YRITYKSEN INFORMATIIVINEN NÄYTÖNSÄÄSTÄJÄ

LAHDEN AMMATTIKORKEAKOULU
Mediatekniikan koulutusohjelma
Teknisen visualisoinnin suuntautumisvaihtoehto
Opinnäytetyö
14.5.2007
Ilkka Sjöstedt

Lahden ammattikorkeakoulu
Mediatekniikan koulutusohjelma

SJÖSTEDT, ILKKA: Yrityksen informatiivinen näytönsäästäjä

Teknisen visualisoinnin opinnäytetyö, 51 sivua, 1 liitesivu

Kevät 2007

TIIVISTELMÄ

Alun perin näytönsäästäjät olivat hyötysovelluksia, joiden tarkoitus oli pidentää monitorin käyttöikää. Tässä mielessä näytönsäästäjät ovat menettäneet merkityksensä, mutta ne voivat edelleen toimia hyötysovelluksina. Tämä opinnäytetyö käsittelee näytönsäästäjien hyödyntämistä tiedonvälityksessä. Työn tarkoituksena on tutkia periaatteita sekä menetelmiä, joiden avulla näytönsäästäjiä voidaan hyödyntää informaation välityksessä, sekä menetelmiä ja tekniikoita keskitetyn sisällönhallinnan toteuttamiseksi. Lisäksi työssä tutkitaan Flashin mahdollisuuksia näytönsäästäjäsovelluksen toteuttamisessa. Käsiteltyjä periaatteita sekä menetelmiä on sovellettu työn casena Peikko Finland Oy:lle toteutetussa näytönsäästäjätiedotusjärjestelmässä.

Työssä käsitellään erityisesti ulkoisesti tallennetun ja dynaamisen informaation hyödyntämistä näytönsäästäjässä. Ulkoisesti tallennetun informaation hyödyntämisen edellytyksenä on näytönsäästäjäsovelluksen kyky kommunikoida tietoverkon välityksellä ulkoisen sisältölähteen kanssa. Keskitetysti hallitun muuttuvan sisällön tuottamisessa voidaan hyödyntää samoja menetelmiä, kuin joilla voidaan toteuttaa dynaamisia verkkosivuja. Informaation välitykseen on syytä käyttää rakenteellista informaation tallennusmuotoa, kuten yleiskäyttöistä XML-merkkauskieltä.

Flash on varsin hyvin informatiivisen näytönsäästäjän toteutusmenetelmäksi soveltuva sovelluskehitystyökalu. Flash mahdollistaa ulkoasun, esitystekniikan sekä sovelluslogiikan toteuttamisen. Flash kykenee hakemaan informaatiota ulkoisesti tallennetuista tiedostoista tai verkko-osoitteista ja erityisesti XML-muotoon tallennetun sisällön käsittelyyn Flash tarjoaa kattavat menetelmät. Ainut merkittävä haitta Flashin käytössä näytönsäästäjän toteutuksessa on se, että Flash-sovellus vaatii jatkokäsittelyä, jotta sitä voisi hyödyntää käyttöjärjestelmän näytönsäästäjänä.

Peikko Finland Oy:lle toteutettu näytönsäästäjätiedotusjärjestelmä toteutettiin yhdistämällä Flash 8:lla toteutetut näytönsäästäjäsovellukset PHP-palvelinohjelmoinnilla toteutetun sisällön hallinta- ja jakelujärjestelmän kanssa yhtenäiseksi järjestelmäksi. Sovellusten väliseen tiedonsiirtoon käytettiin XML-merkkauskieltä. Case osoittautui onnistuneeksi ja toimivaksi järjestelmäksi, joka osoitti näytönsäästäjien avulla toteutetun tiedotusjärjestelmän olevan potentiaalinen väline yrityksen sisäisen tiedonkulun tehostamiseksi.

Avainsanat: näytönsäästäjä, informaation välitys, Flash, XML, multimedia, palvelinohjelmointi, sisällönhallinta

Lahti University of Applied Sciences
Faculty of Technology

SJÖSTEDT, ILKKA: Informative screen saver for company use

Bachelor's Thesis in Visualization Engineering, 51 pages, 1 appendix

Spring 2007

ABSTRACT

This thesis deals with the use of screen savers for distributing information. The goal of the work was to study and find out ways to make use of computer screensavers in information distribution. Another goal was to study ways to accomplish centralized content management. One of the goals was also to examine the use of the Flash authoring environment for producing an informative screensaver. These methods and principles were applied in the case part of the thesis where a screensaver serving as a news and information system was created for Peikko Finland Oy, an international manufacturer of fastening products for concrete structures.

This thesis especially deals with the use of dynamic and external information in screensavers. A key feature of informative screensavers is the ability to communicate through a network with an external data source. For managing dynamic content and creating centralized data distribution, the same methods can be used as in making dynamic Web pages. For transferring the data, a structured data format, such as XML, should be used.

Flash turned out to be a very suitable tool for creating informative screensavers. Flash offers the means for creating the appearance, the visualization and the application logic. Flash is also capable of making use of external data, and offers versatile means for using XML. The only major downside of Flash in creating a screensaver is that a screensaver application built with Flash needs further processing before it can be used as actual screen saver in operating system.

The screensaver information system created for Peikko Finland Oy was accomplished by combining the use of Flash and server-side programming. The screensaver applications were created with Flash 8, and the centralized content management and distribution system was created with PHP and MySQL. The communication between these systems was done by using XML. The case was considered to be successful and useful. The case also demonstrated that screensavers can be effectively used as a way of making distribution of information more efficient.

Keywords: screensaver, communication, Flash, XML, multimedia, server-side programming, content management

TERMIT JA LYHENTEET

Bumpmap, 3D-grafiikassa käytettävä tekniikka, jolla kappaleen pinnan muodoista saadaan yksityiskohtaisempia ilman kappaleen geometrian muuttamista.

CSS, Cascading Style Sheets, erityisesti (X)HTML:n ja XML:n yhteydessä käytettävä tyyliohje, jolla määritellään dokumentin ulkoasu.

Flash Player, Flash-kehitysympäristössä toteutettujen sovellusten ja esitysten toistamiseen vaadittava ohjelma.

Frame, yksittäinen staattinen kuva, joiden joukko muodostaa liikkuvaa kuvaa. Esityksen *Frame raten*, eli kuvanopeuden ollessa 24 fps, näkymä päivitetään 24 kertaa sekunnissa.

GD-kirjasto, palvelimella ajettava aliohjelmisto, jonka avulla kuvatiedostoille voidaan suorittaa erilaisia toimenpiteitä.

HTML, Hypertext Markup Language, Internet-sivujen muodostamiseen käytetty merkkauskieli. Variantteja mm. XHTML ja DHTML.

HTTP, Hypertext Transfer Protocol, yhteyskäytäntö, jota mm. Internet-selaimet ja Web-palvelimet käyttävät tiedonsiirtoon.

JavaScript, Netscape Communications Corporationin kehittämä pääasiassa Web-ympäristössä käytettävä komentosarjakieli.

JPEG, Joint Photographic Experts Group, häviöllistä pakkausta hyödyntävä bittikarttagrafiikan tallennusformaatti.

MySQL, SQL-tietokannan hallintajärjestelmä, joka on suosittu Web-palveluiden tietokantana.

Portaali, Internetin yhteydessä tarkoittaa verkkopalvelua, joka oman sisältönsä lisäksi tarjoaa pääsyn useisiin muihin verkkopalveluihin.

Relaatiotietokanta, tietotekniikassa käytetty termi tietovarastolle. Kokoelma tietoja, joilla on yhteys toisiinsa.

Renderöinti, kuvan luominen geometrisesta mallista tietokoneohjelman avulla.

RSS, Really Simple Syndication, Internetin uutissyötteiden muodostamiseen käytettävä tekniikka.

Skriptikieli, komentosarjakieli. Usein ohjelmaan tai sovellukseen sisällytettyjä komentoja, jotka suoritetaan ohjelman tai sovelluksen suorituksen aikana.

SOAP, Simple Object Access Protocol, XML-kieleen pohjautuva tietoliikenneprotokolla.

Syntaksi, tietotekniikassa ohjelmakoodin kielioppi.

TFT, Thin Film Transistor, nykyisin yleisin tietokoneiden monitoreissa käytetty kuvanmuodostustekniikka.

URL, Uniform Resource Locator, merkkijono, jolla määritellään tietyn tiedon sijainti.

UTF, Unicode Transformation Format, merkistöstandardi, joka kattaa suurimman osan maailman kirjoitettujen kielten käyttämistä merkeistä.

VPN, Virtual Private Network, menetelmä, jolla useampi yksityinen verkko voidaan yhdistää julkisen verkon yli.

Web-hotelli, verkkosivujen julkaisemista varten tarjottava palvelu, jossa tarjotaan asiakkaan käyttöön verkkopalvelimen resursseja sekä palveluita.

Web-palvelin, verkkoon kytketty laitealustan ja ohjelmiston yhdistelmä, joka ottaa vastaan verkon yli tulevia pyyntöjä, ja vastaa niihin palauttaen haluttua informaatiota.

WWW / Web, World Wide Web

XML, Extensible Markup Language

1 JOHDANTO

Tiedonkululla on keskeinen merkitys nykyisessä informaatioyhteiskunnassa. Erityisesti yritysmaailmassa toimiva tiedonvälitys on keskeinen osa organisoitua toimintaa. Tehokas tiedonkulku yrityksen sisäisessä organisaatiossa on haaste, johon vastaamisen merkitys kasvaa organisaation laajetessa. Tässä opinnäytetyössä käsitellään menetelmiä, joiden avulla tietokoneiden näytönsäästäjiä pystyttäisiin hyödyntämään tiedonvälityksessä. Toimintaperiaatteensa johdosta näytönsäästäjät ovat potentiaalinen väline tiedostuskanavan toteuttamiseksi, jonka on tarkoitus toimia muiden tiedonvälityskanavien ohella ja tukena. Tiedotusvälineenä toimiva näytönsäästäjä on kuin ilmoitustaulun sähköinen versio, johon huomio kiinnittyy helposti ennalta suunnittelematta.

Tämänkaltaisen sähköisen tiedotuskanavan tarjoamia mahdollisuuksia ei kannata jättää hyödyntämättä, sillä menetelmä mahdollistaa keskitetyn sisällön hallinnan, jolloin jaetun informaation ylläpitäminen ja päivittäminen on vaivatonta. Lisäksi keskitetysti ylläpidetty sisältö takaa sen, että kaikkialla yrityksen sisällä on saatavissa sama ja yhdenmukainen informaatio.

Oleellinen osa informatiivisen näytönsäästäjän toimintaa on sisällön dynaaminen päivittäminen. Tässä työssä käsitelläänkin keskeisesti sisällön keskitettyä hallintaa sekä tiedonsiirtoa keskitetyn lähteen ja näytönsäästäjien välillä. Toteutusmenetelminä työssä keskitytään käsittelemään Flashia näytönsäästäjäsovelluksen toteutuksen osalta sekä palvelinohjelmointia tiedon hallintajärjestelmän sekä jakelun osalta. Työssä käsitellään myös erityisesti XML-merkkauskieltä näiden kahden sovelluksen kommunikointivälineenä.

Käsiteltyä teoriaa hyödynnetään työn case-osuudessa, jossa toteutetaan Peikko Finland Oy:lle näytönsäästäjien välityksellä toimiva tiedotusjärjestelmä. Järjestelmän haluttiin edesauttavan tiedoteluontoisen informaation kulkua yrityksen sisällä, ja parantavan etenkin tuotannossa toimivan henkilöstön tavoitettavuutta. Lisäksi järjestelmän toivottiin osaltaan vahvistavan yrityksen yrityskuvaa sekä sisäistä imagoa.

2 NÄYTÖNSÄÄSTÄJÄ INFORMAATIOKANAVANA

Näytönsäästäjä on sovellus, joka käynnistyy, kun tietokone on yhtä mittaa käyttämättä tietyn pituisen ajan. Alun perin näytönsäästäjien tarkoituksena oli suojata näyttöä kuvan palamiselta kiinni näyttöön. Kuvan kiinni palamisena tunnettu ilmiö liittyy kuvaputkimonitoreihin johtuen tekniikasta, jolla kuva luodaan ruudulle. Ilmiö on havaittavissa etenkin vanhoissa monitoreissa, jotka suurimman osan käyttöiästään ovat näyttäneet samaa staattista kuvaa. Kuva, jota ruudulla on näytetty pitkään, saattaa olla himmeänä nähtävissä myös silloin, kun ruudulla näkyy jotakin muuta tai jopa silloin, kun näyttö on sammutettu. Tätä ongelmaa ei käytännössä esiinny TFT-tekniikkaa käyttävissä paneelinäytöissä, jotka ovat merkittävässä määrin korvanneet kuvaputkinäytöt. (Screensaver, 2007.)

Tässä suhteessa näytönsäästäjät ovat menettäneet melko pitkälti alkuperäisen tarkoituksensa. Varsinkin koska tavanomaisessa käytössä näytön sammuttaminen automaattisesti järjestelmän kautta on useassa suhteessa, mm. näytön käyttöiän ja energiankulutuksen kannalta järkevämpi ratkaisu sen sijaan, että ruudulla näytettäisiin näytönsäästäjää. (Screensaver, 2007.)

Näytönsäästäjiä kuitenkin käytetään laajamittaisesti, ja ne ovat ilmeisen suosittuja tietokonesovelluksia. Esimerkiksi Internetin suosittu hakukone Google löytää hakusanan ”screensaver” perusteella yli 40 miljoonaa osumaa (25.3.2007). Myös yleisimpien käyttöjärjestelmien mukana toimitetaan näytönsäästäjiä. Voidaankin todeta, että nykyään näytönsäästäjillä on tavallisesti enää viihteellinen tai koristeellinen painoarvo, eikä niiden käytölle useinkaan ole kovin montaa rationaalista tarvetta. Näytönsäästäjillä on kuitenkin edelleen edellytyksiä toimia hyötysovelluksina. Näytönsäästäjän avulla tietokoneen joutoaikaa voidaan valjastaa hyötykäyttöön. Tällä periaatteella onkin toteutettu useita projekteja, joissa kotikäyttäjien koneilla toimivia näytönsäästäjäsovelluksia käytetään laajamittaisiin laskentatoimituksiin. Hyvä esimerkki tällaisesta projektista on SETI@home –projekti, jossa projektiin osallistuvien henkilöiden koneiden vapaata laskenta-aikaa käytetään radioteleskoopin tallentaman datan analysointiin.

Näytönsäästäjät tarjoavat mahdollisuuden myös muunkin tyyppiseen hyötykäyttöön. Näytönsäästäjät ovat esillä, kun konetta ei käytetä. Jos näytönsäästäjä tarjoaa jotakin informaatiota, huomio sen esittämään sisältöön todennäköisesti kiinnittyy ohimennen tai suunnittelematta. Jos esimerkiksi näytönsäästäjää käytetään esittämään ajankohtaista tai usein päivittyvää sisältöä,

käyttäjän ei aktiivisesti tarvitse vahtia uutta sisältöä, vaan hän voi vaikkapa tauolta palattuaan todeta sisällön muuttuneen tai pysyneen ennallaan. Myös julkisissa, tai muissa vastaavissa tiloissa olevilla koneilla koneiden joutoaikaa voitaisiin hyödyntää tekemällä näytönsäästäjästä informaatiokanava. Kun kone ei ole aktiivisesti käytössä, se voisi toimia esimerkiksi ilmoitustauluna.

3 MENETELMIÄ INFORMATIIVISEN NÄYTÖNSÄÄSTÄJÄN TOTEUTTAMISEKSI

3.1 Edellytykset informatiiviselle näytönsäästäjälle

Jotta näytönsäästäjän avulla voitaisiin esittää informatiivista sisältöä, on sisällön käytännössä oltava dynaamista, eli päivittyvää. Näytönsäästäjään kiinteästi tallennettu informaatio ei tule kysymykseen, jos näytönsäästäjän halutaan kykenevän näyttämään ajankohtaista tai muuttuvaa informaatiota. Toisin sanoen ollakseen käytännöllinen, informatiivisen näytönsäästäjän on kyettävä hakemaan informaatiota ulkoisesta lähteestä. Tässä tapauksessa ulkoisella lähteellä tarkoitetaan informaatiota, jota ei ole ennalta tallennettu itse näytönsäästäjäsovellukseen.

Yksinkertaisimmillaan ulkoista informaatiota hyödyntävä näytönsäästäjä voi näyttää esimerkiksi kellonajan tai päivämäärän. Näitä tietoja ei tietenkään ole tallennettu itse näytönsäästäjään, vaan näytönsäästäjä hakee nämä tiedot tietokoneelta, jolla se on toiminnassa. Samaan tapaan käyttäjän tietokoneelta saataviin tietoihin perustuen olisi mahdollista toteuttaa näytönsäästäjä, joka näyttää tietoja koneen tilasta, kuten vapaan muistin määrän, suorittimen lämpötilan tai vapaan kiintolevyn osuuden. Myös käyttäjän toimiin perustuvia muuttuvia tietoja voitaisiin esittää, kuten vaikkapa listaus viimeksi avatuista tiedostoista, roskakorin sisällöstä tai Internet-selaimen sivuhistoriasta.

Mielenkiintoisemmat mahdollisuudet tarjoaa kuitenkin toteutus, jossa näytettävää informaatiota ei noudeta paikallisesti käyttäjän koneelta. Tällainen näytönsäästäjä pystyy esittämään informaatiota, joka on täysin koneen käyttäjästä riippumatonta. Ulkoisesti vaikkapa Internetin välityksellä tietoa hakeva näytönsäästäjä voi esimerkiksi näyttää uusimpia uutisotsikoita, tai seurata käyttäjän suosikkiblogeja, eli verkkopäiväkirjoja. Ulkoisia seurattavia lähteitä voi tietysti olla useita.

Tietoverkon välityksellä yhteisestä keskitetystä lähteestä näytettävän tietonsa hakevilla näytönsäästäjillä voidaan näin ollen myös toteuttaa esimerkiksi tiedotusjärjestelmä. Useille koneille asennetut näytönsäästäjät voivat hakea esitettävän informaation joko Internetin tai paikallisen verkon kautta. Kyky hakea informaatio verkon välityksellä on myös edellytys sille, että useilla eri päätteillä olevat näytönsäästäjät voivat esittää saman informaation.

3.2 Dynaamisen tiedon tuominen näytönsäästäjälle

Päivittyvän informaation siirtäminen useisiin koneisiin asennettuihin näytönsäästäjiin on järkevää toteuttaa käyttämällä verkkoon luotua kohdetta, josta päivittyvä sisältö noudetaan näytönsäästäjälle. Näytönsäästäjälle vain määritellään, mistä esitettävä informaatio löytyy.

