Juha Aalto

Dynaamisen verkkosovelluksen kehitysprosessi

Metropolia Ammattikorkeakoulu
Insinööri (amk)
Tietotekniikka
Insinöörityö
23.3.2011

Metropolia

Tiivistelmä

Tekijä Otsikko Sivumäärä Aika	Juha Aalto Dynaamisen verkkosovelluksen kehitysprosessi 38 sivua + 2 liitettä 23.3.2011
Tutkinto	insinööri (amk)
Koulutusohjelma	tietotekniikka
Suuntautumisvaihtoehto	ohjelmistotekniikka
Ohjaaja	yliopettaja Jaana Holvikivi

Insinöörityön tavoitteena oli luoda muusikko Antti Kuukalle yleiskäyttöinen dynaaminen WWW-sovellus tekemänsä musiikin ja omien ajatustensa julkituontiin. Päämääränä oli nivoa yhteen sosiaalisesta mediasta tuttuja toimintoja, mutta samalla toteuttaa ne asiakkaan vaatimusten ja graafisten mieltymysten mukaisesti. Opinnäytetyö syntyi tarpeesta ottaa kokonaisvaltaisesti haltuun henkilökohtaisen web-sivun suunnittelu- ja toteutusprosessi.

Sovellus jakaantui kaikille avoimeen yleiseen osaan ja asiakkaan käytössä olevaan hallintasivustoon. Työssä kiinnitettiin huomiota sekä teknisiin ratkaisuihin että sovelluskehitysprosessin vaiheisiin ja menetelmiin, joiden teoriaan syvennytään myös tässä raportissa. Erityinen painoarvo on annettu käytettävyyssuunnittelun merkitykselle sovelluskehityksen osana.

Sovelluksen toteutuksen ja ylläpidon kustannukset haluttiin minimoida, minkä vuoksi kaikki työkalut ja tekniikat ovat ilmaisia ja avoimeen lähdekoodiin perustuvia. Sekä kehitys- että tuotantoympäristössä käytettiin Apachen HTTP-palvelinohjelmaa ja SQLite-tietokantaa. Sivuston rakenteen ja toiminnallisuuksien toteuttamiseen käytettiin PHP-, XHTML- ja CSS-ohjelmointi- ja merkintäkieliä sekä soittolistassa XML-pohjaista XSPF-tiedonkuvausformaattia. Tietokannaksi valittiin pienikokoinen ja kevyt SQLite, jota PHP tukee natiivisti versiosta 5 eteenpäin, sillä se ei vaadi erillistä tietokantapalvelinta ja on siksi resurssiystävällinen sekä helposti varmuuskopioitavissa.

Vaatimus- ja informaatioarkkitehtuurimäärittelyn pohjalta käynnistynyt sovelluskehitys-prosessi toteutettiin inkrementaalista ja iteratiivista metodia hyödyntäen. Metodien joustavuuden ansiosta asiakkaan toivomat muutokset ja lisäykset olivat helposti toteutettavissa riippumatta projektin etenemisvaiheesta. Lopputuloksena oli kokonaisuutena eheä sovellus, jonka yleiskäyttöistä runkoa voidaan hyödyntää myös tulevissa projekteissa.

Avainsanat	WWW-suunnittelu, PHP, XHTML, käytettävyys

Metropolia

Abstract

Author(s) Title Number of Pages Date	Juha Aalto The development process of a dynamic web application 38 pages + 2 appendices 23 March 2011
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructor(s)	Jaana Holvikivi, Principal Lecturer

The purpose of this thesis was to create a dynamic web application for musician Antti Kuukka to promote his music and bring forth his thoughts. The primary objective was to bundle together many familiar features of social media, yet implementing them in respect to the client's requirements and predilections for graphic design. The thesis was born out of the need to take comprehensive possession over the process of web development and design.

The application consists of two separate sections, the public site open for all visitors and the administrator pages accessible only by the client. Attention was paid to both the technical decisions and the stages as well as the methods of the application development process. The theory section in the report discusses this development process and also takes into consideration the important aspects of usability.

An attempt was made to keep the expenses of the development process to a minimum; hence all the chosen tools and technologies were free and based on open source software. Both production and development environments were using the Apache HTTP server software and SQLite databases. The application structure and functionalities were done using PHP, XHTML and CSS programming and markup languages and the playlist utilized a XML-based XSPF information description format. The SQLite database, supported by PHP since version 5, was chosen because it does not require a separate database server, making it a lightweight and easy-to-backup storage system.

Incremental and iterative development methods were used throughout the project. The flexibility of the methods ensured easy modifications to the application regardless of its development stage. The outcome of this thesis was a comprehensive web site to suit the needs of the client, and also a functional framework to utilize in future projects.

Keywords	web development, PHP, XHTML, usability

Sisällys

Lyhenteet

CSS	*Cascading Style Sheets.* Tyylitiedosto, jolla määritellään HTML-dokumentin ulkoasu.
DOM	*Document Object Model.* XML-pohjaisten dokumenttien muokkauksen mahdollistava ohjelmointirajapinta.
GD-kirjasto	*Graphics Draw*. Kuvien muokkaukseen tarkoitettu ohjelmistokirjasto.
HTML	*Hypertext Markup Language.* Kuvauskieli tekstin rakenteen merkintään.
IDE	*Integrated Development Environment.* Sovellus, joka tarjoaa ohjelmoijalle monipuoliset työkalut ohjelmien kehittämiseen.
JavaScript	Komentosarjakieli, jolla voidaan luoda WWW-sivuille dynaamista toiminnallisuutta.
MIME-tyyppi	*Multipurpose Internet Mail Extensions.* Sisältötyyppi, joka ilmaisee HTTP:ssä välitetyn datan muodon.
PHP	*PHP: Hypertext Preprocessor.* Palvelinpohjainen ohjelmointikieli dynaamisten web-sovellusten luontiin.
RSS	*Really Simple Syndication.* Verkkopalvelun tarjoama XML-tiedosto, joka sisältää toistuvasti päivittyvää sisältöä.
SHA-1	*Secure Hash Algorithm.* Tiivistefunktio, joka tuottaa mistä tahansa datasta 40 merkkiä pitkän tiivisteen.
SQLite	Kevyt relaatiotietokanta, joka ei tarvitse erillistä tietokantapalvelinta.
SSL	*Secure Sockets Layer*. Internet-sovellusten tietoliikenteen salausprotokolla. TLS:n edeltäjä.
TLS	*Transport Layer Security*. Internet-sovellusten tietoliikenteen salausprotokolla. SSL:n seuraaja.
UTF-8	*Universal Character Set Transformation Format — 8-bit.* Merkistöstandardi, joka käsittelee merkkejä kahdeksanbittisinä tavuina.
W3C	*World Wide Web Consortium.* Internetin standardeja kehittävä ja ylläpitävä järjestö
XHTML	*eXtensible Hypertext Markup Language.* WWW-sivujen merkintäkieli, joka yhdistää HTML:n ja XML:n ominaisuuksia.
XML	*eXtensible Markup Language.* Merkintäkieli, jonka avulla tiedon merkitys on kuvattavissa tiedon sekaan.
XSPF	*XML Shareable Playlist Format.* XML-pohjainen soittolistaformaatti.

1 Johdanto

Internet on pohjimmiltaan kommunikaatiota yksilöiden, yhteisöjen ja organisaatioiden välillä. Web-teknologiat ovat vain viestinvälityksen rakennuspalasia. Suunnittelijan on kyettävä näkemään verkkosovelluksen todellinen päämäärä ja tarkoitus teknisten ratkaisujen takana. On tärkeää löytää sivustolle sen tarkoituksen ja kohdeyleisön mukainen luonne. Itseisarvona ei ole visuaalisesti loistelias sovellus, vaan dialogi yleisön kanssa.

Onnistunut kommunikaatio edellyttää sovelluskehitysprosessin osa-alueiden hallintaa. On huomattava, että web-suunnittelu ei ole pelkästään teknologiaosaamista, vaan sisällöllä, sisällön rakenteellisuudella, visuaalisella suunnittelulla sekä sovelluksen ja käyttäjän välisellä interaktiolla on kullakin oma tärkeä roolinsa prosessissa.

Opinnäytetyön toimeksiantaja, muusikko Antti Kuukka, halusi hyödyntää Internetiä julkaisufoorumina saadakseen äänensä kuuluviin, luodakseen kontakteja sekä yhdistääkseen ystäviään ja muita samanhenkisiä ihmisiä. Tästä tarpeesta syntyi Saunantakana.fi, perinteisen kotisivun, blogin, gallerian ja vieraskirjan yhdistelmä. Opinnäytetyön raportissa kuvataan dynaamisen verkkosovelluksen kehitysvaiheet suunnittelusta toteutukseen ja paneudutaan erityisesti käytettävyyden ja visuaalisuuden näennäisen dikotomian ratkaisemiseen. Koska tausta-ajatuksena oli tutustua laaja-alaisesti verkkosovelluksen luomisprosessiin ja selvittää jokaisen osa-alueen vaikutus projektin kokonaiskuvaan, pyritään näitä lähtökohtia tuomaan esille mahdollisimman monipuolisesti myös tässä raportissa.

Sovelluksen toteutuksen ja ylläpidon kustannukset haluttiin minimoida, minkä vuoksi kaikki työkalut ja tekniikat ovat ilmaisia ja avoimeen lähdekoodiin perustuvia. Sekä kehitys- että tuotantoympäristössä käytetään Apachen HTTP-palvelinohjelmaa ja SQLite-tietokantaa. Sivuston rakenteen ja toiminnallisuuksien toteuttamiseen käytetään PHP-, XHTML- ja CSS-ohjelmointi- ja merkintäkieliä sekä soittolistassa XML-pohjaista XSPF-tiedonkuvausformaattia.

Sovellus jakautuu kahteen osaan, kaikille avoimeen perussivustoon ja ylläpitäjälle tarkoitettuun hallintasivuun. Huolellisella käytettävyyssuunnittelulla haluttiin taata ylläpitäjän toimintojen loogisuus sekä perussivuston selkeys, joka omalta osaltaan houkuttelee kävijöitä palaamaan sivustolle uudelleen. Lisäksi sovellukseen tehtiin uutisosion RSS-syötetoiminto kohderyhmään kuuluvien, sivuilla jo vierailleiden käyttäjien paluun tehostamiseksi sekä vieraskirjan kaltainen palvelu, jolla vierailijoiden ja omistajan välistä interaktiota saatiin lisättyä.

2 Ohjelmointikielet ja -tekniikat

XML

XML (eXtensible Markup Language) on merkintäkieli, jolla kuvataan tietoa konekielisessä muodossa. XML-dokumentti koostuu elementeistä, jotka sisältävät itse tiedon ja tätä tietoa kuvaavan merkinnän, tagin. XML:n suunnittelussa on tähdennetty yksinkertaisuutta ja yleiskäyttöisyyttä, minkä vuoksi dokumentin oikeellisuus voidaan todentaa helposti kahden määritelmän avulla: dokumentin on oltava sekä hyvin muodostettu (Well-formed), jolloin se täyttää kaikki syntaksisäännöt, että validi (Valid), jolloin XML-dokumentti on jonkin dokumenttityypin mukainen. Hyvinmuodostetussa dokumentissa tulee olla vain yksi juurielementti, juurielementin sisällä olevat elementit eivät voi mennä ristiin toistensa kanssa ja jokaisella elementillä on oltava sekä alku- että loppumerkki. [1.]

XHTML

HTML (Hypertext Markup Language) on WWW-sivujen kuvaukseen tarkoitettu merkintäkieli. XHTML (eXtensible Hypertext Markup Language) on yleisnimitys HTML 4.01 -standardin jälkeisille versioille, joissa käytetään XML-kielen mukaista tiukempaa kielioppia. Selaimet tulkitsevat HTML:ää hyvin eri tavoin ja toisinaan määritelmien vastaisesti, mikä hankaloittaa kehitystyötä sekä selaimien välisen yhteensopivuuden syntyä. Tärkein syy XHTML:n kehittämiseen olikin yksikäsitteisen notaation luominen, joka parantaa toimintavarmuutta myös HTML:n monitulkintaisuudesta kärsiville uudemmille medialaitteille (matkapuhelimet, kannettavat päätelaitteet ja televisiot). [2.]

Tässä työssä käytetään XHTML 1.0 Strict -standardin mukaista merkkausta, joka on World Wide Web Consortiumin (W3C) suosittelema vaihtoehto. Strict-versiossa merkkaus on pääosin vain rakennetta kuvailevaa, jolloin dokumentin ulkoasun muutokset on tehtävä erillisillä välineillä, usein CSS:llä. [3, s. 141.]

CSS

CSS (Cascading Style Sheets) on erityisesti WWW-dokumenteille, mutta myös muille rakenteellista kuvauskieltä sisältäville dokumenteille (XML, MathML) kehitetty ulkoasua ja esitystapaa (asettelu, värit, fontit ja kuvat) määrittelevä tyylikieli. CSS:n tarkoituksena on erillisten muotoiluelementtien poistaminen itse kuvauskielestä, jolloin alkuperäinen dokumentti pysyy vain rakenteellisen informaation kuvauksena. Tässä mielessä CSS:ää voidaan pitää metatietona, sillä se jäsentää vain sen, miten itse tieto tulisi esittää.

Rakenteen erottaminen selkeyttää dokumenttien sisältöä, vähentää toistuvien tyylimääreiden vaatimaa tilantarvetta sekä tarjoaa joustavuutta ja yleiskäyttöisyyttä. Samaa tyylitiedostoa voidaan käyttää useammissa dokumenteissa tai vastaavasti käyttää näyttölaitekohtaisia tyylimäärittelyjä. [4, s. 27.]

PHP

PHP (rekursiivinen lyhenne sanoista PHP: Hypertext Preprocessor) on palvelinpohjainen komentosarjakieli, jota käytetään erityisesti dynaamisten WWW-sivujen luonnissa. PHP-koodi voidaan upottaa suoraan HTML:n joukkoon, jolloin PHP-moduulilla varustettu WWW-palvelin tulkitsee dokumentin ja tuottaa tästä web-sivun. Koska koodi suoritetaan jo palvelimella ja tuloksena on normaali HTML-sivu, ei PHP:n käyttö vaadi erityistä tukea selaimelta ja on täten toimintavarma sekä asiakkaan kannalta kevyt käyttää. Etuna on myös ohjelman pääsy palvelimella oleviin tietokantoihin ja tiedostoihin. PHP perustuu avoimeen lähdekoodiin ja sen käyttö on ilmaista. [5.] Palvelinpohjaisista komentosarjakielistä PHP on ylivoimaisesti yleisin: arvion mukaan vuoden 2011 alussa lähes 80 % kaikesta palvelinpuolen ohjelmoinnista on toteutettu PHP:llä [6].

