KYMENLAAKSON AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma / ohjelmistotekniikka

Ossi Lommi

INFOMONITORIN TOTEUTUS CAKEPHP-SOVELLUSKEHYKSELLÄ

Opinnäytetyö 2012

TIIVISTELMÄ

KYMENLAAKSON AMMATTIKORKEAKOULU
Tietotekniikka

LOMMI, OSSI	Infomonitorin toteutus CakePHP-sovelluskehyksellä
Insinöörityö	46 sivua
Työn ohjaaja	Laboratorioinsinööri Marko Oras
Toimeksiantaja	Kymenlaakson ammattikorkeakoulu
Toukokuu 2012	
Avainsanat	CakePHP, sovelluskehys, MVC

Tässä opinnäytetyössä on kuvattu Kymenlaakson ammattikorkeakoulun tiedotussovelluksen uudistaminen. Sovelluksen vaatimien uudistusten toteuttaminen vanhaan versioon havaittiin kuitenkin hyvin työlääksi. Niinpä sovellus päätettiin tehdä uudelleen hyödyntämällä nykyaikaisia ja nopean kehityksen mahdollistavia tekniikoita.

Tässä työssä käsiteltyjä asioita ovat sovelluksen vaatimusmäärittely sekä kehitysympäristö. Tärkeimmät osat ovat toteutustekniikkana käytetyn CakePHP-sovelluskehyksen kuvaaminen sekä käyttöliittymään ja tietokantaan liittyvien yksityiskohtien selvittäminen. Myös prosessin aikana kohdattuja ongelmia on selvitetty sekä mahdollisten lisäominaisuuksien tarvetta pohdittu. Työssä on pyritty myös avaamaan sovelluksen teknistä puolta niin, että mahdollisten muutosten tekeminen olisi seuraavalle kehittäjälle helpompaa.

Työn pääpaino on ollut käyttäjäystävällisen ja loogisesti toimivan sovelluksen kehittäminen tunnettuja ja joustavia web-teknologioita hyödyntäen. Ohjelmoinnissa on pyritty noudattamaan hyviä ohjelmointikäytäntöjä ja kirjoittamaan helposti hallittavaa koodia.

Työn lopputuloksena saatiin aikaan sovellus, joka täytti sille asetetut perusvaatimukset. Sovellukseen voidaan katsoa kuitenkin jääneen vielä kehitettävää sekä optimoitavaa. Omaa osaamista laajennettiin omaksumalla uusia tekniikoita sekä syventämällä aikaisemmin opittuja taitoja.

ABSTRACT

KYMENLAAKSON AMMATTIKORKEAKOULU
University of Applied Sciences
Information Technology

LOMMI, OSSI	Rebuilding an Info Display Application Using CakePHP Framework
Bachelor's Thesis	46 pages
Supervisor	Marko Oras, Laboratory Engineer
Commissioned by	Kymenlaakson ammattikorkeakoulu
May 2012	
Keywords	CakePHP, framework, MVC

The objective of this study was to rebuild the info display application of Kymenlaakso University of Applied Sciences. Implementing new features in the old version was considered quite difficult. The new version was built using modern techniques that enable fast development.

This thesis discusses the requirement specification and the development environment of the application. The most relevant sections concern the CakePHP framework and the details of the database and the user interface. Also, the problems faced during the development process and the possible need of additional features are discussed. The paper also explores the technical features of the application in order to ease the work of a future developer executing possible modifications.

The main goal was to develop a user-friendly and logically functioning application by taking advantage of well-known and flexible web development technologies. The application was written good programming practices in mind. The purpose was to produce source code that is easily maintained.

As the result of the project, an application meeting the fundamental requirements set was produced. The application can be considered adequate but it leaves room for some more development and optimization. New techniques were adopted and programming skills were also improved.

SISÄLLYS

TIIVISTELMÄ

ABSTRACT

1 JOHDANTO 6

2 VAATIMUSMÄÄRITTELY 7

2.1 Toimintaperiaate 7

2.2 Ilmoitukset 7

2.3 Monitorit 8

2.4 Esitys 9

2.5 Muut vaatimukset 9

3 KEHITYSYMPÄRISTÖ 10

3.1 CakePHP-sovelluskehys 10

3.1.1 MVC-arkkitehtuuri 11

3.1.2 CakePHP:n konventiot 12

3.1.3 Vaatimukset ja asennus 13

3.2 MySQL 15

3.3 jQuery 15

3.4 Ajax 15

3.5 Työkalut 16

3.5.1 NetBeans IDE 16

3.5.2 MySQL Workbench 17

3.5.3 Mozilla Firefox 18

4 TIETOKANTA 18

4.1 Taulut 18

4.2 Relaatiot 20

4.3 MySQL Events 21

5 KÄYTTÖLIITTYMÄ 21

5.1 Peruskäyttäjän näkymä 22

5.2 Hallintapuolen näkymä 23

5.2.1 Etusivu 23
5.2.2 Uusi ilmoitus 24
5.2.3 Monitorit 27
5.2.4 Esitys 28
5.2.5 Lataa kuva 31
6 TIETOTURVA 31
6.1 Validointi 32
6.2 Data Sanitization 33
6.3 Object Relational Mapping 33
7 SOVELLUS 33
7.1 Controller 34
7.2 View 36
7.3 Model 37
7.4 webroot 38
8 KÄYTETTÄVYYDEN TESTAAMINEN 39
8.1 Lomakkeiden testaaminen 40
8.2 Kontrollointi 41
9 HYLÄTYT RATKAISUT JA KEHITYSEHDOTUKSET 41
10 POHDINTA 43
LÄHTEET 45

1 JOHDANTO

Infomonitori on Kymenlaakson ammattikorkeakoulun käytössä oleva web-pohjainen sovellus, jonka kautta henkilökunta voi tiedottaa ajankohtaisista tapahtumista sekä muista yleisistä asioista. Sovellusta käytetään internet-selaimella ja se sisältää kaksi erilaista käyttöliittymää. Ensimmäinen näistä on kaikille käyttäjille tarkoitettu näkymä, joka listaa sovellukseen tallennetut ilmoitukset luetteloksi. Ilmoituksia voi tarkastella yksitellen tai vaihtoehtoisesti käynnistää esityksen, jossa valitun toimipisteen ilmoitukset vaihtuvat automaattisesti tietyin väliajoin. Henkilökunnalle tarkoitettu käyttöliittymä sisältää lisäksi toiminnot ilmoitusten luomiseen, muokkaamiseen ja poistamiseen.

Infomonitorin uudistustyö lähti sovellukseen halutuista lisäominaisuuksista, joista tärkeimpänä oli uusi KyAMK:n vierailijoille tarkoitettu ilmoitustyyppi ja -näkymä. Sovelluksen haluttiin voivan näyttää perinteisistä ilmoituksista poikkeavia ilmoituksia, jotka näkyisivät vierailijoille toimipisteiden aulatiloihin sijoitettavista pystyasentoon käännetyistä monitoreista. Nämä ilmoitukset sisältäisivät ainoastaan ajankohtaisiin tapahtumiin liittyvää informaatiota, kuten tapahtuman nimen, ajankohdan sekä pitopaikan.

Muutoksia alettiin aluksi toteuttaa olemassa olevaan sovellukseen, mutta tämä huomattiin erittäin vaivalloiseksi ja hankalaksi johtuen vanhan sovelluksen rakenteesta ja ohjelmointitavasta. Nopeasti tultiin siihen tulokseen, että koko sovellus olisi sekä mielekästä että järkevää tehdä alusta asti uudelleen, niin tällä kertaa lisättävien ominaisuuksien, kun mahdollisesti tulevaisuudessa tehtävien muutostöiden vuoksi. Tämä opinnäytetyö antoi mahdollisuuden infomonitorisovelluksen täydelliseen uudistamiseen.

Web-sovelluksia voidaan kehittää käyttäen apuna niin sanottuja ohjelmistokehyksiä, joilla voidaan vähentää ohjelmointityötä kehyksen sisältämien valmiiden ohjelmaosasten avulla. Vanhasta poiketen, uusi sovellus päätettiin rakentaa kyseistä menetelmää hyödyntäen. Ohjelmistokehykseksi valittiin vapaan lähdekoodin CakePHP-ohjelmistokehys. Sovellus käyttää tiedon tallennukseen niin ikään vapaasti hyödynnettävää MySQL-tietokantaa.

2 VAATIMUSMÄÄRITTELY

Infomonitorisovelluksen määrittely aloitettiin projektiin kuuluneen työryhmän keskeisellä palaverilla tammikuussa 2012. Sovelluksen keskeisimmät uudistukset olivat oppilaitoksen vierailijoille suunnatun ilmoitusjärjestelmän toteuttaminen, graafisen ilmeen uudistus sekä toimintavarmuuden parantaminen. Toteutustekniikan valintaan ja ohjelmointityöhön annettiin vapaat kädet. Seuraavissa kappaleissa on kuvattu lyhyesti uuden infomonitorisovelluksen keskeiset vaatimukset.

2.1 Toimintaperiaate

Infomonitorin perusidea päätettiin toteuttaa samalla tavalla kuin aikaisemmassakin versiossa. Sovellukselle rakennetaan web-pohjainen käyttöliittymä. Käyttöliittymä sisältää omat näkymänsä peruskäyttäjälle sekä ilmoitusten hallintaan. Molemmat näkymät näyttävät ilmoitukset listana, josta yksittäiset ilmoitukset avautuvat tarkasteltaviksi linkkiä painamalla. Sovelluksesta voidaan käynnistää esitys, jossa ilmoitukset vaihtuvat tietyin väliajoin. Ilmoituksia voidaan lisätä, muokata sekä poistaa käyttöliittymän kautta.

Jokainen Kymenlaakson ammattikorkeakoulun tiloissa sijaitseva näyttölaite, jossa esityksiä pyöritetään, on varustettu omalla tietokoneella. Jokaiseen monitoriin käynnistettävän esityksen sisältö siis voidaan valita yksilöllisesti. Esitystä käynnistettäessä valitaan kyseisen monitorin sijaintia ja tehtävää vastaavat ilmoitukset käyttöliittymän kautta. Tämän jälkeen sovellus menee tilaan, jossa sisältö päivittyy automaattisesti.

2.2 Ilmoitukset

Vanhasta poiketen uudessa sovelluksessa on kaksi erilaista ilmoitustyyppiä. Toinen on suunnattu opiskelijoille ja toinen oppilaitoksen vierailijoille. Ilmoitusten vaatimukset poikkeavat toisistaan jonkin verran. Molempien ilmoitusten esitysnäkymä on tämän takia myös erilainen. Yhteisiä vaatimuksia ilmoitustyypeille on otsikko- ja tekstikenttien lisäksi ilmoituksen alkamis- sekä päättymisaika, ilmoituksen kielivaihtoehto sekä monitori, johon ilmoitus linkitetään. Ilmoitus voidaan yhdistää vain kyseisen ilmoitustyypin mukaiseen monitoriin ilmoitusten eroista johtuvien ristiriitaisuuksien välttämiseksi. Suurin ero ilmoitustyyppien välillä on se, että opiskelijailmoitusten esitysnäky-

mä on sovitettu vaakatasossa olevaan monitoriin, vierailijailmoitusten pystymalliseen monitoriin.

Opiskelijoille ja oppilaitoksen henkilökunnalle suunnatut ilmoitukset ovat omaa tyyppiään. Näitä esitetään oppilaitoksen tiloissa ja käytävillä sijaitsevissa monitoreissa, kuten tähänkin mennessä. Tähän ilmoitustyyppiin on lisätty mahdollisuus kirjoittaa ilmoitus tarpeen mukaan suomeksi, englanniksi tai molemmilla kielillä. Kaksikielisenä tehdyt ilmoitukset näytetään samassa näkymässä allekkain.

Oppilaitoksen vierailijoille osoitetut ilmoitukset ovat toinen ilmoitustyyppi. Niillä informoidaan kävijöitä tapahtumista ja opastetaan heitä oikeaan paikkaan. Ilmoituksia on tarkoitus esittää oppilaitoksen toimitilojen auloihin sijoitetuissa pystymallisissa monitoreissa. Vierailijailmoitukset voidaan syöttää järjestelmään ainoastaan yhdellä kielellä kerrallaan, sillä suomenkielistä tapahtumaa ei haluta mainostaa englanniksi ja päinvastoin. Tarvittaessa voidaan luoda kaksi erikielistä ilmoitusta, joissa on sama sisältö.

2.3 Monitorit

Jokainen ilmoitus vaatii tiedon siitä, mitä oppilaitoksen kampusta tai osaa se koskee. Vanhassa sovelluksessa tätä käsitettä vastaavat toimipisteet, mutta tässä sovelluksessa käytetään termiä *monitori*. Monitori vastaa paremmin sovelluksen toimintaa kuvaavaa mielikuvaa ja niitä voi olla toimipistettä kohden useampia. Käsite on ainoastaan selvyyden vuoksi. Mitään rajoituksia ei sovelluksessa tämän suhteen ole, ja esimerkiksi Metsolan toimipistettä varten voidaan sovellukseen lisätä vaikka jokaista näyttöpäätettä kohden oma monitorinsa.

Monitorien kohteet on määritelty sovelluksessa uudestaan ja tämänhetkinen lista on esitetty taulukossa yksi. Kyseiset monitorit lisätään tietokantaan valmiiksi. Monitoreja varten on sovelluksessa myös oma näkymänsä, jonka kautta niitä voidaan lisätä tai poistaa.

Taulukko 1: Infomonitorisovelluksen monitorit

Monitorin nimi	Sijainti	Tyyppi
Metsolan kampus	Kotka	Opiskelija
Metsä- ja puutalouden laboratoriot	Kotka	Opiskelija
Jylpyn kampus	Kotka	Opiskelija
Kasarminmäki, Päärakennus	Kouvola	Opiskelija
Kasarminmäki, Paja	Kouvola	Opiskelija
Kasarminmäki, Kirjasto	Kouvola	Opiskelija
Kasarminmäki, Mediakasarmi	Kouvola	Opiskelija
Metsolan kampus	Kotka	Vierailja
Jylpyn kampus	Kotka	Vierailja
Kasarminmäen kampus	Kouvola	Vierailja

2.4 Esitys

Sovelluksesta on voitava käynnistää ilmoitusten esitys. Esityksen ulkoasun on oltava erilainen riippuen kohdemonitorista, koska tietoja esitetään eri tavalla. Kaikkia ilmoituksia ei voi automaattisesti valita esitettäväksi jossakin monitorissa. Vanhasta versiosta poiketen esitysmahdollisuutta ei tarjota peruskäyttäjälle, sillä tämä nähtiin turhana toimintona, joka aiheuttaisi epäjohdonmukaisuutta sovellukseen. Opiskelijailmoitukset esitetään vaakatasoisessa näkymässä ja vierailijailmoitukset pystymallisessa näkymässä. Ilmoitusten voimassaoloaika määräytyy niiden päättymisajankohdan perusteella.

Opiskelijailmoituksia esitetään yksi ilmoitus kerrallaan. Yhden ilmoituksen esilläoloaika määritetään esityksen käynnistysvaiheessa. Ilmoituksen tekstille varattu tila pyritään täyttämään mahdollisimman hyvin fonttikokoa muuttamalla, ettei lyhyen ilmoituksen kohdalla näkymään jäisi tyhjää tilaa. Vierailijailmoituksia esitetään kerralla viisi kappaletta. Ilmoitusten näyttäminen määräytyy ilmoituksen alkamisajankohdan perusteella. Sovellus hakee tietokannasta viiden seuraavan ilmoituksen tiedot ja esittää ne näkymässä allekkain alkamisjärjestyksessä.

2.5 Muut vaatimukset

Esityspohjien graafinen ilme sekä etusivun otsikkoteksti on Viestintäpalveluiden käsialaa. Väreinä näissä käytetään Kymenlaakson ammattikorkeakoulun tunnusvärejä. Samoja värejä hyödynnetään myös sovelluksen muissa osissa, tai vastaavasti käyte-

tään näiden kanssa yhteensopivia, neutraaleja värejä. Molempiin esityspohjiin lisätään myös päivämäärä sekä kellonaika.