Näytönsäästäjän ja keskitetyn informaatiolähteen välille on luotava tai määriteltävä kommunikointimenetelmä. Tällainen kahden sovelluksen välinen keskustelu voi tapahtua esimerkiksi verkko-osoitteen perusteella haettavana tiedostona, joka sisältää näytettävän informaation, ja jolla on näytönsäästäjäsovelluksen ennalta ymmärtämä rakenne.

Näytönsäästäjän päivittyvän sisällön toteuttamiseksi voidaan soveltaa samoja tekniikoita, joilla on mahdollista toteuttaa dynaamista sisältöä tarjoavia WWW-sivustoja. Menetelmiä, joilla voidaan toteuttaa esimerkiksi Internet-sivusto joka päivittyy usein, tai joka tarjoaa käyttäjän pyyntöjen mukaista sisältöä tai hakutuloksia, voidaan käyttää myös näytönsäästäjän näyttämän informaation hallintaan. Myös verkossa käytettäviä uutissyöteformaatteja on mahdollista hyödyntää näytettävän sisällön siirtämiseen. Syötteillä ei luoda varsinaisia verkkosivuja, vaan tarjotaan tietoa, joka on tarkoitettu luettavaksi erillisellä ohjelmalla tai palvelulla (What is RSS, 2007).

Molemmissa tapauksissa informaation lähteenä toimiva verkkosivu tai syöte muodostetaan palvelinsovelluksella näytönsäästäjän kutsuessa sivua tai syötettä. Kumpaa tahansa menetelmää käytettäessä palvelimella muodostetaan tekstimuotoinen tiedosto, joka tulkitaan ja muotoillaan näytönsäästäjässä. Tätä voidaan verrata esimerkiksi HTML-sivuun, joka on komentoja sisältävä tekstitiedosto. HTML-tiedosto sisältää sivun tekstisisällön ja ulkoasun määrittelyn. Selainohjelma tulkitsee nämä määrittelyt, ja muodostaa verkkosivun lopullisen ulkoasun.

Palvelimen tuottama sisältö voidaan esittää myös sellaisenaan, jos palvelin tuottaa näytönsäästäjälle valmiiksi muotoillun tekstitiedoston joka on selkeästi luettavissa. Tällöin kuitenkin menetetään merkittävä osa järjestelmän mahdollisuuksista, koska valmiina tekstinä siirrettävä materiaali ei voi sisältää määrittelyitä, joita ei ole tarkoitettu käyttäjän luettavaksi. Tällaisia määrittelyjä voi esimerkiksi olla tekstin muotoilumäärittelyt tai linkitykset kuviin.

Tämän takia on järkevää käyttää informaation siirtämiseen jotakin siirto- tai tallennusformaattia, jolla on rakenne ja joka voi sisältää sisältöön liittyviä määrityksiä. Lopullinen käyttäjälle näytettävä tieto tulkitaan ja muotoillaan vasta näytönsäästäjäsovelluksessa. Esimerkkinä tällaisista formaateista XML, HTML ja RSS. Vaihtoehtoisesti tietoa voidaan siirtää myös muuttuja-arvo -pareina.

3.3 Tiedon siirtoon hyödynnettävissä olevia tekniikoita

3.3.1 Palvelinpohjaiset verkkosivut

Dynaamisen sisällön lisääminen verkkosivulle

Perinteinen, ja nykyään jopa vanhanaikainen tapa toteuttaa Web-dokumentti on toteuttaa staattinen, eli muuttumaton verkkosivu. Tavanomainen pelkällä HTML-kielellä toteutettu Web-sivu on tyypillinen staattinen Web-dokumentti, jonka sisältö ei muutu, ellei sivun tekijä muuta sivua. Staattisia verkkosivuja voidaan aivan perustellusti käyttää sisällön julkaisemiseen, mikäli sisällön ei tarvitse päivittyä automaattisesti tai sisältöä ei tarvitse päivittää usein. (Rantala, 2002, 5.)

Staattinen verkkosivu ei kuitenkaan tule kysymykseen, kun halutaan välittää muuttuvaa ja ajankohtaista informaatiota kahden sovelluksen tai palvelun ja käyttäjän välillä. Tähän tarpeeseen tarvitaan dynaaminen verkkosivu, joka muodostetaan muuttuvan tiedon pohjalta. Tämän saavuttamiseksi on olemassa useita tekniikoita, joilla Web-dokumentti luodaan automaattisesti palvelinsovelluksella asiakassovelluksen pyyntöjen mukaisesti. Tällaisia dynaamisesti muodostettuja Web-sivuja voidaan kutsua web-sovelluksiksi, mikäli ne on luotu jotain tiettyä sovellusaluetta silmällä pitäen. (Rantala, 2002, 6.)

Webin toimintamalli

Webin toiminta perustuu asiakas-palvelin malliin, joka rakentuu yleisellä tasolla kolmesta eri osasta:

- **asiakasohjelma**, joka lähettää palvelimelle toimintapyyntöjä
- **palvelinohjelma**, joka vastaa asiakasohjelman pyyntöihin
- **yhteyskäytäntö**, eli protokolla, joka määrittelee miten asiakas- ja palvelinohjelma kommunikoivat keskenään.

(Rantala, 2005, 2.)

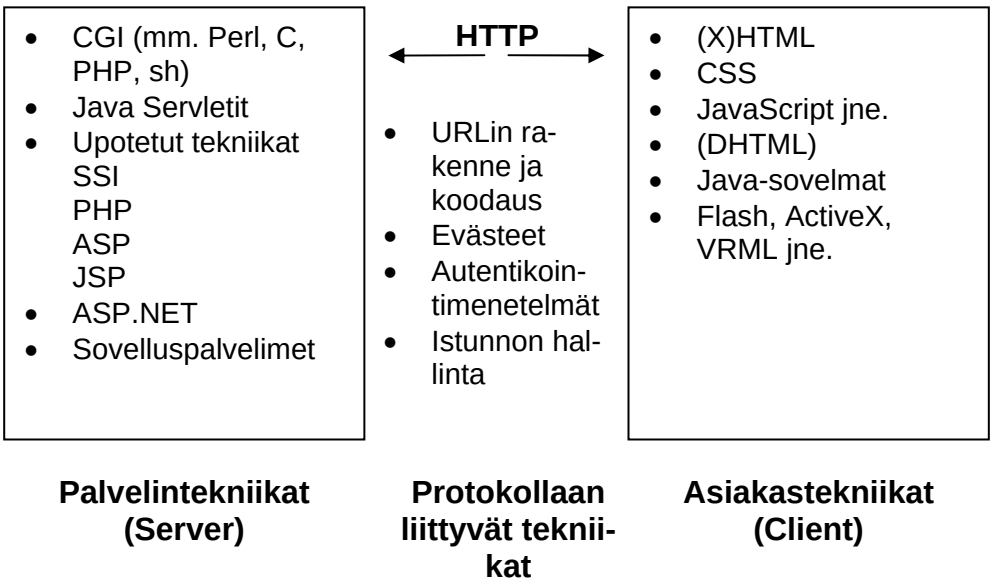
Tyypillisessä tapauksessa asiakasohjelmana toimiva Internet-selain lähettää palvelupyynnön, kuten uuden Web-sivun avaaminen, Web-palvelimelle, joka vastaa pyyntöön lähettämällä halutun Web-sivun selaimelle. Selain ja Web-palvelin kommunikoivat keskenään käyttäen HTTP-yhteyskäytäntöä. (Rantala, 2005, 2.)

Web-sovellusten toteuttamistekniikkaa

Web-sovellusten toteuttamiseksi käytettävissä olevat tekniikat voidaan vastaavasti karkeasti ryhmitellä kolmeen ryhmään:

- asiakastekniikat, jotka suoritetaan Web-selaimessa
- palvelintekniikat, jotka suoritetaan verkkopalvelimella
- HTTP-yhteyskäytäntöön liittyvät tekniikat.

(Rantala, 2005, 5.)



KUVA 1. Webin asiakas-palvelin –malli, ja toteutustekniikoita (Rantala, 2005, 2).

Asiakastekniikoihin lukeutuvat muun muassa, HTML ja XHTML, DHTML, CSS, JavaScript ja muut selaimessa suoritettavat skriptikielet ja Java-appletit. Lisäksi myös selainlaajennuksina toimivat ohjelmat, kuten ActiveX -komponentit, VRML ja Flash lukeutuvat asiakaspuolen tekniikoihin. Asiakastekniikat suoritetaan käyttäjän koneella tyypillisesti Web-selaimessa. (Rantala, 2005, 5.)

Palvelintekniikoilla tarkoitetaan menetelmiä, joilla Web-palvelimien toiminnallisuus toteutetaan ohjelmistotasolla. Palvelintekniikoihin kuuluvat muun muassa Java -tekniikat ja CGI (*Common Gateway Interface*) sisältäen tekniikat kuten Perl, C ja PHP. Myös upotetut tekniikat, kuten SSI, PHP, ASP ja JSP, ovat palvelintekniikoita, samoin kuin sovelluspalvelintekniikat, kuten esimerkiksi Macromedian ColdFusion. (Rantala, 2005, 5.)

Yhteyskäytäntöön liittyvillä tekniikoilla tarkoitetaan toimintoja, joilla palvelin- ja asiakastekniikat kommunikoivat käyttäen menetelmiä, jotka mahdollistavat tietyt toiminnot, kuten istunnon hallinnan, autentikoinnin ja evästeet. Myös URL-osoitteen rakenne ja koodaus lasketaan yhteyskäytäntöön liittyviin tekniikoihin. (Rantala, 2005, 6.)

Sovelluslogiikka

Kokonaiseen Web-sovellukseen vaaditaan kolme osa-aluetta: käyttöliittymä, sovelluslogiikka ja näiden välinen viestintä. Käyttöliittymä on nimensä mukaisesti käyttäjän liittymä, eli kommunikointimenetelmä sovelluksen kanssa. Käyttöliittymä toteutetaan Web-sovelluksissa asiakastekniikoita käyttäen. Asiakastekniikoilla toteutettu dynaamisuus Web-sovelluksessa rajoittuu pääsääntöisesti käyttöliittymän toimintoihin ja hallintaan. (Rantala, 2005, 6.)

Varsinaiseen sovelluslogiikkaan tarvitaan lähes poikkeuksetta palvelinpuolen tekniikkaa, sillä käsiteltävä informaatio sijaitsee lähes aina palvelimen tietojärjestelmissä eikä asiakassovelluksen tai käyttäjän tietokoneella. Hyvin usein käsiteltävä tieto on tallennettuna tietokantoihin. Perinteisesti Web-palveluissa käytetään relaatiotietokantoja, joita käsitellään SQL (*Structured Query Language*) -kielen avulla. Tyypillisesti Web-sovellukset joudutaan integroimaan näihin tietokantoihin. (Rantala, 2002, 10.)

Käytännössä kaikki palvelinsovellukset tarvitsevat toiminta-alustakseen palvelinpuolen infrastruktuurin, johon sisältyy muun muassa käyttöjärjestelmä-, Web-palvelin- ja tietokantapalvelinratkaisu. Verkkosivujen muodostamiseen ja verkkosovelluslogiikan toteuttamiseen käytettävät tekniikat ovat riippuvaisia tällaisesta toiminta-alustasta. (Rantala, 2002, 10.)

Varsinaisen sovelluslogiikan toteuttamiseen on saatavilla laaja joukko erilaisia palvelinpään tekniikoita. Dynaamisten verkkosivujen muodostamiseen käytettäviä tekniikoita ovat mm:

- **SSI** (*Server Side Includes*): Nimensä mukaisesti SSI:n avulla Web-sivuihin voidaan upottaa yksinkertaisia palvelinpään toimintoja. SSI:n avulla voidaan esimerkiksi upottaa useisiin verkkodokumentteihin sama, usein esiintyvä osio, kuten vaikkapa navigointipalkki. SSI:llä generoidut dokumentit tunnistaa usein tiedoston .shtml –päätteestä.
- **CGI** (*Common Gateway Interface*): CGI on yleinen liityntä Web-palvelimen ja sen suorittamien ohjelmien välillä. CGI:tä voidaan hyödyntää useilla eri ohjelmointi- tai skriptikielillä.
- **PHP** (*PHP: Hypertext Preprocessor*): Vapaa ja ilmainen skriptikieli, eli upotustekniikka, joka on saatavissa useille eri alustoille.
- **ASP** (*Active Server Pages*): ASP on lähinnä Microsoftin Web-palvelimilla käytetty upotustekniikka. ASP:n kanssa voidaan käyttää eri ohjelmointikieliä.

- **ASP.NET**: ASP.NET on ASP:n uudempi versio, joka perustuu Microsoftin .NET -sovelluskehitysympäristöön. ASP.NET -sivujen toteuttaminen muistuttaa enemmän Windows-sovelluksen tekemistä kuin tyypillistä skriptikielen käyttämistä. ASP.NET -alustalla ohjelmointikielenä voi käyttää mitä tahansa .NET -alustan kielistä.
- **Java-servletit**: Tekniikka Web-dokumenttien luomiseen Web-palvelimella dynaamisesti Java-soveltimien avulla.
- **JSP** (*JavaServer Pages*): JSP on Sun Microsystemsin kehittämä tekniikka, joka on toimintaperiaatteeltaan samantyyppinen kuin PHP. JSP-dokumentit käännetään Java-servleteiksi ennen suorittamista.

(Rantala, 2005, 7-8.)

Tyypillinen Web-sisällön toteuttamiseen tarkoitettu skriptauskieli tulkitsee HTML-dokumentin sisään upotetun ohjelmakoodin verkkopalvelimella, jolloin verkkopalvelin muodostaa verkkosivun sisällön sitä pyydettäessä (Rantala, 2002, 10). Eri skriptauskielistä tässä työssä käsitellään tarkemmin PHP-kieltä, sillä se on avoin, ilmainen ja hyvin yleisesti käytetty Web-ohjelmointikieli.

PHP

PHP on työkalu dynaamisten Web-dokumenttien luomiseen, ja se on suunniteltu ja kehitetty erityisesti Web-sovelluskehitystä varten. PHP on ohjelmointikieli, jolla upotetaan HTML-dokumenttien sisään Web-palvelimella tulkattavaa koodia. Tällaisia tulkattavia ohjelmointikieliä kutsutaan skriptikieliksi, eli komentosarjakieliksi. PHP:n syntaksi muistuttaa C-kieltä, ja se on myös sukua Perl -ohjelmointikielelle. (Rantala 2002, 12; PHP, 2007.)

PHP on alun perin Rasmus Lerdorfin kehittämä. Lerdorf aloitti kehitystyön vuonna 1994 ja julkaisi vuonna 1995 kokoelman Perl-skriptejä – tuolloin vielä nimellä ”*Personal Home Page Tools*”. Lerdorf kuitenkin pian laajensi kielen toiminnallisuutta muun muassa lisäämällä mahdollisuuden kommunikoida tietokantojen kanssa. Tässä vaiheessa kieli sai nimen PHP/FI (*Personal Home Page / Forms Interpreter*), ja Lerdorf päätti, että PHP on avoimen lähdekoodin projekti. Version PHP/FI 2.0 jälkeen vakiintui lyhenne PHP, joka on lyhenne sanoista PHP: Hypertext Preprocessor. (Rantala, 2002, 12; PHP, 2007.)

PHP:n versio 3 julkaistiin vuonna 1998, ja tämä oli ensimmäinen versio, joka muistutti merkittävästi nykyisin käytössä olevaa PHP:ta. Tämän version kehittivät Andi Gutmans ja Zeev Suraski yhteistyössä Lerdorfin kanssa. Sittemmin PHP:n ydintä on uudistettu useammankin kerran, ja PHP on tällä hetkellä versiossa 5. (PHP, 2007.)

PHP:n etuja

PHP on erityisen soveltuva tekniikka verkkosovelluksen toteuttamiseen, sillä kieli on nimenomaan suunniteltu tähän käyttötarkoitukseen. Lisäksi aloituskynnys PHP:n käyttöön on erittäin matala, sillä PHP:n perusrakenne HTML-dokumenttien sisään upotettavana skriptikielenä on yksi helpoimmin opittavista. Juuri koodin HTML-dokumenttiin sisällyttämisen ansiosta PHP:llä toteutetun sisällön määrää voi lisätä vähitellen. PHP:n kielioppi muistuttaa hyvin pitkälti C-kieltä, joka on yksi tunnetuimmista ohjelmointikielistä. PHP on myös tarkoitukseensa tehokas ja monipuolinen, ja myös hyvin suosittu. Suosionsa ansiosta PHP on laajalti tuettu, joten esimerkiksi PHP:tä tukevia Web-hotelleja ei ole vaikea löytää. Suosionsa ansiosta PHP on myös kattavasti dokumentoitu, ja Internetistä löytyy runsaasti apua PHP:n käytössä. Lisäksi PHP on avoimena ohjelmistona vapaasti saatavilla useille eri toimintaalustoille. (Rantala, 2005, 10.)

3.3.2 Uutissyötteet

Uutissyötteiden toimintaperiaate

Uutissyötteet ovat viime vuosina nopeasti yleistynyt verkkopalvelumuoto, jolla tyypillisesti tarjotaan osa Internet-sivustosta erillisesti luettavaksi, tai sisällytettäväksi muille sivustoille. Tyypillisesti uutissyötteillä tarjotaan nopea yleiskatsaus sivuston viimeisimmäksi päivitetystä tai lisätystä sisällöstä. Tällainen yleiskatsaus voi esimerkiksi olla listaus tuoreimmista uutisotsikoista, ja lyhyet kuvaukset itse uutisista. Lukuisat kaupalliset sekä yksityiset sivustot tarjoavat nykyään uutissyötteitä helpottamaan ja nopeuttamaan päivittyvän sisällön seuraamista. Uutissyötteiden avulla käyttäjä voi seurata haluamiaan verkkolähteitä yhdestä sijainnista lukijasovelluksellaan (eng. *Aggregator*). Käyttäjä näkee uuden sisällön, kun sitä on saatavilla ilman, että käyttäjän tarvitsee aktiivisesti käydä tarkistamassa, onko seurattavan sivuston sisältö päivittynyt. Lukijasovelluksen kautta seurattava sisältö on helposti nähtävissä

ilman ylimääräistä navigointitarvetta, ja käyttäjä voi nopeasti ja helposti valita artikkelit, jotka hän haluaa lukea kokonaisuudessaan. (Web syndication, 2007; Web feed, 2007.)

