojenkäsittelyn koulutusohjelma

Digimedia

KÄRKELÄ MIKKO: Verkkoautomessujen tekninen toteutus

Huhtikuu 2011

Opinnäytetyö 54 s.

Opinnäytetyö käsittelee verkkoautomessujen teknistä toteutusta. Verkkoautomessut olivat Alma Mediapartners Oy:n syksyllä 2010 toteuttama verkkotapahtuma. Verkkoautomessut koostuivat sivustosta, jolla kävijät pääsivät katsomaan erilaisia autoiluun liittyviä videoita, kuten uusien autojen esittelyitä. Olin Alma Mediapartnersilla työharjoittelussa kesällä 2010, ja tehtäviini kuului messujen verkkosovelluksen ohjelmointi suunnitelmien pohjalta.

Opinnäytetyön tavoitteena oli toteuttaa verkkoautomessuille sivusto olemassa olevien suunnitelmien pohjalta ja dokumentoida siinä käytetyt menetelmät ja ratkaisut. Tarkoituksena oli luoda toimiva sivusto, jolla käyttäjä voi helposti tutustua messujen tarjoamaan sisältöön. Keskeisimmät toteutettavat ominaisuudet olivat eri lähteistä tulevien videoiden katselun mahdollistava kohdesivu, navigaatio- ja sisällön arviointijärjestelmät sekä jakomahdollisuus sosiaalisiin medioihin. Lisäksi sovellus oli toteutettava ilman tietokantoja, joten sisältödatan säilömiseen oli käytettävä XML-tiedostoja, mikä toi toteutukseen omat haasteensa. Tähän dokumenttiin kirjattiin sovelluksen keskeisimpien osien toteutusmenetelmät ja tekniikat. Lisäksi perehdyin hieman videosisältöjen näyttämiseen verkkosivuilla yleisesti sekä HTML 5:n video-ominaisuuksiin.

Sovelluksen toteutuksen jälkianalysoinnin perusteella valitut tekniikat ja toteutusmenetelmät olivat kyseisen projektin tarkoitukseen ja skaalaan sopivat. Toteutusmenetelmät olivat yleisesti toimivia, ajatuksella suunniteltuja ja johdonmukaisia. Sovelluksessa ei ilmennyt esimerkiksi yhteensopivuusongelmia eri selainten välillä. Ongelmakohdaksi muodostui mobiililaiteyhteensopivuus, koska sovellusta ei suunniteltu niillä toimivaksi. Käytetyt tekniikat, kuten PHP, JavaScript, Flash ja XML, olivat toteutukseen riittäviä ja perustellusti valittuja. Tekniikat myös tukivat toisiaan hyvin.

Avainsanat: verkkomessut, web-video, XML.

ABSTRACT

Tampere University of Applied Sciences

Degree Programme in Business Information Systems

Digital Media

KÄRKELÄ, MIKKO: Technical implementation of an Online Car Show

Bachelor's thesis 54 pages

April 2011

This bachelor's thesis covers the technical implementation of a web site for an online car show, called "verkkoautomessut". Verkkoautomessut was held in fall of 2010 by Alma Mediapartners. The main content of the car show was comprised of various car-related videos, such as introductions of new cars. I was working at Alma Mediapartners during my internship in the summer of 2010 and programming the web application for verkkoautomessut was one of my tasks.

The aim of this bachelor's thesis was to implement the web application according to existing designs and to document the technologies and solutions used. The plan was to create a functional web site that would allow the users to easily access the content available on the site. Some of the key components that had to be implemented were the target page, which would enable users to watch videos from different sources, the page navigation system, the content rating system and functionality that would allow sharing of content on social media services. The application had to be implemented without using a database. XML-files were used to store data instead, which brought about some challenges. This document was written to describe and analyze the methods used in implementing these components and the technologies behind them. In addition, I familiarized myself with various ways of displaying video content on the web and the video features of HTML 5.

The analysis of the implementation methods was done after the application was finished. The methods and technologies used were found to be suitable for the scope and purpose of the project. The implementation was well-thought through and functional. There were no serious problems such as web browser incompatibilities. The largest problem was the application's incompatibility with some mobile devices, although it was never designed for them. The technologies used, such as PHP, JavaScript, Flash and XML, were sufficient and their use was justified.

Keywords: online car show, web-video, XML.

SISÄLLYS

1 JOHDANTO

Tämä opinnäytetyö käsittelee Alma Mediapartners oy:n syksyllä 2010 toteuttamien verkkoautomessujen WWW-sivujen kehitystyössä käytettyjä tekniikoita ja ratkaisuja. Tämän dokumentin tavoite on luoda yleiskatsaus sovelluksen keskeisimpien osien toimintaan ja toteutukseen. Opinnäytetyöni koostui tämän dokumentin kirjoittamisen lisäksi varsinaisen sovelluksen toteuttamisesta.

Verkkoautomessut pidettiin 11.10. - 21.11.2010 osoitteessa www.verkkoautomessut.fi. Sivuston pääasiallinen sisältö koostui erilaisista autoiluun liittyvistä videoista, kuten uusien autojen esittelyistä ja huoltovinkeistä. Sen pääsponsoreina toimivat Iltalehti, Autotalli.com ja Tuulilasi. Suuri osa sivuston videosisällöstä tuli Iltalehden netti-tv:n kautta, mutta sisältöpaikkoja myös myytiin niitä haluaville yrityksille. Toisena tulonlähteenä toimivat sivuston mainospaikat. Käyttäjiä houkuteltiin sivustolle sekä informatiivisella että viihteellisellä sisällöllä.

Alma Mediapartners oy on osa Alma Mediaa oleva yritys, joka kehittää ja ylläpitää erilaisia verkkopalveluita. Yksi näistä palveluista on autoliikkeitä internetissä palveleva Autotalli.com, jonka kautta käyttäjät voivat etsiä autoliikkeiden tarjoamia autoja ympäri Suomen. Verkkoautomessut olivat siten luonnollinen osa yrityksen toimintaa. Olin Alma Mediapartnersilla 20 viikon työharjoittelussa vuoden 2010 huhtikuusta syyskuuhun ja määräaikaisena tuotesuunnittelijana harjoittelun jälkeen joulukuuhun saakka. Verkkoautomessujen www-sovelluksen kehittäminen oli osa tehtäviäni työharjoittelijana. Lisäkseni sovelluksen kehittämiseen osallistui toinen harjoittelija.

Tämän opinnäytetyön tavoite oli kehittää verkkoautomessujen WWW-sovellus ja kirjoittaa siitä tekninen raportti. Sovellukselta vaadittu keskeisin ominaisuus oli eri lähteistä tulevien videoiden katselun mahdollistava kohdesivu, jolla käyttäjä voisi myös antaa palautetta ja arvosanansa sisällöstä sekä jakaa messujen sisältöä sosiaalisiin medioihin. Lisäksi sovellukseen oli toteutettava sisältöön helpon käsiksi pääsyn takaava navigaatiojärjestelmä, joka koostuu navigaatiovalikosta sekä erillisestä Flash-sovelluksena toteutetusta videoselaimesta. Sovelluksen toteuttamista varten ei ollut käytettävissä tieto-

kantoja, joten sivujen generointiin tarvittavien tietojen tallentamiseen käytettiin XML-tiedostoja. Vaikka XML:ää ei ole varsinaisesti tarkoitettu tiedon säilömiseen, tässä sovelluksessa se tarjosi kätevän vaihtoehdon tietokannoille.

Tämän raportin tarkoitus on avata sovelluksen rakennetta ja sen toimintaa sekä kuvata toteutuksessa käytetyt ratkaisut ja tekniikat. Tämä dokumentti toimii eräänlaisena teknisenä loppuraporttina sovelluksen kehitysprojektille. Varsinaiseen kehitysprosessiin ei kuitenkaan keskitytä, vaan sovelluksen toteutusta kuvataan ainoastaan teknisestä näkökulmasta. Tässä dokumentissa ei myöskään keskitytä sovelluksen käytettävyyteen tai visuaaliseen ilmeeseen liittyviin seikkoihin, elleivät tekniset ratkaisut suoranaisesti liity niihin. Suurin hyöty toimeksiantajalle oli luonnollisesti valmiista sovelluksesta.

Lähteitä käytin lähinnä käytettyjen tekniikkojen esittelyissä. Toteutusmenetelmien analysoinnin tein käymällä kirjoittamaani koodia läpi ajatuksen kanssa miettien, olisiko kyseisen ominaisuuden voinut toteuttaa toisin. En käyttänyt tässä analysoinnissa lähteitä, koska en kiinnittänyt huomiota koodin laatuun, vaan ainoastaan toteutuksen toimivuuteen. Suurin osa lähteistä on verkosta, koska ajantasaista kirjallisuutta on helpompi löytää sieltä ja joissain tapauksissa vain sieltä. Osittain olen käyttänyt Wikipediaa, minkä tiedän olevan hieman kyseenalaista, mutta tässä tapauksessa uskon tiedon olevan paikkansapitävää, sillä käytetty tieto on hyvin yksiselitteistä ja faktaperäistä, eikä riipu kirjoittajan näkökulmista tai mielipiteistä.

Opinnäytteen tuloksena tuotettu varsinainen sovellus saavutti sille asetetut tavoitteet hyvin. Kaikki vaaditut ominaisuudet onnistuttiin toteuttamaan määräajan puitteissa, eikä sovellukseen jäänyt pahempia ongelmakohtia. Toteutetun sovelluksen laajuus pakotti rajaamaan raportissa käsiteltyjä ominaisuuksia. Pyrin käsittelemään ne ominaisuudet, jotka ovat sovelluksen toiminnan kannalta oleellisimpia ja monimutkaisempia. Käsitellyistä tekniikoista jätin suosiolla pois tunnetuimmat, kuten HTML ja CSS. Jätin myös käsittelemättä sovelluksen huutokauppasivun, koska se oli lähes staattista HTML:ää, sekä mainospaikat, koska niiden toteuttaminen vaati omalta kannaltani vain tietyn HTML-koodin pätkän kopioimisen sovellukseen haluttuihin paikkoihin.

2 KÄYTETYT TEKNIIKAT

2.1 PHP

PHP: Hypertext Preprocessor on avoimen lähdekoodin skriptikieli, joka soveltuu erityisesti WWW-kehitykseen. Sillä voidaan kirjoittaa joko itsenäisiä skriptejä tai sitä voidaan upottaa HTML-koodin sekaan. PHP:n ajaminen tapahtuu palvelimella, jolle asennettu tulkki ajaa skriptitiedoston, jonka jälkeen tulokset, kuten generoitu HTML-koodi, lähetetään käyttäjälle. Käyttäjä ei näe selaimellaan varsinaista skriptiä, joka luo käyttäjän haluaman sivun, vaan vain skriptin ajamisen tulokset. (The PHP Group 2011a.) Koodiesimerkissä 1 esitellään yksinkertainen HTML-koodin sekaan kirjoitettu PHP-skripti.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/HTML4/loose.dtd">
<HTML>
    <head>
        <title>Example</title>
    </head>
    <body>

        <?php
            echo "Hi, I'm a PHP script!";
        ?>

    </body>
</HTML>
```

KOODIESIMERKKI 1. Yksinkertainen upotettu PHP-skripti

PHP-koodi aloitetaan merkinnällä <?php ja lopetetaan ?>. PHP-tulkki käsittelee näiden merkintöjen väliin sijoitetun koodin, ja palauttaa käyttäjälle sivun, jossa skripti on korvattu sen tuloksella. PHP:tä voidaan siis käyttää sekä monipuolisten dynaamisten verkkosovellusten toteuttamiseen että yksinkertaisiin HTML-koodin sekaan sijoitettuihin dynaamisuutta vaativiin toimenpiteisiin. Yksinkertaisiin toteutuksiin riittää proseduraalinen ohjelmointi, mutta laajemmissa tapauksissa voidaan käyttää olio-ohjelmointia.

PHP voidaan asentaa yleisempiin käyttöjärjestelmiin, kuten Windowsiin, OS X:ään, Linuxiin, ja erilaisiin UNIX-järjestelmiin. PHP tukee myös useimpia WWW-palvelimia,

kuten Apachea ja IIS:ää. Yksi PHP:n vahvuuksista on tietokantojen käsittely. PHP tukee monia erilaisia tietokantoja, ja tietokantapohjaisten verkkosovellusten rakentaminen on PHP:llä yksinkertaista. PHP:ssä on myös paljon hyödyllisiä tekstin ja XML-tiedostojen käsittelyyn liittyviä toimintoja. Tekstin, kuten HTML-koodin tai XML-tiedostojen, tulostamisen lisäksi PHP:llä voidaan esimerkiksi käsitellä ja tulostaa kuva- ja pdf-tiedostoja. Kaikki PHP:llä generoitu sisältö voidaan tallentaa palvelimelle tiedostoihin myöhempää käyttöä varten. (The PHP Group 2011b.)

PHP valittiin verkkoautomessujen toteutukseen, koska se oli ainoa palvelinpuolen kieli, josta molemmilla teknisestä toteutuksista vastaavilla henkilöillä oli riittävästi kokemusta. Uuden kielen opettelemiseen ei nähty järkevää syytä, eikä projektin toteuttamisesta jollain toisella kielellä olisi ollut näkyvää hyötyä. PHP:n ominaisuudet olivat kyseiseen toteutukseen varsin riittävät, joten sen valinta oli hyvin luonnollista ja perusteltua.

2.2 XML ja DOM

XML eli eXtensible Markup Language on merkintäkieli, jonka avulla voidaan kuvata tietoa sen seassa sekä jäsentää laajoja tietomassoja. Sen avulla voidaan määrittää tiedon rakenne ja siten esimerkiksi helpottaa tiedonsiirtoa erilaisten järjestelmien välillä. XML on täysin riippumaton alustasta, sillä luotuja tietorakenteita voidaan käsitellä millä tahansa ohjelmointikielellä, käyttöjärjestelmällä, laitteistolla jne. XML muistuttaa HTML:ää, mutta niiden käyttötarkoitukset poikkeavat toisistaan. HTML on kehitetty tiedon esittämiseen selaimessa, kun taas XML on kehitetty tiedon kuvaamista varten. XML-kielellä kuvatun tiedon käsittelyyn on helppo kehittää erilaisia sovelluksia.

XML-dokumentti aloitetaan nk. prologilla, joka määrittää dokumentissa käytetyn XML-version ja mahdollisia muita tietoja, kuten tiedoston merkkikoodaus. Jokaisessa XML-dokumentissa on oltava tämä rivi. Prologin muoto poikkeaa hieman dokumentin muiden elementtien rakenteesta. Koodiesimerkissä 2 esitellään yksinkertainen XML-prologi, joka määrittää XML-version ja merkistön koodauksen.

```
<?XML version=”1.0” encoding=”utf-8”?>
```

KOODIESIMERKKI 2. Yksinkertainen XML-prologi

Kuten HTML, XML-dokumentti koostuu sisäkkäisistä tekstimuotoisista elementeistä. Kaikki XML-dokumenttiin sijoitettu tieto tulee olla jonkin elementin sisällä. Dokumentilla tulee olla prologin lisäksi vain yksi juurielementti, jonka sisälle kaikki muut elementit sijoitetaan. XML-elementti aloitetaan vertailumerkkien sisään kirjoitetulla elementin nimellä ja lopetetaan samanlaisella merkinnällä, jossa elementin nimen edessä on kauttaviiva. Elementin aloittavia ja lopettavia merkintöjä kutsutaan myös tageiksi. Jokaisella elementillä on oltava sekä aloittava että lopettava tagi. Jos elementin aloittava tagi on jonkin toisen elementin sisällä, on myös lopettavan tagin oltava saman elementin sisällä. Elementeillä voi olla lisäksi attribuutteja, jotka antavat lisätietoa yksittäisestä elementin ilmentymästä. Attribuutti merkitään aloitustagin sisään. Attribuutilla on nimi ja arvo, joka merkitään nimen perään lainausmerkkien sisään yhtäsuuruusmerkillä nimestä erotettuna. Isot ja pienet kirjaimet katsotaan tageissa ja attribuuteissa eri merkeiksi. Kun XML-dokumentti seuraa näitä syntaksisääntöjä, sanotaan sen olevan hyvin muodostettu. Jos XML-dokumentti ei ole hyvin muodostettu, saattaa sen käsitteleminen erilaisissa sovelluksissa johtaa ongelmiin. (Kahate 2009.) Koodiesimerkki 3 esittelee yksinkertaisen XML-dokumentin.