SQLite

SQLite on hyvin kompakti C-ohjelmointikielellä toteutettu relaatiotietokantajärjestelmä. Toisin kuin monet muut tietokantajärjestelmät, SQLite sijaitsee samassa paikassa sitä käyttävän järjestelmän kanssa eikä täten tarvitse erillistä palvelinta toimiakseen. Täysin toimintakuntoisena SQLite-kirjasto vie levytilaa alle 275 kilotavua tehden siitä pienimmän kokonaisvaltaisen tietokantajärjestelmän. Tietokanta on alustariippumaton

ja se tallennetaan kokonaisuudessaan yhteen tiedostoon, mikä mahdollistaa helpon siirrettävyyden ja varmuuskopioinnin laitteesta toiseen. SQLite käyttää niin sanottua dynaamista tietotyyppijärjestelmää, jossa tauluun tallennettavan tiedon tyyppi voidaan valita erikseen jokaiselle arvolle, toisin kuin useimmissa muissa tietokantajärjestelmissä, joissa tietotyyppi määritellään sarakekohtaisesti. Myöskään sarakeleveyttä ei tarvitse etukäteen määritellä, sillä tietokanta käyttää muistia vain sen verran kuin siihen tallennettu tieto edellyttää. [7.]

3 Suunnittelu

3.1 Vaatimusten määrittely ja informaatioarkkitehtuuri

Porvoolainen musiikin sekatyöläinen Antti Kuukka tarvitsi itselleen internetsivut tekemänsä musiikin promotoimiseen, tarjotakseen äänitys- ja miksauspalveluita sekä tuodakseen julki omia ajatuksiaan. Päämääränä oli tavoittaa samanhenkisiä ihmisiä ja tehdä sivuista heidän kohtauspaikkansa. Sosiaalinen media monine sovelluksineen tarjoaa toki nykyisellään useita vaihtoehtoja ajatusten ja musiikin julkituomiseen, mutta haittapuolena ovat yleensä oman päätäntävallan rajoitukset etenkin graafisten elementtien, mainosten ja toiminnallisuuksien osalta. Saunantakana.fi-sivusto syntyi tarpeesta ottaa kokonaisvaltaisesti haltuun henkilökohtaisen web-sivun suunnittelu- ja toteutusprosessi.

Asiakkaalla on perustason tietotekninen tietämys, toisin sanoen hän on tottunut käyttämään erilaisia, pääosin musiikin tekemiseen liittyviä ohjelmia, kuten moniraituritallentimia, äänieditoreja ja syntetisaattoreita. Tietoteknistä koulutusta hänellä ei ole, joten ohjelmien suhteen hän on itseoppinut. Tämän tiedon pohjalta tehtiin suunnitelmia sivuston käytettävyydestä ja toiminnallisuuksista. Käyttöliittymästä ei kannattanut tehdä liian yksinkertaistettua ja toiminnoiltaan karsittua, sillä se lopulta vain turhauttaisi käyttäjää. Suunnittelussa oli otettava huomioon nykyinen ja tulevaisuudessa mahdollisesti kasvava tietotaito.

Ennen suunnittelun aloitusta on selvitettävä mahdollisimman tarkasti asiakkaan toiveet ja näkemykset sivuston toiminnoista ja rakenteesta. Tarvekartoituksen avulla sovelluksen tekijä pystyy paremmin hahmottamaan käytettävät tekniikat ja komponentit sekä saa yleiskuvan kokonaisuutena toimivasta lopputuloksesta. Kartoituksen myötä on jo projektin alkuvaiheessa mahdollista identifioida sisällön ja sivuinfrastruktuurin vahvuudet ja heikkoudet, määrittää sovelluksen sisältö loogisiin ryhmiin sekä saada alustava kokonaiskuva projektin laajuudesta ja ajankäytöstä.

Informaatioarkkitehtuuriprosessin tulee käynnistyä jo ennen tarvekartoituksen valmistumista. Tämän prosessin aikana sovelluksen tuleva sisältö identifioidaan, järjestetään ja kootaan selkeiksi kokonaisuuksiksi. Informaatioarkkitehtuuri ei koske

vain loppukäyttäjälle näkyvää tietoa vaan myös teknisiä ratkaisuja ja toteutusmalleja. Harkittu rakenne mahdollistaa sovelluksen vakaan ja virheettömän toiminnan sekä muodostaa pohjan tarkoituksenmukaisen ja loogisen käyttöliittymän suunnitteluun.

Tietosisällön konseptuaalisella ja teknisellä suunnittelulla on suora vaikutus sovelluksen lähitulevaisuuden ja pidemmän aikavälin kehitykselle. Käytettävissä olevan informaation järjestäminen sekä loppukäyttäjälle että kehittäjälle selkeiksi kokonaisuuksiksi auttavat osaltaan välttämään vakavimmat ongelmat sovelluksen elinkaaren myöhemmissä vaiheissa. Projektin päätavoitteet on syytä määritellä heti alussa, jotta suunnittelun ja toteutuksen fokus pysyy tärkeimmissä seikoissa. [8, s. 52.]

Saunantakana.fi-projektin asiakas halusi sovellukseen blogi-tyyppisen uutisosion, biografian, kotistudion kaluston esittelysivun, musiikin, kuvien ja sanoitusten julkaisuosion, vieraskirjan sekä linkkilistan ja yhteystiedot. Kaikkien näiden osa-alueiden sisältöä tuli kyetä muokkaamaan erityisten ylläpitosivujen kautta. Päätavoitteet jakaantuivat kolmeen jossain määrin erilliseen yksikköön: vierailijoiden näkemään normaaliin sivuun, asiakkaan käyttämään ylläpitosivuun sekä tietokantaan, johon molemmat edellä mainitut ovat yhteydessä.

Sovelluksen vaatimuksia määriteltäessä tulee ottaa huomioon projektin aikataulu, joka taas on sidoksissa budjettiin. Kaupallisissa projekteissa budjetti rajaa usein hyvinkin tarkoin, kuinka paljon aikaa voidaan kuhunkin työvaiheeseen käyttää. Aikataulun suunnittelu kannattaa tehdä tarkasti ja sen toteutumista seurata alati, jotta millekään työvaiheelle ei jää liian vähän aikaa. Tässä yhteydessä myös perinpohjaisen suunnittelun vaikutus korostuu, sillä pienikin lisäys sovelluksen rakenteeseen toteutuksen jo alettua saattaa aiheuttaa ajallisesti paljon töitä. [9.]

Koska Saunantakana-projekti ei ollut luonteeltaan kaupallinen, ei ennalta määritelty budjetti sitonut ajankäyttöä eikä näin ollen vaikuttanut vaatimusten määrään. Asiakas halusi kuitenkin suuntaa antavia arvioita projektin eri vaiheiden valmistumisajankohdista jo suunnitteluvaiheessa ja tarkentavia arvioita kehitystyön myöhemmissä vaiheissa. Oman ajankäytön arviointi osoittautui haastavaksi etenkin paljon taustatutkimustyötä vaativissa vaiheissa. Projektin edetessä useimpien

muuttujien huomioonottaminen oli kuitenkin helpompaa ja paikkansapitävän arvion tekemiseen harjaantui.

Sovelluksen haluttiin olevan mahdollisimman riippumaton käyttäjän selaimen asetuksista, toisin sanoen selaimelle asetettavien liitännäisten tai ominaisuuksien (esimerkiksi JavaScript) pois kytkeminen ei saanut vaikuttaa sivuston toiminnallisuuteen. Tästä syystä PHP palvelinpohjaisena kielenä oli paras valinta. Muutamien toimintojen, kuten sivuston valikoiden esteettiseen toimivuuteen vaikuttava kuvien ennakkoon lataaminen (preloading), toteuttaminen PHP:n avulla ei kuitenkaan ole mahdollista. Vaihtoehdoiksi jäävät yleisesti kömpelönä pidetty CSS-toteutus ja selaimen ominaisuuksista riippuvainen JavaScript. Päätyminen JavaScriptiin oli perustellumpaa sen selkeämmän rakenteen vuoksi. Vaikka JavaScript olisi kytketty pois päältä käyttäjän selaimesta, mitään oleellista informaatiota ei menetetä eikä käytettävyys huonone, sillä vaikutus rajautuu pääosin vain esteettisiin ominaisuuksiin.

3.2 Käytettävyys

Sovelluksen käytettävyys on laatuominaisuus, jolla määritellään kuinka helposti ja tehokkaasti tuotetta voi käyttää. Käytettävyyden mittaamiselle ei ole omaa yksikäsitteistä järjestelmää, ja yleensä tukeudutaankin kahteen tunnetuimpaan malliin: ISO 9241-11 -standardiin ja Jakob Nielsenin määritelmiin. Ensin mainitussa käytettävyys jaetaan tuottavuuden, tehokkuuden ja miellyttävyyden alakomponentteihin, joita arvioidaan aina suhteessa käyttäjiin sekä työhön ja käyttöympäristöön. Tuottavuus kuvaa tehtävien täydellistä ja virheetöntä onnistumista, tehokkuus resurssien (aika, raha, henkilömäärä) käyttöä ja miellyttävyys käyttökokemusta. [10.] ISO 9241-11 -standardi on suunnattu isommille järjestelmille, organisaatioille ja työelämän vaatimuksia silmällä pitäen [11].

Pienemmän profiilin projekteihin Jakob Nielsenin käytettävyysmalli sopii paremmin. Nielsen jakaa käytettävyyden viiteen laatukomponenttiin:

- Opittavuus: Järjestelmän perustoiminnot pitäisi olla helposti opittavissa jo ensimmäisellä käyttökerralla.
- Tehokkuus: Käyttäjien tulisi suoriutua mahdollisimman nopeasti ja yksinkertaisesti tehtävien tekemisestä.

- Muistettavuus: Käyttäjien tulisi kyetä palauttaa mieliin aikaisemmin saavutettu toimintataso mahdollisimman nopeasti tietyn mittaisen tuotteen käyttämättömyysjakson jälkeen.
- Virheettömyys: Järjestelmä ei saisi antaa käyttäjän tehdä virhettä, ja jos virhe kuitenkin tapahtuu, olisi sen oltava vakavuudeltaan vähäinen. Kriittisiä virhetilanteita ei missään tapauksessa saa syntyä.
- Miellyttävyys: Järjestelmän käytön pitäisi olla mukavaa ja mieluisaa, negatiivisia tuntemuksia ei saisi nousta pintaa. [12, s. 23.]

Riippuu paljon järjestelmän kontekstista – käyttäjäkunnasta ja käyttöympäristöstä – millaiset painotukset kullakin laatukomponentilla on. Vain ottamalla huomioon ja punnitsemalla ne moninaiset tavat, jolla käyttäjän kokemukseen voidaan vaikuttaa, on mahdollista rakentaa käyttäjäystävällisempiä sovelluksia. [13, s. 14.]

Internet-sovellukset ovat vakiintuneet ihmisten välisen kommunikaation peruselementeiksi, ja koska sivuja on paljon tarjolla, massasta erottuminen on elinehto uusille sivustoille. Uusi käyttäjä motivoidaan palaamaan sivulle pääosin kiinnostavan sisällön avulla, mutta myös esteettisillä seikoilla ja käytettävyydellä on huomattava vaikutus positiivisen kokemuksen luonnissa.

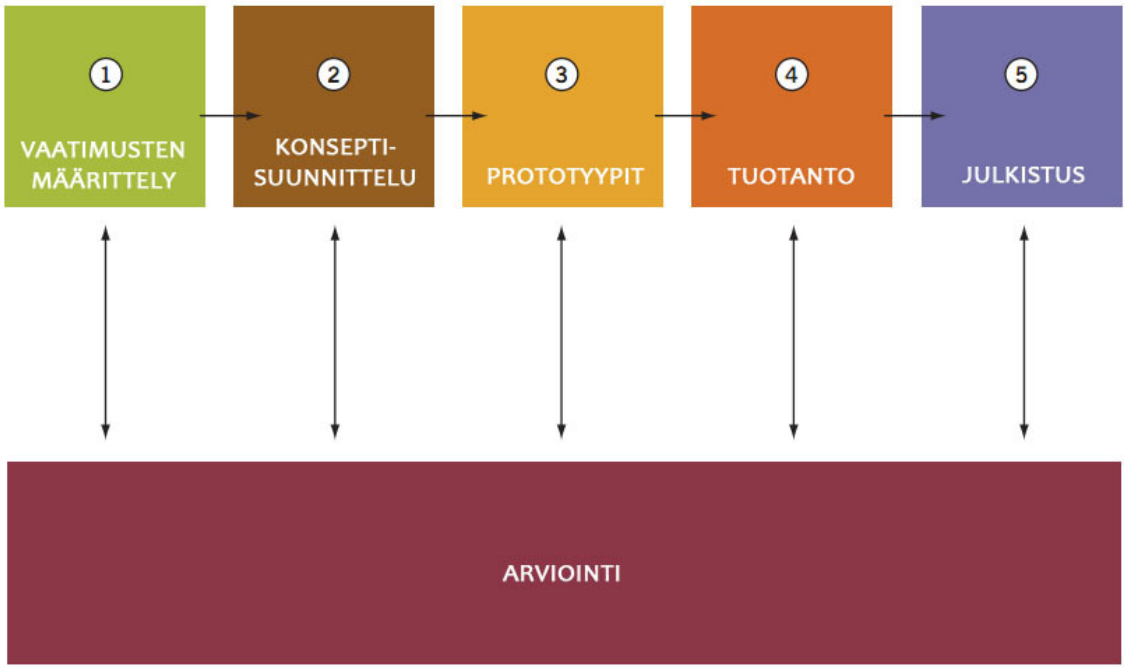
Vaikka internetsivujen käyttäminen pohjautuu verrattaen helppoon linkeistä, painikkeista, valikoista, tekstistä, tekstikentistä ja grafiikasta koostuvaan käyttöliittymään, monet käytettävyysongelmat ovat yleisiä. *Hahmottamiseen* liittyvät ongelmat tulevat esiin yleensä silloin, kun web-sivu on suunniteltu sen mukaan kuinka data on fyysisesti tallennettu (esimerkiksi tietokantaan) sen sijaan, että huomioitaisiin käyttäjän tarpeet. Tietokannan taulun tieto voidaan esimerkiksi helposti tulostaa sellaisenaan näytölle tekemättä lainkaan suodattavia toimenpiteitä. Tällainen toteutus saattaa tehdä sivujen suorittamisen ja ylläpidon helpoksi, mutta käyttäjän toimet voivat hidastua ja virheiden määrä kasvaa. Myös esteettisten seikkojen ylikorostaminen voi tehdä sovelluksen hahmottamisesta hankalaa. Etenkin tekstin ja taustan heikko kontrastiero ja graafisten elementtien paljous ovat yleisiä ongelmia.

