Lappeenrannan teknillinen yliopisto
Teknistaloudellinen tiedekunta
Tietotekniikan koulutusohjelma

KANDIDAATINTYÖ

WWW-pohjaisen julkaisurekisteriohjelmiston suunnittelu ja toteutus

Loppuraportti

Kandidaatintyön aihe on hyväksytty 27.5.2009.

Työn tarkastajana toimii yliassistentti Uolevi Nikula.

Lappeenrannassa 25.8.2009

Hannu Uskelin
Op.num. 0210815

TIIVISTELMÄ

Lappeenrannan teknillinen yliopisto
Teknistaloudellinen tiedekunta
Tietotekniikan koulutusohjelma

Hannu Uskelin

WWW-pohjaisen julkaisurekisteriohjelmiston suunnittelu ja toteutus

Kandidaatintyö

2009

43 sivua, 10 kuvaa, 2 liitettä

Tarkastaja: yliassistentti Uolevi Nikula

Hakusanat: Julkaisurekisteri, julkaisuohjelmisto, Ajax
Keywords: Publication register, publication software, Ajax

Työn tavoitteena oli suunnitella ja toteuttaa uudistettu versio Lappeenrannan teknillisen yliopiston kirjaston käyttämästä WWW-pohjaisesta julkaisurekisteriohjelmasta. Työssä käsitellään muuttuneita tarpeita ja syitä, jotka johtivat uuden ohjelmiston toteuttamiseen. Ohjelmiston uuteen versioon toteutettiin uusi ohjelmistoarkkitehtuuri, joka perustuu eri toiminnot toteuttaviin komponentteihin rajapintoineen. Tässä raportissa kuvataan ohjelmistoon liittyvät vaatimukset, projektin toteutukseen käytetyt tekniset ratkaisut sekä ohjelmiston toiminnallisuus käyttöliittymäkuvineen. Lisäksi käsitellään Ajax-tekniikoiden hyödyntämistä ohjelmiston käyttöliittymän vuorovaikutteisuuden parantamiseen. Lopuksi käsitellään toteutunutta projektia ja lopputuloksen merkitystä.

ABSTRACT

Lappeenranta University of Technology
Faculty of Technology Management
Degree program of Information Technology

Hannu Uskelin

The design and implementing of WWW based publication register application

Candidate work

2009

43 pages, 10 figures, 2 appendices

Examiner: Senior Assistant Uolevi Nikula

Keywords: Publication register, publication software, Ajax

The goal of this project was to design and implement a new version of WWW based publication register application used by Lappeenranta University of Technology library. This work describes changed needs and reasons which led to the new implementation. The new application uses a new software architecture that is based on different components with interfaces. This report describes definitions of requirements, used technology and application functionality with user interface figures. In addition, there is described how Ajax technologies were used to improve user interface interactions. In the end there is the conclusion of the finished project and meaning of final results.

ALKUSANAT

Haluan lausua kiitokset työnantajalleni Content Bakery Oy:lle opiskeluani tukevista joustavista työajoista.

Lappeenrannassa, 25.8.2009,

Hannu Uskelin

SISÄLLYSLUETTELO

SYMBOLILUETTELO

AJAX	Asynchronous JavaScript and XML
CSS	Cascading Style Sheet
DOM	Document Object Model
HTML	Hypertext Markup Language
ISAPI	Internet Server Application Programming Interface
LDAP	Lightweight Directory Access Protocol
MD5	Message-Digest algorithm 5
PCRE	Perl Compatible Regular Expressions
PDF	Portable Document Format
PEAR	PHP Extension and Application Repository
PHP	PHP: Hypertext Preprocessor
SQL	Structured Query Language
W3C	World Wide Web Consortium
XHTML	Extensible Hypertext Markup Language
XML	eXtensible Markup Language
XSS	Cross site scripting

1. JOHDANTO

1.1. Tausta

Työssä toteutettiin Lappeenrannan teknillisen yliopiston kirjastolle uusi versio vanhasta käytössä olleesta julkaisurekisteriohjelmistosta. Julkaisurekisteriohjelmisto on WWW-pohjainen järjestelmä, jonka kautta Lappeenrannan teknillisen yliopiston henkilökuntaan kuuluvat käyttäjät voivat itse tallentaa julkaisurekisteriin eri julkaisujen tiedot, joissa ovat itse yhtenä julkaisun tekijänä mukana. Julkaisurekisteriohjelmiston avulla on seurattavissa eri osastoilla työskentelevien henkilöiden julkaisemat julkaisut ja näiden tietojen pohjalta tuotetaan vuosittain luettelot henkilökunnan julkaisuista. Julkaisutietojen pohjalta määräytyy mm. eri osastoille jaettavat tukirahat. Luettelot henkilökunnan julkaisuista on saatavissa yliopiston kirjaston WWW-sivuilta.

Vanha julkaisurekisteriohjelmisto oli valmistunut kahden henkilön tekemänä erikoistyönä helmikuussa 2004. Ohjelmisto oli palvellut hyvin tarkoitustaan ja sitä haluttiin käyttää myös jatkossa. Muuttuneiden tarpeiden ja uusien toiveiden myötä ohjelmisto haluttiin muuttaa vastaamaan paremmin nykytilanteen tarvetta. Julkaisuista tallennettaviin tietoihin ja luokitteluihin oli tullut muutoksia sekä tiedekuntien mukaan tulo haluttiin lisätä ohjelmiston käyttämään organisaatiohierarkiaan. Samalla nähtiin tarpeelliseksi toteuttaa ja korjata vanhan ohjelmiston käytössä esiin nousseita parannusehdotuksia ja ongelmakohtia.

Alkuperäiseen kehityssuunnitelmaan kuului muutosten toteuttaminen vanhaan olemassa olevaan ohjelmistoon. Syvällinen analyysi vanhan julkaisurekisteriohjelmiston teknisestä toteutuksesta osoitti siinä olevan monia ongelmakohtia, jotka vaikeuttaisivat jatkokehitystä usealla tavalla. Esityslogiikka ja sovelluslogiikka olivat sekaisin samoissa ohjelmakooditiedostoissa tehden ohjelmakoodista vaikeaselkoista. Ohjelmakoodi sisälsi runsaasti samankaltaisia toistuvia rakenteita, jolloin tarvittavat muutokset olisivat kohdistuneet muutaman rakenteen sijaan useisiin kasvattaen virheiden riskiä. Jatkokehityksen kannalta ongelmallista oli myöskin riittävän dokumentaation puuttuminen ja ohjelmakoodin vähäinen kommentointi. Tarvittavien teknisten muutosten toteuttaminen vanhaan ohjelmistoon vaikutti olevan liian hankalaa ja riskialtista. Täten ilmenneiden ongelmakohtien pohjalta tehtiin päätös suunnitella ja toteuttaa ohjelmisto uudella tavalla tuottaen samalla hyvän pohjan tulevaisuuden jatkokehitykselle.

1.2. Tavoitteet ja rajaukset

Projektin tärkeimpänä tavoitteena oli toteuttaa ohjelmistosta uusi versio palvelemaan nykytilanteen muuttuneita tarpeita sekä muodostamaan hyvä pohja tulevaisuuden tarpeille. Tavoitteena oli toteuttaa ohjelmiston uuteen versioon kaikki vanhan ohjelmiston toiminnallisuudet sekä lisätä muutamia uusia toimintoja samalla säilyttäen kaikki aiemmin tallennettu julkaisuja koskeva tieto. Kaikki vanhassa ohjelmistossa ilmenneet ongelmakohdat tuli ratkaista. Koska ohjelmistoon on odotettavissa tarvetta erilaisille muutoksille myös lähivuosina, on yhtenä keskeisenä projektin tavoitteena toteuttaa uudesta ohjelmistosta helposti jatkokehitettävissä oleva.

Vanhasta ohjelmistosta saatujen käyttökokemusten pohjalta pohdittiin kirjaston henkilökunnan kanssa mahdollisia ongelmakohtia ja muutostoiveita sekä mietittiin sopivia ratkaisuja niihin. Ohjelmiston parempaa jatkokehitettävyyttä ajatellen suunnitteluvaiheessa pyrittiin ottamaan huomioon myös kehitysideat, joita ei päätetä ainakaan toistaiseksi toteuttaa. Ohjelmiston uudelleensuunnittelu mahdollisti myös paremmat mahdollisuudet toteuttaa uudet muutostoiveet. WWW-palvelin, jolla vanha ohjelmisto toimii, on riittävä palvelinohjelmistojen osalta ja uuden ohjelmiston tuli toimia moitteetta samalla palvelinalustalla. Toteutuksen kannalta ei ollut tarvetta ottaa käyttöön uusia tekniikoita, vaan haluttuun tavoitteeseen pyrittiin paremmalla ohjelmistosuunnittelulla ja tehokkaammalla olemassa olevien tekniikoiden käytöllä.

Vanha ohjelmisto käytti Lightweight Directory Access Protocol (LDAP) hakemistosta haettuja henkilöiden sähköpostiosoitteita käyttäjien tunnistamiseen. Uudessa ohjelmistossa LDAP:n käyttö päätettiin jättää pois todettaessa sen soveltuvan huonosti sille osoitetun ongelman ratkaisemiseen. Vanhassa ohjelmistossa LDAP hakemiston käyttäminen aiheutti ongelman tilanteissa, joissa hakemistosta ei löytynyt henkilön nimeä tai nimiä löytyi useampia. Tällöin henkilön sähköpostiosoitetta kysyttiin käyttäjältä, joka ei välttämättä tiennyt oikeaa sähköpostiosoitetta, jolloin julkaisurekisteriin saattoi tallentua ajan myötä samoja henkilöitä eri sähköpostiosoitteilla. Toimivan toteutuksen toteuttaminen uuteen ohjelmistoon tulisi viemään liikaa aikaa. Yhtenä projektin tavoitteena oli myös tehdä mahdollisimman hyvin paikkaansa pitävä toteutuksen työaika-arvio sekä pysyä aikataulussa toteutuksen osalta. Työaika-arviota verrattiin kirjattuun tuntikirjanpitoon.

1.3. Työn rakenne

2. Tutkimusongelman esittely

Luku 2 sisältää tutkimusongelman esittelyn muuttuneiden tarpeiden ja muutostoiveiden osalta sekä kuvausta vanhassa ohjelmistossa ilmenneistä ongelmakohdista.

3. Käytetyt tekniikat

Luku 3 sisältää kuvausta projektissa käytetyistä tekniikoista.

4. Ratkaisutavan esittely

Luku 4 sisältää kuvauksen uudesta ohjelmistoarkkitehtuurista ja sen komponenteista sekä kuvauksen teknisistä ratkaisuista.

5. Tekninen toteutus

Luku 5 sisältää kuvauksen projektin kulusta, työaika-arvion toteutumasta ja käyttöliittymän toiminnallisuuksista.

6. Toteutuneen ratkaisun merkitys

Pohdintaa asioista, joita ratkaisulla saavutettiin.

7. Johtopäätökset

Yhteenveto esitetyistä asioista.

2. TUTKIMUSONGELMAN ESITTELY

Kahden opiskelijan tekemä erikoistyönä vuonna 2004 valmistunut julkaisurekisteri-ohjelmisto on palvellut hyvin tarkoitustaan, mutta julkaisuista tallennettavia tietoja koskeviin määräyksiin on tullut muutoksia, joiden seurauksena julkaisurekisteriohjelmistoa haluttiin muuttaa vastaamaan paremmin nykyistä tarvetta. Samalla haluttiin myös tehdä parannuksia ohjelmiston käytettävyyteen sekä korjata muutamia ilmenneitä ongelmakohtia.

2.1. Muuttuneet tarpeet ja muutostoiveet

Julkaisurekisteriin tallennettavat julkaisujen tiedot on aiemmin luokiteltu kuuteen eri kategoriaan: referoidut tiedelehdet, konferenssijulkaisut, patentit, kirjat, muut tieteelliset, julkaisut ja muut julkaisut. Tiedot referoiduista tiedeartikkeleista on aiemmin tallennettu referoitujen tiedelehtien kanssa samaan. Nyt nämä haluttiin luokitella erilleen toisistaan. Myös väitöskirjojen tiedoille haluttiin oma luokitus, kun ne olivat aiemmin tallennettuna muut julkaisut luokittelun alla. Uudet luokittelut lisätään myös mukaan julkaisurekisteri-ohjelmiston ylläpito-osion tilastoihin, julkaisuluetteloon ja julkaisulistauksiin.

