Tampereen ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Jari Kyllönen

Opinnäytetyö

WWW-sivuston modulaarinen rakenne

Case: VirtuaaliViipuri

Työn ohjaaja	Petri Heliniemi, FM
Työn tilaaja	Tampereen ammattikorkeakoulu, Yhteyshenkilö: projektipäällikkö Harri Miettinen

Tampere 11/2009

Tampereen ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma

Tekijä	Jari Kyllönen
Koulutusohjelma	Tietojenkäsittely
Opinnäytetyön nimi	WWW-sivuston modulaarinen rakenne Case: VirtuaaliViipuri
Sivumäärä	38
Valmistumisaika	marraskuu 2009
Työn ohjaaja	Petri Heliniemi
Työn tilaaja	Tampereen ammattikorkeakoulu

TIIVISTELMÄ

Opinnäytetyö käsittelee Tampereen ammattikorkeakoulun ylläpitämän VirtuaaliViipuri-projektin WWW-sivuston toteutusta Zend Framework sovelluskehyksen pohjalta sekä eräitä yleisiä web-selain ohjelmointitekniikoita.

Toimeksiannon tavoitteena on suunnitella ja toteuttaa nykyiseen WWW-sivustoon uusi rakenne, joka noudattaa modulaarista toimintaperiaatetta. Kehitettävän sivustorakenteen vaatimuksina ovat sivuston nopeus, skaalautuvuus ja ylläpidettävyys. Ulkoasullisia seikkoja ei tässä toimeksiannossa oteta huomioon.

Sivuston uusi rakenne toteutetaan vapaan lisenssin Zend Framework sovelluskehyksen avulla, joka kasvattaa jatkuvasti suosiotaan kehyksenä WWW-sivustokehityksessä. Zend Framework perustuu MVC-ohjelmistoarkkitehtuuriin (model-view-controller, malli-näkymä-ohjain) ja se sisältää esimerkiksi valmiita luokkia, joita ohjelmoinnissa hyödynnetään kehitettävän sivuston vaatimien tarpeiden mukaan.

VirtuaaliViipuri-sivusto koostuu WWW-sovelluksesta, MySQL-tietokannasta ja hallintatyökalusta. Sivusto toteutetaan avoimena, jolloin siellä esiintyvä tieto on kaikkien nähtävillä lukuun ottamatta hallintatyökalua, johon vain sivuston ylläpitäjällä on käyttöoikeus. Hallintatyökalun avulla voidaan hallita tietokantaan tallennettua tietoa.

Avainsanat: PHP, WWW-selainohjelmointi, Zend Framework, modulaarinen rakenne, MVC-ohjelmistoarkkitehtuuri.

TAMK University of Applied Sciences
Business Information Systems

Writer	Jari Kyllönen
Thesis	Modular Structure of a Web Site. Case: VirtuaaliViipuri
Pages	38
Graduation time	November 2009
Thesis Supervisor	Petri Heliniemi
Co-operating Company	TAMK University of Applied Sciences

ABSTRACT

This Bachelor's thesis concentrates on the VirtualViipuri project's website hosted by TAMK University of Applied Sciences. The website uses Zend Framework which is an object-oriented web application framework. The thesis also concentrates on some common browser programming techniques.

The aim of the thesis work is to design and implement for the current website a new structure which follows the operational principles of a modular structure. The requirements of the developed website are downloading speed, scalability and maintenance. The appearance issues of the website are left out in this thesis work.

The new structure of the website is developed with Zend Framework, an open source PHP 5 implementation tool. Zend Framework is based on the MVC-architecture (model-view-controller) which is an architecturical pattern used in software engineering. For example, it contains prepared classes which are utilised in the development of the website when necessary.

VirtualViipuri website consists of the website itself, a MySQL database and an administration site. The website is developed so that the contents will be accessible by all users, except for the administration site which is accessible only to the administrator. The administration site is developed for the management of the data saved in the database.

Key words PHP, web programming, Zend Framework, modular structure of a website, MVC software architecture.

Sisällysluettelo

SANASTO **5**

1 JOHDANTO **6**

2 KÄYTETYT TEKNIIKAT **10**

2.1 XHTML 10

2.2 PHP 11

2.3 MYSQL 13

2.4 CSS 14

3 KEHITYSYMPÄRISTÖ **16**

3.1 XAMPP 16

3.1.1 Apache HTTP Server 17

3.1.2 PHPMyAdmin 18

3.2 Zend Framework 20

3.2.1 MVC-ohjelmistoarkkitehtuuri 21

3.2.2 Zend Frameworkin rakenne ja toiminta 22

4 VIRTUAALIVIIPURI-SIVUSTO **25**

4.1 Tietokannan rakenne 26

4.2 Hakemistorakenne 28

4.3 Tiedostorakenne 29

4.4 Sivuston toteutus 31

4.5 Sivuston toiminta 34

5 YHTEENVETO **36**

LÄHTEET **38**

Sanasto

HTML (HyperText Markup Language) on World Wide Webin sivunkuvauskieli.

XML (eXtensible Markup Language) on rakenteellinen kuvauskieli, jolla tiedon merkitys on kuvattavissa tiedon sekaan.

XSLT (Extensible Stylesheet Language Transformations) on XML-pohjainen merkintäkieli XML-tiedostojen muunnoksiin.

XPATH (XML Path Language) on kieli XML-dokumenttien osien osoittamiseen ja tiedon luontiin.

XHTML (eXtensible Hypertext Markup Language) on HTML:stä kehitetty WWW-sivujen merkintäkieli, joka täyttää XML:n muotovaatimukset.

Komentosarjakieli eli skriptikieli on kieli, jolla kirjoitetaan komentosarjoja eli skriptejä, joiden avulla automatisoidaan erilaisia toimintoja.

PHP (PHP Hypertext Preprocessor) on palvelimella suoritettava, tulkattava komentosarjakieli.

PHP-tulkki kääntää ja suorittaa PHP-tiedoston sisältämän koodin.

Perl (Practical Extraction and Report Language) on tulkattava skriptimäinen ohjelmointikieli.

Java on laaja ohjelmistoalusta, johon kuuluu mm. oliopohjainen ohjelmointikieli luokkakirjastoineen.

SGML (Standard Generalized Markup Language) on kieli, jonka avulla voidaan määritellä dokumenttien merkintäkieliä.

SQL (Structured Query Language) on rakenteinen kyselykieli, jota käytetään relaatiotietokantajärjestelmien kanssa kommunikointiin.

MySQL on tehokas SQL-tietokantojen hallintajärjestelmä.

CSS (Cascading Style Sheets) on kaskadinen tyyliohjejärjestelmä, joka on kehitetty erityisesti WWW-dokumentteja varten.

GPL (General Public License) on vapaa ohjelmistolisenssi. Sen tarkoitus on taata oikeus kopioida, muuttaa ja jakaa edelleen ohjelmia ja niiden lähdekoodia.

1 Johdanto

Tässä luvussa selvennetään tarkemmin opinnäytetyön ja toimeksiannon taustoja, tavoitteita sekä tehdään yleiskatsaus työn kannalta olennaisiin lähteisiin.

Aihe ja toimeksiantaja

Opinnäytetyöni sijoittuu aiheeltaan WWW-sivustojen toteutukseen ja siihen käytettäviin toteutustekniikoihin. WWW-sivujen rakenne on olennainen osa sivuston suunnitteluprosessia, jotta sivusto saadaan parhaalla mahdollisella tavalla vastaamaan sen käyttötarkoitusta. WWW-sivuston rakenne tulee olla johdonmukainen, jotta sivuston ympäristö on laajennettavissa sen toiminnallisuuden häiriintymättä.

Modulaarinen rakenne säästää työmäärissä, sillä päivitystilanteissa muutokset on tarpeen tehdä usein vain yhteen paikkaan, josta ne vaikuttavat koko sivustoon. Huolellisesti suunnitellun sivustorakenteen ansiosta sivujen päivitys sujuu vaivattomammin, sivuston elinikä pitenee ja WWW-sivut ovat tulevaisuudessa laajennettavissa ilman, että sivuston runkoon joudutaan tekemään suuria muutoksia.

Tulevaisuudessa yhä useammat WWW-sivustot käyttävät MVC-pohjaista (model-view-controller, malli-näkymä-ohjain) sivustorakennetta, joten rakenne toteutettiin edellä mainittua arkkitehtuuria käyttäen. MVC-pohjaisen ohjelmistoarkkitehtuurin etuna on sen skaalautuvuus, joka mahdollistaa entistä paremman laajennettavuuden ja uusien ominaisuuksien lisäämisen puuttumatta sivuston runkoon. Arkkitehtuurin tyylin mukaan uudet ominaisuudet eli moduulit rakennetaan itsenäisesti, jonka jälkeen ne liitetään osana muuta sovelluskokonaisuutta. MVC-arkkitehtuurin riskinä on, että toimintaperiaatteesta voi tulla vaikeammin ymmärrettävä, koska sovelluskokonaisuus hajotetaan käytännössä kolmelle eri tasolle.

Toimeksiannossa käytin vapaan lisenssin sovelluskehystä nimeltään Zend Framework, joka on PHP-ohjelmointikieleen pohjautuva, MVC-mallia käyttävä valmis luokkakirjasto. Zend Framework sisältää valmiita luokkia, joten kaikkea ei tarvitse ohjelmoida itse. Tämän ansiosta valmiiden luokkakirjastojen käyttö säästää myös aikaa WWW-sovellusten kehitystyössä. Sivuston kuvauskielenä toimii XHTML ja ohjelmointikielenä PHP. Tiedon varastoimiseen sivusto käyttää MySQL-relaatiotietokantaa. Ulkoasumääritykset toteutetaan CSS-tekniikkaa käyttäen.

Opinnäytetyössä kuvattava WWW-sivusto on osa Tampereen ammattikorkeakoulun ylläpitämää VirtuaaliViipuri-projektia, jonka tavoitteena on luoda 3D-mallinnuksella WWW-ympäristöön virtuaalinen Viipurin kaupunki sellaisena suomalaisena kaupunkina kuin se oli syyskuussa vuonna 1939. Projekti käynnistyi vuonna 2003. Tähän mennessä projektin WWW-sivuilla on ollut yli 935 000 kävijää.

Lähdeaineistona VirtuaaliViipuri-projektissa ovat intendentti Juha Lankisen asema- ja julkisivupiirustukset jokaisesta Viipurin keskustan korttelista, jotka hän on tehnyt Viipurin pienoismallia varten sekä hänen kokoelmastaan olevat valokuvat. Lankisen suunnittelemassa Viipurin kaupungin pienoismallissa on yli 3500 rakennusta. Koska projekti on laaja ja koko ajan kehittyvä, on ehdottoman tärkeää, että sivusto on päivitettävissä helposti ja nopeasti uusien mallinnuksien syntyessä.

Tavoite

Toimeksiantona oli suunnitella ja toteuttaa VirtuaaliViipuri-projektin WWW-sivuille uusittu modulaarista toimintaperiaatetta vastaava rakenne. Tällä hetkellä käytössä oleva sivusto sisältää tarpeetonta koodin monistamista ja sen rakenne on toteutettu rakenteella, joka ei ota huomioon tulevaisuuden kehitystarpeita. Tämä tarkoittaa, että sivuston kehittyessä koodi muuttuu helposti epäselväksi. Rakenteen myötä myös tietokannan rakenne ja taulujen väliset suhteet suunniteltiin uudelleen tarvittaessa.

