LAHDEN AMMATTIKORKEAKOULU
Lahti University of Applied Sciences

VERKKOSOVELLUS CAKE-PHP:LLA

Case: Tuntikirjanpito

LAHDEN
AMMATTIKORKEAKOULU
Tekniikan ala
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka
Opinnäytetyö AMK
Kevät 2012
Katja Kettula

Lahden ammattikorkeakoulu
Tietotekniikan koulutusohjelma

KETTULA, KATJA:

Verkkosovellus CakePHP:llä
Case: tuntikirjanpito

Tietotekniikan opinnäytetyö, 52 sivua

Kevät 2012

TIIVISTELMÄ

Tämä opinnäytetyö käsittelee verkkosovelluksen toteuttamista PHP-ohjelmointikehys CakePHP:n avulla. Työ on tehty Tilipalvelu Capellan toimeksiannosta. Sovellus on tarkoitettu tuntikirjanpitoa varten.

CakePHP, jota tässä työssä käsitellään, on kokoelma koodia, kirjastoja ja käytäntöjä. Se on erityisesti suunniteltu webkehittämistä varten. Websivujen tekeminen ohjelmointikehyksellä on nopeaa, koska tärkeimmät toiminnot on jo valmiiksi ohjelmoitu. Työn tavoitteena on havaita, miten CakePHP-ohjelmointikehyksen avulla voi tehdä verkkosivuston ja onko se helpompaa kuin koodata alusta pitäen itse.

CakePHP käyttää MVC (Model View Controller) -ohjelmistoarkkitehtuuria. Käytännössä MVC eriyttää websivun tyypilliset osa-alueet omiksi kokonaisuuksiksi. Tällöin luokat ovat käytössä koko sovelluksessa.

Asiasanat: WWW, MVC, model-view-controller, dynaaminen verkkosovellus, PHP, framework, CakePHP

Lahti University of Applied Sciences
Degree Programme in information technology

KETTULA, KATJA: Web application with CakePHP
Case: keeping working hours

Bachelor's Thesis in software engineering, 52 pages

Spring 2012

ABSTRACT

This thesis is about creating a web application with PHP framework CakePHP. The Co-operating company for the thesis is Tilipalvelu Capella Oy. The application is meant for keeping working hours.

CakePHP is a collection of code, libraries and conventions. It is especially designed for developing web applications. Creating web-pages with a framework is quick because important functions have already been programmed. The goal of this work is to observe how to create a web application with CakePHP and weather it is easier than do all the coding from scratch.

CakePHP uses the MVC (Model View Controller) design pattern. Technically, MVC separates typical sections into their own entities. This way classes are in use throughout the whole application.

Key words: WWW, MVC, model-view-controller, dynamic web application, PHP, framework, CakePHP, database, MySQL

SISÄLLYS

1 JOHDANTO

Web-sivustojen yleistymisen myötä on kasvanut kysyntä helposta ja tehokkaasta tavasta luoda websivuja. Web-sivustoja tehdään monilla erilaisilla tekniikoilla. Ohjelmointikehykset tarjoavat yhdenmukaisen tavan kehittää sivustoja. Sivuston ylläpidettävyyttä helpottaa, kun se on tehty jollain tietyllä rakenteella, jolloin voi olla useampia ohjelmoijia ylläpitäjinä.

Tämän opinnäytetyön toimeksiantaja on Tilipalvelu Capella Oy. Tämä yritys tekee kirjanpidon lisäksi palkanlaskentaa. Työntekijät ilmoittavat työtuntinsa sähköpostitse tai paperilapuilla, joista palkanlaskija syöttää ne laskentataulukkoon. Tämä on työlästä ja aikaa vievää. Työ helpottuisi, jos olisi käytössä internetissä toimiva ohjelma, jossa jokainen voisi syöttää itse tuntinsa. Tässä opinnäytetyössä on tavoitteena toteuttaa sovellus, jonka avulla työntekijät voivat kirjata työtuntinsa ja esimies voi ne hyväksyä.

Tutkimusongelmana tässä opinnäytetyössä on websovelluksen kehittäminen PHP-ohjelmointikehyksellä CakePHP. Työn teoriaosuudessa on tutkittu CakePHP:n toimintaa ja koodia. Työ on rajattu ohjelmointikehyksen käyttöön.

Tässä työssä käsitellään myös MVC (Model View Controller) -suunnittelumallia. Käytännössä MVC-malli eriyttää websivun tyypilliset osa-alueet omiksi kokonaisuuksiksi. Tällöin luokat ovat käytössä koko sovelluksessa, mikä selkeyttää rakennetta. CakePHP pohjautuu MVC-malliin.

Tässä työssä CakePHP:tä on käytetty WAMP-ympäristössä. WAMP tulee sanoista Windows-Apache-MySQL-PHP eli käyttöjärjestelmänä on Windows, webpalvelimena on Apache, tietokantana on MySQL ja ohjelmointikielenä on PHP.

2 TOIMINTAYMPÄRISTÖ

CakePHP tarvitsee toimiakseen webpalvelimen, tietokannan ja PHP-moottorin. Yleisimpiä tekniikoita, joita CakePHP käyttää toiminnassaan, esitellään seuraavaksi.

2.1 (X)HTML

HTML on lyhenne sanoista Hypertext markup language eli hypertekstin merkintäkieli. HTML:llä voidaan merkitä tekstin rakenne eli esimerkiksi, mikä osa tekstistä on otsikkoa ja mikä leipätekstiä. Websivustot on tehty HTML-kielellä (Wikipedia 2012e). Kuviossa 1 on esimerkki HTML-kielestä. HTML-kieli koostuu "tageista" eli < ja > -merkkien sisään kirjoitetuista koodeista. Esim. rivillä 3 <html>-tagi aloitaa websivun ja rivillä 15 </html>-tagi lopettaa sen. Jokainen komento tulee aloittaa ja lopettaa samalla koodilla, mutta lopettavan koodin < ja >-merkkien sisään ennen varsinaista koodia kirjoitetaan /-merkki. Joissakin koodeissa ei ole lopettavaa tagia. Silloin aloitustagi lopetetaan tyyliin />, kuten rivillä 12 HTML-kieltä voi kirjoittaa pienillä tai isoilla kirjaimilla, mutta XHTML-kieltä kirjoitetaan aina pienillä kirjaimilla. XHTML on lyhenne sanoista extensible hypertext language. XHTML on tiukempi muotosäännöissään ja soveltuu useammille päätelaitteille. Esimerkiksi mobiililaitteet osaavat lukea xhtml-kielellä tehtyjä websivuja.

```
1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
2   "http://www.w3.org/TR/html4/strict.dtd">
3  <html>
4      <head>
5          <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
6          <title>Otsikko</title>
7  <link rel="stylesheet" href="sivustylet.css" type="text/css" />
8      </head>
9  <body>
10         <div id="runko">
11         Sisältöä
12         <img src="images/kuva.jpg" />
13         </div>
14 </body>
15 </html>
```

KUVIO 1. Esimerkki HTML-kielestä

2.2 PHP

PHP, joka tulee sanoista preprosessed hypertext, on laajasti käytetty ohjelmointikieli ja sopii erityisen hyvin dynaamisten verkkosovellusten kehittämiseen. Ohjelmointikielen lisäksi PHP-ympäristössä on laaja luokkakirjasto. PHP on komentosarjakieli, jossa ohjelmakoodi tulkitaan vasta ohjelman suoritusvaiheessa. PHP:tä voidaan käyttää useilla eri alustoilla ja käyttöjärjestelmillä. PHP:n ensimmäinen versio julkaistiin vuonna 1995, ja nykyisin PHP on vertailuissa johtava dynaamisten web-palveluiden tuottamiseen tarkoitettu kieli. PHP 5 versiosta lähtien PHP on tukenut olio-ohjelmointia, joka on mahdollistanut luokkien periyttämisen. (PHP documentation 2012.) Kuviossa 2 on esitetty PHP-kielen syntaksia. PHP-kieltä voi kirjoittaa HTML-kielen sekaan. PHP-koodi kirjoitetaan <?php ja ?> tagien sisään.

```
9   <body>
10
11  <?php
12      echo '<p>Tekstiä</p>';
13  ?>
14
15  </body>
```

KUVIO 2. Esimerkki PHP-kielestä

2.3 CSS

CSS tulee sanoista cascading style sheet. Sillä määritetään tyylit websivustoille, esim. tekstien fontit ja värit, elementtien sijainnit ja muotoilut. CSS-tiedosto sisällytetään websivuun kuvion 3. mukaisesti.

```
7   <link rel="stylesheet" href="sivustylet.css" type="text/css" />
```

KUVIO 3. CSS-tiedoston sisällytys websivun head-osioon

Kuviossa 4 on esimerkki css-tiedostosta. Tässä esimerkissä on määritetty body-osiolle taustaväri, fontin väri, fontti, fontin koko ja sivun marginaalin leveys riveillä 26 - 30. a rivillä 32 tarkoittaa linkkiä, jolle on määritelty väri rivillä 33, tekstin korostukseksi alleviivaus rivillä 34 ja fontti on määritetty lihavoiduksi

rivillä 35 h-määrittelyt rivillä 37 ovat otsikoita, ja niille on määritetty fontti normaaliksi rivillä 38 ja alamarginaali on määritetty rivillä 39

```
25  body {
26          background: #003d4c;
27          color: #fff;
28          font-family:'lucida grande',verdana,helvetica,arial
29          font-size:90%;
30          margin: 0;
31  }
32  a {
33          color: #003d4c;
34          text-decoration: underline;
35          font-weight: bold;
36  }
37  h1, h2, h3, h4 {
38          font-weight: normal;
39          margin-bottom:0.5em;
40  }
```

KUVIO 4. Esimerkki css-tiedostosta

2.4 HTTP, POST , GET ja session

Tiedonsiirtoprotokollan HTTP (Hypertext Transfer Protocol) avulla tietoa voi siirtää websivujen välillä. Yleisimmin tietoa pitää siirtää lomakkeiden välillä. Websivustoilla käytetään tiedonsiirtoon usein HTTP:n post- tai get-metodia. Get-metodilla tieto siirtyy osoiterivillä ja tietoa pystyy siirtämään paljon vähemmän kuin post-metodilla. (Korpela 2012.) Kuviossa 5 on esimerkki lomakkeen aloitustagista, jossa on käytetty metodina postia ja tekstikenttä on rivillä 2 Kuviossa 6 on esitetty, miten tieto luetaan, joka on lomakkeelle syötetty.

```
1  <form action='tallenna.php' name='lomake' method='post'>
2    <input type='text' size='50' name='arvo' />
3    </form>
```

KUVIO 5. Metodi post lomakkeen aloituksessa ja tekstikenttä

```
4  <?php
5  echo $_POST["arvo"];
6  ?>
```

KUVIO 6. Arvon lukeminen postista

Sessionin eli istunnon avulla luodaan pysyvä yhteys protokollaan. Istunto syntyy, kun istuntoa tukeva protokolla yhdistää kaksi eri päätettä toisiinsa. Yhdistämisen aikana protokolla luo istuntoavaimen, joka on uniikki, vain yhdistettävien päätteiden käytettävissä oleva tunniste. Esimerkiksi, käyttäjän kirjautumistiedot pysyvät tallessa ja sivusto pystyy tunnistamaan käyttäjän liikkuessa sivulta toiselle. Kuviossa 7 on esitetty, miten sessio aloitetaan (rivi 2), miten sessioon kirjoitetaan (rivi 4) ja miten sessiosta luetaan (rivi 6).

```
1  <?php
2  session_start();
3
4  $_SESSION['teksti'] = "jotakin";
5
6  echo $_SESSION['teksti'];
7  ?>
```

KUVIO 7. Session esimerkki

2.5 Javascript

Javascript on pääasiassa web-ympäristössä käytettävä tulkattava oliopohjainen komentosarjakieli. Javascriptin periytyvyys pohjautuu prototyyppeihin eikä luokkiin kuten yleensä muissa ohjelmointikielissä. Javascriptillä lisätään web-sivuille dynaamista toiminnallisuutta. (Wikipedia 2012f.) Javascriptiä kirjoitetaan <script> ja </script> -tagien väliin, kuten kuviossa 8 on esitetty. Rivillä 2 on kirjoitettu websivulle senhetkinen päiväys.

```
1  <script type="text/javascript">
2   document.write("<p>" + Date() + "</p>");
3  </script>
```