Internet-sivuilla yleisimmät ongelmat liittyvät *navigaatioon* sivustorakenteen sisällä. Käyttäjän kannalta kolme olennaisinta kysymystä ovat ”Missä minä olen nyt?”, ”Miten

pääsen haluamaani määränpäähän?” ja ”Mihin tämä johtaa?”. Käyttäjän on kyettävä annetun informaation perusteella päättelemään, mitä tapahtuu, jos tiettyä linkkiä painetaan, ja viekö se lähemmäs määränpäätä. Navigaatiosuunnittelussa on otettava huomioon sivuston tiedon looginen rakenne, käyttötarkoitukset ja tuon tarkoituksen mukainen kieli sekä nykyisen olinpaikan selkeä esilletuonti. Linkkien tulisi olla selkeitä ja yksikäsitteisiä, ja niiden nimestä pitäisi selvitä kohde tai tieto, johon linkki viittaa.

Tietokantoihin tai tietovarastoihin perustuvilla sivuilla käyttäjälle on tarjottava ainoastaan sellaista tietoa, joka kulloisellekin hetkellä löytyy tietokannasta. *Tietokantaintegraation* yleisin virhetilanne seuraa, kun käyttäjälle näytetään vanhentunutta dataa. Esimerkiksi tietosisällön muokkaukseen tarkoitetun sovelluksen on aina näytettävä uusin data ennen muokkauksen aloittamista, jolloin käyttäjä tietää tarkalleen mitä kohtia tulee muuttaa. [13, s. 4.]

Loppukäyttäjät on otettava huomioon heti projektin alussa ja koko ajan tämän jälkeen. Käytettävyyden tulee olla osa jokaista suunnittelun ja toteutuksen vaihetta (kuva 1). Tämä ei poissulje taiteellisten tai teknisten ratkaisujen huomioimista, vaan korostaa sitä, että jokainen tehty ratkaisu on syytä käydä läpi myös hyvän käytettävyyden näkökulmasta. [13, s. 14.]



Kuva 1. Käytettävyysarviointi ja projektin elinkaari [13, s. 16].

Kun saunantakana.fi-sivuston vaatimus- ja toiminnallisuusanalyysi oli valmis, tehtiin halutut ominaisuudet ja käytettävyysseikat huomioon ottaen vedoksia ulkoasusta. Suunnittelun alkuvaiheessa käyttöliittymän hahmotelmia piirrettiin paperille, jolloin erilaisten tarvittavien ominaisuuksien yhdistely oli helppoa. Rakenteen selkiytyessä versioita siirryttiin toteuttamaan kuvankäsittelyohjelmalla.

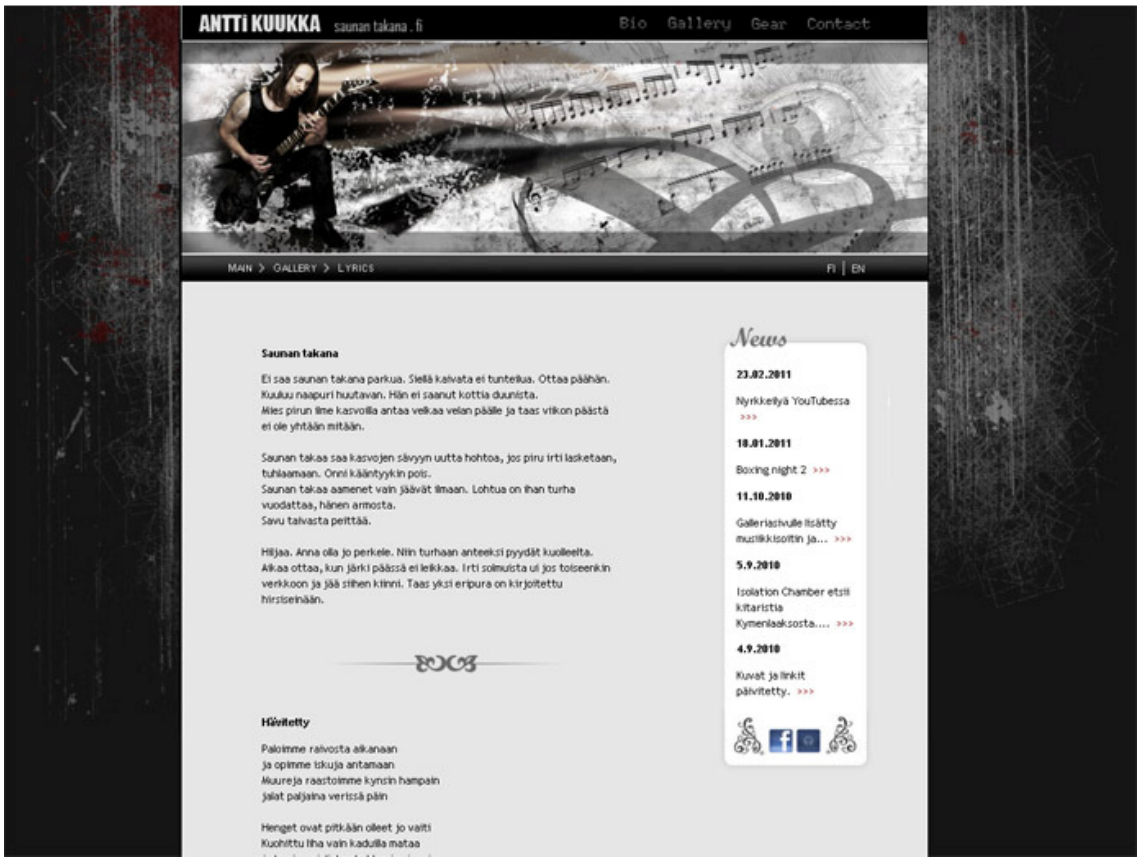
Fysiologisista syistä internetsivujen värimaailman perustaksi suositellaan sinistä, harmaata tai siniharmaata, ja koska etenkin taustan väreiksi suositellaan valkoista, vaalean harmaata tai vaaleita pastellisävyjä, valittiin leipätekstin pohjaksi vaalean harmaa. Värivalinta liittyy suoraan metallimusiikin mustavalkoiseen estetiikkaan ja sopii hyvin sovelluksen kontekstiin. Ihminen hahmottaa huonosti yli seitsemästä väristä koostuvien sivujen informaatiota, joten keskeisten värien määrä on rajattu kolmeen pääväriin (valkoinen, harmaa ja musta), joiden lisäksi esimerkiksi linkeissä esiintyy oranssin sävyjä. [14.]

Värejä ei käytetä missään yhteydessä yksinään tiedon välittämiseen, vaan vain informaation erottelemiseen, kategorioimiseen ja huomion herättämiseen. Hyvän luettavuuden ylläpitämiseksi tekstin ja taustan välillä pyritään aina pitämään mahdollisimman voimakas kontrasti. Sisällön taustaksi valittu harmaa on sävyltään niin vaalea, että musta teksti erottuu siitä hyvin.

Sivustolle saapuvalle vierailijalle on käytävä heti selväksi, mille sivulle hän on tullut ja mihin se on tarkoitettu. Vasempaan yläkulmaan sijoitettuna logo on länsimaiseen lukusuuntaan tottuneelle käyttäjälle kaikkein loogisimmassa paikassa ja kertoo selvästi sivuston nimen (kuva 2). Päänavigointivalikko sijaitsee yläoikealla, josta se on helppo löytää. Tämä ratkaisu mahdollistaa myös suuremman kuva-alan käytön sivuston kannalta tärkeimpään asiaan eli itse tietosisällölle, kun sivuston sisäiseen navigointiin tarkoitettuja linkkejä ei ole ripoteltu ympäri sivua.

Otsikko- eli header-kuva ja mustaa taustaa vasten erottuvat harmaat ornamentinkaltaiset graafiset osat luovat toimintojen suhteen tyhjää tilaa, sillä erottelemalla elementtejä toisistaan voidaan selkiyttää sisällön jaottelua ja ohjata katsetta. Ei ole käyttäjän kannalta järkevää tehdä liian tiukkaan pakattuja sivuja. [15, s. 18.] Otsikkokuva myös jakaa tärkeämmän tason informaation – sivuston nimen ja

päävalikon – erilleen alemman tason navigoinnista – hierarkialinkityksestä ja kielenvaihdosta.

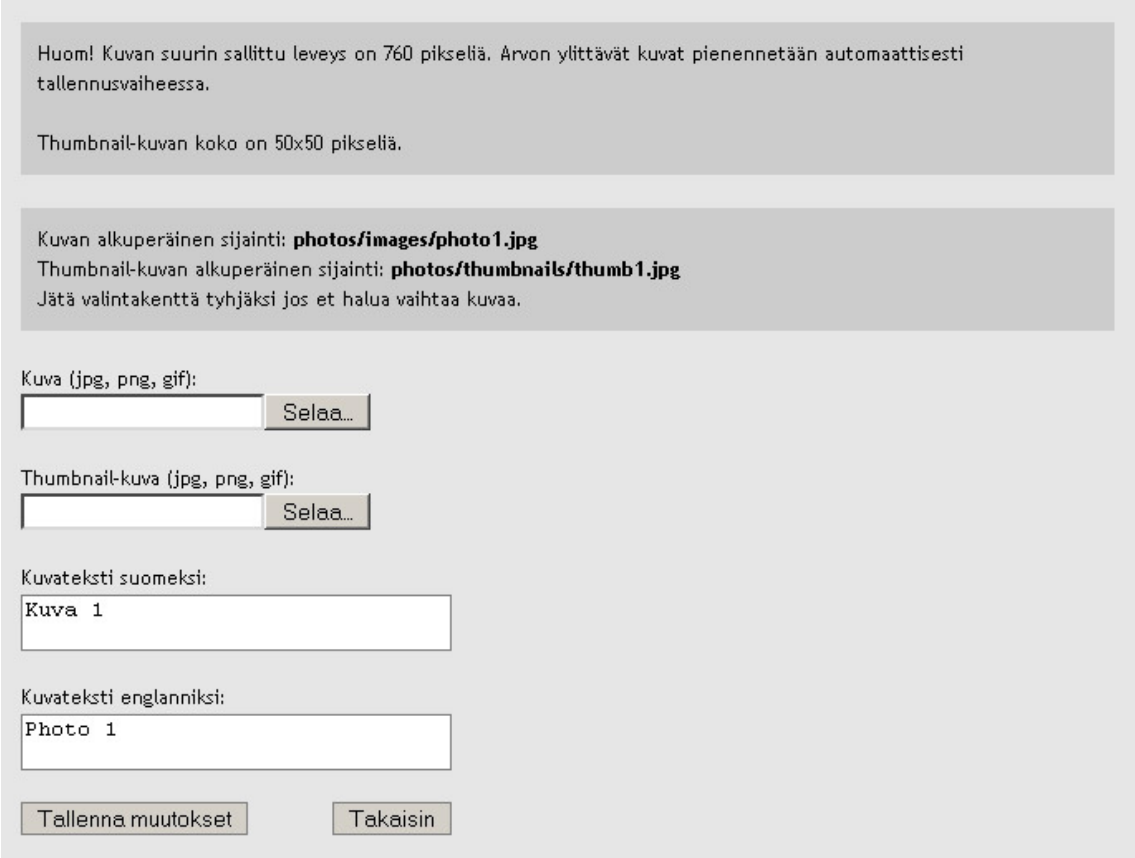


Kuva 2. Esimerkkisivu sovelluksesta.

Käyttäjän on oltava tietoinen siitä, missä kohtaa sivuston hierarkiassa hän kullakin hetkellä on, eikä hänen missään tapauksessa saa antaa eksyä. Sijainnin indikoimiseen käytetään niin sanottua linkkipolku-navigaatiomallia, jossa käyttäjän sijainti osoitetaan suhteessa sivuston sisällön hierarkkiseen jaotteluun. Jokaista ylempää tasoa voi napsauttaa, jolloin selain ohjataan tuohon kohtaan hierarkiassa. Linkkipolku-mallin hyöty on yksinkertaisuus ja pieni tilan tarve, jolloin navigaatio ei vie tarpeettomasti tilaa pääosassa olevalta informaatiolta. Myös tasojen välillä oleva kulmasulkusymboli (>) auttaa ymmärtämään, että aktiivisena oleva sivu kuuluu polussa edellä olevan tason alaisuuteen. [15, s 206]

Teksti on tasattu pääsääntöisesti vasemmalle, joka mahdollistaa paremman lukunopeuden esimerkiksi keskitettyyn tai oikealle rajattuun tekstiin nähden. Fonttina käytetään pääteviivatonta (sans serif) Trebuchet MS:ää, joka toisin kuin pääteviivalliset

(serif) fontit, takaa hyvän luettavuuden myös pienemmillä fonttikoilla. Yleisesti serif-luokan fontit ovat käytössä painotuotteissa, mutta etenkin näyttöpäätteeltä niiden lukeminen on sans serif -luokan fontteja hitaampaa.



Kuva 3. Ohjeistus kuvaa tallennettaessa.