VirtuaaliViipuri-sivuston rakenteen muuttaminen modulaariseksi toi näin ollen hyötyä sivuston ylläpidolle ja kehitykselle. Modulaarisen rakenteen ansiosta sivuston päivitykset nopeutuivat, ja esimerkiksi uusien ulkomaisten kielten lisääminen sivustoon saadaan tehtyä tavallista nopeammin ja vaivattomammin. Opinnäytetyössäni en puutu ulkoasullisiin seikkoihin, vaan työni perustuu puhtaasti rakenteeseen ja sen toiminnallisuuteen.

Toteutuksen testaus suoritettiin työn ohessa ja lopputestauksessa. Työn ohessa testaamisella pyrittiin selvittämään virheet ja mahdolliset häiriötilanteet jo uuden ohjelmakoodin luontivaiheessa. Näin varmistettiin esimerkiksi eri funktioiden toimivuus sivustokehityksen aikana. Tätä voidaan kutsua ns. metodi/luokkatestaukseksi. Toteutuksen aikana ei siis kehitetty mitään erillistä testialustaa testauksen suorittamiseksi, vaan testauksen alustana toimi sivuston kehitysalusta eli paikallinen WWW-palvelin. Lopputestauksen yhteydessä varmistettiin sivuston toimivuus osa kerrallaan.

Lähdeaineisto

Tutkintotyön lähteinä on käytetty useita perinteisiä kirjoja, mutta myös yleisesti käytettyjä verkkolähteitä, joiden tietoa voidaan pitää luotettavana. Vaikka Zend Framework onkin suhteellisen uusi kehysympäristö, on sitä käsittelevää kirjallisuutta jo saatavilla, vaikkakin suppeasti. Kukaan ei ole vielä ehtinyt suomentamaan jo julkaistuja teoksia, joten ne ovat englanninkielisiä.

Julkaistuista teoksista mainittakoon Quentin Zervaasin vuonna 2007 julkaisema Practical Web 2.0 Applications with PHP ja Rob Allenin, Nick Lon sekä Steven Brownin kokoama teos Zend Framework in Action, jotka molemmat ovat erittäin varteenotettavia apuvälineitä Zend Frameworkiin tutustuttaessa. Teokset esittelevät kehysympäristön toimintaa MVC-arkkitehtuuriin perustuen ja antavat havainnollistavia esimerkkejä itse ohjelmointiin.

Molemmat kirjat avaavat kehysympäristön toimintaa hieman pintaa syvemmältäkin, mikä helpottaa sovelluskehyksen toiminnan ymmärtämistä. Lukija pääsee myös tuottamaan itse ensimmäisen WWW-sovelluksensa Zend Frameworkilla, johon kirjat antavat selkeän ja yksityiskohtaisen opastuksen askel askeleelta.

Koska Zend Frameworkia käsittelevää painettua aineistoa on tätä raporttia kirjoitettaessa julkaistu verrattain vähän, perustuu tiedonsaanti esimerkiksi ongelmatilanteissa pääasiallisesti verkkolähteisiin. Zend Frameworkin omilla kotisivuilla on julkisesti esillä kattava manuaali, joka esittelee tarkasti esimerkkien avulla kehysympäristön komponentit ja niiden käyttötarkoituksen. Lisäksi on olemassa useita keskustelukanavia eli foorumeita, joissa muut WWW-sovelluskehittäjät käyvät aktiivista keskustelua Zend Frameworkiin liittyvistä ongelmista ja ominaisuuksista, joten keskustelukanavilta voi usein löytää vastauksia myös omiin ongelmatilanteisiin.

2 Käytetyt tekniikat

Tässä luvussa esitellään yleisesti käytettyjä WWW-sivustojen toteutustekniikoita, joita on käytetty VirtuaaliViipuri-sivuston toteutukseen. Toteutustekniikoiden esittelyssä ei ole tarkoitus perehtyä tekniikoihin syvällisesti, vaan antaa yleiskuvaa käytetyistä tekniikoista ja niiden käyttötarkoituksista. Sovelluskehittäjällä tulee olla hyvä tuntemus erilaisista tekniikoista, jotta kehitettävä WWW-sivusto voidaan toteuttaa tehokkaasti, ja jotta sen toiminta palvelisi suunniteltua tarkoitustaan.

2.1 XHTML

XHTML:n juuret ulottuvat aina 80-luvun puoliväliin asti, jolloin SGML:stä (Standard Generalized Markup Language) ryhdyttiin kehittämään HTML-merkintäkieltä. SGML on metakieli, jonka avulla voidaan määrittää muita dokumentin rakenteen merkintäkieliä. HTML:stä oli tarkoitus kehittää yhtenäinen ja helposti hallittava kieli, jota organisaatiot voisivat käyttää tiedon vaihtamiseen Internetin välityksellä.

Aluksi idea toimi kohtalaisen hyvin, mutta pian siihen liittyvät ongelmat alkoivat näkyä. HTML:stä tuli nopeasti selainkohtainen, koska eri selainten valmistajat kuuntelivat vain tiettyjen kohderyhmien tarpeita ja tiettyä selainta varten tehdyt HTML-dokumentit eivät toimineetkaan muilla selaimilla. Tämän vuoksi alkuperäinen idea HTML:n yhtenäisyydestä katosi.

XML:n (eXtensible Markup Language) synty muutti kehityksen suunnan jälleen kohti standardisoitua merkintäkieltä. 90-luvun puolivälissä alettiin kehittää standardia, jolla voitaisiin korvata raskas SGML jollain kevyemmällä ja siirrettävämmällä kielellä. Juuri XML:n keveyden ja yksinkertaisten perussääntöjen ansiosta se nousi nopeasti suureen suosioon merkintäkielenä. XML:ään perustuva uusi HTML-määritys sai eteensä X-kirjaimen ja uusi XHTML täyttää XML:n muotovaatimukset. Tällaisia ovat mm. pien-

ten kirjainten käyttö kaikissa tunnisteissa, attribuuttien pakolliset arvot sekä elementtien pakollinen sulkeminen.

Tällä hetkellä XHTML:stä on kolme versiota. XHTML 1.0 on pitkälti yhteneväinen HTML 4.01:n kanssa. XHTML 1.0:n erona ovat rajoitukset pienten kirjainten käytössä, elementtien sulkeminen dokumenteissa sekä osa ulkoasuun vaikuttavista elementeistä on osin tai kokonaan poistunut. Kieli jaetaan vielä kolmeen osakokonaisuuteen, jotka ovat Strict, Transitional ja Frameset. Strict noudattaa tiukimmin XML:n muotosääntöjä, kuten elementtien sulkemista. Transitional sisältää vielä jonkin verran muotoiluelementtejä ja Frameset hyödyntää nimensä mukaisesti kehyksiä.

XHTML 1.1 on jo selkeästi tarkemmin määritelty dokumenttityyppi, josta on poistettu ulkoasuun liittyvät määritteet kokonaan, joten ulkoasun toteutukseen tulee käyttää tyyliohjeita, kuten CSS:ää. Laajennettavuuden vuoksi kieli on jaettu moduuleihin, joista jokainen sisältää vain tietynlaisia määrityksiä. Tämä helpottaa myös erityislaitteille rakennettavien selainten toteuttamista esimerkiksi kämmentietokoneissa. XHTML Basic on edellisestä supistettu versio, joka kehitettiin nimenomaan mobiililaitteisiin ja niiden suorituskykyä ajatellen.

2.2 PHP

PHP (Hypertext Preprocessor) on HTML-dokumenttien sisään upotettu ohjelmointikieli, joka pohjautuu olennaisesti C-kieleen, mutta mukana on myös piirteitä Java-, Perl- sekä C++-kielistä. PHP on WWW-palvelimella tulkattava kieli, mikä tarkoittaa, että WWW-sivun sisällä esiintyvä PHP-koodi ajetaan joka kerta, kun WWW-palvelin lähettää sivun selaimelle. Tällaisia kieliä kutsutaan *skriptikieliksi*. PHP on avoin ohjelmointikieli, joten sen lähdekoodi on kaikille vapaasti saatavilla. Kieli laajennuksineen antaa kattavan ja vakaan pohjan nykyaikaisten verkkosovellusten rakentamiselle. PHP:n suuri suosio perustuu sen monipuolisuuteen sekä yhteensopivuuteen erilaisten dynaamisesti toimivien verkkopalveluiden asettamiin vaatimuksiin. (Rantala 2002: 12; Lerdorf, Tatroe & MacIntyre 2006)

Rasmus Lerdorf aloitti kielen kehittämisen jo vuonna 1994, joten häntä pidetään PHP:n luojana. Alun perin PHP oli pieni kokoelma komentoja, jotka helpottivat WWW-pohjaisten sovellusten tekoa. Ensimmäisestä versiosta käytettiin nimeä ”*Personal Home Page Tools*”. Seuraavaa versiota kutsuttiin nimellä PHP/FI (*Personal Home Page / Forms Interpreter*). Sittemmin PHP:tä alettiin kutsua sen nykyisellä nimellään ”*Hypertext Preprocessor*”. (Rantala 2002: 12)

Uusin vakaa versio PHP:stä on tällä hetkellä 5.2.9, mutta 5.3 on jo julkaisun kynnyksellä ja myös PHP 6:n julkaisusta on hetkittäin väläytelty positiivisia ennusmerkkejä. Kuudennessa versiossa uutena ominaisuutena on mm. täysi Unicode-tuki. Tämän hetkisessä versiossa ei kuitenkaan ole tukea useamman tavun mittaisille merkeille, mikä rajoittaa Unicode-merkistön käyttöä. Lisäksi ongelmia aiheuttavat nimiavaruudet, joille tukea ei nykyisestä versiosta löydy. Näin ollen kaikki funktiot ovat yhdessä nimiavaruudessa. Nimiavaruuksien tuki on luvattu versioon 5.3. (php.net - PHP: Hypertext Preprocessor 2009)

PHP-tiedostojen luomiseen tarvitaan jokin editori, jolla tavallisten ASCII-tekstitiedostojen tekeminen on mahdollista. Lisäksi PHP-ohjelmien suorittamiseen tarvitaan WWW-palvelinohjelmisto sekä sen kanssa yhdessä toimiva PHP-tulkki. WWW-hotellipalvelua valitessa täytyy näin ollen selvittää, tukeeko kyseinen palvelu PHP:n hyödyntämistä. Toinen vaihtoehto on käyttää omaa palvelinta, jolle PHP-tulkin asentaminen on kohtuullisen helppoa. (Rantala 2002: 15)

Kun WWW-sivulla oleva PHP-koodi suoritetaan, näkyy selaimessa vain ja ainoastaan sovelluksen tulostus. PHP-koodi on kuitenkin nähtävissä palvelimella olevista tiedostoista, joita voi lukea esimerkiksi jonkin FTP-ohjelman välityksellä. Koodi kirjoitetaan suoraan HTML-muotoilujen sisään. PHP-koodin aloitus ilmoitetaan merkinnällä *<?php*, jonka jälkeen kirjoitetaan haluttu koodi. PHP-koodi lopetetaan merkinnällä *?>* (Esimerkkikoodi 1). Kun koodi ajetaan palvelimella ja palautetaan selaimeen, näkyy näytöllä ainoastaan teksti ”Esimerkki selaimessa näkyvästä tekstistä”. (Heinisuo & Rauta 2007: 13)

```
<h2>
<?php
    echo "Esimerkki selaimessa näkyvästä tekstistä.";
?>
</h2>
```