```
<?XML version=”1.0” encoding=”utf-8” ?>
<autot>
      <auto merkki=”toyota”>
            <vuosimalli>2000</vuosimalli>
            <mittarilukema>100000</mittarilukema>
      </auto>
      <auto merkki=”volvo”>
            <vuosimalli>2005</vuosimalli>
            <mittarilukema>50000</mittarilukema>
      </auto>
</autot>
```

KOODIESIMERKKI 3. Yksinkertainen XML-dokumentti

XML-dokumentille voidaan määritellä erilaisia rakennevaatimuksia, kuten elementtien ja attribuuttien ilmenemispaikat ja -määrät. Tähän voidaan käyttää esimerkiksi Document Type Definition- eli DTD-tiedostoja. Verkkoautomessujen tapauksessa tällaisiin määrittelyihin ei kuitenkaan nähty tarvetta, sillä tiedostot olivat aina samat, niitä oli vähän ja ne muuttuivat harvoin.

XML-muotoisen tiedon käsittelyyn on kehitetty menetelmä nimeltä Document Object Model eli DOM. Se on alustasta ja ohjelmointikielestä riippumaton World Wide Web Consortiumin (W3C) ylläpitämä standardi. Käytännössä XML-muotoinen tieto ladataan ohjelmakoodissa olioksi, jonka jälkeen sen rakennetta ja siihen tallennettuja tietoja voidaan muokata DOM:ssa määriteltyjen metodien avulla.

XML-muotoinen tieto katsotaan DOM:ssa puumaiseksi rakenteeksi, joka koostuu ns. solmuista. Solmuja ovat kaikki XML-dokumentin osaset, kuten elementit, attribuutit ja elementtien sisältämät tiedot. Jokaisella solmulla, paitsi dokumentin juurielementillä, voi olla ns. vanhempia, sisaruksia tai lapsia eli dokumentin osia, jotka sijaitsevat puurakenteessa kyseisen solmun yläpuolella, samalla tasolla tai alapuolella. Dokumenttia käsitellään ja navigoidaan solmujen avulla. Ohjelmoitaessa jokaista solmua käsitellään oliona. Solmuolioilla on erilaisia metodeja ja ominaisuuksia. Tärkeimmät ominaisuudet ovat nodeName, nodeValue ja nodeType. NodeName-ominaisuuden arvo on solmun nimi, joka on elementtien tapauksessa tagin nimi. NodeValue sisältää tekstisolmun arvon ja nodeType kertoo solmun tyypin, eli onko kyseessä elementtisolmu, tekstisolmu jne. DOM:ssa on tärkeää huomata, että elementtisolmu ei sisällä tietoa, vaan elementtisolmulla on lapsenaan tekstisolmu, jonka nodeValue-ominaisuutena on kyseisen elementin sisältämä tieto. (W3Schools. 2011a.)

Myös selaimet käsittelevät verkkosivujen HTML:ää DOM:in avulla. HTML DOM eroaa XML DOM:sta jonkin verran, mutta sen toimintaperiaatteet ovat samat. Poikkeuksia XML DOM:sta ovat esimerkiksi metodi getElementsById ja elementtisolmujen innerHTML-ominaisuus. GetElementsById-metodin avulla voidaan hakea elementtejä HTML-dokumentista niiden id-attribuuttien perusteella. InnerHTML-ominaisuuden avulla päästään käsiksi kyseisen elementtisolmun HTML-sisältöön ilman tekstisolmua. InnerHTML-ominaisuus ei ole osa W3C:n DOM-standardia, mutta useimmat selaimet tukevat sen käyttöä. Elementtien sisältöä voidaan kuitenkin käsitellä myös tekstisolmujen kautta XML DOM:n tapaan. (W3Schools. 2011b.)

2.3 JavaScript, jQuery ja Ajax

JavaScript on pääasiassa web-ympäristöön kehitetty skriptikieli. Sen avulla voidaan lisätä verkkosivuihin dynaamisia toimintoja, joiden toteuttaminen ainoastaan HTML:n ja css:n avulla ei ole mahdollista. JavaScript-koodit ajetaan käyttäjän tietokoneella, eli toisin kuin PHP, JavaScript on ns. asiakaspuolen skriptikieli.

JavaScript ilmestyi ensimmäisen kerran Netscape Navigator 2.0 -selaimen mukana syyskuussa 1995 nimellä LiveScript. Nimi muutettiin JavaScriptiksi Netscape Navigatorin Java-tuen lisäämisen yhteydessä. JavaScriptillä ja Javalla on kuitenkin hyvin vähän yhteistä lukuun ottamatta joitain nimeämiskäytäntöjä. JavaScript perustuu nykyisin ECMAScript-standardiin, joka kehitettiin JavaScriptin pohjalta. (Wikipedia. 2011a.)

JavaScriptin verkkosivustojen toteuttamisen kannalta hyödyllisimpiin ominaisuuksiin kuuluu mm. sivujen muokkaaminen käyttäjän selaimessa sivun avaamisen jälkeen. Sillä voidaan myös toteuttaa erilaisia hiiren osoittimen sijaintiin sivulla, painikkeen painalluksiin tai muihin käyttäjän toimiin perustuvia toimintoja. Käyttäjän lomakkeisiin syöttämiä tietoja voidaan tarkastella ennen niiden lähettämistä ja näyttää käyttäjälle ilmoitus virheellisistä syötteistä. JavaScriptissä on myös toiminnallisuuksia, joilla voidaan saada tietoa käyttäjästä. Voidaan esimerkiksi selvittää, millä selaimella sivu on avattu, tai onko selaimeen asennettu jokin tietty liitännäinen. JavaScriptillä ei voida käytännössä lukea tai kirjoittaa käyttäjän kiintolevyllä sijaitsevia tiedostoja. Poikkeuksena tähän sääntöön ovat selaimen evästeet, joita voidaan sekä lukea että kirjoittaa. JavaScriptillä ei myöskään voida lukea tietoja verkkolähteistä, jotka sijaitsevat eri domainissa, kuin skriptin sisältävä sivusto. (Keogh 2005, 3–4.)

JavaScriptin toiminta perustuu suurelta osin ns. tapahtumiin. Selain seuraa käyttäjän toimintaa, esimerkiksi hiiren osoittimen sijaintia sivulla, ja laukaisee tapahtuman tiettyjen ehtojen täyttyessä. Kun tapahtuma laukeaa, ajetaan kyseiseen tapahtumaan liitetty JavaScript-koodi, joka on yleensä funktionkutsu. Tapahtumien seuraaminen toteutetaan lisäämällä HTML-koodin elementteihin tiettyjä attribuutteja. Esimerkiksi lisäämällä elementille attribuutti onMouseOver, voidaan attribuutin arvona oleva tietty pätkä JavaScript-koodia suorittaa hiiren osoittimen ollessa kyseisen elementin päällä.

(W3Schools 2011c.) Koodiesimerkki 4 avaa ilmoitusikkunan tekstillä ”Kursori on elementin päällä!”, kun osoitin tuodaan kyseisen div-elementin päälle.

```
<div onMouseOver=”alert('Kursori on elementin päällä!');”></div>
```

KOODIESIMERKKI 4. JavaScript-tapahtumien kuunteleminen

JavaScript-koodia voidaan sisällyttää verkkosivuille kahdella tavalla: joko suoraan HTML-koodin sekaan script-elementtien sisälle tai ulkoisena tiedostona script-elementin src-attribuutin avulla. Script-elementit voidaan sijoittaa joko head- tai body-elementin sisälle. (W3Schools 2011d.) Skriptit ajetaan latausjärjestyksessä. Jos jonkin latauksen yhteydessä ajettavan skriptin halutaan vaikuttavan tiettyihin elementteihin sivun HTML-koodissa, on se sijoitettava kyseisten elementtien jälkeen. (Chapman. 2011) Koodiesimerkit 5 ja 6 tulostavat selaimeen tekstin ”Hello world!”. Ensimmäisessä esimerkissä koodi on kirjoitettu suoraan HTML-koodin sekaan ja toisessa ulkoiseen tiedostoon. Script-elementin type-attribuutti kertoo selaimelle, millä kielellä kirjoitettu skripti on kyseessä.

```
<script type=”text/javascript”>
document.write(”Hello world!”);
</script>
```

KOODIESIMERKKI 5. JavaScript HTML-koodin sisällä

```
<script type=”text/javascript” src=”skripti.js”></script>
```

KOODIESIMERKKI 6. JavaScript ulkoisessa tiedostossa

JavaScriptin kirjoittamista voidaan helpottaa käyttämällä erilaisia kirjastoja. Yksi suosituimmista kirjastoista on jQuery. Sen tarkoitus on helpottaa HTML-dokumentin DOM:in elementtien hakemista ja muokkaamista sekä erilaisten visuaalisten efektien toteuttamista. JQueryn avulla on mahdollista toteuttaa verkkosivuille melko näyttäviäkin efektejä ilman Flashia, kuten esimerkiksi erilaisia liukuvia pudotusvalikoita tai häive-efektejä. Kirjastosta löytyy myös erilaisia Ajaxin käyttöön ja CSS-määritysten muokkaamiseen liittyviä toimintoja. JQuery on yhteensopiva useimpien selainten kanssa ja sen toimintoja voi laajentaa liitännäisten avulla. Myös JavaScript-tapahtumien käsittely on helpompaa jQueryn avulla. (Wikipedia. 2011b.)

JQueryä käytetään nk. tehdasmetodi $:n kautta. Tämä metodi palauttaa jQuery-olion, kun sitä kutsutaan parametrinään CSS-valitsin. HTML-dokumentin DOM:iin voidaan tämän jälkeen tehdä muutoksia käyttäen olion metodeita. Ne vaikuttavat kaikkiin niihin HTML-dokumentin elementteihin, jotka vastaavat tehdasmetodille annettua CSS-valitsinta. (Wikipedia 2011b.) Koodiesimerkki 7 hakee HTML-dokumentista sen latauduttua kokonaan kaikki div-elementit, joiden class-attribuuttina on ”piilota”, ja piilottaa ne näkyvistä jQuery-olion hide-metodin avulla.

```
$(document).ready(function(){
      $(”div.piilota”).hide();
});
```

KOODIESIMERKKI 7. Yksinkertainen jQuery-koodi

Usein dynaamisten verkkosivujen toteuttamisessa on tärkeää, että sivua ei tarvitse ladata palvelimelta uudelleen aina kun sivun sisältöä halutaan muuttaa. Tähän tarkoitukseen on kehitetty käytäntö nimeltä Ajax. Nimitys on lyhenne termistä Asynchronous JavaScript and XML. Ajax ei ole varsinainen ohjelmointikieli tai muu tekniikka, vaan menetelmä, jossa eri tekniikoita käytetään yhdessä tiedon lataamiseen palvelimelta ja sivun osien muokkaamiseen ilman koko sivun päivittämistä. Tiedon lataaminen palvelimelta tapahtuu tavallisesti JavaScriptin avulla XMLHttpRequest-olion kautta. XMLHttpRequest on ohjelmointirajapinta, jonka avulla voidaan lähettää http- tai https-pyyntöjä palvelimelle ja vastaanottaa palvelimen vastaus, joka voi olla XML-muotoisen tiedon lisäksi tavallinen merkkijono. Palvelimelta saatu tieto esitetään HTML-dokumentissa dynaamisesti muokkaamalla sen DOM:ia JavaScriptin avulla. (Wikipedia 2011c.) Katso esimerkki Ajaxin käytöstä verkkoautomessujen toteutuksessa luvusta 5.3.2.

2.4 Flash ja ActionScript 3.0

Adobe Flash on Adobe Systemsin kehittämä kehitysympäristö, jonka avulla voidaan luoda erilaisia multimediasovelluksia, kuten animaatioita, mainoksia, pelejä ja interaktiivisia verkkosivustoja. Verkkoautomessujen kannalta Flashin tärkein ominaisuus oli mahdollisuus toteuttaa selaimessa toimivia videosoittimia. Selainkäytössä Flash-sovellukset vaativat selaimelta Flash Player -liitännäisen, joka avaa Flashilla tuotetut swf-tiedostot. Flashin käyttämien videoiden tulee olla Flash Video -formaatissa ja ne voivat

olla joko upotettuna soittimen swf-tiedostoon tai ulkoisessa lähteessä. (Wikipedia 2011d.)

ActionScript on pääasiassa Flash-sovellusten kehittämiseen käytetty ohjelmointikieli. Se on JavaScriptin tavoin ECMAScript-standardin implementaatio, joten kielillä on hyvin paljon yhteistä, kuten sama syntaksi ja semantiikka. ActionScriptistä on käytössä kaksi eri versiota Flash-sovellusten toteuttamiseen. Verkkoautomessujen Flash-sovellukset toteutettiin käyttäen ActionScriptin versiota 3.0. Molemmat versiot ovat syntaksiltaan samankaltaisia, mutta ActionScript 3.0 käyttää eri ohjelmointirajapintoja kuin versio 2. ActionScript 3.0 on aito oliopohjainen ohjelmointikieli, kun taas ActionScript 2 on Flashin käyttöön kehitetty skriptikieli. Kielen versio vaikuttaa myös sovelluksen toimivuuteen. ActionScript 3.0:lla kehitetty sovellus vaatii tietokoneelta jonkin verran vähemmän suorituskykyä, kuin versio 2:lla kehitetty. (Wikipedia 2011e.)

3 VIDEOSISÄLLÖN NÄYTTÄMINEN VERKKOSIVULLA

Videot ovat palvelimelta käyttäjän koneelle lähetettyjä tiedostoja siinä missä kaikki muukin Internetistä tuleva sisältö. Videoiden esitystavoissa on kuitenkin jonkin verran vaihtoehtoja. Yksi tapa on ladata tiedosto kokonaisena käyttäjän koneelle ja toistaa se jollain mediasoittimella, kuten Windows Media Playerilla. Toinen keino on avata tiedosto selaimeen asennettavalla liitännäisellä joko erillisellä sivulla/välilehdellä tai suoraan sivustoon upotettuna. (Multimedia Basics... 2011.)

Liitännäiset ovat ohjelmia, jotka lisäävät toiminnallisuuksia johonkin toiseen ohjelmaan. Ne eivät ole yleensä itsenäisiä ohjelmia, vaan toimivat ainoastaan isäntäsovelluksen tarjoamien palvelujen avulla. (Wikipedia 2011f.) Tässä tapauksessa erilaiset liitännäiset lisäävät mahdollisuuden toistaa videotiedostoja suoraan selaimessa sen sijaan, että käyttäjä joutuisi lataamisen jälkeen itse avaamaan videotiedoston jollain erillisellä mediasoittimella. Suosittuja videon toistamisen selaimessa mahdollistavia liitännäisiä ovat esimerkiksi Adoben Flash Player ja Applen Quicktime Player.

3.1 Videoiden upottaminen HTML-koodiin

Videoita voidaan upottaa sivuihin HTML-kielen object-elementin avulla. Object-elementti tuli HTML-kieleen HTML 4:n mukana ja sen tarkoitus oli toimia kaikenkattavana keinona erilaisten objektien, kuten liitännäisten toistamien videoiden, upottamiseen verkkosivuille. Tämä tarkoitus ei kuitenkaan koskaan täysin realisoitunut, koska eri selaimet tukevat sen käyttöä hieman eri tavoin. Näitä yhteensopivuusongelmia voidaan kuitenkin kiertää esimerkiksi sijoittamalla koodia object-elementin sisälle. Mikäli selain ei pysty suorittamaan object-elementin vaatimaa toimenpidettä, suoritetaan sen sijaan elementin sisälle sijoitettu koodi. (W3Schools 2011e.) Koodiesimerkissä 8 upotetaan HTML-sivulle Quicktime-liitännäisellä toistettava video.

