SAVONIA

OPINNÄYTETYÖ - **AMMATTIKORKEAKOULUTUTKINTO**
TEKNIIKAN JA LIIKENTEEN ALA

OPPIMAA-YMPÄRISTÖ

Opinnäytetyö

TEKIJÄ/T: Saku Kaarakainen

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Koulutusohjelma Tietotekniikan koulutusohjelma	
Työn tekijä Saku Kaarakainen	
Työn nimi Oppimaa-ympäristö	
Päiväys 16. joulukuuta 2014	Sivumäärä/Liitteet 37
Ohjaaja(t) Lehtori Jussi Koistinen, lehtori Sami Lahti	
Toimeksiantaja/Yhteistyökumppani(t) Weego Software Oy	

Tiivistelmä

Tämän opinnäytetyön aiheena oli jatkokehittää @Oppimaan nettisivuja. Tavoitteena oli lähes valmis ja toimiva järjestelmä tuotantoon. @Oppimaa on järjestelmä, jossa luodaan ja käytetään kirjoja sähköisessä muodossa. Sen on suunniteltu ensisijaisesti oppimis-ympäristöön.

Työtä tehtiin ketterän kehitysmallin mukaisesti ja siinä käytettiin PHP, PHPMailer, HTML, JavaScript, jQuery ja CSS –tekniikoita. Käyttöjärjestelminä oli Linux -käyttöjärjestelmät, joissa oli Apache-palvelin ja MySQL –tietokanta. Tietokantaan otettiin yhteys Zend Frameworkillä tehdyn rajapinnan kautta, jota kutsuttiin Zend-rajapinnaksi.

Työhön tehtiin määritelmä ennen opinnäytetyön aloittamista, jonka mukaan järjestelmään oli tarkoitus lisätä sisällönhallintaan, oppilaan ja opettajan työkaluihin uusia ominaisuuksia. Sisällönhallintaan lisättiin lopulta kirjan kopiointi, poistaminen, arkistointi ja äänen sekä kuvan lisäämisen triggerinä. Opettajan työkaluun lisättiin kirjan sivun lähetys PDF-formaatissa sähköpostiin ja oppilaan työkaluun äänten ja kuvien näyttämisen.

Työn spesifikaatio muuttui jatkuvasti kehityksen aikana, jonka takia siihen lisättiin alkuperäiseen määritelmään kuulumattomia toimintoja. Tälläisiä olivat esim. opettajan työkalulla kirjan sivun lähettäminen sähköpostiin PDF-formaatissa, kirjan kopiointi, poisto ja arkistointi.

Opinnäytetyön tuloksena syntyi nettisivut, jotka ovat Weegon julkisella tuotantopalvelimella heidän asiakkaiden käytettävissä. Työstä jäi vielä puuttumaan muutamia osia, jotka tullaan lisäämään jatkokehityksessä, mikäli sellainen tulee.

Avainsanat
PHP, JavaScript, Ajax, jQuery, Zend, Ketterä ohjelmistokehitys, sähköisten kirjojen hallinta

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Information Technology			
Author(s) Saku Kaarakainen			
Title of Thesis Oppimaa Environment			
Date	16. December 2014	Pages/Appendices	37
Supervisor(s) Mr. Jussi Koistinen, Lecturer, mr. Sami Lahti, Lecturer			
Client Organisation /Partners Weego Software Oy			

Abstract

The purpose of this thesis was to develop the Oppimaa environment further. The goal was to get an almost complete and functional environment to the production. Oppimaa is an electronic environment where books are created and used.

The project was made according to the agile development model and PHP, PHPMailer, HTML, JavaScript, jQuery and CSS techniques were used. The project was made on Linux based operating systems with Apache Server and MySQL database. The database was connected with the Zend interface that was made by Zend Framework.

The specification part of the project stated that new elements should be added in the content management, both in the teacher's tool and in the student's tool. Copying, deleting and archiving the book functions were added to content management. The function of sending a page of a book in the PDF format via email was added to the teacher's tool and to the ability to display an image and play a sound was added to the student's tool.

The specification of the project was very mutable during development which is why functions that are not found in the original specification were added. One example of this is sending a page of a book via email with the teacher's tool, copying, deleting and archiving the book.

The result of thesis was a web-site that exists on Weego's public production server, which their customers can use. Some parts are still missing and they will be added in a later development version in case there will be one.

Keywords
PHP, JavaScript, Ajax, jQuery, Zend, Agile, e-learning management

SISÄLTÖ

KÄYTETYT LYHENTEET 6
1 JOHDANTO 7
2 KÄYTETYT MENETELMÄT, TEKNIIKAT JA TYÖKALUT 8
2.1 Ketterä ohjelmistokehitys 8
2.2 HTML 9
2.3 CSS 9
2.4 JavaScript 9
2.4.1 jQuery 9
2.4.2 JSON 10
2.4.3 Ajax 10
2.5 PHP 12
2.6 PHPMailer 12
2.7 MVC 13
2.8 Zend Framework 13
2.9 MySQL 13
2.10 Apache Server 14
2.11 Linux 15
3 @OPPIMAA –JÄRJESTELMÄ 16
3.1 Zend-rajapinta 17
3.2 Sisällönhallinta –työkalu 20
3.2.1 Esimerkki uuden kirjan lisäys 21
3.2.2 Booklist 23
3.2.3 Page Settings 23
3.2.4 Page Triggers 24
3.2.5 Page Elements 25
3.2.6 Assets 26
3.3 Raportointi-työkalu 27
3.4 Oppilaan työkalu 28
4 TOTEUTUS 30
4.1 Oma osuus työstä opinnäytetyönä. 31
4.2 Tehty työ 31

4.2.1 Sisällönluonti 31
4.2.2 Opettajan työkalu 31
4.2.3 Oppilaan työkalu 32
4.2.4 Zend-rajapinta 32
4.2.5 Muut 32
5 POHDINTA JA JATKOKEHITYSEHDOTUKSIA 33
5.1 Yhteenveto ja pohdinta 33
5.2 Jatkokehitys 33
LÄHTEET JA TUOTETUT AINEISTOT. 35

KÄYTETYT LYHENTEET

ADO.NET = ADO.NET on joukko Microsoftin ohjelmistoja, joilla ohjelmoijat saavat yhteyden dataan ja datapalveluihin.
Android = Android on ohjelmistopino, mutta tässä työssä viitataan Androidilla kyseisen ohjelmistopinon käyttöjärjestelmiin.
BSD = Berkeley Software Distribution
C = Ohjelmointikieli
C++ = oliopohjainen ohjelmointikieli
cURL = Client Url. Mahdollistaa datan välityksen protollien välillä. Tässä työssä tarkoitetaan cURL:lla PHP kirjastoa, joka mahdollistaa kommunikoinnin toiseen palvelimeen Client Url:in avulla.
DBM = DataBase Manager.
Forth = Ohjelmointikieli
GPL = General Public License
GUI = Graphical User Interface
HTTP = HyperText Transfer Protocol
iOS = iPhone Operating System. Applen kehittämä käyttöjärjestelmä, joka on käytössä Applen eri laitteissa.
JDBC = Java Database Connectivity
mSQL = (tai) Mini SQL. Historiallisesti tärkeä ja kevyt tietokannan hallintajärjestelmä.
ODBC = Open Database Connectivity
PC = Personal Computer
Perl = Ohjelmointikieli
POSIX = Portable Operating System Interface for UNIX.
Postgres95 = PostreSQL:n edeltäjä
SGML = Standard Generalized Markup Language.
SMTP = Simple Mail Transfer Protocol on internet stantardi sähköisten viestien eli sähköpostien lähettämiseen
SQL = Structured Query Language
SVG = Scalable Vector Graphics, suomeksi skaalautuva vektorigrafiikka.
UNIX = laitteistoriippumaton käyttöjärjestelmä ja käyttöjärjestelmien perhe. Max OS X kuuluu Unixin tuoteperheeseen.
W3C = World Wide Web Consortium. On kansainvälinen yritysten ja yhteisöjen liittymä, joka ylläpitää ja kehittää www:n standardeja, joita kutsutaan suosituksiksi.
WWW = World Wide Web.
XML = eXtensible Markup Language

1 JOHDANTO

Weego Software on kuopiolainen ohjelmistotalo, joka keskittyy sähköisten monikanavaisten ratkaisujen tuottamiseen. Yhtiö on perustettu vuonna 2010 ja sen henkilöstölukumäärä on tällä hetkellä 8. (Weego Software Oy 2014.)

Weegon tärkeimmät asiakkaat ovat Samsung Electronics, FC Real Madrid ja Rovio Entertainment. Yhtiö keskittyy omassa liiketoiminnassa kahteen pääalueeseen; asiantuntijapalvelut toisille yrityksille ja sähköiset oppimisratkaisut Weegon omien tuotteiden pohjalta. Tärkeimpinä toimialoina ovat koulutus ja rakennusteollisuuden toimijat. (Weego Software Oy 2014.)

Työn tavoitteena oli jatkaa Weegon aloittamaa @Oppimaa –ympäristöä. Työn tilaaja eli Weego oli tehnyt aikaisemmin mobiiliversiot Android ja iOS –järjestelmille. He olivat aloittaneet nettisivujen tekemisen jo vuonna 2012, jota itsekin tein ennen opinnäytetyötä.

2 KÄYTETYT MENETELMÄT, TEKNIIKAT JA TYÖKALUT

2.1 Ketterä ohjelmistokehitys

Kymmenkunta kevyttä iteratiivistä ohjelmistonkehysmenetelmää on julkaistu 2000-lukuun mennessä. Siihen mennessä näistä eniten oli kerännyt huomiota Kent Beckin vuonna 1999 kehittämä XP (eXtreme Programming). (Haikala ja Mikkonen 2011, 43.)

Joukko kevyiden menetelmien kehittäjiä, kannattajia ja konsultteja kokoontui keskustelemaan käyttämistään ohjelmistokehysmenetelmistään Utahin hiihtokeskuksella vuonna 2001. He halusivat tehdä kevyitä menetelmiä raskaiden kehitysprosessien vastapainolle, joita he päättivät kutsua ketteriksi menetelmiksi. Kahden päivän aikana sovittiin kyseisten menetelmien yhdistävät periaatteet ja perustettiin näiden menetelmien edistämistä ajava järjestö Agile Alliance. Tämän peruskirjana julkaistiin Agile Manifesto, joka sisältää 12 seuraavaa periaatetta:

1. Tärkein tavoite on pitää asiakas tyytyväisenä toimittamalla hänelle alusta asti sopivin väliajoin toimivia ohjelmistoversioita.
2. Vaatimukset muuttuvat myös kehityksen myöhäisessä vaiheessa.
3. Versioita toimivasta ohjelmasta toimitetaan säännöllisesti.
4. Projektissa liiketoimintarooleissa olevien ja ohjelmistosuunnittelijoiden tulee työskennellä yhdessä päivittäin läpi projektin.
5. Projekteja tekevät motivoituneet työntekijät. He tarvitsevat hyvän tuen, ympäristön ja luoton tekemiselleen.
6. Tieto välittyy parhaiten kasvokkain käydyssä keskustelussa kehitystiimin ja tiimin jäsenten välillä.
7. Toimiva ohjelma on tärkein mittari projektille.
8. Ketterät menetelmat edistävät kestävää kehitystä. Projektin parissa työskentelevien henkilöiden pitäisi ylläpitää tahtia mahdollisimman pitkään.
9. Jatkuva teknisen erinomaisuuden huomiointi ja hyvä suunnittelu parantavat ketteryyttä.
10. Yksinkertaisuus on tärkeintä saadun valmiin työn maksimoinnissa.
11. Parhaimmat arkkitehtuuri, vaatimukset ja suunnitelmat syntyvät itseorganisoituvissa ryhmissä.
12. Ryhmän tulee tarkastella kuinka tulla tehokkaammaksi ja siten hienosäätää toimintaansa projektissa.

(Haikala ja Mikkonen 2011, 43-45.)

Manifestilla tarkoitetaan ohjenuoraa, jolla ketterää kehitystä tehdään. Ketterässä kehityksessä arvostetaan eniten yksilötä ja kanssakäymistä, toimivaa ohjelmistoa, asiakas yhteistyötä ja vastaamista muutoksiin.

Ketterä ohjelmistokehitys on hyvä kehitysmalli sen takia, koska siinä saadaan ensimmäinen versio tuotteesta nopeasti ja sitä myötä asiakaskin pääsee käyttämään sitä nopeasti. Sen lisäksi alkuperäinen suunnitelma ei yleensä ole tiukka, joten voidaan joustaa ja esimerkiksi kuunnella asiakkaan tarpeita.

2.2 HTML

HTML tulee sanoista Hypertext Markup Language. HTML:n kehitys alkoi CERN:n laboratoriossa 80-luvun lopussa. Sitä nykyisin käytetään www:n sivunkuvauskielenä. Sillä kuvataan hyperlinkkejä sisältää tekstiä ja merkitään tekstin rakennetta. HTML:ssä voidaan myös määrittää tyylejä, mutta tyylit ovat ensisijaisesti tarkoitettu määriteltäväksi CSS:ssä. HTML ei ole ohjelmointikieli, koska siinä ei prossesoida tietoa vaan lisätään kontekstia ja rakennetta tekstiin. (Romy 2012-05-04.)

HTML5 on uusin HTML:n virallinen suositus, jonka W3C julkaisi suosituksena 28. lokakuuta 2014. HTML5:n on katsottu syntyneen jo vuonna 2004 ja saaneen virallisen aseman vuonna 2007. Sen uudistuksina ovat mm. <canvas> <embedded>, <video> ja <audio> -elementit ja mahdollisuus näyttää SVG ilman <object> -elementtiä. Näistäkin syistä tässäkin työssä on käytetty nimenomaan HTML5, vaikkei se ollutkaan työn kehitysvaiheessa vielä W3C:n suositus. (Korpela 2011; Mozilla Developer Network 2014-11-04; World Wide Web Consortium 2014-10-28.)