Esimerkkikoodi 1. PHP-koodi upotetaan suoraan XHTML-elementtien sisään <?php sekä *?> merkinnöillä*

2.3 MySQL

MySQL on laajimmin käytetty avoimen lähdekoodin relaatiotietokantojen hallintajärjestelmä tiedon hakemisessa ja varastoimisessa. MySQL:ää käyttävät niin yksityiset henkilöt kuin suuret yrityksetkin, kuten Friendster, joka on maailmanlaajuinen yli 70 miljoonan jäsenen kattava verkkoyhteisö. Friendsteriin tehdään päivittäin yli 1,5 miljardia kyselyä. MySQL on saavuttanut laajan suosion ja sen suuret käyttäjämäärät perustuvat maksuttoman GPL (*General Public License*) -lisenssin lisäksi nopeuteen, siirrettävyyteen ja yhteensopivuuteen minkä tahansa ohjelmointikielen kanssa. (php.net - PHP: Hypertext Preprocessor)

MySQL:n syntyhistoria juontaa juurensa jo vuoteen 1979, mutta varsinainen kehitystyö alkoi vuonna 1995, kun Michael Widenius ja David Axmark lähtivät kehittämään täysin uutta tietokantaohjelman moottoria. Tietokannan ensimmäinen versio julkaistiin vuonna 1996 ja nimettiin Wideniuksen My-tyttären mukaan MySQL:ksi. Uusin versio tästä tietokannanhallintajärjestelmästä on tällä hetkellä 5.0. MySQL:n kehityksestä vastaa ruotsalainen MySQL AB, jonka Sun Microsystems osti miljardilla dollarilla (700 miljoonaa euroa) tammikuussa 2008. (Moisio 2008)

MySQL on monisäikeinen palvelin, mikä tarkoittaa, että palvelimella aloitetaan aina uusi palvelinprosessi, kun yhteys tietokantaan muodostetaan. Yhteydet palvelimeen eivät jaa prosesseja, joten esimerkiksi, jos prosessi päättyy odottamattomasti tai se ylikuormittaa muistia kohtuuttoman paljon, vain yksi prosessi sulkeutuu, eikä koko palve-

lin kaadu. Tämä lisää MySQL:n nopeutta. MySQL:ää voidaan käyttää niin Unix- kuin myös Windows-käyttöjärjestelmässä. (Meloni 2003: 12)

Kyselykielenä toimii standardoitu SQL-kieli. Tietokantaa ei käsitellä suoraan PHP-sovelluksesta käsin, vaan PHP:n kautta muodostetaan yhteys MySQL-tietokantapalvelimeen. PHP lähettää palvelimelle komentoja SQL-kielellä ja MySQL palauttaa tietokannan tauluihin tallennettua tietoa PHP-sovellukselle (Esimerkkikoodi 2). (Heinisuo & Rauta 2007: 39)

```
<?php
    // avataan yhteys tietokantaan
    $connect = mysql_connect("localhost", "username", "password");
    // valitaan käytettävä tietokanta
    mysql_select_db("esimerkkikanta", $connect);
    // luodaan SQL-lause
    $sql = "SELECT * FROM esimerkkitaulu";
    // suoritetaan SQL-lause
    $result = mysql_query($sql, $connect);
    // tulostetaan kyselyn tulos
    echo $result;
?>
```

Esimerkkikoodi 2. Yksinkertainen esimerkki, jossa PHP:n kautta muodostetaan yhteys tietokantaan, suoritetaan SQL-kysely ja tulostetaan kyselyn tulos näytölle.

2.4 CSS

Kaskadiset tyyliohjeet (CSS) on varsin yksinkertainen mekanismi erilaisten tyylien, kuten esimerkiksi fontin tai värin lisäämiseksi WWW-dokumentteihin (Esimerkkikoodi 3). Tyyliohjeet määräävät siis sen, millä tavalla sisältö esitetään WWW-sivuilla. CSS:llä annetut säännöt määrittävät, millaisena dokumentti näkyy selaimessa. Säännöt eivät ole ehdottomia, vaan käyttäjä voi halutessaan olla noudattamatta niitä. CSS:ää käytetään erityisesti (X)HTML:n rinnalla, mutta sitä voidaan yhtä hyvin käyttää muidenkin rakenteellisten dokumenttien, kuten XML (eXtensible Markup Language) tai SVG (Scalable Vector Graphics), kanssa. (w3.org - CSS: Cascading Style Sheets 2009)

CSS-kielien määritelmiä on jo vuodesta 1994 ylläpitänyt ja aktiivisesti kehittänyt World Wide Web Consortium (W3C). Ensimmäinen versio CSS:stä julkaistiin vuonna 1996. Vuonna 1998 julkaistiin CSS2, jonka tuki eri selaimille oli aluksi varsin heikko. Tuorein versio kulkee nimellä CSS2.1, jota laajimmin käytettyjen selainten, kuten Internet Explorer, Mozilla Firefox sekä Opera, uusimmat versiot tukevat hyvin. Seuraavaksi julkaistavaksi odotetaan CSS:n versiota 3. CSS:n lisäksi W3C on työstänyt myös muita W3C-tyylisuosituksia, kuten XPath ja XSLT.

```
h1{
font-style:italic;
color:red;
background:black;
}
```

Esimerkkikoodi 3. Tyyliohjeet voisivat näyttää .css-tiedostossa esimerkiksi tältä. Ensimmäisen tason otsikoiden tyyli on italic, väriltään punaisia ja taustavärinä on musta.

3 Kehitysympäristö

Dynaamisesti toimivat WWW-sivut koostuvat komponenteista, jotka kommunikoivat toistensa kanssa erilaisten tulkkien välityksellä. Käytettävät komponentit ja tulkit määräytyvät sivuston toiminnan ja käyttötarkoituksen mukaan. Jotta valmiit WWW-sivut voidaan julkaista Internetissä, tarvitaan sivujen säilytyspaikaksi WWW-palvelin, tietokanta tiedon tallettamiseen sekä joukko ohjelmointitulkkeja tiedon välittämiseen.

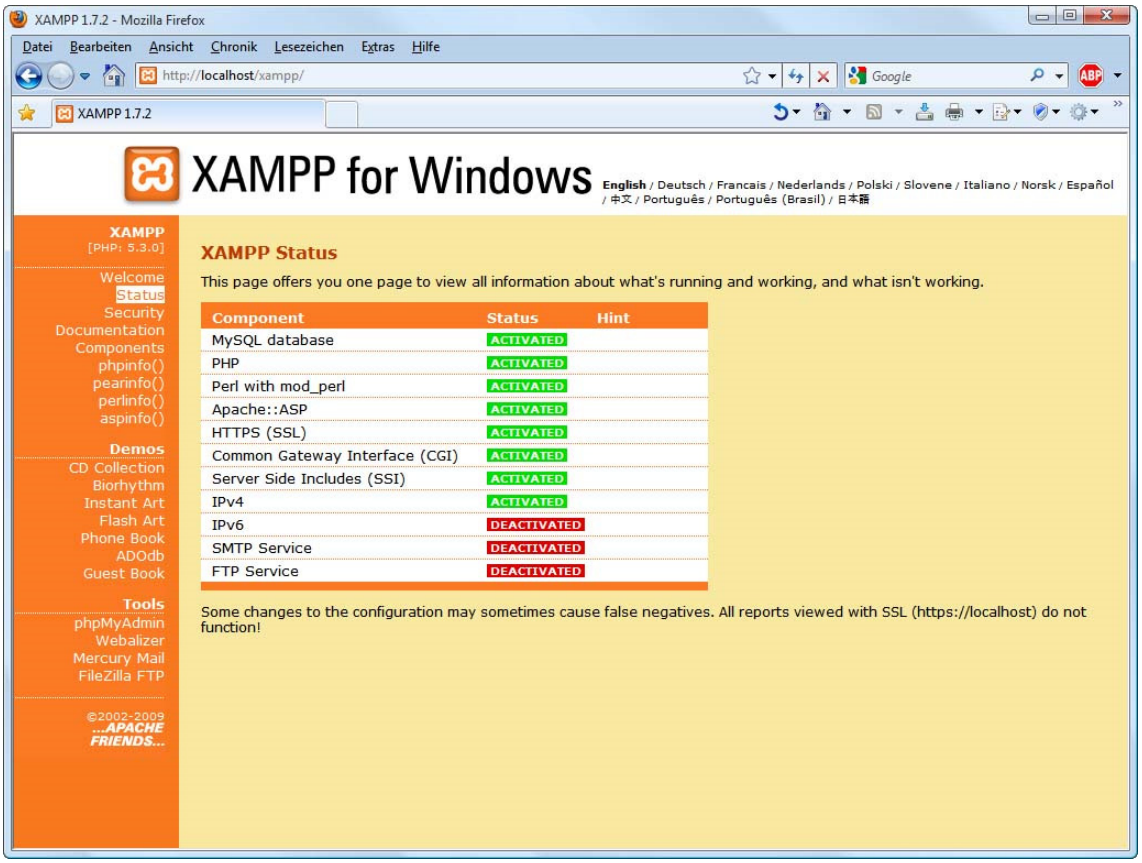
WWW-sivujen kehittäjät päätyvät usein käyttämään erilaisia ohjelmistokokonaisuuksia, joiden avulla perustetaan oikeata palvelinta simuloiva paikallinen WWW-palvelin, jonka alla voidaan ajaa dynaamisia WWW-sivustoja ilman, että sivustoa tarvitsee viedä Internetiin. Yleisimmin tällaisia ohjelmistoja käytetään ainoastaan WWW-sivujen kehittämiseen ja testaamiseen johtuen Apachen heikosta tietoturvasta Windows-ympäristössä.

3.1 XAMPP

XAMPP on ilmainen, vapaasti käytettävä WWW-sivujen kehittämiseen ja testaamiseen kehitetty ohjelmistokokonaisuus. XAMPP on saavuttanut suuren suosion erityisesti aloittelevien WWW-sivujen kehittäjien keskuudessa sen nopean asennuksen ja helppokäyttöisyyden ansiosta. Perusasennuspaketti sisältää Apache-pohjaisen WWW-palvelimen, MySQL-tietokannan, dynaamisten WWW-sovellusten toteutukseen tarvittavat ohjelmointitulkit sekä joitakin helppokäyttöisiä työkaluja sivuston kehittämisen ja ylläpitämisen tueksi.

XAMPP on saatavilla Windowsille, Linuxille, Solarikselle sekä Mac OS X:lle. Ohjelmiston asennus sujuu pääosin graafisen asennusohjelman avulla, joskin osa komponenteista täytyy asentaa edelleen komentoriviltä käsin. Muita suosittuja palvelinohjelmistokokonaisuuksia ovat WAMP, joka on kehitetty Windows-käyttöympäristöön ja vastaavasti LAMP, jonka alustana toimii Linux. Asennuksen jälkeen käyttäjälle avautuu pää-

sivu (Kuva 1), jonka kautta käyttäjä voi tarkastella eri komponenttien tilaa ja asennuspakettiin sisältyvien työkalujen avulla hallita esimerkiksi tietokantaa.