```
<object classid="clsid:02BF25D5-8C17-4B23-BC80-D3488ABDDC6B"
codebase="http://www.apple.com/qtactivex/qtplugin.cab" width="320"
height="256">

<param name="src" value="yourMovie.mov" />
<param name="controller" value="true" />
<param name="autoplay" value="true" />
<param name="autostart" value="1" />
<param name="pluginspage"
value="http://www.apple.com/quicktime/download/" />

<!--[if !IE]> <-->

<object type="video/quicktime" data="yourMovie.mov" width="320"
height="256">
<param name="pluginurl"
value="http://www.apple.com/quicktime/download/" />
<param name="controller" value="true" />
<param name="autoplay" value="true" />
<param name="autostart" value="1" />

<embed src="yourMovie.mov" type="video/quicktime" width="320"
height="256" autostart="true" controller="true" ></embed>

</object>

<!--> <![endif]-->

</object>
```

KOODIESIMERKKI 8. Quicktime-videon upottaminen (Fergus D. 2011.)

Object-elementin classid-attribuutti kertoo selaimelle, mitä liitännäistä elementin sisällön näyttämiseen käytetään. Tässä tapauksessa käytetään Quicktimeä. Liitännäiset erotetaan toisistaan esimerkin kaltaisilla merkkijonoilla. Classid-attribuuttia käyttäen toteutettua object-elementtiä käyttää kuitenkin vain Internet Explorer, joten elementin sisälle on sijoitettu HTML-ehtolausetta käyttäen toinen object-elementti, joka toimii muilla selaimilla. Ehtolausetta joudutaan käyttämään, koska Internet Explorer ei ymmärrä sisäkkäisiä object-elementtejä, vaikka muut selaimet ymmärtävät. Tämän elementin type-attribuutti kertoo selaimelle elementin sisällön MIME-tyyppin, jonka perusteella selain valitsee käytettävän liitännäisen. Data-attribuutti osoittaa toistettavan videon sijainnin, mutta IE-toteutuksen tapauksessa tämä tehdään param-elementin avulla. Object-elementin sisälle sijoitettavilla param-elementeillä voidaan antaa object-elementille erilaisia arvoja, jotka vaikuttavat liitännäisen toimintaan. Embed-elementti on jäänne vanhoista selaimista, kuten Netscape Navigatorista, eikä se ole osa HTML-standardeja. (World Wide Web Consortium 2011; Pint Inc 2011)

3.2 Adobe Flash Player

Videon toistamisen mahdollistavista liitännäisistä ylivoimaisesti suosituin on Adobe Systemsin Flash Player. Se on asennettu noin 99 prosenttiin kaikista Internetiin kytketyistä PC-tietokoneista kehittyneimmillä markkina-alueilla, kuten Yhdysvalloissa, Kanadassa ja Iso-Britanniassa. Lisäksi se mahdollistaa videoiden katselun myös muilla laitteilla. (Adobe Systems 2011.) Flashia käyttää esimerkiksi suosittu videopalvelu YouTube. Koodiesimerkki 9 on verkkoautomessujen toteutuksesta.

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
codebase="http://fpdownload.macromedia.com/pub/shockwave/cabs/Flash/sw
Flash.cab#version=9,0,0,0" width="743" height="455">
      <param name="movie" value="player.swf" />
      <param name="allowScriptAccess" value="always" />
      <param name="allowFullScreen" value="true" />
      <param name="wmode" value="transparent" />
      <param name="FlashVars"
      value="videoURL=video.flv&videoThumb=thumb.jpg" />
<!--[if !IE]>-->
<object data="player.swf" type="application/x-shockwave-Flash"
width="743" height="455">
      <param name="pluginurl"
      value="http://www.adobe.com/go/getFlashplayer" />
      <param name="allowScriptAccess" value="always" />
      <param name="allowFullScreen" value="true" />
      <param name="wmode" value="transparent" />
      <param name="FlashVars"
      value="videoURL=video.flv&videoThumb=thumb.jpg" />
</object>
<!--<![endif]-->
</object>
```

KOODIESIMERKKI 9. Flash-sisällön upottamien object-elementillä

Kuten Quicktime-esimerkissä, tässäkin on käytetty kahta sisäkkäistä object-elementtiä. IE-toteutuksessa käytetään classid-attrbuuttia Flash Playeriin viittaavalla arvolla ja muiden selainten toteutuksessa type-attribuuttia Flash-tiedoston MIME-tyypillä. Param-elementeillä vaikutetaan soittimen toimintaan. Tärkein näistä on tässä tapauksessa FlashVars-nimiattribuutin sisältävä elementti. Tämän elementin value-attribuutin avulla voidaan syöttää erilaisia arvoja Flash-soittimelle, kuten tässä tapauksessa toistettavan videotiedoston ja ennen videon toistamista soittimessa näkyvän kuvan sijainnit.

Flash Player ei ole sinänsä tarkoitettu videoiden toistamiseen, vaan Flash-sovellusten ajamiseen. Jotta sitä voidaan käyttää kyseiseen tarkoitukseen, on luotava Flash-sovellus, joka mahdollistaa videoiden toistamisen. Flash-sovelluksissa voidaan käyttää joko Flash Video- tai MPEG4-muotoisia videoita. Flash-videosoitinten ulkoasu ja toiminnot ovat pitkälti riippuvaisia ainoastaan kyseisen sovelluksen toteutuksesta.

3.3 HTML 5 video-elementti

HTML5 on HTML-standardin viimeisin työn alla oleva versio. Sen työstäminen aloitettiin vuonna 2004. HTML5:n tarkoituksena on päästä irti nykyisistä verkkosivujen rakentamisen yleisistä käytännöistä. Ongelmana on se, että nykyiset HTML- ja XHTML-toteutukset yhdistelevät monien eri määrittelydokumenttien, selainten ja käytäntöjen ominaisuuksia, mikä heikentää verkkosisällön yhteensopivuutta eri sovellusten kanssa. HTML5 pyrkii järkeistämään verkkosisältöjen toteuttamista. Lisäksi se tuo mukanaan joitain uusia ominaisuuksia, joita aiemmissa versioissa ei ole. (Wikipedia 2011g.)

Verkkoautomessujen kannalta tärkein näistä uudistuksista on videoelementti. Koska HTML5 on edelleen työn alla, ei videoelementin käytölle ole selaimissa vielä kovin laajaa tukea. HTML5-standardissa ei määritellä, mitä formaattia selaimessa toistettavien videoiden tulee olla. Tällä hetkellä eri selaimet tukevat hieman eri formaatteja. Kaikkiaan tuettuja formaatteja on kolme: Ogg, MP4 ja WebM. Lisäksi videon ja audion tulee olla enkoodattu oikealla koodekilla. Ogg:tä käytettäessä videon tulee olla enkoodattu Theoralla ja audion Vorbisilla. MP4:n tapauksessa videon H.264:llä ja audion AAC:llä sekä WebM:ää käytettäessä videon VP8:lla ja audion Vorbisilla. Taulukosta 1 käy ilmi, mitä formaatteja yleisimmät selaimet ja mobiilialustat tukevat ja mistä versiosta lähtien.

Taulukko 1. Selainversioiden formaattituet (Pilgrim 2010)

Formaatti	IE	Firefox	Opera	Chrome	Safari	iPhone	Android
Ogg	-	3.5+	10.5+	5.0+	-	-	-
MP4	9.0+	-	-	5.0+	3.0+	3.0+	2.0+
WebM	9.0+	4.0+	10.5+	6.0+	-	-	-

Internet Explorer 9:n tuki WebM:lle ei ole täysin natiivi. Se toimii vain, jos käyttäjän koneelle on asennettu VP8-koodekki. Lisäksi Safari toistaa kaikkia samoja formaatteja kuin Quicktime, mutta Quicktimessa on esiasennettuna tuki vain MP4:lle.

Jotta video toistuu varmasti kaikilla selaimilla, tulee sen olla saatavilla jokaisessa edellä mainitussa kolmessa formaatissa. Yhtä kaikilla selaimilla varmasti toistuvaa formaattia ei ole, eikä todennäköisesti tule olemaankaan ainakaan lähitulevaisuudessa. HTML5:ssä on kuitenkin keino saada käytetty selain valitsemaan siinä toimiva video saatavilla olevista formaateista. Lisäksi viimeisenä varmuuskeinona voidaan käyttää Flash-liitännäistä. Video-elementin sisälle voidaan sijoittaa Flash-videon sisältävä object-elementti, jonka sisältö näytetään, jos käytetty selain ei ymmärrä video-elementtiä.

Ideaalitilanteessa koodiesimerkki 10 riittäisi videon näyttämiseen selaimessa kuin selaimessa. Edellä mainituista syistä johtuen tämä ei ole kuitenkaan tällä hetkellä vielä suositeltavaa, sillä tämä esimerkki toimisi vain osalla käyttäjistä.

```
<video src=”video.ogg”></video>
```

KOODIESIMERKKI 10. Video-elementti yksinkertaisimmillaan

Jos videotiedostoja on vain yksi, voidaan käyttää src-attribuuttia osoittamaan videotiedoston sijainti. Koska videotiedostoja on kuitenkin hyvä olla useampi, jotta sivusto voi tukea useampia selaimia, on käytettävä video-elementin sisälle sijoitettavaa source-elementtiä. Koodiesimerkissä 11 määritetään samalle video-elementille kolme mahdollista lähdettä.

```
<video width=”320” height=”240”>
<source src="video.mp4" type='video/mp4; codecs="avc1.42E01E,
mp4a.40.2"' />
<source src="video.webm" type='video/webm; codecs="vp8, vorbis"' />
<source src="video.ogv" type='video/ogg; codecs="theora, vorbis"' />
</video>
```

KOODIESIMERKKI 11. Video-elementin käyttö usealla videoformaatilla

Selaimet käyvät läpi video-elementin alla olevia source-elementtejä, kunnes löytävät formaatin, jonka kyseinen selain osaa toistaa. Source-elementeille annetaan src-attribuutti, joka määrittelee tiedostojen sijainnit, sekä lisäksi type-attribuutti, joka kertoo selaimelle videon formaatin. Type-attribuutti muodostuu kahdesta osasta, MIME-tyy-

pistä ja koodekeista. MIME-tyyppi kertoo selaimelle videon formaatin, ja koodekkiosa formaatissa käytetyt video- ja audiokoodekit. Pelkkä MIME-tyypin määrittäminen ei välttämättä riitä, vaan on myös varmistettava, että sivuston palvelin on konfiguroitu tukemaan kyseisiä tyyppejä. Video-elementille on hyvä määrittää korkeus ja leveys width- ja height-attribuuteilla. Selaimet eivät skaalaa videoita. Jos elementin koko eroaa videon koosta, keskitetään video automaattisesti elementin keskelle. (Pilgrim 2010.)

4 SOVELLUKSEN TOIMINTA

Sovelluksen etusivu toimii porttina sivuston sisältöihin. Sen huomiota kiinnittävin elementti on Flash-sovellus, jonka linkeistä käyttäjä ohjataan valitsemalleen messuosastolle, eli kyseisen sisältökategorian videoselainsivulle. Etusivulta löytyy myös suorat linkit info-, palaute- ja huutokauppasivuille. Lisäksi etusivun alalaidassa sijaitsee palkki, jossa on suorat linkit messujen kolmeen uusimpaan videoon. Kuvassa 1 on verkkoautomessujen etusivu, jonka käyttäjä näkee ensimmäisenä saapuessa sivustolle.

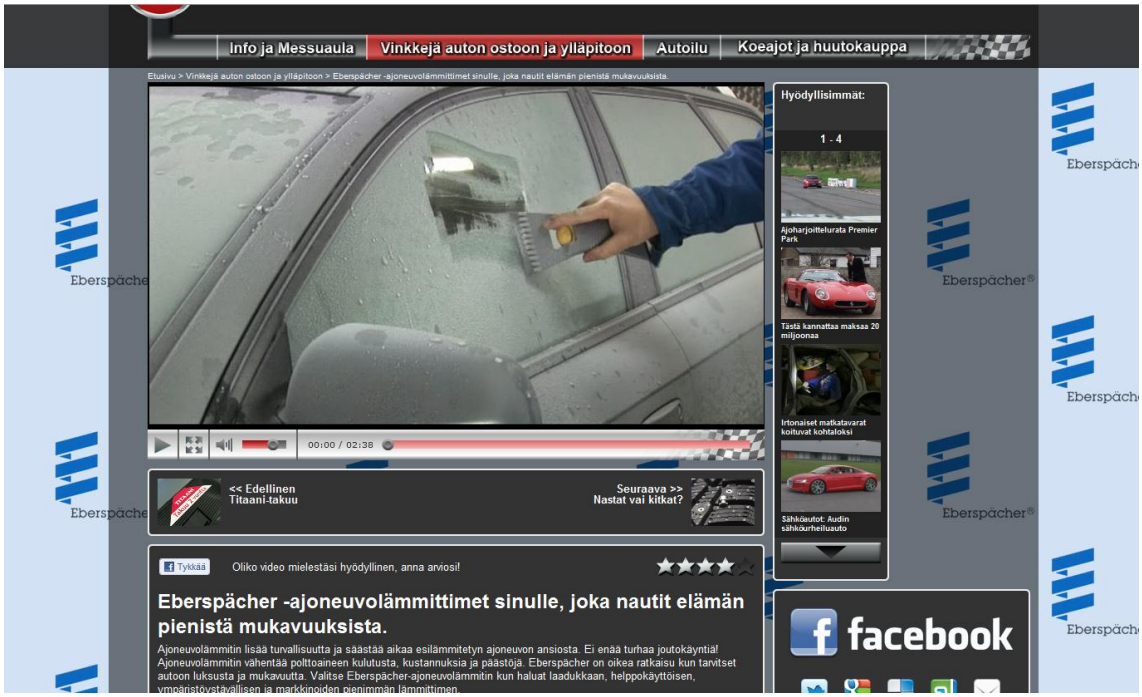
KUVA 1. Verkkoautomessujen etusivu

Sovelluksen tarkoitus käyttäjän näkökulmasta on toimia alustana, jonka kautta käyttäjä pääsee käsiksi häntä kiinnostavaan sisältöön. Jotta navigointi sivuilla olisi vaivatonta, on sovelluksen päänavigaatiovälineenä toimiva valikko sijoitettu jokaisen sivun ylälaitaan ja on ulkonäöltään aina samanlainen. Tämän päävalikon kautta käyttäjä pääsee käsiksi jokaiseen sivuun. Valinnat on jaettu neljän pudotusvalikon alle. Kolme sisältökategorioiden mukaan erotettua valikkoa pitää sisällään varsinaiseen videosisältöön vievät valinnat ja neljännestä valikosta pääsee info- ja palautesivuille. Osassa valikoista valinnat on jaettu vielä valikon sisällä useampaan sivuun, joita käyttäjä voi selata valikon alalaidasta löytyvillä Edellinen- ja Seuraava-painikkeilla. Pudotusvalikot aukeavat vie-

mällä hiiren osoittimen halutun valikon päälle. Tämän huomattiin vaikeuttavan sovelluksen käyttöä mobiililaitteilla.

Sovelluksen päänavigaatiossa on näkyvissä vain kunkin sisältövideon otsikko. Pelkän otsikon avulla käyttäjä ei välttämättä saa tarpeeksi tietoa siitä, mitä otsikon alle kätkeytyy. Tästä syystä sovelluksessa on erillinen videoselain, johon pääsee käsiksi joko etusivulta klikkaamalla haluamaansa aluetta tai klikkaamalla päänavigaation pudotusvalikkojen otsakkeita. Videoselain koostuu taulukon muotoon järjestetyistä korteista, joissa on kuvankaappaus videosta, videon otsikko ja lyhyt kuvaus. Taulukkoa voi vierittää sivuttais- ja pystysuunnissa liikuttamalla hiirtä. Jos osiossa on suuri määrä videoita, jakaantuu taulukko useammalle sivulle ja sen oikeassa alareunassa on sivunvaihtopainikkeet.