Uutissyötepalvelun avulla tieto uusista uutisista tai artikkeleista haetaan automaattisesti lukijasovelluksen avulla. Lukijasovellus voi olla erillinen ohjelma, tai esimerkiksi sähköpostiohjelmaan tai Internet-selaimeen asennettu laajennus. Lukijasovellus voi olla myös Webissä toimiva palvelu, jolloin seurattavien syötteiden otsikot kootaan ja esitetään verkkosivulla. Monet nykyisistä Internet-selaimista sisältävät toiminnon uutissyötteiden lukemiseen oletuksena. Lukijasovellus hakee ja näyttää vain ne syötteet, jotka käyttäjä on itse tilannut. Usein syötteenä tullut lyhyt artikkeli sisältää linkin kokonaiseen artikkeliin. (Uutissyötteet (RSS), 2007.)

Erityyppisiä uutissyöteformaatteja on esiintynyt muutamia Internetin historian aikana. Yksikään aikaisista tekniikoista ei saavuttanut suurta suosiota, ja ne olivat usein tarkoitettu palvelemaan jotakin tiettyä melko tarkkaan määriteltyä tai yksittäistä tarkoitusta. (RSS (file format), 2007.)

RSS, Really Simple Syndication

RSS:nä tunnettu uutissyöteformaatti oli ensimmäinen tekniikka, joka saavutti suuremman käyttäjäkunnan. Alun perin RSS, tällöin lyhenteenä sanoista RDF Site Summary, luotiin tekniikaksi Netscapen portaalille, ja palvelu julkaistiin vuonna 1999. Netscape kuitenkin luopui formaatin käytöstä, ja formaatti jäi vaille omistajaa. Formaatin kehitystä jatkettiin eri tahojen toimesta, ja tämän takia RSS-syötteistä esiintyy rinnakkaisia versioita. Näin ollen RRS ei ole minkään tietyn formaatin määritelmä vaan kattaa kokoelman erilaisia, mutta toisiaan muistuttavia formaatteja, joiden ominaisuuksissa on joitakin eroja. Rinnakkaisista versioista huolimatta RSS-uutissyötteet ovat muodostuneet lähes standardiksi uutissyötteiden julkaisussa. Syötteiden eri versiot ovat suurelta osin keskenään yhteensopivia, tai vähintään takautuvasti yhteensopivia. (RSS (file format), 2007.)

Rinnakkaisten versioiden esiintymisen takia RRS-lukijat eivät tavallisesti ole hirvittävän tarkkoja käytetystä versiosta, vaan ymmärtävät RSS-syötteiden eri versioita. Viimeisin ja yleiseen käyttöön suositeltavin RSS:n versioista on RSS 2.0 (Pilgrim, 18.12.2002). Käytännössä yhteensopivuusongelmia ei juuri esiinny. Teknisesti RSS on XML-kielen laji (RSS-ohje, 2006). Kaikkien

RSS-syötteiden on näin ollen oltava XML 1.0 määritelmän mukaisia (Winer, 2005). Tämä tarkoittaa myös sitä, että RSS on rakenteeltaan varsin yksinkertainen ja yksiselitteisesti tulkittavissa, mikä osaltaan auttaa välttämään yhteensopivuusongelmia RSS-syötteiden eri versioiden välillä.

3.3.3 XML

XML-kielen esittely

XML tulee sanoista Extensible Markup Language, joka voidaan kääntää suomeksi ”laajennettavissa oleva merkkauskieli” (Nykänen, 2001, 4). XML on merkkauskieli, joka on tarkoitettu rakenteisen tiedon esittämiseen. XML ei ole ohjelmointikieli, vaan XML muodostuu joukosta sääntöjä, joita käytetään tekstimuotoisten merkkauskielien suunnitteluun. XML on täysin järjestelmäriippumaton tiedon tallennus- ja esitysformaatti, ja se on myös laajennettavissa, ja tukee myös kansainvälistämistä ja lokalisointia. Näiden ominaispiirteiden ansiosta XML välttää monien muiden standardien ongelmat. Lisäksi XML on ilmainen ja täysin vapaasti käytettävissä oleva standardi. (Nykänen, 2003.)

W3C, eli World Wide Web Consortium, on organisaatio, joka kehittää yhteisiä Webin toimintaperiaatteita ja yhteensopivia Webin spesifikaatioita, ohjeita sekä teknologioita (W3C Suomen toimisto, 30.1.2007). W3C hyväksyi XML 1.0 -spesifikaation standardinomaiseksi suositukseksi yleiskäyttöiseksi tekstimuotoisten dokumenttien merkintäkieleksi helmikuussa 1998. W3C:n määritelmän mukaan XML on yksinkertainen, erittäin joustava tekstiformaatti, joka on kehitetty SGML:n (*Standard Generalized Markup Language*) pohjalta (Extensible Markup Language (XML), 2006).

XML:n kehitystyö aloitettiin vuonna 1996. Suhteellisen nuoresta iästään huolimatta XML on hyvin vakiintunut teknologia. Osaltaan tätä on edesauttanut se, että XML on kehitetty SGML:n pohjalta, joka on XML:n tapaan metakieli, joka on tarkoitettu merkkauskielien määrittelyyn. XML-kielessä on yhdistetty SGML- ja HTML-kielien hyväksi havaitut ominaisuudet, jolloin XML-kielestä on pystytty kehittämään selvästi SGML-kieltä helppokäyttöisempi ja säännöllisempi, mutta silti erittäin joustava merkkauskieli. (Nykänen, 2003.)

Vaikka XML alun perin kehitettiin vastaamaan laajamittaisen elektronisen julkaisemisen tarpeisiin, XML on osoittautunut erittäin monipuoliseksi tiedonsiirron apuvälineeksi niin Internetissä kuin muuallakin (Extensible Markup Language (XML), 2006).

Eräs merkittävä XML-kielen eduista on sen yksinkertainen ja selkeä rakenne. XML-kieltä ei ole tarkoitettu luettavaksi sellaisenaan, mutta yksinkertaisen rakenteensa ansiosta XML-kielellä toteutettu dokumentti on varsin helposti ymmärrettävissä. XML-dokumentissa tieto on aina kuvattu tekstimuodossa, joka on jäsennelty sisäkkäisistä elementeistä koostuvaan runkoon. Lisäksi XML muistuttaa ulkoasultaan yleisesti tunnettua HTML-kieltä. Vaikka XML-muotoon tallennetun tiedon tulkinta on tarkoitettu sitä hyödyntävien sovellusten tehtäväksi, ei XML-muotoisen tiedon tulkintaan tarvita välttämättä erillistä sovellusta, vaan myös ihminen voi lukea ja ymmärtää XML-dokumentin sisällön. (Hermans, North, 2001, 14-15.)

XML:n sääntöjä ja kielioppia

Kuten edellä on todettu, XML muistuttaa ulkoasultaan HTML-kieltä, ja yhteisen taustansa ansiosta kielillä on paljon yhteisiä piirteitä. XML- ja HTML-kieliä yhdistäviä piirteitä on mm. perusidea, jolla kielet merkataan, sekä kielien syntaksi, eli kielioppi. (Nykänen, 2001, 4.) Aivan kuten HTML-dokumentit, myös XML-dokumentit koostuvat elementeistä, jotka alkavat alkutunnisteesta ja loppuvat lopputunnisteeseen. Nämä tunnisteet merkataan <>-merkkien sisään, ja elementin sisältö sijoitetaan alku- ja lopputunnisteen väliin. (Holzner, 2001, 23.)

Ohessa esimerkkinä yksinkertainen XML-dokumnetti, joka määrittelee lyhyen esimerkin:

```
<?xml version="1.0" encoding="UTF-8"?>
<esimerkki>
  <elementti attribuutti="arvo">
    <otsikko>Tervehdys</otsikko>
    <viesti>Tämä on esimerkki yksinkertaisesta XML-
dokumentista</viesti>
  </elementti>
  <elementti attribuutti="eriarvo">
    <otsikko>Perustietoa</otsikko>
    <viesti>Elementit voidaan vapaasti määritellä
kulloisenkin tarpeen mukaan</viesti>
  </elementti>
</esimerkki>
```

Esimerkkidokumentti rakentuu yhden juurielementin, `<esimerkki>`, sisään. Juurielementti sisältää muut dokumentin elementit, esimerkissä `<elementti>`, jotka puolestaan sisältävät alaelementit `<otsikko>` ja `<viesti>`. Elementin sisältö tai ominaisuudet voidaan määritellä myös elementin attribuutteina.

Toisin kuin HTML, XML ei sisällä ennalta määriteltyjä elementtejä. HTML on tarkoitettu kuvaamaan Internet-sivuja, jolloin on perusteltua käyttää ennalta määriteltyjä elementtejä, jotta sivut kuvattaisiin yhteneväisellä merkkaustavalla. XML-kieltä ei sen sijaan ole räätälöity mihinkään tiettyyn tarkoitukseen, vaan XML on metakieli, jolla määritellään merkkauskieliä. XML-kielellä kuvataan tiedon rakenne ilman ennalta määriteltyjä elementtejä. XML-kielen avulla muodostetaan tarkoituksenmukaiset elementit, joiden avulla varsinainen tieto kuvataan. (XML, 2007.)

HTML-kieli ei ole kovinkaan tarkka kielen oikeellisuudesta. Itse asiassa HTML-kielessä on hyvin tyypillistä kielen säännöistä lipsuminen. Internet-selaimet kykenevät tulkitsemaan kieltä, vaikka dokumentin muodostamisessa olisikin otettu hieman vapauksia. Osaltaan myös itse selaimet ovat aiheuttaneet rönsyilyä kielessä laajentamalla norminmukaista HTML-kieltä omilla selainkohtaisilla käskyillään ja toiminnoillaan. (Nykänen, 2001, 6.) Koska HTML-kielessä on ennalta määritelty joukkio elementtejä, joiden käyttäytyminen ja tarkoitus ovat tiedossa, selaimet voivat tehdä olettamuksia kompensoimaan dokumentin puutteellista rakennetta. Web-selain esimerkiksi ei useinkaan välitä, vaikka joltakin elementiltä puuttuisi lopetustunniste, tai jos tunnisteen attribuuttia ei ole merkitty kieliopin mukaisesti lainausmerkkeihin. (Statler, 2002.)

XML-kieli puolestaan on tarkka kieliasusta. XML-kielen sääntöjen mukaan toimivan XML-tulkin kuuluu tarkistaa XML-dokumentin oikeellisuus. Jos XML-dokumentin rakenne ei noudata XML-standardia, kuuluu XML-tulkin antaa virheilmoitus sen sijaan, että näyttäisi virheellisesti muodostetun dokumentin. XML-tulkki voi suorittaa dokumentille kaksi päätarkistusta: onko dokumentti hyvinmuodostettu, ja onko se validi. (Holzner, 2001, 29.)

Vaatimukset hyvinmuodostetulle XML-dokumentille ovat seuraavanlaiset:

- Dokumentti täytyy seurata W3C:n XML 1.0 -määrittelyssä XML:lle asettamia hyvinmuodostetun dokumentin rajoituksia.

- Dokumentti sisältää vain yhden juurielementin, jonka sisällä kaikki muut elementit esiintyvät.
- Elementit voivat olla sisäkkäisiä, mutta ne eivät mene ristiin toisten elementtien kanssa, eli elementit on lopetettava aina samalla tasolla kuin millä ne aloitetaan.
- Jokainen jäsennetty entiteetti, johon dokumentissa viitataan, on hyvin muodostunut.

(Laakkonen, Walkama, 2004, 4; XML, 2007.)

Vaikka XML-dokumentti noudattaisikin näitä sääntöjä, se ei silti välttämättä ole validi, eli rakennekuvausten mukainen. Jotta XML-dokumentti olisi myös validi, on sen täytettävä kaikki seuraavat ehdot:

- Dokumentti on hyvinmuodostettu.
- Dokumenttiin on liitetty rakennekuvaus.
- Dokumentti noudattaa rakennekuvausta ja muita XML-määrittelyssä asetettuja validiusrajoituksia.

(Laakkonen, Walkama, 2004, 4.)

Rakennekuvaus tarkoittaa XML-dokumentin määritelmää, jota dokumentin tulee noudattaa. Rakennekuvaus määrittelee esimerkiksi käytettävät elementit, attribuutit ja näiden sisällön sekä dokumentin nimiavaruuden ja rakenteen asettaen rajoituksia näiden muodostamiselle ja esiintymiselle. Käytännössä rakennekuvaus on XML-dokumentin tulkinta- ja muodostussäännöt. Rakennekuvaus kertoo XML-dokumenttia käsittelevälle sovellukselle, miten sisältöä tulisi käsitellä. XML-dokumenttiin ei ole pakko liittää rakennekuvausta, mutta ollakseen täysin itsenäinen ja alusta- tai sovellusriippumaton, XML-dokumentti tarvitsee rinnalleen rakennekuvauksen, jotta se pystyttäisiin tulkitsemaan yksiselitteisesti missä yhteydessä tahansa. (Laakkonen, Walkama, 2004, 3-4.)

XML-dokumenttien tuottaminen ja muokkaaminen

Useimmissa tapauksissa XML esiintyy sovelluksissa taustalla käyttäjältä näkymättömissä. XML ei ole tarkoitettukaan sellaisenaan käyttäjän käsiteltäväksi, vaan se on paremminkin sovellusten käyttöön tarkoitettu väline.

XML-dokumentteja on kuitenkin varsin helppo luoda ja käsitellä, ja tietyissä tilanteissa XML-dokumenttien käsittely sellaisenaan saattaa olla tarpeellista. Etenkin XML-merkkausta hyödyntävää sovellusta kehitettäessä on varmasti tarpeen muokata itse XML-dokumentteja.

XML-dokumentin käsittelyyn riittää tarvittaessa pelkkä tekstieditori ja perustiedot XML-kielen rakenteesta, sillä XML-dokumentit ovat tekstimuotoisia dokumentteja. XML-dokumenttien käyttökelpoisuus kuitenkin perustuu pitkälti dokumenttien kieliopilliseen oikeellisuuteen, joten ilman automatisointia luotujen XML-dokumenttien oikeellisuus tulisi tarkistaa jollakin XML-kieltä tulkkaavalla ohjelmalla, jolloin dokumentin mahdolliset virheet paljastuvat. Kaikkein yleisin XML-tulkkaukseen kykenevä sovellus lienee Microsoftin Internet Explorer. (Nykänen, 2001, 52-54.)

XML-dokumenttien luomiseen ja muokkaamiseen on saatavilla myös erilaisia tähän tarkoitukseen toteutettuja tai soveltuvia editoreja. Riippuen editorin ominaisuuksista, XML-dokumenttien muokkaamiseen tarkoitetuissa editoreissa on lukuisia etuja tavanomaisiin tekstieditoreihin nähden. Mahdollisia ja tyypillisiä hyödyllisiä ominaisuuksia ovat mm.

- merkkauksen ja elementtien korostus värimerkinnöin
- eri merkkikoodauksien tukeminen
- kieliopin tarkistus dokumenttia kirjoitettaessa
- dokumentin rakenteen esittäminen puu-rakenteena

Nämä apuvälineet auttavat dokumentin rakenteen hahmottamisessa ja auttavat takaamaan dokumentin oikeaoppisen rakenteen. (Nykänen, 2001, 57-59.)

4 FLASHIN KÄYTTÖ NÄYTÖNSÄÄSTÄJÄN TOTEUTTAMISEEN

4.1 Flashin ominaisuudet sovelluskehitysympäristönä

Macromedian kehittämä, ja nykyään Adoben omistuksessa oleva Flash mahdollistaa tarvittavien menetelmien käyttämisen informatiivisen näytönsäästäjän toteuttamiseksi tarjoten kuitenkin samalla melko helposti lähestyttävän kehitysympäristön.

Flash on alun perinkin ollut multimediaesitysten kehittämiseen suunnattu ohjelmisto, ja tässä suhteessa Flash palvelee näytönsäästäjän toteuttamista erin-

omaisesti. Nykyään Flash sisältää myös monipuoliset mahdollisuudet ulkoisen informaation hyödyntämiseen. Flash tarjoaa monipuolisia menetelmiä, joiden avulla voidaan siirtää tietoa Flash-sovelluksen ja verkkopalvelun välillä, joko Flashin ActionScript -ohjelmointikieltä tai datakomponentteja hyödyntäen (Manninen, Marttila, 2006, 383). Nykyisin Flash sisältää datakomponentteja, joiden avulla Flash-sovelluksessa voidaan helposti käsitellä sovelluksen ulkopuolista tietoa (Manninen, Marttila, 2006, 383). Lisäksi Flash sisältää erityisesti XML-merkintäkielen käsittelyyn tarkoitettuja luokkia, jotka helpottavat merkittävästi XML-muotoisen tiedon hyödyntämistä Flash-esityksen yhteydessä.

Nämä seikat tekevät Flashista erittäin käyttökelpoisen välineen dynaamisen informaation näyttämiseen kykenevän näytönsäästäjän toteuttamiseksi. Flashia hyödyntämällä käytössä on menetelmät ulkoasun, esitystekniikan sekä dynaamisen sisällön integroimisen toteuttamiseksi.

4.2 Flashin multimediaominaisuuksien hyödyntäminen

Flash on etenkin Webissä käytettävän multimedian julkaisemiseen kehitetty ohjelmisto. Adobe ilmoittaa Flashin olevan alan kehittynein kehitysympäristö interaktiivisten verkkosivujen, digitaalisten kokemusten ja mobiilisisällön tuottamiseen. Flash on työkalu, jonka avulla pystytään tuottamaan vuorovaikutteista mediaa sisältäen runsaasti videota, grafiikkaa ja animaatiota. (Flash Professional 8: Datasheet, 2007.)

Ulkoasullisesti rikkaan ja näyttävän sovelluksen toteuttamisvälineenä Flash on varsin hyvin myös näytönsäästäjän ulkoasun toteutukseen soveltuva työkalu. Näytönsäästäjien ei myöskään tulisi olla luonteeltaan täysin staattisia, vaan näytönsäästäjän tulisi koostua pääsääntöisesti joko liikkuvista tai sijaintiaan vaihtavista elementeistä. Flash mahdollistaa tarvittaessa melko helposti mielenkiintoisen ja elävän ulkoasun toteuttamisen. Flashin tarjoaminen ominaisuuksien käyttö kannattaa kuitenkin pitää maltillisena, sillä ylenpalttinen huomionhakuisuus näytönsäästäjässä ei välttämättä ole eduksi.