Kuva 1. XAMPP-komponenttien tila. Vasemmalla olevan valikkopalkin kautta käyttäjä pääsee hallitsemaan esimerkiksi tietokantojaan phpMyAdmin-työkalulla. (Kuvankaappaus XAMPP for Windows)

3.1.1 Apache HTTP Server

Apache HTTP Server on avoimeen lähdekoodiin perustuva HTTP-palvelinohjelma, joka on Apache Software Foundationin tunnetuin tuote. HTTP-palvelin on saatavilla useille käyttöjärjestelmille, kuten Windowsille ja Linuxille. Lisäksi se on integroitu Mac OS X:lle. Apachen kehityskaari syntyi NCSA:n httpd-palvelinohjelmistosta, johon lisättiin joukko päivityksiä ja korjauksia, mutta Apache 2.0-version myötä se kirjoitettiin kokonaan uudelleen. (httpd.apache.org - The Apache HTTP Server Project 2009)

Tuorein versio Apache:sta on vuoden vuonna 2009 julkaistu 2.2.14, mutta versioita 2.0 ja 1.3 käytetään silti vielä laajasti. Apache tukee ainoastaan staattisten tiedostojen jakamista HTTP-protokollan yli, mutta sen ydintä voidaan täydentää useilla moduuleilla, joka mahdollistaa palvelimen muokkaamisen omien käyttötarpeiden mukaiseksi.

Palvelin ottaa vastaan HTTP-pyyntöjä TCP/IP-verkosta ja vastaa niihin. Pyyntö voi palauttaa esimerkiksi HTML-dokumentin tai yleensä minkä tahansa tiedoston, mukaan lukien myös virheen. Staattiset dokumentit ovat tiedostoja, jotka lähetään sellaisenaan palvelimelta asiakaskoneelle. Dynaamiseksi dokumentin tekee se, että tiedosto luodaan ohjelmallisesti palvelimella asiakaskoneelle lähetettäväksi. Palvelin voi lisäksi asettaa asiakasohjelmalle pienen määrän dataa, jonka asiakasohjelma palauttaa takaisin palvelimelle, kun haetaan seuraavaa dokumenttia. Näitä kutsutaan evästeiksi. Evästeiden käyttö mahdollistaa istuntojen seuraamisen sekä käyttäjän yksilöimisen.

Apache on pitänyt hallussaan Internetin suosituimman HTTP-palvelimen titteliä jo vuodesta 1996 lähtien. Apachen pitkäkestoisesta suosiosta kertoo myös Netcraftin kesäkuussa vuonna 2009 julkaisema tutkimus, jonka mukaan Apachella on peräti n. 50 prosentin osuus kaikista Internetissä toimivista palvelinohjelmistoista. Toiseksi suosituimman palvelinohjelmiston nimeä kantaa Microsoftin IIS. (netcraft.com - June 2009 Web Server Survey 2009)

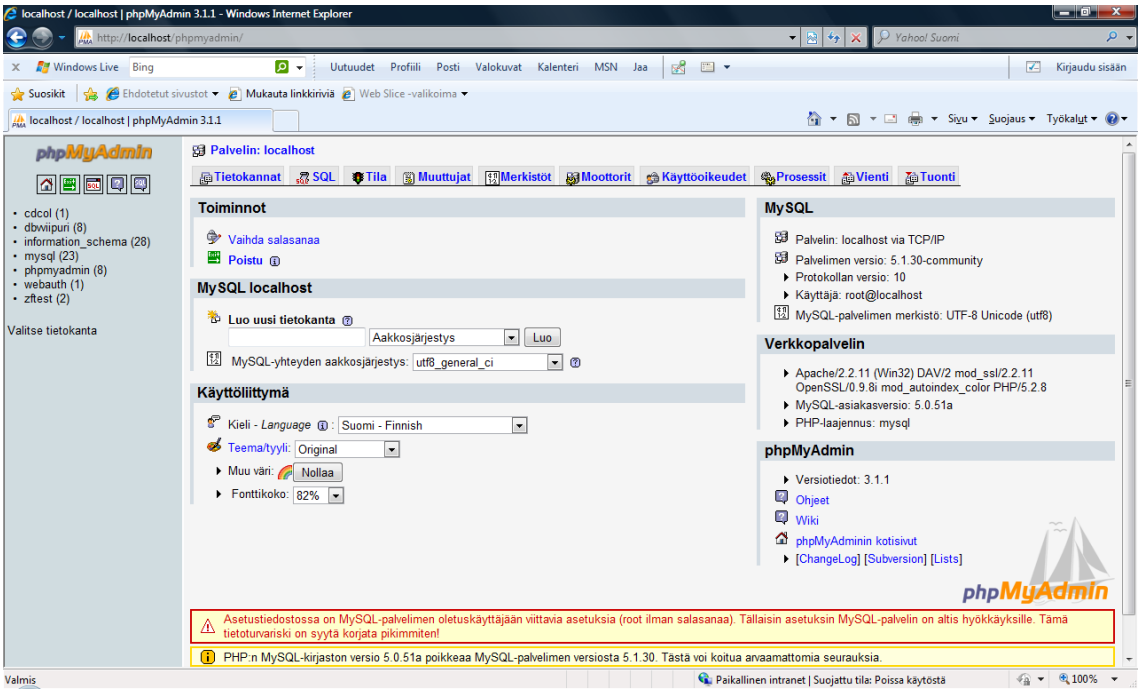
3.1.2 phpMyAdmin

PhpMyAdmin on selainpohjainen hallintatyökalu MySQL-tietokannan hallintaan. Työkalu on saatavilla maksuttomalla GPL-lisenssillä ja se kuuluu osana XAMPP:n perusasennuspakettia. PhpMyAdminin isänä pidetään Tobias Ratschilleria, joka aloitti työkalun kehittämisen erään toisen samankaltaisen tietokannan hallintatyökalun inspiroimana vuonna 1998. Ajanpuutteen vuoksi Ratschiller kuitenkin luopui kehitystyöstä vuonna 2000, minkä jälkeen kolme ohjelmistokehittäjää, Olivier Müller, Marc Delisle sekä Loïc Chapeaux, rekisteröivät ohjelmiston kehityksen koordinoinnin SourceForge.net-

sivustolle. Työkalun kehitystyö kulkee nykyisin nimellä *phpMyAdmin Project*. (phpmyadmin.net - phpMyAdmin 2009)

Selainpohjainen käyttöympäristö tekee työkalusta alustariippumattoman ja phpMyAdmin on saatavilla 57 eri kielelle mukaan luettuna suomi. Työkalu helpottaa tietokannan ylläpitoa ja sen etuna on monipuolinen hallinta, kuten mihin tahansa tietokannan tauluun, kenttään tai riviin kohdistuvat lisäykset, muokkaukset ja poistot. Työkalun käyttöliittymä on graafinen (Kuva 2), mutta sen kautta käyttäjän on myös mahdollista suorittaa SQL-lauseita komentorivipohjaisesti suoraan tietokantaan. Lisäksi phpMyAdminilla voidaan määritellä tietokannan käyttäjät ja käyttöoikeudet.

Version 3.0.0 jälkeen phpMyAdmin vaatii toimiakseen vähintään PHP 5.2 sekä MySQL 5 versiot. Työkalun 2.x versiot ovat kuitenkin vielä käytettävissä ja niitä ylläpidetään edelleen. Tuorein julkaisu phpMyAdminista on versionumeroltaan 3.2.1, joka julkaistiin elokuussa vuonna 2009. (phpmyadmin.net - phpMyAdmin 2009)



Kuva 2. PhpMyAdminin käyttö on helppoa. Päänäkymässä on tietoa mm. käytettävästä palvelimesta ja vasemmassa palkissa listaus tietokannoista palvelimella. (Kuvankaappaus PHPMyAdmin)

3.2 Zend Framework

Zend Framework on olio-pohjainen WWW-sovellusten kehittämiseen tarkoitettu sovelluskehys, joka on toteutettu PHP-ohjelmointikielen pohjalta. Se on vapaan lisenssin tuote, eli sen lähdekoodi on kaikille saatavilla. Zend Frameworkin kehitystyö alkoi vuonna 2005, jolloin monet muut kehysympäristöt, kuten Ruby on Rails ja Spring Framework saavuttivat suurta suosiota WWW-sovelluskehittäjien keskuudessa. Sen ensimmäinen versio, Zend Framework 1.0, julkaistiin 1. heinäkuuta vuonna 2007. (Allen, Lo & Brown 2009)

Sovelluskehys tarkoittaa ohjelmistotuotetta, joka muodostaa rungon sen päälle rakennettavalle sovellukselle. Sen päätarkoituksena on nopeuttaa uusien sovellusten, kuten dynaamisten WWW-sivustojen kehittämistä, joten sitä voidaan yleisesti pitää eräänlaisena ohjelmoinnin apuvälineenä. Kehys pitää sisällään valmiiksi rakennettuja sovelluksen osia, joita ei tarvitse kirjoittaa uudelleen ohjelmistokehityksen aikana. Sovelluskehystä ei siis voida käyttää sellaisenaan, vaan varsinainen ohjelma kostuu kehyksestä ja sen päälle rakennettavasta sovelluksesta. Muita sovelluskehyksiä on toteutettu muun muassa Java-kielen pohjalta.

Zend Framework kehittyy jatkuvasti sen suosion kasvaessa, joten uusia versioita julkaistaan tasaiseen tahtiin. Uusin vakaa julkaistu versio, Zend Framework 1.9.5 julkaistiin tämän opinnäytetyön kirjoittamisen aikana 27.10.2009. Kehitysympäristön toiminnot ja toimintatavat ovat hieman muuttuneet uusien versioiden myötä, mutta sovelluskehyksen rakenne on pysynyt samana. Tämän toimeksiannon toteutuksessa käytetään Zend Frameworkin 2.2.2009 julkaistua versiota 1.7.4, joka oli viimeisin julkaistu versio projektin alkuaikoina.

3.2.1 MVC-ohjelmistoarkkitehtuuri

MVC tulee sanoista Model-View-Controller eli malli-näkymä-ohjain ja se on ohjelmistoarkkitehtuuri, jonka tarkoituksena on erottaa käyttöliittymä muusta sovelluslogiikasta. MVC-arkkitehtuuri perustuu norjalaisen Trygve Reenskaugin jo vuonna 1979 kehittämään malliin. Vaikka malli on vanha, on se edelleen laajasti käytetty malli sovelluskehittäjien keskuudessa eikä se ole juurikaan muuttunut ajan kuluessa.

MVC-arkkitehtuuria käytetään erityisesti sovelluksissa, joissa on graafinen käyttöliittymä. Arkkitehtuuri jaetaan sen nimestä koostuvista sanoista kolmeen osaan. Järjestelmän pilkkominen osiin tuo etua erityisesti suuria järjestelmiä toteutettaessa, koska kehitettävän järjestelmän eri osien riippuvuutta toisiinsa ei ole tai se on verrattain pieni. Näin kehitystyö voidaan hajauttaa esimerkiksi eri työryhmille ilman, että kehitysprosessi hidastuu.