2.3 CSS

CSS on merkintäjärjestelmä, jolla kuvataan dokumenttien erityisesti web-sivujen ulkoasua. CSS on lyhenne sanoista Cascade Style Sheet, joka tarkoittaa kirjaimellisesti suomeksi porrastettuja tyyliarkkeja. CSS kuvaa esim. nettisivuilla näkyviä tyylejä, kuten sivun taustakuvan, fontin värin tai jonkun elementin ulkoasun. CSS tekee sivujen ulkoasun vaikuttamisesta joustavampaa ja myös tehokkaampaa. Vaikka esimerkiksi selainohjelmoinnissa on mahdollista määrittää sivulle tyyliä HTML:llä, CSS tekee sitä selkeämmin ja tehokkaammin. Nykypäivän selainohjelmoinnissa CSS:n on välttämätön tekniikka tyyliasujen kirjoittamiseen. CSS kirjoitetaan yleensä erilliseen tiedostoon, jolla on .css –pääte. Tämä tehdään tietenkin siksi, että sivujen ylläpitäminen on paljon helpompaa. (Korpela 2008, 2.)

2.4 JavaScript

JavaScript on oliopohjainen, heikosti tyypitetty skriptikieli. Se on ECMAScriptin murre. Muita ECMAScriptin tunnettuja murteita ovat ActionScript ja Microsoftin JScript. (Peltomäki 2006, 90; Ulman 2012-03-17, 4.)

JavaScript syntyi vuonna 1995 nimellä Mocha ja muuttui myöhemmin nimeksi LiveScript. Maaliskuussa 1996 Netscape julkaisi JavaScriptin. Viimeisintä versiota 1.8.5 tukevat nykyisin käytännössä kaikki uudet selaimet. (Ulman 2012-03-17, 6.)

2.4.1 jQuery

jQuery on vuonna 2006 jQuery-tiimin kehittämä JavaScript –kirjasto, jonka tarkoitus on yksinkertaistaa JavaScript -koodia ja näin ollen helpottaa kehittäjän työtä. jQueryn iskulause onkin: ”kirjoita vähemmän, tee enemmän” ja se on nykyään maailman suosituin JavaScript –kirjasto. jQueryä kehitetään jatkuvasti ja viimeisimmat vakaat versiot ovat 1.11.1 ja 2.1.1, jotka julkaistiin

1. maaliskuuta 2014 mennessä. Tässä työssä on käytetty versiota 1.9.1. (The jQuery Foundation 2014; W3Techs 2014-12-01a; Methvin 2014-05-01).

```
// Koodi toistetaan kun DOM on ladattu
$(document).ready(function(){

    // JavaScript alert
    alert("Hei maailma!");

    // Laukeaa, kun klitataan p -elementtiä
    $("p").click(function() {

        // Muuttaa isäntäelementin, eli p-elementin taustavärin
        // punaiseksi ja leveyden 500 pikseliä leveäksi
        $(this).css({
            "background-color" : "red",
            "width" : "500px"
        });
    });
});
```

KUVA 1. JavaScript / jQuery -koodi, jossa sivun avautuessa näytetään alert -ikkuna ja muutetaan p-elementin ominaisuuksia sitä klikattaessa. (Saku Kaarakainen.)

2.4.2 JSON

JSON tulee sanoista JavaScript Object Notation ja tarkoittaa suomeksi JavaScript -olion merkintätapaa. Se onkin sisäänrakennettuna nykyisessä JavaScriptissä. JSON kirjoitetaan tekstiformaatissa ja se toimii täysin itsenäisenä. Se käyttää C-pohjaisten kielten merkintätapaa, mikä tekee siitä erinomaisen formaatin tiedonsiirtoon. JSON perustuu avain-arvo pareihin, jossa arvona voi olla tekstiä, taulukko tai olio. (The JSON Group 2014.)

```
{
    "key1" : "value1",
    "key2" : {
        "subkey" : "subvalue"
    }
    "array" : [
        0 : "first",
        1 : "second"
    ]
}
```

```
<?xml version="1.0" encoding="UTF-8" ?>
<key1>
    value1
</key1>
<key2>
    <subkey>
        subvalue
    </subkey>
</key2>
<array>first</array>
<array>second</array>
```

KUVA 2. JSON ja XML. Sama tieto esitetty ensin JSON- ja sitten XML-formaateissa. (Saku Kaarakainen.)

2.4.3 Ajax

Ajax tulee sanoista Asynchronous JavaScript And XML. Ajax on JavaScript tekniikka, jolla palautetaan asynkronisia HTTP-pyyntöjä. Käytännössä sen avulla saa tehtyä vuorovaikutteisempia nettisivuja. Vaikka Ajaxin nimen X –kirjain tulee XML –merkintäkielestä, ei Ajax tekniikalla ole pakko siirtää XML-muodossa tietoa. (Ferguson ja Heilmann 2013-01-26, 49.)

Kuvassa 3 on yksinkertaiset esimerkit Ajaxin käytöstä ensin perinteisellä JavaScript -tekniikalla ja sitten jQuery –kirjastoa hyväksi käyttäen. Kummassakin koodissa lähetetään get_movie_by_actor.php –tiedostolle GET -pyyntö. Parametrina siinä ovat first_name ja last_name ja niiden arvot ovat ”clint” ja ”eastwood”. PHP palauttaa tässä esimerkissä puhdasta tekstiä, joka näytetään alert-ikkunassa pyynnön valmistuessa.

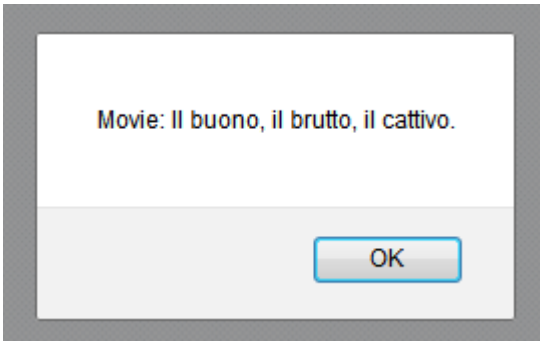
```
//-- Perinteinen JavaScript
var xmlhttp = window.XMLHttpRequest
    ? new XMLHttpRequest() /* IE7+, Firefox, Chrome, Opera, Safari */
    : new ActiveXObject("Microsoft.XMLHTTP"); /* IE5 & IE6 */

// Laukeaa aina, kun xmlhttp:n tila muuttuu
xmlhttp.onreadystatechange=function()
{
    // readyState = 4, kun ajax -pyyntö on valmis
    // status = 200, kun HTTP-pyyntö on onnistunut
    if(xmlhttp.readyState == 4 &&  xmlhttp.status == 200)
    {
        //xmlhttp.responseText pitää sisällään palvelimelta lähetetyn vastauksen.
        alert("Movie: "+xmlhttp.responseText);
    }
}

// Lähetetään first_name ja last_name parametrit get_movie_by_actor -tiedostolle.
xmlhttp.open("GET", "get_movie_by_actor.php?first_name=clint&last_name=eastwood", true);
xmlhttp.send();

//-- jQuery tekniikalla sama
$.ajax({
    type : "GET",
    url : "get_movie_by_actor.php",
    data : {
        "first_name" : "clint",
        "last_name" : "eastwood"
    },
    success : function(responseText) {
        alert("Movie: "+responseText);
    },
    error : function(e) {
        // alert("Ajax -pyyntö epäonnistui.");
    }
});
```

KUVA 3. Ajax -pyynto perinteisellä JavaScriptillä ja jQuery -kirjastoa käyttäen. (Saku Kaarakainen.)



KUVA 4. Esimerkki näytettävästä alert –ikkunasta, joka näytetään Ajax-pyynnön onnistuessa. (Saku Kaarakainen.)

2.5 PHP

PHP (PHP: Hypertext Preprocessor, alun perin Personal Home Page) on laajasti käytetty yleiskäyttöinen open source –skriptikieli. Se soveltuu varsinkin nettisivujen tekemiseen ja sitä voi käyttää HTML-koodin tekemiseen. Tässä työssä on käytetty PHP 5.3 –versiota, jonka tuki päättyi elokuussa 2014. Viimeisin vakaa versio PHP:stä on 5.6.3, joka julkaistiin 13. marraskuuta 2014. PHP seuraava versio on 7.0 (Lemos 2014-08-04.)

2.6 PHPMailer

PHPMailer on PHP-kirjasto, jonka avulla voidaan lähettää turvallisesti sähköpostia. Se tarvitsee toimiakseen toimivan SMTP-palvelimen. Sähköpostin lähettäminen suoraan PHP-koodissa ilman vastaanlaista kirjastoa vaatii hyvän osaamisen SMTP-palvelimesta ja siihen liittyvistä ongelmista kuten spämmäyksestä ja sähköposti-injektioista. (Worx International Inc. 2014.)

PHPMailerin lähdekoodi on julkisesti saatavana GitHubissa osoitteesta https://github.com/PHPMailer/PHPMailer. Kuvassa 5 näkyy esimerkki PHPMailerin käytöstä.

```php
<?php
    # Viittaa PHPMailerin class.phpmailer.php -tiedostoon
    # Muista, että viestin lähettämiseen tarvitaan toimiva SMTP-palvelin
    require_once('../class.phpmailer.php');

    # alustetaan $mail PHPMailer -olioksi.
    $mail = new PHPMailer();

    # Lähetetään contents.html viestinä formatoidaan sitä hieman.
    $body = file_get_contents('contents.html');
    $body = eregi_replace("[\]",'',$body);

    # Ilmoitetaan, keneltä viesti on tullut.
    $mail->AddReplyTo("nimi@yourdomain.com","Etunimi Sukunimi");
    $mail->SetFrom('nimi@yourdomain.com', 'Etunimi Sukunimi');
    $mail->AddReplyTo("nimi@yourdomain.com","Etunimi Sukunimi");

    # osoite, johon viesti lähetetään.
    $address = "jani.korhonen@esimerkki.fi";
    $mail->AddAddress($address, "Jani Korhonen");

    # Aihe
    $mail->Subject = "PHPMailerilla lähettty viesti.";

    # vaihtoehtoinen viesti
    $mail->AltBody    = "Nähdäksesi viestin, käytä HTML:ää tukevaa email vieweriä!";

    # Lisätään html-formaattinen viesti.
    $mail->MsgHTML($body);

    # liitteet
    $mail->AddAttachment("kansio/nimi.gif");

    # Lähetetään viesti.
    # Palauttaa (boolean) true -arvon jos lähetys onnistui.
    echo $mail->Send()
        ? "Viestin lähettäminen onnistui!"
        : "Viestin lähettäminen epäonnistui. Virhe : {$mail->ErrorInfo}";
?>
```

KUVA 5. Yksinkertainen esimerkki sähköpostiviestin lähettämisestä. (Worx International Inc. 2014.)

2.7 MVC

MVC on arkkitehtuurimalli graafisiin käyttöliittymiin perustuvien sovellusten toteuttamiseen. Se tulee sanoista Model-View-Controller, joka on suomeksi malli-näkymä-kontrolleri. Se on kehitetty Xeroxin Palo Alton tutkimuslaboratoriossa 1970 –luvulla. Vaikka MVC-mallilla viitataan usein suunnittelumalliin, se ei ole sitä, vaan tavallisempi filosofia siitä, millä tavalla sovelluksen järjestelmäarkkitehtuuri rakentuu. (Koistinen 2013.)

MVC:n perusajatus on erottaa käyttöliittymä sovelluslogiikasta ja –datasta. Siinä pyritään myös siihen, että järjestelmän siirtäminen graafiselta alustalta toiselle olisi helppoa. MVC-mallissa käytetään hyväksi Observer –suunnittelumallia. Sen hyviä puolia on mallin uudelleenkäyttäminen, useiden eri näkymien käyttäminen, näkymien muunneltavuus ja se on luonnollinen toteutus GUI-rakenteille. Se toisaalta monimutkaistaa järjestelmää ja on tehoton, varsinkin isommissa järjestelmissä. Sen lisäksi näkymä sekä ohjain ovat sidottuja toisiinsa. (Koistinen 2013.)

2.8 Zend Framework

Zend Framework on avoimen lähdekoodin oliopohjainen web-ohjelmistokehys, joka on kirjoitettu PHP 5:lla. Sen kehitti Zend Technologies. Zend Framework käyttää new BSD -lisenssiä. Sen ensimmäinen versio 0.1.1 on julkaistu 3. maaliskuuta vuonna 2006. O'Phinney ilmoitti (2014-09-17), että viimeisin version 2.3.3 julkaistiin 17. syyskuuta 2014. (Zend Technologies Ltd. 2014a, 2014b. 2014c.)

Zend Framework -kehystä käyttävät monet suuret yritykset kuten BBC, Offers.com ja Cisco. Sen lisäksi sen jatkuvaa kehitystä sponsoroivat mm. Google ja Microsoft. Zend-ohjelmistokehyksellä pystyy täydellisesti kirjoittamaan olio-ohjelmointiorientoitunutta koodia. (Hasan 2013-08-26, 2.)

2.9 MySQL

MySQL on vuonna 1995 ruotsalaisen David Axmarkin ja suomalaisen Michael Wideniuksen kehittämä relaatiotietokantaohjelmisto. Tällä hetkellä MySQL:n omistaa Oracle. Viimeisin versio MySQL:stä on 5.6.22 ja se julkaistiin 1. joulukuuta 2014. MySQL on tällä hetkellä toiseksi suosituin tietokanta Oracle –tietokannan jälkeen ja suosituin ilmainen tietokanta. MySQL:lle on laaja määrä julkista materiaalia ja oppaita, koska se on avoimen lähdekoodin yhteisöön kuuluva tietokantaohjelmisto. (Hamilton 2014-09-08; Oracle Corporation 2014-12-01; Solid IT 2014a.)