Sovelluksen tärkein sivu on kohdesivu. Se pitää sisällään varsinaisen videosoittimen, joka näyttää käyttäjän valitseman videon. Välittömästi videosoittimen alapuolella on linkit kyseisen sisältöosaston edelliseen ja seuraavaan videoon. Näiden linkkien avulla käyttäjä voi kätevästi edetä järjestyksessä sisällöstä toiseen. Tämän lisäksi käyttäjä voi antaa arvionsa kyseisestä videosta asteikolla 1-5 klikkaamalla videosoittimen alta löytyvää arviointityökalua, joka sijaitsee videon kuvaustekstikentän yläosassa. Jokaisesta videosta on mahdollista antaa palautetta. Palaute ei näy muille käyttäjille, vaan ohjautuu tiettyyn sähköpostiosoitteeseen. Jatkossa olisi hyvä miettiä, voisiko käyttäjille antaa mahdollisuuden kommentoida messujen sisältöä niin, että se näkyy myös muille käyttäjille. Videosoittimen vieressä vasemmalla puolella on lista 20:sta korkeimmalle arvioidusta videosta. Listassa näkyy kerrallaan vain neljä kohdetta, mutta listaa voi selata eteen- ja taaksepäin video kerrallaan sen ylä- ja alaosasta löytyvillä painikkeilla. Kuvassa 2 on kohdesivu, jolta löytyy sovelluksen tärkeimmät ominaisuudet, kuten videosoitin. Kuvan oikeasta laidasta puuttuu mainosbanneri, koska kuva on otettu sovelluksen testiversiosta, jossa mainokset eivät olleet toiminnassa.



KUVA 2. Verkkoautomessujen kohdesivu

Verkkoautomessujen aikana Mikko.fi-verkkohuutokaupassa oli huudettavana verkkoautomessuihin liittyviä kohteita. Yhdeltä verkkoautomessujen sivulta oli mahdollista nähdä kaikki sillä hetkellä huudettavana olleet kohteet ja huudettavaksi tulemassa olleet kohteet. Huutoja ei ole mahdollista tehdä sovelluksen kautta, vaan huutokauppasivulla on vain lista kohteista, joista saa lisätietoa viemällä hiiren osoittimen kohteen päälle. Klikkaamalla kohteen otsikkoa tai Huuda kohde! -painiketta sovellus ohjaa käyttäjän kyseisen kohteen sivulle Mikko.fi-palvelussa.

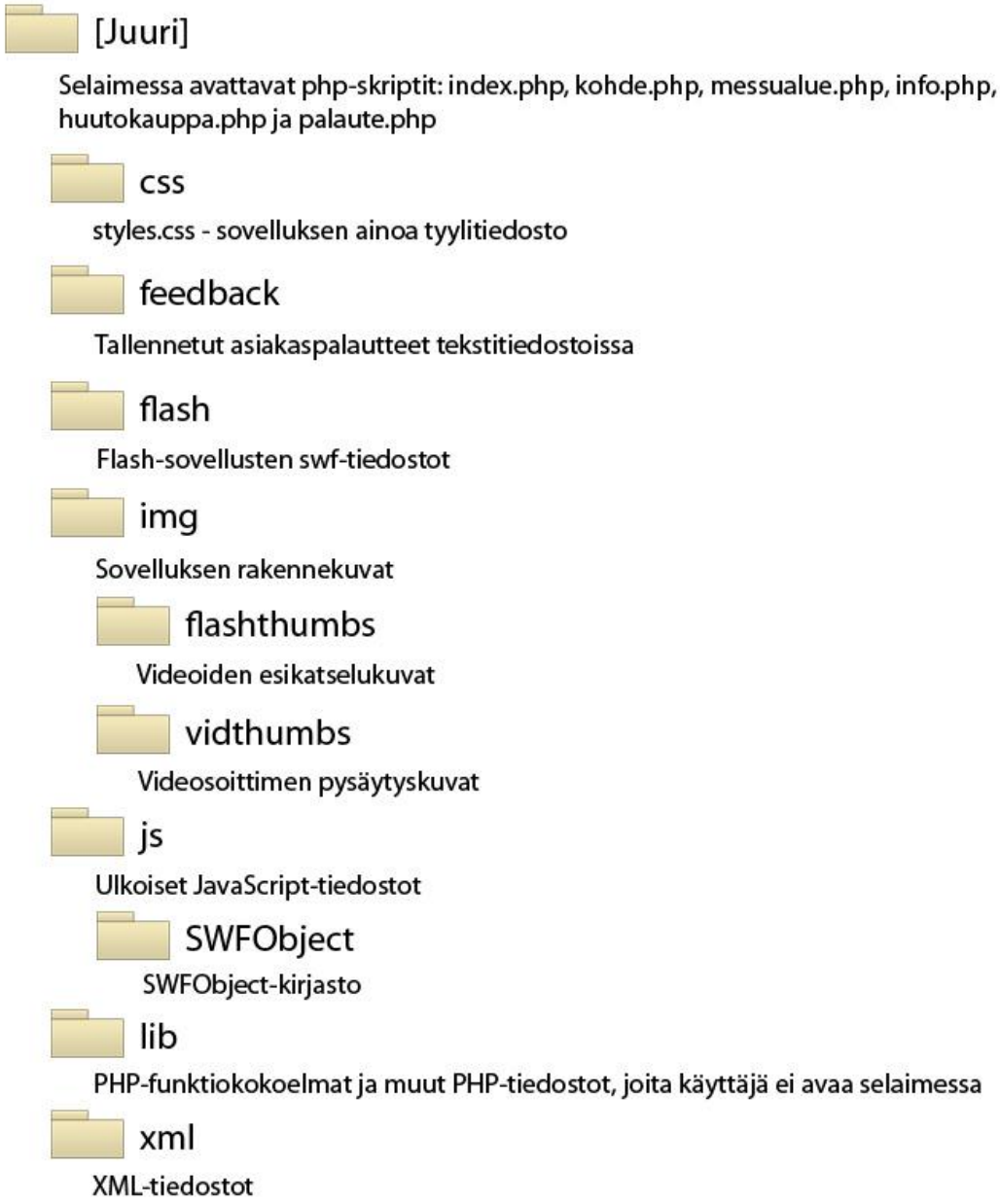
5 SOVELLUKSEN TOTEUTUS

5.1 Rakenteen toteutus

Sovelluksen rakenne haluttiin pitää verrattain yksinkertaisena, joten se toteutettiin staattisia sivuja lähestyvillä menetelmillä. Esimerkiksi PHP-ohjelmoinnissa ei käytetty lainkaan luokkia, vaan toteutus on täysin proseduraalinen. Sovelluksen HTML-koodi tulostetaan toimintojen mukaan erillisiin tiedostoihin jaettujen funktioiden avulla ja niiden toimintaa ohjataan syöttämällä niille halutut parametrit. HTML-standariksi valittiin XHTML 1.0 Transitional, koska se antaa hieman enemmän sovelluksen vaatimia vapauksia, kuin XHTML 1.0 Strict.

Koko sovellus käyttää samaa ulkoista CSS-tyylitiedostoa, jossa määritellään sovelluksen eri osien ulkonäkö. Ulkoista tyylitiedostoa käyttämällä voidaan varmistua siitä, että tehdyt muutokset vaikuttavat varmasti kaikkiin samaa tyyliä käyttäviin elementteihin. Lisäksi sovelluksen ulkonäköä on helpompi muokata, kun kaikki tyylimääritykset on keskitetty samaan paikkaan.

Kaikissa sivuissa käytetyn rakenteellisen HTML:n tulostukseen käytetyt funktiot on sijoitettu printfunctions.php:ksi nimettyyn tiedostoon. Tämä tiedosto sisällytetään jokaiseen sen funktiota käyttävään tiedostoon PHP:n require_once-funktiolla. Tämän funktiokokoelman tarkoituksena on pitää kaikki sovelluksen rakenteessa yleisesti käytetyn HTML:n tulostus yhdessä paikassa. Tiedostossa on esimerkiksi funktio printMeta, jonka avulla tulostetaan HTML-dokumentin alkuun käytetyn standardin tyyppimäärittely, head-elementti, CSS-tyylitiedoston sijainnin määrittävä link-elementti ja dokumentin metatiedot. Funktioiden avulla tulostetaan myös HTML-koodi sivujen varsinaisen rakenteen alku- ja loppuosille, kävijäseurantojen javascript-koodit, jakopalveluihin liittyvät elementit sekä mainospaikat. Sivujen varsinainen sisältö luodaan kunkin käyttäjälle näkyvän sivun omassa PHP-tiedostossa. Kuvasta 3 käy ilmi sovelluksen hakemistorakenne.



KUVA 3. Hakemistorakenne

Sivusto koostuu viidestä käyttäjälle näkyvästä sivusta. Nämä sivut luodaan index.php-, info.php-, kohde.php-, messualue.php- ja huutokauppa.php-skripteissä, jotka ajetaan, kun käyttäjä avaa ne selaimellaan. Näistä sivuista index.php, info.php ja huutokauppa.php näkyvät käyttäjälle aina samanlaisina, eikä niille generoidun HTML-dokumentin sisältö tai rakenne riipu käyttäjän valinnoista. Kohde.php ja messualue.php sen sijaan generoivat HTML-dokumenttinsa saamiensa parametrien mukaisesti. Skriptit ottavat parametrit vastaan sivulla käytettyihin url-osoitteisiin sijoitettuina get-viesteinä.

Koska käyttäjällä ei ole mahdollisuutta saada kovin helposti tietoonsa skriptien vaatimia parametrejä, ovat sovelluksen navigaatiot toteutuksen kannalta tärkeässä asemassa. Päävalikon JavaScript-toteutus hakee kaiken sovelluksen XML-tiedostojen sisältämän tiedon ja luo sen perusteella valikon, jossa on oikeanlaiset get-viestit sisältävät linkit kaikkeen sivuston sisältöön.

Jokaiselle sivulle on linkitetty jQuery-kirjasto ja menu.js-skriptitiedosto. Kun sivu ladataan, hakee menu.js XML-tiedostoista tiedot valikkolinkkien vaatimille parametreille. Tämän jälkeen valikon sivujen HTML-koodit luodaan yksi kerrallaan ja tallennetaan taulukkomuuttujiin messualueen perusteella. Käyttäjän klikatessa valikoiden Seuraava- tai Edellinen-painikkeita, hakee sovellus kyseisistä taulukkomuuttujista valikossa esitettävän sivun HTML-koodin, ja vaihtaa elementin sisällön. Sivun vaihto tapahtuu printPage-funktiossa, joka ottaa parametreina muokattavan valikon div-elementin id-attribuutin sekä halutun sivunumeron. Funktio lisää myös valikon loppuun sivunvaihtopainikkeet riippuen siitä, mikä valikon sivu on kyseessä. Sivunvaihtopainikkeiden painaminen kutsuu previousPage- tai nextPage-funktioita, jotka huolehtivat sivunumeromuuttujien päivittämisestä, ja kutsuvat printPage-funktiota päivitetyllä sivunumerolla. Koodiesimerkki 12 esittelee liukuanimaation toteutukseen vaaditun jQuery-pohjaisen koodin.

```
$(document).ready(function() {
	$('#navBar ul li').hover(function(){
		$(this).children('div').stop(true,
true).delay(200).slideDown(300).css("z-index", "100");
	}, function(){
		$(this).children('div').stop(true,
true).delay(200).slideUp().css("z-index", "0");
	});
});
```

KOODIESIMERKKI 12. Navigaation liukuanimaation toteutus

Valikon animointi tapahtuu jQuery-kirjaston avulla. Div-elementin, jonka id-attribuuttina on #navBar, alla sijaitsevan järjestämättömän listan li-elementtehin on liitetty jQueryn hover-funktio, jota kutsutaan, kun käyttäjä tuo hiiren osoittimen elementin päälle. Osoittimen tullessa elementin päälle tai poistuessa sen päältä, animoidaan sen lapsielementtinä olevan div-elementin korkeusarvoa jQueryn slideDown- ja slideUp-funktioiden avulla. Tämän toteutuksen toimimisen ehtona on, että valikon HTML-rakenne on toteutettu oikein.

5.2 XML-tiedostot

Koska projektilla ei ollut käytössä resursseja tietokantaa varten, jouduttiin kaikki sivujen HTML:n generointia varten tarvittava tieto säilömään XML-tiedostoihin. Tietojen tallentamiseen käytetään muutamaa erillistä XML-tiedostoa, mutta erilaisia rakenteita on vain kahta tyyppiä. Tiedostoja käytetään sisällön generointiin sekä videoiden käyttäjäarvioiden tallentamiseen. Molemmissa tapauksissa nämä tiedot on jaettu kolmeen erilliseen tiedostoon messualueen perusteella, joten tiedostoja on yhteensä kuusi.

Koodiesimerkistä 13 selviää kohde- ja videoselainsivun sisällön generointiin käytettyjen XML-tiedostojen rakenne. Kaikki tiedoston elementit eivät välttämättä sisällä tietoa ja saman elementin eri tapaukset voivat pitää sisällään tyypiltään toisistaan eroavaa tietoa. Elementtien käyttötarkoitukset säilyvät kuitenkin samoina. Taulukossa 2 esitellään kaikkien elementtien käyttötarkoitukset.

```
<content>
    <topic area="vinkit0">
        <id>titaanitakuu</id>
        <tag>A1</tag>
        <producer></producer>
        <brandId>22.1</brandId>
        <wallpaper>tsusho_wallpaper_tile.gif</wallpaper>
        <Flashtitle>Titaani-takuu</Flashtitle>
        <title>Titaani-takuu</title>
        <section>Vinkkejä auton ostoon ja ylläpitoon</section>
        <Flashtype>video</Flashtype>
        <thumb>img/Flash/Flashthumbs/titaanitakuu_thumb.jpg</thumb>
        <Flashdesc>Toyota Titaani-vaihtoautot</Flashdesc>
        <desc>Toyota Titaani-vaihtoautot</desc>
        <flv>TOYOTA_TITAANI.flv</flv>
        <videoStill>img/vidthumbs/titaanitakuu_thumb.jpg</videoStill>
    </topic>
</content>
```

KOODIESIMERKKI 13. Sisältötiedostojen rakenne

TAULUKKO 2. Sisältötiedostojen elementit

content	XML-tiedoston juurielementti, jonka lapsiksi sijoitetaan kaikki muut elementit.
topic	Tämän elementin lapsielementit pitävät sisällään yhden videon tiedot.
id	Videon tunniste, joka erottaa sen muista videoista. Yksittäistä videota haetaan tiedostoista tämän perusteella.
tag	Videoselaimessa näytettävä videon pikatunniste.
producer	Määrittää videoselaimelle, mistä lähteestä kyseinen video on saatu.
brandId	Automerkin tunniste autotalli.comin jälleenmyyjähaulle.
wallpaper	Kohdesivulla käytettävän taustakuvan tiedostonimi.
Flashtitle	Videoselaimessa näytettävä videon lyhennetty otsikko.
title	Kohdesivulla näytettävä pidempi otsikko.
section	Videon messualue.
Flashtype	Tämän elementin sisällön perusteella valitaan kohdesivulla käytettävä videosoitin.
thumb	Videon pikkukuvan tiedostonimi.
Flashdesc	Videoselaimessa näytettävä lyhyt kuvaus videosta.
desc	Kohdesivulla näytettävä pidempi kuvaus.
flv	Viittaus varsinaiseen videoon. Voi olla joko tiedostonimi tai iltalehden netti-tv:tä käyttävien videoiden tapauksessa videon tunniste.
videoStill	Ennen videon toistamisen aloittamista videosoittimessa näytettävän kuvan sijainti.