Yliopiston henkilökuntaan kuuluvat julkaisujen tekijät työskentelevät aina jonkin yliopiston organisaation alaisena, joten julkaisurekisteriohjelmistoon tallennetut julkaisujen tiedot ovat tallennettu aina jonkin organisaation julkaisuksi. Organisaatio-hierarkia koostuu osastoista, laitoksista ja laboratorioista. 1.1.2007 lähtien LTY:ssä toimii kolme tiedekuntaa. Muutos haluttiin päivittää myös julkaisurekisteriohjelmistoon siten, että organisaatiohierarkia olisi edelleen kolmetasoinen ja koostuisi seuraavasti: tiedekunnat, osastot ja laboratoriot.

Julkaisutietojen tallentamiseen tarkoitetut lomakkeet olivat julkaisujen mukaisesti luokiteltu kuudeksi eri lomakkeeksi siten, että jokaiselle julkaisutyypille oli oma lomake. Luokituksen kasvaessa kahdella, referoituja tiedeartikkeleita ja väitöskirjoja varten tarvittiin omat uudet lomakkeet. Lomakkeelle kirjoitetut tiedot tallennettiin heti tietokantaan lomakkeen lähettämisen jälkeen. Tähän haluttiin käyttäjien toiveesta muutos, joka mahdollistaisi tietojen tarkastelun ja mahdollisen korjaamisen ennen varsinaista tietokantaan tallentamista. Muutamat lomakkeissa olevat kentät eivät olleet pituudeltaan sopivia ja näihin haluttiin pieni korjaus. Lomakkeilla oli mahdollista tallentaa samat tiedot

useampaan kertaan ja ongelma esiintyikin helposti, kun useampi samassa julkaisussa mukana ollut henkilö tallensi julkaisun tiedot tietämättään jonkun toisen jo tallentaneen ne. Tähän haluttiin ratkaisu, joka estäisi samojen tietojen tallentamisen uudelleen.

Lomakkeen tietojen lähettämisen jälkeen ohjelmisto kertoi usein, että jonkun tekijäksi ilmoitetun henkilön nimeä ei löydy rekisteristä ja pyytää kirjoittamaan henkilön sähköpostiosoitteen. Tähän haluttiin myös jokin ratkaisu, jotta vastaavilta ilmoituksilta vältyttäisiin jatkossa. Julkaisutietojen tallentajat eivät aina tienneet muiden julkaisussa mukana olevien henkilöiden sähköpostiosoitteita, josta päästään seuraavaan ongelmakohtaa. Ohjelmisto tunnistaa henkilöt heidän sähköpostiosoitteistaan ja rekisteriin tallentuikin helposti samoja henkilöitä eri sähköpostiosoitteilla, jonka seurauksena julkaisuluetteloon listautui muutamia henkilöiden nimiä useampaan kertaan järjestelmän luullessa heitä eri henkilöiksi.

Tallennettaessa julkaisutietoja kategorioihin *Muut tieteelliset julkaisut* sekä *muut julkaisut,* julkaisutietoihin tallentuu automaattisesti merkkijonot *ISSN* ja *ISBN*. Mikäli tietojen tallentaja merkkasi nuo merkkijonot myös tallennuslomakkeelle, *ISSN* ja *ISBN* tulivat lukemaan julkaisuluettelossa kahteen kertaan. Julkaisutiedoissa ilmoitetut numeroiden ja volyymien lukumäärät ilmoitettiin julkaisuluettelossa vain pilkulla erotettuna. Näiden numeroiden eteen haluttiin merkinnät *vol.* ja *num.* Mikäli halutaan tallentaa useampia julkaisutietoja samoilla henkilötiedoilla, haluttiin jokin ratkaisu, jotta samoja henkilöiden tietoja ei tarvitse kirjoittaa aina uudelleen. Muutamat käyttäjät halusivat pystyä itse hakemaan omat julkaisutiedot kaikilta vuosilta ja tulostamaan ne tai tallentamaan omalle tietokoneelle tiedostona. Tutkimus- ja erillislaitoksille haluttiin mahdollisuus saada omat julkaisuluettelot vastaavasti kuin muiden osastojen kohdallakin.

Julkaisurekisteriohjelmiston ylläpito-osion saapuneet julkaisut näkymään haluttiin toiminto, jolla voidaan muuttaa kerralla kaikkien saapuneiden julkaisujen tilaa. Aiemmin jokaisen saapuneihin julkaisuihin listatun julkaisun perässä oli alasvetolaatikko, josta pystyi valitsemaan julkaisulle uuden tilan seuraavista vaihtoehdoista: *vastaanotettu*, *tarkastettu*, *valmis*, *julkaisurekisteriin* ja *poista*. Julkaisujen tilan muuttamiseksi listan alalaidasta piti vielä klikata painiketta *muuta julkaisujen tilat*. Esimerkiksi 100 julkaisun tilan muuttaminen vei huomattavasti aikaa.

Web-pohjaisessa käyttöliittymässä ennen lomakkeen valintaa täytyi kirjoittaa julkaisun tekijöiden lukumäärä julkaisutietojen tallentamiseen käytettävää lomaketta varten. Tämän jälkeen avatussa lomakkeessa näytettiin tämä lukumäärä rivejä tekijätietojen tallentamista varten. Toimintatapaan ehdotettiin dynaamisempaa ratkaisua.

2.2. Vanhan ohjelmiston ongelmakohdat

Teknisestä näkökulmasta vanhassa ohjelmistossa oli muutamia virheitä suunnittelussa ja toteutuksessa. Heti kehityspalvelimella näyttöruudulle tulostui lähes jokaisessa näkymässä huomautuksia ja virheilmoituksia alustamattomien muuttujien käytöstä. Ohjelmisto-arkkitehtuuri oli hyvin yksinkertainen. Lähes jokaista näkymää varten oli oma ohjelmakooditiedosto. Ohjelmiston jatkokehityksen kannalta ongelmallista oli kaiken PHP-ohjelmakoodin, SQL tietokantakyselyiden ja HTML-merkkauskielen sijainti sekaisin samoissa tiedostoissa. PHP-ohjelmoinnin yksi vahvuus on tosin juuri mahdollisuus kirjoittaa PHP-koodia HTML-kielen joukkoon, mutta mahdollisuutta tulisi käyttää siten, että HTML-merkkauskielen joukossa on vain PHP-kielen funktiokutsuja eikä suinkaan kaikkea sovelluslogiikkaa.

PHP-ohjelmoinnissa ei oltu käytetty juuri lainkaan omia funktioita, jonka seurauksena ohjelmakoodi sisälsi huomattavia määriä ylimääräistä toistoa. Esimerkiksi julkaisut oli luokiteltu kuuteen kategoriaan, jolloin ohjelmakoodin joukossa oli aina oma lohkonsa jokaista kategoriaa kohden sen sijaan, että olisi käytetty yhtä funktiota kaikille kategorioille ja funktion parametrejä käyttäen oltaisiin toteutettu nämä eri kategoriavaihtoehdot. Koodin toistossa pahin tilanne oli tilastojen kohdalla. Julkaisurekisteriohjelmistosta oli mahdollista tuottaa 32 erilaista tilastoa julkaisuista. Tämä oli toteutettu käyttäen 32 hieman toisistaan eroavaa noin 700-1100 ohjelmakoodiriviä sisältävää tiedostoa. Kun ohjelmistoon haluttiin lisätä kaksi uutta kategoriaa referoiduille tiedeartikkeleille ja väitöskirjoille, olisi näihin kaikkiin 32 tiedostoon pitänyt lisätä omat toisistaan hieman eroavat lohkot uusille kategorioille. Luetteloita ohjelmistossa oli mahdollista luoda 10 erilaista. Tämä oli toteutettu käyttäen yhtä ohjelmakooditiedostoa, mutta tietojenkäsittelyssä oli käytetty melko massiivista taulukkomuuttujien käsittelyä jonka seurauksena tehokkaallakin tietokoneella meni luetteloiden tuottamiseen useampi minuutti aikaa. Jatkokehityksen kannalta ongelmallista oli myöskin riittävän

dokumentaation puuttuminen ja ohjelmakoodin vähäinen kommentointi. Tietokannan suunnittelussa ei oltu käytetty lainkaan indeksointia. Jokaiselle tietokannan taululle oli määritelty pääavain, mutta ei muita indeksejä. Monissa tietokantakyselyissä tietoa haettiin kuitenkin käyttäen ehtoina muita tietokannan kenttiä kuin pääavainta. Tämän seurauksena tietokantakyselyt olivat muutamissa tapauksissa hyvinkin hitaita. Ongelma oli sama kuin yrittäisi etsiä puhelinluettelosta nimeä jollekin tietylle puhelinnumerolle. Tietokanta joutuu käymään jokaisen taulun tietueen lävitse tulosten löytämiseksi sen sijaan, että tulokset voitaisiin hakea hakemiston perusteella.

Alkuperäiseen projektisuunnitelman mukaisesti kaikki tarvittavat muutokset oli tarkoitus toteuttaa olemassa olevaan ohjelmistoon. Julkaisurekisteriohjelmiston syvällisempi analyysi osoitti kuitenkin, että muutosten toimivuuden ja mahdollisten jatkokehitysten mahdollistamisen vuoksi järkevä ratkaisu oli suunnitella ja toteuttaa koko ohjelmisto uusiksi. Kuten monet sovelluskehittäjät tietävätkin, niin tietyissä tapauksissa vanhan ohjelmiston korjaus tulee kalliimmaksi kuin kokonaan uuden toteutus.

3. KÄYTETYT TEKNIIKAT

3.1. PHP

PHP: Hypertext Preprocessor (PHP) on erittäin suosittu yleiskäyttöinen palvelinpuolen skriptikieli, joka tarjoaa nykyisin kaikki kehittyneen ohjelmointikielen ominaisuudet ja mahdollisuudet Web-palveluiden toteuttamiseen. PHP:n vahvuudet lyhyesti ovat käytännöllisyys ja mahdollisuudet. PHP on löyhästi tyypitetty kieli, jossa muuttujat tyyppeineen luodaan automaattisesti niitä käytettäessä ja tuhotaan automaattisesti skriptin suorituksen loputtua. PHP-ohjelmakoodi voidaan sisällyttää vapaasti Hypertext Markup Language (HTML) merkkauskielen joukkoon mahdollistaen helpon Web-sivun tehostamisen PHP:n avulla. Käytännöllinen PHP-skripti voi koostua lyhimmillään yhdestä rivistä, josta esimerkkinä kuvan 1 nykyisen päivämäärän tulostama skripti, joka voitaisiin lisätä tuollaisenaan Web-sivun HTML-merkkauksen joukkoon.

```
<?php echo date("d.m.Y"); ?>
```

Kuva 1: PHP ohjelmakoodiesimerkki

PHP:n tehokkuudesta ja mahdollisuuksista kertovat mm. sen tuki ja toiminta yli 25:n tietokantaratkaisun kanssa, mukaan lukien MySQL, PostgreSQL, MSSQL. PHP tarjoaa mahdollisuuden luoda ja käsitellä useita erityyppisiä tiedostoja kuten Flash-, kuva- ja PDF-tiedostoja. PHP osaa kommunikoida useiden eri protokollien kanssa, kuten IMAP, POP3, LDAP ja DNS. PHP tarjoaa tehokkaat tavat merkkijonojen jäsentämiseen ja käsittelyyn sekä valtavat määrät muita hyödyllisiä toiminnallisuuksia. PHP soveltuu erinomaisesti WWW-portaalien, keskustelupalstojen, verkkokauppojen ja nykyaikaisten sosiaalisten verkkosivustojen toteuttamiseen. [2, 8]

3.2. MySQL

MySQL tietokantaohjelmisto on erittäin suosittu, ilmainen ja avoimeen lähdekoodiin perustuva relaatiotietokanta. Se sai syntynsä sisäisestä yritysprojektista ja se julkaistiin julkisesti vuonna 1995. Ohjelmisto sai nopeasti valtavan suosion ja sen kehittämistä varten

perustettiin oma MySQL AB yritys. Tietokanta toimii palvelimella demonina, johon voidaan ottaa yhteys paikallisesti tai toiselta palvelimelta. Yhteyden muodostumisen jälkeen tietokantaan voidaan tehdä Structured Query Language (SQL) standardin syntaksin mukaisia kyselyjä. MySQL tietokantaohjelmiston vahvuuksiin kuuluvat ilmaisuuden lisäksi hyvin optimoitu toteutus, tehokkuus ja toiminta lukuisilla eri alustoilla. [2]