MySQL käyttää tietokantakielenään SQL-kieltä. Siihen tunnettuja rajapintoja ovat ADO.NET, JDBC ja ODBC. MySQL:ää tukee lukuisat ohjelmointikielet kuten C-pohjaiset kielet, Java, Perl ja PHP. Vuonna 2000 MySQL:stä tuli avoimen lähdekoodin sovellus ja se alkoi käyttämään GPL-lisenssiä. Vuonna 2008 Sun Microsystems osti MySQL:ää kehitävän MySQL AB:n noin yhden miljardin dollarin

kauppahinnalla, jonka jälkeen vuonna 2010 Oracle osti Sun Microsystemin. (Buytaert 2010-03-10; Euroopan Unioni 2010-01-21; Solid IT 2014b.)

```
-- Käytetään tietokantaa, jonka nimi on kirjakanta
USE kirjakanta;

-- Näytetään kaikki tiedot, mitä on kirja -taulussa
SELECT * FROM kirja;
/* OUTPUT:

_____________________________________________
|nimi       |   julkaisija  |julkaisuvuosi  |
---------------------------------------------
| Matikka   |               | 2003          |
| Historia  | Otava         | 2007          |
| ABC       | Otava         | 1999          |
---------------------------------------------
*/

-- Lisätään uusi kirja kantaa, sen nimi on Aapinen, julkaisija WSOY ja vuosi 2005
INSERT INTO kirja
VALUES ("Aapinen", "WSOY", "2005");

-- Päivitetään Matikka kirjaa lisäämällä sille julkaisija
-- Huomaa, että ilman where -ehtoa kaikkiin kirjoihin tulee annettu arvo
UPDATE kirja
SET julkaisija = "Klaava media"
WHERE nimi = "Matikka";

-- Poistetaan kirja-taulusta, kirjat joiden nimi = "ABC"
DELETE FROM kirja
WHERE nimi = "ABC";

-- Näytetään päivitetty taulu
SELECT * FROM kirja;
/* OUTPUT:

_____________________________________________
|nimi       |   julkaisija  |julkaisuvuosi  |
---------------------------------------------
| Matikka   | Klaava media  | 2003          |
| Historia  | Otava         | 2007          |
| Aapinen   | WSOY          | 2005          |
---------------------------------------------
*/
```

KUVA 6. SQL-koodiesimerkejä. (Saku Kaarakainen.)

2.10 Apache Server

Apache Server on vuonna 1995 kehitetty HTTP-palvelin. Se on suosituin HTTP-palvelin mitattuna sillä, kuinka moni sivusto käyttää sitä. Apache on kirjoitettu pääsääntöisesti XML- ja C –kielillä. Sen lisäksi siinä on käytetty mm. Forth ja C++ -kieliä. Apache –nimitys viittaa Pohjois-Amerikan Apassi-intiaaniheimoihin, jotka olivat tunnettuja heidän tehokkaista sodankäyntistrategioistaan ja ehtymättömästä kestävyydestään. Apache voidaan myös sanoa tulevan englannin kielisistä sanoista A Patchy Web Server, joka tarkoittaa suomeksi hajanaista web-palvelinta. Apache-palvelinta käytetään mahdollistamaan PHP:n tai muun ohjelmointikielen käytön web-sivujen luonnissa. (Black

Duck Software, Inc. 2014-11-10; The Apache Software Foundation 2012, The Apache Software Foundation 2014, W3Techs 2014-12-01b.)

2.11 Linux

Linuxilla viitataan Linux-pohjaisiin käyttöjärjestelmiin. Linuxin ydintä kutsutaan kerneliksi. Linux kernelin loi suomenruotsalainen Linus Torvalds vuonna 1991. Torvalds julkaisi työnsä avoimena, jotta muut voivat osallistua sen kehitykseen ja näin ollen kehittää sitä paremmaksi. Linuxin etu käyttöjärjestelmänä on sen ilmaisuus ja virusten sekä muiden haittaohjelmistojen vähyys. Tätä työtä on tehty enimmäkseen Ubuntu 13.04 - ja Debian käyttöjärjestelmillä. (Grant ja Bull 2012-07-22.)

3 @OPPIMAA –JÄRJESTELMÄ

@Oppimaa on järjestelmä, jossa kirjoja hallitaan käytetään elektronisessa muodossa. Se on suunniteltu ensisijaisesti oppikirjojen käyttöön ja hallintaan, mutta sitä pystyy käyttämään myös muihin tarkoituksiin. Esimerkiksi Weegon eräs asiakas on käyttänyt järjestelmää hyväkseen työmaiden raporttien tekemiseen ja luomiseen. Järjestelmä koostuu kolmesta eri osasta. Ensimmäinen on sisällönhallitsija, joka luo sekä kirjoja, että sisältöä niihin. Toinen osa on opettajan työkalu. Tätä voidaan kutsua myös raportointityökaluksi. Tällä työkalulla esim. opettajat voivat tarkastella oppilaiden vastauksia. Kolmas osa on oppilaan työkalu. Oppilaan työkalun avulla esim. oppilaat voivat käyttää oppikirjoja samalla tavalla kuin tavallista oppikirjaa. Sen netissä toimivaa versiota voidaan kutsua myös web-clientiksi. Weego on tästä osasta tehnyt valmiiksi natiivisovellukset Android- ja iOS –käyttöjärjestelmille. Järjestelmän mobiiliversiot ja tietokanta eivät tarkemmin kuulu tähän opinnäytetyöhön.

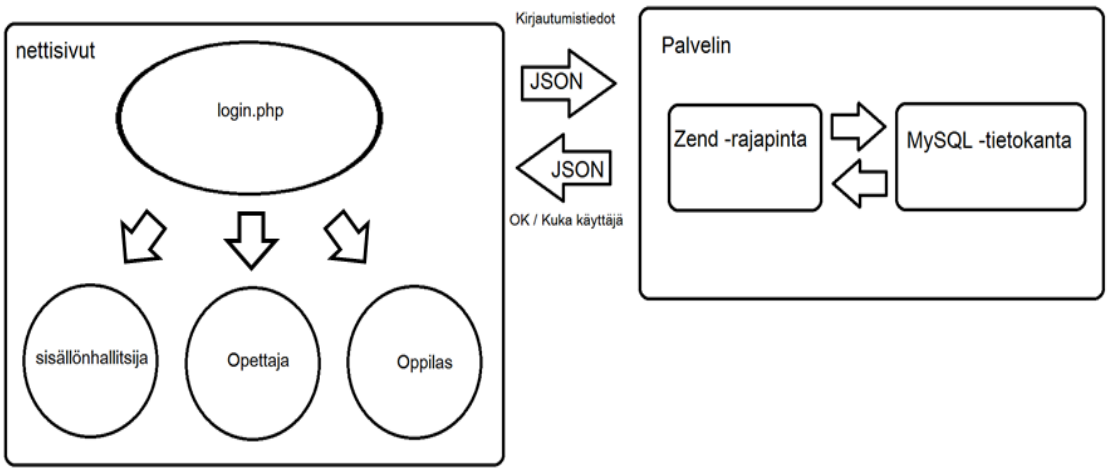
Järjestelmä on tehty Weegon palvelimilla. Niissä HTTP-palvelimena toimii Apache Server ja käyttöjärjestelmänä ovat Linux –käyttöjärjestelmät. Järjestelmästä on olemassa kehitys- ja tuotantoympäristöt, jotka ovat täysin erillisiä toisistaan. Tuotantoversioon pääsee julkisesta verkosta ja se on Weegon asiakkaiden käytössä. Lisäksi käytössäni työtä tehdessä oli Weegon tarjoama työkone, jolla kehitin toimintojen ensimmäiset versiot.

@Oppimaan-järjestelmän nettisivut ovat toteutettu PHP, HTML, CSS ja JavaScript –tekniikoilla. Kuten on aikaisemmin mainittu, sen sivut koostuvat kolmesta eri osasta, jotka ovat sisällönluonti, opettajan työkalu ja oppilaan työkalu. Jokainen osaa käyttää yhtä ns. pääsivua, joihin ladataan tietoa tarvittaessa muista PHP-tiedostoista JavaScriptin avulla.

KUVA 7. Login.php.

Nettisivuilla on yksi kirjautumissivu, joka näkyy kuvassa 7. Sitä käyttävät kaikki @Oppimaa –nettisivujen käyttäjät. Sisäänkirjautuminen tarkastetaan metodilla, joka kysyy ensin Zend-rajapinnalta käyttäjiä. Rajapinta vastaa, löytyykö käyttäjää tietokannasta ja mitkä ovat käyttäjän käyttöoikeudet nettisivuilla. Sen tiedon perusteella kannasta löytyneet käyttäjät ohjataan niiden oletusosaan. Sisällönluojat ohjataan sisällönluontiosaan, opettajat opettajan työkaluun ja oppilaat

oppilaan työkaluun. Sisällönhallitsijoilla on käyttöoikeudet opettajan työkaluun ja heillä sekä opettajilla on myös käyttöoikeudet oppilaan työkaluun. Sisäänkirjautumisesta tallennetaan tieto sessioniin, joka poistetaan jos käyttäjä ei ole aktiivisena riittävän pitkään. Joka kerta sivua päivittäessä järjestelmä tarkastaa sisäänkirjautumisen metodilta, löytyykö käyttäjää sessionissa, onko sillä valittuun osaan oikeus ja löytyykö käyttäjä tietokannasta. Jos yksikin näistä tarkistuksista epäonnistuu, käyttäjä ohjataan kirjautumissivulle.



KUVA 8. Sisäänkirjautuminen. Ensin kysytään käyttäjää rajapinnasta, jonka jälkeen ohjataan käyttäjä oikeaan osaan.

3.1 Zend-rajapinta

@Oppimaa käyttää tietokantanaan MySQL-kantaa. Siihen otetaan yhteys Zend-rajapinnan kautta. Zend-rajapinta on Zend Frameworkillä tehty rajapinta, joka ottaa vastaan ja lähettää tietoa JSON-formaatissa. Tämä rajapinta on toteutettu MVC-mallin mukaisesti. Myös Weegon tekemät mobiilisovellukset käyttävät myös tätä samaa rajapintaa. @Oppimaa –järjestelmään on toteutettu sendJson_cUrl -metodi, joka tekee kommunikoinnin rajapinnan kanssa. Se ottaa parametrina vastaan urlin, joka on käytännössä rajapinnan tietyn metodin nimi ja toisena parametrina lähetettävän JSON:in. Metodi palauttaa rajapinnan lähettämän JSON:in.

Kuten mainittiin, Zend-rajapinta on tehty Zend Frameworkilla, joka on toteutettu MVC-mallilla. Se tarkoittaa käytännössä sitä, että Zend-palvelimella on kolme MVC-malliin kuuluvaa kansiota, jotka ovat models, views ja controllers.

Models –kansiossa on jokaista tietokannan taulua kohden kaksi tiedostoa. Ensimmäinen on taulun niminen php-tiedosto, joka sisältää ns. business logiikan. Toinen on mapperi, joka sisältää varsinaiset tietokantahaut tekevät funktiot.

Views –kansiossa ovat rajapinnasta ulospäin näkyvät tiedostot. Tämän kansion alla on sen lisäksi alikansioita, jotka ovat nimetty tietokannan taulujen mukaan. Näihin alikansiohin on lisätty .phtml –

tiedostot, joissa kaikissa on kuvan 9 mukainen koodi. Tiedostot ovat nimetty metodin nimen mukaan, eli millä niitä kutsutaan rajapinnan ulkopuolelta.

```
<?php echo $this->reply_JSON?>
```

KUVA 9. Koodi, joka kirjoitetaan kaikki views -kansion .phtml tiedostoihin.

Controllers –kansiossa ovat varsinaiset metodit. Siellä on jokaiselle taululle oma kontrolleri. Esim. Books –taululle on siellä BookController.php –tiedosto. Metodit ovat nimetty samannimisiksi, kuin niiden .phtml –tiedostot näkymässä, mutta niiden nimeen perään on lisätty Action –sana, jotta Zend Framework tunnistaa metodit rajapinnasta ulospäin kutsuttaviksi metodeiksi.

```
<?php

class BookController extends Zend_Controller_Action
{

    public function init()
    {
        /* Initialize action controller here */
    }

    public function indexAction()
    {
        // action body
    }

    public function insertbookAction()
    {
        if($this->getRequest()->isPost()) {
            $email = @$jsondetails["email"];
//          ...

            $bookMapper = new Application_Model_BookMapper();

//          ...

            $jsonreply = $data_from_database;
            $this->view->reply_JSON = json_encode($jsonreply);

        } else {
            // Print error
        }
    }
}

?>
```

KUVA 10. Controllereiden perusrakenteesta esimerkki.