Koodiesimerkki 14 kuvaa videoiden arviointidatan tallentamiseen käytettävien XML-tiedostojen rakennetta. Rakenne on muutoin sama kuin sisältötiedostoissa, mutta tunnistetiedon lisäksi topic-elementillä on lapsinaan vain sum- ja votes-elementti. Sum-elementtiin on säilötty kaikkien videoista annettujen arvioiden yhteenlaskettu summa ja votes-elementti pitää sisällään annettujen arvioiden määrän. Näiden tietojen avulla voidaan laskea videon arvioiden keskiarvo, joka näytetään videoselaimessa ja kohdesivulla. Videon arviointitiedot löydetään id-elementin avulla, jonka sisällön tulee olla sama sekä sisältö- että arviointitiedostoissa. Arviointidatan olisi voinut periaatteessa sisällyttää samaan tiedostoon kuin sisältödata, mutta näimme parhaaksi erottaa ne eri tiedostoihin, koska sisältödatan tiedostoja jouduttaisiin todennäköisesti muokkaamaan useammin. Näin vältyttiin mahdollisilta arviointidatan vääristymisiltä, kun sisältötiedostoja muoka-

taan. Lisäksi arviointitiedostojen koko pysyi pienempänä, mikä nopeuttaa sovelluksen toimintaa.

```
<content>
    <topic area="vinkit0">
        <id>titaanitakuu</id>
        <sum>18</sum>
        <votes>5</votes>
    </topic>
<content>
```

KOODIESIMERKKI 14. Arviointidatan XML-tiedostojen perusrakenne

5.3 Kohdesivun toteutus

Kohdesivu on sovelluksen kaikista sivuista tärkein, ja siitä syystä se on myös toteutukseltaan laajin. Sivulta vaaditut toiminnallisuudet olivat videoiden katselemisen mahdollistava videosoitin, videoiden arviointimahdollisuus, selailtava sivupalkki, josta käyttäjä näkee parhaat arviot saaneet videot ja pääsee suoraan niiden kohdesivuille sekä linkit seuraavaan ja edelliseen videoon. Kohdesivulla oli myös mahdollista antaa palautetta kyseisestä videosta, ja jos video koski jotain tiettyä automerkkiä, hakea lähin merkkiä myyvä autoliike autotalli.com:in jälleenmyyjähaulla. Kohdesivun toiminnot mahdollistivat myös videon jakamisen sosiaalisen median palveluissa, kuten Facebookissa.

Generoitu HTML-koodi tulostetaan kohde.php:ssä pääasiassa PHP:n echo-rakennetta käyttäen. Skriptin suorittaminen etenee proseduraalisesti tiedoston alusta loppuun. Skriptin alussa tulostetaan funktioiden avulla HTML-sivun tarvittavat rakenneosat, kuten head-elementti, metatiedot jne. Varsinaisen sisällön HTML-koodi rakennetaan get-viesteinä ja XML-tiedostoista saadun tiedon perusteella erilaisten ehtolauseiden ja silmukoiden avulla. Kun sivu on tulostettu ja lähetetty käyttäjälle, voidaan sitä vielä räätälöidä JavaScriptin ja jQueryn avulla.

5.3.1 Videosoittimien valinta ja upotus

Videoita tuodaan sivulle Iltalehden netti-tv:stä, SuomiTV:stä, sekä verkkoautomessujen omalta palvelimelta itse kehitetyn videosoittimen avulla. Kohde.php valitsee käytettävän soittimen XML-tiedostosta löytyvän Flashtype-elementin sisällön perusteella. Kun kohde.php avataan selaimella, ottaa se ensimmäisenä talteen alue- ja kohde-get-viestit ja tallentaa ne muuttujiin. Aluemuuttujan perusteella valitaan avattava XML-tiedosto. Tämä tapahtuu koodiesimerkissä 15 esitetyllä tavalla. If-lausetta käytetään, jotta vältytään turhilta virheilmoituksilta, mikäli käyttäjä yrittää itse avata kohde.php:tä suoraan selaimessa ilman get-viestejä. Avattava tiedosto asetetaan file-muuttujaan switch case-rakenteen avulla.

```
//Get area and target content. GET sent in url.
if (!is_null($_GET['alue']) && !is_null($_GET['kohde'])) {
      $XMLId = $_GET['alue'];
      $id = $_GET['kohde'];
}

//Set content XML-file according to area.
switch ($XMLId) {
    case 'vinkit':
        $file = 'XML/content1videos.XML';
        break;
    case 'autoilu':
        $file = 'XML/content2videos.XML';
        break;
    case 'uutuudet':
        $file = 'XML/content3videos.XML';
        break;
}
```

KOODIESIMERKKI 15. Get-viestien tallennus ja XML-tiedoston valinta

Kohdemuuttuja viittaa XML-tiedostossa topic-elementin alla sijaitsevan id-elementin sisältöön. Kohde.php:ssä luodaan uusi DOMDocument-olio, ja ladataan siihen aluemuuttujan määrittämä XML-tiedosto. Tämän jälkeen haetaan kaikki tiedoston topic-elementit ja käydään niitä läpi, kunnes löydetään sellainen topic-elementti, jonka alla sijaitsevan id-elementin sisältö vastaa get-viestinä saatua kohdemuuttujaa. Kohdesivun luomiseen vaadittava tieto haetaan kyseisen topic-elementin lapsielementeistä ja tallennetaan muuttujiin. Koodiesimerkki 16 esittelee XML-tiedoston lataamisen PHP:llä. DOMDocument-olioon ladataan XML-tiedosto load-metodilla, ja tämän jälkeen DOM-muotoista tietoa käsitellään DOM-olioiden metodien avulla.

```
//Load XML-file.
$doc = new DOMDocument; //Create new dom document.
$doc->load($file); //Load the requested XML-file to the dom document.
$topics = $doc->getElementsByTagName('topic'); //Save all 'topic'
nodes into a variable.

//Search the nodes untill the requested content is found.
foreach ($topics as $topic) {
      $foundId = $topic->getElementsByTagName('id')->item(0)-
>childNodes->item(0)->nodeValue;
      if($foundId == $id) {
            $flv = $topic->getElementsByTagName('flv')->item(0)-
>childNodes->item(0)->nodeValue;
            $type = $topic->getElementsByTagName('Flashtype')-
>item(0)->childNodes->item(0)->nodeValue;
            $producer = $topic->getElementsByTagName('producer')-
>item(0)->childNodes->item(0)->nodeValue;
...
      }
}
```

KOODIESIMERKKI 16. XML-tiedoston avaaminen ja tiedon kerääminen

Kaikki videosoittimen upottamiseen sivulle käytetyt funktiot on sijoitettu tiedostoon printPlayer.php, joka tuodaan kohde.php:n käytettäväksi require_once-funktiolla. Tiedosto pitää sisällään kolme funktiota: printAltPlayer, printSwfObjectPlayer ja printSuomiTVPlayer. Normaalitilanteessa käytetään joko printSwfObjectPlayer- tai printSuomiTVPlayer-funktiota. SWFObject on pieni JavaScript-kirjasto, jolla voidaan helposti upottaa sivulle Flash-sisältöä, tässä tapauksessa haluttu videosoitin standardeja noudattaen. Koska kaikilla käyttäjillä ei välttämättä ole selaimessa JavaScript käytössä, toteutettiin sovellukseen myös staattinen Flashin upotus object-elementtiä käyttäen. Tästä huolehtii printAltPlayer-funktio. Jos JavaScript on käytössä, korvautuu printAltPlayer-funktion luoma HTML-koodi SWFObjectin vastaavalla. SuomiTV:n soittimen tapauksessa jouduttiin turvautumaan aina staattiseen upottamiseen, koska tapa, jolla se on toteutettu ei tue SWFObjectin käyttöä. SWFObject otettiin käyttöön, koska sen avulla on helppo upottaa Flash-sovelluksia standardien mukaisesti riippumatta käyttäjän selaimesta ja asetuksista. Koodiesimerkki 17 esittelee videosoittimen valinnan switch-case-rakenteen avulla. Jokaisessa switch-case-rakenteen tapauksessa kutsutaan printSwfObjectPlayer-funktiota, mutta hieman eri parametreillä riippuen halutusta soittimesta.

```
if ($type != 'suomitv') {
switch ($type) {
      case 'video':
            printSwfObjectPlayer('Flash/videoPlayer.swf', $flv, $type,
            $videoStill);
            break;
      case 'embed':
            printSwfObjectPlayer('http://www.goodmoodtv.com/internettv
            /application/iltalehti/system/SingleVideoPlayer.swf',
            $flv, $type);
            break;
      case 'youtube':
            printSwfObjectPlayer('http://www.youtube.com/v/' . $flv .
            '?fs=1&amp;rel=0&amp;color1=0x272a2a&amp;color2=0xd8d8d8&a
            mp;border=0&amp;enablejsapi=1&amp;playerapiid=ytplayer',
            $flv);
            break;
}
}
else if ($type == 'suomitv') {
      printSuomiTVPlayer();
}
```

KOODIESIMERKKI 17. Videosoittimen valinta ja tulostusfunktion kutsuminen

Funktiot printSwfObjectPlayer ja printAltPlayer ottavat parametreina halutun soittimen swf-tiedoston sijainnin, videotiedoston sijainnin, videon lähteen ja pysäytetyssä soittimessa näytettävän still-kuvan sijainnin. Still-kuvaa käytetään vain sovellusta varten toteutetussa omassa videosoittimessa. Nämä tiedot on siis haettu aiemmin XML-tiedostosta. Näiden parametrien perusteella rakennetaan param-elementit object-elementin sisään staattisessa upotuksessa tai tallennetaan ne JavaScript-muuttujiin ja annetaan SWFObjectin embedSWF-funktion parametreiksi. Tietoa videon lähteestä käytetään tarvittavien muuttujien tai param-elementtien rakentamiseen, koska eri lähteiden soittimet vaativat hieman erilaiset asetukset. Sitä ei välitetä varsinaisille soittimille. Funktiot palauttavat generoimansa HTML-koodin tai script-elementin sisään sijoitetun JavaScript-koodin.

Videosoittimen upottamisen ja soittimen valinnan toteutuksesta tuli loppujenlopuksi melko sekava ja monimutkainen. Varsinkin toteutetun koodin luettavuus jätti jonkin verran toivomisen varaa. Toteutus onnistui kuitenkin hyvin, kun ottaa huomioon, kuinka monta eri tavalla toteutettua soitinta oli saatava upotettua samaan sovellukseen ja ottamaan siltä vastaan parametreja.

5.3.2 Arviointijärjestelmä

Arviointijärjestelmä on toteutettu osittain JavaScriptillä ja osittain PHP:llä. JavaScriptillä ja jQueryllä pidetään huoli arviointijärjestelmän käyttöliittymästä ja PHP:llä tallennetaan käyttäjän antama arvio sekä palautetaan JavaScriptille esitettäväksi kyseisen videon uusi arvosana. Arviointijärjestelmän toteutuksessa oli tärkeää, että arvion antaminen ei aiheuta sivun uudelleenlatausta ja katkaise videon katselua. Tästä syystä arviointijärjestelmän toteutuksessa käytettiin hyväksi Ajaxia. Kaikki kohdesivun JavaScript-koodit on sijoitettu tiedostoon targetpage.js ja arvion tallentamiseen käytetty PHP-skripti sijaitsee tiedostossa saverating.php.

Arviointi perustuu viiden tähden asteikkoon. Käyttäjä vie hiiren osoittimen tähtien päälle, ja klikkaa tähteä riippuen siitä, kuinka monta tähteä käyttäjä haluaa videolle antaa. Tähtiä korostetaan riippuen siitä, minkä tähden päällä osoitin on. Kun osoitin ei ole tähtien päällä, näytetään videon senhetkinen arvosana.

Tähtien korostaminen on toteutettu yksinkertaisella jQuery-koodilla, joka vaatii arviointijärjestelmältä tietynlaisen HTML-rakenteen. Toteutuksen jQuery-koodit on esitetty koodiesimerkissä 18. Järjestelmän juurena on div-elementti, jonka id-attribuutin arvo on ratingContainer ja taustakuvana himmennetyt tähdet. Tämän elementin alla on div-elementtejä, joiden class-attribuuttina on starbg. Varsinainen toiminnallisuus on sidottu näihin elementteihin. Näiden elementtien alla taas on vielä yksi div-elementti, jonka taustakuvaksi on määritetty korostettu tähti. Kun käyttäjä tuo kursorin jonkin div-elementin päälle, jonka class-attribuutti on starbg, haetaan HTML-rakenteesta jQueryn avulla kyseinen elementti ja kaikki sitä edeltävät sisarelementit. Näiden elementtien lapsielementit, eli korostetun tähden grafiikan sisältävät elementit, tuodaan näkyviin jQueryn show-funktiolla ja kaikki seuraavat sisarelementit piilotetaan hide-funktion avulla. Hide- ja show-funktioita ei voida käyttää suoraan elementtiin, johon mouseenter- ja mouseleave-funktiot on sidottu, sillä kyseisiä funktiota ei suoriteta, jos elementti on piilotettu, kun osoitin tuodaan sen päälle.

Kun kursori viedään kokonaan pois arviointijärjestelmän päältä, tuodaan esiin kyseisen videon arvosana. Tämä tapahtuu kutsumalla showRating-funktiota, joka määrittää kuinka monta tähteä kyseisessä tilanteessa tulee näyttää pyöristämällä videon arvosana. Ar-

vosana lasketaan arviointijärjestelmän HTML-koodin generoinnin yhteydessä ja tallennetaan julkiseen muuttujaan. Funktio hakee kaikki div-elementit, joiden class-attribuuttina on star ja piilottaa ne. Tämän jälkeen valitaan näistä elementeistä niin monta elementtiä slice-funktion avulla, kuin videon arvosana on pyöristettynä, ja tuodaan ne takaisin näkyviin.

```
//Show and hide rating stars on mouse over.
function stars(){
	showRating();
	$("div.starbg").mouseenter(function(){
		$(this).prevAll().andSelf().children().show();
		$(this).nextAll().children().hide();
	});

	$("div#ratingContainer").mouseleave(function() {
		showRating();
	});
}

//Show current rating.
function showRating(){
	$('div.star').hide();
	$('div.star').slice(0, Math.round(targetRating)).show();
}
```

KOODIESIMERKKI 18. Arviointijärjestelmän tähtien korostus

Arviointijärjestelmän toiminta riippuu myös siitä, onko käyttäjä antanut jo arvion kyseisestä videosta. Jos arvio on jo annettu, näytetään käyttäjälle vain videon arvosana, kun osoitin ei ole arviointijärjestelmän päällä ja kiitosviesti, kun osoitin on arviointijärjestelmän päällä. Arvioiduista videoista pidetään kirjaa evästeen avulla. Eväste pitää sisällään merkkijonon, jossa on kaikkien käyttäjän arvioimien videoiden tunnisteet pilkulla eroteltuna. Kun käyttäjä antaa arvion videosta, tarkistetaan ensin, onko evästettä nimellä ratedIds olemassa. Jos eväste on olemassa, haetaan sen sisältö ja erotetaan JavaScriptin split-funktion avulla taulukkomuuttujaan. Tämän jälkeen taulukkoon lisätään arvioidun videon tunniste, ja se liitetään takaisin merkkijonoksi join-funktion avulla, jonka jälkeen merkkijono sijoitetaan takaisin evästeeseen. Jos evästettä ei ole vielä olemassa, luodaan se ja sille annetaan arvoksi kyseisen videon tunniste. Evästeen luomiseen käytettään erillistä createCookie-funktiota, joka ottaa parametreikseen evästeen nimen, evästeen arvoksi halutun merkkijonon ja kuinka monta päivää evästeen halutaan säilyvän käyttäjän selaimessa. Koodiesimerkki 19 esittelee tunnisteen lisäämisen evästeeseen.

```
//Add this id to a cookie for keeping track of which videos have been
rated.
if(readCookie("ratedIds") != null){
      var ids = readCookie("ratedIds").split(',');
      ids.push(id);
      createCookie("ratedIds", ids.join(','), 45);
}else{
      createCookie("ratedIds", id, 45);
}
```

KOODIESIMERKKI 19. Videon tunnisteen lisääminen evästeeseen

Arviointijärjestelmän HTML-koodin generointi tapahtuu JavaScriptin avulla printRatingSystem-funktiossa. Kyseinen funktio ottaa parametreikseen videon tunnisteen ja messualueen. Funktion alussa tarkistetaan, ovatko evästeet käytössä käyttäjän selaimessa. Tarkistus on toteutettu yrittämällä luoda uusi testieväste JavaScriptin kautta ja lukemalla sen idexOf-arvoa. Jos evästeet eivät ole käytössä, ei käyttäjä voi antaa arviota videosta. Muutoin käyttäjä voisi antaa samasta videosta rajattoman määrän arvioita. Käyttäjä voi edelleen antaa useita arvioita poistamalla sovelluksen luoman evästeen selaimestaan ja lataamalla sivun uudelleen, mutta tämän suurempaa estettä useampien arvioiden antamiselle ei koettu järkeväksi toteuttaa. Kun evästeet eivät ole käytössä, näyttää arviointijärjestelmä kyseisen videon arvion, mutta kun käyttäjä tuo osoittimen tähtien päälle, näytetään viesti, joka kertoo arvioinnin vaativan evästeiden käyttöönoton.