Fontit ovat käyttöjärjestelmäsidonnaisia, minkä vuoksi on syytä määritellä joukko vaihtoehtoisia fontteja. CSS:llä määritys voidaan tehdä seuraavasti:

```
font-family: 11px "trebuchet ms", arial, helvetica, sans-serif;
```

jolloin selaimen ei tarvitse turvautua käyttämäänsä oletusfonttiin, joka pahimmassa tapauksessa rikkoo sivuston ulkoasun. Määritykseen valitut fontit kuuluvat kaikki sans serif -luokkaan, minkä vuoksi niiden erot ovat hyvin vähäisiä. Jotta vaihtoehtoisiin fontteihin ei kuitenkaan tarvitsisi tukeutua, suositellaan mahdollisimman tunnetun pääfontin valintaa. Trebuchet kuuluukin Microsoftin ”Core Fonts for the Web” -projektiin, minkä johdosta se on niin yleinen, että sivut kyetään näyttämään alkuperäisillä fonttiasetuksilla lähes kaikkien kävijöiden järjestelmillä. [16.]

Sovelluksen osioihin, joissa käyttäjä ei *a priori* voi tietää oikeaa toimintatapaa, lisättiin avustava ohjeistus. Kuvassa 3 näkyvät ohjelman tarjoamat ohjeet käyttäjän ollessa kuvatietojenmuokkausvalikossa. Käyttäjän toimien kannalta kaikki tärkeimmät tiedot, kuten sallitut kuvakoot, alkuperäisten kuvien sijainnit ja valintakenttien toiminnallisuus, pyritään tuomaan esille.

4 Tekninen toteutus

4.1 Työkalut

NetBeans

Työssä käytettiin ohjelmointiympäristönä NetBeans IDE:n (integrated development environment) PHP-paketoitua versiota 6.9.1. Se on avoimen lähdekoodin ohjelma, jonka ominaisuudet erityisesti editointityökalujen monipuolisuudessa ja koodin täydentämisessä ovat kilpailijoidensa parhaimmistoa [17]. Kirjoitetun koodin testaukseen ja virheiden poistoon on kätevät debug-toiminnot, ja sisäänrakennetulla FTP-ohjelmalla voi tiedostot siirtää suoraan palvelimelle. Ohjelmointiprosessin sujuvuuteen vaikuttavat erityisesti syntaksin ja muuttujaesiintymien värikoodaus sekä koodin semanttinen analyysi ja poikkeavuuksien korostus. NetBeans soveltuu verkkosovellusten kehitysympäristöksi yleisemminkin, sillä PHP:n lisäksi ohjelma tukee muun muassa CSS-, (X)HTML- ja JavaScript-koodia. [18.]

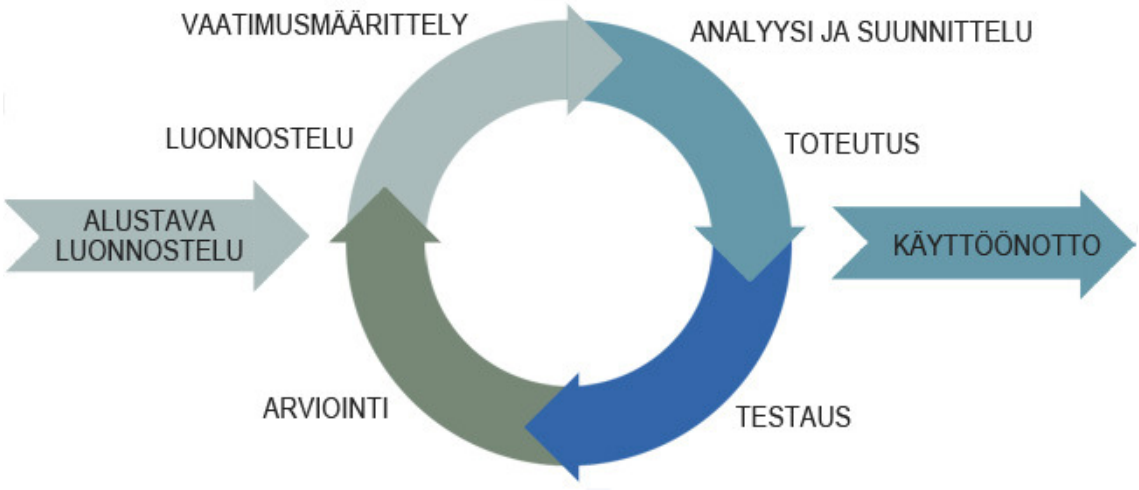
XAMPP

Ilmainen, avoimeen lähdekoodiin perustuva XAMPP (cross-platform (X), Apache HTTP Server, MySQL, PHP ja Perl) on kehitystyökalupaketti, jolla voidaan testata palvelinpohjaisia sovelluksia paikallisella työasemalla ilman yhteyttä oikeaan web-palvelimeen. XAMPP on saatavilla Windowsiin, Linuxiin, Solarikseen ja Mac OS X:ään. Sen pohjana toimii Apachen web-palvelin, johon on liitetty MySQL-tietokantaohjelma ja tulkki PHP- ja Perl-kielille. Mukana on myös phpMyAdmin, PHP-kielellä kirjoitettu ohjelma MySQL-tietokannan ylläpitoon. [19.]

4.2 Ohjelmistokehitysprosessi

Sovelluskehitysprojektia käynnistettäessä on syytä kiinnittää huomiota itse tuotantoprosessiin. Jokainen projekti tulisi analysoida etukäteen ja valita sen perusteella sopivin kehitystyön vaihejakomalli. Tuotantotalouden valmistusprosesseihin perustuvat, mutta laajalti ohjelmistokehityksessä yhä käytettävät vesiputousmalli ja V-malli vaikuttavat liian joustamattomilta soveltuakseen projektin luonteeseen: asiakkaan mahdolliset lisätoiveet suunnitteluvaiheen päätyttyä olisivat hyvin hankalasti

toteutettavissa. Iteratiivisen ja inkrementaalisen ohjelmistokehityksen malli sen sijaan mahdollistaa asiakkaan ehdoilla tapahtuvan tuotantoprosessin. Monet etenkin ketterää ohjelmistonkehitystä (Agile) hyödyntävät metodit, kuten Scrum ja Extreme Programming, pohjautuvat sekä iteratiiviselle että inkrementaaliselle mallille. Kummatkin mallit koostuvat kuvan 4 mukaisista sykleistä. Inkrementaalinen kehitys on työvaihe- ja ajankäyttöstrategia, jossa työ jaetaan pienempiin, toiminnallisuuksiltaan yhteneväisiin osiin, jotka aikataulutetaan, toteutetaan ja valmistuttuaan integroidaan kokonaisuuteen. Jokainen inkrementointi lisää lopullisen tuotteen toimintojen määrää. Inkrementaalinen kehitys auttaa etenkin ohjelman kehittäjiä parantamaan tuotantoprosessia, sillä jokaisen integroimisvaiheen jälkeen voidaan ajankäyttöä ja suunnittelua tarkastella kriittisesti ja muuttaa tarvittaessa menettelytapoja seuraavan inkrementoinnin aikana.



Kuva 4. Inkrementaalisen ja iteratiivisen ohjelmistokehitysprosessin vaiheet.

Iteratiivisuus-mallin mukaisessa kehitystyössä jokainen sykli – iteraatio – muokkaa jo olemassa olevia ominaisuuksia, kunnes riittävän monen iteraatio- ja inkrementointikerran jälkeen tuote on valmis. Vaiheet, jotka vesiputous- ja V-mallissa käydään vain kerran läpi, toistetaan iteratiivisessa mallissa riittävän monta kertaa, jotta lopputulos on asiakkaan toiveiden mukainen. Jokainen iteraatio sisältää siis tulevan syklin päämäärien ja vaatimusten määrittelyn, suunnittelun ja toteutuksen, testauksen sekä arvioinnin. Arvioinnin tulokset otetaan huomioon seuraavaa iteraatiota aloitettaessa, ja näin kehitysprojekti joustaa ja asiakkaan toiveet pidetään etusijalla.

Siinä missä inkrementaalinen malli parantaa kehittäjän toimintatapoja, iteratiivinen menettely auttaa tuotteen laadun optimoimisessa. Asiakas voi tutustua sovellukseen jo kehitysprosessin aikana ja hänen antamansa palautteen perusteella kehityksen suuntaa voidaan tarvittaessa muuttaa. [20.]

Syklin arviointi perustuu käyttäjän palautteeseen sekä ohjelman rakenteen, modulaarisuuden, käytettävyyden, luotettavuuden, tehokkuuden ja saavutettujen tavoitteiden analysointiin. Yhden syklin aikana muutettavien ja lisättävien toiminnallisuuksien määrä kannattaa harkita tarkoin, sillä liian monen tehtävän tekeminen ilman välianalyysia saattaa lisätä turhaa työtä, ja vastaavasti asiakkaan palautteen pyytäminen jokaisen vähäisenkin työvaiheen jälkeen voi olla turhauttavaa ja resursseja haaskaavaa. [21.]

Suuremman mittakaavan projekteissa inkrementaation laajuus määritellään yleensä aikarajoitteisesti ominaisuusrajoitteisuuden sijaan. Ratkaisu puolustaa paikkaansa etenkin kehitysryhmien yhteistyön kannalta, sillä tuloksia voidaan esitellä yhtä aikaa. Koska Saunantakana.fi-sovelluksen toteutus ei tapahtunnut suuren organisaation voimin, päätettiin inkrementaatiot määritellä ominaisuuksien mukaan. Työtä toteutettaessa inkrementaatioiden sisältämät työmäärät vaihtelivat, mutta perusajatuksena oli, että vähintään yksi toimintokokonaisuus (esimerkiksi uutisten tai valokuvien lisäys, poisto ja muokkaus) sisällytetään inkrementaatioon. Vaiheen valmistuttua asiakas sai kokeilla toimintoja, ja tarvittaessa iteraatioiden avulla lopputulos muokattiin asiakkaan toiveiden mukaiseksi, jonka jälkeen siirryttiin seuraavaan inkrementaatioon.

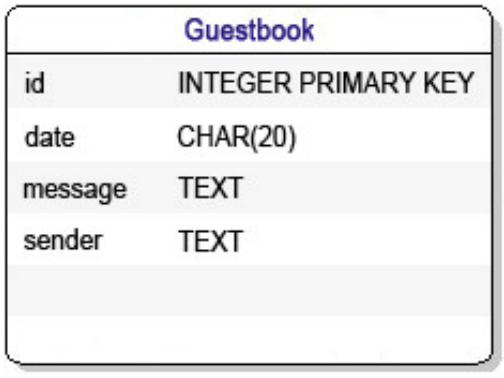
4.3 Tietokantarakenne

Tietokannaksi valittiin SQLiten versio 2, jota PHP 5 ja sitä uudemmat versiot tukevat natiivisti. Koska tietokanta sijaitsee samalla palvelimella itse sovelluksen kanssa, tarvetta erilliselle tietokantapalvelimelle ei ole. Tietokanta koostuu yhdestä binääritiedostosta, ja jos sovelluksen käyttämä palvelin vaihtuu, voi tietokannan siirtää suoraan uuteen kohteeseen ilman ylimääräistä konfigurointia. Äärimmäisestä kompaktiudesta johtuen myös koko tietokannan varmuuskopiointi on helppoa. SQLite soveltuu hyvin määrältään vähäisen ja keskiverron liikenteen internetsivuille, mutta käytännössä suorituskyky riippuu pääosin siitä, kuinka voimakkaasti sovellus

kuormittaa tietokantaa. Yleisesti ottaen alle 100 000 vierailua päivässä saavat sivut toimivat moitteettomasti SQLiten kanssa. [5.]

Sovelluksen tietokanta rakentuu tauluista news, bio, gear, links, guestbook, photos, music, lyrics ja banner, joilla sovelluksen rakenteen ansiosta ei ole relaatioita keskenään. SQLite on käytännössä tyypitön, eli taulujen sarakkeisiin tallennettavan datan muodolla ei ole väliä. Tästä johtuen sarakekohtaista datatyyppiä ei tarvitse edes ilmoittaa, mutta käytännössä näin ei tehdä, sillä tyyppitieto parantaa koodin luettavuutta ja kertoo, minkälaista tietoa sarakkeeseen pyritään laittamaan.

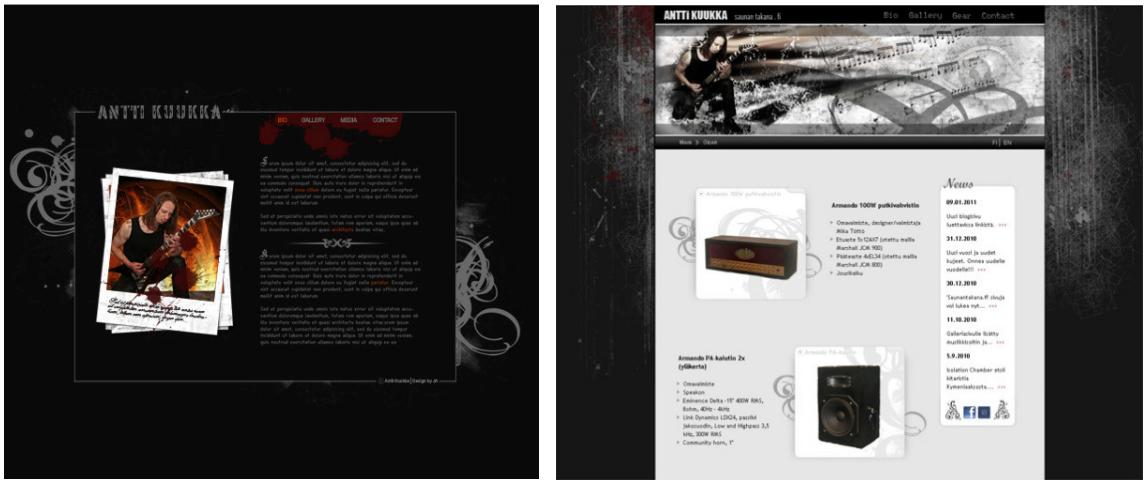
Tauluja luotaessa ei erikseen tarvitse määritellä sarakkeiden leveyksiä, sillä SQLite varaa muistia vain sen verran kuin taulussa oleva tieto edellyttää. Sarakkeen tyyppimäärittelyn yhteydessä mainittava sarakkeenleveys on – aivan kuten itse tyyppimäärityskin – vain selkeyttämässä koodin tulkintaa. Poikkeuksena tyypittömyyteen on sarake, jonka tyypiksi on määritelty INTEGER PRIMARY KEY (pääavain). Tällaiseen sarakkeeseen voi lisätä vain 32-bittisen kokonaisluvun. Kuvassa 5 on sovelluksen vieraskirjan käyttämä taulu, jossa kunkin rivin tunnistenumerona (id) käytetään kokonaislukupääavainta. INTEGER PRIMARY KEY -tyyppiä voi käyttää automaattiseen inkrementointiin, jolloin id:n arvoksi asetetaan uniikki kokonaisluku, joka yleensä on yhden numeron suurempi kuin siihen mennessä suurin käytetty id-numero.