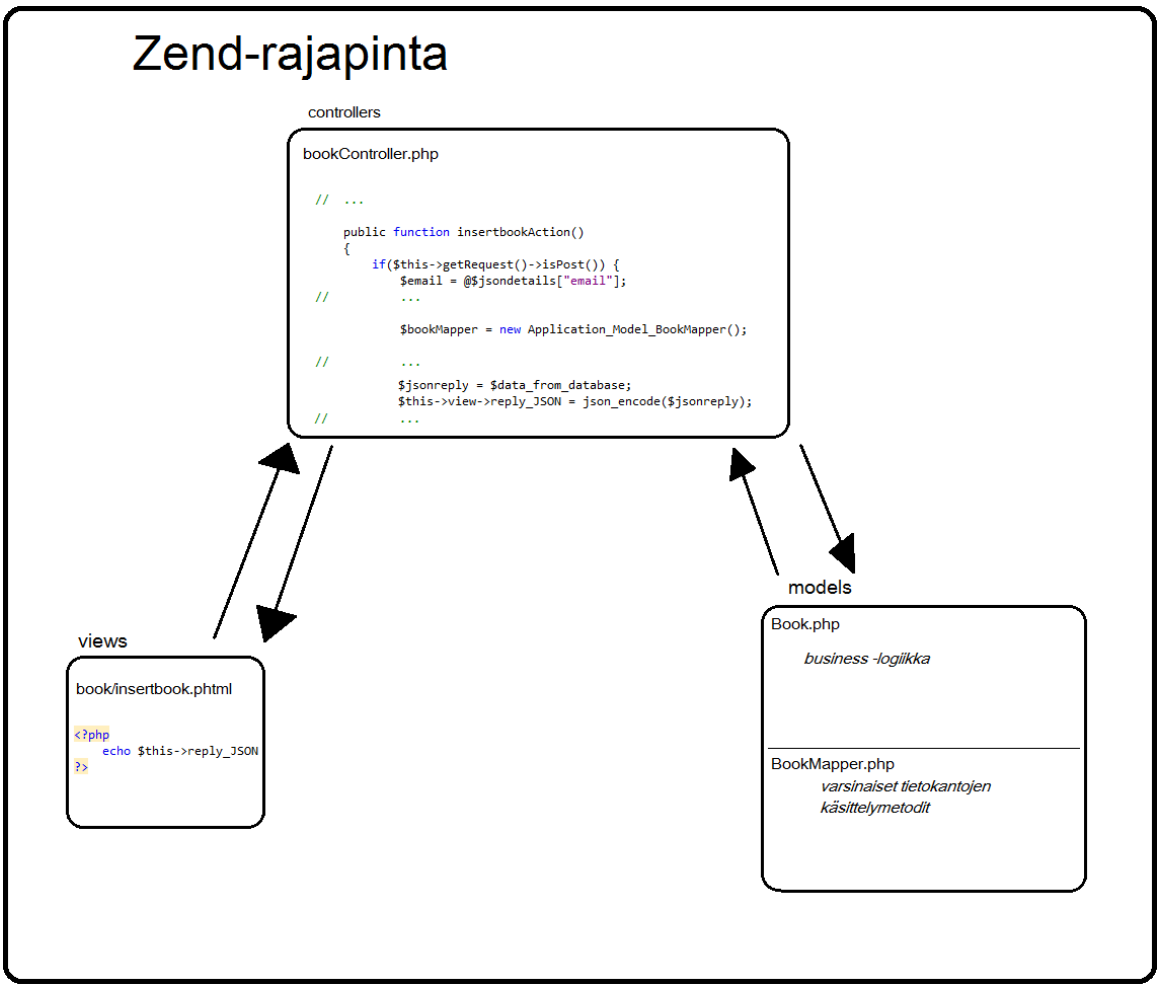
Esimerkkinä nettisivuilla sisällönluoja Kalle haluaa lisätä uuden kirjan järjestelmään. Rajapinta on IP-osoitteessa 192.168.1.1:13443 ja siellä on insertbook –niminen metodi, jonka avulla lisätään kirjoja tietokantaan. Metodi käyttää book –nimistä taulua. Views –kansiossa on siis book –niminen kansio ja

siinä kansiossa on insertbook.phtml –tiedosto. Nettisivut siis lähettävät cUrl -kutsun osoitteeseen 192.168.1.1:13443/book/insertbook, johon annettaan POST-parametrina kuvan 11 mukainen JSON.


```
{
    "email" : "Kalle",
    "bookname" : "uusi_kirja",
    "description" : "kuvaus tähän",
    "layout" : 3,
    "book_category" : "Templates",
    "book_status" : "created",
    "creation_ts" : "2014-09-01 15:15:15",
    "pages" : 3
}
```


KUVA 11. Insertbook -metodille lähetettävä JSON.

Rajapinnan saama kutsu menee views -kansiossa olevan book -kansion insertbook.phtml –tiedostoon. Zend Framework tunnistaa book -kansion, josta se tietää metodin löytyvän BookController.php –tiedostosta. Zend tunnistaa vielä tiedoston nimestä, että kutsuttava metodi on insertbookAction BookController.php -tiedostossa. InsertbookAction -metodissa on luotu Models –kansiossa olevasta book.php –tiedoston luokasta olio, jota on käytetty tietokantojen tietojen tallentamiseen. Sitä kutsutaan nimellä object_book. Samoin on luotu myös Models –kansiossa oleva bookMapper.php -tiedoston luokasta olio, jota kutsutaan nimellä object_bookMapper. Object_bookMapper sisältävät tarvittavat varsinaiset tietokantakyselyt tietokannassa olevaan book -tauluun. JSON:ista saadut parametrit tallennetaan object_book -olioon. Object_book annetaan object_bookMapper –olion sisältämälle metodille, joka suorittaa MySQL INSERT –kyselyn. Tämän jälkeen luodaan JSON, jossa kerrotaan onnistuiko tiedon lisäys ja palautetaan se vastauksena.



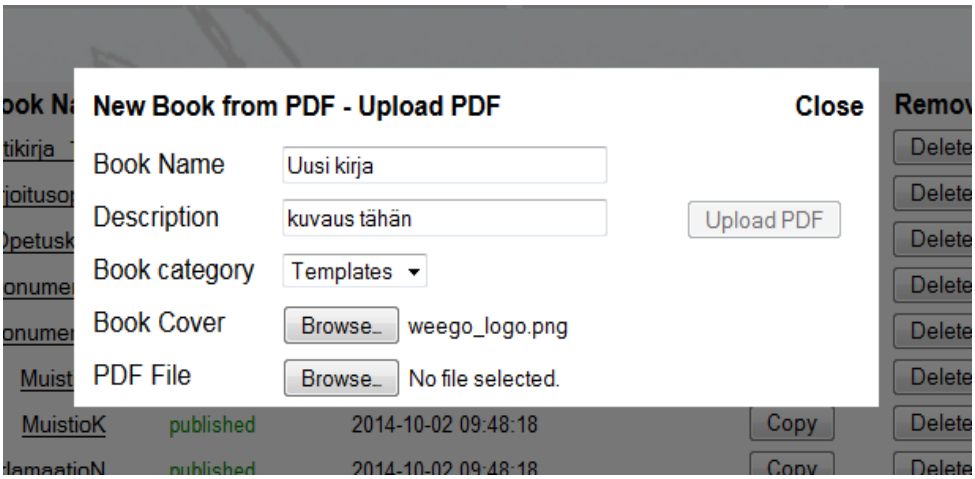
KUVA 12. Tiedon liikkuminen rajapinnassa.

3.2 Sisällönhallinta –työkalu

Sisällönhallinta-järjestelmää käyttävät sisällönluojat. Siellä voidaan luoda, muokata, kopioida ja poistaa kirjoja. Lisäksi testiversiossa on olemassa myös kirjan arkistointi, jossa kirjoja voi siirtää arkistoon ja sieltä takaisin. Sisällönhallintatyökalussa on viisi välilehteä: kirjat (Books), Sivun asetukset (Page Settings), Page Triggers, Sivun elementit (Page Elements) ja Assets. Sisällönhallintatyökalusta pystyy siirtymään myös raportointipuolelle ja takaisin.

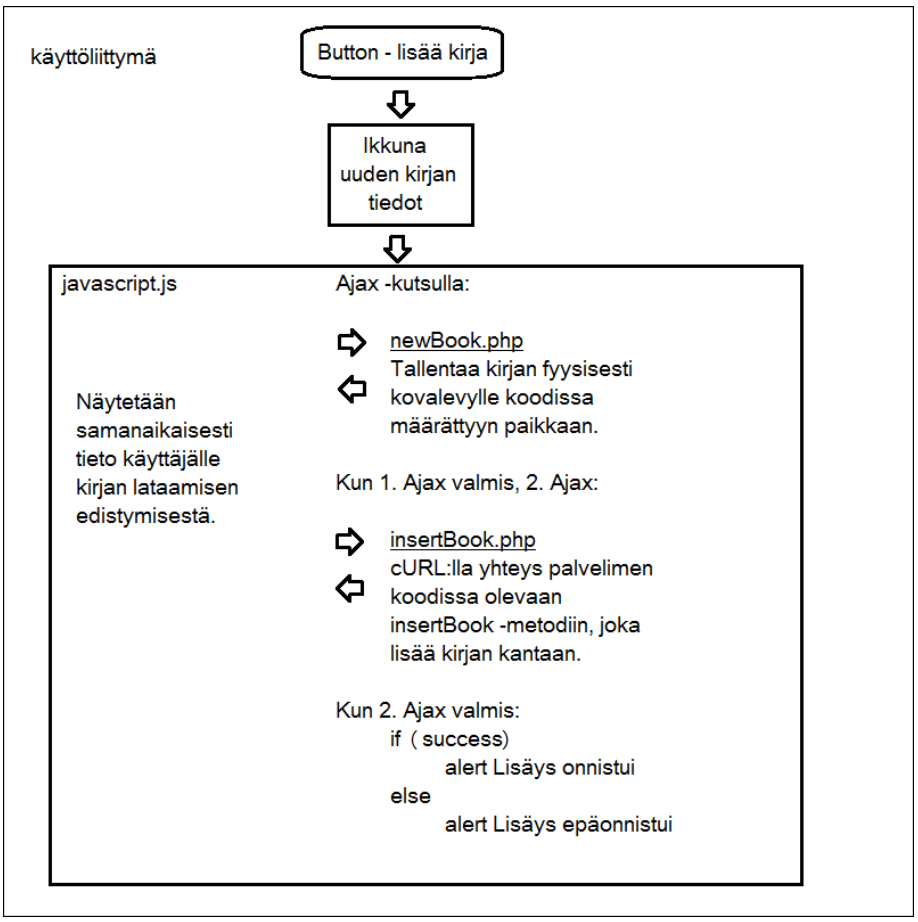
Kirjat tallennetaan järjestelmään books-kansioon kansioina. Kansiot ovat samalla nimellä tietokannassa, mutta kansioiden nimistä on korvattu välit ja erikoismerkit alaviivalla ja niihin on lisätty books_ -etuliite. Kansioissa sivut ovat PNG-tiedostoina. Jokaiselle kirjan sivulle on olemassa pageNN.png ja pageNN.json. NN tarkoittaa siinä liukuvaa numeroa. Esimerkiksi jos sisällönhallitsija on luonut 20 sivun kirjan. Kirjan kansiossa on silloin kirjan sivut page01.png, …, page20.png ja sivuille on elementtejä koskevat JSON-tiedostot page01, …, page20.json. Sen lisäksi kirjan kansiossa on bookCover.png –tiedosto kirjan kantena ja assetit, jotka ovat elementteihin liitetyt tiedostot.

3.2.1 Esimerkki uuden kirjan lisäys



KUVA 13. Uuden kirjan lisäämisikkuna.

Kuvassa 14 näkyy tapahtuman kulku uutta kirjaa lisättäessä. Tapahtuma alkaa painamalla ”lisää kirja” –näppäintä sisällönhallintatyökalussa, joka on Booklist -välilehdellä. Tämä avaa ikkunan, joka näkyy kuvassa 13. Ikkunassa annetaan kirja PDF-formaatissa, kuva kirjan kannesta ja muut tiedot. Tässä esimerkissä käyttäjä antaa kirjan nimeksi Uusi kirja, jolle annetaan kolmen sivun kokoinen PDF-tiedosto. Tietojen lisäämistä ei pysty aloittamaan, ennen kuin välttämättömät tiedot ovat annettu. Näitä ovat kirjan nimi, kansi ja itse kirja. JavaScript lähettää Ajax –kutsun insertBook.php ja newBook.php –sivuille, kun ikkunassa painetaan upload PDF –näppäintä. Lataamisen ajan käyttäjälle näytetään latautumissivua, jossa näytetään tietoa kirjan lataamisen edistymisestä järjestelmään. Sillä välin newBook.php-tiedoston avulla ladataan kirja järjestelmän kovalevylle ja kerrotaan jatkuvasti JavaScriptille lataamisen edistymisestä latautumissivulle. Kun newBook.php:n Ajax -kutsu on valmis, aloitetaan uusi Ajax, jossa päivitetään tietokanta. Tässä kutsutaan insertBook.php:tä, joka ottaa yhteyden palvelimen Zend-rajapinnan insertBook –metodiin. Tämä metodi suorittaa tietokantapäivityksen kirjalle. Käyttäjälle ilmoitetaan tieto kirjan lisäämisen onnistumisesta tai epäonnistumisesta, kun metodit ja Ajax-kutsut ovat päättyneet.



KUVA 14. Tapahtumakulku uuden kirjan lisäämisessä.

Onnistuneen lisäämisen jälkeen ”Uusi kirja” –niminen kirja löytyy tietokannasta ja käyttäjän näkökulmasta kirjalistalta. Books kansiossa on book_uusi_kirja, jossa on kuvatiedostot bookCover.png, page01.png, page02.png ja page03.png ja JSON-tiedostot page01.json, page02.json ja page03.json. JSON-tiedostoihin on kirjoitettu tässä vaiheessa kuvan 15 JSON, jossa on tietenkin background –avaimen arvo sama kuin sen tiedoston nimi.

```
{
    "page": {
        "background": "page01",
        "eraserUsage": "allowed",
        "soundUsage": "allowed",
        "activityType": 0
    },
    "pageElements": []
}
```

KUVA 15. Page01.json -tiedoston sisältö kirjan luomisen jälkeen.