Jos evästeet ovat käytössä, haetaan ratedIds-eväste ja jaetaan se jälleen taulukkomuuttujaan split-funktiolla. Taulukkoa käydään läpi ja verrataan sen arvoja parametrinä saatuun videon tunnisteeseen. Jos kyseisen videon tunniste löytyy taulukosta, generoidaan HTML-koodi, joka näyttää videon arvosanan ja kiitosviestin. Mikäli kyseisen videon tunnistetta ei löydy taulukosta, generoidaan HTML-koodi, jossa tähtielementteihin on sidottu onClick-tapahtuma, joka kutsuu rate-funktiota parametreinään kyseisen videon tunniste ja messualue sekä annettava arvosana, joka riippuu klikatusta tähtielementistä. Videon arvosana lasketaan XML-tiedostosta haettujen äänten määrän ja arvioiden summan perusteella.

Käyttäjän antama arvio tallennetaan rate-funktion avulla. Funktio luo uuden XMLHttpRequest-olion ja lähettää sen avulla arvion palvelimelle tallennettavaksi. Saadut parametrit lähetetään palvelimelle pyynnön mukana post-viesteinä, jotka saverating.php lukee. Saverating.php:n toiminta on hyvin yksinkertaista. Se valitsee avattavan XML-tiedoston post-viestinä saadun messualueen perusteella ja lataa sen uudeksi

DOMDocument-olioksi. Tämän jälkeen annettujen arvioiden määrää lisätään yhdellä ja käyttäjän arvio lisätään arvioiden yhteissummaan. Muutokset tehdään olion nodeihin videon tunnisteen perusteella. Kun muutokset on tehty, lasketaan uudesta äänimäärästä ja äänien summasta keskiarvo, joka on videon uusi arvosana. Lopuksi XML-tiedoston sisältö korvataan juuri muokatun DOMDocument-olion sisällöllä ja uusi arvosana palautetaan käyttäjälle. Targetpage.js ottaa uuden arvosanan vastaan, ja vaihtaa sen julkiseen arvosanamuuttujaan, jolloin näkyvä arvosana vaihtuu ilman sivun uudelleenlataamista. Koodiesimerkki 20 esittelee annetun arvosanan lähettämisen palvelimella sijaitsevan saverating.php:n käsiteltäväksi. Esimerkissä näkyy myös palvelimen vastauksen vastaanottaminen, eli kyseessä on yksinkertainen esimerkki Ajaxin toiminnasta.

```
if (window.XMLHttpRequest){
      XMLhttp=new XMLHttpRequest();
}else{
      XMLhttp=new ActiveXObject("Microsoft.XMLHTTP");
}
XMLhttp.onreadystatechange=function(){
      if (XMLhttp.readyState==4 && XMLhttp.status==200){
            if(XMLhttp.responseText > 0){
                  targetRating = XMLhttp.responseText;
            }
            printRatingSystem(XMLId, id);
      }
}
XMLhttp.open("POST", "lib/saverating.php", true);
XMLhttp.setRequestHeader("Content-type","application/x-www-form-
urlencoded");
XMLhttp.send("area=" + XMLId + "&kohde=" + id + "&rating=" + rating);
```

KOODIESIMERKKI 20. Arvosanan lähettäminen palvelimelle

Arvioinitjärjestelmän toteutus vaikutti kokemusten perusteella toimivalta. Käyttäjien arviot tallentuivat ja videoiden arviot päivittyivät oikein. Jos sovelluksella olisi huomattavasti enemmän käyttäjiä, saattaisi XML-tiedostojen käyttämisestä arviointidatan säilömiseen ilmetä ongelmia. Kahdesta samaan aikaan tapahtuvasta tallennuksesta voisi toinen jäädä toteutumatta tiedoston ollessa käytössä. Tässä skaalassa toteutus kuitenkin vaikutti riittävän toimivalta, eikä suuria vääristymiä päässyt luultavasti tapahtumaan.

5.3.3 Hyödyllisimmät videot

Kohdesivun oikeassa laidassa sijaitsee palkki, josta näkee 20 korkeimman arvosanan saanutta videota. Palkissa näkyy kerralla vain 4 videota, mutta niitä voi selailla videoiden esikatselukuvien ylä- ja alapuolelta löytyvillä painikkeilla. Videoiden otsikkoja tai kuvia klikkaamalla pääsee kyseisen videon kohdesivulle.

Parhaiden videoiden hakeminen ja järjestely tapahtuu printHighlights-funktiossa, joka on sijoitettu toprated.php:hen. Tiedoston funktiot tuodaan kohde.php:n käyttöön require_once-funktiolla. Funktion alussa ladataan kaikki XML-tiedostot ja tallennetaan jokaisen tiedoston kaikki topic-elementit muuttujiin tiedostoittain. Tämän jälkeen topic-elementit yhdistetään kahteen taulukkomuuttujaan siten, että toinen pitää sisällään kaikki sisällöngenerointitiedostojen elementit ja toinen arviointitiedostojen elementit.

Kun topic-elementit on saatu taulukoihin, aloitetaan sen taulukon läpikäynti, jossa on arviointitiedostojen topic-elementit. Jokaisen elementin kohdalla haetaan videon tunniste, videon saamien arvioiden määrä ja arvioiden summa. Näiden perusteella lasketaan videon arvosana, eli arvioiden keskiarvo. Keskiarvoa verrataan 20 alkion taulukkoon, johon korkeimmat arvosanat sijoitetaan. Jos taulukosta löytyy pienempi arvosana kuin kyseisen videon saama arvosana, sijoitetaan se taulukkoon pienemmän arvosanan paikalle. Taulukkoa käydään aina läpi järjestyksessä siten, että arviot säilyvät järjestyksessä pienimmästä suurimpaan. Kun taulukkoon syötetään uusi arvio, siirretään pienempiä alkioita yksi paikka taaksepäin siten, että jokainen alkio korvaa yhden itseään pienemmän. Tällöin viimeinen alkio jää kokonaan pois korvautuessaan toiseksi viimeisellä.

Pelkällä suuruuden mukaan järjestetyllä arvosanataulukolla ei voida vielä generoida HTML-koodia. Tarvitaan taulukko, jossa sisältötiedostojen topic-elementit ovat samassa järjestyksessä kuin kyseisten videoiden arvosanat arvosanataulukossa. Kun arvosanataulukkoon lisätään uusi arvo, tallennetaan sen paikka taulukossa muuttujaan. Seuraavaksi sisältötiedostojen topic-elementeistä etsitään se elementti, jonka tunniste on sama kuin sillä arvosanatiedostojen topic-elementillä, jonka arvosana sijoitettiin arvosanataulukkoon. Tämä sisältötietoelementti tallennetaan samankokoiseen taulukkomuuttujaan ja samaan paikkaan, kuin arvosanataulukossa. Näin saadaan aikaan 20 alkiota käsittävä taulukko, jossa arvosanataulukon arvosanoja vastaavat sisältöelementit ovat samassa

järjestyksessä kuin arvosanat. Koska elementit ovat järjestyksessä pienimmästä suurimpaan, täytyy taulukon alkioiden järjestys vielä muuttaa käänteiseksi. Koodiesimerkissä 21 on esitettynä funktio, joka työntää taulukkomuuttujaan uuden arvon tiettyyn alkioon, ja siirtää edellisiä alkioita taaksepäin, pudottaen viimeisen alkion ulos taulukosta. Koodiesimerkki 22 esittelee käsittelyssä olevan videon arvosanan vertaamisen 20 parhaan arvosanan taulukkoon sekä arvosanataulukon ja sisältöelementtitaulukon uudelleenjärjestämisen.

```
function insertValue($array, $key, $value) {
	for($i = 0; $i <= $key; $i ++){
		if($i != $key){
			$array[$i] = $array[$i+1];
		}else{
			$array[$i] = $value;
		}
	}
	return $array;
}
```

KOODIESIMERKKI 21. Uuden alkion syöttäminen taulukkoon

```
for($n = 0; $n < count($topAvgs); $n++){
//Set key position if a match is found.
	if($avg > $topAvgs[$n]){
		$key = $n;
	}else if($avg == $topAvgs[$n]){
		if ($votes >= $topVotes[$n]){
			$key = $n;
		}
	}
}
//If key is set, add the new topic into top videos array.
if(isset($key)){
	$topVotes = insertValue($topVotes, $key, $votes);
	if($topAvgs[$key] <= $avg){
		$topAvgs = insertValue($topAvgs, $key, $avg);
		$topTopic = null;
		//Find the corresponding video in the topic array.
		foreach ($topicArray as $video) {
			$vId = $video->getElementsByTagName("id")->item(0)-
>childNodes->item(0)->nodeValue;
			if ($vId == $id) {
				$topTopic = $video;
				$vId = null;
				break;
			}
		}
		$topVids = insertValue($topVids, $key, $topTopic);
	}
}
```

KOODIESIMERKKI 22. Arvosanojen vertaaminen ja taulukon muokkaus

HTML-koodia generoitaessa käydään läpi taulukkoa, johon on sijoitettu arvosanojen mukaan järjestetyt sisältöelementit. Jokaisen elementin tiedoista luodaan pätkä HTML-koodia, joka esittää palkissa yhden videon esikatselukuvan ja otsikon, jotka toimivat samalla linkkeinä videon kohdesivulle. Nämä koodinpätkät yhdistetään peräkkäin, jolloin ne järjestyvät arvosanan mukaan parhaasta huonoimpaan. Videoiden selaaminen on toteutettu jQueryn avulla. Kaikki div-elementit, joiden sisään videoiden kuvat ja otsikot on sijoitettu, ovat neljää ensimmäistä elementtiä lukuun ottamatta piilotettu määrittämällä CSS-tiedostossa niiden display-arvoksi none. Kun käyttäjä painaa listaa alaspäin selaavaa painiketta, ensimmäinen näkyvä div-elementti piilotetaan jQueryn hide-funktion avulla ja ensimmäinen piilotettu elementti tuodaan näkyviin. Koska piilotetut elementit eivät vaikuta sivun asemointiin, hyppää toinen näkyvä elementti palkissa ensimmäiseksi ja ensimmäinen piilotettu elementti tulee esiin palkin viimeiseksi. Tämä luo käyttäjälle illuusion siitä, että elementit liikkuvat palkissa painikkeita painamalla. Koodiesimerkki 23 esittelee Edellinen-painikkeen toteutuksen. Esimerkin muuttujat first ja last pitävät sisällään tiedon siitä, monesko hyödyllisimmät videot sisällään pitävän div-elementin lapsielementeistä kuuluu olla ylimpänä ja monesko alimpana. Items-tauluk-komuuttuja pitää sisällään kaikki XML-tiedoston sisältöelementeistä generoidun HTML-koodin div-elementit. Halutut elementit haetaan taulukosta first- ja last-muuttujien avulla. Muutujien arvoja muutetaan käyttäjän klikatessa painiketta. Seuraava-painke toimii samalla periaatteella.

```
$('div#previousHighlight').click(function() {
	if(last < items.length) {
		items.eq(first).hide();
		items.eq(last).show();
		first++;
		last++;
		setCurrentHighlights(first+1, last);
	}
	if(last == items.length){
		$(this).hide();
	}
	if(first > 0){
		$('div#nextHighlight').show();
	}
});
```

KOODIESIMERKKI 23. Edellinen-painikkeen jQuery-toteutus

Jälkikäteen ajatellen hyödyllisimpien videoiden listan olisi voinut tehdä hieman järkevämminkin. Sen sijaan, että jokaiselle käyttäjälle olisi kohdesivua ladattaessa generoitu

aina uusi lista hyödyllisimmistä videoista, olisi voitu luoda yksi HTML-tiedosto, joka olisi pitänyt sisällään ajantasaisen HTML-koodin listasta. Tämä koodi olisi voitu sitten yksinkertaisesti ladata tiedostosta kohdesivulle. Tällä tavalla listaa olisi tarvinnut päivittää vain silloin, kun käyttäjät antavat uusia arvioita. Listan päivittäminen olisi voitu tehdä saverating.php:ssä arvion tallentamisen yhteydessä. Nykyinen toteutus kuitenkin on myös toimiva, mutta aiheuttaa ehkä hieman turhaa työtä palvelimelle ja hidastaa sivun latausta.

5.4 Videoselaimen toteutus

Sovelluksen videoselain toteutettiin Flash-sovelluksena. Videoselaimen päätarkoitus oli toimia navigaatiovalikkoa visuaalisempana ja informatiivisempana navigointityökaluna. Se loi käyttäjälle myös vaikutelmaa messuilla kiertelemisestä tuomalla videot esiin omina paketteinaan tietynlaisina messukojuina. Videoselaimessa esitetään kuva videosta, videon otsikko, lyhyt kuvaus videosta ja videon arvosana.

Selain on upotettu messualue.php:ssä generoituun HTML-koodiin. Messualue.php ottaa vastaan yhden get-viestin, jonka arvona on esitettävä messualue. Saatu arvo lähetetään edelleen Flash-sovellukselle param-elementin avulla. Selainsovellus käyttää tätä parametriä avattavien XML-tiedostojen valitsemiseen. Jos käyttäjällä ei ole asennettuna Flash Player -liitännäistä, näytetään videoselaimen tilalla lista kyseisen messualueen videoista, jossa näkyy vain videoiden esikatselukuvat ja otsikot, jotka toimivat myös linkkeinä kohdesivulle.

Selainsovellus toteutettiin oliopohjaisesti Actionscript 3.0:lla. Luokkia on yhteensä seitsemän. Tärkeimmät näistä ovat Main, XMLLoader, ContentWrapper ja ContentBox. Main-luokka käynnistää sovelluksen ja asettaa tarvittavat alkuarvot muuttujiin. Se myös pitää sisällään sovelluksen pääajastimen, jonka avulla on toteutettu selaimen animointi hiiren liikkeiden mukaan. ContentWrapper-luokka muodostaa videoselaimen sisällön luomalla sisälleen olioita ContentBox-luokasta. XMLLoader-luokka pitää sisällään toiminnallisuudet XML-tiedostojen avaamiselle ja niiden elementtien arvojen hakemiselle. Kun sovellus käynnistyy, haetaan ensimmäisenä HTML-koodissa param-elementillä sovellukselle lähetetty messualueen tunniste. Tämän tunnisteen avulla sovellus tietää,

minkä messualueen videoita käyttäjä haluaa selata ja valitsee avattavat XML-tiedostot. Tunnisteen hakeminen esitellään koodiesimerkissä 24.

```
var browserType:String;
var FlashVars:Object = this.loaderInfo.parameters;
if (FlashVars.area) {
      browserType = String(FlashVars.area);
} else {
      trace("Browser type not found.");
      browserType = "vinkit";
}
```

KOODIESIMERKKI 24. Param-elementissä annettujen tietojen hakeminen

Sisällön generointi aloitetaan luomalla olio ContentWrapper-luokasta. Se saa parametrikseen messualueen tunnisteen. Olion luominen tapahtuu Main-luokan metodissa SetStage, joka myös luo selaimen taustagrafiikkoja ja käynnistää sovelluksen ajastimet. ContentWrapper-olion muodostin lataa parametrinä saadun messualueen perusteella valitun XML-tiedoston uuteen XMLLoader-olioon. Kun lataaminen on suoritettu, alkaa ContentWrapper luoda sisälleen olioita ContentBox-luokasta, jotka pitävät kukin sisällään tiedot yhdestä videosta. ContentWrapper-luokassa ladattua XML-tiedostoa käytetään vain selainsovelluksen sivun rakenteen muodostamiseen. Videoiden määrä XML-tiedostossa määrittää, kuinka monta saraketta videoiden selaimeen muodostamassa taulukossa on. Mitä enemmän videoita on, sitä useampia sarakkeita selaimeen muodostuu. Videoiden muodostama taulukko yritetään pitää mahdollisimman tasasivuisena, sillä taulukon muoto vaikuttaa selaimen animointiin. Jos taulukko on hyvin leveä tai korkea, on animaation liukuminen toiseen suuntaan nopeampaa, mikä tekee videoiden selailusta hankalaa ja epäintuitiivista. XML-tiedoston lataaminen Flash-sovellukseen esitellään koodiesimerkissä 25.

```
myLoader = new URLLoader();
myLoader.load(new URLRequest("XML/content" + fileNum + "videos.XML"));
XMLReady = false;
myLoader.addEventListener(Event.COMPLETE, XMLHandler, false, 0, true);
private function XMLHandler(e:Event):void {
      myXML = new XML(e.target.data);
      XMLReady = true;
}
```