KUVIO 9. Esimerkki javascriptistä

2.6 XML

XML, joka tulee sanoista Extensible Markup Language, on merkintäkieli tai standardi, jolla tiedon merkitys on kuvattavissa tiedon sekaan. XML-kieltä

käytetään sekä formaattina tiedonvälitykseen järjestelmien välillä että formaattina dokumenttien tallentamiseen. XML-kieli on rakenteellinen kuvauskieli, joka auttaa jäsentämään laajoja tietomassoja selkeämmin. XML:n kehittäjä on World Wide Web Consortium (Wikipedia 2012h). Kuviossa 10 on esimerkki xml-merkintäkielestä. Tässä on esitetty menun luominen. Riveillä 15 ja 21 aloitus- ja lopetustagit. Riveillä 17, 18 ja 19 ovat menun painikkeet.

```
15  <menu id="file" value="File">
16    <popup>
17      <menuitem value="New" onclick="CreateNewDoc()" />
18      <menuitem value="Open" onclick="OpenDoc()" />
19      <menuitem value="Close" onclick="CloseDoc()" />
20    </popup>
21  </menu>
```

KUVIO 10. Esimerkki xml-merkintäkielen käytöstä

2.7 JSON

JSON tulee sanoista javascript object notation, mutta ei silti ole javascriptistä riippuvainen. Sitä voidaan jäsennellä monilla muillakin kielillä. Se on kevyt tiedonsiirtomuoto, joka ei vie paljon kaistaa tiedonsiirrossa. Pääasiallisesti JSON:a käytetään tiedonsiirtoon palvelimen ja websovelluksen välillä. JSON koostuu nimi-arvopareista. Nimi ja arvo erotetaan toisistaan pilkulla. (json.org 2012.) Kuviossa 11 on esimerkki JSON-merkinnän käytöstä, ja tällä esimerkillä sovellus pystyy jäsentelemään edellisen kuvion 10. mukaisen XML-tiedoston.

```
1  {"menu": {
2    "id": "file",
3    "value": "File",
4    "popup": {
5      "menuitem": [
6        {"value": "New", "onclick": "CreateNewDoc()"},
7        {"value": "Open", "onclick": "OpenDoc()"},
8        {"value": "Close", "onclick": "CloseDoc()"}
9      ]
10   }
11 }}
```

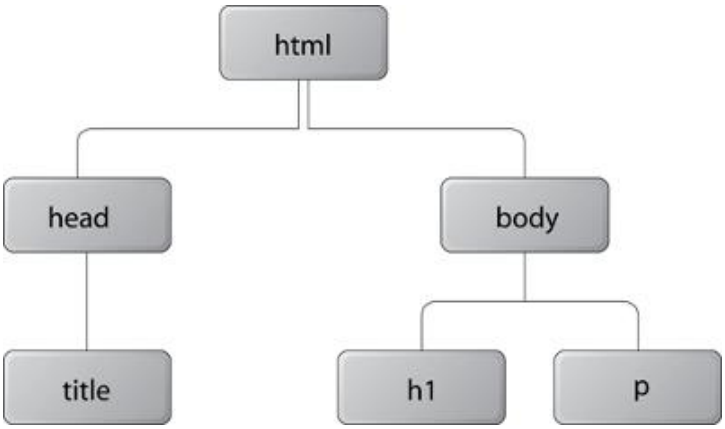
KUVIO 11. JSON-esimerkki

2.8 Jquery ja DOM

DOM tulee sanoista Document Object Model. Elementti HTML-, XHTML- tai XML-sivulla muodostaa DOM-solmun ja solmut yhdessä muodostavat DOM-puun. Kuviossa 12 näkyy XHTML-koodi ja kuviossa 13 koodista muodostettu DOM-puu.

```
1  <html>
2  <head>
3  <title>Otsikko</title>
4  </head>
5  <body>
6  <h1>tekstiä</h1>
7  <p id="elementti">jotakin tekstiä</p>
8  </body>
9  </html>
```

KUVIO 12. XHTML-koodi



KUVIO 13. DOM-puu

Jquery on kokoelma javascript-kirjastoja, jotka täytyy erikseen ladata ja ottaa käyttöön verkkosivustolla. Jqueryn avulla voi käsitellä DOM-elementtejä ja tehdä esimerkiksi animaatioita, siirrellä elementtejä sivustolla, piilottaa ja näyttää elementtejä ja elävöittää sivustoa monella tavalla. Jquery-kirjasto on AJAX-yhteensopiva, ja funktiot ja kirjastot lataavat tietoa palvelimelta ilman, että sivua täytyy ladata uudestaan (jquery documentation 2012). CSS liittyy myös läheisesti jqueryyn ja DOM:in. Yleinen käyttötapa on ladata CSS-tyyli elementille jqueryn avulla. Kuviossa 14 on esitetty, miten elementille määritetään CSS-tyyli.

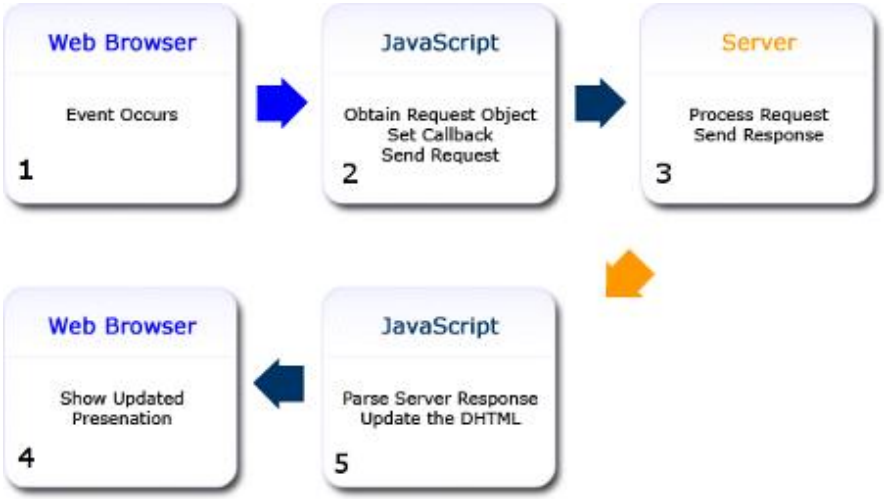
```
1   $('#elementti').addClass('tyyli');
```

KUVIO 14. Jquery esimerkki

2.9 Ajax

Ajax, joka tulee sanoista asynchronous javascript and XML, on joukko web-sovelluskehityksen tekniikoita, joiden avulla web-sovelluksista voi tehdä vuorovaikutteisempia. Ajaxissa selainohjelma vaihtaa pieniä määriä dataa palvelimen kanssa taustalla niin, ettei koko verkkosivua tarvitse ladata uudelleen käyttäjän tehdessä muutoksen. (Wikipedia 2012b.) Esimerkiksi tietokannasta voi hakea tietoa johonkin websivun osioon ilman, että koko websivua pitäisi päivittää. Kuviossa 15 on kaavio ajaxin toiminnasta.

1. Selain tekee pyynnön.
2. Javascript lähettää XMLHttpRequestObject pyynnön eteenpäin.
3. Palvelin käsittelee pyynnön. Käsittelyn voi hoitaa esim. PHP.
4. Javascript välittää tuloksen selaimelle.
5. Selain näyttää tuloksen.



KUVIO 15. Ajaxin toimintaperiaate

3 MVC

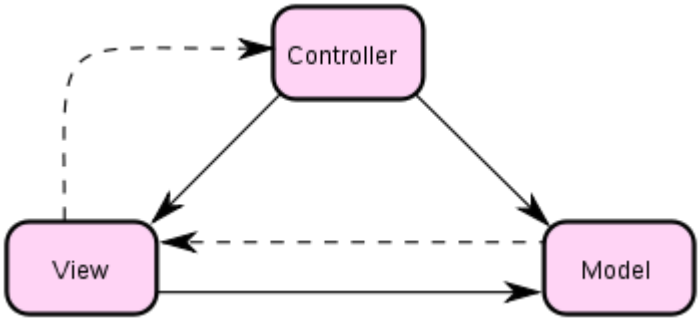
MVC tulee sanoista Model View Controller eli malli-näkymä-ohjain. Sen tarkoituksena on käyttöliittymän erottaminen sovelluslogiikasta (Wikipedia 2012g). Malli-näkymä-ohjain –ohjelmistoarkkitehtuurityylissä käyttäjän syötteet, ohjelmalogiikka ja visuaalinen palaute on jaettu kolmeen toisistaan selvästi erotettuihin komponentteihin, joista jokainen on erikoistunut omaan tehtäväänsä (Rapa 2008). MVC:n pääperiaate on se, että kaikki funktiot kirjoitetaan vain kerran. Kaikki funktiot ovat sovelluksen käytettävissä joka osiossa. Tämä vähentää päällekäisyyttä (Golding 2008, 3-4).

3.1 MVC:n historiaa

MVC-suunnittelumallia on kehitetty jo yli kolmekymmentä vuotta. Ensimmäisen kerran sen esitteli norjalainen Trygve Reenskaug. Hän työskenteli Xerox PARC:n Smalltalk-ohjelmointikielen kehitysryhmässä. He pyrkivät luomaan yleiskäyttöisiä komponentteja, joiden avulla pystyisi luomaan vaivattomasti vuorovaikutteisia graafisia järjestelmiä. Malli-näkymä-ohjain syntyi tämän luomistyön tuloksena (Rapa 2008).

3.2 MVC:n rakenne

Kuviossa 16 on esitetty yksinkertaisesti rakenne MVC-mallille



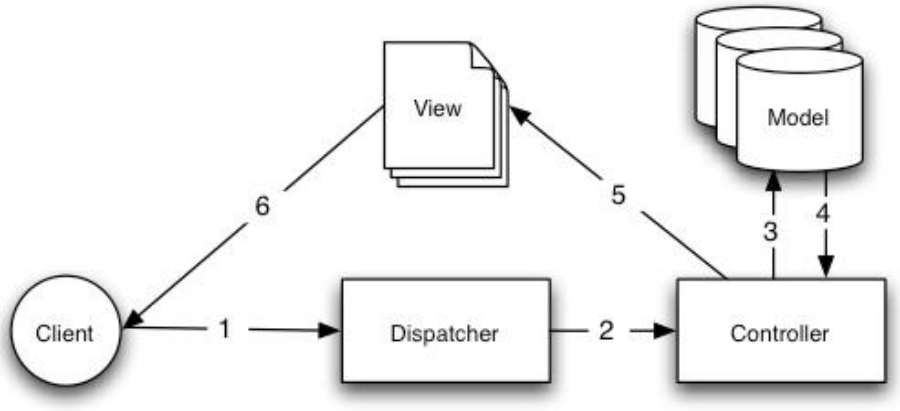
KUVIO 16. MVC-malli yksinkertaisesti

MVC koostuu seuraavista osista:

- Malli, joka huolehtii järjestelmän sovellusaluekohtaisen tiedon tallentamisesta, ylläpidosta ja käsittelystä.
- Näkymä, joka määrittää käyttöliittymän ulkoasun ja mallin tietojen esitystavan käyttöliittymässä.
- Ohjain eli kontrolleri, joka vastaanottaa käyttäjältä tulevat käskyt sekä muuttaa mallia ja näkymää vastauksena niihin. (Wikipedia 2012g.)

Kuviossa 17 on esitetty MVC:n rakenne tarkemmin

1. Client tekee pyynnön.
2. Dispatcher (lähettäjä) käsittelee pyynnön ja välittää oikealle ohjaimelle (Controller).
3. Ohjain (Controller) käsittelee ja reitittää käyttäjän pyynnöt mallille.
4. Malli käsittelee datan, esim. hakee tiedot tietokannasta ja välittää ne ohjaimelle.
5. Näkymä (View) tulkitsee mallilta tulevan datan.
6. Näkymä näyttää sisällön käyttäjälle. (CakePHP documentation 2012g).