3.3. JavaScript

Netscape kehitti aikoinaan skriptauskielen Internet-ohjelmoijille helpottamaan HTML-merkkauskielen tagien muokkaamista dynaamisesti. Skriptauskielen tavoitteena oli mahdollistaa aiempaa käyttäjäystävällisempien verkkosivujen luominen ja se sai lopulta nimekseen JavaScript. Myöhemmin havaittiin, että verkkosivuja voitiin käsitellä olioina, joka johti Document Object Model (DOM) ohjelmointirajapinnan syntymiseen. JavaScript toi helpon tavan muokata verkkosivun sisältöä DOM-rajapinnan avulla. JavaScript joutui kuitenkin kehittäjien keskuudessa huonoon valoon lähinnä sen huonon virheidenjäljityksen ja kehitystyökalujen puuttumisen vuoksi. Lopputuloksena JavaScriptiä käytettiin lähinnä vain verkkosivuilla olevien lomakkeiden kanssa. Ajaxin myötä JavaScriptin merkitys on noussut kuitenkin täysin uudelle tasolle sen toimiessa Ajaxin tärkeimpänä komponenttina. Tällä hetkellä sen voidaan sanoa olevan yksi merkittävimmistä Web-palveluiden kehittämiseen liittyvistä ohjelmointikielistä. JavaScriptin merkityksestä kertoo sekin, että selainohjelmien kehittäjät tehostavat jatkuvasti selainohjelmiensa JavaScriptin suorituskykyä. [1, 7]

3.4. Ajax

Termi Asynchronous JavaScript and XML (Ajax) esiintyy nykyisin useissa nykyaikaiseen Web-palveluun liittyvissä asiayhteyksissä. Ajax ei ole mikään yksittäinen tekniikka, vaan joukko teknologioita, jonka ylivoimaisesti tärkein komponentti on JavaScript skriptauskieli ja sen XMLHttpRequest-olio. Itse Ajaxin käyttämissä teknologioissa ei ole varsinaisesti mitään uutta. JavaScript julkaistiin jo joulukuussa 1995 Netscape Navigator 2.0 Internet-selaimen myötä ja XMLHttpReques julkaistiin 1999 ActiveX-komponenttina Microsoft Internet Explorer 5 selaimen myötä. XMLHttpReques-olio lähettää POST tai GET tyyppisen HTTP-pyynnön WWW-palvelimelle ja vastaanottaa palvelimen palauttaman vastauksen. Vaikka termin nimi Ajax viittaa myös eXtensible Markup Language (XML)

merkintäkieleen, ei XML ole teknologian käytön kannalta mitenkään välttämätön. Ajaxin kanssa tiedonsiirrossa käytetty data voi olla XML:n sijaan missä tahansa merkkeinä esitettävässä muodossa. Tyypillisesti www-palvelimen palauttama data on JavaScript-komentosarja tai Extensible Hypertext Markup Language (XHTML) merkkauskieltä. [1, 7]

3.4.1. Mitä Ajax sitten tarkoittaa?

Lyhyesti ilmaistuna Ajax on asynkronisen tiedonsiirron selainohjelman ja WWW-palvelimen välillä mahdollistava teknologia, jota käyttämällä tietoa voidaan välittää selainohjelman ja WWW-palvelimen välillä ilman, että koko sivua tarvitsee ladata aina uudelleen. Ajaxia käyttämällä Web-sivustolla olevat toiminnallisuudet saadaan toimimaan kuten tavallisessa työpöytäsovelluksessa. Ajaxia ei pidä kuitenkaan sekoittaa pelkästään JavaScriptillä tehostettuun vuorovaikutteisuuteen, jossa käyttäjän tekemän valinnan jälkeen Web-sivun sisältö muuttuu ilman päivitetyn sivun uudelleenlatausta. Ajaxin käytöstä on kyse vain kun käyttäjän tekemän valinnan jälkeen selainohjelma lähettää HTTP-pyynnön palvelimelle ja palvelin palauttaa vastauksen takaisin selainohjelmalle, joka käsittelee datan ja päivittää sisällön sen mukaisesti. Tiedonvaihto selainohjelman ja palvelimen välillä tapahtuu asynkronisesti ja käyttäjältä piilossa. Tiedonvaihto voidaan suorittaa myös synkronisesti mikäli palvelimen palauttamien vastausten järjestyksellä on väliä. Asynkronisen tiedonsiirron etuna on se, että käyttäjä voi tehdä useita valintoja ilman, että jokaisen valinnan jälkeen selainohjelma jäisi odottamaan vastausta palvelimelta samalla torjuen käyttäjän antamat syötteet. [1, 7]

3.4.2. Miksi Ajax?

Ajaxin käytön tarkoitus on parantaa Web-sivustolta saatavaa käyttökokemusta tehostamalla vuorovaikutteisuutta ja toiminnallisuuksien nopeutta. Koska tieto käyttäjän tekemästä valinnasta voidaan lähettää palvelimelle ja muutokset voidaan näyttää ilman raskasta kokonaisen sivun uudelleenlatausta, saadaan Web-palvelun käytöstä nopeampaa ja mielekkäämpää. Ajaxin vahvuus on sen pohjautuminen jo olemassa olevien tekniikoiden käyttöön, joten se toimii kaikilla nykyaikaisilla selainohjelmilla ilman mitään lisäosien asentamista.

Ajaxin käytön merkittäviin pioneereihin kuuluva Google toi Ajaxin tunnetummaksi Gmail, Google Maps ja Google Suggest -palvelujen myötä. Ajaxin tuomaa hyötyä voidaan havainnoida vertailemalla esimerkiksi Google Maps karttapalvelua vanhaan keltaisten sivujen karttapalveluun. Keltaisten sivujen karttapalvelussa näytöllä näytettävän kartan vieritys tapahtui karttakuvan laidoilla olevia nuolia klikkaamalla. Jokaisen klikkaamisen jälkeen Web-sivu ladattiin uudelleen palvelimelta yhdellä pykälällä vieritetyllä karttakuvalla päivitettynä. Google Maps karttapalvelun toimintaperiaatteena on ladata etukäteen WWW-palvelimelta Ajaxia käyttäen näytöllä näkyvän karttakuvan ympärillä olevat kartan osat. Tällöin selainohjelmalle ladattu karttakuva on aina hieman suurempi, mitä siitä kerralla mahtuu näkyville. Karttaa hieman vieritettäessä näytölle piirtyy heti uutta karttakuvaa ilman ylimääräistä viivettä ja samalla palvelimelta ladataan taas karttakuvaa ympäröivät osat valmiiksi. Kartan vieritys on toteutettu JavaScriptillä siten, että karttaa voidaan vierittää hiiren painike alas painettuna ja karttakuvaa raahaamalla.

Toinen hyvä esimerkki Ajaxin käytöstä on ehdotettujen hakutulosten tarjoaminen Web-sivulla olevaan hakutoimintoon. Käyttäjän kirjoittama hakusana lähetetään aina jokaisen näppäinpainalluksen jälkeen palvelimelle, joka tyypillisesti hakee hakua vastaavat tulokset tietokannasta ja lähettää selainohjelmalle takaisin. Selainohjelma käsittelee tiedot ja näyttää ne hakukentän alapuolella hakuun vastaavina tuloksina.

3.5. Prototype ja Script.aculo.us

Prototype on yksi suosituimmista JavaScript-kehyksistä, joiden tarkoituksena on helpottaa dynaamisten Web-palveluiden kehitystä. Prototype tarjoaa luokkapohjaiset työkalut DOM ohjelmointirajapinnan käyttöön sekä kattavan Ajax-kirjaston. XML ja XHTML-dokumentit esitetään DOM:ssa puumaisena tietorakenteena, jonka muokkaamiseen DOM tarjoaa JavaScript:llä käytettävän rajapinnan. Vuorovaikutteisuuden lisäys verkkosivuille perustuu nykyisin yhä enemmän dokumentin muokkaamiseen DOM:n kautta. Prototypen Ajax-kirjastoa käyttämällä voidaan helposti toteuttaa selainohjelman ja WWW-palvelimen välinen kommunikaatio XMLHttpRequest-oliota käyttäen. Kirjasto tarjoaa monipuoliset parametrit HTTP-pyyntöjen lähettämiseen ja tietojen käsittelyyn. JavaScript-kehysten yksi etu on niiden toimiminen selainriippumattomana rajapintana selainohjelman ja DOM-rajapinnan välillä. Selainohjelmat, etenkin vanhemmat, käsittelevät DOM:ia hieman eri tavoin, jolloin selainohjelman eroavaisuudet eivät jää ohjelmoijan ratkaistavaksi

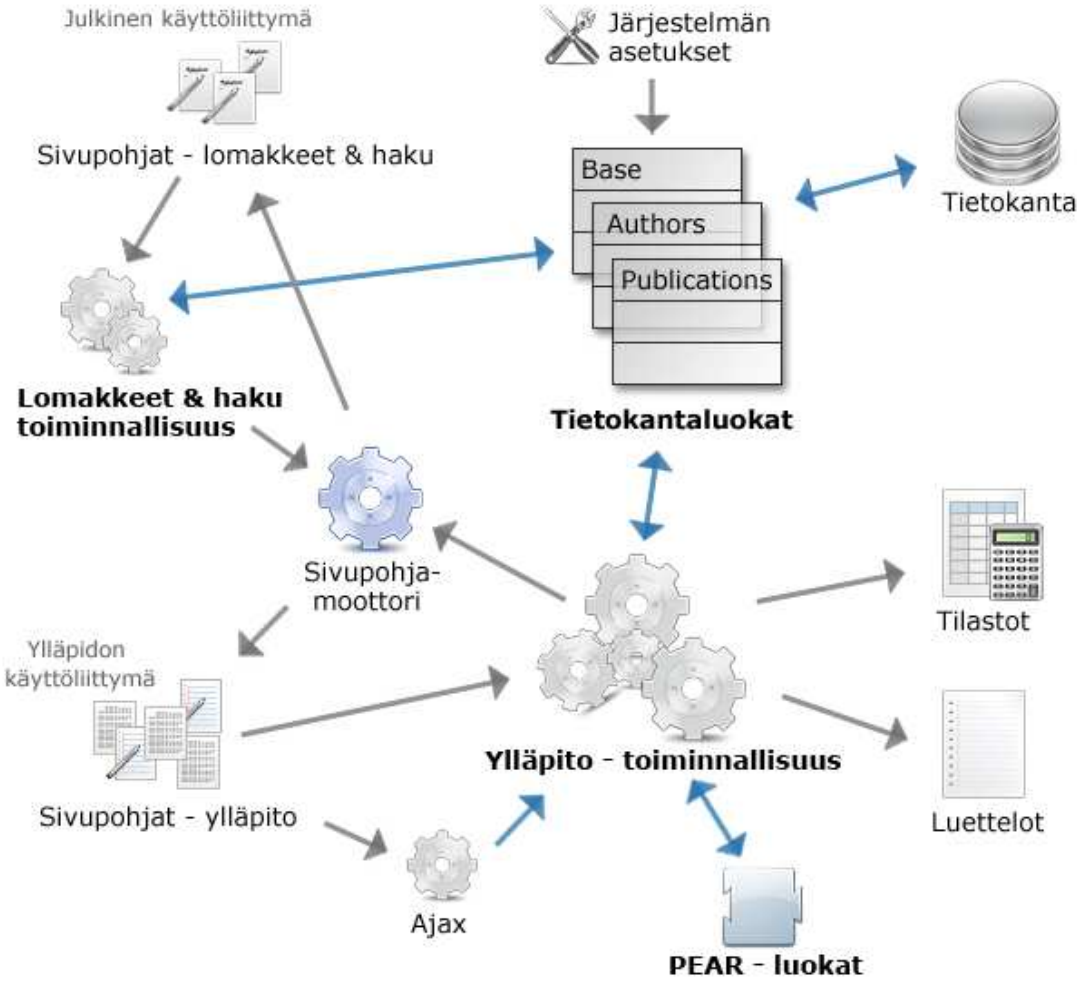
JavaScript-kehystä käytettäessä. Muita suosittuja JavaScript-kehyksiä ovat jQuery, Dojo ja MooTools. Script.aculo.us on suosittu JavaScript-käyttöliittymäkirjasto ja laajennus Prototypeen. Se tarjoaa kirjastot selainriippumattomien raahaa ja pudota toimintojen sekä visuaalisten tehosteiden toteuttamiselle. Tässä projektissa päädyttiin Prototypen käyttämiseen siihen pohjautuvan Script.aculo.us-kirjaston myötä sekä Prototypen ollessa allekirjoittaneelle jo entuudestaan tuttu. [5, 6]