Harkitsemisen arvoisia multimediaominaisuuksien käyttötapoja tietosisältöä tarjoavien näytönsäästäjien kohdalla voisi olla esimerkiksi rauhallisesti liikkuvat taustat tai kuvat ja sijaintiaan ajoittain muuttavat elementit. Flash mahdollistaisi myös äänen käyttämisen, mutta äänien hyödyntämistä näytönsääs-

täjässä kannattanee harkita tarkkaan. Liiallinen ominaisuuksien käyttö ja temppuilu useimmiten muodostuvat helposti ärsyttäviksi ominaisuuksiksi.

4.3 Ulkoisen sisällön tuominen Flashiin

4.3.1 Tiedonsiirto Flash-sovelluksen ja verkkopalvelun välillä

Flash-sovelluksen taustalla voi toimia useita käyttäjälle näkymättömiä verkkotekniikoita, joiden avulla Flash-sovelluksesta pystytään tekemään dynaaminen, eli automaattisesti sisältöään muuttava (Manninen, Marttila, 2006, 378). Tällaisista Flash-sovelluksista jotka näyttävät ja muokkaavat ulkoiseen lähteeseen tallennettua sisältöä, Macromedia käyttää nimitystä RIA, Rich Internet Applications. Termillä viitataan mihin tahansa Flash-sovellukseen, joka esittää ulkoisesti tallennettua informaatiota graafisesti näyttävissä puitteissa. (What you can do with Flash, 2007.)

Laajemmissa Flash-sovelluksissa ulkoisesti tallennetun sisällön hallinta korostuu. Verkon välityksellä toimivassa Flash-sovelluksessa tiedonsiirtoon kuluvaa aikaa voidaan säästää hakemalla haluttu mediasisältö vasta sitä tarvittaessa, sen sijaan että koko sisältö noudettaisiin yhdellä kertaa. Ulkoisesti tallennettu sisältö myös helpottaa sisällön päivittämistä. Tekstimuotoinen sisältö voidaan ulkoistaa tekstipohjaisiin tiedostoihin, jolloin se voidaan ladata sovellukseen Flashin tarjoamilla menetelmillä. Tekstimuotoisia tiedostoja kannattaa käyttää, jos tiedon määrä on pieni. Suurempia tietomääriä siirrettäessä kannattaa käyttää apuna tietokantoja. Tällöin joudutaan käyttämään hyödyksi palvelinpuolen ohjelmointia, sillä Flash-sovellus ei sellaisenaan kykene lataamaan tietoa tietokannoista. (Manninen, Marttila, 2006, 381-382.)

Tietoa voidaan siirtää Flash-sovelluksen ja verkkopalvelun välillä usealla eri tavalla. Vuonna 1999 Flashin versiossa 4 esiteltiin LoadVariables -metodi, joka mahdollisti tiedon välittämisen verkkopalvelusta Flash-sovellukselle. Nykyään Flash sisältää monipuolisen joukon erilaisia luokkia ja komponentteja, jotka ovat tarkoitettu Flash-sovelluksen ja verkkopalvelun väliseen kommunikointiin. Flash versio 8 esitteli myös mahdollisuuden siirtää tiedostoja Flash-sovelluksen ja verkkopalvelimen välillä. (Manninen, Marttila, 2006, 383.)

Flash käyttää tiedon välitykseen HTTP-protokollaa, ja sen POST- ja GET-metodeja. GET-metodi välittää siirrettävän tiedon kohteeseen viittaavan pal-

velun verkko-osoitteen yhteydessä. GET-metodi on luonteeltaan avoin, eli lähetettävä tieto on helposti saatavissa verkko-osoitteesta. POST-metodia käytettäessä välitettävä tieto liitetään HTTP-header-osioon, ja se on näin ollen turvallisempi menetelmä verkkosovelluksen ja -palvelun väliseen kommunikointiin. Flash Player käyttää oletuksena POST-metodia tiedon siirtämiseen käytettäessä esimerkiksi LoadVars ja XML-luokkia. (Manninen, Marttila, 2006, 383.)

4.3.2 Muuttuja-arvo -parit

Yleiset periaatteet

Ehkäpä suoraviivaisin menetelmä välittää ulkoista informaatiota Flash-sovellukselle on sijoittaa tieto muuttuja-arvo -pareihin. Flash pystyy lataamaan informaatiota merkkijonoina, jotka muodostuvat muuttuja-arvo -pareista. Tekstimuotoinen merkkijono voi sisältää useita muuttuja-arvo -pareja, jotka kuitenkin on erotettava toisistaan ”&“-merkillä. Esimerkiksi henkilön perustiedot voitaisiin esittää muuttujien *etunimi*, *sukunimi*, *pituus* ja *paino* avulla:

```
etunimi=Jukka&sukunimi=Palmu&pituus=178&paino=69
```

Muuttuja-arvo -parit ovat erotettu toisistaan ”&”-merkeillä, ja jokaiselle muuttujalle on annettu arvo, ja koko lauseke on kirjoitettu yhtenäisesti. Flash voi hakea muuttuja-arvo -parit sisältävän merkkijonon esimerkiksi tekstitiedostosta. Muuttujien määrää ei ole rajoitettu, vaan tällä menetelmällä voidaan välittää lukuisia muuttujia arvoineen Flash-sovellukselle. Esimerkin mukainen tekstitiedosto voitaisiin kirjoittaa käsin, tai se voidaan muodostaa palvelinsovelluksella. (Solis, 2007.)

Flash-sovellukselle voidaan vaihtoehtoisesti myös lähettää muuttujia URL-osoitteen yhteydessä. Tällöin Flash-sovelluksen täytyy sijaita Internet-sivulla, ja sivu on muodostettava dynaamisesti palvelinsovelluksella. (How to transfer variables from url string to Flash movie, 2006.) Tällöin on myös otettava huomioon URL-osoitteen pituus. Suurin sallittu pituus riippuu käytettävästä selaimesta tai palvelinohjelmistosta. Esimerkiksi Internet Exploreria käytettäessä URL-osoitteen pituus voi maksimissaan olla 2048 merkkiä (support.microsoft.com, 2006).

Lisäksi Flash asettaa joitakin rajoituksia käytettävissä olevien merkkien suhteen ladattaessa tietoa muuttujien avulla. On tiettyjä merkkejä, joita ei voida lukea suoraan tekstinä, ja jotka aiheuttavat virheitä tai ei-toivottuja tuloksia. Tällaisia merkkejä voidaan kuitenkin käyttää, kun ne muutetaan verkko-osoitteista tuttuun URL-koodattuun muotoon, jolloin ne ovat myös Flashin kannalta yksiselitteisiä. (URL Encoding: Reading special characters from a text file, 2005.)

LoadVars

LoadVars-luokan avulla voidaan ladata informaatiota ulkoisesta tiedostosta tai lähettää ja vastaanottaa tietoa Flash-sovelluksen ja verkkopalvelinsovelluksen välillä. Tällainen mainittu verkkopalvelinsovellus voi esimerkiksi olla palvelimella käytössä oleva ohjelmointikieli tai palvelinpohjainen verkkosivu, joka palauttaa pyyntöä vastaavan tuloksen. Ulkoista tietoa Flash-sovellukselle voidaan siirtää palvelimella sijaitsevasta tiedostosta tai palvelimella toimivan palvelun kautta. LoadVars-metodin kohde määritellään URL-osoitteen avulla. Jotta Flash pystyisi hyödyntämään tällä menetelmällä haettua tietoa, täytyy kohteena olevan tiedoston sisältää muuttuja-arvo –pareja, jotka ladataan ja sisällytetään pyynnön tehneeseen LoadVars-objektiin omina ominaisuuksinaan. Samoin myös palvelinohjelman ollessa kohteena, palvelinohjelman on palautettava Flash-sovellukselle muuttuja-arvo –pareja. (Manninen, Marttila, 2006, 384.)

Sen lisäksi, että LoadVars-luokalla voidaan hakea Flash-sovellukseen ulkoista tietoa, voidaan sen avulla myös lähettää muuttujia palvelinsovellukselle luokan `send()` ja `sendAndLoad()` -metodeilla. Metodit eroavat toisistaan käytännössä siinä suhteessa, että send -metodia käytettäessä tiedon lähettämisen onnistumisesta ei saada mitään tietoa. SendAndLoad -metodia käytettäessä palvelinohjelmisto voi palauttaa Flash-sovellukselle tietoa, jonka perusteella voidaan päätellä lähetyksen onnistuminen. (Manninen, Marttila, 2006, 385.)

4.3.4 Flash Remoting

Flash Remoting on Macromedian kehittämä tekniikka, joka mahdollistaa palvelinpuolen tekniikan hyödyntämisen suoraan Flash-sovelluksen kautta. Tekniikka mahdollistaa esimerkiksi tietokantayhteydet, sähköpostin lähettämisen

ja tiedostojen lukemisen ja kirjoittamisen. Menetelmää käytettäessä tietoliikenne tapahtuu binäärisessä muodossa HTTP-protokollan yli Macromedian määrittelemässä AMF-formaatissa (*Action Message Format*). Flash Remoting -tekniikkaa käytettäessä tietoa ei tarvitse sijoittaa muuttuja-arvo -pareihin, ja erona LoadVars- ja XML-objekteihin myös monimutkaisia tietotyyppejä, kuten objekteja ja taulukoita, voidaan siirtää suoraan tekniikan käyttämän yhteyssillan yli. Myös XML-muotoista dataa voidaan siirtää AMF-formaatissa Flash Remoting -yhteyden yli. (Manninen, Marttila, 2006, 397.)

Flash Remoting -yhteys tarvitsee toimiakseen tuen palvelimelta. Jotta palvelin kykenisi lähettämään Flash-sovellukselle AMF-muotoista dataa, on palvelimena käytettävä Macromedian ColdFusion -palvelinta, tai vaihtoehtoisesti palvelimella toimivalle ympäristölle on asennettava laajennus, joka lisää palvelimelle AMF-tuen. AMF-tuki on saatavissa laajennuksena ainakin Java-, Microsoft .NET- sekä myös muihin SOAP-pohjaisiin palveluihin. On myös olemassa ilmaisia projekteja, joiden avulla AMF-tuki voidaan toteuttaa, kuten AMFPHP. Toimiakseen myös Flash-sovellus tarvitsee Remoting-komponentit. (Solis, 2007; Manninen, Marttila, 2006, 397.)

4.4 XML:n käyttö Flashin kanssa

4.4.1 Yleistä Flashin XML-ominaisuuksista

XML-kieltä voidaan hyödyntää Flash-sovelluksissa sekä ulkopuolisen tiedon lataamisessa että tiedon välittämisessä Flash-sovelluksen ja palvelimen välillä. (Manninen, Marttila, 2006, 387.)

XML-muotoinen dokumentti on helppo ja käytännöllinen tapa tallentaa rakenteellista informaatiota. XML on erityisesti käytännöllinen vaihtoehto tietokannoille, varsinkin jos tietoa ei ole kovin suuria määriä, ja sitä on ainoastaan tarkoitus lukea Flash-sovelluksen kautta. XML:n käyttö Flashin yhteydessä on perusteltua myös sen ansiosta, että Flashiin on sisällytetty monipuolinen tuki XML-dokumenteille XML-objektien avulla. (Solis, 2007.)

Flash-sovelluksen kanssa käytettäessä XML-dokumentin rakennekuvaus ei ole välttämätön, etenkin jos XML-sisältöä ei ole tarkoitus käyttää muissa sovelluksissa. Flashin kannalta riittää, että XML-dokumentti on hyvinmuodostettu ja että sen rakenne vastaa oletusta, jolla Flash-sovellus dokumenttia käsittelee. (Solis, 14.1.2007.)

4.4.2 Tiedon lataaminen Flash-sovellukseen ulkoisesta XML-tiedostosta

XML-muotoon tallennettu tieto voidaan ladata Flash-sovellukseen XML-luokan `load()` -metodilla. Metodille määritellään kohteena olevan XML-dokumentin sijainti URL-osoitteen avulla. Flash Player aloittaa kohteen lataamisen heti `load()` -komentoa kutsuttaessa, ja sovellus suorittaa `onLoad()` -tapahtuman heti, kun ladattava XML-dokumentti on käytettävissä. Flash Player suorittaa LoadVars-objektin onLoad-tapahtuman automaattisesti heti, kun kaikki muuttujat on ladattu, jolloin ohjelman ei tarvitse erikseen tarkkailla, milloin haettava data on käytettävissä. OnLoad-tapahtuma tulee käsitellä omana funktionaan, jolle välitetään tieto latauksen onnistumisesta Boolean-tyyppisenä muuttujana. (Manninen, Marttila, 2006, 384, 388.)

Oheinen lyhyt esimerkkikoodi lataa ”content.xml” -nimisen XML-tiedoston, ja tulostaa sen sisällön Flashin Output -ikkunaan:

```
var myXML:XML = new XML();
myXML.ignoreWhite = true;
myXML.load("content.xml");
myXML.onLoad = function(succes:Boolean){
          if(succes) trace(this);
          else trace("Virhe XML-dokumentin
lataamisessa!")
}
```

Usein XML-dokumentti muotoillaan siten, että sen elementteihin ja niiden väliin jätetään välilyöntejä, jotta XML-dokumenttia olisi helpompi tulkita sellaisenaan. Varsinkin sisäkkäisten elementtien sisentäminen selkeyttää XML-dokumentin rakennetta huomattavasti. Oletusarvoisesti Flash Player käsittelee välimerkit XML-solmuina, mikä saattaa aiheuttaa virheitä XML-dokumentin tulkkausprosessissa. Tämän takia onkin suositeltavaa, että Flash Player komennetaan jättämään tyhjät tekstisolmut huomioimatta käyttäen XML-objektin ignoreWhite-ominaisuutta, asettamalla sille arvon ”true”, kuten edellisessä esimerkissä. (Manninen, Marttila, 2006, 388-389.)

Ongelmaksi saattavat muodostua myös skandinaaviset merkit, jotka voivat aiheuttaa sen, että XML-dokumentista ei saada ladattua tietoa, joka sisältää näitä merkkejä. Ongelma voidaan välttää esimerkiksi tallentamalla dokumentti käyttäen UTF-8 -koodausta, ja määrittelemällä XML-dokumentti vastaamaan kyseistä koodausta. Näin Flash osaa ladata erikoisia merkkejä sisältävän tiedon oikein XML-objektiin. (Manninen, Marttila, 2006, 389.)

```
<?xml version=”1.0” encoding=”UTF-8”?>
```

Vaihtoehtoisesti System.useCodepage -ominaisuuden avulla Flash voidaan asettaa käsittelemään dokumenttia laitteiston oletuksena olevalla merkistöllä. Tällöin dokumentin sisältö toimii ja näkyy oikein, jos sovellusta ajetaan samaa merkistöä käyttävällä järjestelmällä kuin millä lähdetiedosto on luotu. Tätä menetelmää tulisi käyttää ainoastaan, jos sovelluksen käyttökohteet on rajattu ennalta tiedetylle alueelle, jolla on käytössä yhtenevä merkistö. Varmin tapa saada merkistöt toimimaan suunnitellulla tavalla on käyttää joko UTF-8 tai UTF-16 -koodausta. (Manninen, Marttila, 2006, 389.)

Flashin XML-luokka mahdollistaa myös XML-muotoisen tiedon lähettämisen. XML-muotoisen tiedon lähettämiseen on käytettävissä XML-luokan `send()`- ja `sendAndLoad()` -metodit. Molemmat metodit muuttavat XML-objektin sisällön automaattisesti XML-dokumentiksi, ja lähettävät sen parametrina määriteltyyn URL-osoitteeseen. Samoin kuin LoadVars-luokan vastaavilla metodeilla, send -metodia käytettäessä lähetyksen onnistumisesta ei saada mitään tietoa, vaan Flash-solvellus sokeasti lähettää XML-muotoisen datan määriteltyyn kohteeseen. SendAndLoad -metodille määritellään toisena parametrina XML-objekti, jolle kohteena olevasta URL-osoitteesta voidaan lähettää paluuviestinä XML-muotoista dataa, jonka perusteella lähetyksen onnistuminen voidaan varmistaa. (Manninen, Marttila, 2006, 390.)

4.5 Näytönsäästäjän tekeminen Flash-sovelluksesta

Informatiivisen näytönsäästäjän toteutukseen Flash on ominaisuuksiensa puolesta erinomainen väline, sillä se tarjoaa menetelmät näytönsäästäjäsovelluksen toiminnallisuuden sekä ulkoasun luomiseen. Flash-sovelluksia ei sellaisenaan voi kuitenkaan käyttää näytönsäästäjänä, vaan ne on muutettava käyttöjärjestelmän hyväksymiksi näytönsäästäjäsovelluksiksi.

Windowsin tapauksessa näytönsäästäjät ovat tavallisia Windows-sovelluksia, joille on annettu oma tiedostopäätteensä ’.scr’ (Popescu, 2005). Näin ollen Flash-sovellus tarvitsee rinnalleen näytönsäästäjänä toimivan Windows-sovelluksen, joka toimii myös Flash Playerinä. Flash-ohjelma pystyy julkaisemaan Flash-esityksen Windows-sovelluksena, joka sisältää myös Flash Playerin. Tällä menetelmällä toteutetut sovellukset eivät kuitenkaan sellaisenaan ole käyttökelpoisia näytönsäästäjiä, vaikka käynnistyvätkin näytönsäästäjänä uudelleennimeämisen jälkeen. Näytönsäästäjän keskeiset ominaisuu-

det, kuten koko näytölle avautuminen ja sovelluksen sulkeutuminen hiirtä liikutettaessa, ovat itse näytönsäästäjäsovelluksen ominaisuuksia – Windows ainoastaan käynnistää näytönsäästäjäksi määritellyn sovelluksen.

Windows-ohjelmoititaitoiset voivat ohjelmoida Flash-esityksen käynnistävän näytönsäästäjäsovelluksen itse, mutta ohjelmointitaidottomat joutunevat turvautumaan jonkinlaiseen apuohjelmaan, joka tekee muunnoksen näytönsäästäjäksi joko suoraan SWF-tiedostosta tai itsenäisestä Flash Playerin sisältävästä Flash-sovelluksesta. Tähän kykeneviä apuohjelmia on saatavissa erittäin runsaasti.

5 KESKITETYN INFORMAATION JAKELU JA HALLINTA

5.1 Toimintaympäristö

Informaatiokanavana toimiva näytönsäästäjäjärjestelmä tarvitsee ulkoisen lähteen esitettävälle sisällölle. Mikäli näytönsäästäjät toimivat yrityksen sisäisenä tiedotuskanavana, vaativat ne rinnalleen keskitetyn järjestelmän sisällön hallintaan ja jakeluun. Tällainen järjestelmä voidaan toteuttaa luomalla verkossa toimiva palvelu, jonka kautta sisältöä sekä jaellaan että hallitaan.