Sovelluksessa on oltava myös kuvien lataustoiminto. Toiminnon avulla sovellukseen voidaan tallentaa kaksi erilaista kuvaa, yksi molempia monitoreja kohden. Kuvaa näytetään monitorissa silloin, kun ei ole esitettäviä ilmoituksia. Kuvaa voidaan esittää monitoreissa esimerkiksi kesäaikana. Uuden kuvan tallentaminen korvaa vanhan tiedoston.

3 KEHITYSYMPÄRISTÖ

Web-sovelluksia on mahdollista kehittää joko paikallisesti, mikäli työhön käytettävään tietokoneeseen on asennettu tarkoitukseen soveltuvat ohjelmistot, tai oikeassa palvelinympäristössä, jos tähän on tarvittavat resurssit. Paikallinen kehitys on kokemuksen mukaan epäsuositeltavaa, koska ohjelmistoon jää melko helposti osia, jotka eivät sitten toimikkaan oikein lopullisessa käyttöympäristössä. Tämä voi johtua esimerkiksi ympäristöjen ohjelmistoversioiden eroavaisuuksista. Parempi käytäntö on suorittaa ohjelmiston kehitys ympäristössä, joka on mahdollisimman lähellä lopullisen käyttöympäristön vaatimuksia. Infomonitorisovelluksen kehitykseen käytettiin tässä tapauksessa ICT-LAB:n palvelimia.

3.1 CakePHP-sovelluskehys

Ohjelmistokehityksen yksi periaate on se, ettei pyörää niin sanotusti tule keksiä uudelleen. Näin ollen uutta sovellusta ei haluttu kirjoittaa alusta asti omin käsin, kuten edellisen version kohdalla oli tehty. CakePHP on PHP-sovelluskehys, joka tarjoaa valmiin pohjan web-sovellusten kehitykseen. Se sisältää sisään rakennettua koodia, kirjastoja sekä luokkia ja tarjoaa valmiina laajan kirjon muita kehityksessä usein käytettyjä toimintoja. CakePHP on ladattavissa ilmaiseksi osoitteesta http://cakephp.org. (Syam & Bari 2008, 6.)

Päädyin valinnassani CakePHP-sovelluskehykseen, koska se vaikutti toteuttavan vaatimukseni parhaiten. CakePHP on kevyt ja dokumentointi vaikutti kattavalta. Testikokeilun jälkeen havaitsin sen helposti asennettavaksi ja selkeäksi. Halusin myös perehtyä sovelluskehyksien käyttöön paremmin ja laajentaa taitojani web-ohjelmoijana.

CakePHP on MIT-lisenssin alainen tuote. Tämä tarkoittaa, että ohjelmisto käytettävissä ilmaiseksi ilman rajoituksia kenen tahansa toimesta, kunhan kyseinen lisenssi on sisällytetty ohjelmistosta tehtyihin kopioihin tai sen osiin. (Open Source Initiative.)

3.1.1 MVC-arkkitehtuuri

Monimutkaisia sovelluksia kirjoitettaessa ohjelmakoodista tulee helposti epäjärjestelmällistä. Tällainen koodi on hyvin virhealtista ja pientenkin muutosten teko voi olla vaivalloista. Tilanne vaikeutuu edelleen, mikäli saman sovelluksen parissa työskentelee useita ohjelmoijia, jotka seuraavat kukin omia ohjelmointikäytäntöjään. CakePHP:n käyttämä MVC-arkkitehtuuri määrittelee sovellukselle tiukan rakenteen, joka varmistaa, että kokonaisuus pysyy helpommin hallittavana. (Syam & Bari 2008, 7-8.)

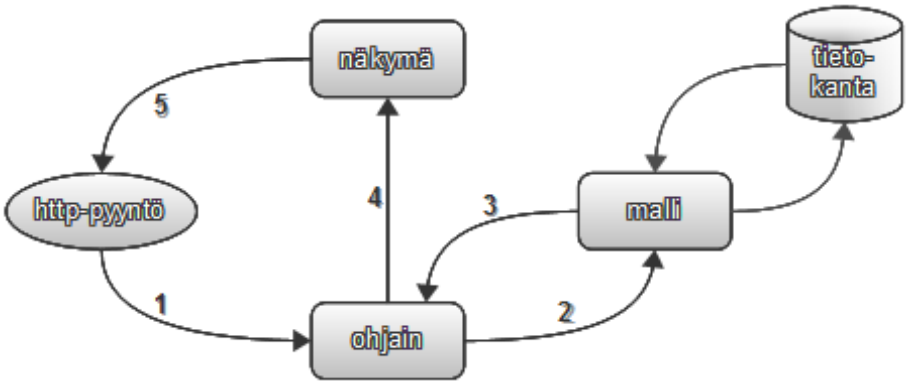
MVC-arkkitehtuuri (MVC, model-view-controller) erottelee sovelluksen käyttöliittymän ja sovelluslogiikan sekä -datan toisistaan. MVC on lyhenne sanoista model (malli), view (näkymä) ja controller (ohjain). (Koskimies & Mikkonen 2005, 142.)

MVC on laajalti käytetty ohjelmistokehityksen arkkitehtuuri tai suunnittelumalli, jossa ohjelmakoodi on jaettu kolmeen edellä mainittuun osaan. Kyseisten osien tarkka tehtävä vaihtelee eri sovelluskehysten kesken. Seuraava osien tehtävien kuvaus on tehty CakePHP:n näkökulmasta. (Syam & Bari 2008, 8.)

Mallit edustavat tietokannan tauluja, joten jokaista taulua kohden on oltava oma mallinsa. Kaikki tietueiden hakemiseen, lisäämiseen, muokkaamiseen ja poistamiseen liittyvä PHP-koodi tulisi sijaita mallissa. Mallissa määritellään myös kyseisen taulun tietueita koskevat datan validointisäännöt sekä relaatiot muihin malleihin. Mallia voidaan pitää sovelluksen datakerroksena. Validointi sekä relaatiot käsitellään myöhemmissä kappaleissa. (Syam & Bari 2008, 8.)

Ohjaimet kontrolloivat sovelluksen logiikkaa. Kun käyttäjä esimerkiksi painaa sivuilla olevaa linkkiä tai tallentaa täyttämänsä lomakkeen, lähetetään palvelimelle http-pyyntö. Pyyntö välittyy oikealle ohjaimelle, joka käsittelee pyynnön ja generoi vastauksen. Ohjainta voidaan pitää sovelluksen logiikkakerroksena, joka välittää pyynnön mallille ja siltä vastauksena saamansa datan vastaavasti näkymälle. (Syam & Bari 2008, 8.)

Näkymät tulostavat vastauksena saadun datan. Ne koostuvat tavallisesti kuvauskielestä, kuten HTML, sekä siihen sisälletystä PHP-koodista. Tilanteesta riippuen tulostus voi olla myös muunlaisessa muodossa. Näkymät ovat sovelluksen esityskerros. Kuvassa yksi on esitetty MVC-arkkitehtuurin toimintaperiaate. (Syam & Bari 2008, 9.)



Kuva 1: MVC-arkkitehtuurin toimintaperiaate

1. Ohjaimelle toimitetaan selaimelta saatu http-pyyntö.
2. Ohjain käsittelee pyynnön ja kutsuu mallia, joka noutaa halutun datan.
3. Malli vastaa ohjaimelle joko lähettämällä tai tallentamalla datan.
4. Ohjain lähettää tulostettavan datan näkymälle.
5. Näkymä tulostaa datan määritellyssä formaatissa.

3.1.2 CakePHP:n konventiot

Yksi CakePHP:n oleellisimmista asioista on ymmärtää sen konventioita. Tämä tarkoittaa käytäntöä, jolla sovelluksen osat tulee nimetä ja mihin ne tulee sijoittaa, jotta CakePHP voi löytää ne ja käyttää niitä oikein. Noudattamalla näitä käytäntöjä, konfigurointia ei vaadita juuri ollenkaan. (Syam & Bari 2008, 10.)

Käytäntöjen opetteluun kannattaa uhrata hetki, niin säästää itsensä loputtomalta konfiguroinnilta sekä loppujuoksussa myös paljon aikaa. Käytäntöjä noudattamalla saavutettu yhtenäinen rakenne helpottaa myös muiden ohjelmoijien osallistumista kehitykseen. Käytännöt eroavat hieman CakePHP:n eri versioiden kesken. Koska tämä sovellus on toteutettu CakePHP:n versiolla 2.0.4, seuraavassa on esitelty version 2.x käytännöt pääpiirteittäin. (Cake Software Foundation b.)

Ohjainten luokkanimet kirjoitetaan yhteen, isoin alkukirjaimin (CamelCase), monikkomuodossa ja loppuun lisätään sana Controller. Käytännön mukaan nimettyjä ohjaimia olisivat siis esimerkiksi PeopleController sekä LatestArticlesController. Tietyn ohjaimen metodia kutsutaan osoiteriviltä ohjaimen nimellä, ilman Controller-sanaa, sekä lisäämällä kyseisen ohjaimen jonkin metodin, eli toiminnon (action), nimi. Esimerkiksi News-ohjaimen latest-metodia kutsuttaisiin kirjoittamalla osoiteriville http://www.example.com/news/latest. Mikäli osoiteriville ei anneta toista parametria, toteutetaan oletuksena ohjaimen index-niminen metodi, jos tällainen on olemassa. Metodille voidaan myös viedä parametreja lisäämällä niitä kutsutun toiminnon perään. Kaksi parametria ottavaa News-ohjaimen latest-metodia kutsuttaisiin siis esimerkiksi näin: http://example.com/news/latest/1/2. (Cake Software Foundation b.)

CakePHP:ssä tiedostot nimetään tavallisesti niiden sisältämien luokkien mukaan. Luokka nimeltä MyClass löytyisi siis tiedostosta MyClass.php. Malleja edustavien luokkien nimet kirjoitetaan myös yhteen, isoin alkukirjaimin, mutta yksikkömuodossa. Malleja vastaavat tietokannan taulujen nimet tulee olla monikossa ja sanat erotettu alaviivalla. Tietokannan taulut kirjoitetaan tavallisesti myös pienin alkukirjaimin. Malli nimeltä BigPerson vastaisi siis tämän käytännön mukaan tietokannan taulua nimeltä big_person. (Cake Software Foundation b.)

Näkymät nimetään kyseisen ohjaimen metodin mukaisesti. News-ohjaimen latest-metodia vastaa siis näkymä nimeltä latest. Näkymien tiedostonimet kirjoitetaan pienillä alkukirjaimilla ja sanat erotetaan alaviivalla. Näkymien tiedostopääte on ctp. Kertauksena siis: tietokannan taulua nimeltä people vastaa malliluokka nimeltä Person, jonka tiedostopolku on /app/Model/Person.php. Ohjainluokan PeopleController sijainti on /app/Controller/PeopleController.php ja ohjaimen index-metodia vastaa näkymä nimeltä index.ctp, joka löytyy kansioista /app/View/People/. (Cake Software Foundation b.)

3.1.3 Vaatimukset ja asennus

CakePHP on suunniteltu asennettavaksi ensisijaisesti Apache web-palvelimelle tämän yleisyyden vuoksi, mutta se on konfiguroitavissa myös muille http-palvelimille. CakePHP 2.x:n vaatimuksena on, että web-palvelimen PHP-versio on vähintään 5.2.8 ja MySQL-versio vähintään 4. MySQL-palvelimen lisäksi on mahdollisuus käyttää myös useita muita tietokantapalvelimia. (Cake Software Foundation d.)

Asennus suoritetaan purkamalla CakePHP:n sisältävä zip- tai tar.gz -paketti www-tilan kotihakemistoon. Tämän jälkeen navigoidaan selaimella sivuston www-osoitteeseen ja vastassa pitäisi olla kuvan kaksi kaltainen näkymä.

Release Notes for CakePHP 2.0.4.

Notice (1024): Please change the value of 'Security.salt' in app/Config/core.php to a salt value specific to your application [CORE\Cake\Utility\Debugger.php, line 713]

Notice (1024): Please change the value of 'Security.cipherSeed' in app/Config/core.php to a numeric (digits only) seed value specific to your application [CORE\Cake\Utility\Debugger.php, line 717]

Your version of PHP is 5.2.6 or higher.

Your tmp directory is writable.

The *FileEngine* is being used for caching. To change the config edit APP/Config/core.php

Your database configuration file is NOT present.
Rename APP/Config/database.php.default to APP/Config/database.php

Kuva 2: CakePHP:n asennusnäkymä

Kaksi ensimmäistä, punaiselle taustavärillä olevaa tekstiä ovat turvallisuusvaroituksia. Niissä käyttäjää kehotetaan muuttamaan core.php-tiedostossa määriteltyjen salt- ja cipherSeed -lukujen arvoja. Salt, eli suola, on mielivaltainen merkkijono, jota käytetään tietoturvaan liittyvien tiivistesummien laskemiseen. Suolattu tiiviste lasketaan suolan ja merkkijonon yhdistelmästä. Vastaavasti merkkijonojen salaamiseen ja salauksen purkamiseen käytetään cipherSeed-arvoa. Tämän arvon tulisi sisältää ainoastaan numeroita, kun taas suola voi koostua mistä tahansa merkeistä. Oletusarvojen käyttäminen altistaa sovelluksen vakaville tietoturvauhille, sillä sama oletusarvo on jokaisessa kyseisen CakePHP-version kopiossa ja näin ollen kenen tahansa selvitettävissä.

Vihreällä pohjalla olevat tekstit ilmaisevat kunnossa olevat asetukset. Kuvassa kaksi sovelluskehys on todennut, että palvelimen PHP-kielen versio vastaa kyseisen CakePHP-version vaatimusta. Web-palvelimella on oltava kirjoitusoikeus sovelluksen app/tmp-hakemistoon. Tavallisesti tämä tarkoittaa Apache-ympäristössä kyseisen hakemiston liittämistä *apache*-ryhmään ja myöntämällä ryhmälle kirjoitusoikeuden (Notaras 2008). Sovellus ilmoittaa myös, että välimuistiin kirjoittamista hoitaa oletusarvoisesti *FileEngine*, joka käyttää varastoinnin kohteena kovalevytilaa.

Keltaisella pohjalla oleva teksti ilmoittaa, ettei tietokannan konfigurointiin vaadittua tiedostoa löydy. Tiedoston pohja on olemassa, mutta sen nimi on vain muutettava ole-

tusarvosta. Tietokantayhteyden muodostamiseksi samaiseen tiedostoon on myös määriteltävä käytettävän tietokannan parametrit. Näiden muutosten jälkeen CakePHP on konfiguroitu ja sovelluskehitys voidaan aloittaa.

3.2 MySQL

Infomonitorisovelluksen tietokantana on käytetty MySQL-relaatiotietokantaa, joka on monipuolinen, joustava sekä suorituskykyinen, sekä hyvin yleinen www-palveluiden taustalla käytetty tietokanta. MySQL noudattaa asiakas-palvelin-arkkitehtuuria. Tämä tarkoittaa sitä, etteivät sovellukset koskaan käsittele tietokantaa suoraan, vaan jonkin palvelinohjelman välityksellä. Tässä tapauksessa käsittely tapahtuu PHP:n kautta. MySQL:n puolesta puhuu sen helppo asennettavuus ja keveys, eikä se vaadi juuri huolenpitoa. Sitä voidaan käyttää niin suurten kuin pientenkin www-palveluiden taustalla. (Heinisuo 2004, 34-35.)

MySQL-palvelin mahdollistaa monipuoliset käyttöoikeusmäärittelyt. Tietokantapalvelimelle voidaan luoda lukuisia eritasoisia oikeuksia tietokantoihin ja tauluihin sisältäviä käyttäjätunnuksia. MySQL:n kyselykieli noudattaa peruskomentojen osalta standardi-SQL-kieltä. MySQL myös laajentaa SQL-komentokantaa omalta osaltaan, tehden siitä erilaisen, mutta ei vähemmän kattavaa. (Heinisuo 2004, 34-35.)