Model eli malli on usein jollain tavalla linkitetty tietokantaan ja se keskittyy järjestelmän sovellusaluekohtaisen tiedon käsittelyyn, tallentamiseen ja ylläpitoon. Malli ei ole riippuvainen näkymästä tai ohjaimesta, joten se voidaan suunnitella, toteuttaa ja testata riippumatta järjestelmän muista osista. Samaan malliin voidaan tehdä erilaisia käyttöliittymiä, kuten perinteinen graafinen käyttöliittymä ja WWW-käyttöliittymä.

View eli näkymä määrittelee käyttöliittymän ulkoasun ja sen, kuinka mallin tiedot esitetään käyttöliittymässä. Sovellukseen on mahdollista toteuttaa useita näkymiä erilaisten käyttötarkoitusten mukaan, jotka ovat riippumattomia toisistaan, mutta käyttävät samaa mallia. Käyttöliittymän ulkoasu eli näkymä voidaan vaihtaa muuttamatta ohjainta ja päinvastoin ohjain voidaan vaihtaa muuttamatta näkymää.

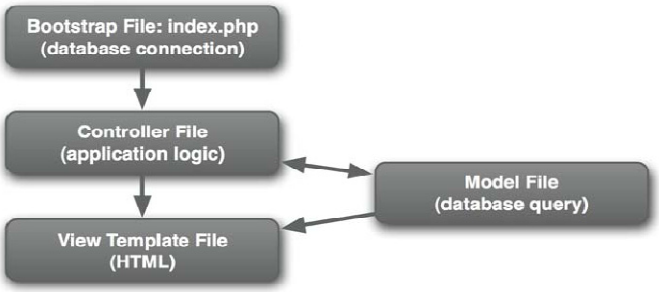
Controller eli ohjain vastaanottaa käyttäjän syötteet ja käskyt. Käyttäjän syöttämien komentojen perusteella se ohjailee sovellusta mallia ja näkymää muuttamalla vastaamaan käyttäjän komentoja. Ohjain voi vaikuttaa sekä malliin, että näkymään tai vaihtoehtoisesti vain toiseen niistä. Esimerkiksi käyttäjä klikkaa hiirellä, jolloin ohjain ottaa vastaan tämän tapahtuman ja muuntaa sen sisällön tilaan vaikuttavaksi operaatioksi.

3.2.2 Zend Frameworkin rakenne ja toiminta

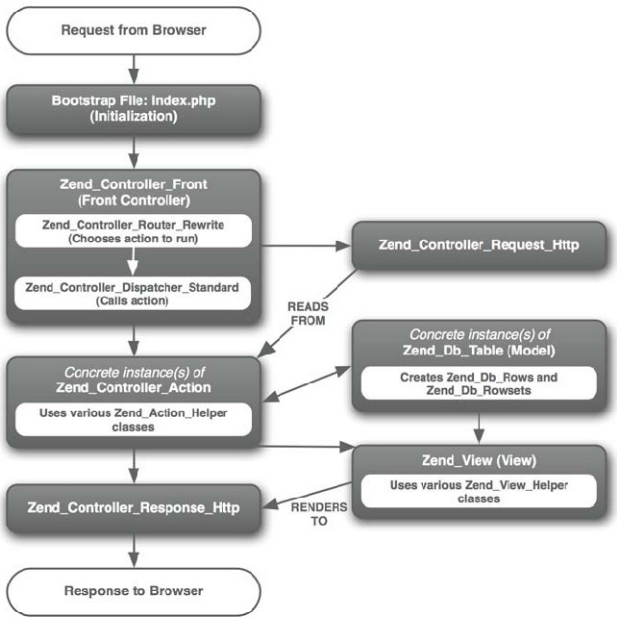
Zend Frameworkin avulla rakennettu WWW-sivusto näkyy ulospäin käyttäjälle ainoastaan osoiterivin kautta. Koska Zend Frameworkissa on sisäänrakennettu URL:n esikäsittelijä, näyttää se osoiterivin helpommin muistettavassa muodossa. Kun perinteisellä tavalla toteutetun WWW-sivuston osoiterivillä näkyy esimerkiksi teksti: *http://virtuaaliviipuri.tamk.fi/projekti.php?lang=fi*, voisi vastaavasti Zend Frameworkin avulla toteutetun sivuston osoiterivi olla esimerkiksi: *http://virtuaaliviipuri.tamk.fi/public/fi/project/*.

Sivuston kehittäjälle tärkeimpänä huomioon otettavana seikkana on isojen ja pienten kirjainten välinen ero esimerkiksi luokissa ja tiedostojen nimissä. Sivuston mallien ja ohjainten nimet kirjoitetaan isolla alkukirjaimella. Myös kaikkien luokkien nimet aloitetaan isolla alkukirjaimella. Näkymien ja sivupohjien nimet kirjoitetaan päinvastoin kokonaan pienellä. (Zervaas 2009)

Zend Frameworkin suunnittelumallin mukainen rakenne toteutetaan sovelluskehittäjän toimesta. Tuloksena syntyy kuvan 3 kaltainen rakenne, jonka toimintaperiaatetta selvitetään syvemmin vaihe vaiheelta kuvassa 4. Sivuston hakemistorakenne (Kuva 5) muodostuu pääperiaatteessa kolmesta kansiosta: *application, library* ja *public*, joiden alle sijoitetaan sovelluksen toimintaan tarvittavat tiedostot käyttäen erilaisia alikansioita.



Kuva 3. Zend Frameworkin MVC-mallin mukainen WWW-sovelluksen rakenne. Bootstrap file on tässä tapauksessa nimeltään index.php. Se on ns. esilatausohjelma, jonka kautta ajetaan sovelluksen konfigurointi. (Allen, Lo & Brown 2009)



Kuva 4. Zend Frameworkin toiminta (Allen, Lo & Brown 2009)

1. Ensimmäisessä vaiheessa käyttäjän selain lähettää palvelimelle pyynnön. Esilatausohjelma (*index.php*) alustaa ja konfiguroi sovelluksen (Kuva 4).

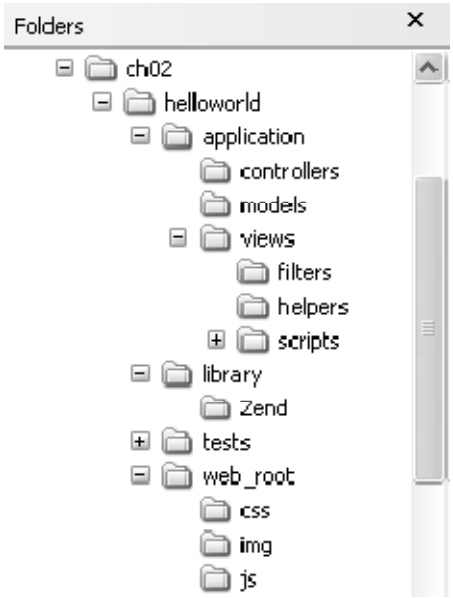
2. Zend_Controller_Front on Zend Frameworkin ydin. Se tarkistaa, onko käyttäjän osoiteriville syöttämässä osoitteessa domainin jälkeisiä osia. Tarkistuksen jälkeen Front Controller ohjaa pyynnön eteenpäin oikealle toiminto-ohjaimelle (*Zend_Controller_Action*) (Kuva 4).

3. Käytettävän ohjaimen mukaan muodostetaan malli, joka pohjautuu tässä tapauksessa luokkaan *Zend_Db_Table*. *Zend_Db_Table* tarjoaa olio-pohjaisen pääsyn erilaisiin tietokantoihin, kuten *MySQL* tai *Oracle* (Kuva 4).

4. Ohjain suorittaa määritellyn funktion mallin avulla, minkä jälkeen muodostetaan näkymä (*Zend_View*). *Zend_View* tarjoaa PHP-pohjaisen järjestelmän. Tämä tarkoittaa sitä, että näkymien skriptit toteutetaan käyttäen PHP:tä (Kuva 4).

5. Tämän jälkeen näkymä lähetetään vastauksena selaimelle ja käyttäjälle näkyy haluttu sivu (Kuva 4).

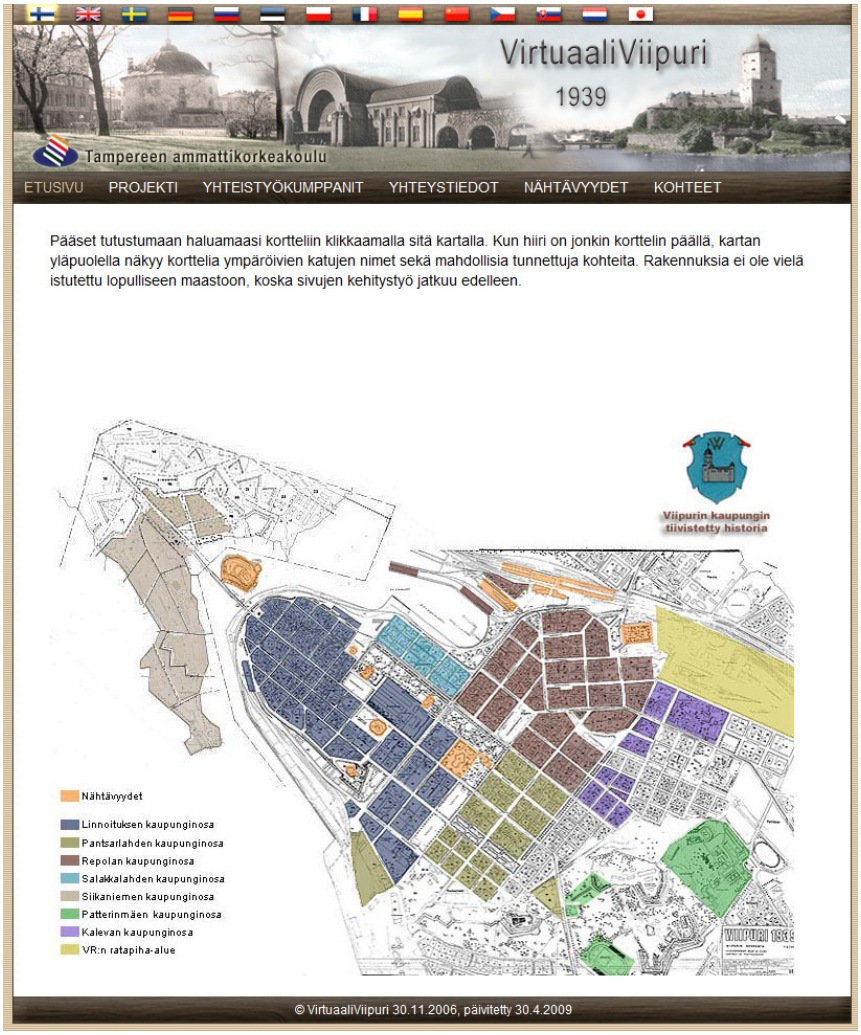
Zend Frameworkin hakemistorakenne on selkeä, ja siitä voidaan helposti nähdä sovelluskehyksen tarkoituksen erottaa kehitettävän sovelluksen toimintalogiikka näkymästä. Sovelluksen mallit, ohjaimet ja näkymä ovat eri kansioissa erotettuna itse sivuston juuresta, joka lisää sivuston tietoturvaa. Kuvan esimerkin mukaisessa hakemistorakenteessa on luotu *HelloWorld*-sovellus, jossa sivuston juuri on *web_root*-niminen kansio. (Kuva 5). Kansioon sisältyvät kaikki sivuston kuvat, tyylimääritykset sekä esimerkiksi JavaScript-komentosarjatiedostot.