3.6. Sivupohjamoottori

Tyypillinen tilanne dynaamisuuden lisäämisessä Web-sivuille on lisätä sovelluslogiikan ohjelmakoodia suoraan HTML-merkkauksen joukkoon haluttuihin kohtiin. Ratkaisu kuulostaa helpolta ja suoraviivaiselta. Dynaamisuuden lisääntyessä ja logiikan mennessä monimutkaisemmaksi tullaan lopulta siihen pisteeseen, jossa alkujaan selkeältä näyttänyt HTML-merkkauskieli on pieninä paloina sekaisin sovelluslogiikan ohjelmakoodin kanssa. Ongelma voidaan kuitenkin välttää ottamalla käyttöön sivupohjamoottori. Sivupohjamoottorin käytön tarkoitus on erottaa sovelluslogiikka kokonaan esityslogiikasta. Sivupohjamoottoria käytettäessä varsinainen sovelluslogiikan ohjelmakoodi sijaitsee omassa tiedostossaan ja esityslogiikan HTML-merkkauskieli on omassa sivupohjassa. Sivupohjamoottorien toimintaperiaate perustuu sivupohjaan erityisillä erotinmerkeillä merkittyjen tagien korvaamiseen varsinaisella datalla. Tageilla tarkoitetaan tässä yhteydessä erotinmerkeillä, kuten aaltosulkeilla ympäröityä merkkijonoa tai muuttujan nimeä. Itse sivupohja on tavallisesti yksittäinen kokonainen Web-sivu, jossa varsinaisen datan sijaan näytetään vain tageilla paikat varsinaiselle datalle. Sivupohjaan voidaan tehdä ulkoasumuutoksia ilman, että sovelluslogiikkaa tarvitsisi muuttaa lainkaan ja vastaavasti sovelluslogiikkaa voidaan muuttaa koskematta Web-sivun ulkoasuun. Web-sovelluksen kehitys nopeutuu ja helpottuu oleellisesti sovelluslogiikan ja esityslogiikan ollessa omina erillisinä kokonaisuuksina. [2]

4. RATKAISUTAVAN ESITTELY

Julkaisurekisteriohjelmisto rakentuu kuvan 2 mukaisesta ohjelmistoarkkitehtuurista, jossa ohjelmisto on jaettu eri toiminnot toteuttaviin komponentteihin rajapintoineen.



Kuva 2: Uusi ohjelmistoarkkitehtuuri

Järjestelmän ytimen muodostavat ylläpito-osion, lomakkeiden ja haun toiminnallisuudet toteuttavat sovelluslogiikan osat yhdessä tietokantaluokkien ja tietokannan kanssa. Toiminnallisuuksista vastaava sovelluslogiikka hakee ja tallentaa tietoja tietokantaan vain tietokantaluokkien kautta. Sovelluslogiikka käyttää PHP Extension and Application Repository (PEAR) –verkkosivuston tarjoamia luokkia tilastotietojen tallentamiseen Microsoft Excel taulukkolaskentaohjelman tiedostoiksi. Näyttöruudulle eri näkymiä

tuotettaessa sovelluslogiikan tietokannasta hakemat tiedot välitetään sivupohjamoottorin kautta sivupohjiin ja edelleen näyttöruudulle. Järjestelmäympäristöä koskevat asetukset ovat yhdessä omassa tiedostossa, josta luokat hakevat ne käyttöönsä. Monimutkaisempi sovelluslogiikka sijaitsee toiminnallisuuksista vastaavissa osissa luokkien vastatessa lähinnä tietojen tallentamisesta tietokantaan ja tiedon hakemisesta. Käyttäjän tekemät valinnat menevät toiminnallisuuksista vastaaville osille suoraan tai Ajax komponentin kautta käsiteltäväksi ja muuttuneet näkymät palautuvat sivumoottorin käsittelemien sivupohjien kautta takaisin käyttäjän selainohjelmaan. Arkkitehtuuri on osittain modulaarinen mahdollistaen esimerkiksi tietokantaohjelmiston tai Web-sivujen ulkoasun vaihtamisen toiseen hyvin helposti.

4.1. Järjestelmäympäristö

Julkaisurekisteriohjelmiston uusittu versio toteutettiin vanhan järjestelmän mukaisesti Web-sivustona, joka toimii Lappeenrannan teknillisen yliopiston WWW-palvelimella. Palvelimen käyttöjärjestelmänä toimii Linux ja palvelinohjelmistona Apache. Julkaisurekisteri käyttää olemassa olevaa MySQL tietokantaa. Uusi julkaisurekisteri-ohjelmisto perustuu jo olemassa olevien tekniikoiden tehokkaampaan hyödyntämiseen, jolloin vanhan palvelimen ominaisuudet ovat riittävät myös julkaisurekisteriohjelmiston uuteen versioon. Nämä olemassa olevat ohjelmistot ja tekniikat olivat allekirjoittaneelle jo entuudestaan tuttuja, joten ei ollut suositeltavaa ottaa ylimääräisiä riskejä käyttämällä vieraita tekniikoita. Samalla vältyttiin uusien ohjelmistojen asennukselta ja niiden mahdollisesti aiheuttamilta ongelmilta. Täten oli hyvä ratkaisu käyttää jo hyvin toimivaa alustaa julkaisurekisteriohjelmistolle. Julkaisurekisteri toimii kaikilla yleisimmillä Internet-selaimilla, joihin voidaan tällä hetkellä luokitella seuraavat selaimet: Microsoft Internet Explorer 6, 7 ja 8, Mozilla Firefox 3 ja Safari 3. Julkaisurekisteriohjelmiston käyttöön suositeltava näytön minimi tarkkuus on 1024x768 pikseliä.

4.2. Ohjelmointikielet

Julkaisurekisteriohjelmiston palvelinpuolen ohjelmointikielenä käytetään PHP-skriptauskielen versiota 4.2. Palvelimella ei ollut tukea PHP:n uudemmalle viidennelle versiolle, joten ratkaisu rajattiin käyttämään vanhempaa versiota. PHP 5 sisältää mm. paremmat olio-ohjelmointimahdollisuudet, mutta ei kuitenkaan mitään ratkaisevaa uutta

tätä projektia varten. Selainpohjaisessa käyttöliittymässä käytetään World Wide Web Consortium (W3C) standardin XHTML 1.0 Transitional mukaista XHTML merkkauskieltä sekä JavaScriptiä AJAX tekniikoiden toteuttamiseen. PHP ohjelmakoodi on muotoilultaan yleisen PEAR:n ohjelmointistandardin mukaista. PHP-ohjelmoinnissa on hyödynnetty PHP 4:n olio-ohjelmointimahdollisuuksia.

4.3. Ratkaisut muuttuneiden tarpeiden toteuttamiseksi

Julkaisurekisteriohjelmiston uuteen versioon lisättiin julkaisutiedoille kaksi uutta luokitusta: referoidut tiedeartikkelit ja väitöskirjat. Jokainen aiempi luokitus piti sisällään muista poikkeavaa luokitukselle ominaista tietoa, jolloin jokaiselle luokitukselle oli luontevaa luoda oma taulu tietokantaan. Täten väitöskirjoille ja referoiduille tiedeartikkeleille lisättiin tietokantaan myös omat taulut. Julkaisurekisterissä jokaisen julkaisun tiedot ovat liitetty vähintään yhden organisaation julkaisuksi. Organisaatiolistaan lisättiin tiedekunnat ja laitoksien nimet jätettiin pois. Muutoksen jälkeen organisaatiohierarkia oli edelleen kolmetasoinen. Tietokantaan oli aiemmin merkattu jokaiselle organisaatiolle tieto siitä, minkä organisaation alle se kuului. Ratkaisun hyvä puoli on sen mahdollistama rajaton hierarkiasyvyys. Ongelmana on kokonaiskuvan rakentaminen koko hierarkiasta. Yhden organisaation tietoja tarkastelemalla pystyy hierarkiasta määrittelemään vain sen, minkä organisaation alla kyseinen organisaatio on. Uutena ratkaisuna jokaiselle organisaatiolle määriteltiin hierarkiatunniste, joka määrää organisaation paikan hierarkiassa. Hierarkiatunniste on kuusinumeroinen tunniste, jonka kaksi ensimmäistä numeroa kertovat tiedekunnan, keskimmäiset kaksi osaston ja viimeiset kaksi numeroa laboratorion. Organisaation paikkaa organisaatiohierarkiassa pystyi nyt muuttavaan vapaasti vain hierarkiatunnistetta muuttamalla. Hierarkia on nyt määrätty kiinteästi kolmetasoiseksi, mutta tasojen määrän muuttamiselle ohjelmiston ylläpitäjän toimesta ei ole tarvetta, joten ratkaisu on rajoittunut, mutta riittävä.

Julkaisutietojen tallentamista varten väitöskirjoille ja referoiduille tiedeartikkeleille luotiin omat lomakkeet. Samalla lomakkeiden tekninen toteutus suunniteltiin kokonaan uusiksi. Kaikki kahdeksan eri lomakenäkymää käyttävät tietoja syötettäessä yhtä samaa sivupohjaa ja tietojen tarkistusvaiheessa toista yhteistä sivupohjaa. Samalla toteutettiin ominaisuus, jossa tietojen kirjoittamisen ja lomakkeen lähettämisen jälkeen annetut tiedot näytetään vielä uudelleen näyttöruudulla tarkistusta varten.

Ongelmatilanteeseen, jossa henkilö lisää jo lisätyn julkaisun tiedot uudelleen julkaisurekisteriin, ei löytynyt ratkaisua. Yksittäisen julkaisun yksilöi käytännössä vain artikkelin nimi. Artikkelin nimet ovat tyypillisesti hyvin pitkiä ja ne voi kirjoittaa eri tavoin, jolloin uutta julkaisua lisättäessä jo tallennettujen artikkelin nimien vertailu tallennettavaan ei välttämättä tuota tulosta vaikka tallennettujen joukossa olisikin jo kyseinen julkaisu. Ongelmaa kuitenkin lievennettiin lisäämällä käyttäjien käytettäväksi julkaisuhaku. Hakuun oman nimen syöttämällä näkee listan kaikista julkaisuista, joissa oma nimi esiintyy. Tällöin käyttäjällä on mahdollisuus ennen uusien julkaisutietojen tallentamista tarkistaa onko kyseinen julkaisu jo lisätty jonkun toisen tekijän toimesta.

4.4. Vanhassa ohjelmistossa olevien ongelmakohtien ratkaisu

Uuden ohjelmiston kehityspalvelimella oli käytössä kaikki mahdolliset virheiden ja huomautusten ilmoitukset. Tällä tavoin oli hyvin helppo havaita ja korjata yleisimmin esiintyvät ohjelmointivirheet.

Vanhassa ohjelmistossa oli useita ongelmakohtia varsinaiseen ohjelmakoodiin liittyen. Kaikki PHP-ohjelmakoodi, SQL-kyselykieli ja HTML-merkkauskieli sijaitsi sekaisin samoissa tiedostoissa, jonka seurauksena ohjelmakoodi oli hyvin sekavaa. Ongelma voitiin ratkaista helposti sivupohjien käytöllä eriyttämällä toimintalogiikka erilleen ulkoasun määrittelystä ja esityslogiikasta. Ohjelmiston uudessa versiossa käytetään sivupohjia ja PEAR Integrated Template sivupohjamoottoria. Toisena hyvänä vaihtoehtona ratkaisulle olisi ollut tunnettu Smarty sivupohjamoottori, mutta se rikkoo osittain sivupohjien periaatetta sallien omien ohjausrakenteiden sisällyttämisen sivupohjaan. Ratkaisua valittaessa valinta kallistui yksinkertaisemman ja allekirjoittaneelle jo entuudestaan tutun PEAR Integrated Template sivupohjamoottorin puolelle.

Ohjelmiston uudessa versiossa jokaista web-sivua ja toiminnallisuutta varten on oma sovelluslogiikka vain ja ainoastaan PHP-ohjelmakoodia sisältävissä omissa tiedostoissaan. Tämän sovelluslogiikan käytössä on kolme tietokantaluokkaa, jotka toimivat rajapintana tietokannan ja sovelluslogiikan välissä. Kaikki web-sivujen ulkoasun rakenteen muodostava HTML-merkkauskieli on sivukohtaisissa sivupohjissa. Sivujen visuaalinen ilme on määritelty käyttäen Cascading Style Sheet (CSS) tyylimäärittelyjä ja kaikki nämä

määrittelyt ovat yhdessä omassa tiedostossaan. Käyttöliittymän toiminnallisuuksiin liittyvät JavaScript ohjelmakoodit ovat omissa tiedostoissaan. Lopputulos on selkeästi jaoteltu kokonaisuus, jonka kehitystyö oli nopeaa ja johon muutosten teko on helppoa.