Keskitetyn tiedonjakelun toimintaympäristön määrittelee pitkälti tiedonjakelukanavan käyttötarkoitus. Jos sisältöä, kuten vaikkapa maailman uutistapahtumia, halutaan välittää avoimesti hyvin laajalle yleisölle, on keskitetty tiedonjakelu käytännössä perustettava Internetiin. Internet on kansainvälinen ja avoin tietoverkko, jolloin sen välityksellä yhteen ja samaan tietolähteeseen on käytännössä rajoittamaton pääsy.

Jos informaation saatavuutta halutaan rajoittaa, Internet ei välttämättä ole paras mahdollinen jakelukanava. Jos Internetin välityksellä jaettavaan informaatioon pääsyä halutaan rajata tietylle alueelle tai käyttäjäryhmälle, on käyttäjät, joille sisältö halutaan olevan saatavilla, kyettävä tunnistamaan jollakin keinolla.

Mikäli informaation jakelu tapahtuu hyvin paikallisesti ja rajatulla toiminta-alueella, kuten esimerkiksi yrityksen sisällä, ei Internetiä kannata käyttää sisällön jakeluun lainkaan, ellei sille ole mitään erityistä tarvetta. Tällaisessa tapauksessa sisällön jakelun ja hallinnan toteuttaminen yrityksen sisäisessä verkossa on todennäköisesti järkevin ratkaisu. Käytettäessä sisäistä verkkoa

jakelukanavana on jaettava informaatio myös automaattisesti saavuttamattomissa yrityksen verkon toimialueen ulkopuolisille käyttäjille. Tarvittaessa sisäiseen verkkoon voidaan kuitenkin päästä käsiksi esimerkiksi suojatun VPN-etäyhteyden kautta.

5.2 Hallinta- ja jakelujärjestelmä

Toimintaympäristöstä riippumatta keskitetty sisällön jakelu vaatii järjestelmän, jonka kautta sisältöä voidaan organisoida sekä tuoda se saataville. Fyysisesti jaeltava informaatio sijaitsee verkkoon kytketyllä tietokoneella, eli palvelimella, joka myös tarjoaa sisällön verkkoon. Sisällön hallinta ja jakelu ovat tiiviisti yhteydessä toisiinsa, sillä molemmat toiminnot käsittelevät samaa informaatiota. Tämän takia on suositeltavaa, että sisällön käsittelyyn tarvittavat sovellukset toimivat samassa sijainnissa, kuin mihin sisältö on myös tallennettu. Näin ollen yksinkertainen mutta samalla toimiva ratkaisu on toteuttaa verkkopalvelimelle joko yhdistetty tai erilliset verkkosovellukset sisällön jakeluun ja hallintaan.

Yksinkertaisimmillaan keskitetty sisällön jakelu voidaan toteuttaa laittamalla verkkoon tiedosto, jonka sisältö luetaan sisällön loppukäyttäjälle näyttävällä sovelluksella. Tällaisessa järjestelyssä sisällön päivittäminen on kuitenkin erittäin kankeaa ja työlästä, ja lisäksi järjestely on erittäin altis virheille etenkin jos tiedosto luodaan tai muokataan suoraan ihmisen toimesta. Siksi onkin syytä toteuttaa järjestelmä, joka dynaamisesti luo pyydetyn sisällön joko automaattisesti muodostettuna tiedostona tai jollakin muulla menetelmällä. Tällöin sisältö tallennetaan joko tietokantaan tai erillisiin tiedostoihin, joista haluttu tieto haetaan automaattisesti pyydettävää sisältöä muodostettaessa. Tällöin myös tietokantaan tai erillisiin tiedostoihin tallentaminen on syytä toteuttaa automatisoidun järjestelmän välityksellä, jolloin käyttäjän ei tarvitse huolehtia kuin tuottamastaan sisällöstä. Järjestelmä pitää huolen, että sisältö tallennetaan oikein ja järjestelmän vaatimalla tavalla.

5.3 Hallintajärjestelmän käyttöliittymä

5.3.1 Web-käyttöliittymä

Käytännössä informaation hallintajärjestelmä toimii sovellusteknisesti verkkopalvelimella, joten se tarvitsee käyttöliittymän, jonka välityksellä palveli-

mella sijaitsevaa informaatiota käsitellään ja hallitaan. Järjestelmän käyttöliittymäksi olisi mahdollista toteuttaa erillinen sovellus, jonka avulla käyttäjät pystyisivät hallitsemaan järjestelmän tietoja. Yksinkertaisempi, mutta kuitenkin useimmissa tapauksissa riittävän joustava menetelmä on toteuttaa Internet-selaimella käytettävä Web-käyttöliittymä tiedonhallintajärjestelmälle.

Web-käyttöliittymän avulla hallintajärjestelmä on käytettävissä tutussa ympäristössä Internet-selaimella, eikä järjestelmän käyttäminen vaadi ohjelmistojen asentamista käyttäjien tietokoneisiin. Samalla myös järjestelmään pääsyä rajoittavat ainoastaan käyttöoikeudet, sekä mahdollinen sisäverkon toimialue. Toisin sanoen Web-käyttöliittymän kautta järjestelmä on joustavasti käytettävissä.

Web-käyttöliittymä voidaan toteuttaa esimerkiksi palvelinohjelmointia hyödyntävällä verkkosivulla, jonka välityksellä voidaan käsitellä esimerkiksi tietokantaa. Ja mikäli myös tiedon välitykseen käytetään dynaamisesti palvelinsovelluksella muodostettua dataa, kannattanee myös Web-käyttöliittymä toteuttaa samaisilla palvelinpuolen tekniikoilla, mikäli mahdollista. Näin toteutettuna sisällön tallentaminen, säilytys, käsittely ja hakeminen tapahtuvat samalla sovellusalustalla ja samoilla tekniikoilla toteutettuina.

Web-käyttöliittymä olisi täysin toteutettavissa myös Flashilla, mutta tämä toteutusmenetelmä saattaisi aiheuttaa ylimääräisiä rajoituksia Web-käyttöliittymän käyttämiselle. Vaikka Flashin käyttö on erittäin yleistä – Adobe ilmoittaa Flash Playerin olevan käytössä 98 % Internetin käyttäjistä – ei Flash välttämättä ole käytettävissä kaikilla päätteillä (Flash Player Penetration, 2007). Tämä saattaa olla tilanne varsinkin yritysverkossa, sillä Flash Player on ohjelma, joka on asennettava koneelle erikseen. Myös järjestelmän käytettävyys tarpeen tullen mobiilisovelluksilla tulisi ottaa huomioon.

5.3.2 Käyttäjien autentikointi

Jotta kuka tahansa ei pääsisi muokkaamaan järjestelmän sisältöä verkossa toimivan hallintajärjestelmän kautta, on käyttöliittymään syytä toteuttaa jonkinlainen käyttäjien autentikointimenetelmä. Erittäin tyypillinen autentikointimenetelmä on salasanoihin ja käyttäjätunnuksiin perustuva käyttäjien tunnistus. Autentikointi on mahdollista suorittaa useilla eri tasoilla ja toteutustekniikoilla.

Kontrolloidussa verkossa, jossa käyttäjien on kirjauduttava järjestelmään verkkoon päästäkseen, myös pääsyä hallintajärjestelmään voidaan rajoittaa verkon käyttäjätilien perusteella. Käyttäjien pääsyä hallintajärjestelmän verkkosijaintiin voidaan rajoittaa verkon käyttöoikeuksien avulla.

Käyttäjän tunnistus voidaan suorittaa myös HTTP-protokollan perusautentikoinnin avulla *HTTP Basic Authentication* -menetelmällä. HTTP-protokollan perusautentikointi perustuu käyttäjätunnukseen ja salasanaan, joita Web-palvelin pyytää selaimen välityksellä käyttäjältä Web-sovellukseen mentäessä. Verkkopalvelin vertaa annettua tunnusta ja salasanaa palvelimelle tallennettuihin tietoihin, ja tunnuksen ja sitä vastaavan salasanan täsmätessä sallii käyttäjän pääsyn verkkosivulle. Jatkossa Web-selain lähettää autentikointitiedot palvelimelle jokaisen pyynnön yhteydessä kunnes selain sammutetaan, jotta tunnusta ja salasanaa ei kysyttäisi jokaisen käyttäjän tekemän toimenpiteen yhteydessä. (Rantala, 2005, 230-232.)

Käyttäjän tunnistus ja autorisointi voidaan toteuttaa myös verkkosovelluksessa itsessään, esimerkiksi PHP-kielellä. Näin toteutettuna tunnusten ja salasanojen kysely tapahtuu itse verkkosovelluksen kautta, kuten myös annettujen tunnusten ja salasanojen todennus. Tämän toteutusmenetelmän etuja on mm. se, että näin voidaan luoda erilaisia käyttäjätasoja tai -ryhmiä, joilla on erilaiset käyttöoikeudet palveluun. Käyttäjätiedot voidaan tallentaa erilliseen tiedostoon tai tietokantaan, jolloin käyttäjäryhmiä ja -tasoja voidaan hallita myös verkkosovelluksen kautta. (Rantala, 2005, 232-233.)

6 CASE: NÄYTÖNSÄÄSTÄJÄTIEDOTUSJÄRJESTELMÄ PEIKKO FINLAND OY:LLE

6.1 Projektin kuvaus

6.1.1 Peikko Finland Oy

Peikko Group on suomalainen perheyritys, joka on betonirakentamisen kiinnitysosien edelläkävijä Suomessa. Yritys perustettiin vuonna 1965 Teräspeikko -nimellä, mutta yrityksen nimi muutettiin Peikoksi vuonna 2005 kansainvälisen toiminnan kasvettua merkittävämmäksi. Yrityksen ensimmäinen tuote, sandwich-elementtien kuoret toisiinsa liittävä ansas, oli alan ensimmäinen teollisesti valmistettu tuote.

Peikko Group keskittyy Euroopan rakennusteollisuuden palvelemiseen, ja Peikko Gruopin emoyhtiö Peikko Finland Oy:llä on tytäryhtiöitä 15 Euroopan maassa. Peikko Groupin liikevaihto vuonna 2006 oli 60 milj. EUR, ja yrityksessä toimii yli 500 työntekijää.

6.1.2 Työn tarkoitus ja tavoitteet

Työn lähtökohtana oli yrityksen halu parantaa yrityksen sisäistä viestintää. Ratkaisua lähdettiin hakemaan näytönsäästäjillä, jotka toimisivat sekä kalentereina että eräänlaisena informaatiokanavana sekä tiedotusvälineenä, jonka sisältö on keskitetysti hallittavissa. Samalla näytönsäästäjä toimii yrityksen yrityskalenterin digitaalisena toteutuksena ja on näin ollen osaltaan myös PR-väline.

Osana Peikko Finland Oy:lle toteutettavaan projektiin kuului myös painettavan yrityskalenterin toteutus. Painettu kalenteri on ensisijaisesti PR-tuote, jota jaetaan yhteistyökumppaneille ja asiakkaille. Uudistetun kalenterin tarkoituksena oli vahvistaa Peikon imagoa kansainvälisenä yrityksenä ja toimia osana yrityksen uudistettua graafista ulkoasua. Lisäksi lähtökohtaisesti kokonaan yrityksen oman kalenterin toteuttamista puolsi halu korvata tyypillinen maisema- ja tyttökalenteri persoonallisemmalla yrityskalenterilla.

Uuden kalenterin vaatimuksena oli näyttävä sekä huomiota kiinnittävä ja yrityksen graafisen ilmeen mukainen ulkoasu. Kalenterin kuvallisena teemana oli Peikko Finland Oy:n tytäryritysmaat yhdistettynä Peikon tuotteisiin kevyen vitsin yhteydessä.

Näytönsäästäjän yhtenä lähtökohtana oli yrityskalenterin digitaalinen toteutus. Näytönsäästäjän tulisi olla toiminnallinen kalenteri, joka näyttää vähintään kuluvan kuukauden kalenteriruudukon sekä kuluvan päivän sekä vaihtaa esitettävää kuvaa kuukauden mukaan. Lisäksi näytönsäästäjän tuli toimia tiedotuskanavana, jonka välityksellä on mahdollista tiedottaa ajankohtaisista sekä tulevista tapahtumista tai muista huomioitavista asioista.

Tiedotteiden lisäämiseen ja hallintaan oli oltava oma keskitetty järjestelmänsä, jonka käyttöoikeudet on hallittavissa ja rajattu. Tiedotteiden voimassaoloaika tulee olla määriteltävissä, ja tiedotteiden yhteyteen tulee voida haluttaessa lisätä kuvan. Näytönsäästäjät tulevat käyttöön Peikon sisäiseen verkkoon kytkettyihin koneisiin, joissa on käytössä Windows XP -käyttöjärjestelmä.

6.2 Valitut työkalut, menetelmät ja tekniikat

Varsinainen näytönsäästäjäsovellus toteutettiin Flash 8 Professional -ohjelmalla. Flash osoittautui ominaisuuksiensa ansiosta soveltuvan erittäin hyvin sisällöltään dynaamisen näytönsäästäjän toteutukseen. Flash tarjoaa työkalut niin ulkoasun kuin toimintalogiikan toteutukseen. Lisäksi Flash tukee monipuolisesti ulkoisen sisällön käyttämistä Flash-esityksissä. Etenkin XML-muodossa olevan tiedon käsittelyyn Flash tarjoaa monipuoliset ominaisuudet.

Ongelmana Flashin käytössä näytönsäästäjän toteutukseen oli lähinnä Flash-sovelluksen muuntaminen käyttöjärjestelmän ymmärtämäksi näytönsäästäjäksi. Casen toteutuksessa turvauduttiin apuohjelmaan, joka luo Windows-näytönsäästäjän, joka käynnistyessään avaa sisällytetyn Flash-esityksen. Lopullinen näytönsäästäjä luotiin InstantStorm 1.0 -ohjelmalla, joka on ominaisuuksiltaan suppea, mutta ilmainen ja täysin vapaasti käytettävissä. Ohjelman suppealla ominaisuusvalikoimalla ei ollut käytännön merkitystä, sillä kaikki näytönsäästäjän toiminnallisuus on toteutettu itse Flash-sovelluksessa, jonka luotu näytönsäästäjä vain käynnistää.

Näytönsäästäjissä näytettävän sisällön jakelu- ja hallintajärjestelmä toteutettiin käyttäen toiminta-alustana yrityksen sisäistä ”helpdesk” -verkkopalvelinta, jolla oli käytettävissä PHP-ohjelmointikieli sekä MySQL tietokanta. Käytettävissä oli PHP:n versio 5, ja MySQL -tietokannan hallintaan kehitysvaiheessa käytettiin phpMyAdmin -sovellusta.

Sisällön hallinta- ja jakelujärjestelmä toteutettiin PHP-palvelinohjelmoinnilla. Sisältö tallennetaan hallintajärjestelmän kautta palvelimelle käyttäen MySQL tietokantaa tekstisisällön tallentamiseen, ja näytönsäästäjissä käytettävä vaihtuva kuvamateriaali tallennetaan palvelimelle kuvatiedostoina järjestelmän kautta. Näytönsäästäjien käyttämä XML-tiedosto luodaan dynaamisesti PHP-kielellä tietokantaan tallennetun sisällön perusteella.

Sisällön jakelujärjestelmän ja näytönsäästäjien välinen tiedonsiirto hyödyntää HTTP-protokollaa yrityksen sisäisen verkon yli. Sisältö siirretään palvelimen ja näytönsäästäjien välillä XML-kielellä muotoiltuina tekstidokumentteina sekä kuvatiedostoina, jotka näytönsäästäjät lataavat suoraan verkkopalvelimelta.

Sekä näytönsäästäjässä että painetussa yrityskalenterissa käytetyn kuvamateriaalin toteutukseen käytettiin 3Ds MAX 8 -pintamallinnusohjelmaa. Mallinnetun kuvamateriaalin muokkaamiseen ja kaiken muun kuvamateriaalin toteutukseen käytettiin Adobe Photoshop CS2 ja Illustrator CS2 -ohjelmia. Painetun yrityskalenterin taittoon käytettiin Adobe InDesign CS2 -taittoohjelmaa.

6.3 Näytönsäästäjään ja kalenteriin toteutettu kuvamateriaali

6.3.1 Yrityksen imagon ja brand manualin mukainen toteutus

Kalenterin sekä näytönsäästäjän yleisen ilmeen tuli vastata Peikko Groupin graafista ulkoasua. Sekä kalenterissa että näytönsäästäjässä käytettiin soveltaen samoja graafisia elementtejä, värejä, asettelua sekä kirjasimia, kuin mitkä ovat myös käytössä muissa Peikko Groupin julkaisuissa. Ulkoasujen suunnittelussa oli mahdollista ottaa tiettyjä vapauksia tuoreen ja ennen kaikkea näyttävän ulkoasun luomiseksi säilyttäen kuitenkin selkeän yhdenmukaisuuden muun graafisen materiaalin kanssa. Sekä kalenterin että näytönsäästäjän ulkoasuissa käytettiin selvästi tavallista julkaisua enemmän tummia sävyjä sekä mustaa. Kalenterin ulkoasussa käytettiin runsaasti syvää mustaa sekä näyttävyyden, että tavanomaisesta valkopohjaisesta kalenterista erottuvuuden aikaansaamiseksi. Näytönsäästäjässä käytettiin samaa kuvamateriaalia kuin kalenterissakin, ja tumman yleissävyn käyttäminen näytönsäästäjässä on perusteltua jo käyttötarkoituksesta johtuen, joskin näytönsäästäjän yleisilmeen määrittelee hyvin pitkälti taustana käytetty kuva.

6.3.2 Pohja- ja taustamateriaali

Kalenterin kuvallisena teemana oli Peikko Finland Oy:n tytäryritykset, joita kalenterin tekohetkellä oli kolmetoista kappaletta. Kalenterin kanteen sekä jokaiselle kuukaudelle etsittiin kuva jokaisesta tytäryritysmaasta. Kuvien teemana olivat joko yleisesti tunnetut tai kyseessä olevalle maalle tyypilliset, maahan helposti yhdistettävät, rakennukset tai arkkitehtuuri. Lisäksi kalenterin kuviin tuli sisältyä kevyt vitsi ja huomion kiinnittäjä, joka tässä tapauksessa oli Peikon yleisimpien tuotteiden lisääminen kuviin epätavallisissa käyttötarkoituksissa. Tämä toteutettiin manipuloimalla valokuvia niin, että kuvista on tarkkaan katsottaessa löydettävissä Peikon tuotteita korvaamassa jotakin todellisuudessa esiintyvää. Kuvien realistisuus tuli kuitenkin säilyttää niin, et-

tä kuvat näyttivät edelleen aidoilta valokuvilta ilmeisestä logiikanvastaisuudesta huolimatta. Alkuperäismateriaalina toimineet kuvat etsittiin ja ostettiin Internetin suositusta kuvapankkipalvelusta iStockphoto.com:sta.