Kuva 5. Zend Frameworkilla toteutetun WWW-sivuston tyypillinen hakemistorakenne (Allen, Lo & Brown 2009)

4 VirtuaaliViipuri-sivusto

VirtuaaliViipuri-projektin päätavoitteena on luoda 3D-mallinnuksella virtuaalinen Viipurin kaupunki sellaisenaan kuin se oli syyskuussa vuonna 1939 suomalaisena kaupunkina. Virtuaalinen kaupunki on sijoitettu WWW-ympäristöön ja sinne lisätään uusia alueita sitä mukaa kun opiskelijat saavat niitä mallinnettua. VirtuaaliViipuri-sivusto sijaitsee osoitteessa: http://www.virtuaaliviipuri.fi (Kuva 6), ja se on avoin kaikille. Sivuston kehityksestä ja ylläpidosta vastaavat Tampereen ammattikorkeakoulun opiskelijat ja henkilökunta.



Kuva 6. VirtuaaliViipuri WWW-sivuston etusivu

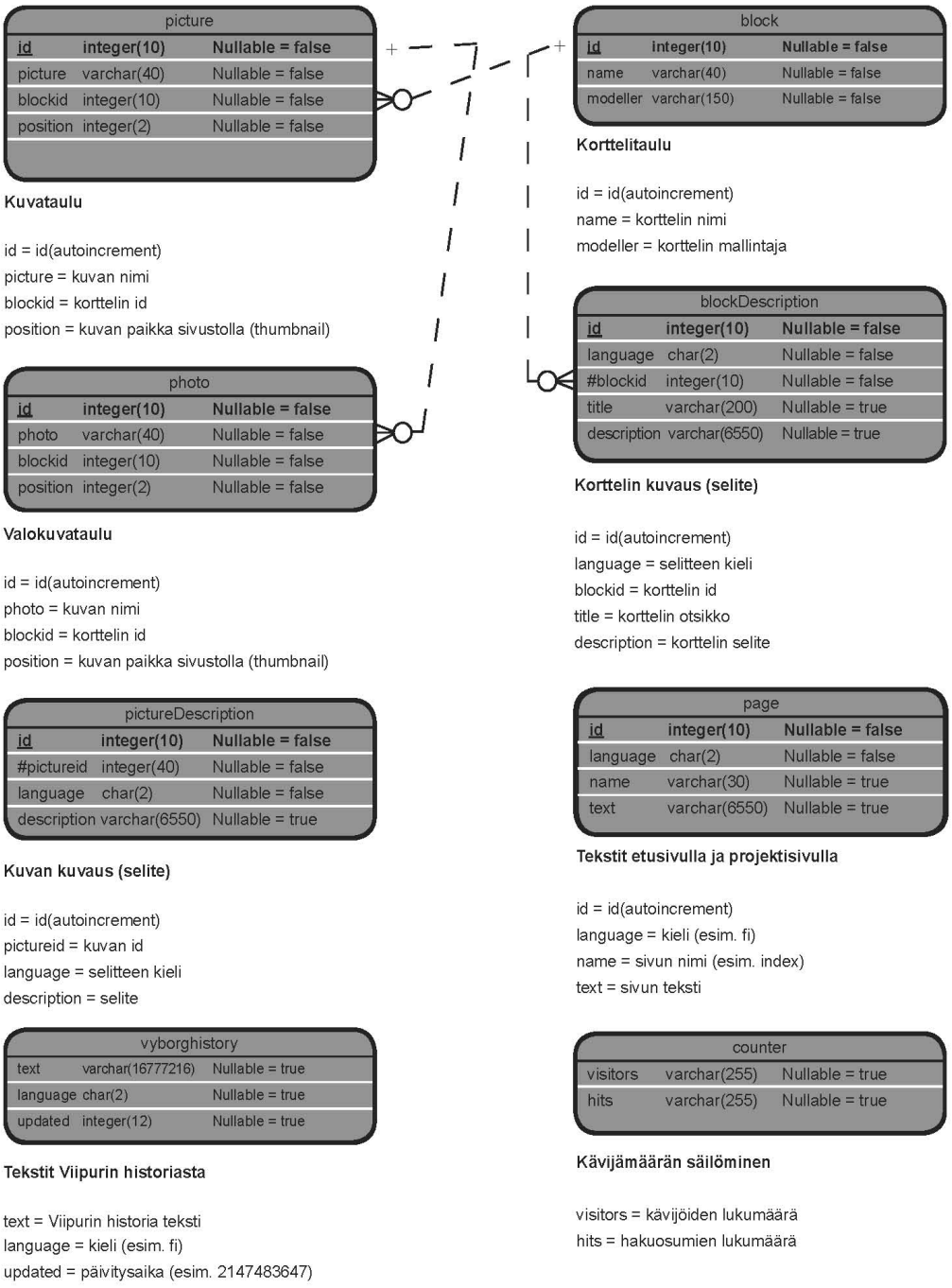
4.1 Tietokannan rakenne

WWW-sivustoa rakentaessa tietokannan määrittely täytyy suunnitella huolellisesti ja ottaa kaikki tärkeät tekijät huomioon. Koska kysymyksessä on kokonaisen kaupungin mallintaminen, on ilmiselvää, että esimerkiksi sivustolla näytettävien kuvien ja videoiden määrä on huomattavan suuri. Tästä johtuen myös tietokantaan tallennetaan suuri määrä tietoa. Tietokantaa suunniteltaessa täytyy varmistaa, että tietokannan tauluissa on riittävästi tilaa kasvulle esimerkiksi uusia kieliä tai kuvia lisättäessä. Tällä vältytään tietokantarakenteen muuttamiselta tulevaisuudessa.

Mikäli tietokannan suunnittelussa ja toteutuksessa ei ole keskitytty oikeisiin seikkoihin, näkyy se selvästi loppukäyttäjälle. Huonosti suunniteltu tietokanta voi näkyä käyttäjälle sivuston hitautena tai pahimmassa tapauksessa sivuston toimimattomuutena. Hyvin suunnitellun tietokannan rakenteen tärkeimpiä ominaisuuksia ovat muutosjoustavuus, selkeys, eheys sekä kattavuus. (Hovi & Huotari & Lahdenmäki 2003: 20-21)

VirtuaaliViipuri-sivuston tietokannan rakenne uudistettiin kokonaan vuonna 2008, joten hyväksi todetun rakenteen muuttaminen ei ollut tarpeen. Uudistetun tietokannan rakenteen (Kuva 7) pääasiallinen tavoite oli tietokannan skaalautuvuus uusien kielikäännöksien ja korttelien syntyessä. Uudistuksen myötä tietokannan rakenne säilyy muuttumattomana uusia kortteleita tai kieliä lisättäessä, mikä tarkoittaa, että esimerkiksi kaikkien kortteleiden kuvaukset säilötään samassa taulussa kaikille eri kielille. Kieliä maailmassa on n. 6000, jolloin nykyinen tietokannan rakenne mahdollistaa sivuston kääntämisen kaikille maailman kielille, koska tauluissa on riittävästi tilaa kasvulle. Taulujen rivien kasvaessa niiden luettavuus hieman kärsii, mutta tietokannan rakenne säilyy eheänä ja selkeänä.

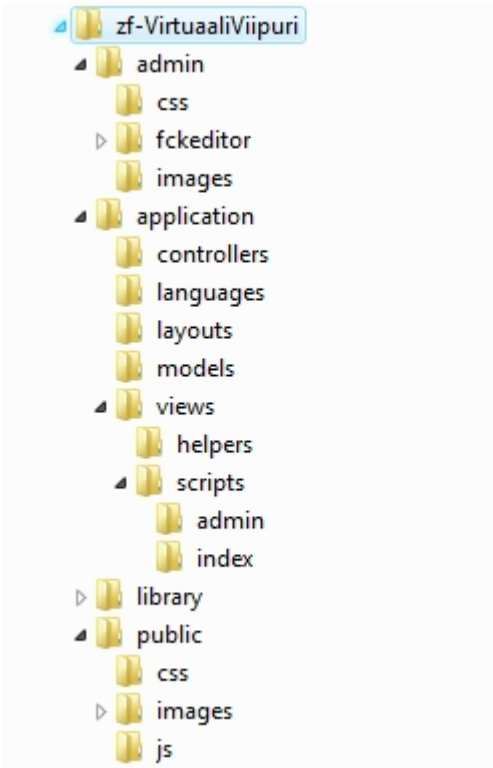
TIETOKANNAN TAULUT
JA TAULUJEN SELITYKSET



Kuva 7. VirtuaaliViipuri-sivuston tietokannan rakenne

4.2 Hakemistorakenne

Zend Frameworkin ansiosta sivuston hakemistorakenteesta muotoutuu varsin yksinkertainen ja selkeä (Kuva 8). Zend Frameworkin asennus on erittäin helppoa: asennus tapahtuu kopioimalla *library*-niminen kansio sovelluksen juureen. Kansio pitää sisällään kaikki Zend Frameworkin toimintaan tarvittavat tiedostot, joten tietoturvan kannalta kansio kannattaa pitää erillään varsinaisesta sovelluksen juuresta (*public*-niminen kansio, joka sisältää *index.php*-tiedoston). Myös *application*-kansio tulee pitää erillään sivuston juuresta. Näin estetään mahdollisuus väärinkäytöksiin, koska edellä mainittuihin kansioihin ei voi päästä käsiksi nykyisillä selaintekniikoilla. Tarkempi kuvaus kansioiden sisältämistä tiedostoista tulee esille seuraavassa kappaleessa. (Allen, Lo & Brown 2009: 21-22)



Kuva 8. VirtuaaliViipuri-sivuston hakemistorakenne

4.3 Tiedostorakenne

Zend Frameworkin ansiosta WWW-sivuston tiedostorakenteesta syntyy selkeä ja yksinkertainen. Itse sovellus määritetään omaan erilliseen kansioonsa, yleinen ja hallintasivusto omaansa sekä Zend Frameworkin toimintaan tarvittavat tiedostot ovat omassa kansiossaan.

application-kansio

Kansio sisältää kaiken sovelluksen ajamiseen tarvittavan ohjelmakoodin. MVC-arkkitehtuurin mukaisesti mallit, näkymät ja ohjaimet ovat eritelty omiin kansioihinsa (*models, views ja controllers*), jotka sisältävät kaikki sivustossa tarvittavat mallit, näkymät ja ohjaimet sekä julkisella puolella että hallintaosiossa. Kansion juureen sijoitetaan myös tietokannan konfigurointitiedosto (Kuva 9), jossa määritellään käytettävä tietokanta.

```
[general]
db.adapter = PDO_MYSQL
db.params.host = 127.0.0.1
db.params.username = root
db.params.password =
db.params.dbname = dbvviipuri
```

Kuva 9. Tietokannan konfigurointitiedosto määrittää käytettävän tietokanta-adapterin ja tietokannan määritteet.