KUVIO 17. MVC:n rakenne

3.3 MVC:n etuja

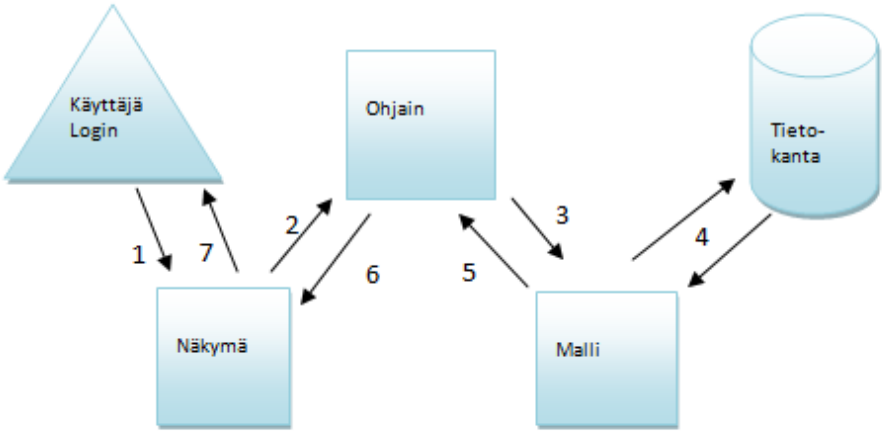
MVC:n etuja on, että malli, näkymä ja ohjain ovat itsenäisesti toimivia jolloin rakenne on selkeä. Käyttöliittymäkoodin ja muun sovelluskoodin voi erottaa toisistaan ja laajennettavuus on helppoa, koska samalle mallille voi luoda uusia näkymiä (Vanhatupa 2011). Tiimityö on helpompaa, kun jokaisella ohjelmoijalla voi olla oma selkeä alueensa (Golding 2008).

3.4 MVC JA CAKE-PHP

CakePHP käyttää malli-näkymä-ohjain-rakennetta. Esimerkkinä seuraavassa miten sisäänkirjautuminen toimi CakePHP:ssä malli-näkymä-ohjain -suunnittelumallia hyväksikäyttäen.

Kuviossa 18 on sisäänkirjautumisen kaavio.

1. Käyttäjä syöttää käyttäjätunnuksen ja salasanan lomakkeelle.
2. Näkymä, joka näyttää lomakkeen, lähettää lomakkeelle syötetyn datan ohjaimelle.
3. Ohjain lähettää pyynnön mallille hakea tietokannasta vastaavuutta käyttäjätunnukselle ja salasanalle.
4. Malli tuottaa SQL-kyselyn ja ajaa sen tietokannassa.
5. Tietokanta-ajon tuloksesta riippuen, malli palauttaa joko true tai false arvon ohjaimelle.
6. Ohjain käsittelee tuloksen ja hakee sopivan näkymän lähetettäväksi käyttäjälle.
7. Lopullinen tulos eli kirjautuminen onnistunut-viesti tai virhe-viesti näytetään käyttäjälle. (Golding 2008, 17.)



KUVIO 18. Kaavio sisäänkirjautumisesta

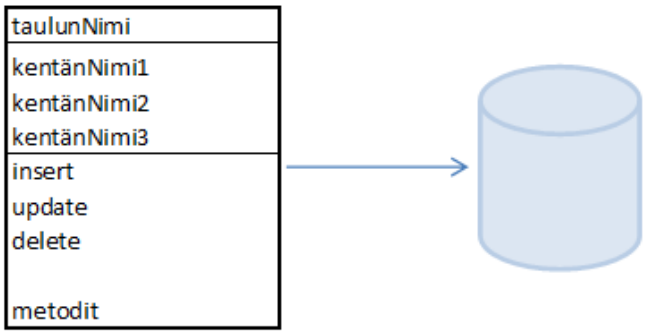
4 CAKE-PHP

4.1 Yleistä CakePHP:stä

CakePHP on ohjelmointikehys, joka on kehitetty PHP-ohjelmointikielellä. CakePHP on ilmainen, ja se on julkaistu MIT (Massachusetts Institute of Technologyssä) -lisenssillä. MIT-lisenssi on vapaa ohjelmistolisenssi, ja lähdekoodia voi vapaasti muokata ja kopioida. CakePHP käyttää tunnettuja malleja, kuten Active Recordia, Association Data Mappingia ja Front Controlleria (CakePHP readme-tiedosto 2012). CakePHP:n voi asentaa monenlaisille alustoille, mutta tässä työssä on käytetty Apache-webpalvelinta ja MySQL-tietokantaa.

4.1.1 Active Record

Active Record on malli, joka varastoi tietoa relaatiotietokantaan (Wikipedia 2012a). CakePHP käyttää Active Record -mallia tietokantakyselyjen käsittelyssä. Active Record on tapa poistaa koodin kertaantumista ja päästä käsiksi dataan tietokannassa. Active Record -mallissa tietokannan taulu tai näkymä on yksi luokka ja tietokannan rivi on yksi olion ilmentymä. Active Recordin rajapinnat sisältää esim. lisää, muokkaa ja poista funktiot (Fowler 2012.) Kuviossa 19 on esitetty Active Recordista kaavio.



KUVIO 19. Active Record

4.1.2 Front Controller

Front Controller -malli käsittelee keskitetysti pyyntöjä ja CakePHP:n tapauksessa pyynnöt lähetetään ohjaimelle. FrontController on sama asia kuin aiemmin MVC-luvussa mainittu dispatcher. CakePHP:ssä dispatcher vaatii bootstrap.php-tiedostoa, joka esitellään myöhemmin ja kutsuu dispatcher-luokkaa (CakePHP documentation 2012c). Kuviossa 20 näkyy kutsu rivillä 95

```
93          App::uses('Dispatcher', 'Routing');
94
95          $Dispatcher = new Dispatcher();
96          $Dispatcher->dispatch(new CakeRequest(), new CakeRespon
97
```

KUVIO 20. Dispatcher-kutsu

4.1.3 Apache-webpalvelin ja osoiterivi

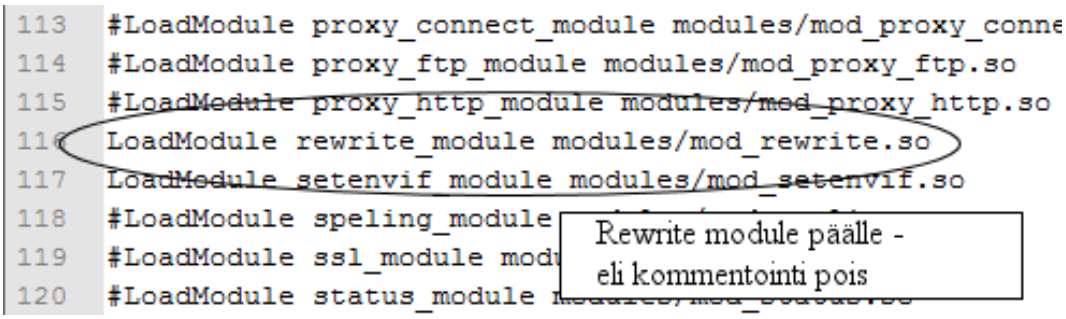
Apache on ilmainen ja yleinen webpalvelin (Apache 2012.) Apache-palvelimen mukana tulee mod_rewrite-moduuli. Se mahdollistaa internetosoitteiden manipuloinnin. Osoite on helpommin käyttäjän luettavissa. Myös hakukoneet indeksoivat sivun paremmin, kun osoite on selkokielinen. CakePHP näyttää sisältöä osoiterivin perusteella. CakePHP:n oletusreitti on seuraavaa muotoa:

http://domain/{application}/{controller}/{action}/{parameter 1}/→

Eli jos on luotu esim. Users-niminen Controller, tiedot saa näkyviin:

http://domain/application/users

Käytettäessä CakePHP:tä Apache-palvelimelta täytyy sallia mod_rewrite-moduulin käyttö (Golding 2008, 12.) Kuviossa 21 on osa Apache-palvelimen konfigurointi-tiedostosta, jossa havainnollistetaan, miten sallitaan osoitteiden uudellenkirjoitus.

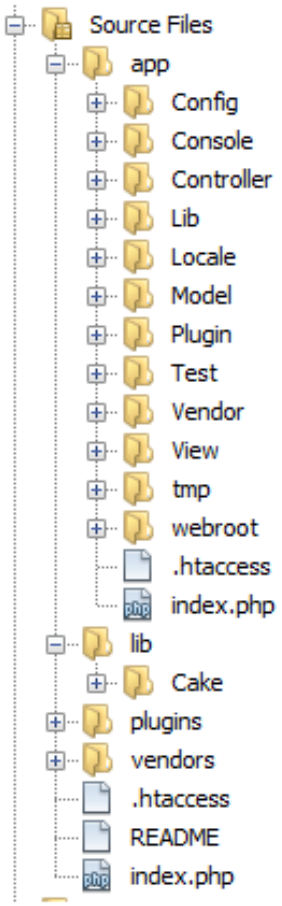


KUVIO 21. Apache konfiguraatio-tiedosto rewrite module

4.2 CakePHP hakemistot ja tiedostot yleisesti

Tiedostojen nimeäminen on tärkeää tehdä oikein CakePHP:ssä ja nimeämisessä on oma periaatteensa. Cake osaa itsenäisesti yhdistää tietokannan oikeaan näkymään, kunhan se on nimetty oikein. Tiedostot nimetään pienillä kirjaimilla ilman välilyöntejä tai erikoismerkkejä ja kirjainten pitää olla väliltä a-z. Jokaiselle mallille on yksi tietokantataulu. Malli nimetään yksikössä, tietokantataulu monikossa. (Golding 2008, 19 20). Tietokannan tauluja ei kannata nimetä samoiksi kuin Caken elementtejä, esim. taulua ei kannata nimetä views- tai controllers-nimisiksi. Myöskään PHP:n funktioiden nimiä, kuten date, ei kannata käyttää taulujen nimissä (Golding 2008, 37).

Kuviossa 22 on esitetty hakemistorakenne. Juurihakemistossa sijaitsee neljä hakemistoa: app, lib, plugins ja vendors. App-kansio on kehittäjän kannalta tärkein, ja siellä sijaitsee ne kansiot, joihin omia tiedostoja talletetaan. App-kansiossa on mm. config-, controller-, model-, view- ja webroot-kansiot. Config-kansiossa sijaitsee konfigurointi-tiedostot, josta kerrotaan enemmän konfigurointi-luvussa. Controller-kansioon tulee ohjaintiedostot, josta kerrotaan enemmän ohjain-luvussa. Model-kansioon tulee mallitiedostot, josta kerrotaan enemmän malli-luvussa. View-kansioon tulee näkymätiedostot, josta kerrotaan enemmän näkymä-luvussa. Webroot-kansioon tulee sivuston ulkoasuun liittyvät tiedostot esim. kuvat, javascript-tiedostot, css-tiedostot ja template-tiedostot. Lib-kansiossa on Cake-kansio, jossa sijaitsee CakePHP:n ydin.



KUVIO 22. CakePHP:n hakemistorakenne

4.3 Konfigurointi

Config-kansiossa on määrittelytiedostot tietokannalle, oikeuksille, sähköpostitukselle, ytimelle ja reitityksille. Seuraavaksi esitellään konfigurointi-tiedostojen sisältöjä.

4.3.1 core.php-tiedosto

Core.php-tiedostossa määritetään, näytetäänkö sivulla virheviestejä, kirjoitetaanko sessioneita ja miten pitkään pidetään tietoja välimuistissa. Kun sovellus on kehitysvaiheessa, virhetiedot näytetään, mutta ennen sovelluksen julkaisemista, virhetietojen näyttö piilotetaan.

4.3.2 bootstrap.php-tiedosto

Bootstrap.php-tiedosto ladataan heti core.php-tiedoston jälkeen ja on oletuksena melko tyhjä. Tähän tiedostoon ohjelmoija voi määrittää omia funktioita, ja jos sovelluksessa käytetään liitännäisiä, niiden lataaminen tapahtuu bootstrap-tiedostossa.