Kuva 5. Tietokannan taulu.

4.4 Graafinen ilme

Asiakas halusi sovelluksen ulkoasun ottavan vaikutteita raskaan rock- ja metallimusiikin tummanpuhuvasta estetiikasta. Genrejen viitekehys heijastui myös verkko-osoitteen saunantakana.fi nimivalinnassa. Näiden kahden spesifin vaatimuksen pohjalta laadittiin Photoshop CS3 -kuvankäsittelyohjelmalla sarja vedoksia sivuston mahdollisesta ulkonäöstä. Vedoksissa oli näyttävyyden ja käytettävyyden suhteen tehty erilaisia ratkaisuja, ja viimeisen valintakierroksen kahdesta vaihtoehdosta (kuva 6) asiakas päätyi malliin, jossa käytettävyyttä painotettiin hieman enemmän.



Kuva 6. Viimeisen valintakierroksen vaihtoehdot.

Visuaalisuutta ei kuitenkaan voinut täysin uhrata käytettävyyden vuoksi, sillä sovellus toimisi asiakkaan käyntikorttina ja sivuston muotokielen haluttiin heijastelevan asiakkaan persoonaa. Sovelluksen yhtenäinen graafinen identiteetti voidaan ottaa käyttöön myös muissa yhteyksissä, kuten sivuston mainostamiseen tarkoitetuissa bannereissa, käyntikorteissa, julisteissa ja levynkansissa.

Otsikkokuvan vaikutus sivuston yleisilmeeseen on suuri, ja vaikutusta haluttiin korostaa mahdollistamalla useampien kuvien käyttö, jotka kukin tuovat eri tavoin esiin sivuston taustalla olevia teemoja. Sovelluksen julkistusvaiheessa kahta erilaista otsikkokuvaa käytetään pääosa-alueiden (uutiset, biografia, galleria, kalusto, yhteystiedot) yhteydessä. Uusia kuvia voi myöhemmin lisätä ja tältä osin uudistaa sivuston ilmettä. Asiakkaan kanssa järjestettiin kolmena eri ajankohtana valokuvausistuntoja, jolloin

hankittiin valtaosa sivuston visuaalisesta materiaalista. Kuvauksia tehtiin sekä ulkoilmassa että tarkoitusta varten rakennetussa kotistudiossa.

4.5 Käytettävyystestaus

Kuten edellä on jo mainittu, käytettävyys on otettava huomioon jo suunnitteluvaiheessa. Jokaista yksityiskohtaa ei ole kuitenkaan mahdollista suunnitella etukäteen täysin aukottomasti, minkä vuoksi käytettävyyttä on voitava testata tavalla tai toisella projektin edetessä. Saunantakana.fi-sovelluksen kehitysvaiheiden aikana käytettiin käytettävyystarkastus- (usability inspection) ja käyttäjätestausmetodeja (user testing). Käytettävyystarkastuksessa sovelluksen kehittäjä arvioi jokaisen inkrementoinnin päätteeksi syklin aikana valmistuneet toiminnot. Objektiivisen tuloksen saavuttamiseksi käytetään Tom Brinckin, Darren Gerglen ja Scott Woodin luomaa tarkistuslistaa (liite 1). Sovellus, tai opinnäytetyön tapauksessa usein sovelluksen osa, tarkistetaan listan avulla. Lista käydään läpi kohta kohdalta kirjaten jokainen ongelma ja virhetilanne muistiin. Löydettyjen ongelmakohtien laajuus määrää, siirtyykö niiden korjaus seuraavaan iteraatioon vai ovatko parannustoimenpiteet välittömästi tehtävissä. Korjausten jälkeen testataan vielä uudet toiminnot loppukäyttäjällä. Yleensä testaus tapahtui tarkkailemalla käyttäjän toimia, jolloin käytettävyyden kannalta ongelmalliset ratkaisut kyettiin välittömästi identifioimaan. Tämän kaltainen testaus paljastaa hyvin kattavasti ongelmakohdat, sillä lähestymistapa on käyttäjäläheinen. [13, s. 406.]

Käyttäjätestauksessa tuotteen loppukäyttäjälle tai kohderyhmään kuuluvalle henkilölle annetaan tehtäviä, jotka hänen tulee suorittaa sovelluksen avulla. Testihenkilön toimia tarkkaillaan, ja tilanne voidaan dokumentoida esimerkiksi videoimalla tai kirjoittamalla muistiinpanoja. Mikään ei ole sovelluksen ja käyttöliittymän parantamisen kannalta valaisevampaa kuin nähdä tuotteen todellisen käyttäjän epäonnistuvan niissä toiminnoissa, jotka ovat hänen käytettäväkseen suunniteltu. Tämän, ja myös metodin suhteellisen halpuuden vuoksi, käyttäjätestaus on suosituimpia käytettävyystestauksen muotoja.

Saunantakana.fi-sovelluksessa valtaosa toiminnallisuuksista sijaitsee ylläpitosivuilla, johon myös käytettävyystestauksessa luonnollisesti keskityttiin. Koska lopullisen sovelluksen hallintasivujen käyttäjiä on vain yksi, projektin tilaaja itse, haluttiin

yleispätevämmän testituloksen saamiseksi käyttää kahta ulkopuolista testihenkilöä. Jokaisen hallintasivuihin keskittyneen inkrementoinnin ja iteraation tuloksena syntynyt laajempi toimintokokonaisuus (esimerkiksi uutisten tai valokuvien lisäys, poisto ja muokkaus) alistettiin valmistumisensa jälkeen käyttäjätestaukseen. Testeissä koehenkilölle annettiin sivuston toimintaan liittyviä tehtäviä, kuten järjestelmään kirjautuminen ja musiikkikappaleen poisto tai kuvan lisääminen, ja suorituksista kirjoitettiin muistiinpanoja. Lopuksi koehenkilöiltä pyydettiin yhteenveto kokemuksesta ja suorituksen aikana kohtaamistaan hankaluuksista. Näin löytyneet virheet koskivat pääosin toimintonäppäimien asettelua ja ryhmittelyä sekä sivuilla näytettävien ohjeistusten puutteellisuuksia.

Kahden testausmetodin käyttö rinnakkain eliminoi tehokkaasti virhetilanteiden, sovelluksen käytön hitauden, virheellisten alitajuisten mallien ja muistamiseen liittyvien ongelmien aiheuttamat käytettävyyshäiriöt. Projektin ajankäyttöä suunniteltaessa on kuitenkin huomioitava etenkin käyttäjätestauksen vaatima ajantarve sekä ongelmallisiksi havaittujen toimintojen parantaminen. [13, s. 423.]

4.6 Tietoturva

PHP:n käytön yleisyys johtuu suurelta osin sen helppokäyttöisyydestä ja matalasta oppimiskynnyksestä. Monet muissa ohjelmointikielissä hankalasti toteutettavissa olevat toiminnot voidaan PHP:ssä tehdä hyvin yksinkertaisesti. Kokematonkin ohjelmoija voi suhteellisen helposti rakentaa dynaamisen, käyttäjältä syötteitä ottavan internetsovelluksen. Aloituskynnyksen mataluus ja yksinkertaiset toiminnot voivat kuitenkin olla myös ongelmallisia etenkin tietoturvan kannalta, sillä turvallisuus ei ole ohjelmointikielen vaan ohjelmoijan ominaisuus. Mikään kieli ei itsessään estä turvattoman koodin kirjoittamista, sillä kielen ominaisuuksien käytöstä vastaa viime kädessä ohjelmoija. [22, s. 15.]

Käytettävyyden ohella myös tietoturva on otettava huomioon projektin jokaisessa vaiheessa. Etenkin suunnittelun aikana on hyvä tiedostaa valittujen tekniikoiden ja toimintatapojen vaikutus sovelluksen turvallisuuteen. Kehitys- ja tuotantoympäristöt on pidettävä erillään toisistaan, ja tuotantopalvelimella tulee olla ainoastaan viimeisin versio sovelluksesta eikä minkäänlaisia testisovelluksia. Kaikki testit on tehtävä kehitysympäristössä. Tuotantopalvelimen tulee olla mahdollisimman suljettu, ja muiden

kuin ylläpitäjän käyttöoikeuden rajoitettu vain lukemiseen, jos sovelluksen käyttö ei muuta edellytä. Tuotantopalvelimen data on tasaisin väliajoin varmuuskopioitava palvelimen ulkopuoliseen sijaintiin, jotta mahdollisen palvelinrikon sattuessa tuore tieto on palautettavissa. Saunantakana.fi-sivuston osalta palveluntarjoaja tekee päivittäisen varmuuskopioinnin, jonka lisäksi asiakas ottaa viikon välein itselleen tietokannasta kopion. [22, s. 25.]

Sovellusta luotaessa ei riitä, että tietoturvaa parantavat komponentit huomioidaan ohjelmoinnin aikana, vaan turvallisuutta on myös aktiivisesti testattava. Tässä työssä jokainen käyttäjäsyötekenttä testattiin eri tavoin muun muassa SQL-injektiota, HTML:n tai JavaScriptin avulla tehtäviä Cross-site Scripting -hyökkäyksiä ja muuta virheellistä dataa vastaan. Lisäksi muutamassa tapauksessa luotiin koefunktioita, joilla voitiin todentaa yksittäisten osa-alueiden turvallisuus. [22, s. 260.]

4.6.1 Palvelinkonfiguraatio

Palvelimen PHP-asetusten tarkistaminen luo pohjan turvalliselle kehitystyölle. `phpinfo()`-funktiokutsulla saadaan selville palvelimen php.ini-konfiguraatiotiedoston muuttujat ja niiden arvot. Jos asetuksia ei voi muuttaa, kuten toisinaan kaupallisten palveluntarjoajien tapauksessa, täytyy tämä ottaa huomioon ohjelmointiratkaisuissa.

Suurin yksittäinen tietoturvallisuutta heikentävä tekijä on PHP-asetuksissa päälle asetettu `register_globals`-direktiivi. Sen alkuperäisenä tarkoituksena oli helpottaa ohjelmointia asettamalla kaikki käyttäjän antamat syötteet globaaleiksi, mutta sittemmin sen ongelmallisuus on havaittu ja PHP:n versiosta 4 eteenpäin se on oletuksena poissa päältä. [22, s. 4.]

PHP:n virheraportointitoiminto antaa ohjelmoijalle mahdollisuuden havaita sellaisia virheitä, jotka muuten saattaisivat jäädä huomaamatta, ja etenkin kehitysympäristössä toiminnon käyttö on suositeltavaa. Ulkopuoliselle virheraportit tarjoavat kuitenkin tavan tutustua sovelluksen heikkouksiin ja tekevät järjestelmääntunkeutumisyritykset helpommiksi. Raportointi kannattaa lopullisessa toimintoympäristössä kytkeä pois `display_errors`-asetuksella. Sovelluksen jatkokehityksen kannalta tarve syntyneiden virheiden kirjaamiseen on kuitenkin olemassa. Yleinen metodi, jota myös

tässä työssä käytettiin, on ohjata raportit `log_errors`-asetuksella suojattuun sijaintiin palvelimella, jonka jälkeen tiedoston käyttöoikeudet annetaan vain pääkäyttäjälle. [23.]

4.6.2 Syötteen tarkistus

Dynaaminen web-sovellus perustuu palvelimen ja asiakkaan väliseen tiedon vaihtoon. Käyttäjä kommunikoi palvelimen kanssa pääosin erilaisten lomakkeiden avulla. Tässä käyttäjän antamassa syötteessä piilee suurin vaara sovelluksen toiminnalle. Järjestelmän saaman syötteen suodatus on internetsovellusten tietoturvan kulmakiviä. Suodattamalla varmistutaan, että tieto on halutussa muodossa ja ettei se ole järjestelmälle haitallista. Syötteellä tarkoitetaan tässä yhteydessä kaikkea sitä dataa, jonka järjestelmä saa välittömän toimintaympäristönsä ulkopuolelta, kuten asiakkaalta, tietokannasta tai RSS-syötteenä. Validointi on suoritettava palvelimella, eli tässä tapauksessa PHP:n avulla, sillä validointi asiakkaan puolella selaimessa esimerkiksi JavaScriptillä ei täytä turvallisuustavoitteita ollen helposti ohitettavissa.

Kaikki muu kuin odotettavissa oleva data kannattaa suodattaa pois. Sovellus on sitä tietoturvallisempi, mitä tiukempia suodatussääntöjä käytetään. Jos tekstikentän sisällön odotetaan olevan vain numeroita, ei minkäänlaisten muiden merkkien käyttöä tule sallia. Saunantakana.fi-sovelluksessa tekstikenttien sisältöä ei voida rajata aivan niin tarkasti, mutta turvallisuussyistä erikoismerkkien käyttöä on rajoitettu. Lisäksi syötteiden pituuksia valvotaan liian suurten tietomäärien lähettämisen estämiseksi, ja `htmlentities`- ja `strip_tags`-funktioilla muutetaan merkit HTML-muotoon sekä poistetaan kaikki HTML-elementit. [24, s. 272.]

SQL-injektiot tai -hyökkäykset ovat yksi epäonnistuneen syötevalidoinnin yleisimmistä seurauksista. Esimerkkinä voidaan tarkastella tilannetta, jossa käyttäjältä kysytään käyttäjätunnusta ja salasanaa. Jos syöte viedään sellaisenaan tietokantaan,

```
SELECT * FROM users WHERE name='$käyttäjätunnus' AND pass='$salasana'
```

voi tunkeutuja kirjoittaa salasanakenttään

```
' OR '1'='1
```

Tuloksena saadaa lauseke

```
SELECT * FROM users WHERE name='käyttäjä' AND pass='' OR '1'='1'
```

jossa ehto 1=1 on aina tosi ja kirjautuminen järjestelmään onnistuu ilman salasanaa. Ongelman ratkaisu piilee heittomerkkien käsittelyssä. Jos heittomerkkien eteen sijoitetaan kenoviiva eli niin sanottu pakomerkki (escape character), ohjelma tulkitsee heittomerkin vain tavalliseksi sisällöksi eikä se aiheuta ongelmia. Pakomerkkien lisääminen voidaan automatisoida kytkemällä palvelimen `magic_quotes_gpc`-toiminto päälle. Tällöin kaikki syötteet tarkistetaan erikoismerkkien varalta ja pakomerkki lisätään tarvittaessa. Yleensä toimintoa ei kuitenkaan suositella käytettäväksi, sillä jo turvalliseksi todettujen syötteiden tarpeeton läpikäynti on palvelimen resurssien haaskausta. Suositeltavampi vaihtoehto on käyttää `addslashes()`-funktiota vain ongelmallisten syötteiden yhteydessä. [23.]