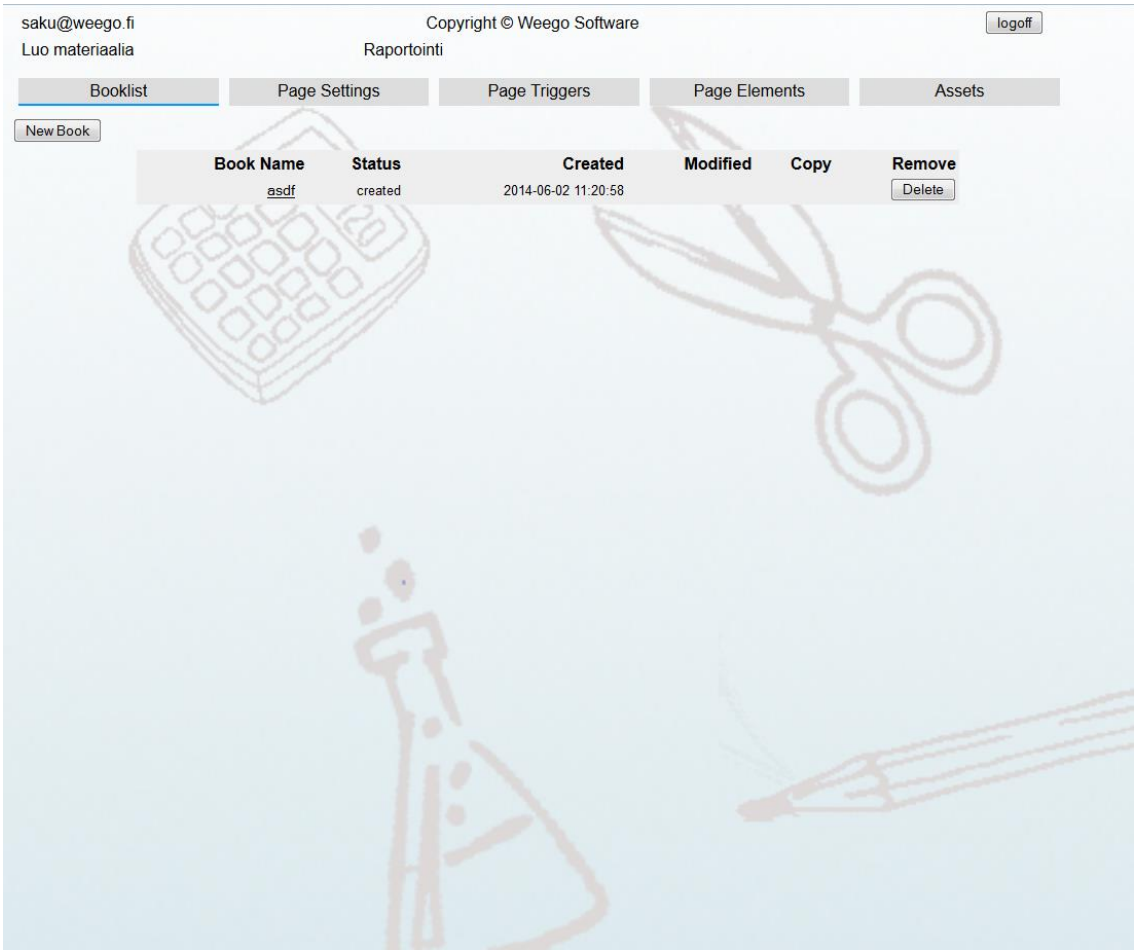
Järjestelmä on luonut tässä esimerkissä books –kansioon book_uusi_kirja –nimisen kansion. JSON –tiedostoon lisätään oletuksena kaksi elementtiä, jotka ovat page ja pageElements. Page –elementtiin tallennetaan oletuksena generoitu olio. Se sisältää tietoa sivun oletusasetuksista, joita voidaan muokata järjestelmän Page Settings –välilehdellä. Sen lisäksi tallennetaan pageElements –niminen tyhjä taulukko, jonne tallennetaan Page Elements –sivulla tallennettavat asiat, eli sivun elementit.

PageElement -taulukkoon tallennetaan sisällönhallitsijan luomat elementit olioina. Tässä vaiheessa pageTriggers –elementtiä ei vielä kirjoiteta JSON-tiedostoon.

3.2.2 Booklist

Järjestelmään sisäänkirjautuessa ensimmäisenä avautuu booklist –välilehti, jossa näkyy käyttäjän hallittavat kirjat. Booklist –välilehdellä on mahdollista lisätä uusia kirjoja ja kopioimaan sekä poistamaan olemassa olevia kirjoja. Kirjojen muokkaaminen tapahtuu muilla välilehdillä.

Booklist välilehti sisältää myös arkistointimahdollisuuden, joka on saatavilla vain testiversiossa. Sivulta voi valita haluamansa kirjat arkistoon, jolloin niiden sisältöä ei pysty enää muokkaamaan. Kirjan voi kuitenkin palauttaa arkistosta, jonka jälkeen kirja on taas muokattavissa.

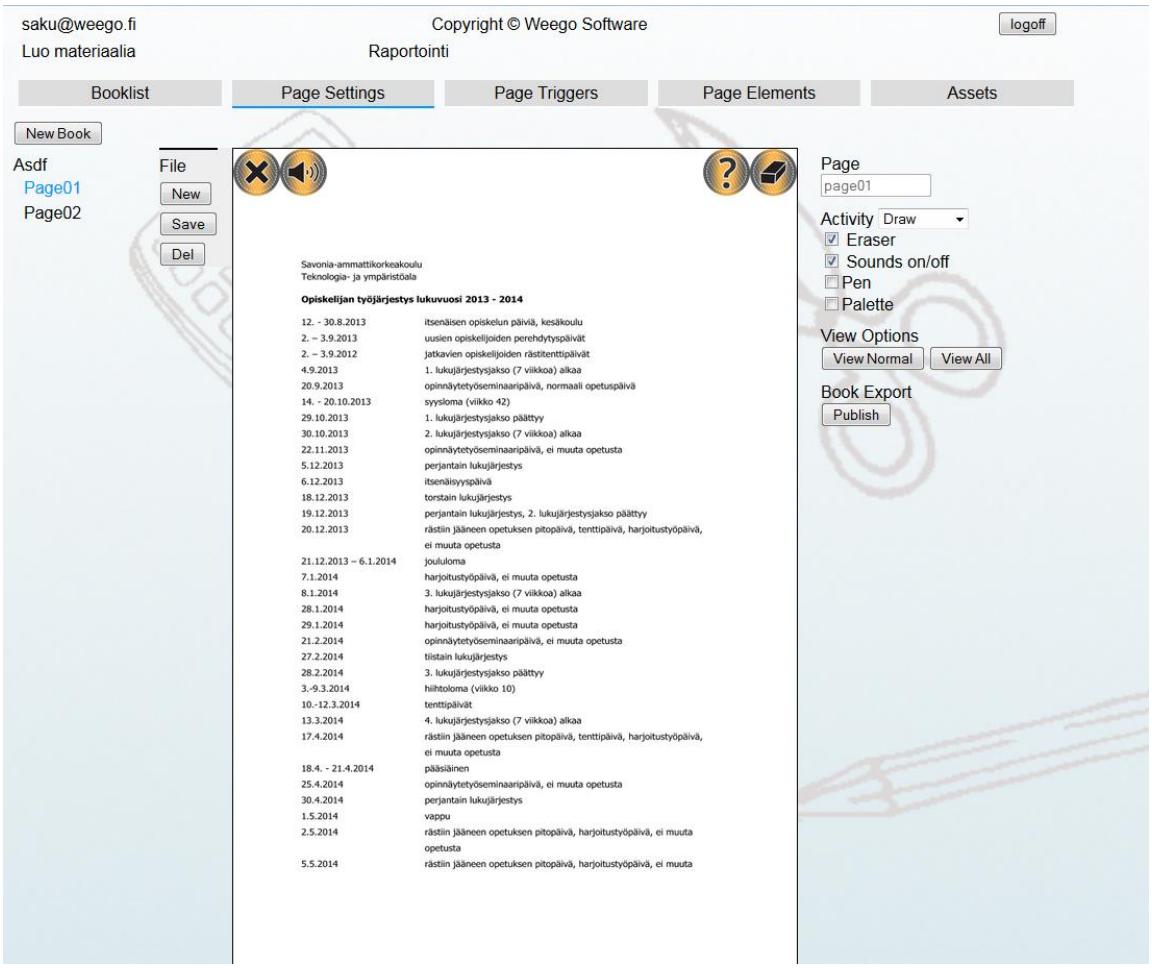


KUVA 16. Ensimmäinen näkymä sisällönluojalle kirjautuessa järjestelmään.

3.2.3 Page Settings

Page Settings –välilehdellä tehdään valitun kirjan sivulle asetuksia, kuten mitä aktiviteettiä kirja käyttää. Aktiviteetillä tarkoitetaan käytettäviä elementtejä kokonaisuutena, joita hallitaan Page Elements -välilehdellä. Elementit ovat asioita, kuten tekstikenttä tai äänen tallennusnäppäin, joita käyttäjät käyttävät oppilaan tai opettajan työkaluilla. Lisäksi siinä valitaan, käyttääkö sivu ääntä,

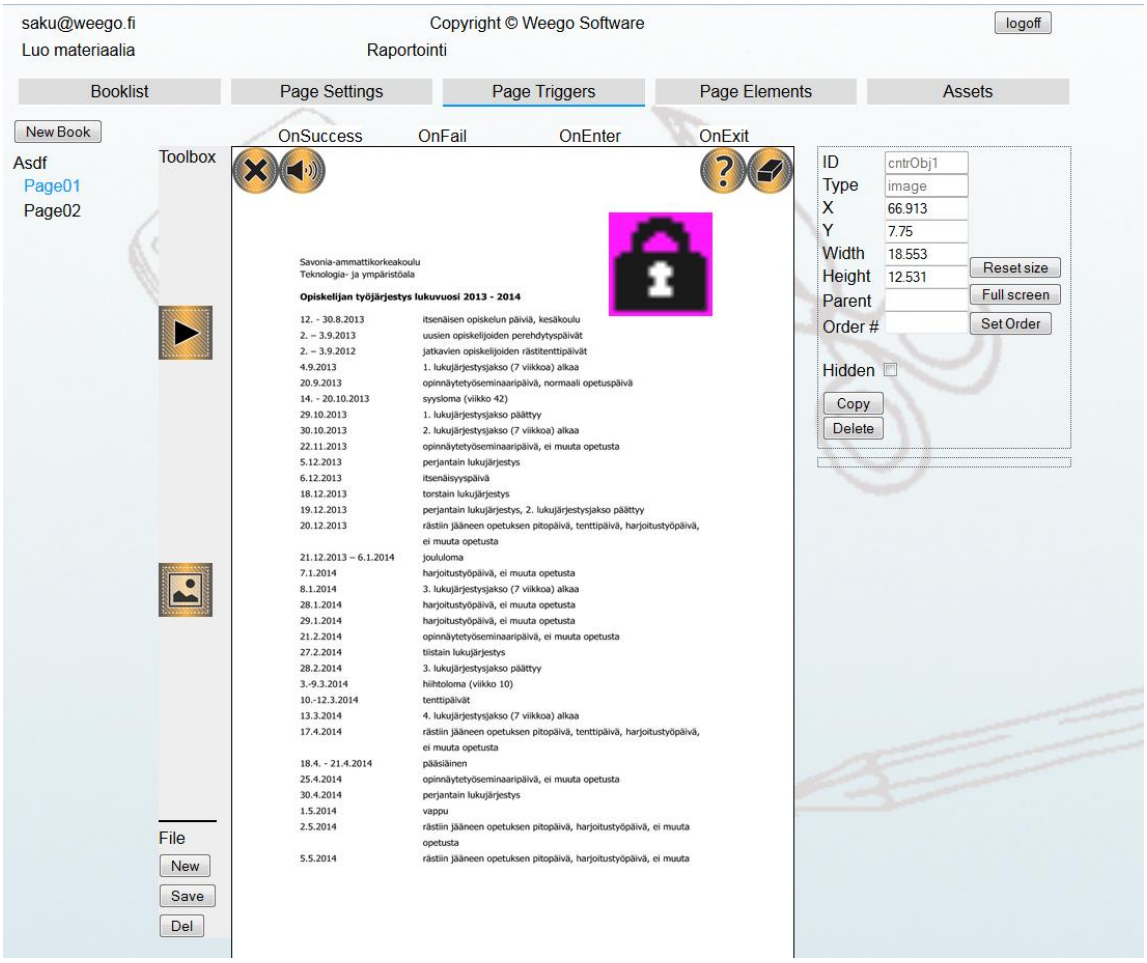
kynää, kumia tai valmiita muotoja eli paletteja. Välilehdellä pystyy myös poistamaan vanhoja sivuja ja lisäämään uusia.



KUVA 17. Sisällönluoja - Page Settings.