4.3.3 routes.php-tiedosto

Tässä tiedostossa määritetään reitit eli polut ohjaimiin. Kuviossa 23 kuvataan, miten reititys määritetään. Ensimmäisessä rivissä on määritetty reitti ohjaimeen nimltä Pages ja välitetään parametri näkymään home. Toisessa rivissä on reitti samaan ohjaimeen, mutta näkymää ei ole määritetty.

```
24      * Here, we are connecting '/' (base path) to controller called 'Pages',
25      * its action called 'display', and we pass a param to select the view file
26      * to use (in this case, /app/View/Pages/home.ctp)...
27      */
28             Router::connect('/', array('controller' => 'pages', 'action' => 'display', 'home'));
29     /**
30      * ...and connect the rest of 'Pages' controller's urls.
31      */
32             Router::connect('/pages/*', array('controller' => 'pages', 'action' => 'display'));
33
```

KUVIO 23. Reititys

4.3.4 acl.ini.php-tiedosto

acl.ini-tiedostoa käytetään oikeuksien määrittämiseen. Ohjelman eri osioille voi määrittää oikeuksia yksittäiselle käyttäjälle sekä ryhmälle. Kuviossa 24 on esitetty miten oikeudet määritetään. Deny-määreellä kielletään oikeus kyseiseen olioon ja allow-määreellä sallitaan. Aco tulee sanoista access control object.

```
;-------------------------------------
;Users
;-------------------------------------

[username-goes-here]
groups = group1, group2
deny = aco1, aco2
allow = aco3, aco4

;-------------------------------------
;Groups
;-------------------------------------

[groupname-goes-here]
deny = aco5, aco6
allow = aco7, aco8

```

KUVIO 24. Oikeuksien määrittäminen

Kuviossa 25 esitetään funktio, jolla tarkistetaan käyttäjän oikeudet. Rivillä 9 katsotaan käyttäjän id:n perusteella, kuka käyttäjä on ja jos ei ole riittäviä oikeuksia, annetaan tieto Access denied, rivi 14. Jos on pääsy, annetaan tieto access allowed, rivi 20. Tämä funktio sijoitetaan ohjain-tiedostoon.

```
<?php
function checkAccess($aco)
    {
        // Check access using the component:
        $access = $this->Acl->check($_SESSION['user_id'], $aco, $action = "*");

        //access denied
        if ($access === false)
        {
            echo "access denied";
            exit;
        }
        //access allowed
        else
        {
            echo "access allowed";
            exit;
        }
    }
?>
```

KUVIO 25. checkAccess-funktio

4.3.5 database.php-tiedosto

Database-tiedostolla määritetään tietokanta. Tietokantoja voi olla useampia. Kuviossa 26 havainnollistetaan, miten määritys tapahtuu. Määrityksiin kuuluu

tietolähde rivillä 63, joka on tässä tapauksessa MySql. Rivillä 64 määritetään yhteyden pysyvyys eli se, sulkeutuuko yhteys, kun koodin suorittaminen loppuu. Tässä on määritetty false eli se tarkoittaa sitä, että yhteys sulkeutuu. Host eli isäntä, joka on rivillä 65, on tässä tapauksessa paikallinen isäntä. Tietokannan käyttäjätunnus ja salasana löytyvät riveiltä 66 ja 67, tietokannan nimi on rivillä 68 ja tietokannan taulujen etuliite on rivillä 69.

```
59  */
60  class DATABASE_CONFIG {
61
62          public $default = array(
63                  'datasource' => 'Database/Mysql',
64                  'persistent' => false,
65                  'host' => 'localhost',
66                  'login' => 'user',
67                  'password' => 'password',
68                  'database' => 'esimerkkisovellus',
69                  'prefix' => '',
70                  //'encoding' => 'utf8',
71          );
```

KUVIO 26. Tietokanta-määrittely

4.3.6 email.php-tiedosto

Email.php-tiedostossa määritetään sähköpostitukseen liittyvät asetukset. Kuviossa 27 on osa email.php-tiedostoa. Ensimmäisessä taulukossa on smtp-asetukset ja toisessa sähköpostin asetuksia. Esim. lähettäjä on rivillä 67, vastaanottaja on rivillä 69.

```
52          public $smtp = array(
53                  'transport' => 'Smtp',
54                  'from' => array('site@localhost' => 'My Site'),
55                  'host' => 'localhost',
56                  'port' => 25,
57                  'timeout' => 30,
58                  'username' => 'user',
59                  'password' => 'secret',
60                  'client' => null,
61                  'log' => false
62                  //'charset' => 'utf-8',
63                  //'headerCharset' => 'utf-8',
64          );
65
66          public $fast = array(
67                  'from' => 'you@localhost',
68                  'sender' => null,
69                  'to' => null,
70                  'cc' => null,
71                  'bcc' => null,
72                  'replyTo' => null,
73                  'readReceipt' => null,
74                  'returnPath' => null,
```

KUVIO 27. Osa email konfigurointi-tiedosto

4.4 App-luokka

App-luokka lataa CakePHP:n luokat, määrittää luokkien polut ja sijainnit. Kuviossa 28 on esitetty, miten luokkia kutsutaan. Luokkakutsun ensimmäinen osa on kansion nimi ja toinen osa on luokan nimi. Rivillä 456 ensimmäinen Model on kansio nimeltä Model ja toinen Model on luokka nimeltä Model. Rivillä 457 tarkistetaan, että tiedosto löytyy, ja rivillä 458 tarkistetaan, että luokka on olemassa. Vastaavasti rivillä 472 Configure on kansion nimi ja PhpReader on luokan nimi. Configure-kansiossa on PhpReader.php-niminen tiedosto, jossa on PhpReader-niminen luokka. Rivillä 474 tarkistetaan, että kyseinen luokka on olemassa.

```
455         public function testClassLoading() {
456                 $file = App::import('Model', 'Model', false);
457                 $this->assertTrue($file);
458                 $this->assertTrue(class_exists('Model'));
459
460                 $file = App::import('Controller', 'Controller', false);
461                 $this->assertTrue($file);
462                 $this->assertTrue(class_exists('Controller'));
463
464                 $file = App::import('Component', 'Auth', false);
465                 $this->assertTrue($file);
466                 $this->assertTrue(class_exists('AuthComponent'));
467
468                 $file = App::import('Shell', 'Shell', false);
469                 $this->assertTrue($file);
470                 $this->assertTrue(class_exists('Shell'));
471
472                 $file = App::import('Configure', 'PhpReader');
473                 $this->assertTrue($file);
474                 $this->assertTrue(class_exists('PhpReader'));
```

KUVIO 28. Esimerkki siitä, miten App::importia käytetään

4.5 Malli

Mallit ovat luokkia, jotka huolehtivat sovelluksen toimintakerroksesta. Tämä tarkoittaa sitä, että malli vastaa melkein kaikesta, mikä liittyy tiedon tallentamiseen, validointiin, ylläpitoon ja käsittelyyn. Yleensä malli vastaa tietokannan taulua, mutta ne eivät ole rajoittuneet pelkästään siihen. Malleilla voidaan esittää tietoa myös tiedostoista, ulkoisista webserviceistä tai csv-tiedostojen riveistä. Mallit voivat olla yhteydessä toisiin malleihin (CakePHP Documentation 2012g). Kuviossa 29 esitetään, miten malli luodaan. Malli-tiedostot luodaan kansioon App/models. Tiedostonimet kirjoitetaan yksikkömuodossa. Tässä esimerkissä on käytetty users-taulua, joten kansioon models pitää olla luotuna tiedosto nimeltä user.php

```
1  <?php
2  class User extends AppModel{
3      var $name = 'User';
4  }
5  ?>
```

KUVIO 29. Mallin luominen

Esimerkissä luokka User perii AppModel-luokan. AppModel perii Model-luokan ja Model-luokassa tapahtuu toimintalogiikka. AppModel-luokka on oletuksena tyhjä. Sinne voi lisätä omia toiminnallisuuksia, ja kaikki mallit perivät nämä toiminnot. Model-luokassa on useita kymmeniä funktioita, jotka osallistuvat tiedon käsittelyyn. Kuviossa 30 esitetään Model-luokan Read-funktio. $fields muuttujalla välitetään yhden kentän nimi merkkijonona tai usean kentän nimet array-muodossa (taulukkona). Jos muuttuja jätetään tyhjäksi, kaikki kentät haetaan. $id muuttuja määrittää erillisen tietueen id-kentän, joka luetaan. Read-funktio palauttaa aina taulukon (CakePHP documentation 2012g).

```
1419        public function read($fields = null, $id = null) {
1420                $this->validationErrors = array();
1421
1422                if ($id != null) {
1423                        $this->id = $id;
1424                }
1425
1426                $id = $this->id;
1427
1428                if (is_array($this->id)) {
1429                        $id = $this->id[0];
1430                }
1431
1432                if ($id !== null && $id !== false) {
1433                        $this->data = $this->find('first', array(
1434                                'conditions' => array($this->alias . '.' . $this->primaryKey => $id),
1435                                'fields' => $fields
1436                        ));
1437                        return $this->data;
1438                } else {
1439                        return false;
1440                }
```

KUVIO 30. Malli-luokan read-funktio

4.6 Ohjain

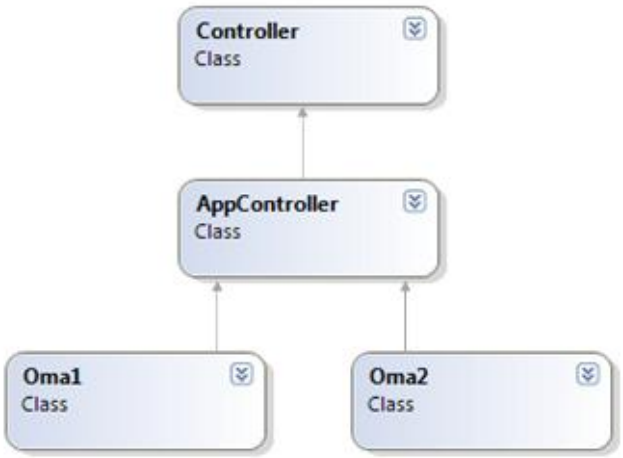
Ohjaimet tarjoavat monenlaisia metodeja, joita kutsutaan toiminnoiksi. Oletuksena kaikki toiminnot on määritetty julkisiksi ja niihin pääsee käsiksi suoralla osoitteella eli url:lla. Ohjain reitittää käskyt oikealle mallille ja näkymälle. Kun reitti on määritetty, ohjaimen toimintoa kutsutaan. Ohjain käsittelee pyynnön tulkkauksen, varmistaa, että oikeaa mallia kutsutaan ja että oikea näkymä palautetaan. Ohjaimet tulisi pitää suppeina ja mallit laajoina. Tämä helpottaa koodin uudelleenkäyttöä ja testaamista (CakePHP documentation

2012b). Yleensä ohjaimet hallitsevat yhden mallin logiikkaa, mutta niiden on myös mahdollista käsitellä useampia malleja. Kuviossa 31 esitetään ohjaimen luonti. Luokan nimi on sama kuin tietokannan taulun nimi, johon se linkitetään. Tämä luokka tallennetaan Controllers-kansioon nimellä users_controller.php. Scaffold-määrettä käytetään testausvaiheessa.

```
1  <?php
2  class UsersController extends AppController {
3      var $name = 'Users';
4      var $scaffold;
5  }
6  ?>
```

KUVIO 31. Ohjaimen luonti

Sovelluksen ohjain perii AppController-luokan. AppController-luokka perii Controller-luokan samoin kuin mallissa. Kaikki määreet ja metodit, jotka luodaan AppController-luokkaan, ovat käytettävissä kaikissa ohjaimissa. Kuviossa 32 on esitetty ohjaimen luokkakaavio. Oma1 ja Oma2 ovat omia ohjaimia, jotka perivät AppController-luokan.



KUVIO 32. Ohjain luokkakaavio

Kuviossa 33 on esitetty, miten ohjain lataa mallin. $modelClass kertoo mallin nimen ja $id mallin ID-kentän arvon (rivi 656). Jos mallia ei löydy, tulee virhe

tietokantataulu puuttuu ja Cake luo dynaamisen mallin toistaiseksi (rivi 658 ja 671-672). Funktio listaa myös mallille määritetyt liitännäiset (rivi 666).

```
656         public function loadModel($modelClass = null, $id = null) {
657                 if ($modelClass === null) {
658                         $modelClass = $this->modelClass;
659                 }
660
661                 $this->uses = ($this->uses) ? $this->uses : array();
662                 if (!in_array($modelClass, $this->uses)) {
663                         $this->uses[] = $modelClass;
664                 }
665
666                 list($plugin, $modelClass) = pluginSplit($modelClass, true);
667
668                 $this->{$modelClass} = ClassRegistry::init(array(
669                         'class' => $plugin . $modelClass, 'alias' => $modelClass, 'id' => $id
670                 ));
671                 if (!$this->{$modelClass}) {
672                         throw new MissingModelException($modelClass);
673                 }
674                 return true;
675         }
```

KUVIO 33. loadModel-funktio

4.7 Näkymä

Näkymä-tiedostoilla on pääte .ctp (CakePHP Template) ja nämä tiedostot sisältävät logiikan, jolla ohjaimelta tuleva tieto esitetään. Näkymä-tiedostot tallennetaan app/View-kansioon, jonne luodaan kansio samalla nimellä kuin ohjain. Näkymät nimetään samoiksi kuin ohjaimen toiminnot, johon näkymä liittyy eli ohjaimessa voi olla toiminto add ja silloin näkymä-tiedosto on nimeltään add.ctp (CakePHP documentation 2012i).