4.5. Tietokanta ja tietokantaoptimointi

Tietokantaohjelmistona käytetään olemassa olevaa MySQL 4 tietokantaa. Kaikki tietokannan taulut ovat tyypiltään InnoDB tauluja, jotka mahdollistavat mm. transaktioiden käyttämisen ja rivikohtaisen lukituksen. Yleisimmin käytetty MyISAM taulutyyppi ei näitä tue. Transaktioiden käyttäminen tietokantakyselyissä on hyvin perusteltua muutamien julkaisurekisteriohjelmiston toimintojen kohdalla. Kun yhden toiminnon toteuttamiseksi tehdään useampia toisistaan riippuvaisia tietokantakyselyitä, on ehdotonta tietojen eheyden kannalta, että kaikki tietokannan tietoja muuttavat kyselyt onnistuvat tai mitään muutoksia ei tehdä. Rivikohtaisesta lukituksesta on hyötyä muutettaessa yksittäisen tietueen arvoa, kun vain muutettava tietue voidaan lukita koko taulun sijaan. [2]

Vanhan ohjelmiston tietokannassa ei käytetty lainkaan indeksointia. Uudessa tietokannassa jokaiselle taululle on määritelty useampia indeksejä pääavaimen lisäksi. Indeksien käytöllä on oleellinen merkitys tietokantahakujen nopeuteen. Esimerkiksi voidaan ottaa julkaisutietojen haku vuosiluvun perusteella. Ilman indeksien käyttöä taulun jokainen tietue on luettava ja verrattava tallennettua vuosilukua haettavaan arvoon. Indeksiä käytettäessä haku kohdistuu tyypillisesti B-puu puurakenteeseen perustuvaan indeksiin, josta on osoittimet varsinaiseen tietueeseen. B-puusta vuosilukuja haettaessa haettavan vuosilukujoukon tietueet löytyvät nopeasti.

Alkuperäinen tietokanta oli taulurakenteeltaan suunniteltu hyvin sopivaksi vaaditun tiedon tallentamiseen eikä taulurakenteisiin vaadittu mitään merkittäviä muutoksia. Tietokanta-kyselyt olivat yksinkertaisia *select – from – where* -lauseita ja yhden toiminnon suorittamisessa niitä saatettiin tehdä useita. Koska käytössä oleva MySQL on relaatiotietokanta, niin ohjelmiston uudessa versiossa relaatioita myös käytetään. Tietokantakyselyissä tietoa haetaan useista tauluista kerralla hyödyntäen näin relaatio-ominaisuutta. Samalla ohjelmiston toimintoja saadaan tehostettua joissain tapauksissa huomattavasti, kun tarvittava tulosjoukko tietoa saadaan yhdellä kyselyllä sen sijaan, että tehtäisiin useampi erillinen kysely ja tietoja yhdisteltäisiin käyttäen PHP:tä.

4.6. Tietoturva

Julkaisurekisteriohjelmiston saatavuuteen ei ollut tarvetta tehdä muutoksia. Julkaisutietojen tallentamiseen tarkoitettu julkinen sekä ylläpito-osion käyttöliittymä käyttää jo aiemmin toteutettua WWW-palvelinohjelmiston tarjoamaa hypertext access -kirjautumista. Julkiseen osioon pääsee yliopiston yleistunnuksilla ja ylläpito-osioon on tunnukset vain julkaisurekisterin ylläpitäjillä. Tällöin ei ole tarvetta suojata yksittäisiä tiedostoja erikseen, kun WWW-palvelin hoitaa suojauksen jo matalammalla tasolla. Niin kutsuttuja SQL-injektioita ja Cross site scripting (XSS) hyökkäystä vastaan suojauduttiin seuraavalla ratkaisulla. Data välittyy aina käyttäjältä ohjelmistolle GET tai POST tyyppisinä HTTP-pyyntöinä. Data luetaan ohjelmiston käyttöön GET- ja POST-muuttujista. Ennen kuin GET- tai POST-muuttujan välityksellä saatu data tallennetaan tietokantaan, sille suoritetaan toimenpiteet, jotka muuntavat kaikki erikoismerkit HTML entiteeteiksi ja poistavat erikoismerkit. Jos järjestelmään yrittää tallentaa esimerkiksi JavaScript-ohjelmakoodia tai SQL-lauseen, järjestelmä käsittelee sekä näyttää sen vain tavallisena merkkijonona eikä yritä suorittaa siinä olevia komentoja. SQL-injektiolla tarkoitetaan menetelmää, jossa Web-lomakkeen kautta syötetyllä tietyllä tavalla muotoillulla SQL-lausekkeella aiheutetaan järjestelmässä tilanne, jossa lausekkeen komennot suoritetaan tietoa tuhoten tai käyttäjälle luottamuksellista tietoa palauttaen. XSS-hyökkäyksellä tarkoitetaan tilannetta, jossa käyttäjä pystyy tallentamaan esimerkiksi Web-lomakkeen kautta järjestelmään omaa ohjelmakoodia siten, että järjestelmä tulkitsee merkkijonot ohjelmakoodiksi ja suorittaa sen. Tällä tavoin hyökkääjä saa pahimmillaan julkisen Web-sivun käyttäytymään haluamallaan tavalla. Sivulle voi esimerkiksi luoda huomaamattoman näkymän haittaohjelmia tyrkyttävälle ulkopuoliselle sivustolle.

5. TEKNINEN TOTEUTUS

Teknisessä toteutuksessa peruspilari oli yksinkertaisuus ja suoraviivaisuus. Ohjelmiston suunnittelu on tehty käyttäen oliopohjaista ajattelutapaa ja ohjelmisto rakentuu ohjelmistoarkkitehtuurista, jossa ohjelmisto on jaettu eri toiminnot toteuttaviin komponentteihin rajapintoineen. Toteutukseen on sovellettu ohjelmistokehityksen iteratiivista mallia ja ketteriä menetelmiä. Toteutus tuli koostumaan vaatimusmäärittelyn pohjalta tehdystä koko järjestelmän kattavasta järjestelmä- ja ohjelmisto-arkkitehtuurisuunnittelusta sekä tämän jälkeen useista iteraatioista, joissa määriteltiin, suunniteltiin, toteutettiin ja testattiin ensin tärkeimmät ydintoiminnot ja tähän lisättiin useissa vaiheissa lisää ominaisuuksia ja toimintoja.

Vaatimusmäärittely tehtiin tarkastelemalla vanhassa ohjelmistossa olevia toiminnallisuuksia säilyttämällä ne myös uuden ohjelmiston vaatimuksiin. Yliopiston kirjaston henkilökunnan kanssa käytyjen palaverien pohjalta vaatimusmäärittelyä täydennettiin vastaamaan haluttua lopputulosta. Vaatimusmäärittelyn pohjalta aloitettiin koko järjestelmän kattava ohjelmistoarkkitehtuurin ja siihen kuuluvien osakokonaisuuksien suunnittelu. Toteutuksen alkuvaiheeseen kuului teknisten määrittelyjen tekeminen ja tietokantasuunnittelu. Tietokantasuunnittelun jälkeen vuorossa oli tietokannan, tietokantaluokkien ja yleisen rakenteen toteutus. Perustan valmistuttua siirryttiin toteuttamaan julkisen käyttöliittymän tietojen tallennuslomakkeita ja julkaisuhakua. Ylläpito-osiosta suunniteltiin ja toteutettiin ensimmäisenä oleellisin julkaisutietojen listaus ja tietojen tarkastelu. Tästä jatkettiin kaikkien perustoiminnallisuuksien toteuttamiseen. Jokaiselle osakokonaisuudelle tehtiin useita iteraatioita, joissa suunniteltiin, toteutettiin ja testattiin täydentäviä lisätoimintoja. Toteutusprosessin aikana pidettiin palavereja, joissa katselmoitiin jo tehtyä, pohdittiin ilmenneitä ongelmakohtia sekä suunniteltiin käyttöliittymää koskevien yksityiskohtien toteutusta. Projektin lähestyessä loppua, vanhan ohjelmiston tietokannan tiedot siirrettiin uuteen tietokantaan sekä suoritettiin vielä koko ohjelmiston kattava testaus. Vanhan ohjelmiston tietokannan tiedot muunnettiin ohjelmallisesti uuden tietokannan käyttämään muotoon ennen siirtoa. Vanha tietokanta sisälsi noin 3500 julkaisun tiedot 2700 eri henkilöltä. Julkaisuista noin 1500 oli luokiteltu konferenssijulkaisuiksi, 900 muihin julkaisuihin ja 550 referoituihin tiedelehtiin loppujen

jakautuen tasaisesti muihin luokitteluihin. Testauksessa ilmenneiden virheiden korjauksien jälkeen ohjelmisto toimitettiin yliopiston WWW-palvelimelle, jossa ohjelmiston toiminta testattiin uudelleen. Projektin toteutukseen kuluvaksi työaika-arvioksi arvioitiin määrittelyjen jälkeen noin 170 tuntia, mutta projektin edetessä nousi esille muutostarpeita ja ongelmakohtia, joiden seurauksena tuntikirjanpitoon kirjautui lopulta 209 tuntia.

5.1. Käyttöliittymä

Julkaisurekisteriohjelmiston toiminnallisuudet ja WWW-pohjainen käyttöliittymä jakautuvat kahteen osakokonaisuuteen: julkiseen julkaisutietojen lisäys-osioon ja vain ylläpitäjille tarkoitettuun hallintaosioon. Näistä ylläpito-osio kattaa arviolta noin 90 prosenttia koko ohjelmistosta. Käyttöliittymäelementtien sijoittelun perusrakenne on jaettu kolmeen pääalueeseen kuvan 3 mukaisesti.

Kuva 3: Käyttöliittymän perusrakenne

Sivun ylälaidassa on vain otsikko ja grafiikkaa. Vasemmassa laidassa on ylläpito-osiossa päävalikko. Keskellä sivua on varsinainen sivukohtainen sisältö ulottuen sivun alalaitaan saakka. Teknisesti sivupohjia käyttäen tämä on jaettu myös kolmeen osakokonaisuuteen seuraavasti: Ylälaidan rakenteessa ladataan tyylimäärittelyt, JavaScript-kehys, ulkoasurakenteen perusasettelu ja valikko. Alalaidan rakenteeseen kuuluu vain HTML-merkkauskielen ulkoasurakenteen lopetusmäärittelyt. Ylä- ja alalaidan väliin ladataan varsinainen sivukohtainen sivupohja tietoineen. Kaikki globaalit määrittelyt ovat määritelty yhteen kertaan ja näin esimerkiksi valikkoa muutettaessa muutos kohdistuu vain yhteen tiedostoon eikä kaikkien sivunäkymien sivupohjiin.

Käyttöliittymän ja käytettävyyden suunnittelussa on käytetty Steve Krugin käytettävyyslakia: ”Älä pakota minua ajattelemaan”. Web-sivuston tulisi olla mahdollisimman päivänselvä. Tavoitteena oli esittää käyttöliittymän elementit mahdollisimman selkeinä sekä näyttäen vain lyhyesti ja ytimekkäästi olennaisin. Ylimääräisistä hämmennystä aiheuttavista elementeistä ja toiminnallisuuksista pyrittiin eroon. Virheelliset syötteet pyrittiin korjaamaan mahdollisimman hyvin automaattisesti ja tarvittaessa käyttäjälle annettiin selkeä ilmoitus siitä, mikä syötteessä on vikana. [4]

5.2. Tavalliselle käyttäjälle näkyvät toiminnallisuudet

5.2.1. Valikko ohjeistukseen

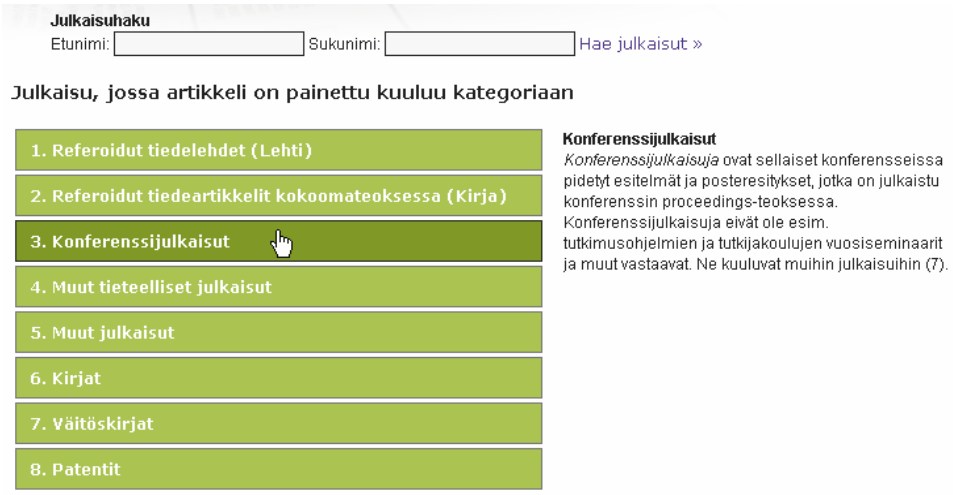
Sivun ylälaidassa sijaitsee linkit yliopiston julkaisuihin, luokitteluohjeeseen, yleisiin julkaisutietoja koskeviin ohjeisiin sekä linkki siirtymiseen suomenkielisen ja englanninkielisen version välillä.