3.3 jQuery

Web-sivujen asiakaspäässä, eli selaimessa, käytetään usein JavaScriptiä hoitamaan erilaisia tehtäviä, kun halutaan esimerkiksi vähentää palvelinpään kuormitusta tai lisätä sivuston dynaamisuutta. jQuery on JavaScript-kirjastoa käyttävä, helposti omaksuttava ja laajalti käytetty, sekä hyvin selainriippumaton tekniikka hyödyntää JavaScriptiä web-sivustolla. Se on suunniteltu helpottamaan sivuston eri elementtien muokkaamista ja manipuloimista. Sillä voidaan esimerkiksi helposti muokata CSS- tyylimäärittelyjä, lisätä animaatioita sekä liittää elementteihin käyttäjän toiminnan seurauksena laukeavia skriptejä. (Verens 2009, 10.)

3.4 Ajax

Aina kun perinteisellä tavalla tehty PHP-sivu tarvitsee dataa palvelimelta, sen on tehtävä uusi http-kutsu ja tämän jälkeen sivu on ladattava uudelleen. Tämä pakottaa käyt-

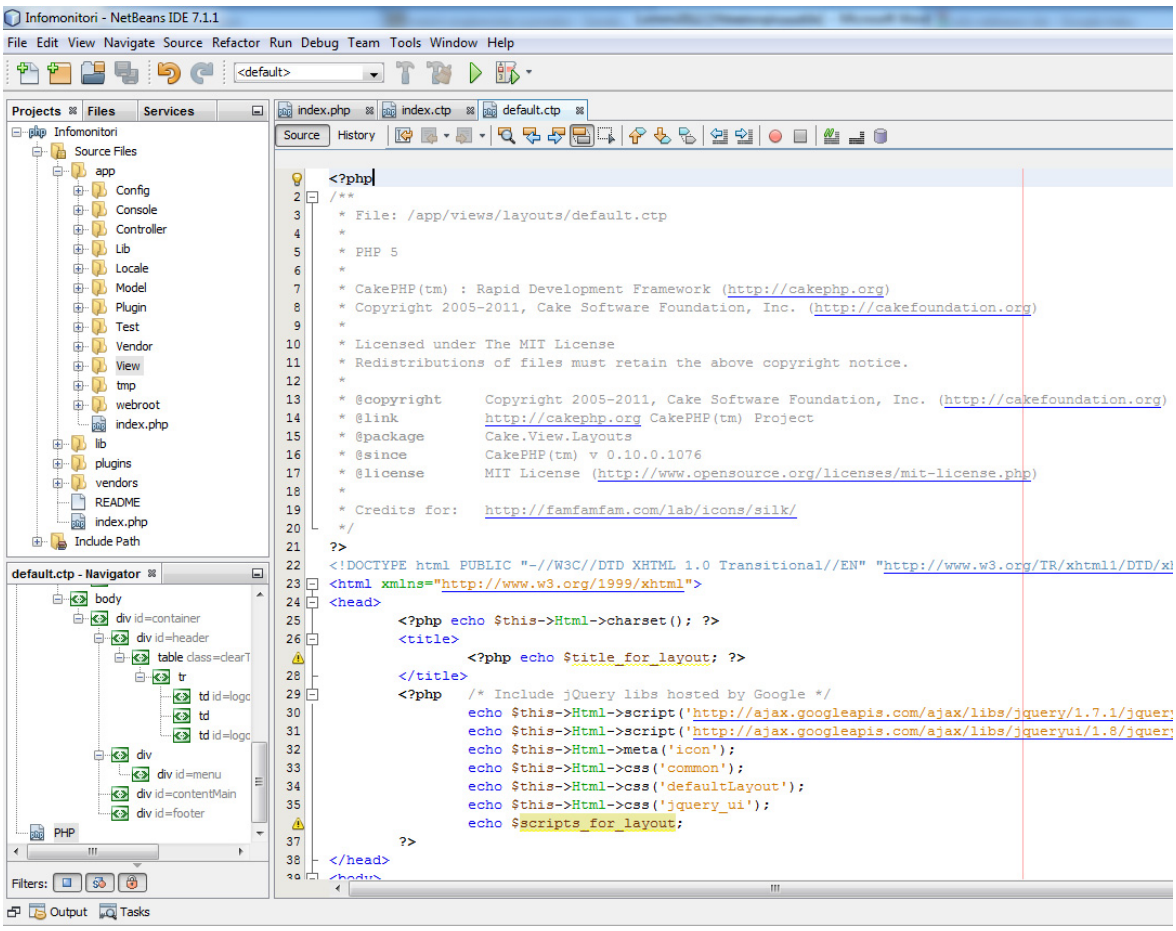
täjän odottamaan koko sivun latautumista. Web-sivustojen vuorovaikutteisuutta voidaan lisätä käyttämällä tekniikkaa nimeltä Ajax. Ajax on lyhenne sanoista Asynchronous JavaScript and XML. Se mahdollistaa taustalla tehtävät palvelinkutsut JavaScriptin avulla ja datan noutamisen ja päivittämisen ainoastaan tarvittavien sivun elementtien osalta ilman, että koko sivua on ladattava uudelleen. Käyttäjä voi jatkaa työskentelyä, kuten lomakkeen täyttämistä, samalla kun Ajax-kutsu suorittaa taustalla esimerkiksi kenttien validointia. (Brinzarea-Iamandi & Darie & Hendrix 2009, 14.)

3.5 Työkalut

Seuraavassa on lyhyt esittely infomonitorisovelluksen tekemisessä käytetyistä ohjelmista. Näiden ohjelmien käyttö ei ollut välttämätöntä, mutta ne on havaittu hyviksi ja niihin on muodostunut henkilökohtainen mieltymys. Ohjelmien tarjoamat ominaisuudet ovat helpottaneet kehitystyötä virtaviivaistamalla toimintoja sekä tarjoamalla avustavaa lisäinformaatiota. Toistuvien työvaiheiden automatisointi tai eliminointi nopeuttaa ohjelmointityötä sekä vähentää virhealttiutta.

3.5.1 NetBeans IDE

NetBeans IDE (Integrated Development Environment) on vapaan lähdekoodin, useita eri ohjelmointikieliä tukeva kehitysympäristö, joka on saatavissa kaikille yleisimmille käyttöjärjestelmille. NetBeans osaa muun muassa korostaa ohjelmakoodia värein, sisentää sitä, merkitä kirjoitusvirheitä, listata ehdotuksia ja täydentää koodia automaattisesti. Se sisältää myös tietokantojen hallintaominaisuudet sekä suuren valikoiman lisäosia ja laajan kehittäjäyhteisön. Kuvassa kolme on esitetty NetBeansin perusnäkymä. (NetBeans.)



Kuva 3: NetBeans IDE perusnäkymä.

3.5.2 MySQL Workbench

Infomonitorisovelluksen tietokanta on toteutettu MySQL Workbench -ohjelmalla. MySQL Workbench on visuaalinen työkalu tietokantojen suunnitteluun, kehittämiseen ja hallinnointiin. Sillä voidaan suunnitella monimutkaisia ER-malleja (entity-relationship), kuten tietokantojen loogista rakennetta kuvaavia kaavioita.

MySQL Workbenchin SQL-editori tarjoaa visuaaliset työkalut SQL-kyselyiden luomiseen, toteuttamiseen ja optimointiin sekä syntaksin värikorostuksen. Ohjelma sisältää myös tietokantayhteyksien hallintapaneelin sekä työkalut palvelinten konfigurointiin, käyttäjien hallintaan sekä tietokannan tilan monitorointiin. (Oracle Corporation a.)

3.5.3 Mozilla Firefox

Kymenlaakson ammattikorkeakoulun sivujen käyttöön suositeltu selain on Mozilla Firefox (Kymenlaakson ammattikorkeakoulu 2010). Myös infomonitorisovelluksen käyttöön suositellaan Firefox-selainta, koska kehitystyö ja suurin osa testausta on suoritettu tällä selaimella. Firefox-selaimeen on saatavilla kattava määrä lisäosia. Näistä web-kehityksessä erityisen hyödyllisiä ovat muun muassa Firebug sekä Web Developer. Firebugin avulla voidaan tutkia web-sivujen elementtejä ja saada arvokasta tietoa sivujen rakenteesta ja tyylimäärittelyistä. Se sisältää myös toiminnot JavaScriptin virheiden etsintään sekä Ajax-kutsujen seuraamiseen. Web Developer on selaimeen asennettava työkalupalkki, josta löytyy kattava määrä erilaisia toimintoja web-kehitykseen. Nämä lisäosat todettiin suorastaan välttämättömiksi tätä sovellusta tehdessä.

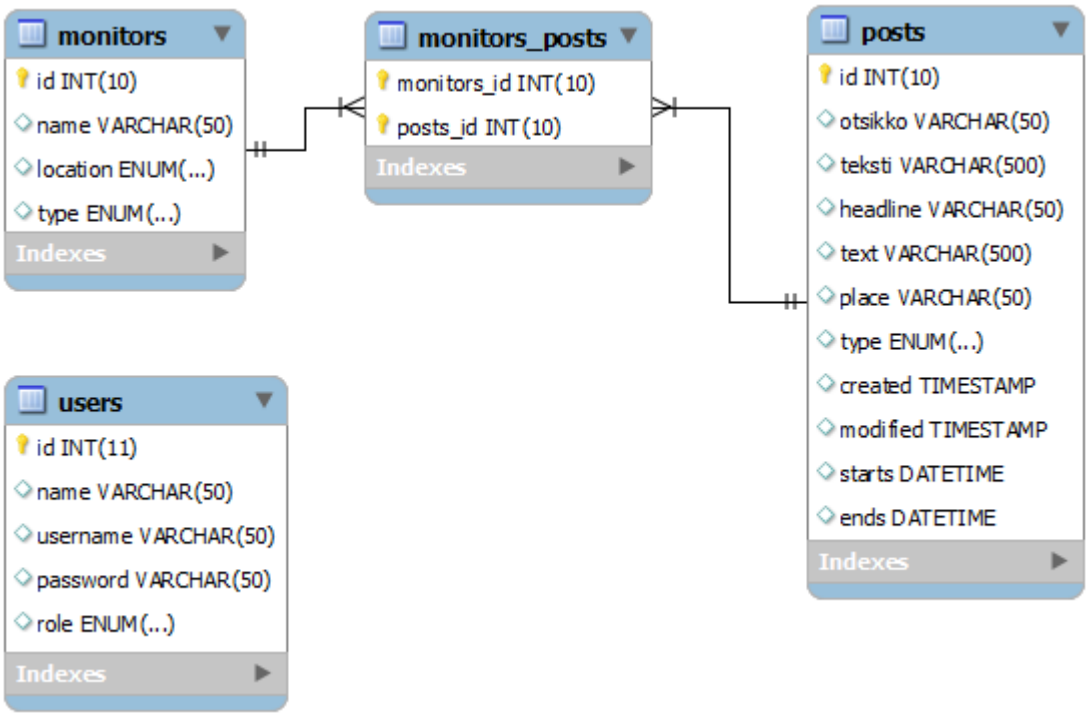
4 TIETOKANTA

Infomonitorisovellus käyttää ilmoitusten ja muun tiedon tallentamiseen MySQL-tietokantaa, joka on esitelty pääpiirteittäin luvussa kolme. Tässä luvussa paneudutaan itse sovelluksen tietokannan sisältöön, rakenteeseen, tietotyyppeihin ja relaatioihin.

4.1 Taulut

Tietokannoissa tieto tallennetaan tauluihin. Taulu luodaan tavallisesti jokaista tiettyä kokonaisuutta varten. Esimerkiksi tämä sovellus käyttää taulua posts, joka sisältää kaikki ilmoitukset. Yksi ilmoitus on taulun tietue, toisin sanoen, jokaista tietokannan tauluun tallennettua ilmoitusta vastaa yksi tietue.

Kuvassa neljä on esitetty MySQL Workbench-ohjelmalla luotu infomonitorisovelluksen tietokannan ER-malli. Taulujen nimet noudattavat CakePHP:n nimeämiskäytäntöjä. Linkkitaulu monitors_posts on niin ikään nimetty käytännön mukaan. Se on yhdistelmä linkitettyjen taulujen nimistä alaviivalla erotettuna. Taulujen nimet kirjoitetaan tässä tapauksessa aakkosjärjestyksessä.



Kuva 4: Infomonitorisovelluksen tietokannan ER-malli

Kuten edellä on mainittu, taulu posts sisältää sovellukseen tallennetut ilmoitukset. Sillä, kuten muillakin tietokannan tauluilla lukuun ottamatta linkkitaulua, on taulun perusavaimena toimiva, kokonaislukutyyppinen id-kenttä. Tämä kenttä yksilöi taulujen tietueet. Taulu sisältää kentät sekä ilmoituksen suomen- että englanninkielisille otsikoille sekä leipäteksteille. Kenttien tietotyypit ovat merkkijonoja (VARCHAR), joille on annettu sulkeissa niiden enimmäispituus. Kenttä place saa arvon vierailijailmoituksessa, ja sillä tarkoitetaan kyseisen tilaisuuden pitopaikkaa. Kenttä type on tyyppiä ENUM, eli sen sallitut arvot on määritelty sulkeissa. Tämä kenttä määrää ilmoituksen tyypin ja voi siis olla joko Student tai Visitor. Kentät created ja modified ovat aikaleimoja, jotka saavat automaattisesti arvonsa, kun ilmoitus luodaan tai sitä muokataan. Ilmoituksen alkamis- ja päättymisajankohdat on tallennettu kenttiin starts ja ends. Näiden tietotyyppi on DATETIME, joka sisältää päivämäärän sekä kellonajan muodossa vvvv-mm-dd hh:mm:ss.

Taulu monitors sisältää tiedot toimipisteistä ja monitoreista. Monitorin yksilöivän id-kentän lisäksi tietue muodostuu monitorin nimestä, sijainnista sekä monitorityypistä. Sijainnille voidaan antaa arvoksi joko Kotka tai Kouvola. Monitorintyyppi voi olla jälleen joko Student tai Visitor.

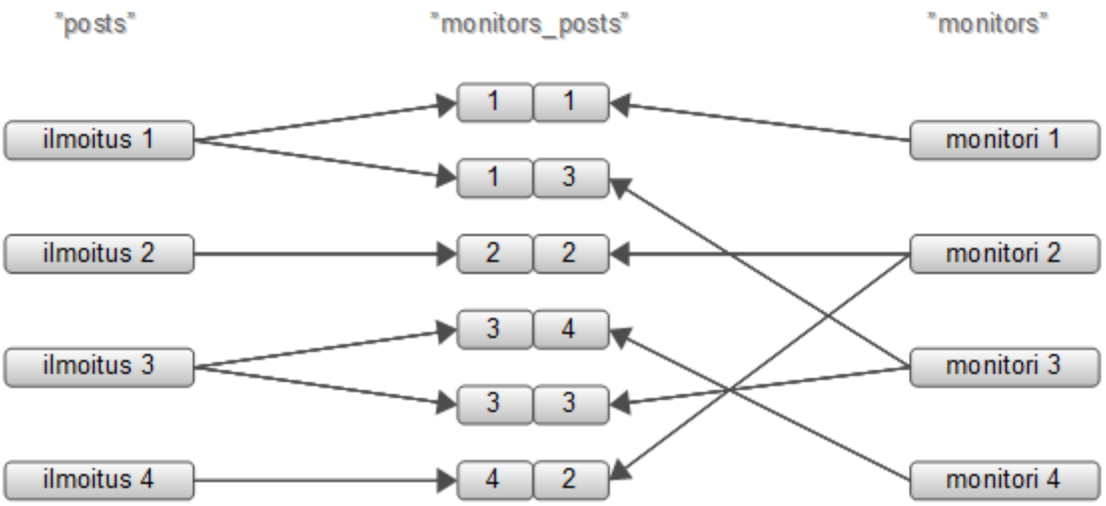
Ilmoitusten ja monitoreiden yhteys määritellään linkkitaulussa monitors_posts. Taulun tietueet koostuvat molempien taulujen perusavainten arvopareista. Ilmoitusten ja monitorien välille on määritelty niin sanottu monta-moneen-suhde. Tämä tarkoittaa, että yksi ilmoitus voi liittyä useaan monitoriin ja vastaavasti yksi monitori moneen ilmoitukseen. Tätä suhdetta on kuvattu CakePHP:n näkökulmasta luvussa Relaatiot.