Näkymä-kerros koostuu eri osista:

- Näkymät ovat sivun eri osia, jotka kohdistuvat tiettyyn toimintoon.
- Asettelut (layout) ovat näkymä-tiedostoja, jotka sisältävät sivun ulkoasuun liittyvää koodia.
- Avustajat kapseloivat näkymien logiikkaa. Avustajien avulla voi tehdä lomakkeita, AJAX-toiminnallisuutta, sivuttaa mallilta tulevaa tietoa tai näyttää RSS-syötteitä.

Kuviossa 34 esitetään esimerkki näkymän luomisesta. Näkymään kirjoitetaan olio-kutsuja html-tagien sekaan. Tässä esimerkissä div-elementtiin haetaan otsikko ja sisältö. Tämä tallennetaan .ctp-tiedostona View-kansiossa sijaitsevaan kansioon, joka on nimetty samannimiseksi kuin ohjain.

```
<html>
    <head>
        <title><?php echo $title_for_layout?></title>
        <?php echo $this->Html->css('tyylit'); ?>
    </head>
<body>
    <div class="sisalto">
        <h2><?php echo $this->fetch('title'); ?></h2>
        <?php echo $this->fetch('content'); ?>
    </div>
</body>
</html>
```

KUVIO 34. Esimerkki näkymätiedostosta

4.8 Komponentit

Komponentit ovat ohjaimien käytettävissä. Jos useassa eri ohjaimessa on samoja toimintoja, ne voi ottaa käyttöön molempiin komponentin kautta. CakePHP:ssä on myös valmiina komponentteja.

- tietoturvakomponentti
- sessionien käsittely
- sähköpostitus
- evästeiden käsittely
- käyttäjien todennus

4.9 Tietokanta

Hyvin suunniteltu tietokanta on tärkeä asia CakePHP:n toimivuuden kannalta. Monet käytännöt on sidottu tietokannan rakenteeseen. Scaffolding-ominaisuus, josta kerrotaan seuraavassa luvussa, ei toimi, jos tietokanta on huonosti suunniteltu. CakePHP nojaa vahvasti tietokantataulujen nimiin mallin ja ohjaimen

toiminnassa (Golding 2008, 39). CakePHP tukee monia tietokantoja, kuten PostgreSQL-, MySQL-, IBM DB2-, MSSQL-, Oracle-, SQLite- ja ADOdb-tietokantaa. Tässä työssä on käytetty MySQL-tietokantaa. Myös XML-tiedostoja voi käyttää tietolähteinä.

4.9.1 Tietokanta-yhteydet

CakePHP:ssä on neljää erilaista tietokantayhteyttä. Data mapping -mekanismilla voidaan luoda yhteys kahden eri tietomallin välille. CakePHP:ssä data mapping -mekanismia käytetään linkittämään tietokannan tauluja keskenään. Taulukossa 1. on esitetty tietokantataulujen yhteystyypit.

TAULUKKO 1. Tietokannan taulujen suhteet (CakePHP documentation 2012a)

Suhde	**Yhdistämistyyppi**	**Esimerkki**
yksi-yhteen	hasOne	Käyttäjällä on yksi profiili
yksi-moneen	hasMany	Käyttäjällä on useita artikkeleita
monta-yhteen	belongsTo	Monta artikkelia kuuluu käyttäjälle
monta-moneen	hasAndBelongsToMany	Artikkelilla on, ja kuuluu monelle eri lähteelle

Yhteydet määritellään malliluokassa. Yhteyden voi määritellä yhdelle muuttujalle tai taulukolle. Kuviossa 35 esitetään yhteyden määrittäminen taulukon 1. esimerkin mukaisesti. Käyttäjällä (User) on yksi profiili (rivi 4) ja monta artikkelia (rivi 5), jotka on hyväksytty (rivi 8) ja lajiteltu nousevasti (rivi 9).

```
1  <?php
2  class User extends AppModel {
3      public $name = 'User';
4      public $hasOne = 'Profile';
5      public $hasMany = array(
6          'Article' => array(
7              'className'  => 'Article',
8              'conditions' => array('Article.approved' => '1'),
9              'order'      => 'Article.created DESC'
10         )
11     );
12 }
13 ?>
```

KUVIO 35. Tietokantayhteyksien määrittäminen koodissa

4.9.2 Scaffold

Scaffolding-ajatus on lähtöisin Ruby on Rails -suunnittelumallista. Scaffolding liittyy läheisesti tietokantapohjaisiin sovelluksiin. Tällä tekniikalla ohjelmoija määrittelee, miten sovellus luo, hakee ja päivittää (CRUD) tietoja. CakePHP:ssä scaffolding on websovelluksen rakentamisen apuna. Se on erinomainen tapa selvittää tietokannan toimivuus ja relaatiot ennen varsinaista websivuston luomista. Myöhemmässä vaiheessa muutosten teko vie aikaa (Golding 2008). CakePHP:n scaffold olettaa, että kenttä, jonka nimi päättyy _id, on viiteavain tauluun, joka on nimetty samoin kuin mitä edeltää _id:tä (CakePHP documentation 2012a).

4.10 Avustajat

CakePHP:n mukana tulee avustajia. Seuraavaksi esitellään tärkeimpiä sisäänrakennettuja avustajia, mutta on myös olemassa asennettavia avustajia. Yleensä avustajaa käytetään ohjaimessa, mutta avustajaa voi kutsua myös näkymästä muuttujalla, joka on saman niminen kuin avustaja. (Golding 2008, 137.)

4.10.1 HTML-avustaja

Html-avustajalla voi kirjoittaa helposti html-merkkauskieltä. Tämä helpottaa esimerkiksi taulukoiden tekemistä (Golding 2008, 148.) Kuviossa 36 esitetään, miten luodaan taulukko html-avustajalla.

```
1  <?php
2  echo "<table>";
3  echo $this->Html->tableCells(array(
4      array('Sarake1', 'Sarake2', 'Sarake3'),
5      array('Sarake1', 'Sarake2', 'Sarake3'),
6  ));
7  echo "</table>";
8  ?>
```

KUVIO 36. Taulukko html-avustajalla.

Kuvion 36. mukainen koodi luo kuvion 37 mukaisen taulukon.

```
<table>
    <tr>
        <td>Sarake1</td>
        <td>Sarake2</td>
        <td>Sarake3</td>
    </tr>
    <tr>
        <td>Sarake1</td>
        <td>Sarake2</td>
        <td>Sarake3</td>
    </tr>
</table>
```

KUVIO 37. Taulukko

4.10.2 Form-avustaja

Form-avustajalla luodaan lomakkeita. Kuviossa 38 esitetään, miten näkymässä on luotu lisää uusi käyttäjä lomake. Type-määreellä voidaan määritellä, millaisen lomakekentän avustaja luo. Jos kentän arvo on taulun nimi ja alaviiva id, CakePHP olettaa, että se on viiteavain ja hakee automaattisesti pudotusvalikkoon arvot kyseisen taulun name-kentästä. Mikäli tietokannassa kentän arvo on not null eli sitä ei saa jättää tyhjäksi, CakePHP laittaa punaiset tähdet merkitsemään, että se on pakollinen tieto. Jos kentän on jättänyt tyhjäksi, ja kun painaa tallenna (Submit)-painiketta, tulee virheilmoitus ja kehotus täyttää puuttuva tieto. Kuviossa 39 esitetään, miltä lomake näyttää.

```
<div class="users form">
<?php echo $this->Form->create('User');?>
        <fieldset>
                <legend><?php echo __('Add User'); ?></legend>
        <?php
                echo $this->Form->input('name');
                echo $this->Form->input('password',
                        array('type' => 'password'));
                echo $this->Form->input('firstname');
                echo $this->Form->input('lastname');
                echo $this->Form->input('office_id');
        ?>
        </fieldset>
<?php echo $this->Form->end(__('Submit'));?>
</div>
```

KUVIO 38. Esimerkki form-avustajan käytöstä

KUVIO 39 Lomake, joka on tehty form-avustajalla

4.10.3 JS-avustaja

Aiemmin CakePHP:ssä oli Ajax-avustaja sekä javascript-avustaja, jotka nyt on korvattu JS-avustajalla. JS-avustajalla tehdään jQuery-toimintoja. JS-avustajaa varten pitää ladata haluttu javascript-kirjasto, joka viedään webrootin js-kansioon. Sivun head-osassa pitää kyseistä kirjastoa kutsua, jotta se on käytössä. Kuviossa 40 on esimerkki, miten kirjasto otetaan käyttöön. Jquery-sanan tilalle vaihdetaan kirjaston nimi (CakePHP Documentation 2012f) .

```
1  <?php
2  echo $this->Html->script('jquery');
3  ?>
```

KUVIO 40. JS-avustajan käyttöönotto head-osiossa

4.10.4 Tekstiavustaja

Tekstiavustajan avulla pystyy muokkaamaan tekstiä eri tavoin, mm.

- Muuntaa teksti linkiksi.
- Muuntaa http- tai ftp-alkuiset tekstit hypertekstiksi.
- Korostamaan tekstiä.
- Lyhentämään tekstiä tietyn pituiseksi. (CakePHP Documentation 2012h.)

Kuviossa 41 esitetään, miten teksti-avustajaa käytetään. Ensimmäisessä esimerkissä sähköpostiosoite muutetaan linkiksi, toisessa esimerkissä sana korostettu korostetaan. Korostukselle pitää tehdä tyylimääre css-tiedostoon.

```php
<?php
// ensimmainen esimerkki
$teksti = 'ota yhteyttä osoitteeseen info@example.com';
$linked_text = $this->Text->autoLinkEmails($teksti);

// toinen esimerkki
$teksti = 'Jotain tekstia, korostettu sana';
echo $this->Text->highlight($teksti, 'korostettu',
array('format' => '<span class="highlight">\1</span>'));
?>
```

KUVIO 41. Teksti-avustajan käyttö

4.10.5 XML-avustaja

Monesti on tarvetta käyttää XML-tiedostoja tietolähteinä. PHP5-version mukana tulee jo XML-jäsentelijä, joten jäsentelijää ei erikseen tarvitse tehdä. Kun luodaan uusi xml-luokka, se muuttaa taulukon (array) SimpleXMLElement-elementiksi tai DOMDocument-objektiksi. Tämä toimii myös päinvastoin eli XML-elementistä tai DOMDocumentistä voidaan tehdä taulukko (CakePHP Documentation 2012c). Kuviossa 42 on esimerkki xml-käytöstä, jossa on tehty xml-tiedostosta taulukko. Kuviossa 43 on esimerkki xml-tiedostosta ja kuviossa 44 on xml-tiedostosta muodostettu taulukko.

```php
<?php
//SimpleXMLElement
$xml = Xml::build('tiedosto.xml', array('return' => 'simplexml'));

// DomDocument
$xml = Xml::build('tiedosto.xml', array('return' => 'domdocument'));
?>
```

KUVIO 42. XML-käyttö

```xml
<?xml version="1.0"?>
<menu>
    <popup>
        <menuitem>1</menuitem>
        <menuitem>2</menuitem>
        <menuitem>3</menuitem>
    </popup>
</menu>
```

KUVIO 43. Esimerkki xml-tiedostosta

```php
<?php
$xmlArray = array(
    'menu' => array(
        'popup' => array(
            array(
                'menuitem' => 1),
            array(
                'menuitem' => 2),
            array(
                'menuitem' => 3)
            )
        )
    );
?>
```

KUVIO 44. xml-tiedostosta muodostettu taulukko

4.11 Komponentit

Seuraavaksi esitellään CakePHP:n yleisimpiä komponentteja. Komponentit sijaitsevat controllers-kansion alakansiossa components.