5.2.2. Julkaisuhaku henkilön nimen perusteella

Hakutoiminto etsii annetun etu- ja sukunimen perusteella haettavan henkilön kaikki julkaisut tietokannasta. Tulokset ovat järjestettävissä julkaisun otsikon tai lisäys-päivämäärän mukaiseen järjestykseen. Haun tulokset on myös mahdollisuus tulostaa tai tallentaa tiedostoksi. Hakutoiminto on näkyvillä jokaisessa normaali käyttäjälle näkyvässä näkymässä. Hakutulokset avautuvat aina uuteen pieneen ikkunaan, jolloin käyttäjä voi käyttää hakua esimerkiksi kesken tietojen kirjoittamisen, ilman että sivusiirtymän seurauksena avoinna olevaa näkymää menetettäisiin.

5.2.3. Valikko lomakkeen valinnalle

Etusivun pääsisältöalueella sijaitsee luokituksen valinta kahdeksasta eri vaihtoehdosta. Navigoitavuutta on helpotettu jättämällä vanhat pienet valintapainikkeet pois ja käyttämällä kuvan 4 mukaisia pitkiä luokituksen nimen sisältäviä valintapainikkeita. Luokitustiedot haluttiin saada helposti nähtäville, jolloin hyväksi ratkaisuksi muodostuu luokitustietojen näyttäminen valintapainikkeiden oikealla puolella silloin, kun hiiren kursori on valintapainikkeen päällä. Valintapainiketta klikkaamalla käyttäjälle näytetään kyseisen luokituksen mukainen lomake tietojen kirjoittamista varten.



Kuva 4: Luokituksen valinta

5.2.4. Lomakkeet

Julkaisutiedot ovat luokiteltu kahdeksaan eri kategoriaan, joista jokaiselle on oma lomakenäkymä. Tyypilliset lomakkeelle kirjoitettavat tiedot ovat:

- Tekijöiden nimet ja organisaatiot, joihin he kuuluvat.
- Julkaisun nimi.
- Julkaisun julkaisuvuosi.
- Sivunumerot, joissa julkaisua käsitellään.
- ISBN ja ISSN.
- Mahdolliset huomautukset.
- Onko julkaisu ilmestynyt suomessa vai ulkomailla ja onko kyseessä kotimainen vai kansainvälinen julkaisu.

Lisäksi lomakkeelle täytetään luokitukselle ominaiset tiedot, joita ovat mm. konferenssin nimi, lehden nimi, väitöskirjan nimi, sivumäärä, volyymi ja numero. Lomakkeelle on mahdollista lisätä ja poistaa tekijöiden tietojen kirjoittamiseen varattuja rivejä täysin dynaamisesti. Lomakkeen tietojen lähettämisen jälkeen käyttäjälle näytetään vielä vahvistusnäkymä, jossa annetut tiedot ovat selkeästi otsikoidussa muodossa sekä

julkaisuluettelon käyttämässä muodossa käyttäjän tarkistettavana. Tietojen tarkistuksen jälkeen voidaan palata takaisin korjaamaan tietoja tai tallentaa tiedot tietokantaan. Tietojen tallentamisen jälkeen käyttäjälle näytetään jälleen etusivu, johon on GET-muuttujana välitetty tallennetun julkaisun tunniste. Mikäli käyttäjä siirtyy lisäämään uuden julkaisun tietoja, on edelliseen julkaisuun lisätyt tekijöiden tiedot jo valmiiksi täytetty lomakkeelle.

5.2.5. Kieliversiot

Uusi julkaisurekisteriohjelmisto on julkiselta käyttöliittymältään vanhan tavoin suomen- ja englanninkielille käännetty. Alkusuunnitelmissa oli tarkoitus toteuttaa kieliversio käyttäen samaa sivupohjaa molemmille kieliversioille niiden helpon muokattavuuden ja jatkokehityksen takia, mutta ajan puutteen vuoksi englanninkieliset versiot sivuista toteutettiin vain suomenkielisistä tehdyillä ja englanninkielelle käännetyillä kopioilla. Yhteistä sivupohjaa käytettäessä sivupohjat sisältäisivät ns. tageja, jotka korvattaisiin varsinaisella suomen- tai englanninkielisellä tekstillä ennen sivun näyttämistä. Kaikki suomenkieliset tekstit olisivat yhdessä kielitiedostossa ja englanninkieliset omassaan. Näin kaikki käyttöliittymään tehtävät muutokset voitaisiin tehdä yksiin sivupohjiin, jolloin muutokset päivittyisivät samalla kertaa kaikkiin eri kieliversioihin. Toisin sanoen, tämä menetelmä mahdollistaisi tarvittaessa hyvin helpon usean eri kieliversion toteuttamisen.

5.3. Ylläpitäjälle näkyvät toiminnallisuudet

5.3.1. Ylläpitäjälle näkyvä valikko

Ylläpitäjälle näkyvässä valikossa ovat seuraavat kohdat:

- Saapuneet julkaisut – lista julkaisuista, joiden tila on 1-3.
- Julkaisut – lista julkaisuista, joiden tila on 4.
- Luettelot – lomake vuosiluetteloiden luomiseen ja valikko niiden selaamiseen.
- Tilastot – lomake vuosittaisten tilastojen luomiseen
- Henkilön muokkaus – henkilöiden yhdistäminen ja nimen muuttaminen.
- Sarjojen muokkaus – osastojen sarjojen lisäys ja muuttamien.
- Osastolistan muokkaus – osastolistan muuttamien.
- Julkaisurekisterin dokumentit – linkit kaikkiin ohjelmiston dokumentteihin.

- Syöttölomakkeiden lukinta
- Virheilmoitukset – lista palvelimella tapahtuneista virheistä.
- Tietokanta – tietokannan taulujen kaikki tiedot taulukkomuodossa.
- Vanha kanta – vanhan tietokannan tiedot taulukkomuodossa.

5.3.2. Saapuneiden julkaisujen listaus

Saapuneet julkaisut on ylläpito-osion oletusnäkymä. Saapuneissa julkaisuissa näytetään kaikki julkaisut, joiden tilana on *vastaanotettu*, *tarkastettu* tai *valmis*. Haun tulokset näytetään kuvan 5 mukaisena listana. Jokaisesta julkaisusta näkyy ensimmäisessä sarakkeessa kyseiselle listalle ominainen julkaisun järjestysnumero. Toisessa sarakkeessa on julkaisun nimi ja kolmannessa sarakkeessa on päivämäärä, jolloin jotain julkaisun tietoa on viimeksi muutettu. Neljännessä sarakkeessa on näkyvissä julkaisun tila.

Kuva 5: Saapuneiden julkaisutietojen näyttäminen

Tulokset voidaan lajitella nimen, päivityspäivämäärän tai tilan mukaan nousevaan tai laskevaan järjestykseen. Julkaisurekisteriin siirrettyjen julkaisujen näkymä on vastaavanlainen, mutta siinä näytetään vain julkaisut, joiden tilaksi on tallennettu 4 sekä lajittelu voidaan suorittaa myös vuosiluvun mukaan. Saapuneiden julkaisujen ja julkaisurekisteriin siirrettyjen julkaisujen näkymissä sivun ylälaidassa on julkaisuhaku, jolla voidaan hakea tietyn henkilön tai osaston julkaisut. Julkaisujen tietojen näyttäminen ja tilan muuttamien on toteutettu Ajaxia hyödyntäen. Julkaisun tietoja voidaan tarkastella helposti klikkaamalla julkaisun nimeä kuvan 5 näkymässä, jolloin tiedot näytetään julkaisulistan päälle avautuvassa pienessä kehyksessä. Julkaisun nimeä klikattaessa selainohjelma lähettää Ajaxia käyttäen valitun julkaisun tunnisteen HTTP-pyyntönä palvelimelle. Palvelin hakee tunnistetta käyttäen oikean julkaisun tiedot tietokannasta, muodostaa niistä sivupohjamoottoria sekä oikeaa sivupohjaa käyttäen näkymän ja palauttaa lopputuloksen takaisin selainohjelmalle. Selainohjelma vastaanottaa palvelimen vastauksen ja näyttää sen julkaisulistan päälle avatussa kehyksessä. Kehyksessä olevasta linkistä voidaan siirtyä lomakkeelle, jonka kautta julkaisun tietoja voidaan muuttaa. Julkaisua poistettaessa käyttäjältä pyydetään vahvistus toimenpiteen suorittamiselle. Avautunutta kehystä ei tarvitse erikseen sulkea, vaan listasta voidaan valita suoraan toisen julkaisun tiedot näytettäväksi. Näin listan läpikäynti nopeutuu ja helpottuu huomattavasti, kun sivusiirtymiä ei tapahdu ja uuden sivun latausta ei tarvitse odotella. Julkaisun tilan muuttamien on myös Ajaxilla tehostettu. Käyttäjän klikatessa julkaisun tilaa ilmaisevaa numeroa, selainohjelma lähettää jälleen julkaisun tunnisteen ja valitun uuden tilan tiedot HTTP-pyyntönä palvelimelle. Palvelin päivittää tilan muutoksen tietokantaan ja palauttaa selaimelle JavaScript-komennon. Selain vastaanottaa ja suorittaa komennon, joka vaihtaa uutta tilaa ilmaisevan numeron värilliseksi. Värin vaihtuminen ilmaisee näin tietojen tallennuksen onnistumista. Listan lopusta löytyy lisäksi toiminto, jolla kaikkien listassa olevien julkaisujen tila voidaan vaihtaa kerralla halutuksi.

5.3.3. Vuosittaisten julkaisuluetteloiden tuottaminen

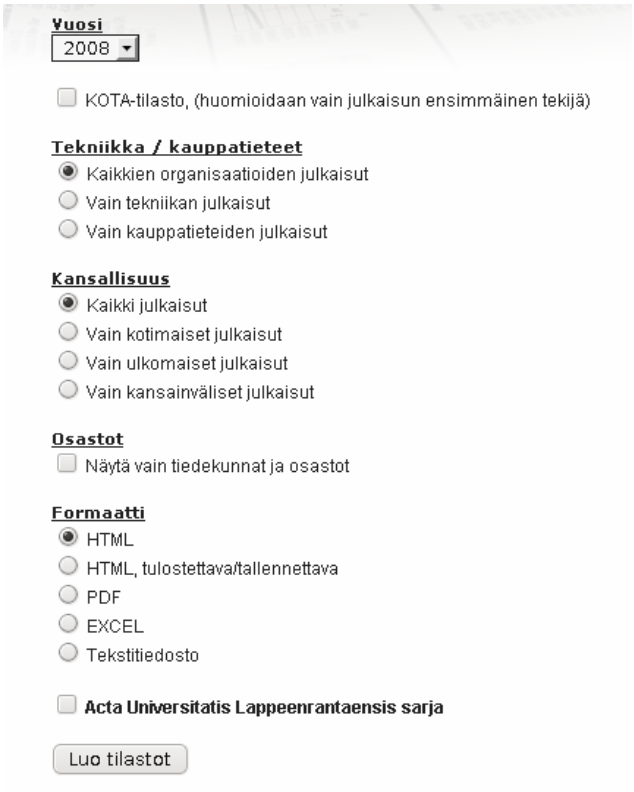
Vuosittaisten julkaisuluetteloiden tuottaminen tapahtuu kuvan 6 mukaisella lomakkeella. Luettelot voidaan tuottaa kaikkien tai vain tietyn osaston julkaisuista. Lomakkeelta valitaan lisäksi minkä vuoden julkaisut, missä tilassa olevat julkaisut ja mihin luokituksiin kuuluvat julkaisut otetaan mukaan. Julkaisuluetteloista tuotetaan yhdellä kerralla aina kolme eri HTML-muotoista luettelotiedostoa. Yhteen on listattu kaikki henkilöiden nimet,

jotka löytyvät julkaisuluettelosta. Toisena tiedostona on varsinainen julkaisuluettelo, jossa julkaisut ovat listattuna aakkosjärjestyksessä olevien henkilöiden nimien alle. Kolmas on kuten edellinenkin, mutta luettelossa on vain niiden henkilöiden nimet, jotka ovat ensimmäisenä tekijänä julkaisun tiedoissa. Julkaisuarkistossa nämä kolme tiedostoa ovat listattuna peräkkäin, siten että uusimmat ovat aina listan ylimpänä. Teknisessä ratkaisussa sivupohjamoottori on merkittävässä asemassa suorittaessaan luetteloon kymmeniä kertoja toistuvien rakenteiden monistamisen tietoineen mahdollistaen silti hyvin yksinkertaisen sovelluslogiikan käyttämisen.