Tietokanta sisältää myös taulun users, jolla pidetään kirjaa sovelluksen käyttäjistä. Tietue muodostuu käyttäjän nimestä, käyttäjätunnuksesta, salasanasta sekä roolista. Salasana tallennetaan tietokantaan aina salattuna. Käyttäjälle annetun roolin perusteella sovellukseen voitaisiin luoda käyttäjätasokohtaisia oikeuksia. Tällaista erottelua ei ole kuitenkaan tässä sovelluksessa tehty, mutta muutos on helppo toteuttaa tulevaisuudessa, mikäli tarve ilmenee.

4.2 Relaatiot

CakePHP:ssa mallien suhteet määritellään sovelluskehyksen omien assosiaatiotyyppien avulla. Aikaisemmin mainittu monta-moneen-suhde on CakePHP:ssa nimeltään hasAndBelongsToMany, josta käytetään akronyymia HABTM. HABTM-suhdetta on käytettävä silloin, kun halutaan luoda useita yhteyksiä kahden mallin välille eikä yhdistettävien taulujen välinen suhde saa olla poissulkeva. Esimerkiksi ruokareseptejä tallentavassa Recipe-mallissa käytetty Ingredient-mallin tomaatti, ei vaikuta eksklusiivisesti sen käyttämiseen salaattireseptissä. (Cake Software Foundation a.)

Infomonitorisovelluksessa HABTM-suhdetta vastaa ilmoitusten sekä monitoreiden välinen yhteys. Jokainen ilmoitus on yhdistettävä vähintään yhteen monitoriin, mutta sovelluksen näkökulmasta tämä yhteys ei saa ”käyttää loppuun” kyseistä monitoria, koska samaan monitoriin voidaan haluta liittää ilmoituksia myös vastaisuudessa. Sama pätee myös toisin päin: yhden monitorin on voitava liittyä moneen eri ilmoitukseen. Kuvassa viisi on havainnollistettu infomonitorin monta-moneen-suhdetta.



Kuva 5: Infomonitorin ilmoitusten ja monitorien välinen yhteys

4.3 MySQL Events

Tapahtumat (Events) ovat tehtäviä, joita tietokantapalvelin suorittaa automaattisesti niille määritellyn aikataulun mukaisesti. Tapahtuma koostuu yhdestä tai useammasta SQL-lauseesta, jotka suoritetaan haluttu määrä kertoja tai vastaavasti tietyn aikaintervallin välein. Tapahtumien suorittamisesta huolehtii MySQL Event Scheduler. Tuki tapahtumille on ollut MySQL-versiosta 5.1.6 lähtien. (Oracle Corporation b.)

Infomonitorisovelluksen tietokantaan kertyy ajan myötä suuri määrä turhaa tietoa, sillä ilmoitukset eivät häviä mihinkään vanhetessaan, ainoastaan niiden päättymisajankohtaan sidottu esittäminen web-sivustolla lopetetaan. Tämän takia MySQL-palvelimelle on luotu viikoittain suoritettava tapahtuma, joka poistaa vanhentuneet ilmoitukset posts-taulusta. Tapahtuman luomiseen, muokkaamiseen ja poistamiseen käyttäjällä on oltava EVENT-oikeus kyseiseen tietokantaan (Oracle Corporation c).

5 KÄYTTÖLIITTYMÄ

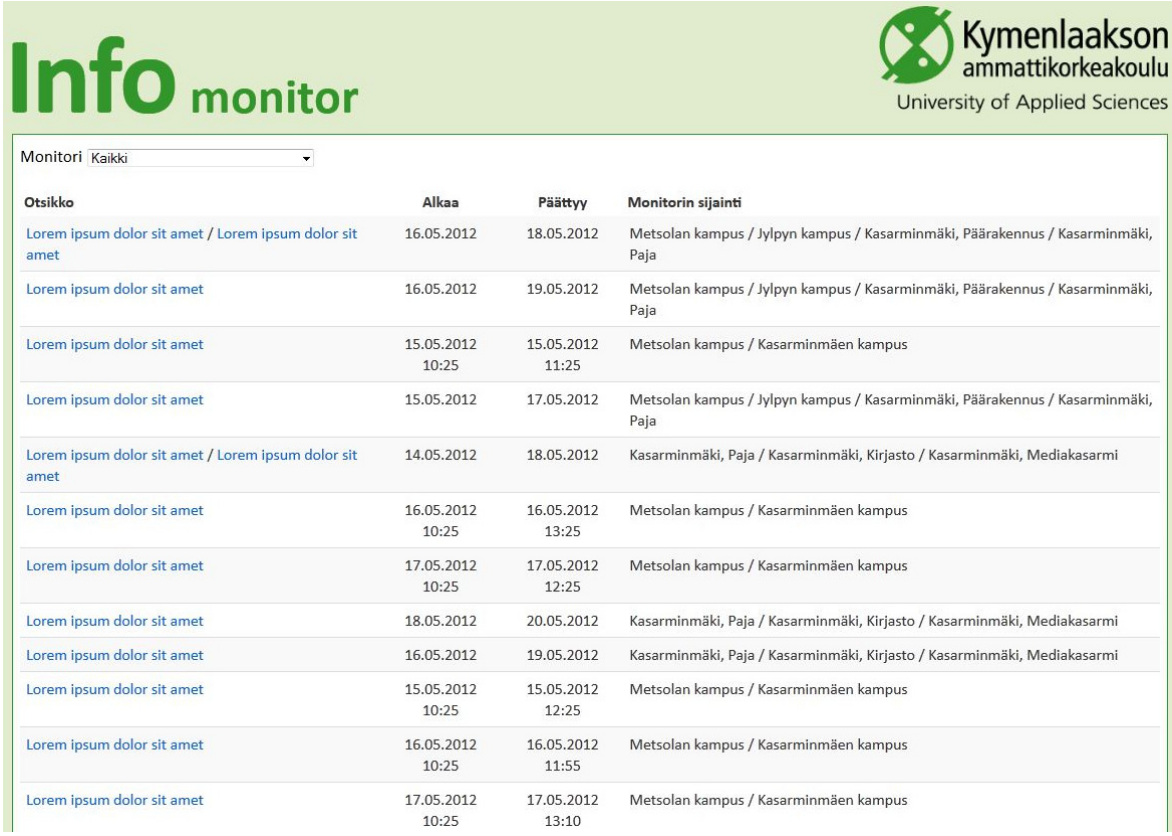
Infomonitorisovellus on pyritty suunnittelemanaan siten, että sen käyttö olisi mahdollisimman suoraviivaista ja helppoa. Tämä on osaltaan sitä, että käyttöliittymän ulkoasulla pyritään ohjaamaan käyttäjää toimimaan oikein, toisaalta myös kontrollia, jolla rajoitetaan käyttäjän mahdollisuuksia toimia väärin. Sovelluksessa on kaksi päänäkymää: kaikille käyttäjille tarkoitettu ja avoin julkinen näkymä (kuva kuusi) sekä kirjautuneille käyttäjille tarkoitettu hallintanäkymä (kuva seitsemän). Peruskäyttäjien näky-

mässä voi ainoastaan nähdä listatut ilmoitukset ja lukea niitä, hallintapuolen näkymä on laajempi sisältäen toiminnot ilmoitusten ja monitoreiden hallintaan.

5.1 Peruskäyttäjän näkymä

Tämä on sovelluksen etusivu, jolle käyttäjä ohjautuu navigoidessaan osoitteeseen info.kyamk.fi. Näkymä on tarkoitettu ilmoitusten lukemiseen internetin kautta ja se on suunnattu kaikille käyttäjille. Näkymä näyttää oletuksena kaikki sovellukseen tallennetut, voimassa olevat ilmoitukset kunkin omalla rivillään. Ensimmäisenä on ilmoituksen otsikko tai otsikot, riippuen siitä, onko ilmoitus tallennettu yksi- vai kaksikielisenä. Ilmoituksen otsikko on linkki, josta ilmoitus avautuu luettavaksi. Rivillä näytetään myös ilmoituksen alkamis- ja päättymisajankohta, sekä ne monitorit, joihin ilmoitus on liitetty.

Ilmoituksia voi suodattaa halutun toimipisteen monitorin mukaan pudotusvalikosta valitsemalla. Valinta laukaisee Ajax-kutsun, joka noutaa valitun monitorin ilmoitukset palvelimelta ja listaa ne näkymään. Ajax-tekniikan ansiosta, ainoastaan *ilmoituslista* päivitetään eikä koko sivua tarvitse ladata uudelleen.



Otsikko	Alkaa	Päättyy	Monitorin sijainti
Lorem ipsum dolor sit amet / Lorem ipsum dolor sit amet	16.05.2012	18.05.2012	Metsolan kampus / Jylpyn kampus / Kasarminmäki, Päärakennus / Kasarminmäki, Paja
Lorem ipsum dolor sit amet	16.05.2012	19.05.2012	Metsolan kampus / Jylpyn kampus / Kasarminmäki, Päärakennus / Kasarminmäki, Paja
Lorem ipsum dolor sit amet	15.05.2012 10:25	15.05.2012 11:25	Metsolan kampus / Kasarminmäen kampus
Lorem ipsum dolor sit amet	15.05.2012	17.05.2012	Metsolan kampus / Jylpyn kampus / Kasarminmäki, Päärakennus / Kasarminmäki, Paja
Lorem ipsum dolor sit amet / Lorem ipsum dolor sit amet	14.05.2012	18.05.2012	Kasarminmäki, Paja / Kasarminmäki, Kirjasto / Kasarminmäki, Mediakasarmi
Lorem ipsum dolor sit amet	16.05.2012 10:25	16.05.2012 13:25	Metsolan kampus / Kasarminmäen kampus
Lorem ipsum dolor sit amet	17.05.2012 10:25	17.05.2012 12:25	Metsolan kampus / Kasarminmäen kampus
Lorem ipsum dolor sit amet	18.05.2012	20.05.2012	Kasarminmäki, Paja / Kasarminmäki, Kirjasto / Kasarminmäki, Mediakasarmi
Lorem ipsum dolor sit amet	16.05.2012	19.05.2012	Kasarminmäki, Paja / Kasarminmäki, Kirjasto / Kasarminmäki, Mediakasarmi
Lorem ipsum dolor sit amet	15.05.2012 10:25	15.05.2012 12:25	Metsolan kampus / Kasarminmäen kampus
Lorem ipsum dolor sit amet	16.05.2012 10:25	16.05.2012 11:55	Metsolan kampus / Kasarminmäen kampus
Lorem ipsum dolor sit amet	17.05.2012 10:25	17.05.2012 13:10	Metsolan kampus / Kasarminmäen kampus

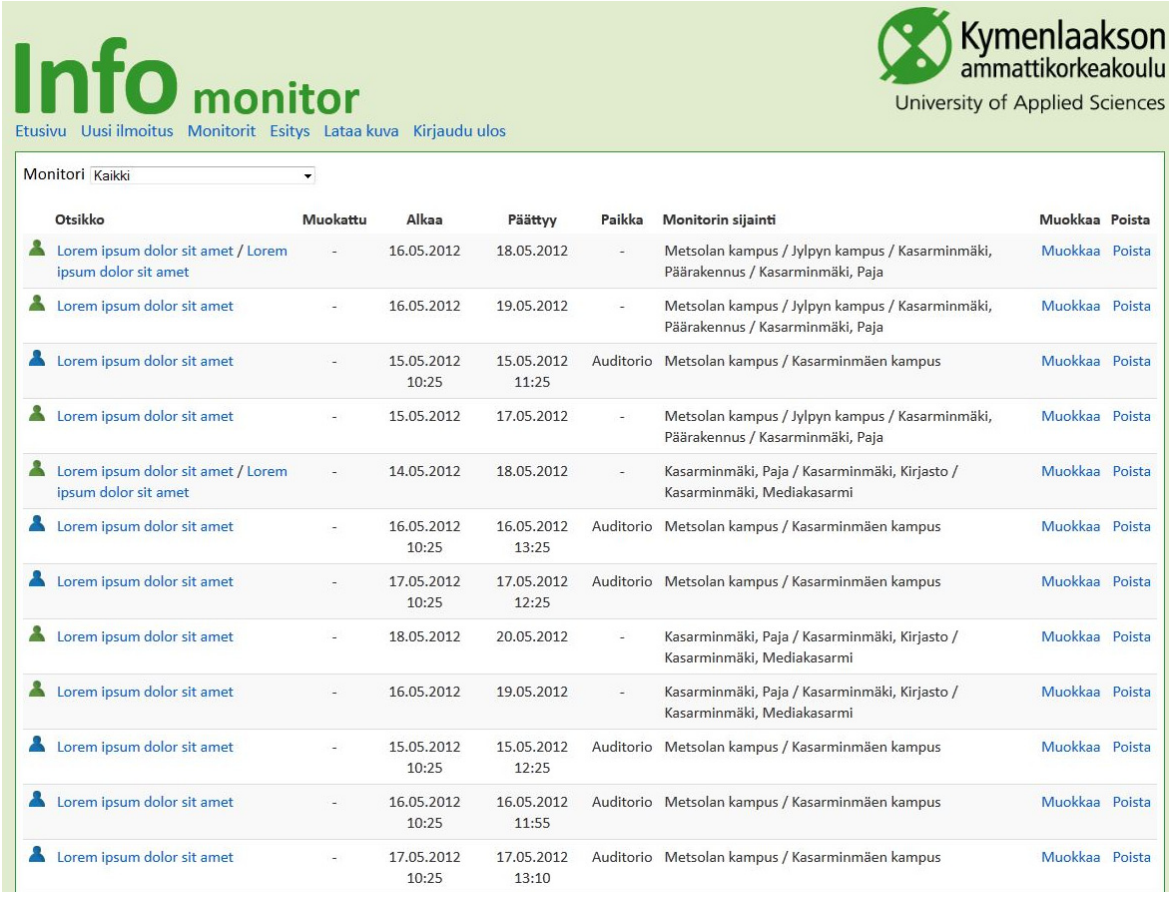
Kuva 6: Peruskäyttäjän näkymä

5.2 Hallintapuolen näkymä

Kirjautuneilla käyttäjillä on hieman peruskäyttäjää laajempi ja enemmän informaatiota sekä toimintoja tarjoava näkymä. Peruskäyttäjästä poiketen, kirjautunut käyttäjä saa käyttöönsä sovelluksen päävalikon. Päävalikko tarjoaa linkit etusivun lisäksi muihin sovelluksen toimintoihin, joihin peruskäyttäjällä ei ole tarvetta tai tietoturvan kannalta oikeutta päästä.