3.2.4 Page Triggers

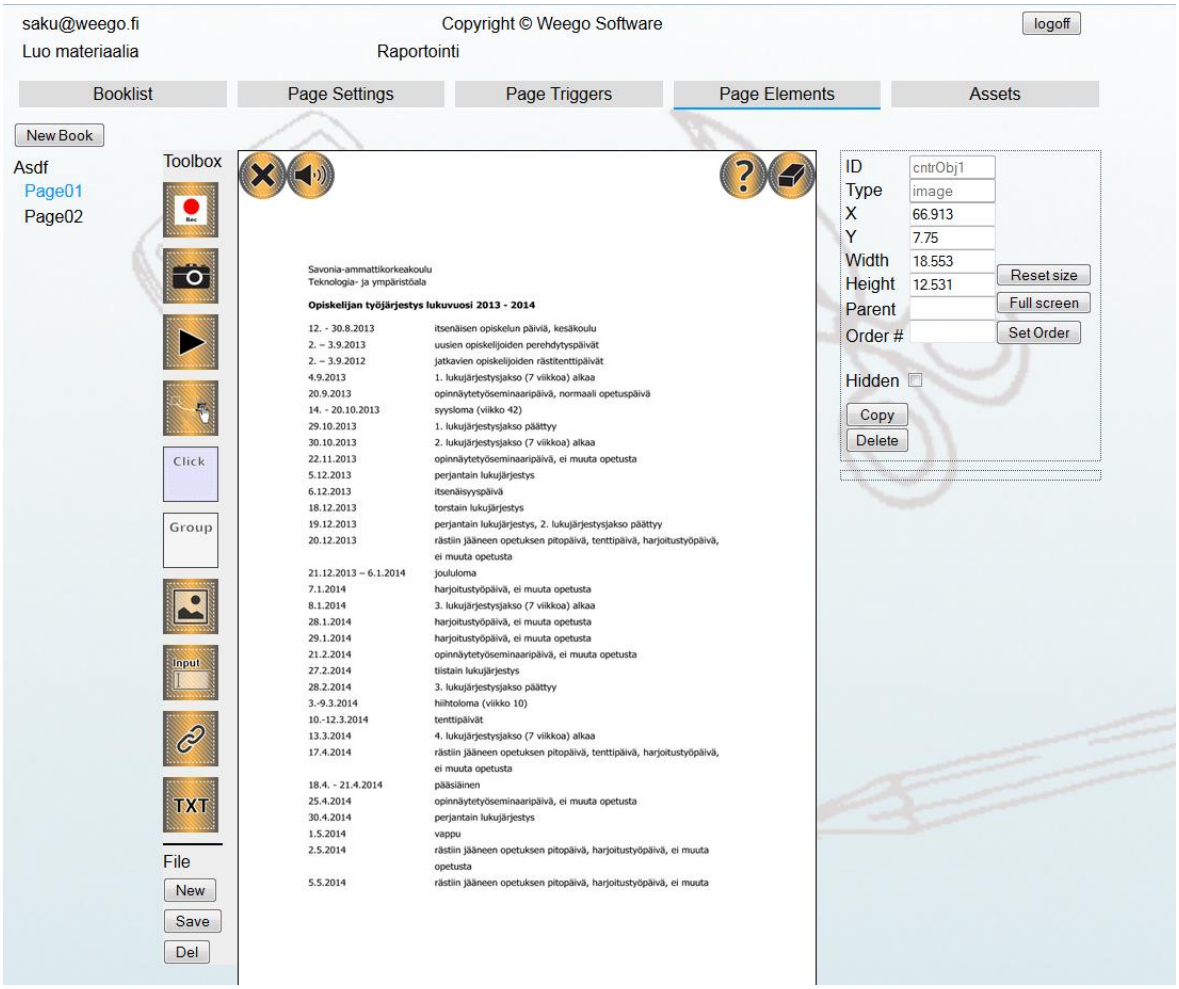
Page Triggers –välilehti sisältää neljä alivälilehteä. Näitä ovat OnSuccess, OnFail, OnEnter ja OnExit. Alivälilehdet ovat nimetty ns. Triggereiden eli tapahtumien mukaan. Triggereitä käytetään jonkun tapahtuman laukeamiseen oppilaan työkalussa. OnSuccess laukeaa käyttäjän suoritettua jokin tehtävä onnistuneesti. OnFail laukeaa taas päinvastaisessa tilanteessa eli käyttäjän epäonnistuessa tehtävässään. OnEnter laukeaa aina kirjan sivun avautuessa ja OnExit laukeaa kirjan sivulta poistuttaessa. Page Triggers –välilehdellä lisätään siis tapahtumakohtaisia elementtejä. Valittavia elementtejä tapahtumiin ovat kuva ja ääni. Esimerkiksi lisättäessä kuva OnSuccess –välilehdellä kirjan sivulle, oppilaan työkalulla tehdyn onnistuneen tehtävän jälkeen näytetään aina kyseinen kuva sillä kirjan sivulla, jolle se on työkalulla lisätty.



KUVA 18. Sisällönluoja - Page Triggers.

3.2.5 Page Elements

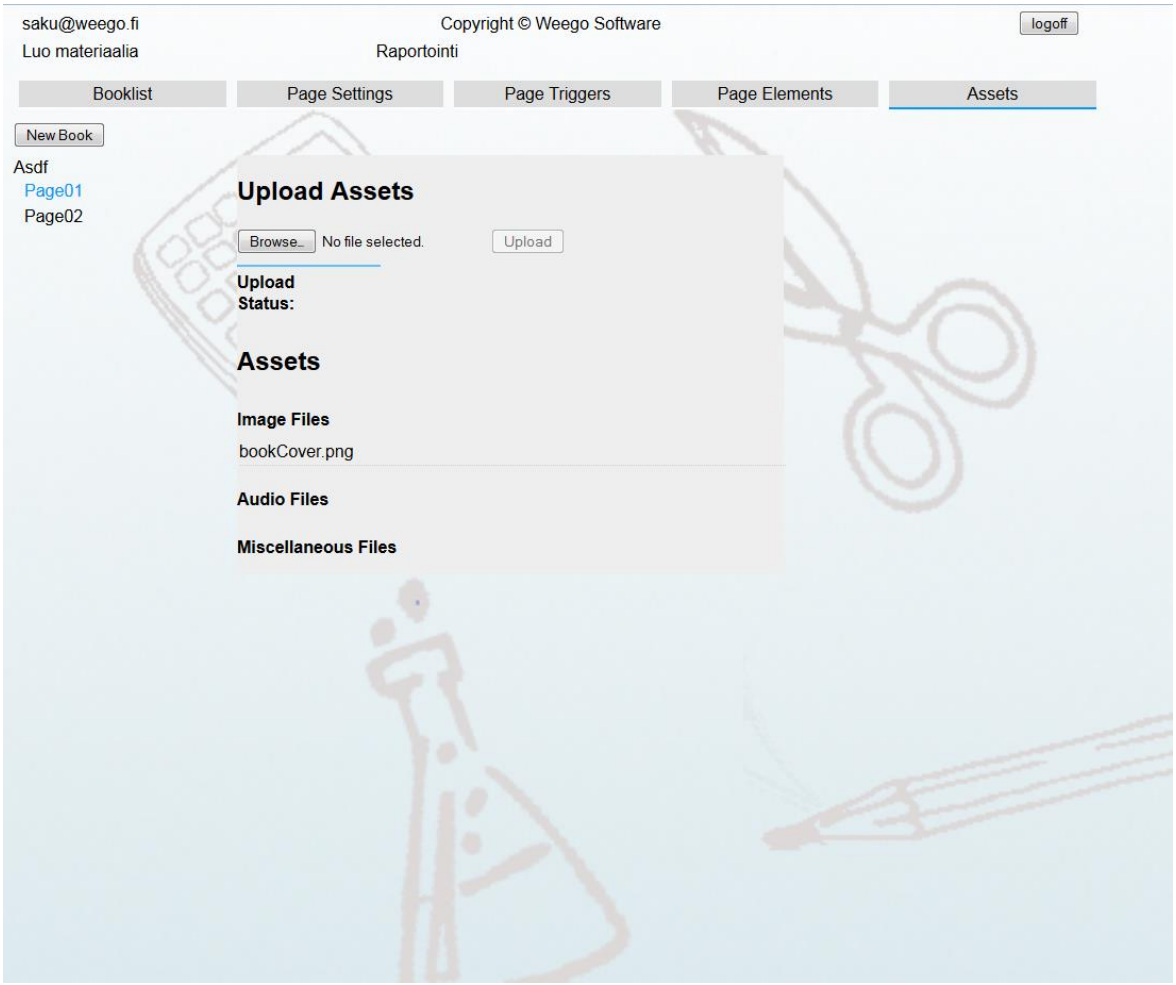
Page Elements –välilehdellä lisätään erilaisia elementtejä sivulle. Elementtejä ovat record, camera, audio, drag, click, group, image, input, link ja text. Sivulle lisätty record -elementti nauhoittaa käyttäjän luomaa ääntä. Camera -elementti tallentaa käyttäjän syöttämän kuvan. Audio -elementti on sisällönhallitsijan tallentama ääni, joka soitetaan kyseistä elementtiä painettaessa oppilaan työkalussa. Drag –elementillä raahataan kuva valittuun pisteeseen. Se siis sisältää kaksi kohdetta, jotka ovat itse kuva ja sen kohde. Group -elementin avulla voidaan ryhmitellä muita elementtejä. Image -elementti lisää kuvan sisällönhallitsijan valitsemaan kohtaan. Input –elementillä pystyy oppilaan työkalussa syöttämään tekstiä valittuun input –kenttään. Link -elementillä sisällönhallitsija voi lisätä kirjan sivulle hyperlinkin. Text -elementin avulla sisällön hallitsija kirjoittaa tekstiä, joka näytetään tekstinä oppilaalle ja opettajalle.



KUVA 19. Sisällönluoja - Page Elements.

3.2.6 Assets

Assetteja ovat tiedostot, joita kirja käyttää hyväkseen. Niitä lisätään järjestelmään Assets – välilehdellä. Välilehdellä näkyät kaikki assetit, jotka ovat liitetty valittuun kirjaan. Assets -tiedostot ovat luokiteltu kolmeen ryhmään: kuvatiedostot, äänitiedostot ja muut.



Kuva 20. Sisällönluoja – Assets.

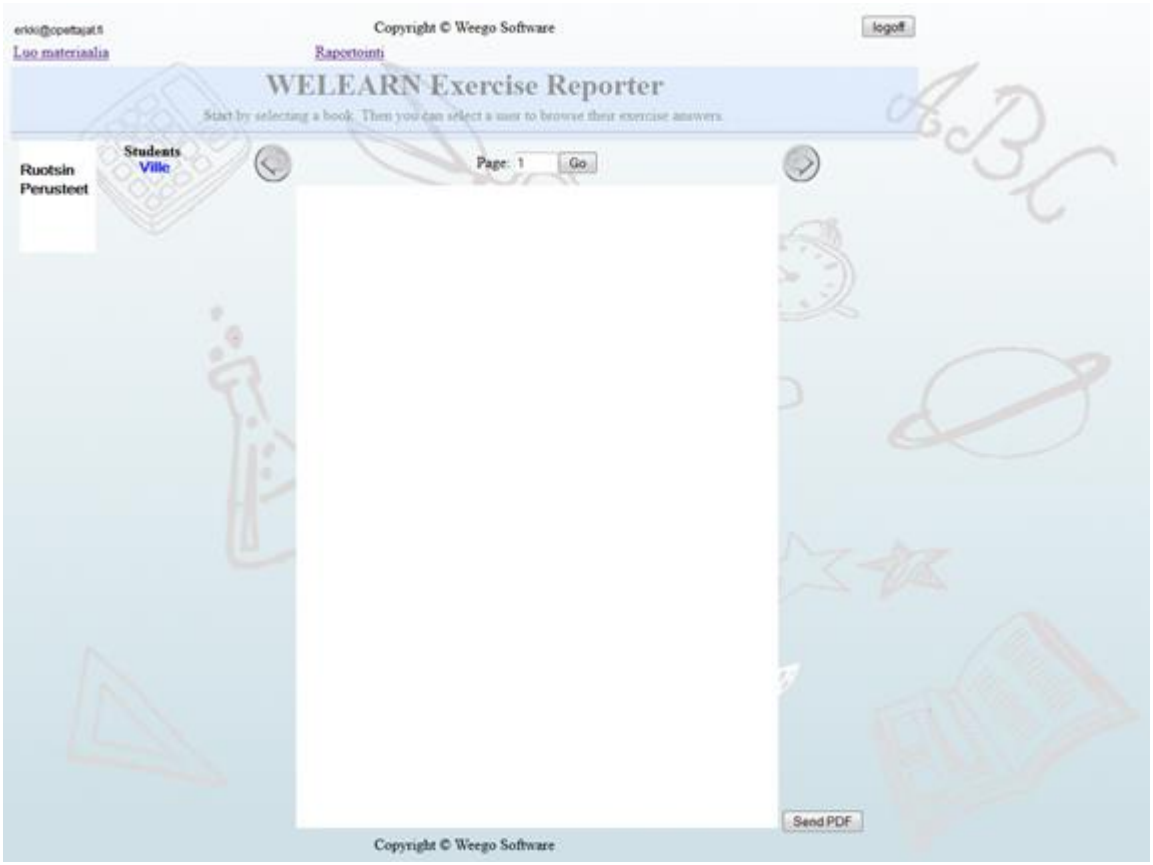
3.3 Raportointi-työkalu

Raportointi-työkalu, jota kutsutaan myös opettajan työkaluksi on järjestelmän toinen osa. Työkalu on ensisijaisesti suunniteltu opettajan käyttöön, mutta sitä voi käyttää myös muuhun tarkoitukseen. Työkalua pystyy käyttämään esimerkiksi rakennustyömaalla oppilaan työkalulla tehtyjen raporttien katselmointiin. Työkalulla luetaan oppilaan tekemiä vastauksia. Oppilaan työkalulla otetaan kuvakaappaus sivusta, jota raportointi-työkalulla tarkastellaan. Tällä hetkellä opettajan työkalussa ei ole mahdollisuutta kirjoittaa tietoa raportteihin, eli opettaja ei voi kommentoida tai arvioida vastauksia vaan joutuu vielä kirjaamaan käsin tiedot raportista.

Opettajan ja oppilaan kirjat ovat samassa kansiossa Zend-rajapinnassa. Kirjat ovat rajapinnassa, jotta kyseisiä kirjoja pystyy käyttämään myös mobiililaitteilla. Kirjat ovat rajapinnassa storage – nimisessä kansiossa. Siellä kirjat ovat kansioina nimettyinä samannimisinä kuin tietokannassa, mutta niistä on korvattu väliviivat ja erikoismerkit alaviivalla. Kirjan sisällä on jokaisen oppilaan vastaus omassa kansiossaan. Kansiot on nimetty sillä ID:llä, millä oppilas löytyy tietokannasta. Sen sisällä on jokaisesta vastauksesta eli raportista kuvakaappaus. Kuvakaappaukset ovat nimetty N_capture.jpg, jossa N tarkoittaa oppilaan vastauksen sivunumeroa.

Työkalua käytetään niin, että opettaja valitsee ensin tarkastettavan kirjan. Tämän jälkeen opettaja valitsee oppilaan, jonka vastausta hän haluaa katsoa. Sen jälkeen avautuu kirjan ensimmäinen sivu.

Esimerkkinä opettaja Erkki tarkastaa Ville –nimisen oppilaan vastauksen. Villen ID on tietokannassa 5. Ville on tehnyt ruotsin perusteet –kirjan sivun 1 tehtävät ja tallentanut sen arvioitavaksi. Villen vastaus siis löytyy järjestelmän kovalevyltä osoitteesta storage/ruotsin_perusteet/5/1_capture.jpg. Opettaja Erkki avaa vasemman reunan kirjalistasta ruotsin perusteet. Sen jälkeen tulee viereen Students otsikon alle oppilaat, jotka ovat tehnyt siihen kirjaan vastauksia, joista opettaja valitsee Villen. Valittuaan sen opettaja näkee sivuilla kuvan 21 tilanteen, jossa hän voi arvioida Villen vastauksia. Tällä hetkellä hän voi ainoastaan kirjata käsin vastauksia tai lähettää haluamaansa sähköpostiosoitteeseen PDF-tiedoston, joka sisältää kaikki villen vastaukset.