Palvelimelle ladattavien tiedostojen validoinnissa pätevät samat perussäännöt kuin tekstikenttien syötteen tarkistuksessa. Yleisin tapa aiheuttaa vahinkoa on lisätä kuvatiedoston binääridatan joukkoon esimerkiksi PHP-koodia, joka suojauksen puutteellisuuden vuoksi suoritetaan palvelimella tarjoamalla näin hyökkääjälle tietoa tai jopa pääsy järjestelmään. Tiedoston oikeellisuus on siis varmistettava mahdollisimman kattavasti. Palvelin tallentaa ladattavan tiedoston superglobaaliin `$_FILES`-taulukkomuuttujaan, josta saadaan tiedoston nimi, sisällöstä kertova MIME-tyyppi, koko, tilapäinen tallennuspolku ja mahdolliset lataamisen yhteydessä tulleet virheilmoitukset. Jos tiedoston oletetaan olevan kuva, tarkistetaan, että MIME-tyyppi vastaa sovelluksessa sallittuja kuvatyyppejä (esimerkiksi JPEG, GIF tain PNG). Hyökkääjä voi myös yrittää ladata PHP-tiedoston muuttamalla sen tyypin kuvaa vastaavaksi. Tämän vuoksi on tarkastettava tiedostopääte ja sallittava vain haluttuja kuvatyyppejä vastaavat päätteet, sillä palvelin ei yleensä tulkitse tällaisia tiedostoja. Etenkin kaupallisten palveluntarjoajien yhteydessä, jollaista Saunantakana.fi-sovelluksessakin käytetään, tästä ei kuitenkaan voida olla varmoja. Lisäturvallisuutta luodaan tallentamalla tiedosto palvelimen juurihakemiston ulkopuolelle, rajoittamalla hakemiston käyttöoikeuksia ja poistamalla komentosarjojen suoritusmahdollisuus .htaccess-tiedoston avulla. [24, s. 288.]

4.6.3 Salasanaturvallisuus

Saunantakana.fi-sovelluksen ylläpito-osaan kirjaudutaan antamalla käyttäjätunnus–salasana-pari. Kirjautumistunnuksia ei tulisi koskaan säilyttää palvelimella selkokielisinä, vaan ne olisi salattava (encrypt) tai tiivistettävä (hash). Tässä työssä salasana tiivistetään kryptografisella SHA-1-tiivistefunktiolla (Secure Hash Algorithm*).* [24, s. 56.] Tiivistämisessä datasta luodaan kiinteämittainen merkkijono, jota voidaan käyttää alkuperäisen tiedon tunnistamiseen. Tiivisteestä ei voida palauttaa lähtötilannetta, mutta identtiset lähtötilanteet tuottavat aina saman tiivisteen. Jos tietokannassa säilytettävät salasanatiivisteet joutuvat hyökkääjän haltuun, ovat ne hyvin hankalasti hyödynnettävissä juuri palauttamattomuutensa vuoksi. SHA-tiivistettä käytetään monissa tietoturvajärjestelmissä ja -protokollissa, kuten SSL:ssä, SSH:ssa ja Ipsecissä. [25.]

SHA-1-tiivistefunktio tuottaa 160-bittisen, 40 merkkiä pitkän tiivisteen. Kiinteämittaisuuden haittapuolena on, että kaksi eri lähtötilannetta voi tuottaa täsmälleen saman tiivisteen, joskin mahdollisuus päällekkäisyydelle on 2^{80}. Tiivistettä pidetään vielä toistaiseksi melko käyttökelpoisena, vaikkakin matemaattisten algoritmien avulla on onnistuttu tuottamaan päällekkäisyys jo 2^{69} yrityksen jälkeen. Murtoyritys voidaan tehdä myös sateenkaaritauluja (Rainbow table) hyödyntämällä, jolloin murrettavia tiivisteitä verrataan esimerkiksi etukäteen laskettujen sanakirjan sanojen tiivisteisiin. Tällaisten murtoyritysten tekeminen vaatii kuitenkin valtavasti työtä ja laskentatehoa, joten matalan profiilin suojaukseen, jota tämäkin opinnäytetyö edustaa, SHA-1 sopii hyvin. [26.]

Yksinkertaisimmillaan käyttäjän antama salasana tiivistetään PHP:n `sha1()`-funktiolla ja tiivistettä verrataan palvelimella olevaan salasanan tiivisteeseen. Jos merkkijonot ovat samat, on käyttäjä kirjoittanut oikean salasanan. Tietoturvallisuuden parantamiseksi salasanaan on suositeltavaa yhdistää lisämerkkijono, niin sanottu suola (salt), joka hankaloittaa varsinkin sateenkaaritaulujen käyttöä. Hyvä suola on satunnainen ja jokaiselle käyttäjälle erikseen generoitavissa. Saunantakana.fi-sovelluksessa suola luodaan uudelleen aina salasanan vaihdon yhteydessä seuraavalla tavalla:

```
$salt = substr(sha1(uniqid(rand(), true)), 0, 23);
```

jossa `uniqid()` luo aikaan perustuvan uniikin tunnisteen. Tunnisteesta generoidaan SHA-1-tiiviste, jota vielä lyhennetään tunnistamisen vaikeuttamiseksi. Käyttäjäkohtainen suola voidaan tallentaa erikseen tietokantaan, mutta tässä työssä se liitetään salasanan tiivisteen yhteyteen:

```
$hash = $salt . sha1($salt . $password);
```

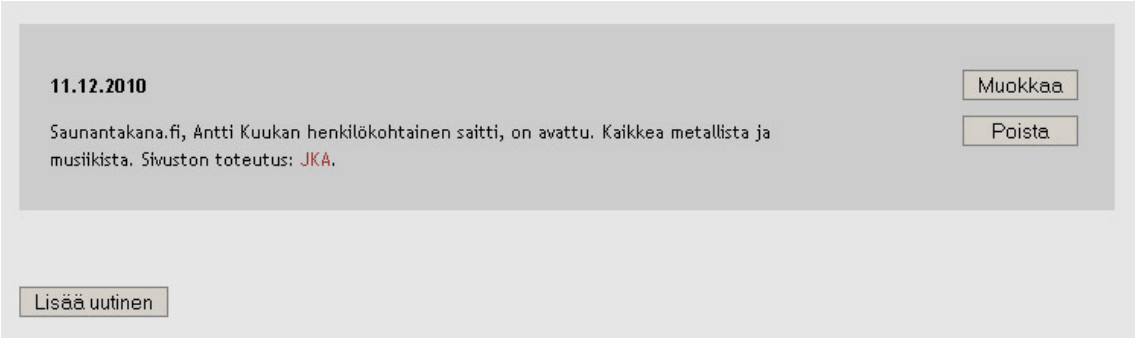
Tuloksena on 63 näennäissatunnaisesta merkistä koostuva merkkijono, jonka pituudesta ei voi suoraan päätellä suolan sijaintia eikä sitä, mitä tiivistysalgoritmia merkkijonon generoimiseen on käytetty. Toteutus hankaloittaa sekä sateenkaaritaulujen avulla tehtäviä murtoyrityksiä että alkuperäisten salasanojen vääriin käsiin joutumista, jos tunkeutuja pääsee käsiksi itse tietokantaan.

Järjestelmän suurin heikkous on salaamaton yhteys asiakkaan ja palvelimen välillä. PHP:n palvelinpohjaisuudesta johtuen asiakkaan lomakkeeseen syöttämä salasana välitetään selkokielisenä HTTP-otsakkeessa palvelimelle, jossa SHA-1-tiivistys vasta tapahtuu. Salasanan välityksen aikana on mahdollista, että tunkeutuja käyttää esimerkiksi kirjautumisen salakuuntelua (Packet Sniffing) tai välimieshyökkäystä (Man-in-the-middle Attack). Tällöin selkokielinen salasana joutuu hyökkääjän haltuun, eikä tiivistämisellä tai suolaamisella ole vaikutusta lopputulokseen. [27, s. 559.]

Suojattu yhteys voidaan muodostaa vain varmennepohjaisella salausprotokollalla, kuten TLS (Transport Layer Security) tai SSL (Secure Sockets Layer), tai kysymys–vastaus-todennuksella (Challenge-response Authentication). Mikään muu metodi ei takaa tietojen turvallista siirtymistä asiakkaan ja palvelimen välillä. Jopa salasanan tiivistäminen asiakkaspuolella, esimerkiksi JavaScriptin avulla, on hyödytöntä, jos jompaa kumpaa edellä mainittua tekniikkaa ei käytetä. [24, s. 109.] Saunantakana.fi-projektin puitteissa ei kumpaakaan metodia ehditty implementoimaan. Erityisesti kysymys–vastaus-todennuksen toteuttaminen on hyvin työlästä ja käytössä yleensä vain isoissa järjestelmissä, mutta sen sijaan tulevaisuuden kehitysideana on salatun yhteyden luominen palveluntarjoajalta ostettavan SSL-varmenteen avulla.

5 Hallintasivun toiminnallisuus

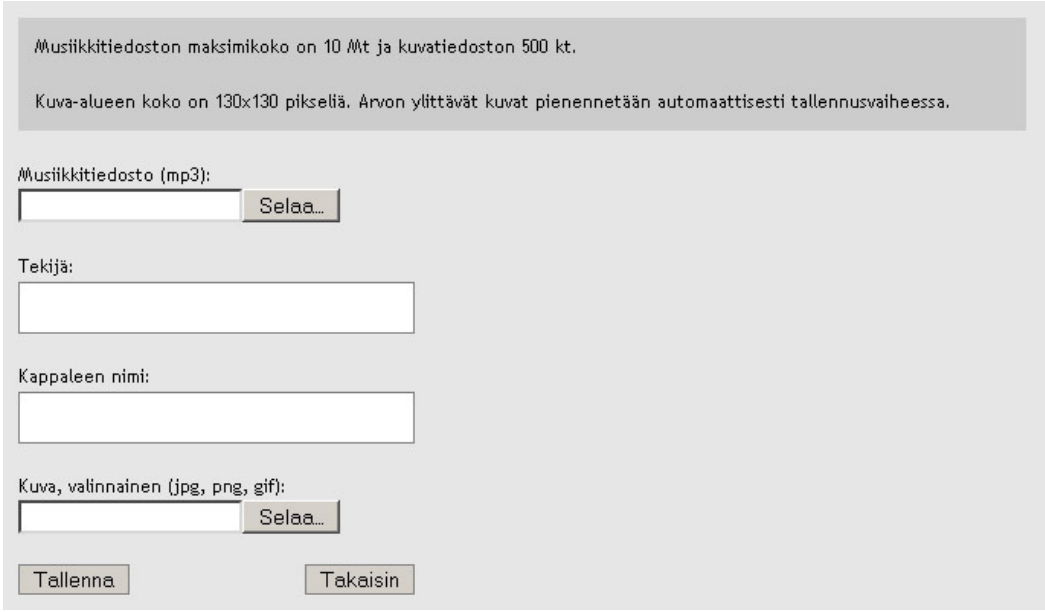
Sovelluksen hallintasivu ja julkinen osa ovat perustoiminnoiltaan ja ulkoasultaan toistensa kaltaisia. Hallintasivulla sisältöä pääsee luonnollisesti myös muokkaamaan, ja tämä tapahtuu kuvassa 7 näkyvien perusperiaatteiden mukaisesti. Kunkin alakategorian (uutiset, kuvat, musiikki, lyriikat, kalusto, vieraskirja, linkit ja bannerivaihto) ensimmäisellä sivulla näytetään kaikki kategorian alainen sisältö, jonka jälkeen käyttäjä voi lisätä uutta tai muokata ja poistaa olemassa olevaa dataa. Virhepoistojen estämiseksi käyttäjältä pyydetään vahvistus poistopaininikkeen painalluksen jälkeen.



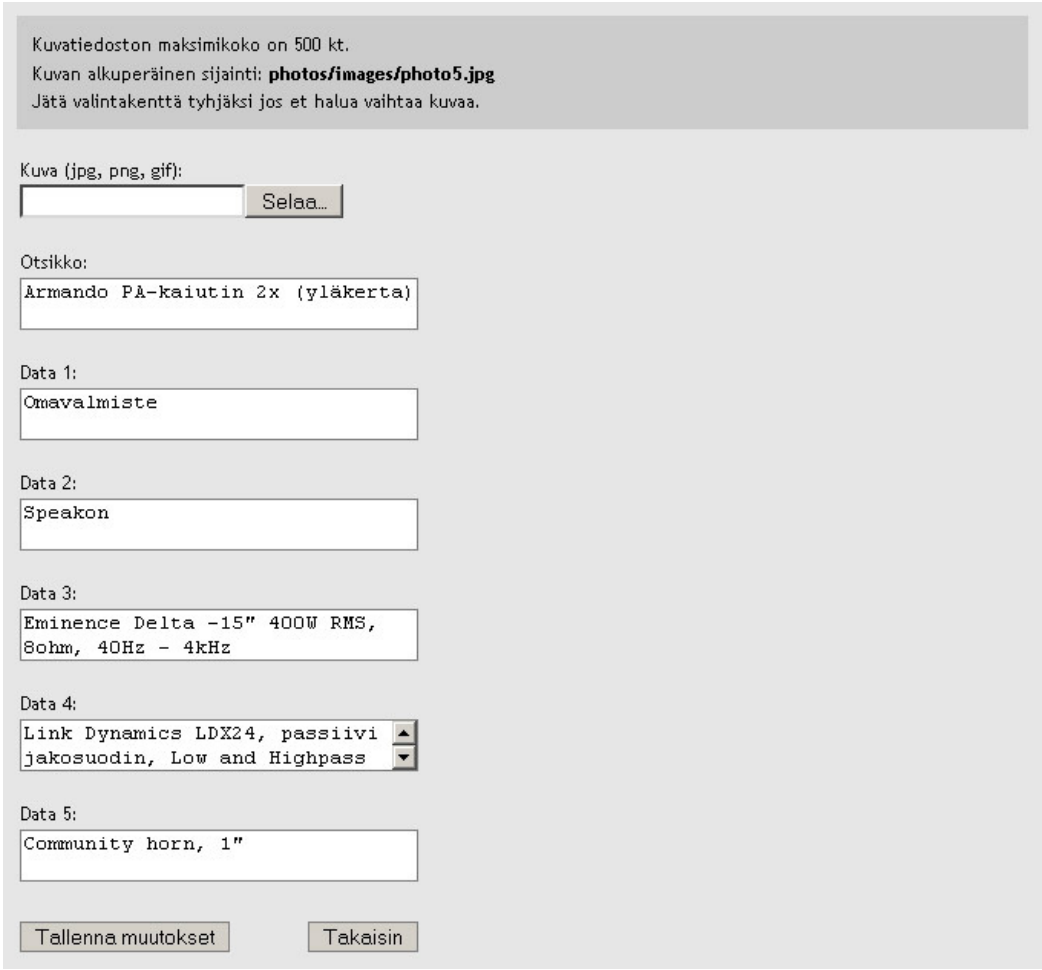
Kuva 7. Uutinen hallintasivustolla.