Konfigurointitiedosto on muotoa *ini*. Toinen yleisesti käytetty formaatti konfigurointitiedostolle on *xml*. Lisäksi kansio sisältää sivuston kielitiedostot niin julkiselle puolelle kuin hallintaosioonkin sekä apufunktion *view->helpers*-kansion alla, jota sovellus käyttää tyylitiedostojen sijainnin paikantamiseen. *Layouts*-kansion sisältä löytyvät sivupohjat eli *layoutit*.

admin- ja public-kansio

Kansioiden juuressa sijaitsevat *.htaccess* (Hypertext Access) -tiedostot, joilla määritellään Apachen hakemistokohtaiset asetukset, kuten esimerkiksi estetään kansioiden selaus (Kuva 10). Juuressa sijaitsee lisäksi *index.php*-tiedosto eli esilatausohjelma. Molemmat kansiot sisältävät tyylitiedostot (*css*-kansio), javascript-tiedostot (*js*-kansio) sekä kuvat, joita mm. sivupohjissa käytetään (*images*-kansio).

Admin-kansion alta löytyvä *Fckeditor*-kansio sisältää hallintaosiossa käytettävän tekstieditorin. Varsinaiset mallinnukset ja videot kortteleista sijaitsevat *public*-kansiossa ja edelleen *images*-kansiossa, jossa jokaiselle korttelille on oma kansionsa korttelin nimen mukaan.

```
# Rewrite rules for Zend Framework
RewriteEngine on
RewriteCond %{REQUEST_FILENAME} !-f
RewriteRule .* index.php
# Security: Don't allow browsing of directories
Options -Indexes
# PHP settings
php_flag magic_quotes_gpc off
php_flag register_globals off
php_flag short_open_tag on
```

Kuva 10. Hakemistokohtaiset asetukset määritellään .htaccess-tiedostossa. Esimerkiksi tietoturvan kannalta on tärkeää estää WWW-selaimelta kansioiden selaus. Tiedostossa sallitaan lisäksi URL:n uudelleenkirjoitus Zend Frameworkille sekä määritetään yleiset PHP:n asetukset.

library-kansio

Kansio sisältää Zend Frameworkin toimintaan tarvittavat tiedostot, jotka sijaitsevat *library*-kansion alta aukeavasta *Zend*-kansiosta. Kansiosta löytyvät myös kaikki Zend Frameworkin komponentit kansioittain.

4.4 Sivuston toteutus

Index.php-tiedosto alustaa sovelluksen (Kuva 11). Tiedosto pitää sisällään hakemistojen määritykset käytettäville ohjaimille ja luokille, jotta ne voidaan nopeasti ladata myöhemmin. Tähän tarvitaan *Loader.php*-tiedosto ja se sisältää *Zend_Loader*-luokan. Tiedosto otetaan käyttöön *include*-komennolla. *Index.php*-tiedosto loppuu komentoon *dispatch*, joka käynnistää sovelluksen.

```
1   <?php
2   error_reporting(E_ALL|E_STRICT);
3   ini_set('display_errors', 1);
4   date_default_timezone_set('Europe/Helsinki');
5   // directory setup and class loading
6   set_include_path('.' . PATH_SEPARATOR . '../library/'
7   . PATH_SEPARATOR . '../application/models'
8   . PATH_SEPARATOR . get_include_path());
9   include "Zend/Loader.php";
10  Zend_Loader::registerAutoload();
11  Zend_Session::start();
12
13  // load configuration
14  $config = new Zend_Config_Ini('../application/config.ini', 'general');
15  $registry = Zend_Registry::getInstance();
16  $registry->set('config', $config);
```

Kuva 11. Index.php-tiedostossa alustetaan sovellus. Sivustokehityksessä on hyödyllistä käyttää virheiden raportointia (error_reporting), mikä auttaa mahdollisten sivuston toimintaan liittyvien virheiden paikantamisessa.

Sivuston toiminto-ohjaimet löytyvät tiedostosta *IndexController.php* (Kuva 12) ja vastaavasti hallintasivuston toimintaohjaimet tiedostosta *AdminController.php*. Toiminto-ohjainluokka periytyy *Zend_Controller_Action*-luokasta. Luokka sisältää kaikki funktiot, joita käytetään esimerkiksi halutun sisällön näyttämiseksi sivulta toiselle siirryttäessä. Käytettävä funktio määräytyy halutun toiminnon mukaan.

```
1  <?php
2  /*Setting up controller: In Zend Framework, the controller is a class that
3  must be called {Controller name}Controller. Note that {Controller name} must
4  start with a capital letter. This class must live in a file called
5  {Controller name}Controller.php within the application/controllers
6  directory. */
7  class IndexController extends Zend_Controller_Action
8  {
9      /* Setting up actions: Each action is a public function within the controller class that must be
10     named {action name}Action. In this case {action name} should start with a lower
11     case letter and again must be completely lowercase. */
12     function indexAction(){
13
14     // get and assign the language parameter
15     $request = $this->getRequest();
16     $this->view->assign('language', $request->getParam('language'));
17
18     $language = $this->_request->getParam('language');
19
20     // get index text from database
21     $content = new Content();
22     $text = $content->select('text')
23             ->from('page')
24             ->where('page.language = ?', $language)
25             ->where('page.name = ?', 'index')
26             ;
27
28     // pass the result to the view component
29     $this->view->page = $content->fetchRow($text);
30     }
```

Kuva 12. IndexController.php-tiedosto sisältää sivuston funktiot, kuten esimerkiksi etusivulla näytettävän tekstin, joka haetaan tietokannasta valitun kielen perusteella.

Funktiossa käytettävä malli taas määräytyy sen mukaan, mitä tietokannan taulua funktiossa käytetään. Etusivulla käytetään *index*-funktiota, joka käyttää tietokannan *page*-taulua. Mallissa määritetään luokka, josta malli periytyy sekä käytettävän tietokannan taulun nimi (Kuva 13).

```
1  <?php
2  class Content extends Zend_Db_Table
3  {
4  protected $_name = 'page';
5  }
```

Kuva 13. Mallin nimi on Content. Content-luokka periytyy Zend Frameworkin Zend_Db_Table-luokasta. Mallin käyttämän tietokannan taulun nimi on page.

View eli näkymä koostuu näkymän tiedostosta ja sivupohjasta. Näkymä on komponentti, jolle funktion lopputulos siirretään ja sen kautta käyttäjän selaimeen tulostetaan haluttu sivu (Kuva 14). Näkymän tiedosto on muotoa *.phtml*. Tiedosto voi sisältää esimerkiksi niin XHTML-, PHP- kuin JavaScript-kieltä tai kaikki edellä mainittuja yhdessä. Sivupohja eli *layout* on tiedosto, johon pohjautuen näkymä luodaan (Kuva 15). Sivupohja sisältää muun muassa sivun pakolliset XHTML-dokumentin määritykset ja elementit.

```
1  <?php
2  //assign the language parameter and include the correct language file
3  $language = $this->escape($this->language);
4  include '../application/languages/' .$language. '_ui.php';
5
6  // display the result of the indexAction function
7  echo $this->page->text; ?>
```

Kuva 14. Näkymänä toimii index.phtml-tiedosto, jossa tulostetaan käyttäjälle haluttu sisältö.

```
28 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
29 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
30 <html xmlns="http://www.w3.org/1999/xhtml">
31
32 <head>
33  <meta http-equiv="Content-Type" content="text/html; charset=utf-8 />
34  <meta name="description" content="Hankkeen päätavoitteena on luoda 3D-mallinnuksella
35   www-ympäristöön virtuaalinen Viipurin kaupunki sellaisena kuin se oli syyskuussa 1939
36   suomalaisena kaupunkina." />
37  <meta name="keywords" content="viipuri, virtuaaliviipuri, virtuaali, virtuaalinen,
38   projekti, 3d, 3d-mallinnus, tamk, karjala" />
39
40  <title>
41  <?php
42      // print title
43      echo $title;
44  ?>
45  </title>
46
47  <link rel="stylesheet" type="text/css" media="screen"
48          href="<?php echo $this->baseUrl();?>/css/jee.css" />
49
50  </head>
51
52  <body>
53  <div id="container">
```

Kuva 15. Sivupohja eli layout sisältää pakolliset XHTML-dokumentin määritykset.

4.5 Sivuston toiminta

VirtuaaliViipuri-sivuston navigointi on helppoa ja käyttäjää ohjeistetaan aktiivisesti sivustolla liikkumisessa. Osoitteen syötettyään käyttäjä ohjautuu etusivulle, josta pääsee navigoimaan sivustolla. Sivuston oletuskielenä on suomi, mutta kielen voi vaihtaa haluamakseen sivuston yläreunan lippupalkista. Käyttäjä pääsee kätevästi selaamaan mallinnettuja kortteleita etusivulla näkyvästä Viipurin kartasta tai vaihtoehtoisesti *Projekti*-sivun alareunassa olevasta kartasta.

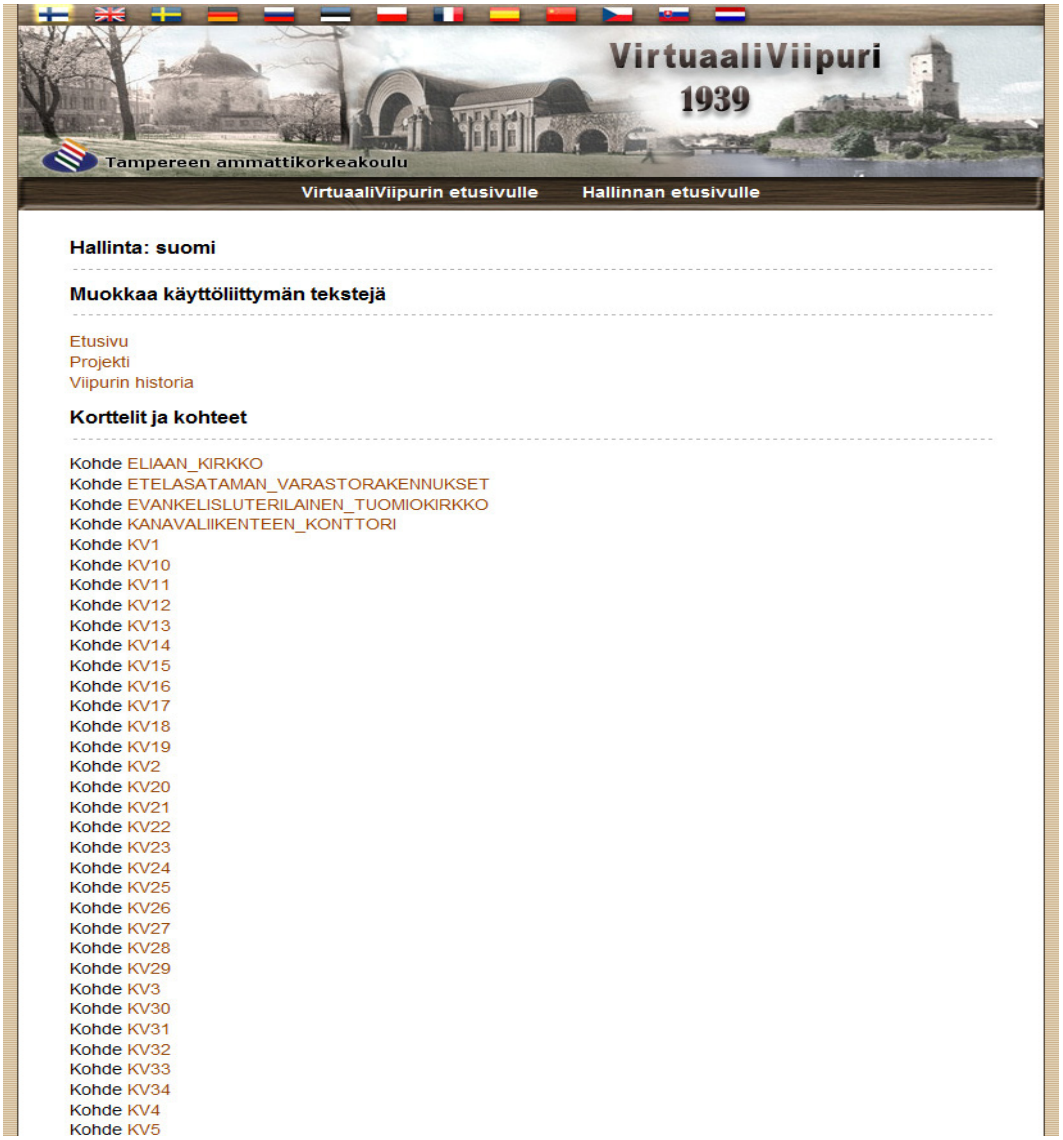
JavaScriptilla toteutettu kartta toimii interaktiivisesti käyttäjän näytöllä ja hiiren kohdistus korttelien päälle tulostaa korttelin nimen kartan yläpuolelle, mikäli kortteli on mallinnettu. Viipurin logon kautta pääsee sivulle, jossa kerrotaan hieman Viipurin historiasta. Hiiren painallus korttelin päällä avaa sivun, jossa annetaan lisätietoa valitusta korttelista ja sen mallinnuksesta. Sivulle listataan mallinnettuja kuvia ja valokuvia pienoiskoossa ja lisäksi mahdollisesti myös videokuvaa korttelista. Sovellus hakee korttelin kansiosta kaikki mallinnetut kuvat. Kuvien paikka sivulla määräytyy tietokannasta haettavan paikkatiedon perusteella.