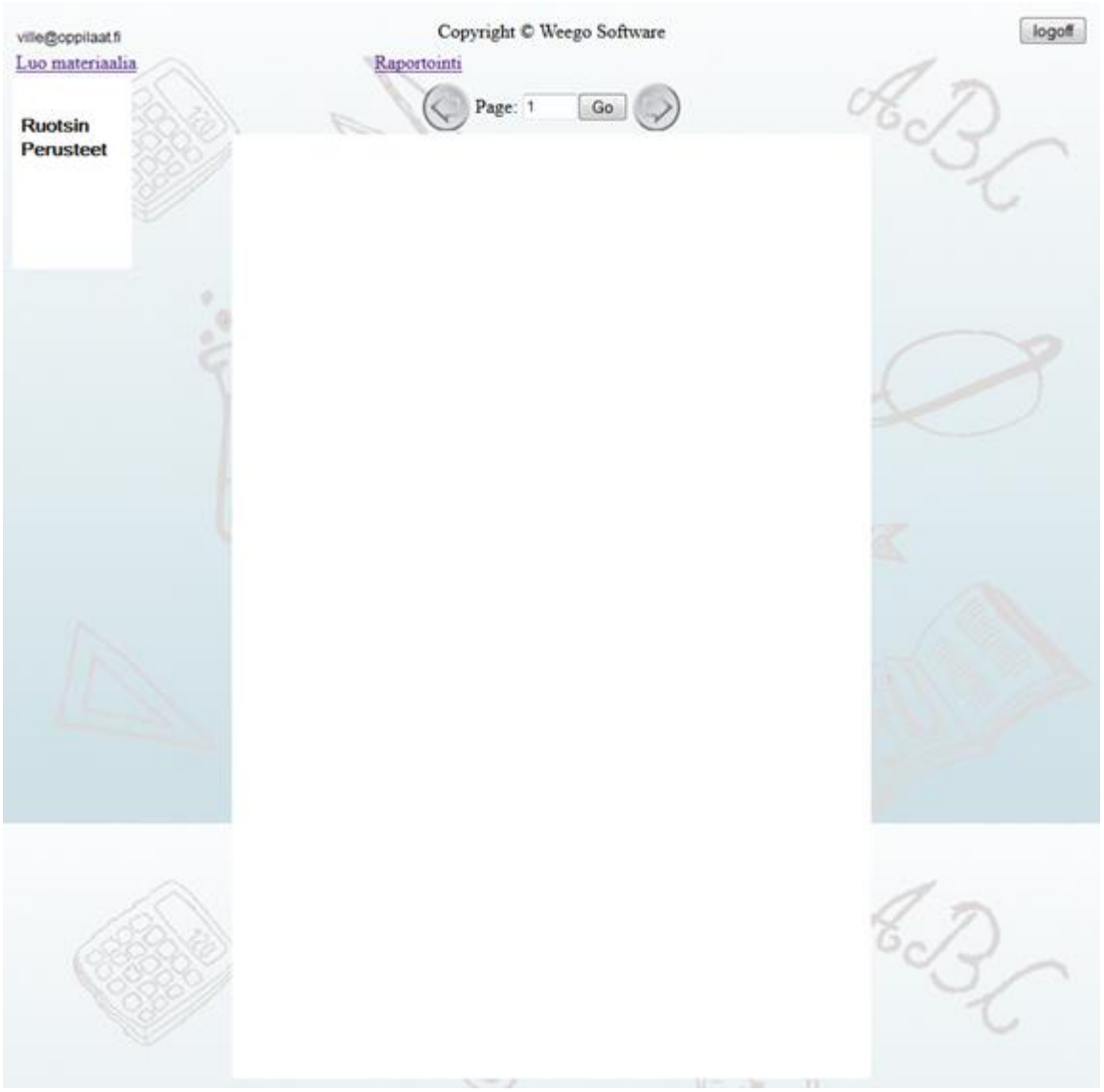
KUVA 21. Opettajan näkymä.

3.4 Oppilaan työkalu

Oppilaan työkalulla luetaan kirjoja. Siitä on olemassa myös omat erilliset versiot Android ja iPhone –käyttöjärjestelmille tabletteihin ja puhelimiin. Kuten on jo aiemmin mainittu, oppilaan kirjat ovat samassa storage –kansiossa Zend-rajapinnassa. Oppilaan vastaukset ovat kirjan kansiossa zip-tiedostoina. Esimerkiksi ruotsin perusteet –nimisen kirjan osoite on storage/ruotsin_perusteet/ruotsin_perusteet.zip

Ensimmäisenä oppilaan oikeuksilla kirjautuessa tulee lista kirjoista, joihin oppilaalla on käyttöoikeus. Kirjaa klikattaessa avautuu kirjan ensimmäisen sivu. Kirjat sisältävät elementit, joita sisällönhallitsija on luonut. Esimerkiksi, jos sisällönhallitsija on lisännyt jollekin sivulle OnEnteriin sivunavausäänen, kyseinen ääni soitetaan sivua avattaessa. Järjestelmä tallentaa automaattisesti käyttäjän eli oppilaan

tekemät muutokset. Muutoksilla tarkoitetaan esimerkiksi, jos käyttäjä on ottanut kuvan image-elementin kautta tai kirjoittanut tekstiä input-elementtiin. Sitten kun oppilas haluaa tallentaa vastauksensa opettajan luettavaksi, hän painaa save –näppäintä. Tallentaminen ei ole vielä mahdollista ja se tehdään mahdollisessa jatkokehityksessä.



KUVA 22. Oppilaan näkymä.

4 TOTEUTUS

Weego oli tehnyt vuonna 2012 järjestelmän, joka oli silloin nimeltään welearn. Kyseiseen järjestelmään oli siihen mennessä tehty valmiiksi tuotteiksi Android- ja iOS-järjestelmille toimivat ohjelmat oppilaan työkaluina. Heillä oli silloin keskeneräisenä nettisivuilla toimiva sisällönhallinta-järjestelmä. Kehitin viimeisessä harjoittelussani näitä nettisivuja ja päädyin jatkamaan samaa työtä opinnäytetyönä.

Weegon kanssa pidettiin palaveri opinnäytetyötä aloittaessa, jossa suunniteltiin työhön lisättävät tehtävät. Tehtävät eivät olleet kankeasti sidottu suunnitelmaan, joten ominaisuuksia saatettiin vaihtaa tarvittaessa esimerkiksi asiakkaan pyynnöstä. Tälläinen joustava suunnitelma oli mahdollista, koska työtä tehtiin ketterän ohjelmistokehityksen mukaisesti.

Kuten on aikaisemmin mainittu, työtä tehtiin ketterän ohjelmistokehityksen mukaisesti. Toimeksiantaja ei ottanut kantaa käyttämiini työkaluihin vaan tarjosi vaihtoehtoja. Itse päädyin kuitenkin käyttämään järjestelmiin valmiiksi asennettuja ohjelmistoja. Omalla koneella käytin geany-editoria ja kehitys- ja tarvittaessa palvelimilla käytin vim-editoria.

Opinnäytetyölle oli merkitty takaraja. Tämän takia aikakäsitys erosi perinteisestä ketterän ohjelmistokehityksen periaatteesta. Pidimme toimeksiantajan eli Weegon pomon kanssa joka päivä pikaisen tilannekatsauksen siitä, miten pitkälle on edetty projektissa. Pidimme myös palavereita, kun joku tietty kokonaisuus saatiin valmiiksi ja annoin tilannepäivityksen aina valmiista testiversiosta. Tämä tapa oli siinä mielessä mielestäni hyvä, että tuotantoversiossa oli käytännössä jatkuvasti toimiva versio järjestelmästä. Se oli myös hyvä, ettei suunnitelma ollut tiukka, koska jos toimeksiantaja tai sen asiakas halusi järjestelmään jonkin ominaisuuden, sen kehittäminen alkoi nopeasti. Myöskin jos jokin suunniteltu ominaisuus kävi ylimääräiseksi tai turhaksi, se oli helppo pudottaa alkuperäisestä suunnitelmasta. Itse pidän tälläistä joustavaa tapa mielekkäänä, koska reagoiminen on nopeaa, eikä asiakkaan tarvitse tietää kaikkea haluaamaansa heti.

Töiden alussa meni hieman aikaa uusiin työtapoihin. Aluksi yksin tehdessäni nettisivuja riitti, että kehitin ensimmäisen version työkoneella ja siirsin sen jälkeen testipalvelimelle testattavaksi. Kun testiversiota oli tarpeeksi pitkään testattu palvelimella, se voitiin todeta riittävän luotettavaksi siirtää tuotantopalvelimelle. Tämän jälkeen sain uudet määritelmät, eli seuraavan tehtävän, jonka tein samaan tapaan kuin edellisen.

Projektin loppuvaiheessa Weegon työntekijät alkoivat kehittämään järjestelmään muita osia. Sen takia siirryttiin käyttämän Git –versionhallintaohjelmistoa. Työtapaa jouduttiin muuttamaan niin, että testiversiota ei siirretty suoraan testipalvelimelle, vaan se tallennettiin ensin Gittiin. Se että kaikki työntekijät käyttävät Git-ohjelmistoa, tekee kehittämisestä helpompaa, koska kaikilla on jatkuvasti ajan tasalla oleva versio projektista.

4.1 Oma osuus työstä opinnäytetyönä.

Alkuperäiseen suunnitelmaan kuului muokata järjestelmän kaikkia kolmee osaa. Sisällönhallintatyökaluun oli tarkoitus lisätä drag, radiobutton, checkbox ja list –elementit. Opettajan työkaluun piti lisätä kuvan, äänen ja kuvakaappauksen lisäksi myös mahdollisuus näyttää tekstiä tietokannasta. Oppilaan työkalulla aikaisemmin pystyi vain katsomaan kirjoja ja syöttämään niissä oleviin input –elementteihin tekstiä. Siihen haluttiin lisätä drag, path ja click –elementit toimivina ominaisuuksina.

4.2 Tehty työ

Lopulliseen työhön lisäsin yhtenäisen, joskin melko karkean otsikon, joka näytetään kaikissa järjestelmän osissa. Tässä näkyy sisäänkirjautunut käyttäjä, sen organisaatio, uloskirjautuminen ja navigointi järjestelmin muihin osiin. Lisäksi hioin harjoittelussani tekemän kirjautumissivun käyttämään HTML5 ominaisuuksia JavaScriptin sijasta.

4.2.1 Sisällönluonti

Sisällönluonti-työkaluun oli tehty aikaisemmin kirjan poistaminen. Siinä kirjan poistaminen tarkoitti sitä, että tietokantaan kirjan status sai arvoksi deleted, eikä sitä sen jälkeen voinut enää käyttää. Poistaminen muutettiin niin, että se poistaa kirjan kokonaan sekä tietokannasta, että kovalevyltä. Tämän muutoksen takia poistamiselle täytyi tehdä Zend-palvelimelle oma metodi. Kirjan kopiointi myös saatettiin loppuun. Järjestelmään tein täysin uutena ominaisuutena kirjan arkistoinnin. Järjestelmän kovalevylle lisättiin uusi archive –niminen kansio, jonne laitetaan kaikki arkistoidut kirjat. Vaikka tämä toiminto saatiin toimimaan kunnolla, sitä ei koskaan ole siirretty jostain syystä tuotantoversioon. Page Settings -välilehdelle lisäsin kirjan julkaisemisen, joka mahdollistaa kirjan käyttämisen oppilaan työkalulla, kunhan oppilas saa kirjalle käyttöoikeuden.

Page Triggers –välilehdelle lisäsin äänen ja kuvan lisäämismahdollisuuden, kuten alkuperäisen määritelmänkin mukaan oli tarkoitus tehdä. Tein loppuun drag –elementin, joka oli ollut keskeneräinen ennen työn alkua.

4.2.2 Opettajan työkalu

Opettajan työkaluun lisättiin elementit toimivina, kuten alkuperäiseen määritelmään kuului. Uutena ominaisuutena lisäsin mahdollisuuden lähettää kirja käyttäjän valitsemaan sähköpostiosoitteeseen PDF –formaatissa.

Toisena uutena ominaisuutena lisäsin kuvakaappauksen kroppauksen. Kroppaaminen tarkoittaa sitä, että kuvakaappauksesta leikataan pois työkalupalkki, joka näkyy mobiilisovelluksilla otetuissa kuvakaappauksissa. Android- ja iOS –käyttöjärjestelmien mobiilisovelluksissa työkalupalkit ovat oppilaan työkalun sovelluksissa eri kohdissa, jonka takia ne ovat ne ovat kuvakaappauksissakin eri

kohdissa. Ominaisuus saatiin toimimaan, mutta se osasi leikata työkalupalkin pois vain tietyillä laitteilla otetuista kuvista. Sen takia ominaisuus jäi keskeneräiseksi ja vaatii jatkokehitystä.

4.2.3 Oppilaan työkalu

Lisäsin äänien toistomahdollisuuden ja kuvien näyttämisen oppilaan työkaluun, kuten oli tarkoitus. Siinä huomioitiin myös triggerit, joita luodaan sisällönhallintatyökalussa Page Triggers –välilehdellä. Esimerkiksi jos sisällönluoja on laittanut kirjan ensimmäiselle sivulle onEnter –triggeriin avausäänen, se toistetaan aina kun kirjan ensimmäinen sivu avaataan. Oppilaan työkalussa olivat myös tekstikentät virheellisissä kohdissa, joten korjasin ne oikeille paikoilleen.

4.2.4 Zend-rajapinta

Kaikki lisäämäni metodit käyttävät pääsäntöisesti book –taulua getbyuser –metodia lukuunottamatta, joka käyttää organization -taulua. Se tarkoittaa sitä, että näkymässä kaikki lisätyt tiedostot ovat books kansiossa ja getbyuser on organization kansiossa. Sen lisäksi kaikki metodit ovat kirjoitettu BookController.php –tiedostoon, paitsi tietenkin getbyuser on OrganizationController.php –tiedostossa.