5.2.1 Etusivu

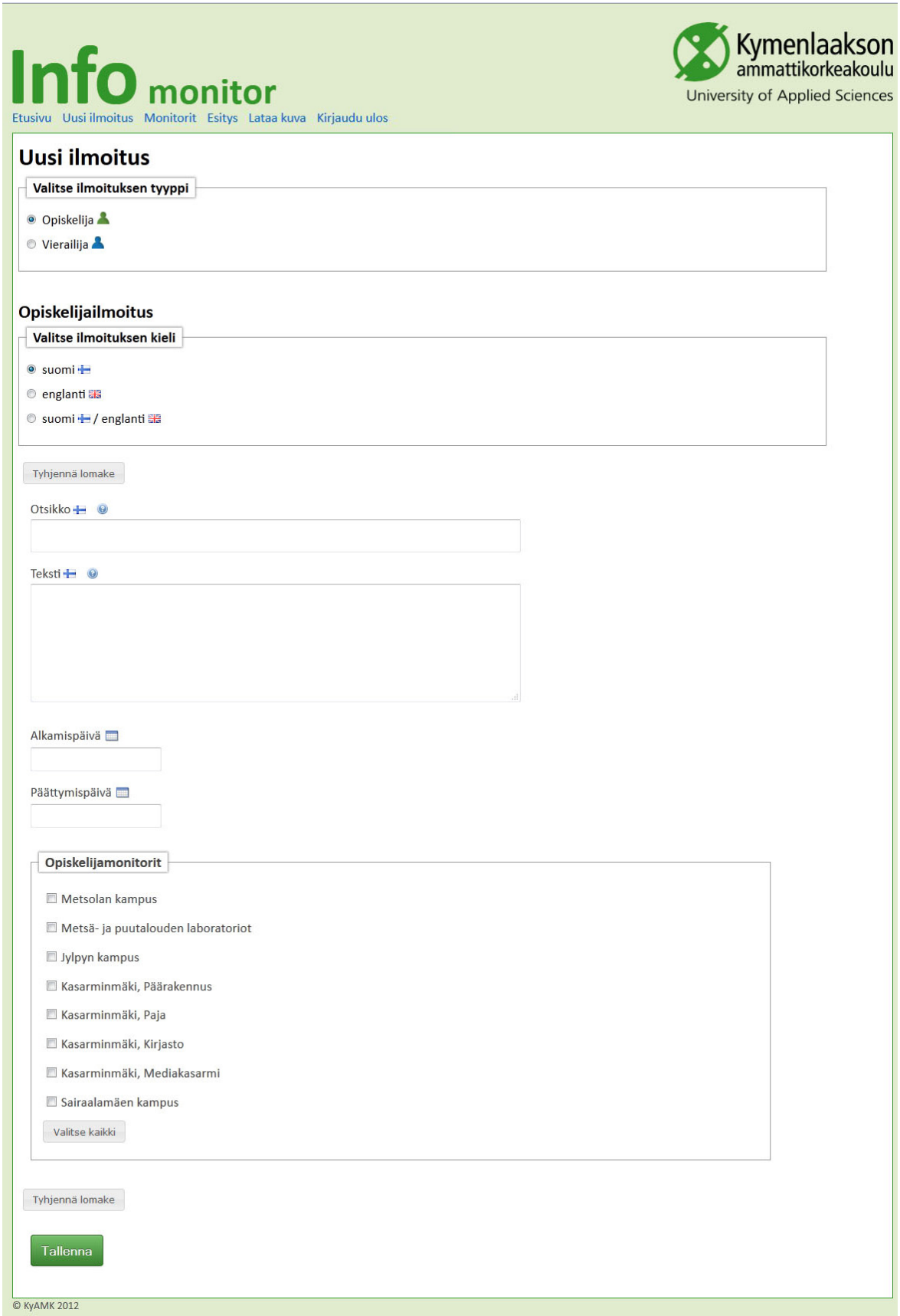
Hallintapuolen etusivu on periaatteessa peruskäyttäjän näkymää vastaava, mutta laajempi. Ilmoituksen edessä on pieni kuvake, joka ilmaisee ilmoituksen tyypin: vihreä vastaa opiskelijailmoitusta ja sininen vieraille suunnattua ilmoitusta. Perusnäkymän tavoin, otsikosta valitsemalla ilmoituksen pääsee lukemaan ja rivillä näytetään ilmoituksen alkamis- ja päättymisajankohdat sekä monitorit, joissa ilmoitusta esitetään. Lisäksi rivillä näytetään ilmoituksen mahdollinen muokkausajankohta. Jos ilmoitusta ei ole muokattu, näytetään sarakkeessa oletuksena tiedon puuttumista ilmaiseva viiva. Sarakkeessa ”Paikka” näytetään vierailijamonitorin ilmoituksessa vaadittu paikkatieto. Opiskelijailmoituksessa ei vaadita kyseistä tietoa, joten näiden kohdalla sarakkeessa on viiva. Lisäksi jokaisen rivin lopussa on ”Muokkaa”- ja ”Poista”-linkit. Ensimmäisestä kyseinen ilmoitus avautuu muokattavaksi uuteen näkymään. Jälkimmäinen poistaa ilmoituksen varmistusikkunan esittämisen jälkeen.



Kuva 7: Hallintapuolen perusnäkymä

5.2.2 Uusi ilmoitus

Sekä opiskelija- että vierailijailmoitukset tallennetaan saman näkymän kautta. Ilmoitusten sisällössä on eroa, joten lomake mukautuu käyttäjän valintojen mukaan joko paljastaen tai piilottaen osia siitä. Mukautuvia osia ovat ilmoituksen tyypistä johtuvat kieli- ja monitorivaihtoehdot, lisäkentät ja tekstikenttien vaihtelevat enimmäispituudet sekä kalenterikomponenttien ominaisuudet. Ilmoitustyypin valinta on johdonmukaisesti ensimmäisenä sivun ylälaidassa. Näkymä on esitetty kuvassa kahdeksan.

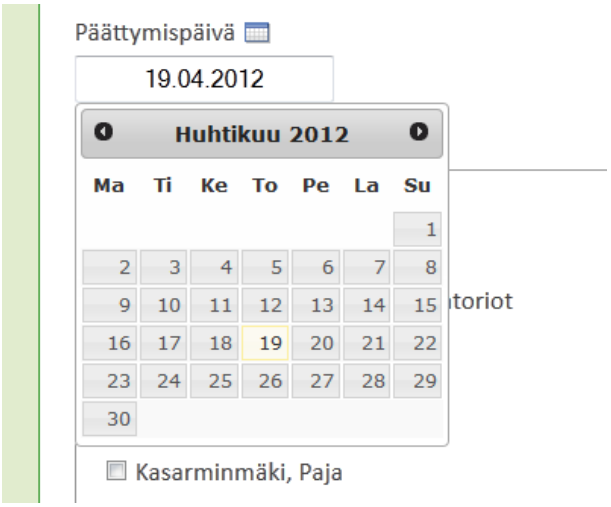


Kuva 8: Uuden ilmoituksen lisäysnäkymä

Sovellus näyttää oletuksena opiskelijailmoitusta vastaavan lomakkeen, jossa kielivaihtoehdon kohdalla on valittuna ainoastaan suomi. Opiskelijailmoituksessa kielivaih-

toehtoja on kolme: suomi, englanti tai sekä että. Valintaa muuttamalla, sovellus tuo esiin tarvittavat tekstikentät käännökselle tai vastaavasti piilottaa ne. Tekstikenttiin syötetään otsikon mukainen sisältö. Pienet Suomen ja Iso-Britannian liput vihjaavat kummalla kielellä kyseisiin kenttiin on tarkoitus kirjoittaa. Tekstikenttien enimmäispituudet on määritetty tietokannassa ja CakePHP osaa käyttää näitä arvoja automaattisesti. Näin ollen käyttäjä ei voi antaa kentässä pidempää arvoa, kun mikä tietokantaan mahtuu. Ilmoituksen tekstiä voi muotoilla lisäämällä rivinvaihtoja. Rivinvaihtojen määrä on rajoitettu jQuery-funktiolla, jotta käyttäjä ei nakuta niitä liikaa ja saa ilmoituksen tekstiä räjähtämään sille varatun alueen yli.

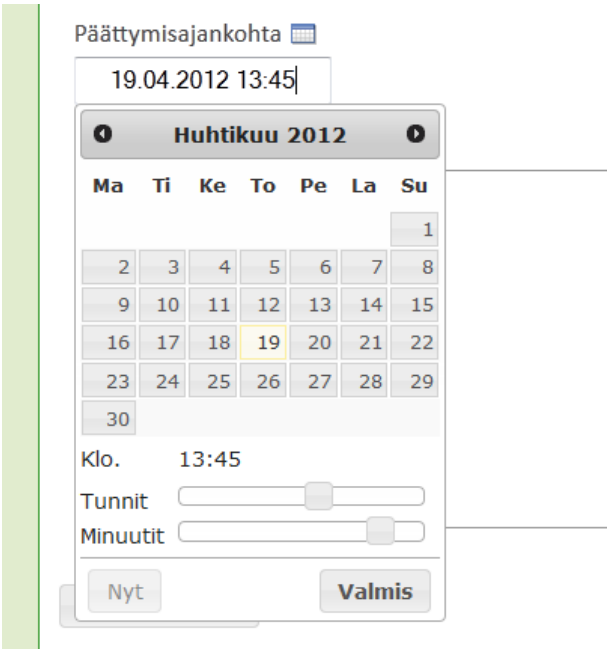
Lippujen vieressä oleva kuvake antaa lyhyen lisäohjeen kentän täytöstä, kun hiiren kursori viedään sen päälle. Alkamis- ja päättymispäivä -kentissä määritellään ilmoituksen voimassaoloaika. Valinta tehdään kentästä aukeavan kalenterikomponentin avulla, joka on esitetty kuvassa yhdeksän. Lopuksi lomakkeella on listattu ne monitorit, joiden tyypiksi on tietokannassa määritetty Student. Kaikkien listan monitoreiden valintaa on helpotettu Valitse kaikki -painikkeella, josta valinta voidaan myös kumota. Sekä lomakkeen alkupuolella että lopussa on painikkeet, joilla kaikki lomakkeen kentät voidaan tyhjentää kerralla.



Kuva 9: Opiskelijailmoituksen kalenterikomponentti

Vierailijailmoitukset eroavat opiskelijailmoituksista vaatimuksiltaan. Kun opiskelijailmoitus voidaan tarvittaessa tehdä kaksikielisenä, vierailijailmoituksen kieli voi olla ainoastaan jompikumpi. Tämä johtuu vierailijailmoitusten esittämistavan luomista rajoitteista: ilmoitusten käyttämä tila on rajallisempi. Lisäksi päätettiin, että on tarpeetonta mainostaa suomenkielistä tapahtumaa englanniksi ja päinvastoin. Myös Teksti-

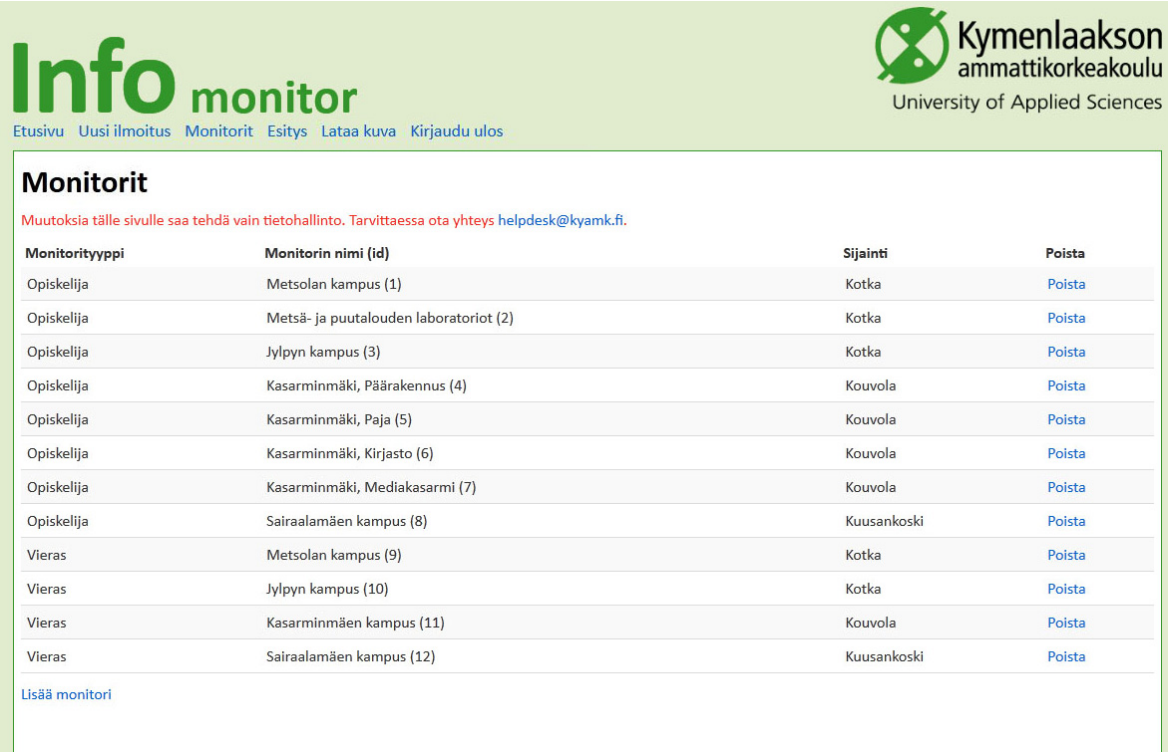
kentän pituus on rajattu opiskelijailmoitusta lyhyemmäksi. Vierailijailmoitus vaatii myös lisäkentän tilaisuuden tai tapahtuman pitopaikalle. Koska myös alkamis- ja päättymiskellonajat on määriteltävä, on näitä ajankohtia vastaavista kentistä aukeava kalenterikomponentti terästetty valinnan mahdollistavilla toiminnoilla. Kyseinen kalenteri on esitetty kuvassa kymmenen.



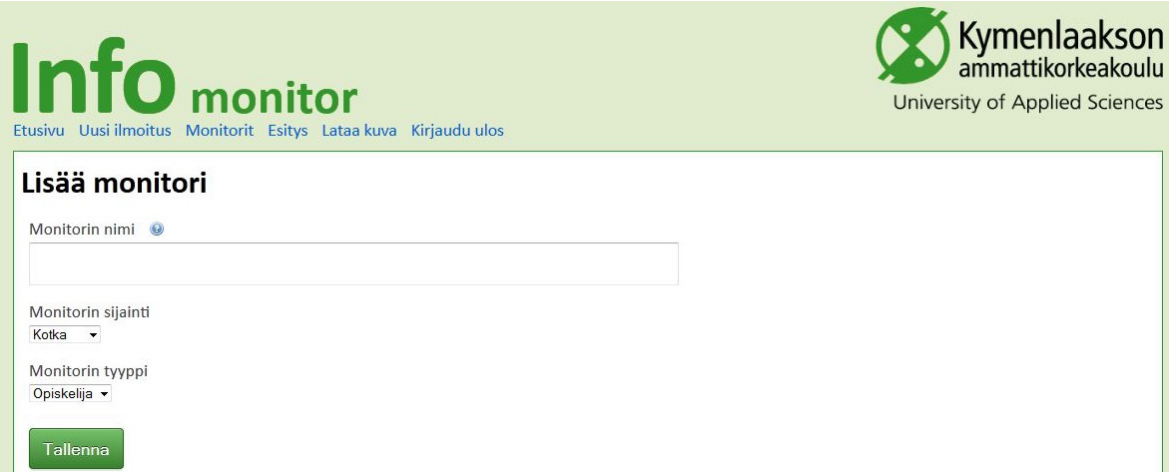
Kuva 10: Vierailijailmoituksen kalenterikomponentti

5.2.3 Monitorit

Uutta ilmoitusta lisättäessä, on siihen liitettävä vähintään yksi monitori. Monitorit ovat toimipistekohtaisia. Vierailijamonitoreja on yksi jokaista kampusta kohden. Opiskelijamonitoreja on enemmän tapauksesta riippuen. Monitorit on listattu omassa näkymässään etusivun ilmoituksia vastaavalla tavalla. Listassa on esitetty monitorin tyyppi, nimi ja sijainti. Monitorin nimen perässä on suluissa monitorin yksilöllinen id-numero. Jokaisella rivillä on myös linkki, josta kyseisen monitorin voi poistaa. Ennen poistamista sovellus esittää varmistuksen toimenpiteestä. Kyseinen näkymä on esitetty kuvassa 11. Listan alalaidassa on linkki, josta aukeaa kuvan 12 mukainen näkymä monitorien lisäämiseen.