4.11.1 Sivutus

Jos tietokannasta tulee tietueita paljon, niitä on hankala lukea yhdeltä sivulta. Sivuttaja jakaa tietueet useammalle sivulle. Sivuttajaa on myös mahdollista käyttää JS-avustajan kanssa ajax-tyylisten linkkien luomiseksi (CakePHP Documentation 2012c), esim. tietueita voi selata helpommin, jos jokainen tietue muuttuu eri väriseksi, kun sen päälle vie hiiren. Kuviossa 45 on esitetty, miten sivutus lisätään ohjaimeen. Sivujen määrä on rajoitettu 25:een. Monia muitakin määreitä sivutukseen voi lisätä, esimerkiksi miten tietueet ryhmitellään tai missä järjestyksessä ne luetellaan.

```
1  <?php
2  class UsersController extends AppController {
3
4      public $paginate = array(
5          'limit' => 25
6      );
7  }
8  ?>
```

KUVIO 45. Esimerkki sivutuksesta ohjaimessa

Kuviossa 46 on esitetty, miten näkymässä sivutus otetaan käyttöön. Rivillä 31 on previous- eli takaisin-linkki. Rivillä 34 on next- eli seuraava-linkki. Rivillä 33 on määritetty erotin eli se, mitä sivunumeroiden väliin tulee. Tässä tapauksessa se on välilyönti.

```
29          <div class="paging">
30          <?php
31                  echo $this->Paginator->prev('< ' . __('previous'),
32                          array(), null, array('class' => 'prev disabled'));
33                  echo $this->Paginator->numbers(array('separator' => ''));
34                  echo $this->Paginator->next(__('next') . ' >',
35                          array(), null, array('class' => 'next disabled'));
36          ?>
37          </div>
```

KUVIO 46. Esimerkki sivutuksesta näkymässä

4.11.2 Autentikointi

Käyttäjien tunnistamista tarvitaan verkkosovelluksissa usein. CakePHP:ssä on tätä varten suunniteltu autentikointi-komponentti, jonka avulla voi myös tarkistaa käyttöoikeuden kyseiseltä käyttäjältä. Autentikointitapoja on kolme:

- lomakeautentikointi, joka on oletustapa: käyttäjätunnus ja salasana kulkevat POST-metodilla
- basic autentikointi
- digest autentikointi.
    - Basic ja Digest ovat molemmat HTTP-autentikointeja, joita voi käyttää esimerkiksi Webpalveluja käytettäessä. Ero näiden kahden välillä on salasanan käsittelyn välillä. Digest autentikoinnissa salasana on tiivistetty eli salattu. (CakePHP documentation 2012c).

Kuviossa 47 esitetään, miten autentikointi-komponenttia käytetään. Rivillä 3 kerrotaan, että malli, jota halutaan käyttää, on User, rivillä 4 kerrotaan, minkä kenttien arvoja tutkitaan. Autentikointi-komponentilla voidaan myös rajata oikeuksia. Rivillä 6 on deny-funktiokutsu, jossa on poistettu lisäys- ja muokkausoikeudet.

```
1  <?php
2  $this->Auth->authenticate = array('Form');
3  $this->Auth->userModel = 'User';
4  $this->Auth->fields = array('username' => 'email', 'password' => 'password');
5  // lisays- ja editointioikeudet poistettu
6  $this->Auth->deny(array('add', 'edit'));
7  ?>
```

KUVIO 47. Autentikointi-komponentti esimerkki

4.11.3 Sähköpostitus

Sähköpostitus toimii luokkakutsuna, ja sitä voi käyttää sovelluksen missä osassa tahansa. Kuviossa 48 on esitetty, miten sähköpostitus-toimintoa käytetään. Rivillä 3 on lähettäjä, tässä esimerkissä taulukkona. Se toimisi myös ilman taulukkoa, mutta lähettäjäksi voi laittaa myös jotain muuta kuin varsinaisen sähköpostiosoitteen. Tässä tapauksessa vastaanottajan lähettäjäkentässä lukee sivuston nimi. Rivillä 4 on vastaanottajan osoite. Rivillä 5 on viestin aihe ja rivillä 6 on varsinainen viestiteksti.

```
1  <?php
2  $email = new CakeEmail();
3  $email->from(array('kenelta@sposti.com' => 'Sivuston nimi'));
4  $email->to('kenelle@sposti.fi');
5  $email->subject('Mita asia koskee');
6  $email->message('Viesti');
7  $email->send();
8  ?>
```

KUVIO 48. Esimerkki sähköpostituksesta

4.11.4 Sessionit

Session-komponentin avulla voi tallentaa tietoa pysyvästi liikuttaessa sivujen välillä. Kuviossa 49 on esimerkki sessioniin kirjoittamisesta ja lukemisesta. Sessionissa käytetään piste-notaatiota eli User.name muodostuu User-nimisestä sessiosta ja name-nimisestä arvosta.

```
1  <?php
2  // Sessioniin kirjoittaminen
3  $this->Session->write('User.name', 'Testaaja');
4
5  // Sessionista lukeminen
6  $username = $this->Session->read('User.name');
7  ?>
```

KUVIO 49. Esimerkki sessioniin kirjoittamisesta ja lukemisesta

4.12 PHPCaken tietoturva

Verkkosivuja vastaan tehdään hyökkäyksiä. SQL-injektiot ovat tietokantoihin kohdistuvia hyökkäyksiä, joissa hyökkääjä antaa tietokantapalvelimelle SQL-komentoja, joilla yritetään esimerkiksi poistaa tietoa tai tauluja. CakePHP:n sanitize-luokka on tarkoitettu haitallisten hyökkäyksien estämiseen. Sanitize on kirjasto ytimessä. Sitä voi käyttää missä vain, mutta parhaiten se toimii mallissa tai ohjaimessa. Periaatteena on, että tietokannassa säilytetään HTML-muotoista tietoa, joka siistitään näytettäessä. (PHPCake documentation 2012d.) Kuviossa 50 on esimerkki sanitize-luokan käytöstä.

```
1  <?php
2  $this->data = Sanitize::clean($this->data, array('encode' => false));
3  ?>
```

KUVIO 50, Sanitize-luokan kutsu

Tietoturva-komponentin avulla saa tietoturvaa paremmaksi sovelluksessa. Sillä saa estettyä mm.

- sen millaisia HTTP-pyyntöjä sovellus hyväksyy.
- cross site scripting -hyökkäyksen, joka mahdollistaa koodin syöttämisen, ja sen avulla mahdollisesti tunkeutumisen verkkosivulle (Wikipedia 2012d.)
- lomakkeen manipulointia eli SQL-injektointeja (CakePHP Documentation 2012d).

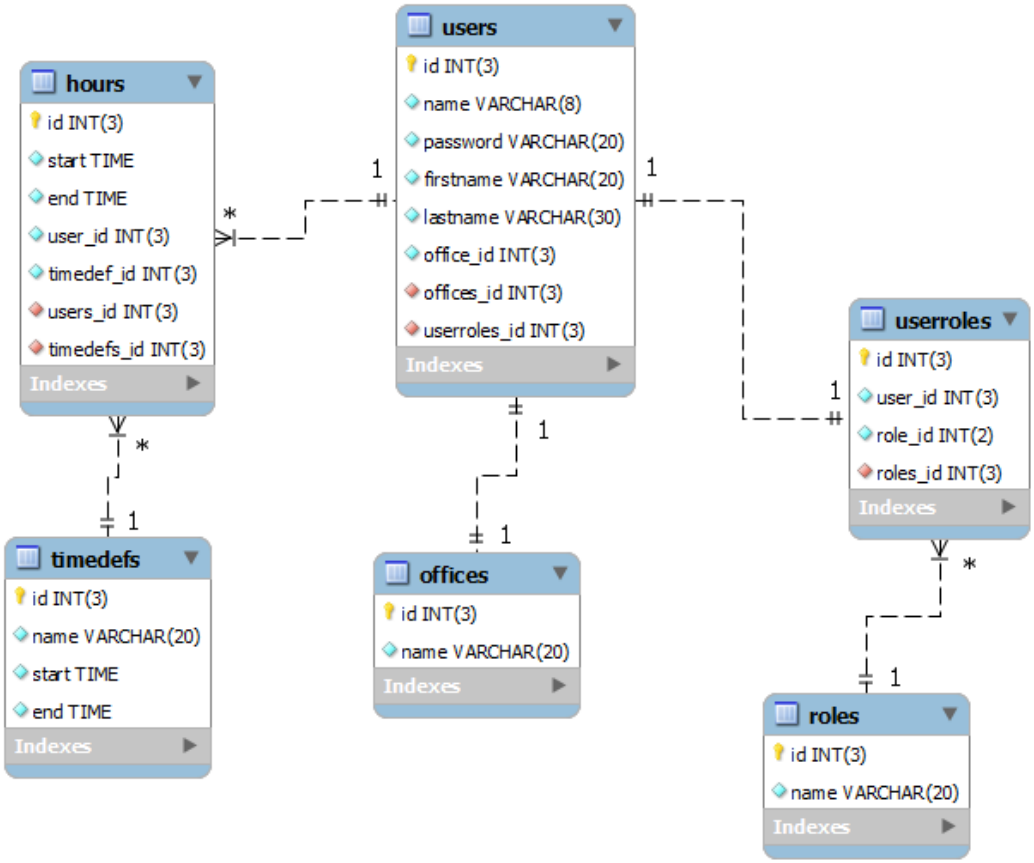
5 SOVELLUS

5.1 Sovelluksen asiakasvaatimukset

Sovellus on työtuntien kirjaamista varten. Sovelluksessa pitää olla sisäänkirjautuminen ja käyttäjien tunnistus. Sovellus toimii selainpohjaisena internetissä, joten julkisesti siitä ei saa näkyä kuin sisäänkirjautumislomake. Käyttäjien pitää pystyä kirjaamaan työtuntejansa ja esimiehen pitää pystyä lisäämään käyttäjiä, käyttäjärooleja, työaikojen määritelmiä, toimipisteitä sekä hyväksymään työntekijöiden ilmoittamia työtunteja. Työaikojen määritelmiin kuuluu mm. normaali työaika, iltatyö, yötyö, loma-aika. Työtunnit on pystyttävä viemään myös Exceliin, koska palkanlaskentaohjelma on yhteensopiva Excelin kanssa.

5.2 Tietokanta

Kuviossa 51 on esitetty tietokannan rakenne.



KUVIO 51. Tietokannan rakenne

Tietokannan suhteet:

- users-taulusta on yksi-yhteen suhde userroles taulun user_id-kenttään. Yhdellä käyttäjällä voi olla vain yksi rooli.
- users-taulusta on yksi-moneen suhde hours-tauluun. Yhdellä käyttäjällä on monta työaikaa.
- users-taulusta on yksi-yhteen suhde offices-tauluun. Yhdellä käyttäjällä on yksi toimipiste.
- roles-taulusta on yksi-moneen suhde userroles-tauluun. Roolilla voi olla monta eri käyttäjää.
- times_def-taulusta on yksi-moneen suhde times-tauluun. Yhdellä työajan määrityksellä voi olla monta eri työaikaa.

Kuviossa 52 on esitetty tietokannan luontilauseet. Jokaisessa taulussa on auto_increment-muotoinen id-kenttä, joka on pääavain. Auto_increment tarkoittaa sitä, että arvo lisääntyy automaattisesti yhdellä aina kun uusi rivi tallennetaan.

Viiteavaimia ei CakePHP:ssä määritetä tietokantaan, vaan ne otetaan huomioon koodissa. Users-tauluun tulee käyttäjät ja office_id-kentällä määritetään, missä käyttäjän toimipiste on. Hours-tauluun lisätään käyttäjän työtunnit ja start- ja end-kentät määrittävät työajat ja timedef_id, onko työaika normaali työaikaa vai ilta- tai yötyötä. Timedefs-tauluun määritetään normaali-, ilta- ja yötyö. Office-tauluun määritetään toimipisteet, ja id-kenttä on viiteavain users-taulun office_id-kenttään. Roles-tauluun määritetään roolit, ja taulun id-kenttä on viiteavain userroles-taulun role_id-kenttään.

```
CREATE TABLE IF NOT EXISTS `users` (
  `id` int(3) NOT NULL AUTO_INCREMENT,
  `name` varchar(8) NOT NULL,
  `password` varchar(20) NOT NULL,
  `firstname` varchar(20) NOT NULL,
  `lastname` varchar(30) NOT NULL,
  `office_id` int(3) NOT NULL,
  PRIMARY KEY (`id`)
) ;

CREATE TABLE IF NOT EXISTS `offices` (
  `id` int(3) NOT NULL AUTO_INCREMENT,
  `name` varchar(20) NOT NULL,
  PRIMARY KEY (`id`)
);

CREATE TABLE IF NOT EXISTS `roles` (
  `id` int(3) NOT NULL AUTO_INCREMENT,
  `name` varchar(20) NOT NULL,
  PRIMARY KEY (`id`)
);

CREATE TABLE IF NOT EXISTS `userroles` (
  `id` int(3) NOT NULL AUTO_INCREMENT,
  `user_id` int(3) NOT NULL,
  `role_id` int(2) NOT NULL,
  PRIMARY KEY (`id`)
) ;

CREATE TABLE IF NOT EXISTS `hours` (
  `id` int(3) NOT NULL AUTO_INCREMENT,
  `start` time NOT NULL,
  `end` time NOT NULL,
  `user_id` int(3) NOT NULL,
  `timedef_id` int(3) NOT NULL,
  `approved` INT( 1 ) NOT NULL DEFAULT '0',
  PRIMARY KEY (`id`)
) ;

CREATE TABLE IF NOT EXISTS `timedefs` (
  `id` int(3) NOT NULL AUTO_INCREMENT,
  `name` varchar(20) NOT NULL,
  `start` time NULL,
  `end` time NULL,
  PRIMARY KEY (`id`)
) ;
```