Pienoiskuvia klikkaamalla tulostetaan sivu, jossa näkyy haluttu kuva isommassa kuvakoossa ja mahdollisesti jonkinlainen kuvaus kuvasta. Kuvaus haetaan tietokannasta sen mukaan, mikä kuva käyttäjän selaimessa näkyy. Kuvia voi selata tällä sivulla vapaasti kuvan molemmilla laidoilla olevista nuolista.

Pääsy muihin sivuston osiin onnistuu navigointipalkin kautta. *Projekti*-sivulla kerrotaan VirtuaaliViipuri-projektista ja sen kehityksestä. *Yhteistyökumppanit*-sivu esittelee nimensä mukaisesti projektin yhteistyökumppanit. Lisäksi tällä sivulla olevan tilauslomakkeen kautta käyttäjä voi tilata Harri Miettisen teoksen Historian havinaa Viipurissa. *Yhteystiedot*-sivu tulostaa hankkeen johtokunnan yhteystietoja. *Nähtävyydet*-sivulta pääsee selaamaan Viipurin merkittävimpiä nähtävyyksiä. Viimeisenä navigointipalkissa on *Kohteet*, josta pääsee pikalinkkien kautta tarkastelemaan esimerkiksi Viipurin kouluja tai ravintoloita.

Hallintasivusto (Kuva 16) on suojattu käyttäjätunnuksella ja salasanalla väärinkäytösten estämiseksi. Mikäli käyttäjä syöttää käyttäjätunnuksen tai salasanan väärin, ohjataan hänet sivulle, jossa kerrotaan käyttöoikeuden puuttuvan. Käyttöön oikeuttavan käyttäjätunnuksen ja salasanan jälkeen käyttäjälle tulostetaan sivuston hallinnan etusivu.

Tällä sivulla on linkit kaikkiin kohteisiin, joita on mahdollista muokata hallintaosion kautta. Todellisuudessa tämä tarkoittaa niitä tekstejä, jotka ovat tallennettu tietokantaan. Kielen, johon muokkaus kohdistuu, voi valita haluamakseen sivun yläreunassa olevasta lippupalkista. Muut kohteet, kuten navigointipalkin tekstit, täytyy muuttaa suoraan ohjelmakoodiin.



Kuva 16. VirtuaaliViipuri WWW-sivuston hallintasivu

5 Yhteenveto

Opinnäytetyön aiheeksi valitsin WWW-sivustot ja niiden toteuttamistekniikat. Yleisesti käytetyt perinteiset toteuttamistekniikat ovat jo tulleet tutuksi suoritettujen opintojen kautta, joten osaamista web-selainohjelmointiin oli jo ennen opinnäytetyön aloittamista. Sen sijaan MVC-arkkitehtuurin mukainen rakenne oli täysin uusi lähtökohta sivuston toteutuksessa ja se loi haasteita varsinkin toimeksiannon alkuvaiheissa.

Toimeksiannon tavoitteena oli suunnitella ja toteuttaa VirtuaaliViipuri-projektin WWW-sivuille uusi rakenne modulaarisella toimintaperiaatteella. Rakenne toteutettiin käyttäen valitsemaani Zend Framework sovelluskehystä, joka on vapaan lisenssin sovelluskehys ja noudattaa MVC-mallin mukaista rakennetta.

Zend Frameworkista tulikin koko projektin haastavin osuus, koska se poikkeaa suuresti perinteisestä tavasta kehittää WWW-sivustoja. Sovelluskehyksen rakenteen ja sen toiminnan ymmärtämiseen tarvittiin syvällistä perehtymistä ja siihen käytetty aika voidaan laskea kuukausissa. Perehtymisen aikana WWW-selainohjelmointiin kehittyi aivan uudenlainen ajattelutapa. Tämä ajattelutapa on syöpynyt mieleen niin syvälle, että se on syrjäyttänyt aikaisemman ajattelutapani ja tulen käyttämään sitä varmasti myös tulevaisuuden WWW-projekteissa.

Sovelluskehykseen perehtymisen aikana ymmärtämys kasvoi ja samalla vahvistui myös ajatus Zend Frameworkin käyttötarkoituksista. Zend Framework määritellään sovelluskehyksenä, mutta tosiasiassa sen voisi pikemminkin mieltää luokkakirjastona. Valmiit luokat todellakin säästävät sovelluskehittäjän aikaa, koska se kutistaa jonkin verran ohjelmakoodia ja poistaa tarpeetonta koodin toistoa. Tästä on hyötyä etenkin suuremmissa sovelluksissa, joissa sen jo lukuisat ja jatkuvasti lisääntyvät ominaisuudet voidaan ottaa laajemmin käyttöön.

Sovelluskehyksien käyttö web-selainohjelmoinnissa on silminnähden lisääntynyt sen saaman suuren suosion kautta WWW-sovelluskehittäjien keskuudessa. Parhaiten sen

huomaa siitä, kun vierailee aktiivisesti Internetin eri sivustoilla. Opinnäytteen aiheesta tehdyn taustatyön kautta kävi ilmi, että erityisesti yritykset ovat omaksuneet tavan toteuttaa sivustojaan sovelluskehyksiä käyttäen. Zend Frameworkin kehitystyö on siis onnistunut pääsemään lähemmäs tavoitettaan, joka on vakinaistaa Zend Framework yhtenä ”de facto” -kehyksistä WWW-sovelluskehityksessä.

Aiheeseen perehtymisen myötä kehittyi myös uudenlainen tyyli katsella WWW-sivustoja. Uudelle sivustolle saapuessaan huomaa nykyään heti ulkoasun jälkeen tarkastelevansa poikkeuksetta sitä, millaista tekniikkaa käyttäen sivusto on toteutettu, ja onko sen kehityksessä käytetty mahdollisesti jotain sovelluskehystä.

Toimeksiannon tavoitteet saavutettiin hyvin, mikä näkyy sivuston nopeudessa ja ohjelmakoodin vähenemisenä. Uusi rakenne selkeyttää sivuston toiminnallisuutta ja pidentää sivuston ikää. Muutokset esimerkiksi sivuston yhteen sivuun voidaan tehdä helposti yhteen paikkaan eli sivuun kohdistuvaan ohjelmakoodiin. Sivuston turvallisuus oli myös yksi uuden rakenteen lähtökohdista ja se otettiin huomioon heti kehitystyön alkuajoista lähtien. Näin ollen sivustoon kohdistuvien väärinkäytösten riski saadaan pidettyä minimissään.

VirtuaaliViipuri-sivuston kehitystyö jatkuu tämän opinnäytetyön jälkeenkin. Yksi tulevaisuudessa kehitettävistä osista voisi olla hallintasivun laajentaminen entisestään niin, että muutosten tekeminen itse ohjelmakoodiin saataisiin mahdollisimman pieneksi. Näin ollen hieman kokemattomienkin sivuston ylläpitäjien työ helpottuisi. Seuraava kehitystarve voisi olla kuitenkin hakutoimintojen lisääminen sivustoon. Tämä idea on vilahtanut aikaisemmin myös hankkeen kokouksissa, joten hakukentän ilmestyminen sivustolle on vain ajan kysymys.

Zend Framework ei vielä suoraan tarjoa valmista hakukomponenttia, joten sellainen jouduttaisiin itse kehittämään. Hakukenttään voitaisiin kirjoittaa esimerkiksi kadun nimi, jonka jälkeen tuloksena syötettäisiin ne korttelit, jotka kyseisellä kadulla sijaitsevat. Tämän johdosta käyttäjälle annettaisiin enemmän mahdollisuuksia tutustua Viipurin kaupunkiin virtuaalisesti.

LÄHTEET

Painetut:

- *Allen, Rob & Lo, Nick & Brown, Steven 2009. Zend Framework in Action.*
- *Heinisuo, Rami & Rauta, Ilkka 2007. PHP ja MySQL Tietokantapohjaiset verkko-palvelut.*
- *Hovi, Ari & Huotari, Jouni & Lahdenmäki, Tapio. 2003. Tietokantojen suunnittelu & indeksointi.*
- *Lerdorf, Rasmus, Tatroe, Kevin, MacIntyre, Peter 2006 Programming PHP.*
- *Meloni C. Julie 2003. MySQL Trainer Kit.*
- *Rantala, Ari 2002. PHP Web-ohjelmoinnin peruskirja.*
- *Zervaas, Quentin 2007. Practical Web 2.0 Applications with PHP.*

Internet:

- *httpd.apache.org - The Apache HTTP Server Project 2009.* [online] [viitattu 16.8.2009] http://httpd.apache.org
- *Moisio, Aleksi 2008. MySQL - Suomalainen menestystarina.* [online] [viitattu 2.6.2009] http://www.digitoday.fi/bisnes/2008/01/16/mysql--suomalainen-menestystarina/20081468/66 - Digitoday. www.digitoday.fi
- *netcraft.com - June 2009 Web Server Survey 2009*. [online] [viitattu 13.7.2009] http://news.netcraft.com/archives/2009/06/17/june_2009_web_server_survey.html
- *php.net - PHP: Hypertext Preprocessor 2009.* [online] [viitattu 1.6.2009] http://www.php.net
- *phpmyadmin.net - phpMyAdmin 2009*. [online] [viitattu 11.8.2009] http://www.phpmyadmin.net/home_page/index.php
- *w3.org - Cascading Style Sheets 2009.* [online] [viitattu 4.6.2009] http://www.w3.org/Style/Activity