Valinnan tehtyään käyttäjä näkee toimintoa vastaavan näkymän. Kuvassa 8 nähdään lomake musiikkikappaleen lisäämiseksi. Jokaisen kentän täyttäminen on pakollista, minkä vuoksi tallenna-napin painalluksen jälkeen tarkistetaan ensin, että yksikään kenttä ei ole jäänyt tyhjäksi ja tämän jälkeen vielä kunkin kentän datan validius. Virheellistä sisältöä löydettäessä datan jatkokäsittely lopetetaan ja käyttäjää pyydetään korjaamaan ongelmakohta. Jos ohjeistuksen ja lisäinformaation antamisen on katsottu helpottavan käyttäjän toimintaa, esitetään nämä tiedot kyseessä olevalla sivulla ensimmäisenä, toiminnallisuuslomakkeen yläpuolella.

Tietojen muokkaamisen esimerkki löytyy kuvasta 9. Muokkaus toimii perusperiaatteiltaan kuten uuden tiedon lisäys, mutta muokkaussivulle siirryttäessä vanhat tiedot haetaan valmiiksi lomakkeen kenttiin.



Kuva 8. Musiikkikappaleen lisääminen.



Kuva 9. Tuotteen tietojen muokkaaminen kalustosivulla.

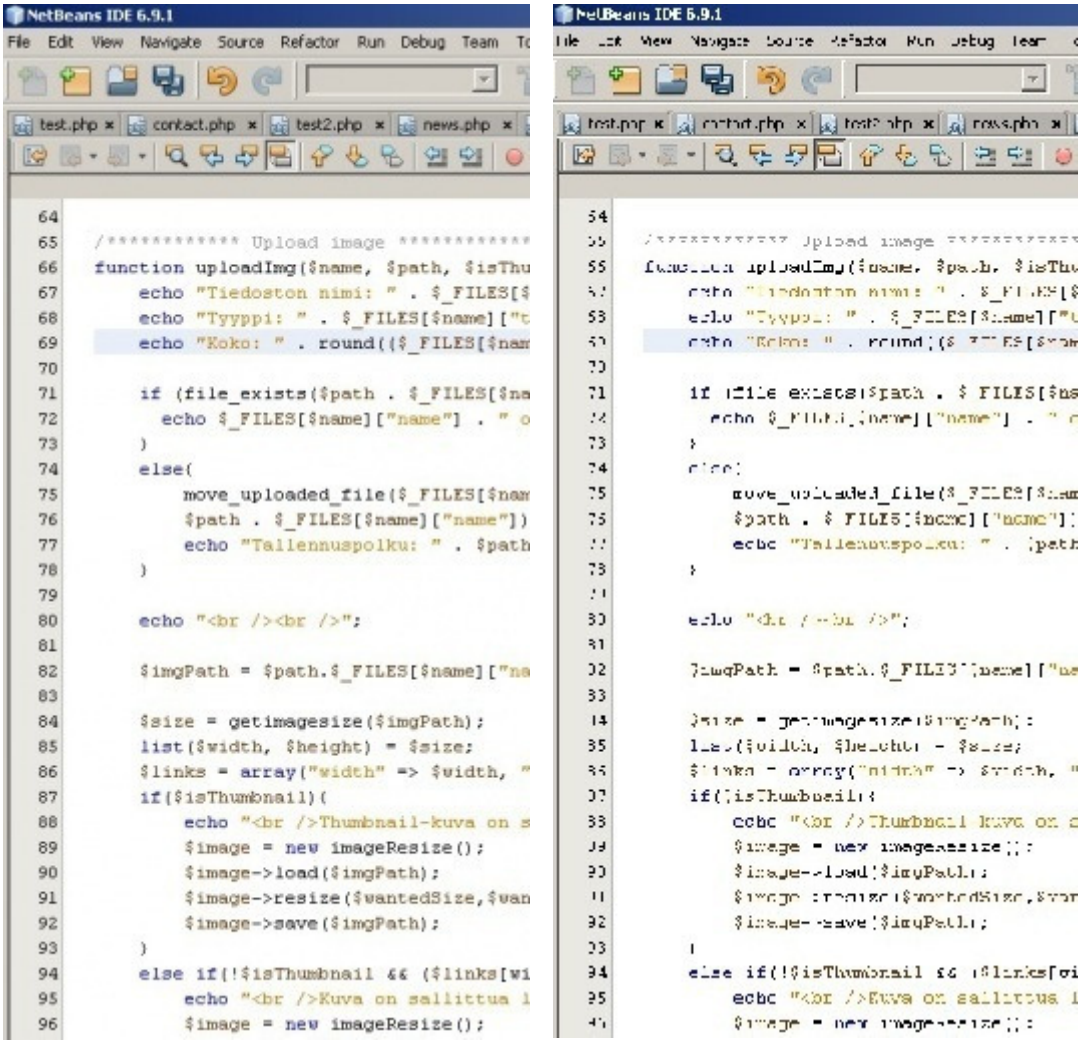
RSS-syötteet (Really Simple Syndication) madaltavat vierailijoiden sivustolle paluun kynnystä, sillä sovelluksen linkkiä ei enää tarvitse kaivaa kymmenien, jopa satojen muiden kirjanmerkkien seasta. Syötteidenlukijat ovat yleistyneet merkittävästi, ja esimerkiksi Mozilla Firefox -selaimessa sellainen on jo sisäänrakennettuna, mikä lisää entisestään RSS:n käytön hyödyllisyyttä kävijämäärän maksimoimisessa. Sivuston uutisosio päivittyy useiten ja sisältää vierailijoita kiinnostavaa tietoa, minkä vuoksi on luonnollista lisätä syötteet koskemaan tätä osiota. RSS on XML-muotoista dataa, jossa listataan muun muassa jokaisen syötteen nimi, kuvaus ja linkki sijaintiin. Aina kun uutisosion sisältöä muutetaan millään tavoin – uutta sisältöä lisätään tai vanhaa editoidaan tai poistetaan – rakennetaan RSS-tiedosto kokonaan uudelleen. Näin tiedosto on aina varmasti ajan tasalla myös poistettujen uutisten osalta ja samalla vältetään resurssien käytön kannalta epäedullinen vain muuttuneiden tietojen editointi.

5.1 Kuvien lisääminen

Sovelluksessa on kuvagalleria, johon asiakkaalla on mahdollisuus lisätä kuvia. Koska palveluntarjoajan PHP-palvelimella on käytössä GD-kirjasto (Graphics Draw), voidaan sovellukseen ladattavaa kuvaa muokata. GD-kirjasto tarjoaa mahdollisuuden luoda kuvia erilaisten viivojen ja muotoja avulla, ja lisäksi tämän työn kannalta hyödyllisiä toimintoja kuvien koon ja mittasuhteiden muokkaamiseen. Gallerian kuvat voidaan nähdä yhdellä kertaa 50 x 50 pikselin kokoisina niin sanottuina esikatselukuvina (thumbnail), joita napsauttamalla päästään tarkastelemaan valittua kuvaa isompana. Kuvien lisääminen on tehty joustavaksi siten, että käyttäjä voi itse lisätä erillisen esikatselukuvan, mutta tarvittaessa järjestelmä tekee sen alkuperäisestä kuvasta. Tällä tavoin otetaan huomioon käyttäjän kehittyminen sekä annetaan hänen valita tilanteen mukainen ratkaisutapa. Valikko olemassaolevan kuvan muokkaamiselle ohjeistuksineen nähdään kuvassa 3 sivulla 13.

Esikatselukuvan manuaalinen lisääminen on perusteltua silloin, kun alkuperäisestä kuvasta halutaan rajata vain pieni osa tai jos kuvan muoto poikkeaa paljon säännöllisestä nelikulmiosta. Automaattisesti generoitava esikatselukuva muodostetaan alkuperäisestä kuvasta pienentämällä reunat 50 pikselin pituisiksi, tarvittaessa kuvasuhteita venyttämällä. Tämän vuoksi ratkaisu ei aina ole esteettisesti toivottavin.

Sovelluksen aktiivinen sisältöala on 835 pikseliä leveä, ja visuaalisista syistä johtuen lisättävät kuvat eivät saa ylittää tätä arvoa. Käytettävyyden parantamiseksi käyttäjän ei tarvitse muistaa sallittuja raja-arvoja, vaan minkä tahansa kokoinen kuva on lisättävissä. Käyttäjän tallentama kuva viedään tilapäishakemistoon, jossa sen validius tarkistetaan muun muassa MIME-tyypin, tiedostopäätteen ja tiedostokoon perusteella. Jos kuvatiedosto on oikeellinen, tarkistetaan sen leveys GD-kirjaston `imagesx()`-funktiolla. Leveyden ylittäessä sallitun lasketaan uudet leveys- ja korkeusarvot säilyttäen silti alkuperäinen kuvasuhde sekä luodaan `imagecreatetruecolor()`-funktiolla uusi kuva-aihio, johon saadaan alkuperäisen kuvan sisältö `imagecopyresampled()`-funktion avulla. Kuvan koko on muutettavissa kahta vaihtoehtoista funktiota, `imagecopyresampled()` ja `imagecopyresized()`, käyttäen.



Kuva 10. imagecopyresampled- ja imagecopyresized-funktioiden vaikutus kuvanlaatuun.

Jälkimmäinen on nopea ja kevytkäyttöinen, mutta tarjoaa laadultaan heikon ja usein aliasoituneen lopputuloksen (kuva 10, oikea puoli), soveltuen täten resurssikriittisiin järjestelmiin. Sovelluksessa käytössä oleva ensimmäinen vaihtoehto tuottaa parempilaatuisia kuvia, mutta vastaavasti suuremman suoritintehon kustannuksella (kuva 10, vasen puoli). Koska sovellukseen lisätään kuvia suhteellisen harvoin, ei hetkellinen suoritinkuormitus ole ongelma. [28.] Onnistuneen koon muokkauksen jälkeen kuva tallennetaan lopulliseen hakemistoon ja käyttäjää informoidaan kuvakoon muutoksesta.

5.2 Soittolista

Sovelluksen dynaaminen sisältö tallennetaan tietokantaan, lukuun ottamatta soittolistaa, joka käyttää hyväkseen XML-pohjaista XSPF-formaattia (XML Shareable Playlist Format). Ratkaisuun päädyttiin XSPF:n monikäyttöisyyden ja siirrettävyyden vuoksi. Tarjolla olevat muut soittolistaformaatit ovat tietyin tavoin rajoittuneita, kuten esimerkiksi Microsoftin ja Applen kaupalliset ASX ja ITL/ITML, rakenteettomat M3U ja M4U ja binääristä muotoa käyttävä QuickTime. Tietoja ei myöskään haluttu tallentaa suoraan tietokantaan, sillä käytössä olevaa soittolistaa tulee voida suoraan hyödyntää myös muissa XSPF:ää tukevissa sovelluksissa. Liitännäisten avulla XSPF-soittolistoja voidaankin käyttää muun muassa WinAmp-, VLC- ja Foobar2000-mediasoittimissa, Mozilla Firefoxissa sekä WordPress-sisällönhallintaohjelmistossa.

Soittolistan tehtävänä on identifioida toistettavien kohteiden tyyppi (ääni, video, kuva, teksti, soittolista ja niin edelleen) ja määrittää kohteiden toistojärjestys. Metadatan, kuten musiikkikappaleen nimen ja tekijätiedon, tallennukseen soittolistaa ei tiukimman määritelmän mukaisesti tulisi käyttää, joskin XSPF tukee myös tätä ominaisuutta. Suositellumpana vaihtoehtona pidetään metatietojen hakemista Gracenoten ja MusicBrainzin kaltaisista ulkoisista tietolähteistä `link`- ja `meta`-elementtejä käyttäen. Musiikkilistan kannalta lähestymistapa soveltuu kuitenkin paremmin tunnetuille artisteille, joiden levyjen ja kappaleiden tiedot löytyvät tietokannoista jo etukäteen. Saunantakana.fi-sovelluksessa päädyttiin tämän vuoksi hyödyntämään soittolistan metatietoelementtejä (koodiesimerkki 1), jolloin kappaleen lisääjä itse määrittelee tarvittavat tunnistetiedot (`creator` ja `title`). [29.]

```
<?xml version="1.0" encoding="UTF-8"?>
<playlist xmlns="http://xspf.org/ns/0/" version="1">
  <title>Antin lista</title>
  <creator>Antti Kuukka</creator>
  <info>http://www.saunantakana.fi</info>
  <trackList>
    <track>
      <location>playlist/ST3rend.mp3</location>
      <creator>AK</creator>
      <title>Resolve</title>
      <image>playlist/images/rImg.jpg</image>
    </track>
    <track>
      <location>playlist/Ahjo-Demo.mp3</location>
      <creator>AK</creator>
      <title>Demo 5</title>
      <image>playlist/images/demo5.jpg</image>
    </track>
  </trackList>
</playlist>
```

Esimerkkikoodi 1. XSPF-tiedoston rakenne.