6.3.3 Työvaiheet ja toteutusmenetelmät

Mallintaminen

Menetelmäksi tuotteiden lisäämiseksi kuviin valittiin 3D-mallintaminen yhdistettynä kuvankäsittelyyn. Mallintamiseen käytettiin 3D Studio MAX 8 -ohjelmaa, vaikka kaikista Peikon tuotteista oli saatavilla valmiit 3D-kokoonpanot SolidWorks -ympäristöön. Käytettävän ohjelman valintaperusteina olivat muun muassa 3Ds MAXin monipuolisemmat ominaisuudet verrattuna PhotoWorksilla laajennettuun SolidWorksiin, sekä kirjoittajan SolidWorksia laajempi käyttökokemus 3Ds MAXista. Myös 3Ds MAXin visuaaliseen toteutukseen soveltuvuus katsottiin merkittäväksi eduksi verrattuna huomattavasti teknisempiin lähtökohtiin pohjautuvaan SolidWorksiin.

Kuvissa näkyvät tuotteet mallinnettiin alusta asti 3Ds MAXissa sen sijaan, että olemassa olevat SolidWorks -kokoonpanot olisi tuotu 3Ds MAXiin. Lopullisissa kuvissa näkyvien tuotteiden tarkkuudeksi riitti silmämääräinen tunnistettavuus ja oikeellisuus. Koska mittatarkkaa toteutusta ei vaadittu, saavutettiin tuotteiden uudelleenmallintamisella merkittävästi kevyemmät, eli pintojen lukumäärältään pienemmät 3D-mallit tuotteista. Näin toteutettuna 3D-mallien käsiteltävyys ja hallinnointi oli helpompaa ja nopeampaa. Tämä toteutus mahdollisti myös tuotteiden mallien helpon muokattavuuden kunkin kuvan tarpeisiin.

Tuotteiden mallit pyrittiin toteuttamaan mahdollisimman yksinkertaisiksi ja kevyiksi säilyttäen kuitenkin tuotteiden todenmukainen ulkonäkö kaavaillulla esitystarkkuudella. Lähes kaikissa kuvissa mallinnetut tuotteet näkyvät vain pienenä osana kuvaa, joten useimpien tuotteiden kohdalla huomattavat yksityiskohtien karsimiset ja karkea mallinnustarkkuus olivat mahdollisia. Näin myös monet tuotteiden yksityiskohdista pystyttiin toteuttamaan esimerkiksi bumpmapin avulla, jolloin mallien geometria pystyttiin pitämään hyvin yksinkertaisena.

3D-mallit sijoitettiin valokuviin käyttämällä alkuperäistä valokuvaa, tai osaa siitä, kameranäkymän taustana. 3D-ympäristössä ollut virtuaalinen kamera si-

joitettiin yksinkertaisia geometrisia muotoja apuna käyttäen niin, että 3D-ympäristön perspektiivi kameranäkymän kautta tarkasteltuna vastasi valokuvan perspektiiviä. Näin tuotteiden 3D-malleja oli mahdollista liikutella 3D-ympäristössä niin, että ne säilyttivät valokuvaa vastaavan perspektiivin. Myös mallit valaisevien ja varjot piirtävien valonlähteiden sijainti ja suunta etsittiin samaan tapaan vertaamalla valon ja varjojen suuntaa valokuvassa esiintyviin.

Tuotteiden 3D-mallien pintamateriaalit muokattiin jokaista kuvaa varten erikseen. Pintamateriaalien värit, teksturointi sekä heijastukset muokattiin kutakin kuvaa varten niin, että renderöidyt mallit sopisivat alkuperäisiin valokuviin ympäristöönsä sekä kuvan yleiseen väritykseen.

Renderöinti

Kuvat Renderöitiin 3Ds MAXilla. Mallit renderöitiin tilanteesta riippuen joko yksi tai useampi kuvassa esiintyvä tuote kerrallaan ilman taustaa. Tarkoituksena ei missään tapauksessa ollut tuottaa valmista kuvaa vaan materiaalia jatkokäsittelyä varten. Renderöityihin tuotteiden kuviin pyrittiin saamaan oikea asento, valokuvan mukaiset varjot sekä ympäristön mukainen väri.

Kuvamanipulointi

Kuvamanipulointiin käytettiin Adobe PhotoShop CS2 -ohjelmaa, joka on erittäin tehokas ja yleisesti käytetty kuvankäsittelyn yleistyökalu. Kuvapankista ostettuja kuvia muokattiin tarvittaessa esimerkiksi poistamalla kuvista ihmiset tai muut häiritsevät elementit, kuten kohteen edessä häiritsevästi näkyvät oksat ja lipputangot tai kuvan taustalla näkyvät epäoleelliset kohteet, kuten rakennustelineet.

Ennen renderöityjen kuvien tuomista valokuviin kuvista poistettiin rakennuksen elementit, jotka oli tarkoitus korvata renderöidyillä Peikon tuotteilla, mikäli renderöity kuva ei täysin peittänyt korvattavaa kohdetta.

Tämän jälkeen renderöidyt kuvat liitettiin valokuvan päälle omille tasoilleen. Tuotteiden kuvat renderöitiin melko suurella resoluutiolla, ja skaalattiin PhotoShopissa oikeaan kokoon. Tuotteet myös renderöitiin kokonaisina, jolloin kuvissa piiloon jäävät osat piilotettiin käyttäen PhotoShopin *Layer Mask* -toimintoa.

Kun renderöidyt kuvat oli upotettu luonnollisen näköisesti oikeisiin sijainteihinsa, kuvan renderöityjä osia muokattiin ja käsiteltiin tarpeen mukaan. Renderöityjä elementtejä saattoi olla tarpeen mm. pehmentää kevyesti, jotta ne eivät erottuisi kuvasta muuta valokuvaa terävämpinä. Useiden kuvien kohdalla renderöityihin osiin oli myös tarpeen lisätä valoisuus- ja värikohinaa, jotta renderöidyt kohdat vastasivat myös näiltä osin aitoa valokuvaa. Myös renderöityjen kohteiden sävy, valoisuus ja kontrasti hienosäädettiin sopivaksi muuhun kuvaan nähden.

Jälkikäsittely

Kun renderöidyt tuotteet oli liitetty valokuviin, käsiteltiin kuvia edelleen. Tässä vaiheessa kuvien sävy, valoisuus ja kontrasti hienosäädettiin koko kuvan osalta. Koska suurin osa kuvista oli vaakasuuntaisia ja painettu kalenteri pystysuuntainen, lähes kaikkia kuvia täytyi kasvattaa korkeussuunnassa, joko ylä- tai alareunasta, tai molempiin suuntiin. Kuvien korkeutta kasvatettiin joko lisäämällä kuvaan taivasta tai jatkamalla kuvan alareunaa esimerkiksi jatkamalla näkyvää rakennusta tai kuvan alareunan maastoa alaspäin. Kuvien jatkaminen tehtiin kopioimalla, piirtämällä ja keksimällä kuvaan uutta materiaalia. Tässä työvaiheessa tärkeimpinä työkaluina toimivat PhotoShopin kloonaustyökalu, ns. *healing brush* -työkalu sekä perspektiivikloonaus -toiminto. Erityisen suurta huomiota tuli kiinnittää siihen, että kuvista ei selkeästi huomaisi toistuvia kohtia. Tämän takia kuvan osia kopioitiin pienissä paloissa ja vapaalla kädellä käytettävillä kopiointityökaluilla. Joistakin kuvista vaihdettiin taivas kokonaan uuteen, jolloin kuvassa näkyvää taivasta ei tarvinnut kopioida. Tämän jälkeen muutoin valmiit kuvat rajattiin ja skaalattiin tarvittavaan kokoon ja resoluutioon.

6.4 Sisällönhallintajärjestelmän toteutus

6.4.1 Toimintaympäristö

Sisällönhallintajärjestelmä toteutettiin hyödyntäen palvelinohjelmointia. Toiminta-alustana käytettiin Unix-palvelinta, jossa on käytössä niin kutsuttu LAMP-WWW-palvelinohjelmistoyhdistelmä (Linux, Apache, MySQL, PHP). Käyttöjärjestelmänä palvelimella toimii CentOS Linux, ja palvelimella on käytössä Apache HTTP-palvelinohjelma, PHP-ohjelmointikieli sekä

MySQL-tietokanta. Palvelin toimii yrityksen sisäisessä verkossa, päätarkoituksenaan toimia IT-hallinnon apuna tukipyyntöjen organisoinnissa.

Palvelin on kytketty ainoastaan yrityksen sisäiseen verkkoon, joten esimerkiksi sisäverkon ulkopuolelta Internetin kautta palvelin ei ole tavoitettavissa. Näin ollen sisällön hallintajärjestelmän ja näytönsäästäjien välinen kommunikointi tapahtuu kokonaan Peikko Finland Oy:n sisäisessä verkossa. Tämä järjestely sopii kyseessä olevaan käyttötarkoitukseen erittäin hyvin, sillä järjestelmä on tarkoitettu vain yrityksen sisäiseen tiedotukseen. Kaikki yrityksessä käytössä olevat tietokoneet ovat kytkettynä yrityksen sisäiseen verkkoon, joten kaikki käytössä olevat näytönsäästäjät pystyvät muodostamaan yhteyden palvelimeen, jolta tiedot haetaan.

Koska järjestelmään tallennettu tieto on saatavilla ainoastaan sisäisessä verkossa, tiedotteiden hallinnan on myös tapahduttava sisäisen verkon kautta. Sisällönhallintajärjestelmän katsottiin parhaaksi toimia Web-käyttöliittymänä Internet-selaimen välityksellä. Näin ollen hallintajärjestelmän käyttö vaatii pääsyn yrityksen verkkoon. Hallintajärjestelmää pystytään kuitenkin käyttämään esimerkiksi VPN-yhteyden avulla, jolloin etätyöasema on yhdistettynä yrityksen verkkoon suojatun yhteyden välityksellä Internetin yli. Hallintajärjestelmän sijaitseminen suljetussa verkossa on kuitenkin tietoturvan kannalta hyvä ratkaisu, sillä järjestelmän ollessa saavuttamattomissa julkisen verkon kautta, järjestelmän mahdollisesti aiheuttamat ilkivalta- ja tietoturvariskit vähentyvät merkittävästi.

6.4.2 Sovelluslogiikka ja käytettävät tekniikat

Sisällön hallintajärjestelmän sovelluslogiikka on toteutettu pääosin PHP-ohjelmoinnilla. Sekä hallintajärjestelmän Web-käyttöliittymä, tiedot sisältävän tietokannan käsittely että näytönsäästäjien lataaman informaatiotiedoston dynaaminen luominen on toteutettu PHP-kielellä hyödyntäen siihen integroituja MySQL-tietokantojen käyttöominaisuuksia. Hallintajärjestelmän käyttöliittymän toteutuksessa hyödynnettiin marginaalisesti myös JavaScriptiä.

Hallintajärjestelmän käyttöliittymä on toteutettu niin, että Internet-selaimen kannalta sivu on tavanomainen HTML-dokumentti. Järjestelmään lisättävien tiedotteiden tiedot syötetään tavallisen HTML-lomakkeen avulla. Tiedot syötetään tekstikenttiin, tai ne valitaan muilla tavanomaisilla lomakkeen valintavaihtoehdoilla. Tiedotteen näkyvyysaikaväli määritellään asettamalla uutisen

ensimmäinen ja viimeinen näyttöpäivä. Päivämäärä valitaan valitsemalla päivä linkistä aukeavasta kalenteriruudukosta haluttua päivämäärää klikkaamalla. Uuteen ikkunaan aukeava kalenterinäkymä luodaan dynaamisesti PHP:n avulla. Kalenterinäkymää voi vapaasti selata eteen (ja taakse), ja kalenterinäkymän ikkuna sulkeutuu automaattisesti päivämäärän valinnan jälkeen, ja valittu päivä tulostetaan päivämäärän syötekenttään JavaScriptin avulla. Päivämäärä on myös mahdollista kirjoittaa suoraan syötekenttään ilman kalenterivalitsimen käyttöä.

Käyttäjän tallentaessa tiedotteen, lomakkeen tiedot lähetetään PHP-skriptille, joka tarkistaa ja käsittelee tiedot mahdollisten virheiden varalta. Mikäli annetuissa tiedoissa ei havaita automaattisesti korjaamattomissa olevia ongelmia, tiedot tallennetaan MySQL-tietokantaan. Jos käyttäjä on liittänyt uutisen yhteyteen kuvan, lähetetty tiedosto varmistetaan oikean tyyppiseksi kuvatiedostoksi, minkä jälkeen kuva tarvittaessa skaalataan GD-kirjaston toimintojen avulla sallittuihin kokorajoihin, ja tiedosto tallennetaan yksilöidyllä tiedostonimellä palvelimelle. Tiedot palvelimelle tallennetuista kuvatiedostoista tallentuvat omaan SQL-tauluunsa, johon viitataan yksilöidyn tunnisteen avulla tiedotteiden tiedot sisältävästä SQL-taulusta.

Kun tiedote poistetaan järjestelmästä, tai tiedotteen kuva vaihdetaan tai poistetaan, poistetaan tiedotteen yhteydessä tallennettu kuva myös palvelimelta. Palvelimella siis säilytetään ainoastaan järjestelmässä viitattuna olevia kuvatiedostoja. Järjestelmästä poistettaessa tiedote tai pelkkä tiedotteeseen liittyvä kuva, poistetaan myös vastaavat viittaukset tietokannasta, jolloin myöskään tietokannan koko ei paisu ajan mittaan.

Käyttöliittymän yksinkertainen toteutus, eli selaimen kannalta tavanomainen HTML-sivusto, mahdollistaa järjestelmän käyttämisen tarvittaessa myös mobiililaitteilla.

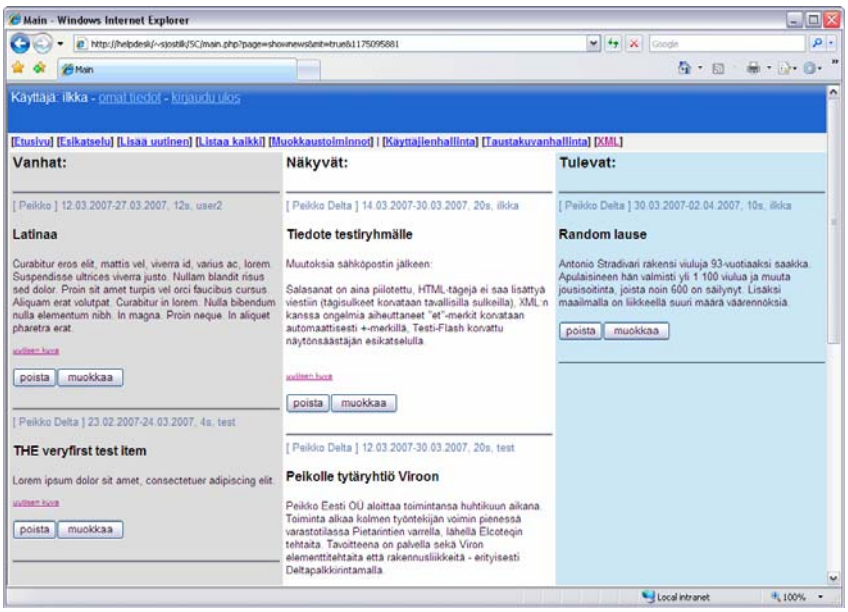
Näytönsäästäjän pyytäessä palvelimelta tietoja näytettävästä sisällöstä, tiedot sisältävä XML-tiedosto luodaan dynaamisesti senhetkisten tietojen pohjalta. XML-dokumentti luodaan PHP-skriptin avulla. XML-dokumentti tulostetaan käyttäen UTF-8 merkistöä, jolloin skandinaaviset tai muutkaan erikoiset merkit eivät aiheuta ongelmia. Näytettävien uutisten valikointi ja suodatus suoritetaan osittain suoraan SQL-tietokantahaun yhteydessä sekä PHP:n avulla.

6.4.3 Käyttöliittymä

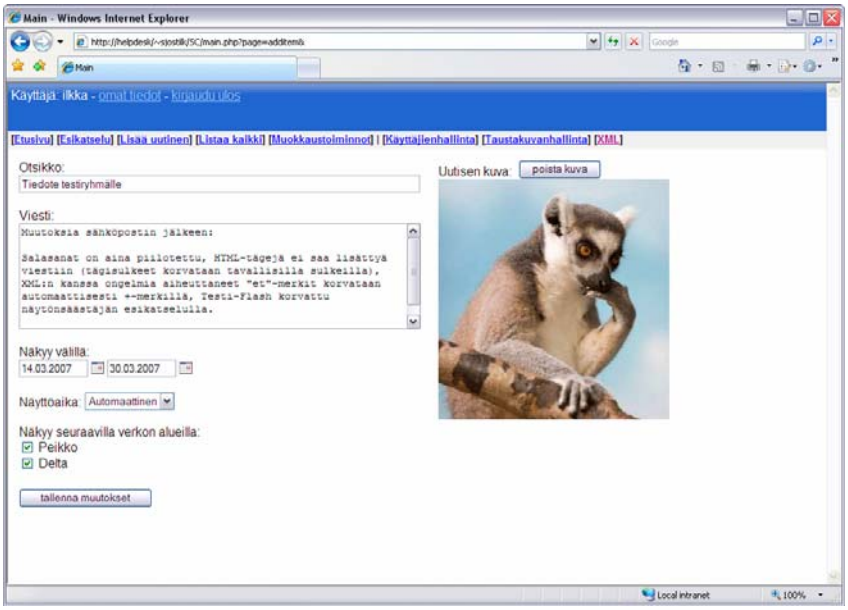
Näytönsäästäjätiedotusjärjestelmän sisällön hallinta tapahtuu täysin hallintajärjestelmän käyttöliittymän välityksellä. Tarkoituksena on, että hallintajärjestelmän käyttöliittymä ja näin ollen koko järjestelmän sisällön hallinta on helppokäyttöinen ja helposti omaksuttava järjestelmä, eikä vaadi käyttäjältä tekniikan erityistuntemusta. Käyttäjän huolena tulee olla ainoastaan itse sisällöstä vastaaminen ilman teknisistä näkökohdista huolehtimista.