Kuva 6: Julkaisuluetteloiden tuottaminen

5.3.4. Tilastot

Tilastojen muodostaminen oli työn vaikein osuus. Alun ongelmana oli ymmärtää asiakasta siinä, miten tilastojen kuuluisi todellisuudessa toimia. Useamman iteraatiokierroksen ja katselmoinnin jälkeen tilastointi saatiin kuitenkin toimimaan kuten oli tarkoituskin. Tilastojen tuottaminen tapahtuu valitsemalla halutut valinnat kuvan 7 mukaisella lomakkeella.



Kuva 7: Tilastojen tuottaminen

Tilastot näytetään taulukkomuodossa kuvan 8 mukaisesti. HTML-muotoisen taulukon lisäksi tilastot on mahdollista luoda Portable Document Format (PDF) ja Microsoft Excel-tiedostoina sekä tekstitiedostona, jossa arvot ovat puolipilkuilla erotettuina. PDF-dokumentin luomiseen käytetään samaa jo vanhassa ohjelmistossa ollutta kolmannen osapuolen komponenttia. Excel-tiedoston luomiseen käytetään valmista PEAR Spreadsheet Excel Writer –luokkaa.

Organisaatio	Ref. lehdet	Ref. kirjat	Konf.	Muut tieteel.	Muut	Kirjat	Väitösk.	Patentit
LUT METALLI	79	6	130	11	41	0	3	4
KEMIANTEKNIIKAN OSASTO	45	2	42	6	19	0	0	2
Kuitutekniikan laboratorio	0	0	1	0	0	0	0	0
Teknillisen kemia laboratorio	3	0	5	0	2	0	0	1
Epäorgaanisen ja analyyttisen kemian laboratorio	3	0	9	0	3	0	0	0
Orgaanisen kemian laboratorio	0	0	0	0	0	0	0	0
Fysikaalisen kemian laboratorio	2	0	0	0	2	0	0	0
Systeemitekniikan laboratorio	0	0	0	0	0	0	0	0
Erotustekniikan laboratorio	17	1	13	5	2	0	0	0

Kuva 8: Tilastonäkymä

5.3.5. Henkilöiden muokkaus

Henkilöiden muokkausnäkymässä on listattuna aakkosjärjestyksessä kaikki tietokannasta löytyvät henkilöt ja henkilöiden julkaisujen lukumäärät. Klikkaamalla listassa olevaa henkilön nimeä, päästään näkymään, jossa nimeä voidaan muuttaa tai jossa henkilö voidaan yhdistää tai liittää toiseen henkilöön. Henkilöiden yhdistämisellä tarkoitetaan tilannetta, jossa saman henkilön julkaisuja löytyy myös jollain virheellisesti kirjoitetulla nimellä. Tällöin voidaan virheellinen nimi yhdistää oikeaan henkilön nimeen, jolloin virheellisen nimen alle kuuluvat julkaisut siirtyvät oikean nimen alle ja virheellinen henkilön nimi poistetaan tietokannasta. Henkilöiden liittämisellä tarkoitetaan tilannetta, jossa henkilön nimi on muuttunut esim. avioliiton myötä ja uusi nimi halutaan liittää vanhaan, siten että kummallekin nimelle löytyy omat julkaisunsa ja julkaisuluettelossa kyseisten nimien perässä näkyy sulkeissa myös tämä toinen nimi.

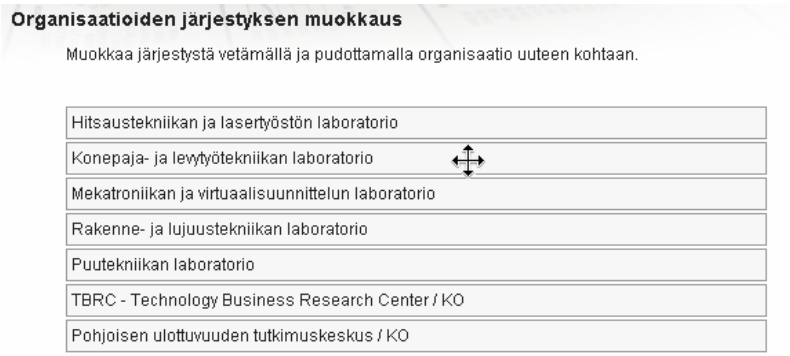
5.3.6. Sarjojen muokkaus

Sarjojen muokkaus on toteutettu hyvin yksinkertaisesti käyttäen kahta monirivistä HTML-lomakkeen kenttää. Toisessa kentässä sarjojen nimet ovat suomenkielellä ja toisessa englanninkielellä. Jokainen sarja on kirjoitettu omalle rivilleen. Ratkaisu on kuten yksinkertainen tekstin muokkaamiseen tarkoitettu ohjelma, jossa rivien tietoja sekä järjestystä on helppo muuttaa.

5.3.7. Osastolistan muokkaus

Osastolistan muokkausnäkymässä kaikki organisaatiot ovat listattuna yhtenä suurena taulukkona, jossa on ilmoitettu osaston tunniste, hierarkiatason tunniste, suomen- ja englanninkieliset nimet sekä onko osasto näkyvissä julkaisutietojen tallentamiseen tarkoitetuissa lomakkeissa. Listan lopussa on lomake, jolla voidaan lisätä uusia organisaatioita yllä mainitut tiedot syöttämällä. Organisaation tunnistetta klikkaamalla avataan lomake, jolla listassa olevia tietoja voidaan muuttaa. Vanhassa ohjelmistossa organisaatioiden järjestyksen muuttamien tapahtui yksi kerrallaan organisaation tietoja muuttamalla. Ohjelmiston uuteen versioon toteutettiin Ajaxia ja JavaScriptiä hyödyntämällä toiminto, jolla järjestystä muutetaan vain raahaamalla haluttu organisaatio uuteen kohtaan listassa hiiren painike alas painettuna kuvan 9 mukaisessa näkymässä. Hiiren painikkeen vapauduttua selainohjelma lähettää palvelimelle HTTP-pyyntönä listan

organisaatioiden tunnisteista uudessa järjestyksessä. Palvelin luo automaattisesti näille organisaatioille uudet hierarkiatunnisteet ja päivittää ne tietokantaan vanhojen tilalle. Palvelin lähettää selaimelle vastauksena tiedon toimenpiteen onnistumisesta, jonka selain näyttää organisaatiolistan yläpuolella käyttäjälle.



Kuva 9: Organisaatioiden järjestyksen muuttaminen

5.3.8. Syöttölomakkeiden lukinta

Syöttölomakkeiden lukitsemisnäkymässä julkaisutietojen lisäämiseen tarkoitetut lomakkeet voidaan ottaa pois käytöstä esimerkiksi vuoden alussa tapahtuvan julkaisuluetteloiden ja tilastojen tuottamisen ajaksi. Lomakkeiden lukitseminen on toteutettu kirjoittamalla palvelimella olevaan tekstitiedostoon tieto lomakkeiden tilasta. Julkaisurekisteriohjelmiston julkisen etusivun avauksen yhteydessä tämä tieto tarkistetaan tiedostosta ja käyttäjälle näytettävä näkymä määräytyy sen mukaisesti.

5.4. Testaus

Ohjelmistokehityksen iteratiivisen ja ketterien menetelmien mukaisesti ohjelmiston testaus oli jatkuvasti toistuvaa. Kokonaisuudessaan testaus jaettiin kolmeen osakokonaisuuteen. Kehitysvaiheessa jokainen toteutettu toiminto testattiin käytetyn menetelmän mukaisesti ja havaitut virheet korjattiin. Toimintojen muodostaessa jonkin toiminnallisuus-kokonaisuuden kuten saapuneiden julkaisujen listaus tai HTML-muotoisen tilaston luominen, kokonaisuus testattiin käyttäen yleisimpiä syötteitä. Projektin lopussa suoritettiin kolmas ja viimeinen testaus, joka kattoi koko järjestelmän testaamisen eri WWW-selainohjelmilla ja käyttäen jälleen ohjelmalle tyypillisiä syötteitä. Pienten osakokonaisuuksien testaus on nopeaa ja helppoa, koska PHP on tulkattava kieli eikä

ohjelmakoodin kääntämistä suoriteta ja Web-sivun uudelleen latauksella voidaan helposti toistaa testattava toimenpide. Kuten WWW-pohjaisen ohjelmiston toteutukseen kuuluu, sen tulee toimia vähintäänkin kaikilla yleisimmin käytössä olevilla WWW-selainohjelmilla. Tähän projektiin määriteltiin yleisimmin käytetyiksi WWW-selaimiksi Mozilla Firefox 2, Microsoft Internet Explorer 6, Opera 8,5 ja Safari 3 sekä näiden uudemmat versiot. Käyttöliittymän näkyminen oikein ja toiminnallisuuksien toimiminen testattiin näillä selaimilla sekä niiden sen hetken uusimmilla saatavilla olevilla versioilla. Koska toteutuksen tuli toimia myös hyvin vanhalla Internet Explorer 6 versiolla, uusimpien asiakaspuolen tekniikoiden käyttö jätettiin pois. Ulkoasulliset ratkaisut toteutettiin käyttäen vanhoja tapoja yhteensopivuus-ongelmien välttämiseksi. Kokonaisuudessa kaikki toimi eri WWW-selaimissa kuten oli tarkoitettukin eikä suurempiin ongelmiin törmätty. Testatessa ohjelmistoa lopuksi yliopiston WWW-palvelimella, havaittiin muutamia helposti korjattavia ongelmia, jotka johtuivat palvelimen käyttämästä vanhemmasta PHP skriptauskielen versiosta. Uudemman version käyttäminen kehityspalvelimella oli kuitenkin tietoinen valinta ja vaikka siitä aiheutuvat rajoitteet olivat hyvin tiedossa, ohjelmiston toteutukseen tuli aluksi muutama toimimaton ratkaisu. [3]

5.5. Dokumentointi

Julkaisurekisteriohjelmistosta tuotettiin sopimusten mukaisesti tekninen määrittely, sekä ohjeistukset ylläpitäjälle ja jatkokehittäjälle. Tekninen määrittely pitää sisällään tiedot käytetystä ympäristöstä ja arkkitehtuurista, tietokannan taulukuvaukset, kuvauksen WWW-sivujen rakenteesta, navigoitavuudesta ja käyttöliittymästä sekä kuvauksen ohjelmiston toiminnallisuuksien toteutuksesta. Ylläpitäjän ohjeistus on ylläpitäjille tarkoitettu julkaisurekisteriohjelmiston käyttöohje. Käyttöohjeessa on kuvattu tallennettujen julkaisujen hallinta, luetteloiden ja tilastojen muodostaminen, henkilötietojen, sarjojen ja osastolistan muuttaminen. Jatkokehittäjälle suunnattu ohjeistus pitää sisällään tietoa jatkokehittäjälle ja toimii lisäyksenä tekniselle määrittelylle. Ohjeistus sisältää mm. kuvauksen järjestelmän tiedostoista ja niiden merkityksestä. Lisäksi ohjelmiston luokkien PHP-ohjelmakoodista tuotettiin kattava dokumentaatio käyttäen phpDocumentor-ohjelmaa. Dokumentaatioon listattiin kaikki luokkamuuttujat, metodien nimet, kuvaukset, parametrit ja niiden paluuarvot. Kaikki dokumentaatio toimitettiin sopimusten mukaisesti yliopiston kirjastolle sähköisessä ja painetussa muodossa.

6. TOTEUTUNEEN RATKAISUN MERKITYS

Käytännön näkökulmasta katsottuna toteutunut ratkaisu täyttää kaikki sille asetetu vaatimukset. Osa ratkaisuista toteutettiin jopa hieman käyttäjäystävällisemmin, mitä asiakas oli toivonut. Samalla uudistamalla ohjelmiston ohjelmistoarkkitehtuurin, luotiin hyvä perusta ohjelmiston tulevaisuutta ajatellen. Vanhaa ohjelmistoa jatkokehitettäessä lopputuloksesta olisi luultavasti tullut riskialtis ja erilaisia kompromissejä sisältävä ratkaisu. Kokonaan uuden toteutus tuli olemaan työläs, mutta vanhan jatkokehitys olisi myös vienyt paljon aikaa, eikä sitä kautta olisi päästy tähän toteutuneeseen lopputulokseen.