Kuva 11: Monitorit

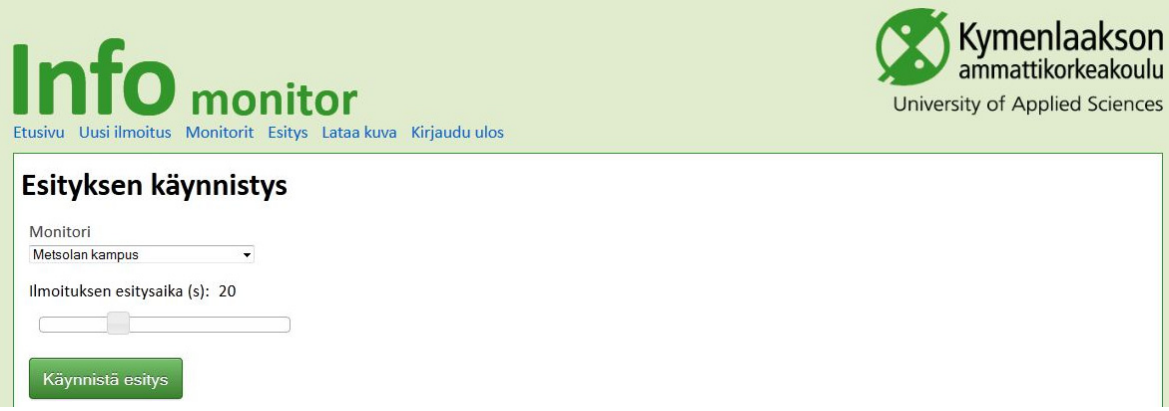


Kuva 12: Monitorin lisääminen

5.2.4 Esitys

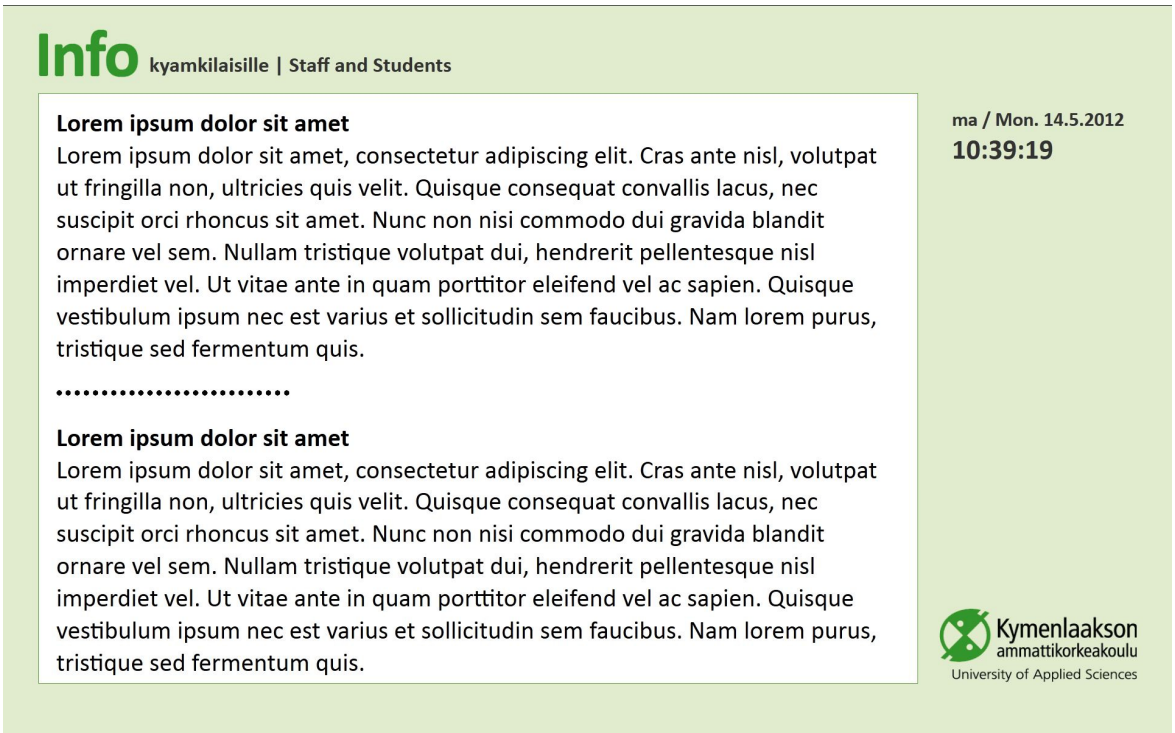
Tämä näkymästä voidaan käynnistää ilmoitusten esitys. Pudotusvalikosta valitaan monitori, johon liitetyt ilmoitukset halutaan näyttää. Liukupalkista voidaan asettaa yksittäisen ilmoituksen esittämisaika sekunteina. Oletuksena ilmoitukset vaihtuvat 20 sekunnin välein. Vierailijamonitoreissa esitysaika tarkoittaa teknisesti intervallia, jolla sovellus käy tarkistamassa tiedot tietokannasta. Näin ollen esitys vaikuttaa staattiselta, mutta päivittyy riippuen tietokantaan tallennettujen ilmoitusten alkamis- ja päättymisajankohdista. Esitysten käynnistäminen kuuluu tietohallinnon tehtäviin. Esitys

voidaan käynnistää myös selaimen osoiteriviltä antamalla posts-ohjaimen show-näkymälle parametreina kyseisen monitorin id sekä ilmoituksen esitysaika sekunteina. Esimerkiksi kirjoittamalla sovelluksen etusivun URL-osoitteen perään /posts/show/1/5. Tämä käynnistäisi Metsolan toimipisteen opiskelijamonitorin esityksen, jossa ilmoitukset vaihtelisivat viiden sekunnin välein. Esityksen käynnistys -näkymä on esitetty kuvassa Kuva 13.



Kuva 13: Esityksen käynnistys

Monitorivalinnan mukaan esitys käynnistyy joko opiskelija- tai vierailijailmoitusten esitysmuotoon. Ensimmäinen on esitetty kuvassa Kuva 14 ja jälkimmäinen kuvassa Kuva 15.



Kuva 14: Opiskelijailmoitusten esitys

Kymenlaakson ammattikorkeakoulu
University of Applied Sciences

Info vierailijalle | Visitors

to / Thu. 19.4.2012
14:17:24

Lorem ipsum dolor sit amet
20.04.2012 klo 14.10-15.10
Auditorio
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras ante nisl, volutpat ut fringilla non, ultricies quis velit. Quisque consequat convallis lacus, nec suscipit orci rhoncus sit amet nunc.

Lorem ipsum dolor sit amet
20.04.2012 at 14.10-16.10
Auditorio
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras ante nisl, volutpat ut fringilla non, ultricies quis velit. Quisque consequat convallis lacus, nec suscipit orci rhoncus sit amet nunc.

Lorem ipsum dolor sit amet
21.04.2012 klo 14.10-15.40
Auditorio
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras ante nisl, volutpat ut fringilla non, ultricies quis velit. Quisque consequat convallis lacus, nec suscipit orci rhoncus sit amet nunc.

Lorem ipsum dolor sit amet
21.04.2012 at 14.10-17.10
Auditorio
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras ante nisl, volutpat ut fringilla non, ultricies quis velit. Quisque consequat convallis lacus, nec suscipit orci rhoncus sit amet nunc.

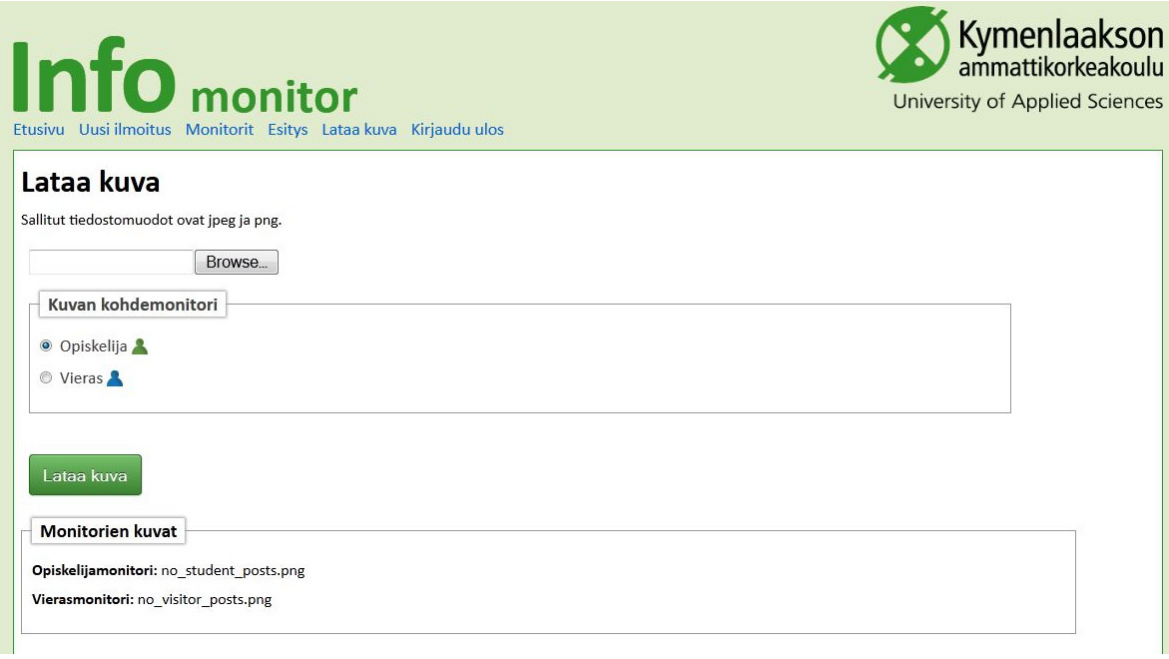
Lorem ipsum dolor sit amet
22.04.2012 at 14.10-16.55
Auditorio
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras ante nisl, volutpat ut fringilla non, ultricies quis velit. Quisque consequat convallis lacus, nec suscipit orci rhoncus sit amet nunc.

Kuva 15: Vierailijailmoitusten esitys

5.2.5 Lataa kuva

Yksi sovellukseen halutuista uusista ominaisuuksista oli mahdollisuus ladata siihen kuva, jota näytettäisiin monitorissa silloin, kun ei olisi voimassa olevia ilmoituksia. Tämä tilanne syntynee lähinnä kesäisin, mutta joka tapauksessa on mielekästä, ettei jokin monitori näytä missään tilanteessa vain tyhjää sivua, jos ilmoituksia ei ole. Kuvien lataus -näkymä on esitetty kuvassa Kuva 16.

Kuvatiedoston lataaminen palvelimelle tapahtuu hakemalla tiedoston polku selauspainikkeen avulla. Koska molemmissa monitorityypeissä on mahdollista käyttää eri kuvaa, on kuvan kohdemonitori myös määriteltävä. Sovellus sallii ainoastaan yhden kuvan monitorityyppiä kohden. Uuden kuvan lataaminen poistaa vanhan ennen uuden tallentamista. Mikäli sovellukseen on jo tallennettu kuvia, näkyvät niiden tiedostonimet alla.



Kuva 16: Kuvan lataus

6 TIETOTURVA

PHP-kielellä kirjoitettujen web-sovellusten sisältö määräytyy usein sivuilla olevien lomakkeiden ja linkkien arvoista ja parametreista. Näin ollen on tärkeää, ettei käyttäjältä saatuun dataan luoteta PHP-ohjelmakoodissa suoraan. Käyttäjältä saadut syötteet on aina tarkistettava ennen kuin niitä käytetään ohjaamaan sovelluksen toimintaa. (Heinisuo 2004, 60.)

6.1 Validointi

Mikäli web-sivun kautta saadun datan muotoa ei ole mitenkään tarkistettu ennen tallennusta, voi se tietokantaan päätyessään aiheuttaa ongelmia. Vääränlainen data voi johtua käyttäjän virheestä tai huolimattomuudesta, tai se voi olla tarkoituksellinen hyökkäys, jolla pyritään aiheuttamaan vahinkoa sivustolle tai tietokannalle. Validoinnilla pyritään varmistamaan, että data on halutun muotoista, järkevää ja turvallista ennen sen ohjelmallista käsittelyä tai tietokantaan tallentamista. Nyrkkisääntönä voidaan pitää, ettei mihinkään käyttäjältä tulleeseen syötteeseen tule luottaa. (Verens 2009, 73.)

Validointi on syytä suorittaa sekä asiakas- eli selainpäässä, että palvelinpäässä. Asiakaspäässä validoinnin rooli on enemmänkin käytettävyyden parantaminen sekä palvelimen kuormituksen vähentäminen. Asiakaspäässä validoinnilla voidaan esimerkiksi tarkistaa, että käyttäjä on muistanut täyttää kaikki web-lomakkeen pakolliset kentät ja että hän on antanut tiedot sellaisessa muodossa, jotta lomake voidaan hyväksyä palvelinpäässä kerralla. Asiakaspäässä validointi tehdään lähes poikkeuksetta JavaScriptiä hyödyntämällä. Koska validointi voidaan tässä tapauksessa ohittaa yksinkertaisesti kääntämällä selaimesta JavaScript pois päältä, on ehdottoman tärkeää, että validointi suoritetaan ainakin palvelinpäässä. (Verens 2009, 73.)

Infomonitorisovelluksessa datan validointi on suoritettu edellä mainitun käytännön mukaisesti. Kun käyttäjä painaa lomakkeen Tallenna-painiketta, suoritetaan tarkistus ensin asiakaspäässä jQueryn validointilisäosaa hyödyntäen. Prosessissa tarkistetaan, että käyttäjä on täyttänyt vaaditut kentät, ei ole valinnut ilmoituksen alkamis- ja päättymisaikoja ristiin ja että ainakin yksi monitori on valittu. Jos jokin tieto ei ole vaatimusten mukainen, sovellus informoi käyttäjää tästä. Sama toistuu, kunnes lomake on täytetty asianmukaisesti.

Validoinnin läpäissyt lomake lähetetään palvelimelle, missä CakePHP huolehtii palvelinpään tarkistuksista. CakePHP sisältää valmiiksi kattavan valikoiman helposti käytettäviä sääntöjä, joita vasten dataa voidaan validoida. Koska lomake on tässä tapauksessa validoitu jo asiakaspäässä, eli sitä ei ole *päästetty* eteenpäin, ennen kuin kaikki lomakkeen kentät on täytetty vaaditulla tavalla, pitäisi sen läpäistä suoraan myös palvelinpäässä tehtävä tarkistus. Mikäli kaikki menee kuvatulla tavalla, saa käyttäjä ilmoituksen onnistuneesta toimenpiteestä.

6.2 Data Sanitization

Validointi on tärkeä osa web-sovellusten tietoturvaa. Mikäli sitä käytetään kuitenkin vain tarkistamaan, ettei käyttäjä ole jättänyt peräänsä tyhjiä kenttiä, jää sovellukseen vielä vakavia aukkoja. On esimerkiksi tarkistettava, ettei käyttäjä, tässä tapauksessa *krakkeri*, käytä haavoittuvuutta hyväkseen ja saa tekstikenttien välityksellä sovellusta imaisemaan vahingollisia skriptejä.

CakePHP sisältää Sanitize-luokan, jolla käyttäjältä saatu data voidaan *siistiä*, eli siitä poistetaan vahingollinen sisältö. Luokkaa voidaan käyttää missä tahansa ohjelmakoodin kohdassa, mutta suositeltava sijainti sille ovat mallit sekä ohjaimet. Käytännössä Sanitize-luokkaa hyödynnetään niin, että jollekin sen monista metodeista annetaan parametrina merkkijono tai kokonainen taulukko, joka halutaan siistiä. Metodi palauttaa datan muodossa, josta haluttu sisältö on poistettu metodista ja sen parametreista riippuen. Metodeja on eri tarkoituksiin, joten halutessa voidaan poistaa merkkijonosta vaikka HTML, jokin haluttu merkki tai paranoidisesti kaikki merkit, jotka eivät ole aplhanumeerisia (a-z A-Z 0-9). (Cake Sofware Foundation c.)