Sisällön hallintajärjestelmän käyttöliittymässä päädyttiin Web-käyttöliittymätoteutukseen. Näin toteutettuna hallintajärjestelmän käyttö tapahtuu Internet-selaimen välityksellä, joka on todennäköisesti useimmille hallintajärjestelmän loppukäyttäjille tuttu ja arkinen väline. Näin myös hallintajärjestelmä on käytännössä käytettävissä mistä tahansa yrityksen verkon toimialueella.

Käyttäjät pystyvät lisäämään sekä muokkaamaan järjestelmässä näkyviä tiedotteita hallintajärjestelmän käyttöliittymän kautta. Lisätessään uutisen tai tiedotteen, käyttäjä määrittelee uutiselle otsikon ja tekstisisällön sekä halutessaan uutisen yhteydessä näytettävän kuvan. Lisäksi uutiskohtaisesti on määriteltävä aikaväli, jolloin uutinen näkyy näytönsäästäjissä päivämäärien perusteella. Oletuksena aikavälinä on viikko kuluvasta päivästä alkaen. Oletusarvoisesti tiedotteen näyttämiseen käytettävä aika määritellään automaattisesti annetun tekstisisällön pituuden perusteella. Tiedotteita pystyy siis lisäämään järjestelmään etukäteen järjestelmän sovelluslogiikan huolehtiessa ajankohtaisten tiedotteiden näyttämisestä itse näytönsäästäjissä. Järjestelmän kautta pystyy myös luonnollisesti selaamaan ja tarkastelemaan sekä tulevia että vanhentuneita tiedotteita. Käyttäjä voi myös halutessaan asettaa itse tiedotteen näytöllä näyttämiseen käytettävän ajan valitsemalla sen muutamasta ennalta määritellystä vaihtoehdosta. Kaikkia tiedotteiden ominaisuuksia, joihin käyttäjä pystyy vaikuttamaan, voi muuttaa jälkikäteen, tai koko uutisen voi poistaa järjestelmästä.



KUVA 2. Sisällönhallintajärjestelmän käyttöliittymän Muokkaustoiminnot -sivu järjestelmän kehitysvaiheessa.



KUVA 3. Sisällönhallintajärjestelmän tiedotteiden lisäämiseen ja muokkaamiseen käytettävä sivu.

Hallintajärjestelmän kautta vaikutetaan myös näytönsäästäjien toimintaan ja ulkoasuun. Järjestelmän käyttöönottovaiheessa ainut vaikutusmahdollisuus oli näytönsäästäjien kuukauden mukaan vaihtuvien taustakuvien hallinta, sillä tarvetta muunlaiselle toimintaan vaikuttamiselle ei ollut tullut esille. Jokaiselle kuukaudelle määritellään ja siirretään palvelimelle näytönsäästäjän käynnistyessä näytönsäästäjään ladattava taustakuva.

6.4.4 Käyttäjienhallinta

Hallintajärjestelmän käyttöä on rajoitettava niin, että järjestelmään ei ole avointa pääsyä kenellä tahansa yrityksen sisällä mahdollisten väärinkäytöksien ehkäisemiseksi. Käyttäjien tunnistus ja hallinta päätettiin toteuttaa kirjautumismenetelmällä. Käyttääkseen hallintajärjestelmää, käyttäjän on kirjauduttava Web-käyttöliittymäsivustolle käyttäen henkilökohtaista käyttäjätunnusta ja salasanaa. Näin hallintajärjestelmän käyttäminen ei ole sidottu esimerkiksi Windows-verkkoon kirjautumiseen. Järjestelmän itsenäinen käyttäjien hallinnoiminen mahdollistaa myös erilaisten käyttäjätasojen tai -profiilien käytön. Tämä on tarpeen järjestelmän käyttäjien ja näytönsäästäjien asetusten hallinnan tapahtuessa myös saman hallintajärjestelmän kautta. Näin käyttäjät voidaan jakaa peruskäyttäjiin ja laajennetut oikeudet omaaviin käyttäjiin. Peruskäyttäjät voivat lisätä uutisia järjestelmään ja muokata sekä poistaa itse lisäämiään uutisia. Laajemmilla oikeuksilla varustetut käyttäjät voivat muokata ja poistaa myös muiden lisäämiä uutisia, muokata järjestelmän käyttäjätilejä sekä näytönsäästäjien toimintaan vaikuttavia asetuksia. Esimerkiksi näytönsäästäjien taustakuvan vaihto on mahdollista ainoastaan korkeamman tason käyttäjätunnuksilla.

6.5 Näytönsäästäjän toteutus Flashilla

6.5.1 Vaatimukset ja tavoitteet näytönsäästäjälle

Vaikka näytönsäästäjissä näyttöä säästävä vaikutus ei ole enää keskeinen tarkoitus, kannattaa tämä seikka kuitenkin ottaa huomioon toteutettaessa näytönsäästäjää, joka on tarkoitettu informaation esittämiseen. Luonnollisesti tapauksessa, jossa näytönsäästäjä toimii viestintäkanavana, on tarkoituksenmukaista, että näytöllä näkyy näytönsäästäjä sen sijaan, että monitori sammutettaisiin. Koska kohdeyrityksen laajasta konekannasta huomattava osuus on edelleen varustettu kuvaputkinäytöillä, etenkin tuotantotiloissa, on kuvan kiinnipalamisilmiö syytä ottaa huomioon näytönsäästäjän toteutuksessa. Täysin staattisia tai muuttumattomia elementtejä tulisi välttää.

Yrityksen kannalta näytönsäästäjän keskeisin ominaisuus on näytönsäästäjien toimiminen tiedotuskanavana. Tiedotteen yhteyteen haluttiin olevan mahdollista liittää myös kuva. Mahdollisten tiedotteiden lisäksi näytönsäästäjien haluttiin näyttävän kuluvan kuukauden kalenterin. Näytönsäästäjän ulkoasusta haluttiin näyttävä, mutta hillitty. Ulkoasun haluttiin olevan vakuuttava, mutta

ei erityisen huomiota vetävä – näytönsäästäjien ei haluttu keräävän tarpeetonta huomiota.

Näytönsäästäjäjärjestelmän tekniselle toteutukselle annettiin varsin vapaat kädet. Yrityksen puolelta käytettäviin menetelmiin ei otettu juuri lainkaan kantaa, mutta näytönsäästäjäjärjestelmän haluttiin toimivan olemassa olevilla järjestelmillä.

6.5.2 Tiedonsiirto Flash-sovelluksen ja palvelimen välillä

Näytönsäästäjien ja keskitetyn sisällön tarjoavan palvelimen välinen kommunikointi tapahtuu lähiverkon yli HTTP-protokollaa käyttäen. Tekstimuotoinen sisältö sekä muu näytettävään sisältöön liittyvä informaatio siirretään näytönsäästäjälle XML-dokumenttiin muotoiltuna. Käynnistyessään näytönsäästäjä kutsuu palvelimella sijaitsevaa PHP-skriptiä, joka palauttaa dynaamisesti luodun XML-dokumentin. XML-dokumentti luodaan palvelimen tietokantaan tallennetun informaation perusteella, ja siihen sisällytetään ainoastaan ajankohtainen sisältö. XML-tiedosto sisältää mm. tiedon näytönsäästäjässä näytettävistä kuvista URL-osoitteina, joiden perusteella Flash-sovellus lataa ne palvelimelta.

6.5.3 Flash-näytönsäästäjäsovelluksen toteutus ja toiminta

Toteutuksen lähtökohdat

Eräs merkittävä lähtökohta näytönsäästäjän toteutukselle oli sekä näytönsäästäjien, että koko järjestelmän sisällön, toiminnan ja ominaisuuksien muokattavuus. Toteutus ja järjestelmän päivittäminen pyrittiin pitämään mahdollisimman joustavana mahdollisia muutoksia ja toiminnallisuuden lisäämistä silmälläpitäen. Sen lisäksi, että lähes kaikki sisältö ladataan näytönsäästäjään ulkoisesta lähteestä, myös varsinainen näytönsäästäjänä toimiva Flash-tiedosto ladataan palvelimelta näytönsäästäjän käynnistyessä. Päätteille asennettuihin näytönsäästäjiin on sisällytetty ainoastaan ns. käynnistäjäsovellus, joka käynnistyessään lataa palvelimelta, ja aktivoi varsinaisen näytönsäästäjän. Näin toteutettuna varsinaista näytönsäästäjää voidaan päivittää ilman, että lukuisia koneisiin asennettuja näytönsäästäjäsovelluksia tarvitsee päivittää. Tällä toteutuksella voidaan ottaa myös huomioon tapaus, jolloin yrityksen kannettava tietokone ei ole yhteydessä yrityksen verkkoon. Mikäli käynnistä-

jäsovellus ei saa yhteyttä palvelimelle, se toimii itsenäisenä, yksinkertaisempana näytönsäästäjänä. Myös käynnistäjäsovellus on toteutettu Flashilla.

Yhteyden todentaminen ja sisällön käsittely

Kun päätteen näytönsäästäjä aktivoituu, käynnistyy tietokoneelle asennettuun näytönsäästäjään sisällytetty Flashilla toteutettu käynnistäjäsovellus. Käynnistyessään sovellus testaa, saadaanko yhteys palvelimelle muodostettua. Mikäli yhteyttä ei saada, siirtyy käynnistäjäsovellus näytönsäästäjätilaan. Tämä vaihtoehtoinen näytönsäästäjä näyttää kellonajan.

Kun yhteys palvelimelle saadaan muodostettua, lataa käynnistäjäsovellus varsinaisen Flash-tiedostona olevan näytönsäästäjän palvelimelta, ja käynnistää sen itsensä tilalle. Vaikka käytännössä koko näytönsäästäjä sisältöineen ladataan palvelimelta sen käynnistyessä, käynnistyminen tapahtuu nopeasti, sillä ennen käynnistymistä siirrettävän datan määrä on pieni, ja se siirretään nopean lähiverkon yli. Lopullinen näytönsäästäjä aktivoituu heti, kun palvelimelta on ladattu Flash-sovellus, XML-muotoinen sisältö sekä CSS-tyylitiedosto. Näiden yhteenlaskettu koko on tekstisisällöstä riippuen vähintään n. 20 kilotavua ja lähes poikkeuksetta alle sata kilotavua.

Myös varsinaisessa näytönsäästäjäsovelluksessa on varotoimenpiteenä vastaava vaihtoehtoinen ulkoisesta informaatiosta riippumaton näytönsäästäjä kuin käynnistäjäsovelluksessakin. Tämä siltä varalta, että yhteys palvelimelle saadaan muodostettua, mutta esimerkiksi toiminnan kannalta keskeiset PHP tai SQL-tietokanta eivät olisi toimintahäiriön vuoksi käytettävissä. Tässä tilanteessa näytönsäästäjä ei kykene lataamaan XML-sisältöä, ja se aktivoi vaihtoehtoisen näytönsäästäjätoiminnon.

Näytönsäästäjän taustakuva sekä mahdolliset tiedotteiden kuvat ladataan niitä kutsuttaessa. Kummatkaan eivät ole näytönsäästäjän toiminnan kannalta kriittisiä, joten niiden latautumista etukäteen ei ole tarve odottaa. Kuvat tulevat näkyviin heti latauduttuaan. Tiedotteiden kuvia kutsutaan vasta, kun itse tiedote näkyy näytönsäästäjässä. Käytännössä kuvat siirtyvät lähiverkon yli lähes välittömästi.

Näytönsäästäjien tekstisisältö sekä muu sisältöön liittyvä informaatio ladataan heti käynnistyksessä XML-objektiin.

Ladattavan XML-tiedoston rakenne on seuraavanlainen:

```
<?xml version="1.0" encoding="UTF-8" ?>
<screensaver_data>
  <conf>
    <caldata dlkm="31" eka="4" pvm="19" />
    <bgimage>http://helpdesk/screensaver/bg_img/iso
maisemakuva.jpg</bgimage>
  </conf>
  <data>
    <nitem>
      <title>Otsikko</title>
      <content>Tiedotteen viesti</content>
      <image>http://helpdesk/screensaver/tatti ja
etana.jpg</image>
      <author>Ilkka S</author>
      <vistime>30</vistime>
    </nitem>
  </data>
</screensaver_data>
```

XML-rakenteen `<conf>` -osio määrittelee kuluvan kuukauden kalenterin muodostamiseen tarvittavat tiedot sekä taustakuvan URL-osoitteen. Jokainen tiedote on omassa `<nitem>` -solmussaan `<data>` -solmun alla. Mikäli `<data>` -solmun alla ei ole ainuttakaan lapsisolmua, eli järjestelmässä ei ole aktiivisia tiedotteita, näytönsäästäjässä näytetään vain kalenteri, kellonaika sekä taustakuva.

XML-informaation käsittely tapahtuu ActionScript -ohjelmoinnilla XML-komponenttien sijaan. Siirrettävään XML-tiedostoon ei sisälly rakennekuvausta, vaan näytönsäästäjäsovellus käsittelee sisältöä ennalta määritellyn kaavan mukaan.

Kalenterin luomiseen tarvittavat kolme numeerista arvoa generoidaan palvelimella PHP:n avulla sen sijaan, että kuluvan kuukauden kalenteri muodostettaisiin Flashissa. Näin tehdään, koska PHP tarjoaa ActionScriptiä monipuolisemmat aika- ja päivämäärätoiminnot, ottaen huomioon esimerkiksi karkausvuodet. Näin kalenteri myös pitää paikkansa vaikka kohdetietokoneella olisi virheellinen päivämäärä.

Näytönsäästäjän ulkoasu

Näytönsäästäjän ulkoasu on toteutettu melko yksinkertaisia ja pelkistettyjä komponentteja hyödyntäen. Ulkoasu muodostuu hallitsevasta taustakuvasta, jonka päällä on hieman läpikuultavia elementtejä, jotka sisältävät Peikko

Groupin logon, kellon ja kalenterin, sekä tiedotteet. Väreinä näissä elementeissä on käytetty ainoastaan kahta voimakkaasti murrettua väriä tekstin otsikon, kuluvan päivän ja tiedotteen lisääjän korostamiseksi. Muuten nämä pysyvät elementit ovat mustavalkoisia kontrastin, luettavuuden sekä hillityn tyylikkyyden takaamiseksi. Lisäksi näin vältetään pysyvien elementtien värien riitely taustakuvan kanssa, ja taustakuvan avulla pystytään muuttamaan koko näytönsäästäjän väriskaala.

Ulkoasussa ei ole käytetty lainkaan liikkuvia elementtejä, vaan näkymän vaihtuvuus perustuu vaihtuvaan sisältöön. Näytönsäästäjän käynnistyessä eri elementit saavat kuitenkin hieman satunnaista vaihtelua sijaintiinsa, jotta voimakkaita kontrasteja sisältävät elementit eivät olisi aina täysin samassa kohdassa. Näin pyritään välttämään vielä käytössä olevien kuvaputkimonitorien vaurioituminen pitkällä aikajänteellä.

Tiedotteiden teksti näytetään niille varatun elementin sisällä, jonka korkeus määräytyy tekstin pituuden mukaan. Tällöin tekstikenttä peittää taustakuvaa ainoastaan tarvittavan määrän verran. Tekstin ollessa pidempi kuin kerralla varattuun tilaan mahtuisi, tekstin koko skaalataan 75 %:iin alkuperäisestä. Teksti ladataan dynaamiseen tekstikenttään XML-sisältönä, jolloin tekstin muotoilu määritellään palvelimelta ladatussa CSS-tiedostossa.

KUVA 4. Näytönsäästäjäsovellus Flash Player -ikkunassa.

Laitevaatimus- ja verkonrasitusnäkökohdat

Yrityksessä on käytössä hyvin vaihteleva konekanta, joten myös vanhempien koneiden suoriutuminen näytönsäästäjän ajamisesta tuli ottaa huomioon varsinkin, kun Flash-esitystä näytetään koko näytön alalla. Liikkumattoman sisällön käyttäminen oli myös tässä suhteessa harkittu ratkaisu. Tasaisesti tai jatkuvasti liikkeessä olevan sisällön sulavuuden takaaminen kaikilla alustoilla olisi erittäin epätodennäköistä. Varsinkin läpikuultavat elementit saattaisivat muodostua raskaiksi koko näytön peittävässä animoidussa Flash-esityksessä. Yrityksen logolla varustettu nykivä näytönsäästäjä ei taatusti tuota haluttua mielikuvaa.

Vaikka näytönsäästäjässä ei käytetäkään animoituja elementtejä, jotka paljastaisivat mahdollisen hidastumisen tai nykimisen, on näytönsäästäjänä toimiva Flash-sovellus pyritty tekemään mahdollisimman kevyeksi, jotta työasemien resursseja ei käytetä tarpeettomasti lähes staattisen näytönsäästäjän piirtämiseen. Flash-player päivittää näkymää ainoastaan sisällön vaihtuessa, ja silloinkin vain muuttuneen alueen osalta. Kuvaa ei siis päivitetä jokaisessa framessa, ja mm. kellon aika päivitetään parin sekunnin välein.

Näytönsäästäjän käynnistyessä palvelimelta ladattava datan määrä on melko pieni, mikäli käytetyn taustakuvan tiedostokoko ei ole erityisen suuri. Käynnistyksen yhteydessä ladattava kokonaisdatamäärä on varsin helppo pitää alle 300 kilotavun, kunhan taustakuvan kokoon ja pakkaukseen on kiinnitetty huomiota. Nopealle lähiverkolle tämänsuuruinen datamäärä on marginaalisen pieni. Taustakuvan lisäksi käynnistyksessä ladattava datamäärä jää käytännössä hyvin pieneksi.

Tiedotteiden yhteyteen lisätyt kuvat skaalataan automaattisesti sallittuihin rajoihin ja pakataan JPEG -muotoon jo palvelimelle siirrettäessä. Kuvien pakkauksessa käytetään kuvaformaatin progressiivista pakkausta asetuksilla, jotka tähtäävät pieneen tiedostokokoon kuvan laadun kuitenkaan kärsimättä merkittävästi. Lisäksi nämä kuvat ladataan sitä mukaa, kun tiedotteita esitetään. Kertaalleen ladattua sisältöä ei ladata uudelleen ennen näytönsäästäjän uudelleen käynnistymistä.

Sisällön päivittymistä ei tarkkailla näytönsäästäjien ollessa käynnissä, mutta siltä varalta, että tietokone on pitkään yhtämittaisesti käyttämättä, näytönsäästäjänä toimiva Flash-sovellus käynnistyy uudelleen ennalta määritellyn ajan jälkeen. Itse näytönsäästäjä ei sulkeudu tämän tapahtuessa.