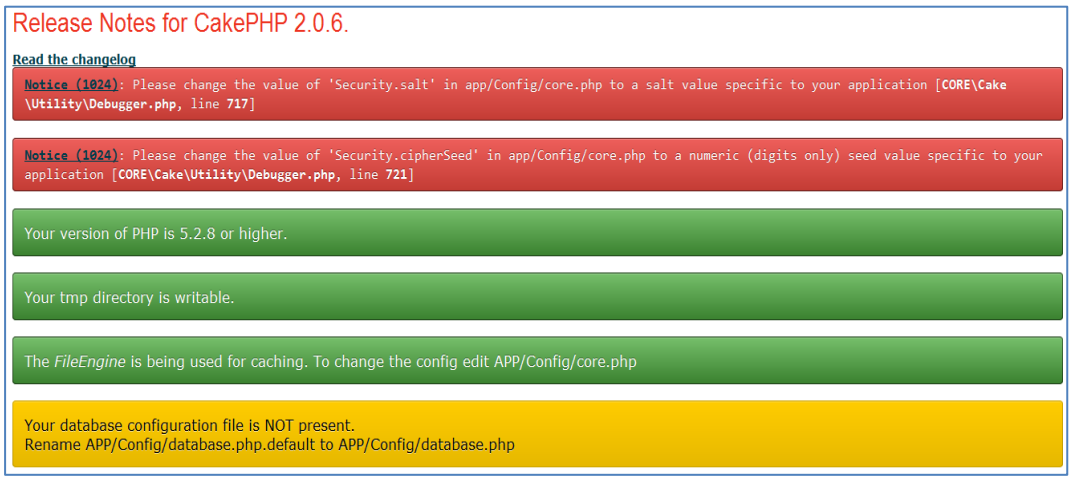
KUVIO 52. Tietokannan luontilauseet

5.3 Sivuston luominen

Sivuston voi luoda joko kirjoittamalla tarvittavat tiedostot itse tai vaihtoehtoisesti niin, että CakePHP luo automaattisesti ohjaimet, mallit ja näkymät. Tässä käydään läpi asennus konsolin asennusvelhon avulla.

5.3.1 Sivuston luomisen alkuvalmistelut

Kun CakePHP:n asennusosoitteeseen mennään ensimmäisen kerran saadaan kuvion 53. mukainen sivu. Siinä näkyvät ensin asiat, jotka vaativat korjausta. Esim. tässä Security.salt- ja Security.cipherSeed-arvot pitää muuttaa tietoturvasyistä ja tietokannan konfigurointi-tiedosto pitää muuttaa database.php.default-nimisestä database.php-nimiseksi.



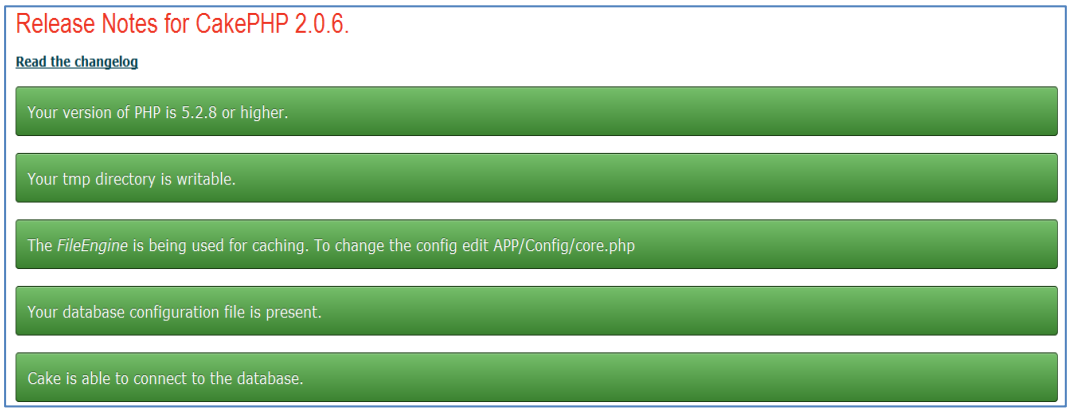
KUVIO 53. CakePHP:n ensimmäisen julkistuksen sivu

Security.Salt ja Security.cipherSeed arvot ovat satunnaisia merkkijonoja. Ne pitää muuttaa oletuksesta joksikin muuksi tietoturvasyistä. CakePHP käyttää security-arvoja tiivisteiden ja session-muuttujien muodostamiseen, ja ne täytyy olla jotain muuta kuin oletusmerkkijonot. Kuviossa 54 on esitetty merkkijonot.

```
184 /**
185  * A random string used in security hashing methods.
186  */
        Configure::write('Security.salt', 'DYhG93b0qyJfIxfs2guVoUubWwvniR2G0FgaC9mK');
188
189 /**
190  * A random numeric string (digits only) used to encrypt/decrypt strings.
191  */
192         Configure::write('Security.cipherSeed', '76859309657453542496749683645');
193
```

KUVIO 54. Security.salt ja Security.cipherSeed merkkijonot

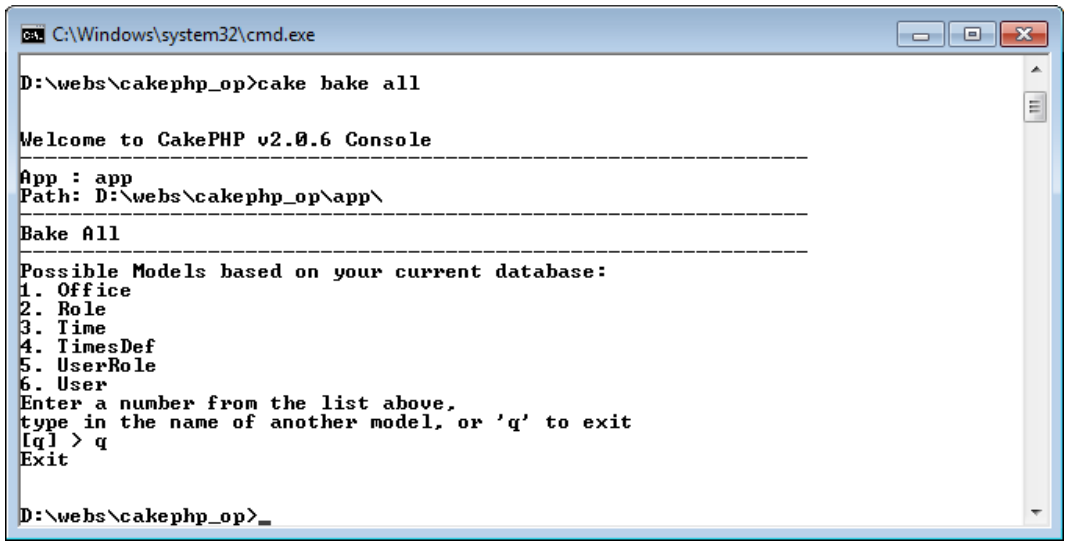
Kun myös database-tiedosto on uudelleen nimetty, ensimmäisestä sivusta tulee kuvion 55. mukainen. Kaikki on kunnossa, eikä muutostarpeita ole.



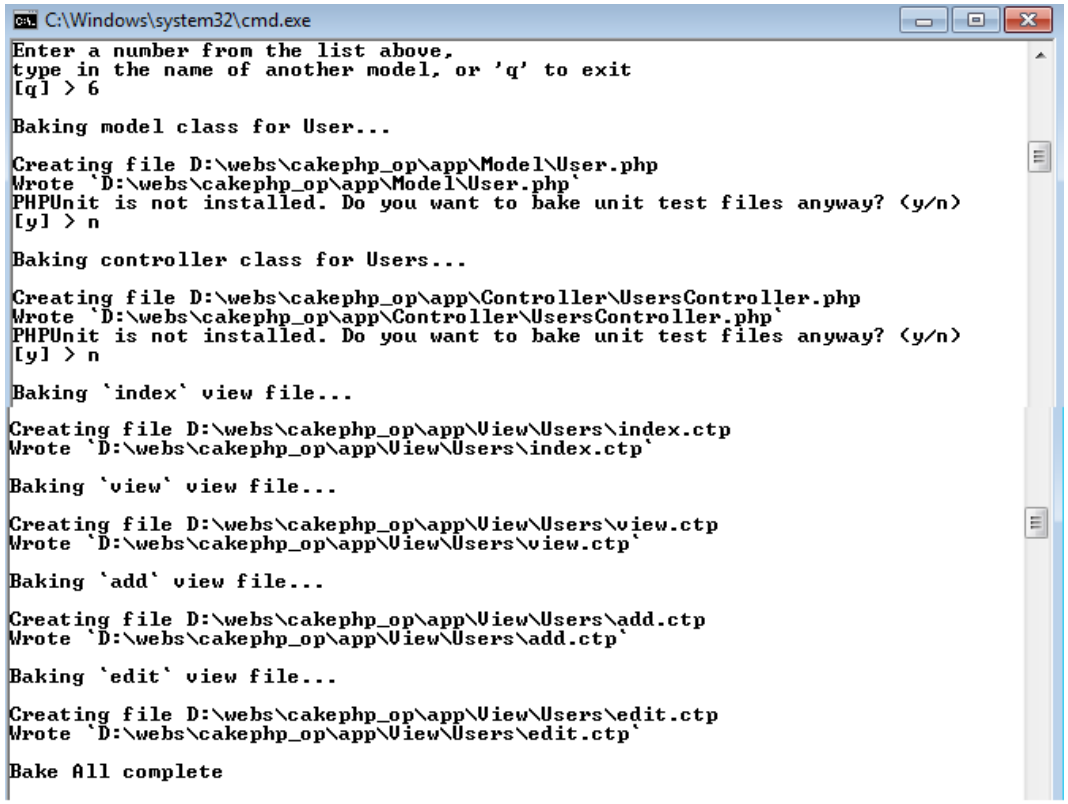
KUVIO 55. CakePHP:n aloitussivu

5.3.2 Mallin, ohjaimen ja näkymän luominen asennusvelhon avulla

Asennusvelhoa käytetään konsoli-ikkunan kautta. Jotta CakePHP:n käskyt toimisivat, konsolin polku pitää lisätä ympäristömuuttujiin. Kun antaa käskyn cake bake all, ohjelma generoi mallin, ohjaimen ja näkymän. Kuviossa 56 ja 57 on esitetty, miltä konsoli-ohjelma näyttää. Oletuksena malli, ohjain ja näkymä luodaan tietokantataulujen mukaan.