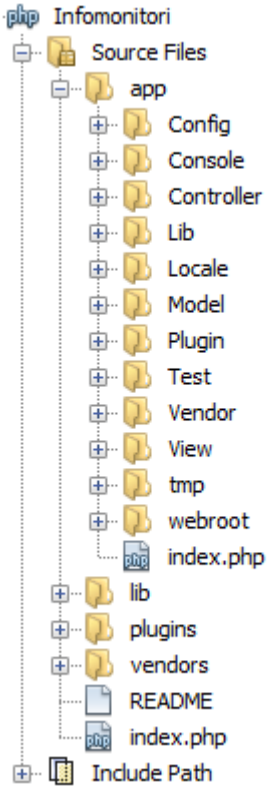
6.3 Object Relational Mapping

CakePHP:ssa on sovelluksen ja tietokannan välistä toimintaa helpottava ominaisuus nimeltä ORM (Object Relational Mapping). Tämä säästää käyttäjän monimutkaisten SQL-lauseiden kirjoittamiselta ja tarjoaa valmiit metodit datan tallennukseen, hakemiseen sekä poistamiseen. ORM:in käyttäminen yhdessä oikeanlaisen taulukkonotaation kanssa suojelee sovellusta tietokantaan kohdistuvilta hyökkäyksiltä, eli niin sanotuilta SQL-injektioilta. (Syam & Bari 2008, 119, Cake Sofware Foundation c.)

7 SOVELLUS

Tässä luvussa käsitellään infomonitorisovelluksen hakemistorakennetta sekä sen toimintaa sovelluskehyksen näkökulmasta. Hakemistorakenteen tunteminen on oleellista, jotta tietää missä mikin tiedosto sijaitsee ja mihin niistä ei kannata koskea. Kansioiden ja tiedostojen runsaus saattaa ehkä yllättää, mutta loppujen lopuksi ohjelmistokehityksen kannalta oleellisia on melko harva.

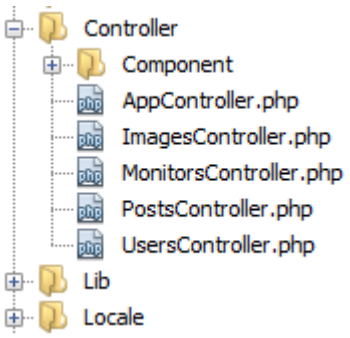
Kuvassa Kuva 17 on esitetty CakePHP-sovelluksen hakemistopuu. Tämä hakemistorakenne saadaan suoraan asennuspaketista CakePHP:tä asennettaessa. Ylimmällä tasolla on neljä kansiota: app, lib, plugins ja vendors. Näistä oleelliset tässä vaiheessa ovat kansiot app ja lib. Ensimmäinen sisältää tämän sovelluksen tiedostot ja toimii ikään kuin työkansiona, jälkimmäinen sisältää CakePHP:n valmiita ydintiedostoja, joita ei ole suositeltavaa muokata. App-kansion sisältämistä kansioista kehitystyön kannalta oleellisimpia ovat Controller-, Model-, View- sekä webroot -kansiot. Asennusvaiheessa sovellukselle annettavat tietokannan parametrit määritellään Config-kansiosta löytyvään database.php-tiedostossa.



Kuva 17: CakePHP-sovelluksen hakemistopuu

7.1 Controller

Sovelluksen ohjaimet sijoitetaan kansioon Controller, jonka sisältö näkyy kuvassa Kuva 18. Näistä jokainen hoitaa omaa kokonaisuuttaan ja tehtäväänsä. AppController on yläluokka, joka periytetään muissa ohjaimissa. Kyseiseen luokkaan voidaan lisätä toimintoja, joiden halutaan olevan kaikkien siitä periytyvien luokkien käytettävissä. Näin periytettävät ohjain-luokat saadaan pidettyä siistimpinä ja yksinkertaisempina, kun samaa sisältöä ei tarvitse kirjoittaa jokaiseen yksitellen. Tämä parantaa myös ohjelmakoodin hallittavuutta.



Kuva 18: Controller-kansion sisältö

Jokainen ohjain siis keskittyy omiin tehtäväkokonaisuuksiinsa. Tässä sovelluksessa nämä kokonaisuudet ovat ilmoitukset, monitorit, käyttäjät sekä kuvat ja näistä jokaista vastaa oma Controller-tiedostonsa. Pääsääntöisesti jokaiseen kokonaisuuteen liittyvät toiminnot hoidetaan omassa ohjain-luokassaan, mutta erottelu hämärtyy jonkin verran Post- ja Monitor -mallien välillä vallitsevan HABTM-suhteen vuoksi. Toiminnot tuntuvat tässä tapauksessa limittyvän, mutta todellisuudessa ne hoidetaan oman mallinsa ja ohjaimensa välityksellä.

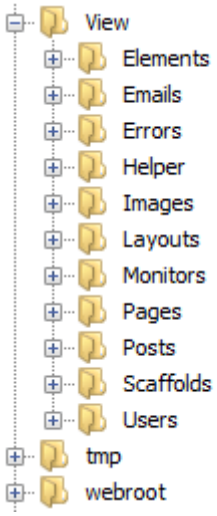
Tämän sovelluksen pääosassa ovat ilmoitukset, joten myös niistä vastaava Posts-ohjain on tärkein sekä *lihavin*, eli se sisältää eniten ohjelmakoodia. Ohjaimen metodit vastaavat etusivujen ilmoituslistojen populoinnista, esityksen sisällön päivittämisestä sekä ilmoitusten esittämisestä, lisäämisestä, muokkaamisesta ja poistamisesta. Sitä voitaisiin pitää sovelluksen *pääohjaimena*, koska sovelluksen www-osoitteeseen navigoiminen reitittyy Posts-ohjaimen toiminnoille, joka huolehtii siis aloitusnäkymän renderoinnista. Mitään hierarkiaa ei kuitenkaan App-ohjainta lukuun ottamatta ohjainten välillä vallitse.

Muut ohjaimet ovat toiminnoiltaan paljon edellistä yksikertaisempia. Monitors-ohjaimen toimintoihin kuuluvat oikeastaan vain monitorien lisäämisen ja poistamisen hoitavat metodit. Monitorin muokkaustoimintoa ei katsottu tarpeelliseksi lisätä, koska monitorin poistaminen ja uuden lisääminen on niin yksinkertaista. Users-ohjain sisältää enemmän metodeja käyttäjien hallintaan. Tätä varten sovellukseen ei ole kuitenkaan rakennettu vastaavia näkymiä, koska sovellusta käytetään toistaiseksi korkeintaan muutamalla yhteisellä tunnuksella. Mikäli vastaisuudessa käyttäjiä halutaan voida lisätä järjestelmällisesti, ovat tarvittavat näkymät helppo tehdä. Lisämuutoksilla sovellukseen voidaan tarvittaessa rakentaa myös eri käyttäjätason rooleja ja näitä vastaavia

näkymiä. Images-ohjain on sovelluksen suppein ja se huolehtii ainoastaan kuvatiedoston lataamisesta palvelimelle.

7.2 View

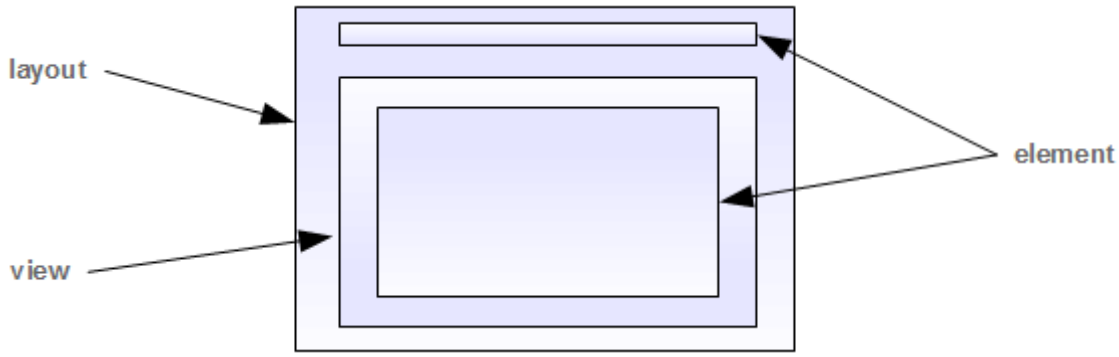
Periaatteessa kaikki, mitä käyttäjä näytöllään näkee, sijaitsee View-kansiossa. Kansion sisältö on esitetty kuvassa Kuva 19. View-kansio sisältää valmiiksi useita kansioita, joiden joukkoon luodaan jokaista sovelluksen ohjainta vastaava kansio. Tämä kansio sisältää kyseisen ohjaimen kaikki näkymät. Posts-kansio sisältää esimerkiksi ohjaimen metodeja add, edit ja view vastaavat näkymät. Kaikkia ohjaimen metodeja varten ei tarvitse olla omaa näkymäänsä eikä näkymän nimen tarvitse välttämättä vastata ohjaimen metodin nimeä. Yhtenäinen nimeäminen automatisoi toimintoja, mutta metodista poikkeava näkymän nimi voidaan määritellä myös ohjaimessa. Ohjaimen metodia kutsuttaessa se siis pyrkii automaattisesti renderoimaan samannimisen näkymän ellei muuta ole määrätty



Kuva 19: View-kansion sisältö

Layouts-kansio pitää sisällään ulkoasupohjat. Nämä ovat sivun *raameja*, joiden sisään näkymät renderoidaan. Karkeasti linjaten sivun layout pitää sisällään sen, mitä html-tagien väliin on kirjoitettu. Se pitää sisällään siis sekä head- että body- elementit. Yhtä karkeasti jatkaen, body-elementin sisältö saadaan näkymästä. Asia ei ole aivan näin mustavalkoinen ja jotta se olisi sekavampi, on vielä yksi sivuston rakennuspalikan tyyppi, joita kutsutaan elementeiksi. Elementit ovat osia, joita voidaan käyttää sekä ulkoasupohjien että näkymien sisällä. Sovelluksen sivu voi koostua yhdestä ulko-

asupohjasta, yhdestä näkymästä sekä useista elementeistä. Kuvassa Kuva 20 on pyritty selventämään sivuston rakennetta.



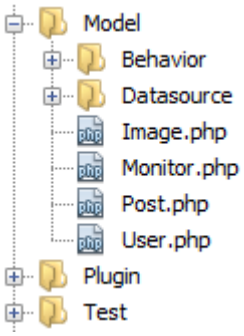
Kuva 20: CakePHP-sivun rakenne

Infomonitorissa on käytetty kolmea eri ulkoasupohjaa. Samaa oletuspohjaa on käytetty kaikissa muissa näkymissä, paitsi ilmoitusten esitysnäkymiä varten on luotu omat pohjansa niiden poikkeavista ominaisuuksista johtuen. Näin ollen näkymien osat on voitu helpommin järjestellä toisistaan riippumatta.

Elementeillä on helppo luoda sovellukseen dynaamisuutta ja näitä onkin käytetty infomonitorissa laajalti. Esimerkiksi kun käyttäjä suodattaa ilmoituksia valitsemalla monitorin etusivun pudotusvalikosta, sovellus suorittaa Ajax-kutsun, jonka palautteena päivitetään näkymän sisällä olevaa elementtiä. Periaatteessa voitaisiin yksinkertaisuuden vuoksi päivittää itse näkymä, mutta peruskäyttäjän ja ylläpitäjän näkymien eroavaisuudet pakottivat rakentamaan verrattain monimutkaisemman ratkaisun. Selventäen tämä tarkoittaa sitä, että itse näkymä on sama molemmilla, mutta sen sisältö tulee eri elementeistä, joiden ominaisuudet poikkeavat toisistaan jonkin verran.

7.3 Model

Sovelluksen mallit sijaitsevat kansiossa Model, jonka sisältö on esitetty kuvassa Kuva 21. Mallit edustavat tietokannan samannimistä taulua, mutta mallia vastaavaa taulua ei ole pakko olla tietokannassa. Jokaista ohjainta kohden on kuitenkin oltava oma mallinsa.

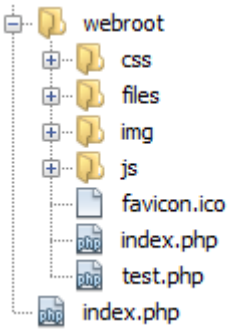


Kuva 21: Model-kansion sisältö

Ilmoituksia edustavan Post-mallin sekä monitoreita edustavan Monitor-mallin välille on määritelty monta-moneen- eli HABTM-suhde. Molemmat mallit sisältävät myös omat sääntönsä lomakkeilta saapuvan datan validointiin. Post-mallissa on oma metodinsa, jolla tarkistetaan, että ainakin yksi monitori on valittu. Samassa mallissa on myös ennen ilmoituksen tallennusta suoritettava metodi, joka muokkaa lomakkeelta saadut päivämäärä- ja kellonaikatiedot oikeaan muotoon ennen niiden tallentamista tietokantaan. User-mallissa on edellä mainittua käyttäjien mahdollista lisäämistä varten määritellyt validointisäännöt valmiina. Mallissa on myös ennen tallennusta suoritettava metodi, joka suorittaa salasanan *kryptaamisen*. Näin ollen tietokantaan ei päädy selväkielisiä salasanoja, mikä olisi vakava tietoturvariski. Myös Image-malli on määritelty, vaikka sitä edustavaa taulua ei tietokannassa olekaan. Mallissa on määriteltävä muuttuja, joka mahdollistaa taulun puuttumisen. Muuten sovellus antaa virheilmoituksen ja sen suoritus keskeytetään.

7.4 webroot

View-kansion ohella eniten sovellukseen ulkoasuun vaikuttavia osia on kuvassa Kuva 22 esitetyssä webroot-kansiossa. Tämä kansio sisältää sovelluksen CSS-tyylitiedostot, kuvat sekä JavaScript-tiedostot.



Kuva 22: webroot-kansion sisältö

CSS-tiedostot sisältävät sivuston ulkoasun muotoiluun, kuten väreihin, fontteihin, kuvien sijoitteluun ja elementtien kokoon sekä keskinäisiin suhteisiin vaikuttavia määrityksiä. Jokaista kolmea ulkoasupohjaa varten on oma CSS-tiedostonsa. Tämä sen takia, että pohjat sisältävät suurin piirtein samat html-elementit, joiden tyylimääritykset kuitenkin poikkeavat toisistaan. Tyylimääritysten kokoaminen samaan tiedostoon olisi tehnyt muotoilusta hankalaa sekä vaikeasti hallittavaa. Kaikille sivuille yhteiset tyylimääritykset on koottu neljänteen CSS-tiedostoon, joka on kaikilla ulkoasupohjilla sama. Jokaista sivua kohden käytetään siis ainakin kahta eri CSS-tiedostoa. CSS-kansio sisältää lisäksi muun muassa jQuery-komponenttien muotoiluun tarkoitettuja tyylitiedostoja.

Img-kansio sisältää sivustolla käytettävät kuvat, kuten logot, liput, sekä kalenteri- ja infokuvakkeet. Kansio pitää sisällään myös omat kansionsa sovelluksen kuvanlataustoiminnon kautta tallennetuille kuvatiedostoille. Js-kansio sisältää sovelluksen JavaScript-tiedostot. JavaScriptiä käytetään monessa kohtaa sovellusta suorittamaan erilaisia tehtäviä ja funktiot on kirjoitettu suurelta osin käyttäen jQueryä. Mikäli funktiota on käytetty ainoastaan yhdessä kohtaa sovellusta, se on voitu kirjoittaa myös suoraan kyseiseen näkymään tai elementtiin. Yleiskäyttöisemmät JavaScript-tiedostot voidaan ladata kyseisestä kansiosta käyttäen CakePHP:n HtmlHelper-luokkaa.

8 KÄYTETTÄVYYDEN TESTAAMINEN

Tärkeä osa ohjelman kehitystyötä on aina myös sen ominaisuuksien ja toiminnan testaaminen. Testaamisella pyritään etsimään virheitä, joiden seurauksena ohjelma ei toimi toivotulla tavalla sekä varmistamaan, että se ylipäänsä tekee asioita, joita siltä vaaditaan.