6.6 Järjestelmän käyttöönotto

Järjestelmän palvelinpuolen toiminta kehitettiin alusta alkaen palvelimella, jolla lopullinen järjestelmä on käytössä. Näytönsäästäjän asennuspaketti luotiin InstantStorm 1.0 -apuohjelmalla. Näytönsäästäjän asennuspaketti ajettiin automatisoidusti verkon koneissa taustalla tapahtuvana asennuksena, joka ei vaadi koneen käyttäjältä toimenpiteitä. Näytönsäästäjien asennuspaketin jakelu ja suorittaminen, sekä Windowsin näytönsäästäjän asetusten asettaminen suoritettiin Windows-verkoissa käytettävällä Group Policy -järjestelmällä, jonka avulla Windows-verkossa olevien koneiden ohjelmistokokoonpanoa sekä asetuksia voidaan hallita keskitetysti.

6.7 Järjestelmän jatkokehitys

Työn kirjoitusvaiheessa sisällönhallintajärjestelmästä puuttui käytön ohjeistus. Järjestelmään on tarkoitus toteuttaa käyttöohje, joka on saatavilla Internet-selaimen välityksellä järjestelmän käytön yhteydessä. Ohje on kaavailtu toteutettavan niin, että ohje on avattavissa kaikkien käyttäjälle avointen toimintojen yhteydessä, jolloin ohje avautuu automaattisesti oikeasta kohdasta.

Käyttöönotettaessa järjestelmä mahdollisti tiedotteiden näkymisaikavälin määrittelyn ainoastaan päivämäärien perusteella. Tämän ominaisuuden laajentaminen ottamaan huomioon myös kellonajan saattaa osoittautua tarpeelliseksi. Työtä kirjoitettaessa järjestelmän käytön kautta nousseita parannus- tai jatkokehitysajatuksia ei ilmaantunut.

Järjestelmän ylläpidettävyys paranisi, jos kaikkia toimintaan liittyviä asetusmuuttujia voitaisiin hallita hallintajärjestelmän kautta ilman, että mahdollisia muutoksia olisi tarpeen tehdä suoraan PHP- tai ActionScript -koodiin. Tämä ei kuitenkaan ole välttämättä tarpeen, sillä palvelinohjelmoinnilla toteutettujen toimintojen merkittävimmät ja todennäköisimmin muutoksia kaipaavat muuttujat on kerätty omaan konfiguraatiotiedostoonsa. Myös muutokset näytönsäästäjäsovelluksiin on melko yksinkertaista toteuttaa ainoastaan korvaamalla palvelimelta ladattava Flash-tiedosto.

6.8 Casen arviointia

Peikko Finland Oy:lle toteutettu näytönsäästäjätiedotusjärjestelmän ensimmäinen versio valmistui odotetun aikataulun mukaisesti sisältäen kaikki käyttöönottoon edellyttävät ominaisuudet. Järjestelmä koostuu sisällön hallintaan käytettävästä selainpohjaisesta hallintajärjestelmästä sekä yrityksen koneisiin asennettavista näytönsäästäjistä, jotka hakevat näytettävän sisällön palvelimelta käynnistyessään. Käyttöönotettaessa järjestelmä mahdollisti tiedotteiden lisäämisen järjestelmään, sekä lisättyjen tiedotteiden muokkaamisen ja poistamisen. Tiedotteen yhteyteen voi halutessaan lisätä kuvan, tiedotteen näkyvyysaikavälin voi määritellä päivämäärien perusteella ja tiedotteen näyttämiseen käytettävää aikaa voi halutessaan muuttaa. Lisäksi tiedotteen näkyvyysaluetta on mahdollista rajata yrityksen verkon eri osissa. Myös näytönsäästäjien taustakuvia sekä käyttäjiä hallitaan järjestelmän kautta. Hallintajärjestelmän käyttöä hallitaan ja rajataan järjestelmään kirjautumisen perusteella. Olen itse tyytyväinen lopputulokseen, ja mielestäni case täyttää merkittävimmät sille asetetuista tavoitteista.

Peikko Finland Oy:lle toteutettu näytönsäästäjätiedotusjärjestelmä on otettu yrityksessä positiivisesti vastaan. Myös osana casea toteutettu yrityskalenteri on kerännyt laajalti kiitosta. Peikko Finland Oy:n laatu- ja ympäristöpäällikkö Outi Savolainen kommentoi tiedotusjärjestelmää seuraavalla tavalla:

> *Kysymykseen "Mitä yrityksessämme voisi parantaa?", vastaa suurin osa vastaajista yrityksestä ja organisaatiotasosta riippumatta "tiedotusta". Tähän ikuiselta tuntuvaan ongelmaan innostuimme hakemaan ratkaisua näytönsäästäjällä, joka sisältää myös perinteisen inforuutuajatuksen. Uusi näytönsäästäjä onkin otettu ilolla vastaan niin tiedotuksesta vastaavien kuin tiedonjanoisten peikkojen keskuudessa. Jatkossa pääsemme mittaamaan uuden näytönsäästäjämme vaikutusta henkilöstömme mielipidekartoituksessa, jossa odotamme tiedotuksen arvosanan huomattavaa paranemista.*

Itse koin casen toteutuksen kiinnostavana ja inspiroivassa mielessä haasteellisena. Työn toteutus kokonaisuutena koostui lukuisista osatekijöistä, joista useimmista minulla ei ollut kovinkaan paljoa, jos lainkaan, edeltävää kokemusta. Ainoastaan kuvamateriaalin toteutuksessa vaaditusta suhteellisen vaativasta kuvankäsittelystä omasin jo lähtökohtaisesti kattavan osaamisen. Kalenterin kuvien toteutukseen vaadittu Maxin hyödyntäminen laajensi huomattavasti muuten varsin vähäisiä mallinnus- ja renderöintitaitojani.

Tiedotusjärjestelmän sisällönhallinnan käyttöliittymän sekä sovelluslogiikan toteuttaminen PHP-palvelinohjelmoinnilla edellytti merkittävästi aiempaa kokemustani laajempaa perehtymistä PHP-ohjelmointiin sekä etenkin MySQL-tietokannan hyödyntämiseen PHP:n kautta.

Näytönsäästäjäsovelluksen toteuttaminen Flashilla opetti minulle erityisesti ulkoisen datan hyödyntämistä Flashissa. Sovelluksen toteutus oli erittäin ActionScript-ohjelmointipainotteinen aiemman Flash-kokemukseni ollessa enemmän ulkoasun ja animaation toteutukseen suuntautunutta. Käyttötarkoituksensa kannalta myös sovelluksen keveyteen, ja erityisesti sen saavuttamiseen oli kiinnitettävä huomiota.

Muista työssä käytetyistä tekniikoista sisällönhallintajärjestelmän käyttöliittymässä hyödynnetty JavaScript sekä tiedon siirtoon käytetty XML olivat molemmat minulle ennalta täysin vieraita käytännön tasolla.

Kaikkiaan työn toteutus laajensi osaamistani valtaisasti työssä hyödynnettyjen erinäisten tekniikoiden ja menetelmien sekä ohjelmistojen tuntemuksessa sekä hyödyntämisessä.

7 YHTEENVETO

Vaikka näytönsäästäjät ovat menettäneetkin alkuperäisen merkityksensä näyttöä säästävinä sovelluksina, tarjoavat ne silti edelleen mahdollisuuksia toimia hyötysovelluksina. Näytönsäästäjät ovat erityisesti potentiaalisia välineitä informaation välitykseen. Näytönsäästäjät soveltuvat erityisen hyvin tiedotuskanavan toteuttamiseen, sillä näin toteutettuna tiedotuskanava ei vaadi kohdeyleisöltään aktiivista seuraamista. Tiedotuskanavana toimiva näytönsäästäjä on esillä, kun pääte ei ole aktiivisessa käytössä.

Tiedotuskanavana toimivat näytönsäästäjät vaativat rinnalleen sisällönhallintajärjestelmän, jotta useiden näytönsäästäjien esittämä sisältö olisi keskitetysti hallittavissa. Sen lisäksi, että tiedotusjärjestelmänä toimivat näytönsäästäjät vaativat ulkoisen lähteen näytettävälle sisällölle, on jonkinlainen sisällönhallintajärjestelmä syytä toteuttaa myös järjestelmään sisältöä lisäävien käyttäjien avuksi. Helppokäyttöinen sisällön hallintaan tarkoitettu järjestelmä on tarpeellinen etenkin, jotta käyttäjien ei tarvitse tuntea järjestelmässä käytettävää tiedon tallennusmuotoa tai järjestelmän toimintaa tekniseltä kannalta. Kun

järjestelmän tekninen toteutus on käyttäjiltä näkymättömissä ja käyttäjät huolehtivat ainoastaan sisällöstä, vältytään käyttäjien aiheuttamilta toimintahäiriöiltä.

Hallintajärjestelmän toteuttamiseen ja sisällön keskitettyyn jakamiseen on verkkopalvelimella toimiva palvelinsovellus erittäin toimiva ratkaisu. Sisällön tallentamiseen ja käsittelyyn voidaan näin käyttää palvelimen tekniikoita. Järjestelmään tallennettu sisältö voidaan tallentaa esimerkiksi tietokantaan, jolloin sen käsittely on nopeaa, helppoa sekä monipuolista. Tietokantoihin tallennettu sisältö voidaan lajitella ja muuntaa kulloinkin tarvittuun muotoon palvelinohjelmointia hyödyntäen. Sekä sisällön jakelu että hallinta voidaan toteuttaa samoilla palvelinohjelmointitekniikoilla.

Työtä tehdessä Flash osoittautui erittäin sopivaksi välineeksi dynaamista informaatiosisältöä esittävän näytönsäästäjäsovelluksen toteuttamiseen. Flash on monipuolinen sovelluskehitystyökalu, joka mahdollistaa sekä visuaalisen että teknisen toteutuksen. Flash tarjoaa myös hyvän tuen ulkoisen ja dynaamisen sisällön hyödyntämiseen. Erityisesti mahdollisuus käyttää kokonaan ulkoiseen järjestelmään tallennettua tai dynaamisesti generoitua sisältöä sekä XML-muotoon tallennettua informaatiota tekivät Flashista sopivan työkalun informatiivisen näytönsäästäjän toteutukseen. XML-muotoon generoidun informaation käyttäminen tiedon siirtoon palvelimen ja näytönsäästäjien välillä osoittautui luontevaksi ratkaisuksi. XML mahdollistaa informaation monipuolisen rakenteen, jolloin sekä itse sisältö, että sisältöön liittyvä informaatio oli mahdollista siirtää yhtenä XML-muotoisena dokumenttina. Myös XML-muotoisen sisällön käsiteltävyys Flashissa osoitti XML:n soveltuvuuden tiedonsiirtovälineenä.

Flashin käyttökelpoisuutta näytönsäästäjän toteutukseen heikentää lähinnä se, että toteutettu näytönsäästäjäsovellus täytyy muuntaa lopulliseen käyttöjärjestelmän hyväksymään näytönsäästäjämuotoon jollakin menetelmällä. Käytännössä tämä ei ole suuri ongelma, sillä kaikki toiminnallisuus voidaan toteuttaa itse Flashilla. Lopullinen näytönsäästäjäsovellus tarvitaan vain käynnistämään toteutettu Flash-sovellus. Flash-esityksen käynnistävä näytönsäästäjäsovellus voidaan ohjelmoida itse, tai vaihtoehtoisesti voidaan käyttää myös jotakin lukuisista saatavilla olevista apuohjelmista.

Tämä työ osoittaa, että näytönsäästäjät soveltuvat mainiosti käytettäväksi tiedonvälitykseen. Erityisesti yrityksen sisäiseksi muiden informaatiokanavien rinnalla toimivaksi tiedotusvälineeksi näytönsäästäjien avulla toteutettu jär-

jestelmä soveltuu hyvin. Työssä myös todettiin Flashin soveltuvan erittäin hyvin informatiivisen näytönsäästäjän toteuttamiseen. Työn case osoitti, että yhdistämällä Flash ja palvelinohjelmointi on mahdollista toteuttaa joustava ja toimiva näytönsäästäjien välityksellä toimiva tiedotusjärjestelmä. Casen toteutuksessa hyödynnettiin ainoastaan erittäin yleisesti käytettyjä tekniikoita, joten vastaavanlainen järjestelmä on mahdollista toteuttaa hyvinkin tyypillisten sekä erilaisten palvelininfrastruktuurien ympärille. Voidaankin todeta, että näytönsäästäjien välityksellä toteutettu informaatiokanava on erittäin varteenotettava vaihtoehto yrityksen sisäisen tiedonkulun tehostamiseen.

LÄHTEET

PAINETUT LÄHTEET

Hermans, P, North, S. 2001. XML Trainer Kit. Helsinki: Oy Edita Ab

Holzner, S. 2001. Inside XML. Helsinki: Edita Oyj

Laakkonen, A, Walkama, P. 2004. Inside XML-Skeema. Helsinki: Edita Publishing Oy

Manninen, P, Marttila, J. 2006. Flash 8 & Actionscript. Jyväskylä: Docendo Finland Oy

Nykänen, O. 2001. XML. Jyväskylä: Docendo Finland Oy

Rantala, A. 2002. PHP. Jyväskylä: Docendo Finland Oy

Rantala, A. 2005. Web-ohjelmointi. Jyväskylä: Docendo Finland Oy

SÄHKÖISET LÄHTEET

Extensible Markup Language (XML) [verkkodokumentti]. W3C, 11.9.2006 [viitattu 25.3.2007]. Saatavissa: http://www.w3.org/XML/

Flash Player Penetration [verkkodokumentti]. Adobe Systems Inc. [viitattu 25.3.2007]. Saatavissa: http://www.adobe.com/products/player_census/flashplayer/

Flash Professional 8: Datasheet [verkkodokumentti]. Adobe Systems Inc. [viitattu 4.3.2007]. Saatavissa: http://www.adobe.com/products/flash/flashpro/productinfo/datasheet/

How to transfer variables from url string to Flash movie [verkkodokumentti]. Polar-lights.com, 21.8.2006 [viitattu 21.1.2007]. Saatavissa: http://www.polar-lights.com/fla/url.html

Nykänen, O. 28.8.2003, XML 10 kohdan tiivistelmänä [viitattu 15.1.2007]. Saatavissa: http://www.w3c.tut.fi/translations/xml/xmlin10pts/

PHP [verkkodokumentti]. Wikipedia.org, 23.1.2007 [viitattu 1.2.2007]. Saatavissa: http://fi.wikipedia.org/wiki/Php

Pilgrim, M. What is RSS [verkkodokumentti]. XML.com, 18.12.2002 [viitattu 1.2.2007]. Saatavissa: http://www.xml.com/pub/a/2002/12/18/dive-into-xml.html

Popescu, R. Simple Windows Screensaver [verkkodokumentti]. CoderSource.net, 6.9.2005 [viitattu 2.3.2007]. Saatavissa: http://www.codersource.net/csharp_screen_saver.aspx

RSS-ohje [verkkodokumentti]. MikroPC.net, 7.10.2006 [viitattu 1.2.2007]. Saatavissa: http://mikropc.net/ohjeet/rss.html

RSS (file format) [verkkodokumentti]. Wikipedia.org, 31.1.2007 [viitattu 1.2.2007]. Saatavissa: http://en.wikipedia.org/wiki/RSS_%28file_format%29

Screensaver [verkkodokumentti]. Wikipedia.org, 24.1.2007 [viitattu 1.2.2007]. Saatavissa: http://en.wikipedia.org/wiki/Screen_saver

Solis, J. Loading data into Flash [verkkodokumentti]. The Flash-DB [viitattu 14.1.2007]. Saatavissa: http://www.flash-db.com/Tutorials/loading/

Statler, T. Creating a JPEG slide show with XML: About XML [verkkodokumentti]. Adobe Systems Inc., 25.6.2002 [viitattu 25.3.2007]. Saatavissa: http://www.adobe.com/support/flash/applications/jpeg_slideshow_xml/jpeg_slideshow_xml04.html

support.microsoft.com, Maximum URL length is 2,083 characters in Internet Explorer [verkkodokumentti]. Microsoft Corporation, 1.12.2006 [viitattu 2.2.2007]. Saatavissa: http://support.microsoft.com/kb/208427

URL Encoding: Reading special characters from a text file [verkkodokumentti]. Adobe Systems Inc., 6.1.2005 [viitattu 2.2.2007]. Saatavissa: http://www.adobe.com/cfusion/knowledgebase/index.cfm?id=tn_14143

Uutissyötteet (RSS) [verkkodokumentti]. Jyväskylän yliopisto, Koulutuksen tutkimuslaitos, Peda.net, 22.1.2007 [viitattu 1.2.2007]. Saatavissa: http://www.peda.net/veraja/kotka/aikuislukio/ideat/rss

W3C Suomen toimisto [verkkodokumentti]. W3C Suomen toimisto, 13.1.2007 [viitattu 31.1.2007]. Saatavissa: http://www.w3c.tut.fi/

Web feed [verkkodokumentti]. Wikipedia.org, 20.1.2007 [viitattu 1.2.2007]. Saatavissa: http://en.wikipedia.org/wiki/Web_feed

Web syndication [verkkodokumentti]. Wikipedia.org, 28.1.2007 [viitattu 1.2.2007]. Saatavissa: http://en.wikipedia.org/wiki/Web_syndication

What is RSS [verkkodokumentti]. RSS Specifications [viitattu 24.3.2007]. Saatavissa: http://www.rss-specifications.com/what-is-rss.htm

What you can do with Flash [verkkodokumentti]. Livedocs.macromedia.com [viitattu 20.1.2007]. Saatavissa: http://livedocs.macromedia.com/flash/8/main/00000016.html

Winer, D. RSS 2.0 Specification [verkkodokumentti]. blogs.law.harvard.edu, 30.1.2005 [viitattu 1.2.2007]. Saatavissa: http://blogs.law.harvard.edu/tech/rss

XML [verkkodokumentti]. Wikipedia.org, 14.1.2007 [viitattu 2.2.2007]. Saatavissa: http://fi.wikipedia.org/wiki/XML

KUVALÄHTEET

KUVA 1. Rantala, A. 2005. Web-ohjelmointi. Docendo Finland Oy, Jyväskylä, sivu 2

KUVA 2. Ruudunkaappaus casessa toteutetusta sisällönhallintajärjestelmästä.

KUVA 3. Ruudunkaappaus casessa toteutetusta sisällönhallintajärjestelmästä.

KUVA 4. Ruudunkaappaus casessa toteutetusta näytönsäästäjäsovelluksesta.

LIITTEET

Liite 1: Kuvia osana casea toteutetusta yrityskalenterista.

Liite 2: CD-rom, sisältää casena toteutetun näytönsäästäjätiedotusjärjestelmän Flashilla toteutetun näytönsäästäjäsovelluksen sekä työn sähköisiä lähteitä.