Toteutuksen teknisestä näkökulmasta merkittävin muutos tapahtui sivupohjamoottorin käytöstä. Konkreettisena esimerkkinä liitteessä 1 on otos vanhan ohjelmiston tyypillistä sovelluslogiikan ohjelmakoodia. Liitteessä 2 on otokset uuden ohjelmiston tyypillisestä sovelluslogiikasta sekä sivupohjasta. Liitteissä olevaa ohjelmakoodia katsomalla on havaittavissa sivupohjamoottorin käyttämisen tuoma selkeys ohjelmakoodiin. Ohjelmiston kehitys nopeutui myös huomattavasti sivupohjia käytettäessä mahdollistaen mm. ohjelmakoodin osien helpon kopioimisen muihin ohjelmalohkoihin. Suorituskyvyssä saavutettiin tilastojen osalta valtava muutos. Vanhassa ohjelmistossa tilastojen luominen kesti useamman minuutin, kun ohjelmiston uudessa versiossa tilastotiedot avautuvat käyttäjälle käytännössä välittömästi. Tulokseen vaikutti eniten käytettävien tietokanta-kyselyiden parempi suunnittelu.

Vanhan ohjelmiston toimintaa analysoitaessa huomiota herätti hieman käytetty ohjelmakoodin suuri määrä. Mielenkiinnon ja vertailun vuoksi ohjelmakoodin määrät laskettiin vertailua varten. Kuvassa 10 on listattuna koneellisen laskennan tulokset vanhan ja uuden toteutuksen ohjelmakoodin määrästä. Ratkaisujen väliset erot ohjelmakoodin määrän suhteen ovat yllättävän suuria. Uuden ohjelmiston kohdalla sovelluslogiikan, sivupohjien, PEAR:n luokkien sekä Ajax- ja JavaScript-kehysten sisältämä ohjelmakoodi on laskettu vielä erikseen havainnollistamaan näiden keskinäistä suhdetta. Tulosten suurta eroa selittää osittain vanhassa ohjelmassa olevat paljon toistuvat osiot, kun taas uudessa oli pyritty mahdollisimman vähään määrään, mutta monikäyttöisiin osioihin. Myös

sivupohjamoottoria käytettäessä samaa sovelluslogiikan komponenttia voitiin käyttää tuottamaan useampia näkymiä käyttäen vain eri sivupohjia.

Kuva 10: Ohjelmakoodin määrän vertailu

6.1. Tulevaisuus

Yliopiston kirjaston palavereissa on ollut jo hieman puhetta julkaisutietojen tallennusta koskevista tulevaisuuden suunnitelmista, jotka enteilevät muutoksia nykyiseen julkaisurekisteriohjelmistoon. Tulevista suunnitelmista ei kuitenkaan ole syytä huolestua. Mitään radikaaleja muutoksia ei ole näillä näkymin tulossa ainakaan lähitulevaisuudessa. Julkaisurekisteriohjelmiston uusitun version ohjelmistoarkkitehtuuri mahdollistaa jatkokehityksessä tehtävien muutosten helpon toteuttamisen ohjelmistoon. Mahdollisten muutosten tulemiseen oli varauduttu jo ohjelmistoa suunniteltaessa, jolloin tehdyissä ratkaisuissa on pyritty käyttämään myös ohjelmistoon kohdistuvia muutoksia suosivia ratkaisuja. Lisäksi kokonaisuus muodostuu useasta osakokonaisuudesta parantaen muutosten toteuttamisen hallittavuutta ja kokonaiskuvan havaitsemista. Nykyinen ratkaisu ei kuitenkaan vähennä jatkokehityksessä käytettävän suunnittelun ja määrittelyn tärkeyttä.

7. JOHTOPÄÄTÖKSET

Työn tuloksena valmistui uudistettu julkaisurekisteriohjelmisto, joka palvelee hyvin nykytilanteen tarpeita sekä luo hyvän perustan mahdolliselle jatkokehitykselle. Ohjelmiston uusi suunnittelu ja toteutus osoittautui hyväksi ratkaisuksi ja tavoitteet saavutettiin helposti. Uusi organisaatiohierarkia tiedekuntineen ja julkaisutietojen luokituksen muutokset saatiin toteutettua halutulla tavalla. Aiemmin vanhassa ohjelmistossa ilmenneet ongelmakohdat ratkaistiin ja etenkin ylläpitäjille näkyvään käyttöliittymään tehtiin ylläpitoa merkittävästi helpottavia parannuksia.

Ohjelmistoon suunniteltiin ja toteutettiin uusi ohjelmistoarkkitehtuuri, jossa ohjelmisto on jaettu eri toiminnot toteuttaviin komponentteihin rajapintoineen. Ohjelmistoarkkitehtuuri muodostaa selkeän, skaalautuvan ja helposti jatkokehitettävissä olevan ratkaisun. Sivupohjien käytöllä esityslogiikka erotettiin täysin sovelluslogiikasta. Tietokannan rakennetta optimoitiin ja tietokantakyselyitä käytettiin tehokkaammin hyödyksi. Uusi ohjelmistoarkkitehtuuri yhdessä eri tekniikoita tehokkaasti hyödyntämällä johti tulokseen, jossa ohjelmiston ohjelmakoodin määrä on vain noin kolmasosa vanhan ohjelmiston sisältämästä määrästä.

Projekti oli työmäärällisesti 23% arvioitua suurempi ja toteutukseen suunniteltu aikataulu ylittyi kahdella viikolla lähinnä testauksessa ilmenneiden ongelmien korjauksen seurauksena. Testausvaiheessa havaittiin useita asioita, joita kannatti muuttaa hieman ja näiden muutosten tekemiseen kului aikaa, jota työaika-arviossa ei oltu otettu huomioon projektia aloitettaessa. Projektin toteutuksessa suurimmat haasteet olivat tilastojen ja julkaisuluetteloiden teknisen toteutuksen suunnittelu. Tilastojen osalta törmättiin aluksi myös laskutoimituksille tyypillisiin ongelmiin tilastojen laskiessa tuloksia väärin. Projektia toteutettiin viikonloppuisin sekä iltaisin työpäivän jälkeen, jolloin aikataulun ylittymiseen osaltaan vaikutti myös ongelmat projektin toteuttamiseen käytettävän ajan varaamisessa.

LÄHTEET

[1] Asleson Ryan, Schutta Nathaniel T., ”Ajax - tehokas hallinta”, 2007.

[2] Gilmore W. Jason, ”PHP 5 & MySQL - tehokas hallinta”, 2005.

[3] Haikala Ilkka, Märijärvi Jukka, ”Ohjelmistotuotanto”, 2004.

[4] Krug Steve, ”Älä pakota minua ajattelemaan!”, 2006.

[5] Prototype Core Team, ”Prototype JavaScript framework”, 2009. Saatavissa: http://www.prototypejs.org [viitattu 24.08.2009]

[6] Wikipedia, ”DOM – Document Object Model”, 2009. Saatavissa: http://en.wikipedia.org/wiki/Document_Object_Model [viitattu 23.08.2009]

[7] Wikipedia, ”JavaScript”, 2009. Saatavissa: http://en.wikipedia.org/wiki/JavaScript [viitattu 22.08.2009]

[8] Zandstra Matt, ”PHP Trainer Kit”, 2001.

LIITE 1: Ote vanhan ohjelmiston sovelluslogiikasta

```
$nakyma = $_GET["nakyma"];
if ($nakyma == '2') {
           $jarjesta_konf = $_GET["jarjesta_konf"];

           if ($jarjesta_konf == 'id') { $result = mysql_query("SELECT * FROM
konferenssi WHERE tila < 4 ORDER BY id ASC",$db); }
           else if ($jarjesta_konf == 'artikkelin_nimi') { $result =
mysql_query("SELECT * FROM konferenssi WHERE tila < 4 ORDER BY artikkelin_nimi
ASC",$db); }
           else if ($jarjesta_konf == 'paivitetty') { $result =
mysql_query("SELECT * FROM konferenssi WHERE tila < 4 ORDER BY paivitetty
ASC",$db); }
           else if ($jarjesta_konf == 'tila') { $result = mysql_query("SELECT *
FROM konferenssi WHERE tila < 4 ORDER BY tila ASC",$db); }
           else { $result = mysql_query("SELECT * FROM konferenssi WHERE tila <
4",$db); }
           $i=0;
           while ($myrow = mysql_fetch_array($result)) {
             $i = $i+1;
      printf("<tr>  <td><font size=\"-1\"><a
href=\"db_yllapito_yksilonakyma.php?konferenssi=%s&nakyma=2\">%s</a></font></td
>  <td><a
href=\"db_yllapito_yksilonakyma.php?konferenssi=%s&nakyma=2\">%s</a></td>
<td><font size=\"-1\">%s</font></td>",
        $i, $i, $myrow["id"], $myrow["artikkelin_nimi"], $myrow["paivitetty"]);
                     $id = $myrow["id"];

                     if($myrow["tila"] == 1) { printf("<td><font size=\"-
1\">vastaanotettu</font></td>"); }
                     else if($myrow["tila"] == 2) { printf("<td><font size=\"-
1\">tarkastettu</font></td>"); }
                     else if($myrow["tila"] == 3) { printf("<td><font size=\"-
1\">valmis</font></td>"); }
                     else if($myrow["tila"] == 4) { printf("<td><font size=\"-
1\">julkaisurekisteriin</font></td>"); }

           echo '<td><font size="-1">';
               echo '<select name="uusi_tila';
               echo $id;
               echo '">
          <option>valitse</option>
          <option>vastaanotettu</option>
          <option>tarkastettu</option>
          <option>valmis</option>
          <option>julkaisurekisteriin</option>
          <option>poista</option>
        </select>
        </font>
              </td>'; // echo
    } // while
} // if($nakyma == '2')
```

LIITE 2: Ote uuden ohjelmiston sovelluslogiikasta ja sivupohjasta

Ote uuden ohjelmiston sovelluslogiikasta

```
if (!$publication = $publicationsObj->getPublicationByID($julkaisutyyppi,
$kategoriaID)) {
     echo "Julkaisua ei löydy!";
     exit();
}
$tpl = new
HTML_Template_IT($settings['wwwRoot'].'templates/'.$settings["tplPath"]);
$tpl->loadTemplatefile('viewLayer.tpl.htm', true, true);

if (isset($_GET["status"])) {
    if ($_GET["status"] == 1) $tpl->touchBlock("tiedotTallennettu");
    if ($_GET["status"] == 0) $tpl->touchBlock("error");
}

switch ($julkaisutyyppi) {
    case 1:
        $tpl->setCurrentBlock("konferenssijulkaisut");
        $tpl->setVariable('artikkelin_nimi', $publication->artikkelin_nimi);
        $tpl->setVariable('sivut', $publication->sivut);
        $tpl->setVariable('konferenssi', $publication->konferenssin_nimi);
        $tpl->setVariable('isbn', $publication->isbn);
        break;
    case 2:
        $tpl->setCurrentBlock("refKirjat");
        $tpl->setVariable('artikkelin_nimi', $publication->artikkelin_nimi);
        $tpl->setVariable('sivut', $publication->sivut);
        $tpl->setVariable('teoksen_nimi', $publication->teoksen_nimi);
        $tpl->setVariable('isbn', $publication->isbn);
        break;
```

Ote uuden ohjelmiston sivupohjasta

```
<div class="title3" style="margin-top:0px;">Tekijät</div>
<!-- BEGIN tekija -->
<input type="hidden" name="organisaatio[]" value="{organisaatioID}" />
<input type="hidden" name="etunimi[]" value="{etunimi}" />
<input type="hidden" name="sukunimi[]" value="{sukunimi}" />
{etunimi} {sukunimi}, {organisaatio}<br />
<!-- END tekija -->

<!-- BEGIN konferenssijulkaisut -->
<div class="title3">Artikkelin nimi</div>
{artikkelin_nimi}
<br /><br />
<span class="title3">Vuosi</span>
{vuosi}

<div class="title3">Sivut</div>
{sivut}

<div class="title3">Konferenssin nimi, aika ja paikka ym.</div>
{konferenssi}

<div class="title3">ISBN</div>
{isbn}
<!-- END konferenssijulkaisut -->
```