Ensimmäinen metodi on removebook. Se poistaa kirjan kaikista tietokannan tauluista, eli se poistaa kirjan kokonaan tietokannasta. Toinen metodi on insertbookfromtemplate, joka luo uuden kirjan valmiin mallin pohjalta. Kolmas on listbooksbyauthor, joka listaa kaikki sisällönhallitsijan kirjat. Neljäntenä on listcategories ja se palauttaa kaikki mahdolliset kategoriat kirjalle. Tätä metodia käytetään sisällönluonnissa mm. uuden kirjan luonnissa kategorian valitsemiseen. Sen takia se on myös käyttäjäkohtainen. Viides metodi on changebookstatus, jolla vaihdetaan kirjan status tietokannassa. Kuudentena metodina on listmybooks. Se listaa valitun oppilaan kaikki kirjat ja sen avulla listataan oppilaan työkalussa näkyvät oppilaan kirjat. Seitsemäs metodi on changedirectorypath. Se vaihtaa tietokannassa kirjan directory pathin eli kantaan muutetaan sijainti, josta kirja löytyy järjestelmässä. Viimeinen metodi on getbyuser. Getbyuser palauttaa kannasta organisaation tiedot haetun käyttäjän perusteella.

4.2.5 Muut

Sain toimeksiantajalta tehtäväksi lähettää PDF-tiedostoja sähköpostina. Weegolla ei ollut aikaisemmin tähän ratkaisua. Päädyin valitsemaan PHPMailer –kirjaston, koska näin ollen säästyi aikaa, joka olisi kulunut oman vastaavan kirjaston tekemiseen. Lisäksi sähköpostin lähettäminen vaatii SMTP –palvelimen, mikä oli Weegolla pelkästään tuotantopalvelimella. Tästä syystä kehittämisen joutui tekemään suoraan tuotantopalvelimelle, mikä ei ole yleensä kovin hyvä käytäntö mahdollisten bugien ja muiden odottamattomien ominaisuuksien takia.

5 POHDINTA JA JATKOKEHITYSEHDOTUKSIA

5.1 Yhteenveto ja pohdinta

Työstä muotoutui loppujen lopuksi melko mielenkiintoinen. Opin mielestäni hyvin työn aikana JavaScript ja PHP –kieliä varsinkin olio-ohjelmointikielinä. Olin harjoittelussani kehittänyt Zend-rajapintaa, joka on tehty Zend Framework –ohjelmistokehystä käyttäen. Sen takia osasin Zend Frameworkin käyttöä hieman, mutta osaamiseni parantui selvästi työtä tehdessä. Metodien lisäämisestä rajapintaan alkoi tulla minulle sujuvaa, sekä aloin ymmärtämään rajapinnan käyttämän ohjelmistokehyksen hyötyjä. Lisäksi koin mielenkiintoisena toimeksiannon, jonka tehtävänä oli lähettää sähköpostiä PHP-koodin avulla.

Opinnäytetyön aloitin tekemään jo maaliskuussa, sekä tein nettisivut osuuden valmiiksi jo kesäkuussa. Työn raportin tekeminen venyi kuitenkin joulukuuhun asti. Kun tein varsinaisia nettisivuja, olisin voinut sen ohella dokumentoida työtä paremmin. Olin kuitenkin dokumentoinut sen verran, että se sisälsi kaiken riittävän materiaalin.

5.2 Jatkokehitys

Tämä projekti on ollut laajin projekti, jossa olen työskennellyt. Jo pelkästään Zend-rajapinnasta riittäisi todennäköisesti aihetta toiselle opinnäytetyölle puhuttamattakaan esimerkiksi puuttuvasta mobiiliversiosta eli sovelluksesta Windows Phone –käyttöjärjestelmälle. Nettisivuissakin olisi vielä paljon tekemistä.

Ensimmäinen työ jatkokehityksessä olisi päivittää PHP 5.3 käyttämään uusinta vakaata versiota, koska sitä ei enää tueta. Ongelmallista saattaa olla esimerkiksi taulukot, koska ne tehdään eri tavalla PHP 5.4:ssa kuin tässä käytetyssä PHP:ssä. Mielestäni kannattaisi tutkia, olisiko järkevää ottaa käyttöön jokin kehys, jolla saisi yhtenäisemmän ulkoasun sivuille.

Tällä hetkellä työssä liian pitkään inaktiivisena olevat käyttäjät kirjataan ulos. Tätä voitaisiin muokata niin, että käyttäjää varoitettaisiin tietyn ajan kuluessa inaktiivisuudesta, ennen kuin hänet kirjataan ulos. Myöskin käyttäjän yrittäessä mennä väärään osaan hänet kirjataan tällä hetkellä ulos. Tähänkin käyttäjäystävällisempi ratkaisu voisi olla vain huomautus käyttöoikeuksien puuttumisesta uloskirjaamisen sijaan.

Ennen kuin kannataisi lisätä varsinaisia ominaisuuksia järjestelmään, järjestelmään kannattaisi tehdä jatkosuunnittelua. Järjestelmä on nimittäin muuttunut melkoisen paljon alkuperäisestä suunnitelmasta. Sen takia työn loppuvaiheessa oli syntynyt tilanteita, joita ei ollut huomioitu alkuperäisessä suunnitelmassa. Esimerkiksi mobiililaitteilla syntyvissä kuvakaappauksissa on työpalkki, joka on eri paikassa riippuen millä alustavalla kuvakaappaus on otettu. Olin saanut tehtäväksi poistaa työkalupalkin kyseisistä kuvakaappauksista pois. Sain sen lopulta tehtyä, mutta ratkaisu oli epäkäytännöllinen ja se toimi osalla laitteista mobiililaitteiden vaihtuvien kuvaleveyksien

takia. Kyseisen toiminnon saisi tehtyä helposti, jos JSON-tiedostossa olisi tietoa siitä, millä laitteella kuvakaappaus on otettu ja sisältääkö kuva työkalupalkin eli tarvitseeko kuvaa leikata.

Työn ketterän kehitysmallin ansiosta työhön on suhteellisen helppoa lisätä uusia ominaisuuksia ja siihen todennäköisesti tuleekin niitä lisää. Olisi kuitenkin hyvä muistaa suunnitella jatkuvasti työn ohella, jotta saadaan minimoitua ongelmat.

Ensimmäinen varsinaiseen työhön lisättävä ominaisuus olisi tallenna –näppäin oppilaan työkaluun. Näppäintä painaessa sivu ottaisi kuvakaappauksen oppilaan kirjasta, joka näytettäisiin opettajalle. Toinen tehtävä voisi olla lisätä opettajan työkaluun mahdollisuuden arvioida oppilaan raportteja tai kirjata niitä johonkin sähköisesti ylös.

LÄHTEET JA TUOTETUT AINEISTOT.

Black Duck Software, Inc. 2014-11-10. Apache HTTP Server [verkkoaineisto] [viitattu 2014-12-01] Saatavissa: https://www.openhub.net/p/apache/analyses/latest/languages_summary

BLANCHARD, Jay 2012-12-03. jQuery and jQuery UI: Visual QuickStart Guide. San Francisco: Peachpit Press. [verkkoaineisto] [viitattu 2014-08] Saatavissa: http://proquest.safaribooksonline.com.ezproxy.savonia-amk.fi/9780133136197

BUYTAERT, Dries 2010-03-10. The history of MySQL AB [verkkoaineisto] [viitattu 2014-11-28] Saatavissa: http://buytaert.net/the-history-of-mysql-ab

Euroopan Unioni, 2010-01-21. Mergers: Comission clears Oracle's proposed acquisition of Sun Microsystems [verkkoaineisto] [viitattu 2014-12-01] Saatavissa: http://europa.eu/rapid/press-release_IP-10-40_en.htm

FERGUSON, Russ, HEILMANN, Christian 2013-01-26. Beginning JavaScript with DOM Scripting and Ajax, Second Edition. [verkkoaineisto] [viitattu 2014-12-08] Saatavissa: http://proquest.safaribooksonline.com.ezproxy.savonia-amk.fi/9781430250920?uicode=savonia

GILMORE, Jason 2005. PHP & MySQL – Tehokas hallinta. (Suom. Arto Kuvaja) 1. painos. Jyväskylä: Gummerus Kirjapaino Oy.

GRANT, Rickford, BULL, Phil 2012-07-22. Ubuntu Made Easy. San Francisco: No Starch Press. [verkkoaineisto] [viitattu 2014-12-08] Saatavissa: http://proquest.safaribooksonline.com.ezproxy.savonia-amk.fi/9781457169564?uicode=savonia

HAMILTON, Coman 2014-09-08. The top 10 SQL and NoSQL databases [verkkoaineisto] [viitattu 2014-12-01] Saatavissa: http://jaxenter.com/the-top-10-sql-and-nosql-databases-108072.html

HASAN, Mahabubul 2013-08-26. Instant Zend Framework 2.0. Birmingham: Packt Pub. [verkkoaineisto] [viitattu 2014-12-08] Saatavissa: http://proquest.safaribooksonline.com.ezproxy.savonia-amk.fi/9781782164128?uicode=savonia

HAIKALA, Ilkka ja MIKKONEN, Tommi 2011. Ohjelmistotuotannon käytännöt. 12. painos. Hämeenlinna: Kariston Kirjapaino Oy.

KOISTINEN, Jussi 2013. [Verkkoaineisto]. Sijainti: Kuopio: Savonia-ammattikorkeakouolun Moodle [verkko-oppimisympäristö]. 12B ETP3300 Suunnittelumallit –kurssi.

KORPELA, Jukka 2011. HTML5 – uudet ominaisuudet. 1. painos. Porvoo: Docendo

KORPELA, Jukka 2008. CSS Verkkosivujen muotoilussa. 1. painos. Porvoo: Docendo

LEMOS, Manuel 2014-08-04. PHP 7 Features and Release Date [verkkoaineisto] [Viitattu 2014-27-11.] Saatavissa: http://www.phpclasses.org/blog/post/242-PHP-7-Features-and-Release-Date.html

METHVIN, Dave 2014-05-01. jQuery 1.11.1 and 2.1.1 Released [verkkoaineisto] [viitattu 2014-11-28] saatavissa: http://blog.jquery.com/2014/05/01/jquery-1-11-1-and-2-1-1-released/

Mozilla Developer Network 2014-11-04. HTML5 element list [verkkoaineisto] [viitattu 2014-11-28] Saatavissa: https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5/HTML5_element_list

Oracle Corporation 2014-12-01. Changes in MySQL 5.6.22 (2014-12-01) [verkkoaineisto] [viitattu 2014-11-28] Saatavissa http://dev.mysql.com/doc/relnotes/mysql/5.6/en/news-5-6-22.html

O'PHINNEY, Matthew Weier 2014-09-17. Zend Framework 1.12.9, 2.2.8, and 2.3.3 Released! [verkkoaineisto] [Viitattu 2014-11-28] Saatavissa: http://framework.zend.com/blog/zend-framework-1-12-9-2-2-8-and-2-3-3-released.html

ROMY, Ben 2012-05-04. Why HTML is Not a Programming Language. [verkkoaineisto] [viitattu 2014-12-01] saatavilla: http://infospace.ischool.syr.edu/2012/04/05/why-html-is-not-a-programming-language/

Solid IT 2014. MySQL System Properties. [verkkoaineisto] [viitattu 2014-12-01] Saatavissa: http://db-engines.com/en/system/MySQL

The Apache Software Foundation 2012. Why was the name 'Apache' chosen? [verkkoaineisto] [viitattu 2014-11-28] Saatavissa: http://www.apache.org/foundation/faq.html#name

The Apache Software Foundation 2014. How Apache Came to Be [verkkoaineisto] [viitattu 2014-11-28] Saatavissa:
http://httpd.apache.org/ABOUT_APACHE.html

The jQuery Foundation 2014. [verkkoaineisto] [viitattu 2014-12-01] Saatavissa: https://jquery.com

The JSON Group 2014. Introducing JSON [verkkoaineisto] [viitattu 2014-11-28] Saatavissa: http://json.org/

The PHP Group 2014. [verkkoaineisto] [viitattu 2014-11-28] Saatavissa: http://php.net/

ULMAN, Larry 2012-03-17. Modern JavaScript: develop and design. Berkeley [Kalifornia]: Peachpit Press [verkkoaineisto] [viitattu 2014-12-08] Saatavissa:
http://proquest.safaribooksonline.com.ezproxy.savonia-amk.fi/9780132905848?uicode=savonia

W3Techs 2014-12-01a. Usage of JavaScript libraries for websites [verkkoaineisto] [viitattu 2012-12-01] Saatavissa: http://w3techs.com/technologies/overview/javascript_library/all

W3Techs 2014-12-01b. Usage statistics and market share of Apache for websites [verkkoaineisto] [viitattu 2014-12-01] Saatavissa: http://w3techs.com/technologies/details/ws-apache/all/all

Weego Software Oy. 2014-20-03. Pasi Ollikainen [sähköpostikeskustelu]

World Wide Web Consortium 2014-10-28. Open Web Platform Milestone Achieved with HTML5 Recommendation [verkkoaineisto] [viitattu 2014-28-11] saatavissa: http://www.w3.org/2014/10/html5-rec.html.en

World Wide Web Consortium 2014. HTML & CSS [verkkoaineisto] [viitattu 2014-11-28] Saatavissa: http://www.w3.org/standards/webdesign/htmlcss#whatcss

Worx International Inc. 2014. Basic Example using Mail() [verkkoaineisto] [viitattu 2014-11-28] Saatavissa: http://phpmailer.worxware.com/index.php?pg=examplebmail

Zend Technologies Ltd. 2014a. About [verkkoaineisto] [viitattu 2014-12-01] Saatavissa: http://framework.zend.com/about/

Zend Technologies Ltd. 2014b. New BSD License [verkkoaineisto] [viitattu 2014-12-01] Saatavissa: http://framework.zend.com/license/

Zend Technologies Ltd. 2014c. Archives [verkkoaineisto] [viitattu 2014-12-01] Saatavissa: http://framework.zend.com/downloads/archives