KOODIESIMERKKI 25. XML-tiedoston lataaminen

Uutta ContentBox-oliota luotaessa sille annetaan parametreinä messualue ja XML-tiedostosta käytettävän topic-elementin numero. XMLLoader-luokasta luotujen olioiden metodit hakevat kaikki messualueen perusteella valitun XML-tiedoston topic-elementit ja valitsevat halutun tiedon numeron perusteella. Näitä metodeja käytetään yksittäisten videoiden tietopakettien luomiseen. Esimerkkinä koodiesimerkki 26:n otsikonhakumetodi.

```
//Get box title.
public function returnTitle(i:int):String {
      var XMLList:XMLList = myXML.topic.Flashtitle;
      var nodeTitle:String = XMLList[i];
      XMLList = null;
      return nodeTitle;
}
```

KOODIESIMERKKI 26. XMLLoader-luokan otsikonhakumetodi

XML-tiedostoista haettuja tekstejä, kuten videon kuvausta, varten luodaan uusia tekstikenttiä ja ne asetetaan videolaatikon alle. Kuvat ladataan erillisellä ImageLoader-luokalla. Kuvat ovat kyseisen luokan olioita. Koska kuvien lataaminen kestää pidempään kuin tekstien, päätettiin kuvien lataaminen toteuttaa siten, että se näkyy käyttäjälle. Kun sovellus ei vielä ole ehtinyt ladata kuvaa, näytetään sen tilalla latausanimaatio. Animaatio ei kerro käyttäjälle latauksen edistymistä, mutta antaa käyttäjän ymmärtää, että kuvaa ladataan. Luokan muodostimen lisäksi ainoa metodi onLoaderReady kuuntelee kuvaa lataavaa Loader-olioa. Kun lataus on valmistunut, poistaa onLoaderReady latausanimaation ja korvaa sen ladatulla kuvalla.

Tietojen lataaminen XML-tiedostoista olisi voitu toteuttaa hieman yksinkertaisemminkin. Tässä toteutuksessa luodaan jokaisen videon laatikon luomisen kohdalla uusi olio XMLLoader-luokasta, joka johtaa siihen, että XML-tiedosto joudutaan lataamaan monta kertaa peräkkäin. Toteutuksesta olisi saatu tehokkaampi siten, että XML-tiedostot olisi ladattu vain kerran ContentWrapper-luokassa, ja halutut tiedot olisi annettu parametreinä ContentBox-olioille.

5.5 Palautejärjestelmä ja Addthis-jakopalvelu

Käyttäjällä oli mahdollisuus antaa palautetta sekä yksittäisistä videoista että verkkoautomessuista kokonaisuutena. Yleispalautetta varten toteutettiin erillinen sivu, jolla on ainoastaan lomake palautteen antamista varten. Sivuun pääsee käsiksi navigaatiovalikon kautta. Yksittäisten videoiden palautteen kohdalla palautelomake sisällytettiin kohdesivuun. Sivulla on palautelinkki, jota klikkaamalla aukeaa sivunsisäinen jQueryllä toteutettu ponnahdusikkuna, joka sisältää yksinkertaisen lomakkeen. Käyttäjän ei tarvitse itse ilmoittaa, mistä videosta antaa palautetta, vaan palautteeseen liitetään automaattisesti tieto siitä, minkä videon kohdesivulta palaute on lähetetty.

Addthis on täysin ilmainen palvelu, jonka avulla käyttäjät voivat jakaa verkkosisältöjä sosiaalisissa medioissa. Se tukee yli 300:a erilaista sosiaalisen median palvelua. Addthis tarjoaa myös mahdollisuuden seurata tilastoja siitä, mitä verkkosivun sisältöjä käyttäjät ovat jakaneet, ja kuinka paljon liikennettä nämä jaot ovat tuoneet sivulle.

5.5.1 Palautejärjestelmä

Palautesivu, eli palaute.php, on melko yksinkertainen. Sivulla on lomake, jossa on kentät palautteelle sivuston sisällöstä, ulkonäöstä ja käytettävyydestä sekä yleisille kommenteille. Käyttäjä voi myös halutessaan antaa nimensä ja sähköpostiosoitteensa, mutta vain yksi palautekentistä on pakko täyttää. Tyhjää lomaketta ei käsitellä, vaan käyttäjälle annetaan virheilmoitus.

Lomakkeen tiedot otetaan vastaan post-viesteinä ja tallennetaan muuttujiin. Muuttujat alustetaan ensin tyhjiksi merkkijonoiksi ja post-viestit tallennetaan niiden päälle, jos niihin on annettu sisältöä. Jos kaikki palautemuuttujat ovat tyhjiä, tulostuu sivulle virheilmoitus, kun se ladataan uudelleen lomakkeen lähettämisen yhteydessä. Lomake lähettää tiedot siis samalle sivulle, kuin mille se on sijoitettu, eli palaute.php:lle. Jos palautetta on annettu, tallennetaan palautteet ja tiedot lähettäjästä kahteen erilliseen merkkijonoon, jotka on muotoiltu helposti luettaviksi. Toinen merkkijono tallennetaan tiedostoon ja toinen lähetetään sähköpostina tietylle vastaanottajalle.

Sähköpostiosoitteen muoto tarkistetaan aina, kun se on annettu. Jos käyttäjä syöttää sähköpostikenttään virheellisen osoitteen, annetaan virheilmoitus ja palautetta ei tallenneta. Tarkistus tehdään myös sähköpostitoiminnon väärinkäyttämisen ehkäisemiseksi. Sähköpostiosoitteen muoto tarkistetaan säännöllisen lausekkeen avulla, käyttäen php:n preg_match-funktiota. Koodiesimerkki 27 esittelee sähköpostiosoitteen tarkistukseen käytetyn funktion. PHP:n Preg_match-funktio ottaa parametreikseen tarkistukseen käytettävän säännöllisen lausekkeen ja tarkastettavan merkkijonon, eli tässä tapauksessa sähköpostiosoitteen.

```
function checkEmail($email){
      if (preg_match("/^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-
Z]{2,4}$/" , $email) || empty($email)) {
            return true;
      }
      return false;
}
```

KOODIESIMERKKI 27. Sähköpostiosoitteen tarkistava funktio

Tiedostoon tallentaminen tapahtuu yksinkertaisesti PHP:n fopen-, fwrite- ja fclose-funktioilla. Palautetiedosto nimetään aina päivämäärän mukaan. Tämä tapahtuu fopen-funktiolla, joka joko avaa tiedoston tai luo sen, riippuen siitä, onko tiedostoa olemassa. Päivämäärä haetaan PHP:n date-funktiolla, jonka parametriksi on annettu ”dmY”. Tämä saa päivämäärä tulostumaan muotoon päivä-kuukausi-vuosi. Yleisen palautteen kohdalla päivämäärän perään lisätään vielä sana ”yleinen” ja tiedoston päätteeksi tulee txt. Koska fopen-funktiolle on annettu kirjoitusmoodiparametriksi ”a”, lisää fwrite aina annetun tiedon tiedoston loppuun. Kaikki samana päivänä tulleet palautteet tallentuvat näin ollen automaattisesti samaan tiedostoon. Palautteeseen liitetään date-funktion avulla myös kellonaika. Koodiesimerkki 28 esittelee palautteen tallentamisen tiedostoon.

```
$file = "feedback/" . date("dmY") . "yleinen.txt";
$handle = fopen($file, "a");

$entry = "--- " . date("H:i") . " ---\n->Lähettäjä: " . $name . "\n-
>Sähköposti: " . $email . "\n->User Agent: " . $userAgent . "\n-
>Kommentteja sisällöstä:\n". $content . "\n->Kommentteja ulkoasusta ja
käytettävyydestä:\n" . $design . "\n->Muita kommentteja:\n" . $general
. "\n";

fwrite($handle, $entry);
fclose($handle);
```

KOODIESIMERKKI 28. Palautteen tallentaminen tiedostoon

Palautteen lähettämiseen sähköpostina käytetään PHP:n mail-funktiota, jolle annetaan parametreiksi vastaanottajan osoite, viestin sisältö ja lähettäjän osoitteen määrittävä http-otsikko. Lähettäjän palautelomakkeeseen antama sähköpostiosoite tulee lähettäjäksi asiakaspalvelun PHP-skriptiltä vastaanottamaan palauteviestiin, jolloin viestiin on helppo vastata. Jos sähköpostiosoitetta ei ole annettu, tulee lähettäjäksi tekaistu osoite, josta lähettäjän tunnistaa palautelomakkeeksi. Koodiesimerkki 29 esittelee palautteen lähettämisen sähköpostina.

```
$mailcontent = "Nimi: " . $name . "\nSähköposti: " . $email . "\n\n---
Kommentteja sisällöstä---\n". $content . "\n---Kommentteja ulkoasusta
ja käytettävyydestä---\n" . $design . "\n---Muita kommentteja---\n" .
$general . "\n";

if(empty($email)){
      $email = "palautelomake@verkkoautomessut.fi";
}
$headers = "From: " . $email . "\r\n";

mail($sendTo, "Verkkoautomessut - Sivustopalaute", $mailcontent,
$headers);
```

KOODIESIMERKKI 29. Palautteen lähettäminen sähköpostina

5.5.2 Addthis

Addthisin käyttöönotto ei vaadi suuria toimenpiteitä. Palveluun voi rekisteröidä käyttäjätunnuksen, jonka perusteella voidaan seurata käyttäjätilastoja. Rekisteröityminen on pakollista vain, jos käyttäjä haluaa seurata tilastoja. Palvelu generoi valmiiksi pätkän HTML-koodia, jonka sijoittamalla omaan sivustoon palvelu saadaan käyttöön. Ennen HTML-koodin generointia on mahdollista valita useasta eri käyttöliittymävaihtoehdosta. Käyttöliittymää voi myös kustomoida vapaasti HTML-koodia ja CSS-määrityksiä muuttamalla.

Addthis käyttää HTML-koodissa tiettyjä class-attribuutteja. Sen toiminta perustuu Addthisin palvelimelta script-elementin avulla tuotavaan ulkoiseen JavaScript-tiedostoon. Tiettyä class-attribuuttia käyttävä a-elementti korvataan JavaScriptissä tiettyä palvelua vastaavalla jakolinkillä. Näiden elementtien CSS-määritykset tulevat automaattisesti ulkoisesta lähteestä, jos niitä ei ole määritetty sovelluksen CSS-tiedostossa.

Verkkoautomessujen toteutuksessa päädyttiin rakentamaan jakopalvelulle oma sovellukseen paremmin sopiva käyttöliittymä. Valmiit käyttöliittymäratkaisut eivät olleet riittävän huomiota herättäviä, sillä jakomahdollisuutta haluttiin korostaa käyttäjille. Jakomahdollisuus toteutettiin sovelluksen etusivulle, kohdesivulle ja huutokauppasivulle. Kohdesivulla jaettu linkki ohjasi aina siihen videoon, jonka kohdesivulla jako oli tapahtunut. Facebook oli jakomahdollisuuksissa etusijalla. Facebook-jaon linkistä tehtiin korostetuin ja lisäksi kohdesivulle lisättiin erillinen tykkää-painike. Kohdesivulla korostetaan jakopalveluista myös Twitteriä, Googlea, Deliciousia ja StumbleUponia. Lisäksi yhdestä jakopainikkeesta voi lähettää linkin kyseiselle sivulle sähköpostiviestinä. Muihin jakopalveluihin pääsee käsiksi erillisestä ponnahdusikkunasta, joka sisältyy Addthisin ominaisuuksiin, eikä sitä tarvinnut lisätä sovellukseen erikseen. Ponnahdusikkunan saa auki jakopainikkeiden alapuolelta löytyvästä linkistä.

Addthisin HTML-koodit tulostavat funktiot löytyvät tiedostosta printfunctions.php. Facbookin tykkää-painikkeen kohdalla HTML-koodit on sijoitettu suoraan tiedostoon kohde.php. Addthisin käyttämien class-attribuuttien tyylimäärittelyt tehtiin sovelluksen omassa CSS-tiedostossa, jolloin ne korvasivat palvelun omat tyylimäärittelyt. Koodiesimerkki 30 esittelee kohdesivun jakopainikkeet sisältävän laatikon HTML-koodit.

```
<!-- AddThis Start -->
<div class="addthis_toolbox">
      <div class="vertical">
            <script type="text/javascript"
            src="http://s7.addthis.com/js/250/addthis_widget.js#userna
            me=verkkoautomessut"></script>
            <div class="custom_images">
            <a class="addthis_button_facebook">
                  <img src="img/facebook.png" width="280" height="64"
                  alt="Jaa Facebookissa" />
            </a>
      </div>
      <div class="addthis_32x32_style addthis_default_style">
            <a class="addthis_button_twitter"></a>
            <a class="addthis_button_google"></a>
            <a class="addthis_button_delicious"></a>
            <a class="addthis_button_stumbleupon"></a>
            <a class="addthis_button_email"></a>
      </div>
      <div class="more">
            <a class="addthis_button_expanded">Jaa muissa palveluissa
            &gt;&gt;</a>
      </div>
</div>
<!-- AddThis End -->
```

KOODIESIMERKKI 30. Kohdesivun jakopainikkeiden Addthis-toteutus

5.4 Testaus

Sovelluksen testaaminen tapahtui pitkälti samanaikaisesti kehitystyön kanssa. Alkuvaiheessa testaus tapahtui paikallisesti kehittäjien koneille asennetun WWW-palvelimen avulla ja loppuvaiheessa sovellusta pyöritettiin erillisellä palvelimella demo-osoitteessa, jotta sovellukseen pääsi käsiksi muualtakin, kuin kehittäjien koneilta. Sovellukseen tehtiin myös joitain muutoksia messujen käynnistymisen jälkeen. Testiosoitteen avulla niitä voitiin testata ennen tuotantoon siirtämistä.

Koska sovelluksen osat eivät juuri vaikuttaneet toistensa toimintaan, voitiin ne kehittää helposti yksitellen toimiviksi saakka. Suurin koko sovelluksen toimintaan vaikuttava seikka oli XML-tiedostojen rakenne, joka täytyi pitää alusta asti mahdollisimman muuttumattomana. Jos XML-tiedostojen rakenteeseen tuli muutoksia, oli pidettävä huoli siitä, että kaikkiin sovelluksen osiin tehtiin tarvittavat muutokset. Melko suurta päänvaivaa sovelluksen toteuttamisessa aiheutti myös selainyhteensopivuus. Varsinkin CSS-määritykset, Flash-sovellusten upottaminen ja jQuery vaativat eri selainten hieman erilaisen toiminnan huomioonottamista.

6 JATKOKEHITYSEHDOTUKSIA

Verkkoautomessuista saadun palautteen ja kokemusten pohjalta arvioituna mahdollisen seuraavan version tärkein vaatimus käyttäjäkokemuksen parantamiseksi olisi sovelluksen interaktiivisuuden lisääminen.