Soittimena käytetään Fabricio Zuardin ActionScript 2:lla tehtyä vapaan BSD-ohjelmistolisenssin alaista XSPF Web Music Playeria. Jos kappaleen tietoihin on lisätty `image`-elementti, näyttää soitin elementin määrittämän kuvan. Kuvan lisääminen tapahtuu samalla tavalla kuin galleriasivullakin, ja ohjelma pienentää kuvaa tarpeen vaatiessa. `location`-elementtiin voidaan lisätä musiikkitiedoston sijaan toinen soittolista, jolloin soitin toistaa kyseisen kappaleen kohdalle tullessaan linkitetyn soittolistan määrittämät kappaleet.

Palveluntarjoajan palvelimelle on asennettu DOM/XML-tuki (Document Object Model), joten XML-dokumentin käsittelyyn voidaan käyttää PHP:n monipuolista `DOMDocument`-luokkaa. Tällöin dokumentti jaetaan elementtiensä suhteen puumaiseksi rakenteeksi, jossa elementtien välillä voidaan liikkua vapaasti ja muutoksia tehdä minne tahansa (vanhempi-sisarus-notaation avulla). Yleensä dokumentti luetaan kerralla muistiin kokonaisuudessaan tai vaihtoehtoisesti käydään uudelleen läpi aina, kun sijaintia puuhierarkiassa halutaan muuttaa. Lähestymistapa ei ole kovin resurssiystävällinen, mutta pienikokoisten dokumenttien, joiksi sovelluksen soittolistan voi lukea, käsittely on hyvin vaivatonta. [30.]

Kappaleiden välillä liikutaan `track`-elementtien järjestysnumeroiden perusteella. Uusi kappaletietue elementteineen luodaan `createElement`-, `createTextNode`- ja `appendChild`-jäsenfunktioilla, jonka jälkeen tietue voidaan lisätä XSPF-tiedostoon `insertBefore`-funktiolla tai korvata vanha tietue uudelle `replaceChild`-funktiolla. `removeChild` poistaa kappaletta koskevat elementit.

6 Yhteenveto

Opinnäytetyön tarkoituksena oli luoda julkaisufoorumi muusikko Antti Kuukan musiikille ja ajatuksille sekä samalla tutustua kokonaisvaltaisesti web-sovelluskehitysprosessin kulkuun. Koska jokainen osa-alue oli projektin totetuttajan vastuulla, tarjosi työ hyvän yleiskatsauksen teoreettisella tasolla graafiseen suunnitteluun, käyttöliittymä- ja tietorakennesuunnitteluun, vaatimusten määrittelyyn sekä ohjelmistokehityksen eri metodeihin, ja käytännön tasolla sovelluksen rakenneratkaisuihin, PHP:n SQLite- ja XML-tukeen sekä tietoturvaan ja myös oman ajankäytön arviointiin.

Sivupohjasta ja -rakenteesta pyrittiin tekemään niin yleiskäyttöinen, että projektiin kehitettyjä ratkaisuja voidaan hyödyntää myös tulevaisuudessa. Päätyminen SQLite-tietokantaan MySQL:n sijaan teki sovelluksesta kevyen, siirrettävän ja helposti varmuuskopioitavan. XML-pohjaisen soittolistaformaatin käyttäminen puolusti paikkaansa yleiskäytettävyydellään, mutta tarjosi samalla hyvän lisän opinnäytetyössä käytettäviin tekniikoihin ja myös opettavaisen kokemuksen SQL:n ja XML:n käytännön käsittelyeroista.

Asiakkaan lisätoiveet, kuten bannerinvaihtosivu, projektin ollessa jo loppusuoralla lisäsivät työn vaativuutta ja aikataulujen venymistä. Inkrementaalisesti ja iteratiivisesti tapahtuva kehitystyö mahdollisti lisäysten tekemisen vielä loppumetreilläkin, mikä toimi samalla osoituksena metodien hyödyllisyydestä. Joustavuuden vuoksi myös asiakastyytyväisyys, jota voidaan pitää tärkeimpänä mittarina, säilyi hyvänä.

Kokonaisuuden suurimpana puutteena voidaan pitää hallinnointisivun tietoturvan riittämättömyyttä suojaamattoman yhteyden osalta. Asiasta on kuitenkin keskusteltu asiakkaan kanssa, ja SSL-sertifikaatin osto palveluntarjoajalta lienee seuraava suurempi sivuston päivitystoimi. Asiakkaan kanssa on neuvoteltu myös uutisosion kehittämisestä enemmän blogimaiseksi viestikohtaisine yleisökommentteineen.

Yhteistyö asiakkaan kanssa sujui hyvin, ja hänen vahva visionsa sivuston toiminnallisuuksien suhteen helpotti alun suunnitteluprosessia. Ennalta määrätty näkemys toisaalta myös pitkitti prosessia, sillä etenkin ulkoasuun ja rakenteeseen

liittyvät detaljit piti toisinaan hioa tarkoin asiakkaan mielikuvia vastaaviksi. Tällaisenaan kokemus vastannee työelämän vaatimuksia ja on jo siksikin hyödyllinen.

Opinnäytetyö syvensi XHTML:n, PHP:n, CSS:n ja SQL:n osaamista, joista kaikkia olin aikaisemmin käyttänyt, mutten välttämättä yhdessä enkä tässä kokoluokassa. Työelämän tarpeita ja vaativampia kokonaisuuksia silmälläpitäen erityisesti huolellinen suunnittelu ja käytettävyyden läpi projektin kestävä huomioiminen pysyvät varmasti tärkeinä opinkappaleina myös tulevaisuudessa.

Lähteet

1 Extensible Markup Language (XML) 1.0 (Fifth Edition). 2008. Verkkodokumentti. W3C. <http://www.w3.org/TR/REC-xml/>. 26.11.2008. Luettu 10.1.2011.

2 XHTML 1.0 The Extensible HyperText Markup Language (Second Edition). 2000. Verkkodokumentti. W3C. <http://www.w3.org/TR/xhtml1/>. Päivitetty 1.8.2002. Luettu 13.1.2011.

3 Korpela, J. & Linjama, T. 2004. XHTML-käsikirja. Jyväskylä: Docendo.

4 Korpela, Jukka. 2008. CSS verkkosivujen muotoilussa. Jyväskylä: WSOY.

5 What is PHP? 2001. Verkkodokumentti. The PHP Group. <http://www.php.net/manual/en/intro-whatis.php>. Päivitetty 4.2.2011. Luettu 10.2.2011.

6 Usage of server-side programming languages for websites. 2009. Verkkodokumentti. W3Techs. <http://w3techs.com/technologies/overview/programming_language/all>. Päivitetty 10.2.2011. Luettu 10.2.2011.

7 Hipp, D. Richard. 2011. Distinctive Features of SQLite. Verkkodokumentti. <http://www.sqlite.org/different.html>. Luettu 21.1.2011.

8 Holzschlag, Molly. 2004. 250 HTML and Web Design Secrets. Indianapolis: Wiley Publishing, Inc.

9 Cotler, E. & Goto, K. 2004. Web ReDesign 2.0 Workflow that Works. eKirja. Berkeley: Peachpit Press.

10 Sinkkonen, Irmeli. 2006. Käyttöliittymät ja käytettävyys. Verkkodokumentti. <http://www.adage.fi/julkaisut/arkisto/kayttoliittymat_ja_kaytettavyys.html>. Luettu 6.1.2011.

11 SFS-EN ISO 9241-11. 2000. Verkkodokumentti. Suomen Standardisoimisliitto SFS ry. <http://sales.sfs.fi/servlets/ProductServlet?action=showquicksearch&keywords=ISO+9241-11&x=20&y=6>. Luettu 23.1.2011.

12 Nielsen, Jakob. 1993. Usability Engineering. San Francisco: Morgan Kaufman.

13 Brinck, T., Gergle, D. & Wood, S. 2002. Usability for the Web. London: Academic Press.

14 Metsämäki, Markku. 1995. Graafinen käyttöliittymä. Helsinki: Painatuskeskus.

15 Nielsen, Jakob. 2000. WWW suunnittelu. Helsinki: Edita Oy.

16 Connare, Vincent. 1997. Trebuchet Nation. Verkkodokumentti. <http://www.microsoft.com/typography/web/fonts/trebuche/default.htm>. Luettu 28.1.2011.

17 Makarov, Alexander. 2009. The Big PHP IDE Test. Verkkodokumentti. <http://www.smashingmagazine.com/2009/02/11/the-big-php-ides-test-why-use-oneand-which-to-choose/>. Luettu 11.2.2011.

18 Developing PHP Applications in NetBeans IDE. 2011. Verkkodokumentti. NetBeans. <http://netbeans.org/kb/trails/php.html>. Luettu 14.2.2011.

19 Siedler, Kai. 2011. XAMPP. Verkkodokumentti. <http://www.apachefriends.org/en/xampp.html>. Luettu 14.2.2011.

20 Zuppa, Frederico. 2010. Agile is Iterative and Incremental Development. Verkkodokumentti. <http://agilebooknote.blogspot.com/2010/02/agile-is-iterative-and-incremental.html>. Luettu 15.2.2011.

21 Cockburn, Alistair. 2007. Incremental versus Iterative Development. Verkkodokumentti. <http://alistair.cockburn.us/Incremental+versus+iterative+development>. Luettu 15.2.2011.

22 Shiflett, Chris. 2005. Essential PHP Security. Sebastopol: O'Reilly.

23 Dickinson, Pax. 2005. Top 7 PHP Security Blunders. Verkkodokumentti. <http://articles.sitepoint.com/article/php-security-blunders>. Luettu 18.2.2011.

24 Snyder, C. & Southwell, M. 2005. Pro PHP Security. New York: Apress.

25 McGlinn, James. 2007. Password Hashing. Verkkodokumentti. <http://phpsec.org/articles/2005/password-hashing.html>. Luettu 21.2.2011.

26 Schneier, Bruce. 2005. Cryptanalysis of SHA-1. Verkkodokumentti. <http://www.schneier.com/blog/archives/2005/02/cryptanalysis_o.html>. Luettu 22.2.2011.

27 Gilmore, Jason. 2008. Beginning PHP and MySQL. New York: Apress.

28 GD and Image Functions. 2001. Verkkodokumentti. The PHP Group. <http://www.php.net/manual/en/ref.image.php>. Päivitetty 18.2.2011. Luettu 23.2.2011.

29 Gonze, L., Friedrich, M. & Kaye, R. 2006. XSPF Version 1. Verkkodokumentti. <http://xspf.org/xspf-v1.html>. Luettu 25.2.2011.

30 Document Object Model (DOM). 2009. Verkkodokumentti. W3C. <http://www.w3.org/DOM/>. Luettu 1.3.2011.

Käytettävyystarkistuslista

A Detailed, General-Purpose Checklist

Architecture and Navigation

- ☐ Does the structure fit the purpose?
- ☐ Is the navigation scheme clear?
- ☐ Where are you?
- ☐ How do you find what you want?
- ☐ Is there a reasonable number of navbar choices?
- ☐ Are navbar choices logically ordered?
- ☐ Do link names match page names?
- ☐ Are links clearly marked?
- ☐ Is there a clearly marked link back to the home page?
- ☐ Is there an option to search for information?
- ☐ Is there a site map?
- ☐ Does every page make it clear which web site you're in?
- ☐ Does the user have control over navigation?

Layout and Design

- ☐ Does page size exceed window size?
- ☐ Is layout consistent between pages?
- ☐ Is there a clear focal point on each page?
- ☐ Does the layout work visually?
- ☐ Is alignment used effectively?
- ☐ Is grouping used effectively?
- ☐ Is there good contrast?
- ☐ Is the layout cluttered?
- ☐ Is it aesthetically pleasing?

Content

- ☐ Is the text clear and concise?
- ☐ Is text organized in small chunks?
- ☐ Are there spelling or grammar errors?
- ☐ Do pages include introductory text?
- ☐ Do multimedia components support the task?
- ☐ Are units of measure clear and unambiguous for international use?
  - ☐ Date and time? ☐ Phone numbers? ☐ Address and postal codes?

Forms and Interaction

- ☐ Do forms support the task?
- ☐ Do dialogues follow a logical progression?
- ☐ Is it clear where to go next?
- ☐ Are dialogue methods concise and consistent?
- ☐ Are form elements used properly?
- ☐ Are elements grouped properly?
- ☐ Are there clear Submit buttons?

Lähde: Brinck, T., Gergle, D. & Wood, S. Usability for the Web, s. 412.

Graphics

- ☐ Is image quality adequate?
- ☐ Do the images include alternate text?
- ☐ Do the images include size information?
- ☐ Do the images use a consistent light source?
- ☐ Are images stored for maximum compression?
- ☐ Is mouse-over feedback provided? Is it useful?
- ☐ Are animations useful? Are there too many? Are they properly compressed?

Color

- ☐ Is the choice of colors appropriate for site?
- ☐ Are too many colors used?
- ☐ Are colors used consistently?
- ☐ Are graphics colors dithered?
- ☐ Do color choices work in grayscale?

Typography

- ☐ Is the text legible?
- ☐ Is the font size large enough?
- ☐ Is the font color appropriate and is there sufficient contrast?
- ☐ Is the text formatted for 10 to 12 words per line?
- ☐ Are there sufficient margins?
- ☐ Are typefaces used properly and consistently?

Error Tolerance

- ☐ Do users need to remember items across pages or sessions?
- ☐ Are confirmations provided before risky or costly actions?
- ☐ Are risky or costly actions reversible?
- ☐ Are entry errors caught locally?
- ☐ Do error pages provide useful information?
- ☐ Do search-error pages provide search broadening tips?
- ☐ Is help available?
- ☐ Is help task-oriented?
- ☐ Is help contextual?

Platform and Implementation

- ☐ Is load-time fast enough? Does it load in 3 to 15 seconds?
- ☐ Do all the links work?
- ☐ Are there broken images?
- ☐ Are pages written to be found by search engines?
- ☐ Does the site work with user's browser?
- ☐ Does the site work with user's hardware platform?
- ☐ Does the site work on high- and low-resolution monitors?
- ☐ Are nonstandard plug-ins required? Are they necessary or useful?

Lähde: Brinck, T., Gergle, D. & Wood, S. Usability for the Web, s. 413.