Suoritin sovelluksen testausta koko kehitystyön ajan. Periaatteessa aina kun tein jonkin muutoksen tai lisäyksen ohjelmakoodiin, testasin sen vaikutusta selaimella. Tämä teki kehitystyötä osaltaan melko hidasta. Moni asia kehitysympäristössä oli kuitenkin minulle uutta, joten oli turvallisempaa tehdä pieniä muutoksia kerrallaan. Näin jonkin muutoksen toimimattomuus oli helpompi jäljittää takaisin päin. Pyrin saamaan lisättävän ominaisuuden toimimaan ensiksi jollakin tavalla. Kun perustoiminnallisuus oli kohdallaan, aloin vasta lisäämään ehtoja ja *lihaa* toiminnan ympärille. Liian suurta osaa kerralla tehdessä myös siihen liittyvien muuttujien määrä kasvaa ja virheitä on vaikeampi jäljittää.

8.1 Lomakkeiden testaaminen

Suuri osa testaamisesta liittyi sovelluksen lomakkeisiin, jotka myös ovat laaja osa sovelluksen ja käyttäjän vuorovaikutusta. Tämä oli pääsääntöisesti käsin tehtävää testausta, eli annoin kentille sekä hyväksyttäviä, että vastaavasti myös vääriä arvoja ja seurasin, oliko ohjelman toiminta toivottua. Koska sovelluksen toimintaperiaatteen mukaisesti ainoastaan voimassa olevat ilmoitukset näytetään, sovellukseen syötetyt ilmoitukset poistuivat näkyvistä aikanaan. Erilaisten ilmoitusten syöttäminen sovellukseen pelkästään testitarkoitusta varten on turhauttavaan ja aikaa vievää työtä. Tämän välttämiseksi loin joukon erilaisia SQL-tiedostoja, joiden suorittaminen loi tietokantaan nopeasti ajantasaista sisältöä. Tällä tavalla pystyin testaamaan helposti kenttien enimmäispituuksia, päivämääriä sekä sovelluksen toimintaa ja sain kiinni monia ajatusvirheitä ja loogisia aukkoja, joiden esille tulo olisi muuten ollut melko epätodennäköistä. Havaitsin, että testaamisessa volyymi, jolla se tehdään, on tässä tapauksessa valttia.

On myös suositeltavaa, että testausta suorittaisi joku muukin kuin itse kehittäjä. Jokainen ihminen ajattelee omalla tavallaan, joten hänen näkemyksensä ohjelman käytöstä voi olla aivan erilainen kuin toisen. Itse ohjelman kehittäjä tuntee sen perinpohjaisesti, joten hän myös varmasti testaa sitä helposti liian *oikein*. Ulkopuolinen testaaja, joka ei tunne ennestään sovelluksen toimintaa, voi löytää siitä arvokkaita puutteita odottamattomalla toiminnallaan. Tämä on oleellista virheiden löytymisen kannalta, mutta myös käytettävyyteen liittyvien epäkohtien osalta. Testaamista on suoritettu myös toimeksiantajan puolelta ja saatuun palautteeseen on pyritty reagoimaan ja siirtämään toivomuksia myös sovellukseen.

8.2 Kontrollointi

Infomonitorissa käyttäjän toimintaa on pyritty myös kontrolloimaan niin, ettei epätoivottujen toimintojen tekeminen olisi edes mahdollista. Pelkkinä tekstikenttinä toteutetut päivämääräkentät ovat hyvä esimerkki potentiaalista ongelmista käytettävyydessä. Mikäli kenttiin kirjoitettavan päivämäärän muotoa ei ole mitenkään kontrolloitu, voi käyttäjä kirjoittaa sen missä muodossa tahansa omasta näkemyksestään riippuen. Kuten aikaisemmin on todettu, väärässä muodossa oleva data voi aiheuttaa ongelmia tietokannassa. Tietokanta ei välttämättä nirsoile niinkään datan muodon suhteen, mutta ongelmia voi tulla siinä vaiheessa, kun data noudetaan kannasta ja tulostetaan näytölle tai tehdään jokin haettuun arvoon perustuva toiminto. Esimerkiksi näytettävät ilmoitukset noudetaan etusivulle sillä perusteella, onko niiden päättymispäivämäärä sama tai myöhempi kuin nykyinen päivämäärä. Jos päivämääräkentän arvo on väärässä muodossa, ilmoitus ei välttämättä tulostu ollenkaan. Samat päivämääräkentät määrittelevät myös vierailijailmoituksissa tapahtumien ajankohdat ja varsinkin tässä tapauksessa on merkitystä sillä, että ne ovat mitä kuuluu.

Väärässä muodossa annettavien syötteiden välttämiseksi infomonitorin päivämääräkentissä käytetään kalenterikomponentteja. Kalenteri avautuu, kun käyttäjä kohdistaa hiiren kursorin mainittuun kenttää. Kalenterista päivämäärä on helppo valita ja se antaa syötteen automaattisesti oikeassa muodossa. Valitun päivämäärän muotoilu käsin on myös estetty. Näin data saadaan ensinnäkin näkymään oikeassa muodossa sekä edelleen muokattua tietokannan kelpuuttamaan muotoon luotettavasti. Samaa ajattelutapaa sovelletaan myös Esitys-näkymässä käytetyn liukuvalitsimen kohdalla: valitsimella saadaan haarukoitua haluttu arvoväli eikä käyttäjä voi tällöin lisätä esitysajaksi jotain tähtitieteellistä. Edellä kuvattujen valitsimien käyttö poistaa myös tarpeen tehdä asiakaspään validointia näiltä osin ja samalla sovelluksen käytettävyys paranee.

9 HYLÄTYT RATKAISUT JA KEHITYSEHDOTUKSET

Infomonitorista saatiin aikaan perusvaatimukset täyttävä kokonaisuus. Monenlaisten vaiheiden kautta sovelluksen tekemiseen saatiin uppoamaan sen verran paljon aikaa, ettei kakkossijalla olleita lisävaatimuksia enää ehditty alkaa miettiä saati toteuttamaan. Lisäominaisuudet olisivat tuoneet sovellukselle lisäarvoa ja niiden toteuttaminen olisi jälleen lisännyt omaa osaamista, mutta mielestäni sovellus täyttää tehtävänsä ilman niitäkin ja koin tärkeämpänä panostaa enemmän hyvin toimivaan ja mietittyyn perus-

sovellukseen. Sovellus on nyt toteutettu uudella tekniikalla ja se on rakenteeltaan helpommin muokattavissa. Tarvittaessa lisäominaisuudet voidaan antaa seuraavan kehittäjän toteutettaviksi.

Ilmoitusten puuttuessa esitettävien kuvatiedostojen lataustoiminto sallii vain yhden kuvan lataamisen monitorityyppiä kohden. Tämä oli määrittelyn minimivaatimus ja kaikista yksinkertaisin vaihtoehto toteuttaa. Ominaisuutta voisi parantaa tulevaisuudessa rakentamalla kuvien hallintaan kunnollisemman käyttöliittymän, joka sallisi useamman kuvan lataamisen ja näyttäisi kaikki palvelimelle tallennetut kuvat esikatselukuvineen. Tarpeen mukaan kuvatiedoston valintaa voitaisiin helposti vaihtaa. Kehittynyt ratkaisu olisi myös mahdollisuus valita jokaista monitoria varten oma kuvansa.

Sovellukseen toivottiin lisäksi toimintoa, jolla opiskelijailmoituksiin voitaisiin lisätä kuvasisältöä. Tämä toimisi esimerkiksi niin, että mahdolliselle kuvalle rajattaisiin tila ilmoituksen oikeasta reunasta ja vastaavasti teksti tiivistyisi vasemmalle jääneellä palstalla. Ominaisuus asettaisi sovellukselle paljon lisävaatimuksia. Ensinnäkin olisi otettava huomioon kuvan lähde: ladataanko kuvat erikseen sovellukseen, jolloin tätä varten tarvittaisiin oma käyttöliittymänsä, vai noudetaanko kuvat jostakin olemassa olevasta varastosta, jolloin vastaavasti tämä yhteys olisi toteutettava. Kuvan koolle ja laadulle olisi asetettava järkevät rajoitukset. Paras tapa olisi sisällyttää sovellukseen oma kuvan rajaustoimintonsa sekä lopputuloksen esikatseluominaisuus. Ilmoitustekstin muotoiluun toivottiin lisäomaisuuksiksi myös tekstin lihavointia sekä mahdollisuutta liittää tekstiin linkki, mutta ajanpuutteen vuoksi ominaisuuksia ei ehditty toteuttaa.

Sovellusta käytetään tällä hetkellä yleisellä käyttäjätunnuksella. Käyttäjäntunnistusta varten sovellukseen voitaisiin rakentaa LDAP-verkkoprotokollaa (Lightweight Directory Access Protocol) käyttävä tunnistus, jolloin henkilökuntaan kuuluvat tai tietyn roolin järjestelmässä omaavat voisivat kirjautua sovellukseen samoilla tunnuksilla kuin muihinkin oppilaitoksen tietojärjestelmiin. Tämä mahdollisuutta tutkittiin sovelluksen kehitysvaiheessa, mutta sopivan dokumentoinnin ja ajan puutteiden vuoksi kirjautuminen päätettiin toteuttaa yksinkertaisemmin. Henkilökohtaisten tunnusten käytön myötä sovelluksen toimintoja voitaisiin rajata käyttäjäkohtaisesti. Esimerkiksi alimman tason käyttäjä voisi hallita vain omia ilmoituksiaan ja vastaavasti ylemmän

tason roolin omaavat pääsisivät käsiksi kaikkiin toimintoihin ja voisivat hallita kaikkia ilmoituksia. Tämän mittakaavan sovelluksessa käyttäjäkohtaiselle erottelulle ei ehkä kuitenkaan olisi käytännön tarvetta.

Käyttäjien kirjautumista helpottava ominaisuus olisi myös vanhan sovellukseen malliin tehty IP-osoiterajaus. Tämä toimi niin, että sovelluksen hallintapuolelle sallittiin pääsy vain tietyn IP-osoiteavaruuden osoitteista. Näin kirjautuminen onnistui automaattisesti henkilökunnan verkkoon kytketyltä koneelta, mutta hankaloitti tietenkin käyttöä esimerkiksi kotoa. Ominaisuuden lisäämistä uuteen sovellukseen voitaisiin selvittää nykyisen kirjautumiskäytännön rinnalle.

10 POHDINTA

Aloitin vanhan infomonitorin työstämisen ensiksi projektiopintoina. Koska minulla oli kokemusta ja taitoa niin vähän, päätin alkaa tekemään muutoksia vanhan sovelluksen päälle sen hyvin sekavasta rakenteesta huolimatta. Huomasin tämän pian erittäin tuskalliseksi ja hitaaksi. Mielessäni vahvistui koko ajan ajatus siitä, että olisi kaiken kannalta parempi, jos sovellus tehtäisiin kokonaan uusiksi. Lopulta kohtalo puuttui peliin ja edesauttoi päätöksen tekoa rikkomalla kehityspalvelimen ja kaikki tehty työ hävisi bittiavaruuteen. Opin samalla jälleen varmuuskopioiden tärkeydestä. Opinnäytetyön läheneminen toi tullessaan ajatuksen, että homman voisi tehdä nyt kunnolla, omaa osaamista samalla kartuttaen. Jotain oli myös opittu kesätöiden aikana.

Olin erittäin tyytyväinen tekemääni valintaan toteutustekniikan suhteen. Sovelluskehyksen käyttö web-ohjelmoinnissa vastaa käsitystäni nykyisin tehtävän web-kehityksen suunnasta ja tämä oli erinomainen tilaisuus perehdyttää itseni kunnolla sellaisen käyttämiseen. Näin ollen en ainoastaan verestänyt taitojani, vaan opin myös paljon uutta.

Työ oli haastava ja olin aika ajoin vaikeuksissa. Ongelmien ratkaisua vaikeutti osaltaan se, että olin tavallaan kaivanut itseni kuoppaan valitsemalla sovelluksen toteutustekniikan niin, että suoranaista asiantuntija-apua ei ollut saatavilla. Apua oli toki saatavilla ohjelmointiin ja kehitysympäristöön liittyvissä ongelmissa, mutta itse sovelluskehykseen liittyvät ongelmatapaukset ratkaisin lähinnä Googlen avustuksella. Näin jälkikäteen ajatellen, olisi ollut fiksu veto tutustua aivan aluksi sovelluskehyksen omi-

naisuuksiin sekä sen vaatimuksiin kehitysympäristön suhteen, eikä syöksyä pää edeltä kehitystyöhön.

Internetin sisältö käy läpi rakennemuutosta staattisista HTML-sivuista monimutkaisiin, interaktiivisiin ja dynaamisiin web-palveluihin. Web-ohjelmointi ei enää ole pelkästään tekstieditorilla tehtävää kuvauskielten hallintaa, vaan oikeaa koodaamista. Kehityksessä mukana rimpuilu vaatii ohjelmoijalta muskeleita, joita sovelluskehys tarjoaa. Kun pyörät kerran on jo keksitty, miksei siis nauttisi niiden tarjoamasta sulavammasta kyydistä?

LÄHTEET

Brinzarea-Iamandi, Bogdan & Darie, Cristian & Hendrix, Audra 2009. AJAX and PHP: Building Modern Web Applications (2nd Edition). Olton Birmingham: Packt Publishing Ltd.

Cake Software Foundation a. Cookbook 2.x. Associations: Linking Models Together. Saatavissa: http://book.cakephp.org/2.0/en/models/associations-linking-models-together.html [viitattu: 16.4.2012].

Cake Software Foundation b. Cookbook 2.x. CakePHP Conventions. Saatavissa: http://book.cakephp.org/2.0/en/getting-started/cakephp-conventions.html [viitattu: 12.4.2012].

Cake Sofware Foundation c. Cookbook 2.x. Data Sanitization. Saatavissa: http://book.cakephp.org/2.0/en/core-utility-libraries/sanitize.html [viitattu: 23.4.2012].

Cake Software Foundation d. Cookbook 2.x. Installation. Saatavissa: http://book.cakephp.org/2.0/en/installation.html [viitattu: 12.4.2012].

Heinisuo, Rami 2004. PHP ja MySQL: Tietokantapohjaiset verkkopalvelut. Jyväskylä: Talentum Media Oy.

Koskimies, Kai & Mikkonen, Tommi 2005. Ohjelmistoarkkitehtuurit. Jyväskylä: Talentum Media Oy.

Kymenlaakson ammattikorkeakoulu 2010. Intranet-sivujen käyttöohjeita. Saatavissa: http://www.kyamk.fi/Intran%20etusivu/Intranetin%20k%C3%A4ytt%C3%B6 [viitattu: 14.4.2012].

NetBeans. NetBeans IDE – The Smarter Way to Code. Saatavissa: http://netbeans.org/features/index.html [viitattu: 14.4.2012].

Notaras, George 2008. Making directory writable by the webserver. G-Loaded Journal. Saatavissa: http://www.g-loaded.eu/2008/12/09/making-a-directory-writable-by-the-webserver/ [viitattu: 14.5.2012].

Open Source Initiative. Open Source Initiative OSI – The MIT License (MIT):Licensing The MIT License (MIT). Saatavissa: http://www.opensource.org/licenses/MIT [viitattu 11.4.2012].

Oracle Corporation a. MySQL Workbench 5.2: Design, Develop, Administer. Saatavissa: http://www.mysql.com/products/workbench/ [viitattu: 14.4.2012].

Oracle Corporation b. Event Scheduler Overview. Saatavissa: http://dev.mysql.com/doc/refman/5.1/en/events-overview.html [viitattu: 16.4.2012].

Oracle Corporation c. The Event Scheduler and MySQL Privileges. Saatavissa: http://dev.mysql.com/doc/refman/5.1/en/events-privileges.html [viitattu: 16.4.2012].

Syam, Anupom & Bari, Ahsanul 2008. CakePHP Application Development. Olton Birmingham: Packt Publishing Ltd.

Verens, Kae 2009. JQuery 1.3 with PHP : Enhance Your PHP Applications by Increasing their Responsiveness through jQuery and its Plugins. Olton Birmingham: Packt Publishing.