Osa videoista tulleesta palautteesta oli pitkälti palautteen antajien mielipiteitä kyseisestä aiheesta. Koska kommentointimahdollisuus on niin yleinen, luultiin videoiden palautelomaketta ilmeisen helposti yleiseksi kommentointiominaisuudeksi. Käyttäjien mielipiteistä kyseisen videon aiheesta ei ole kovin suurta hyötyä sivuston ylläpidon kannalta, eikä siitä ole käyttäjillekään juuri hyötyä, sillä palauteviestit eivät näy muille käyttäjille. Koska kommentointiominaisuudelle vaikuttaisi olevan tilausta, olisi sellaisen kehittäminen mahdolliseen seuraavaan versioon todennäköisesti järkevää. Nykyinen XML-pohjainen toteutus tukisi tällaista ominaisuutta melko huonosti, koska sovelluksen käsittelemät tietomäärät ja tallennusten sekä hakujen määrät saattaisivat kohota liian korkeiksi. Tästä syystä olisi suositeltavaa ottaa tiedon tallennusta ja hakua varten käyttöön jokin tietokanta. Lisäksi tietokantapohjainen kommentointiominaisuus olisi jonkin verran yksinkertaisempi toteuttaa. Tietokannan avulla sovellukseen olisi muutenkin helpompi toteuttaa erilaisia interaktiivisia ominaisuuksia.

Kommentointimahdollisuus nostaa luonnollisesti kysymyksen siitä, pitäisikö sovellukseen toteuttaa käyttäjiä varten jonkinlainen käyttäjätilijärjestelmä. Myös tällaisen toteuttamiseen tarvittaisiin tietokanta. Lisäksi kommentteja pitäisi pystyä ylläpitäjän taholta muokkaamaan tai poistamaan, joten jonkinlainen hallintajärjestelmä olisi myös toteutettava. Kommentointi- ja käyttäjätilijärjestelmän avulla voitaisiin myös antaa näytteilleasettajille mahdollisuus vastata käyttäjien esittämiin kysymyksiin.

Kommentointiominaisuus voitaisiin sijoittaa kohdesivulle videon kuvauksen alapuolelle. Ominaisuus kannattaisi toteuttaa Ajaxin avulla siten, että videon katselu ei keskeytyisi kommentin lähettämisestä tai kommenttisivun vaihtamisesta. Käyttäjätunnuksia tarvittaisiin kolmea tyyppiä: kävijä, näytteilleasettaja ja ylläpitäjä. Kävijä voisi ainoastaan lukea ja kirjoittaa kommentteja omalla nimellään, mutta kommentteja voisi lukea

myös ilman käyttäjätunnusta ja sisäänkirjautumista. Näytteilleasettaja voisi kirjoittamisen lisäksi muokata ja poistaa omia viestejään, jotka sijaitsevat näytteilleasettajan oman sisällön kommenteissa. Näytteilleasettajan viestejä voitaisiin korostaa muiden kommenttien joukosta esimerkiksi jollain värillä. Ylläpitäjä pystyisi kirjoittamisen lisäksi muokkaamaan ja poistamaan kaikkia viestejä. Viestien moderoinnin helpottamiseksi voisi myös kehittää erillisen hallintasivun, jolta ylläpitäjä näkisi kaikkien kohdesivujen viestit yhdellä sivulla aikajärjestyksessä useammalle alasivulle jaoteltuna. Tältä sivulta ylläpitäjä voisi helposti seurata viestejä ja poistaa tai muokata asiattomia.

Kommentointijärjestelmään varten tarvittaisiin tietokantaan ainakin kaksi taulua. Toiseen tauluun sijoitettaisiin itse viestit, joista pitäisi tallentaa ainakin teksti, lähettäjä, aikaleima, viestin tunniste ja kohdesivun tunniste. Teksti, lähettäjä ja aikaleima näkyisivät käyttäjälle, jonka lisäksi aikaleimaa käytettäisiin viestien järjestämiseen. Viestin tunnistetta tarvittaisiin yksittäisten viestien muokkaamiseen ja poistamiseen. Kohdesivulla näytettävät viestit haettaisiin kannasta kohdesivun tunnisteen perusteella. Toinen taulu pitäisi sisällään käyttäjätunnukset. Käyttäjistä pitäisi tallentaa ainakin käyttäjänimi, salasana, käyttäjätyyppi ja näytteilleasettajan kohdesivutunnisteet. Käyttäjätyypin ja kohdesivutunnisteiden perusteella valittaisiin käyttäjän oikeudet, eli sivua generoitaessa luotaisiin kontrollit viestien muokkaamiselle ja poistamiselle. Käyttäjistä voitaisiin myös tallentaa yhteystietoja, jos haluttaisiin esimerkiksi järjestää arvontoja tai lähettää käyttäjille sähköpostia. Sähköpostin avulla voitaisiin luoda myös autentikointijärjestelmä ylimääräisten käyttäjätunnusten luomisen ehkäisemiseksi. Näytteilleasettajien käyttäjätilit luotaisiin ylläpitäjän toimesta ja tavallisia käyttäjiä varten sovellukseen täytyisi lisätä jonkinlainen rekisteröitymislomake.

7 YHTEENVETO

Kokonaisuudessaan sovelluksen tekninen toteutus onnistui hyvin. Kaikki halutut ominaisuudet saatiin toimimaan aikataulun puitteissa. Käytetyt tekniikat tukivat toisiaan hyvin ja ne olivat riittävät sovelluksen toteuttamiseen. Testauksen ja käyttäjäpalautteen perusteella sovellukseen ei myöskään jäänyt merkittäviä teknisiä ongelmia.

Sovelluksen toteuttaminen proseduraalisesti PHP:llä oli projektin kannalta toimiva idea. Sovelluksen rakenne on melko yksinkertainen, eikä sen toteuttamisesta oliopohjaisesti olisi ollut mielestäni kovin suurta hyötyä. Alun perin suunnitelmissa oli toteuttaa sivusto kokonaan staattisilla HTML-sivuilla, mutta haluttujen toimintojen toteuttaminen ei olisi siten ollut mahdollista. Jaoin tärkeät funktiot toimintojen perusteella erillisiin tiedostoihin, mikä helpotti sovelluksen rakenteen hahmottamista ja koodien muokkaamista. Näin vältyttiin myös siltä, että samaa koodia olisi jouduttu toistamaan useammassa tiedostossa. Huomattavasti helpompi tapa oli kirjoittaa esimerkiksi toistuvasti käytetyt funktiot yhteen tiedostoon ja ottaa ne käyttöön halutuissa tiedostoissa require_once-funktion avulla.

PHP oli ominaisuuksiltaan riittävä ja sopi tähän käyttötarkoitukseen hyvin. Sen avulla pystyttiin helposti käsittelemään esimerkiksi Ajaxilla lähetettyjä pyyntöjä ja sovelluksen XML-tiedostoja todella helposti. Myös PHP:n tiedostojenkäsittely- ja sähköpostiominaisuuksista oli hyötyä. PHP on mielestäni erittäin pätevä palvelinpuolen skriptikieli tämänkaltaisten projektien toteuttamiseen.

XML-tiedostojen käyttäminen tiedon säilömiseen tietokannan sijasta on hieman epätavallinen ja epäkäytännöllinen menetelmä. XML:n pääasiallinen tarkoitus on tiedon siirtäminen erilaisten järjestelmien välillä, eikä niinkään sen säilöminen. Tästä huolimatta XML-tiedostoja voidaan käyttää myös tietokannan tavoin, mutta se ei välttämättä ole suositeltavaa. Tässä projektissa se nähtiin kuitenkin tarpeelliseksi menetelmäksi, koska tietokantaa ei ollut käytettävissä, mutta sivujen generoimiseen tarvittavat tiedot oli kuitenkin säilöttävä jotenkin ja niiden sisältöä oli pystyttävä muokkaamaan sovelluksen kautta. PHP:ssä, JavaScriptissä ja Flashissa on valmiina toimivat ominaisuudet XML-

muotoisen tiedon käsittelyyn, joten mitään omia tiedonsäilöntämenetelmiä ei kannattanut kehittää. Sovelluksen toteutus olisi saattanut olla tietokantoja käyttäen hieman yksinkertaisempi, mutta ongelmia XML-tiedostojen käyttämisestä niiden sijaan ei tässä projektissa ilmennyt.

JavaScript on dynaamisten toiminnallisuuksien toteuttamisessa asiakaspuolen WWW-sovelluksissa hyvin laajalle levinnyttä, joten sen käyttäminen myös tässä sovelluksessa oli luonnollista. Vaikka aiempaa kokemusta JavaScriptin tai jQuery-kirjaston käytöstä minulla ei projektiin ryhtyessäni ollut, ei sen oppimiseen tuhlautunut juurikaan aikaa. Sovellukselta vaadittiin käyttäjän toiminnan seuraamista ja toimiin reagoimista ilman sivun uudelleenlataamista sekä sivun räätälöintiä käyttäjän selaimen asetuksista, liitännäisistä yms. riippuen, joten jokin asiakaspuolen skriptikieli oli pakko ottaa käyttöön. Paras vaihtoehto tässä tapauksessa oli mielestäni JavaScript, koska se on yleisesti käytetty ja siitä löytyi kattavasti ohjeita ja esimerkkejä.

JQuery otettiin käyttöön, koska sovelluksesta haluttiin saada visuaalisesti näyttävä, mutta sitä ei haluttu toteuttaa kokonaan Flashilla. JQuery on myös hyvin laajalle levinnyt ja sen käyttöön löytyi internetistä kattavia oppaita ja esimerkkejä. Suurin osa työstä tehtiin jQueryn oman dokumentaation perusteella. Sen käyttö helpotti erilaisten tehosteiden lisäämistä sovellukseen huomattavasti. Samanlaisten tehosteiden toteuttaminen ilman jQuerya olisi vaatinut huomattavasti enemmän aikaa ja ponnisteluita, joten sen käyttö oli hyvin perusteltua.

Haittapuoleksi jQueryn käytöstä osoittautui tässä tapauksessa sivuston yhteensopimattomuus joidenkin kosketusnäytöllisten mobiililaitteiden kanssa. Asiasta saatiin asiakaspalautetta, mutta sen perusteella ei ryhdytty toimenpiteisiin, sillä sovellusta ei ole tarkoitettu eikä suunniteltu mobiililaitteille. JQueryn käyttö sinänsä ei johda yhteensopivuusongelmiin mobiiliympäristöissä, mutta jotkin sen ominaisuudet, kuten osoittimen sijaintiin perustuvat toiminnot, saattavat olla ongelmallisia esimerkiksi kosketusnäytöillä.

Videosoittimen upottamiseen JavaScriptin avulla käytetty SWFObject-kirjasto osoittautui hyväksi valinnaksi. Oman testaamisen perusteella videosoitin näkyi oikein kaikilla yleisimmillä selaimilla JavaScriptin ollessa käytössä, eikä käyttäjiltä tullut asiasta nega-

tiivista palautetta. Myös JavaScript-tuen puuttuessa käytetty staattinen object- ja embed-elementteihin perustuva Flash-sovelluksen upottaminen toimi moitteettomasti. Flash-sisällön, kuten videosoittimen, upottaminen HTML-sivuun saattaa olla hieman mutkikasta johtuen selainten yhteensopivuusongelmista varsinkin, jos halutaan kirjoittaa standardeja noudattavaa HTML-koodia.

Esittelin tässä dokumentissa myös hieman HTML 5:n video-ominaisuuksia. Tämän katsauksen perusteella niitä ei ole vielä kovin suositeltavaa ottaa käyttöön. Ei olisi järkevää muuttaa videoita kolmeen eri muotoon kaikkien selainten tukemista varten, kun Flash-soittimen avulla selvitään yhdellä flv-muotoisella videotiedostolla. Lisäksi HTML 5 -toteutuksessa olisi jouduttu sijoittamaan kaikki videot omalle palvelimelle sen sijaan, että nykyisessä toteutuksessa voidaan esimerkiksi yksinkertaisesti upottaa Iltalehden netti-tv osaksi sovellusta. HTML 5:n video-ominaisuudet olisivat periaatteessa hyvin käytännöllisiä ja yksinkertaisia välineitä videosisällön lisäämiseen, mutta johtuen selainten nykyisestä huonosta tuesta sille, sen käytölle ei ole vielä tarpeeksi painavia syitä. HTML 5:n ja selainten kehitystä kannattaa kuitenkin pitää silmällä

Sosiaaliset mediat ovat nykyään melko suosittu markkinointikanava. Tästä syystä verkkoautomessuihinkin haluttiin sisällyttää ominaisuus, joka helpottaisi messujen sisällön jakamista käyttäjien keskuudessa. Monet sosiaalisen median palvelut tarjoavat ohjelmointirajapinnan, jonka avulla palvelusta erillisille verkkosivuille voidaan luoda mahdollisuus sisällön jakamiseen kyseisessä palvelussa. Katsottiin kuitenkin parhaaksi ottaa käyttöön Addthis-palvelu, jonka avulla pystyy luomaan kerralla jakomahdollisuuden yli kolmeensataan eri palveluun. Jakomahdollisuuden luominen jokaiselle palvelulle erikseen olisi vaatinut huomattavasti enemmän aikaa ja perehtymistä eri palveluiden rajapintoihin. Toinen syy Addthisin käyttöön oli palvelun antama mahdollisuus tarkastella tilastoja käyttäjien tekemistä jaoista. Ilman Addthisiä jakojen tilastoinnin toteuttaminen olisi ollut hankalaa. Addthisin ominaisuudet olivat mielestäni verkkoautomessujen käyttötarkoituksiin erittäin sopivat.

LÄHDELUETTELO

Adobe Systems. 2011. Flash Player Penetration. Luettu 10.1.2011.
http://www.adobe.com/products/player_census/Flashplayer/

Chapman S. JavaScript Execution Order. Luettu 1.3.2011.
http://javascript.about.com/od/hintsandtips/a/exeorder.htm

Fergus, Daniel C. The Object Tag. Luettu 20.1.2011.
http://www.danfergusdesign.com/classfiles/generalReference/objectTag

Kahate, A. 2009. XML & Related Technologies. Delhi: Pearson Education India.

Keogh, J. 2005. JavaScript demystified. New York: McGraw-Hill Professional Pu-
lishing.

Multimedia Basics – Incorporating Video on Your Web Site. 2011. Luettu 3.1.2011.
http://www.cvwp.com/pdf/multimedia_basics_white_paper.pdf

Pilgrim, Mark. 2010. Video on the Web – Dive into HTML5. Luettu 12.1.2011.
http://diveintoHTML5.org/video

Pint Inc. 2011. HTML Element reference. Luettu 20.1.2011.
http://www.HTMLref.com/reference/appa/tag_object.htm

The PHP Group. 2011a. What is PHP? Luettu 14.2.2011.
http://www.php.net/manual/en/intro-whatis.php

The PHP Group. 2011b. What can PHP do? Luettu 14.2.2011.
http://www.php.net/manual/en/intro-whatcando.php

W3Schools. 2011a. XML DOM Tutorial. Luettu 7.3.2011.
http://www.w3schools.com/dom/default.asp

W3Schools. 2011b. HTML DOM Tutorial. Luettu 8.3.2011.
http://www.w3schools.com/HTMLdom/default.asp

W3Schools. 2011c. JavaScript Events. Luettu 3.3.2011.
http://www.w3schools.com/js/js_events.asp

W3Schools. 2011d. JavaScript Where To. Luettu 1.3.2011.
http://www.w3schools.com/js/js_whereto.asp

W3Schools. 2011e. HTML object tag. Luettu 5.1.2011.
http://www.w3schools.com/tags/tag_object.asp

Wikipedia. 2011a. JavaScript. Luettu 1.3.2011.
http://en.wikipedia.org/wiki/Javascript

Wikipedia. 2011b. jQuery. Luettu 2.3.2011.
http://en.wikipedia.org/wiki/JQuery

Wikipedia. 2011c. Ajax. Luettu 3.3.2011.
http://en.wikipedia.org/wiki/Ajax_(programming)

Wikipedia. 2011d. Adobe Flash. Luettu 8.3.2011.
http://en.wikipedia.org/wiki/Adobe_Flash

Wikipedia. 2011e. ActionScript. Luettu 8.3.2011.
http://en.wikipedia.org/wiki/ActionScript

Wikipedia. 2011f. Liitännäinen (tietotekniikka). Luettu 3.1.2011.
http://fi.wikipedia.org/wiki/Liit%C3%A4nn%C3%A4inen_%28tietotekniikka%29

Wikipedia. 2011g. HTML5. Luettu 11.1.2011.
http://en.wikipedia.org/wiki/HTML5

World Wide Web Consortium. 2011. XHTML Object Module. Luettu 20.1.2011.
http://www.w3.org/TR/2002/WD-xHTML2-20020805/mod-object