KUVIO 56. Konsolin aloitusnäkymä



KUVIO 57. Konsoli ’leipoo’ eli luo mallia, ohjainta ja näkymää

Tämä sama tehdään kaikille tauluille. Kuviossa 58 on esimerkki mallista, jonka CakePHP on luonut. Siinä näkyy pieni osa koodia. Rivillä 90 näkyy moneen-yhteys ja rivillä 93 on määritetty viiteavain.

```
74      */
75              public $belongsTo = array(
76                      'Office' => array(
77                              'className' => 'Office',
78                              'foreignKey' => 'office_id',
79                              'conditions' => '',
80                              'fields' => '',
81                              'order' => ''
82                      )
83              );
84
85      /**
86       * hasMany associations
87       *
88       * @var array
89       */
90              public $hasMany = array(
91                      'Userrole' => array(
92                              'className' => 'Userrole',
93                              'foreignKey' => 'user_id',
94                              'dependent' => false,
95                              'conditions' => '',
96                              'fields' => '',
97                              'order' => '',
98                              'limit' => '',
99                              'offset' => '',
100                             'exclusive' => '',
101                             'finderQuery' => '',
```

KUVIO 58. Osa user-taulun mallin koodia

Kuviossa 59 on esimerkki ohjaimen view- ja add-funktiosta, jonka CakePHP on luonut.

```
35          public function view($id = null) {
36                  $this->User->id = $id;
37                  if (!$this->User->exists()) {
38                          throw new NotFoundException(__('Invalid user'));
39                  }
40                  $this->set('user', $this->User->read(null, $id));
41          }
42
43  /**
44   * add method
45   *
46   * @return void
47   */
48          public function add() {
49                  if ($this->request->is('post')) {
50                          $this->User->create();
51                          if ($this->User->save($this->request->data)) {
52                                  $this->Session->setFlash(__('The user has b
53                                  $this->redirect(array('action' => 'index'))
54                          } else {
55                                  $this->Session->setFlash(__('The user could
56                          }
57                  }
                    $offices = $this->User->Office->find('list');
59                  $this->set(compact('offices'));
```

KUVIO 59. User-taulun ohjain

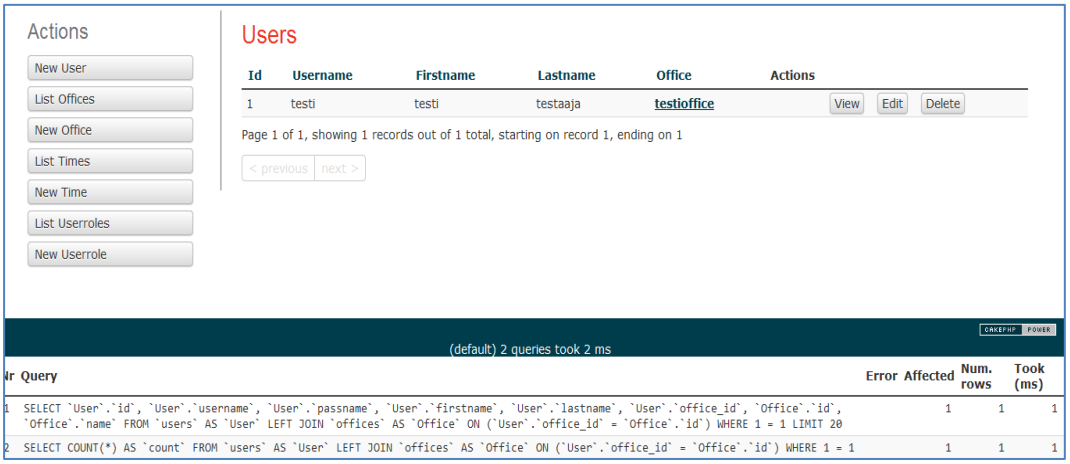
Kuviossa 60 on esimerkki näkymän index-tiedostosta, jonka CakePHP on luonut. Riveillä 5 - 10 on otsikot, joiden avulla tietueita pystyy lajittelemaan, ja rivillä 14 on foreach-silmukka, joka hakee tietokannasta riveillä 16 - 20 määritetyt rivit.

```
<div class="users index">
2           <h2><?php echo __('Users');?></h2>
3           <table cellpadding="0" cellspacing="0">
4           <tr>
5                       <th><?php echo $this->Paginator->sort('id');?></th>
6                       <th><?php echo $this->Paginator->sort('name');?></th>
7                       <th><?php echo $this->Paginator->sort('password');?></th>
8                       <th><?php echo $this->Paginator->sort('firstname');?></th>
9                       <th><?php echo $this->Paginator->sort('lastname');?></th>
10                      <th><?php echo $this->Paginator->sort('office_id');?></th>
11                      <th class="actions"><?php echo __('Actions');?></th>
12          </tr>
13          <?php
            foreach ($users as $user): ?>
15          <tr>
16                  <td><?php echo h($user['User']['id']); ?> </td>
17                  <td><?php echo h($user['User']['name']); ?> </td>
18                  <td><?php echo h($user['User']['password']); ?> </td>
19                  <td><?php echo h($user['User']['firstname']); ?> </td>
20                  <td><?php echo h($user['User']['lastname']); ?> </td>
```

KUVIO 61. User-taulun näkymä

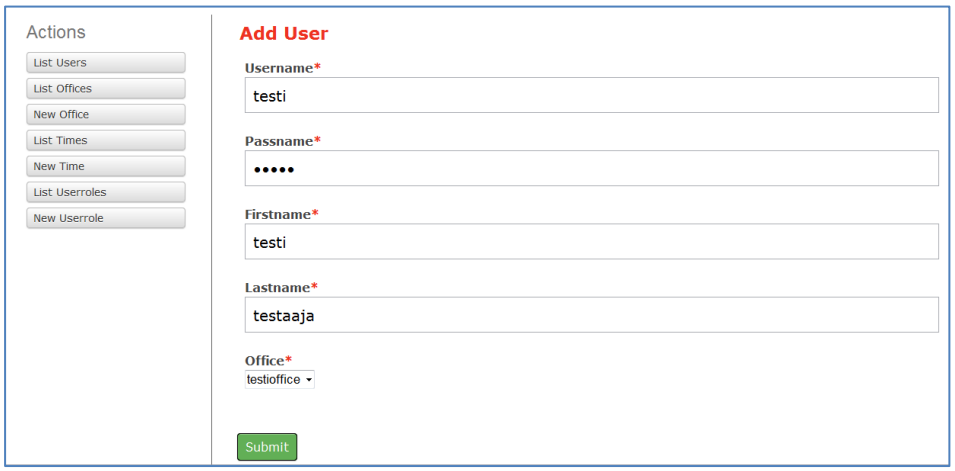
5.3.3 Tietokannan toimivuuden tarkistus

Tietokannan toimivuuden voi tarkistaa menemällä osoitteeseen <sivun osoite>/taulun nimi. Kuviossa 62 esitetään näkymä <sivun osoite>/users. Tietokantaan on lisätty valmiiksi tietueita. Actions-kohtaan on listattu users-tauluun liittyvät yhteydet. Nämä CakePHP on päätellyt sarakkeiden nimistä eli ne missä on _id-nimisiä sarakkeita näkyvät actioneina.



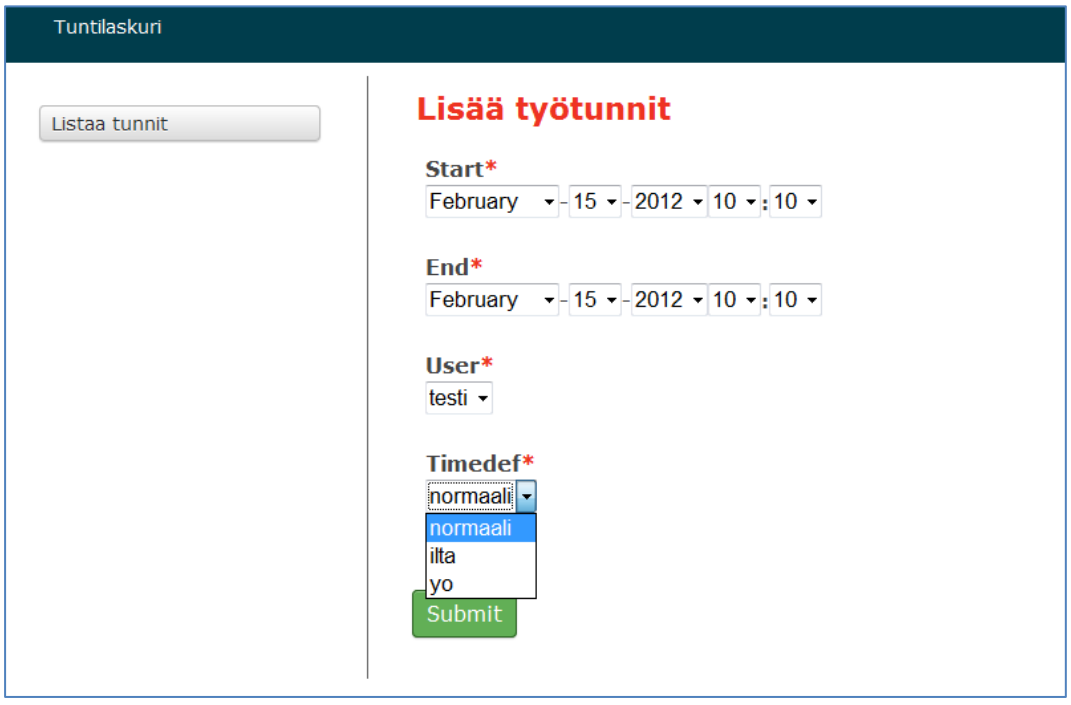
KUVIO 62. Users-näkymä

Kun painetaan new user-painiketta, saadaan kuvion 63. mukainen lomake, jolla voi lisätä uuden käyttäjän. Koska office-kenttä on nimetty office_id eli taulun nimi yksikössä ja _id, Cake osaa tehdä siihen pudotusvalikon, johon listataan kaikki nimi-arvot offices-taulusta. Mikäli uuden käyttäjän luominen onnistuu, tietokanta toimii tältä osin.



KUVIO 63. Lisää uusi käyttäjä lomake

Työtuntien lisäämislomake on kuvion 64. mukainen. CakePHP osaa tehdä aloitusajan ja lopetusajan pudotusvalikoina, koska tietokannassa tietotyyppinä on datetime. Timedef-pudotusvalikon CakePHP luo automaattisesti, koska hours-taulussa on timedef_id-niminen kenttä, jonka CakePHP osaa päätellä viiteavaimeksi.



KUVIO 64. Näkymä tuntien lisäämisestä

Kun toimintoja on testattu ja tietojen lisääminen, muokkaaminen ja poistaminen onnistuu oikein, voidaan päätellä, että tietokanta ja Cake toimivat hyvin yhteen.

5.4 Sovelluksen näkymät

Sovelluksen etusivulla on sisäänkirjautuminen, koska käyttäjä pitää pystyä tunnistamaan. Sovelluksessa on kahta eri käyttäjätasoa: admin ja user. Admin-oikeuksilla pystyy lisäämään, muokkaamaan ja poistamaan käyttäjiä, työajan määrittelyjä, toimipisteitä ja rooleja. User pystyy muokkaamaan omia tietojaan sekä lisäämään työtunnit. Admininilla on oikeudet hyväksyä työtunnit. Tässä sovelluksessa on kaikki toiminnot vain sisäänkirjautuneille.

Kuviossa 64 on aloitussivu.

KUVIO 64. Aloitussivu, login

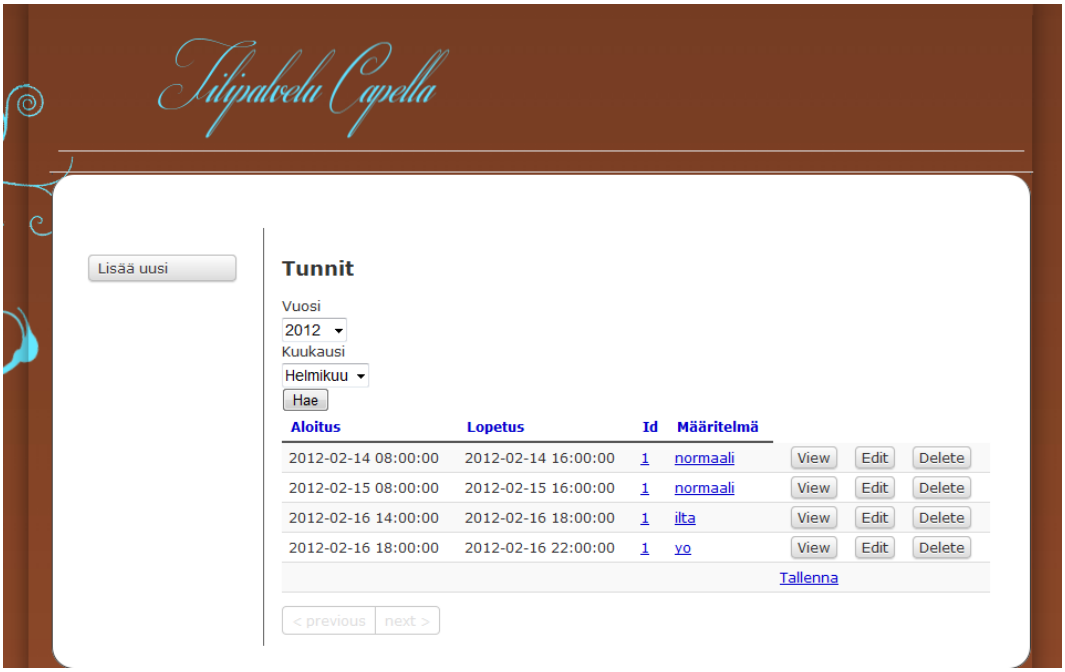
Kuviossa 65 esitetään, miten käyttäjän id luetaan.

```
1  <?php
2  $id = $this->Auth->user('id');
3  ?>
```

KUVIO 65. Käyttäjän id:n lukeminen

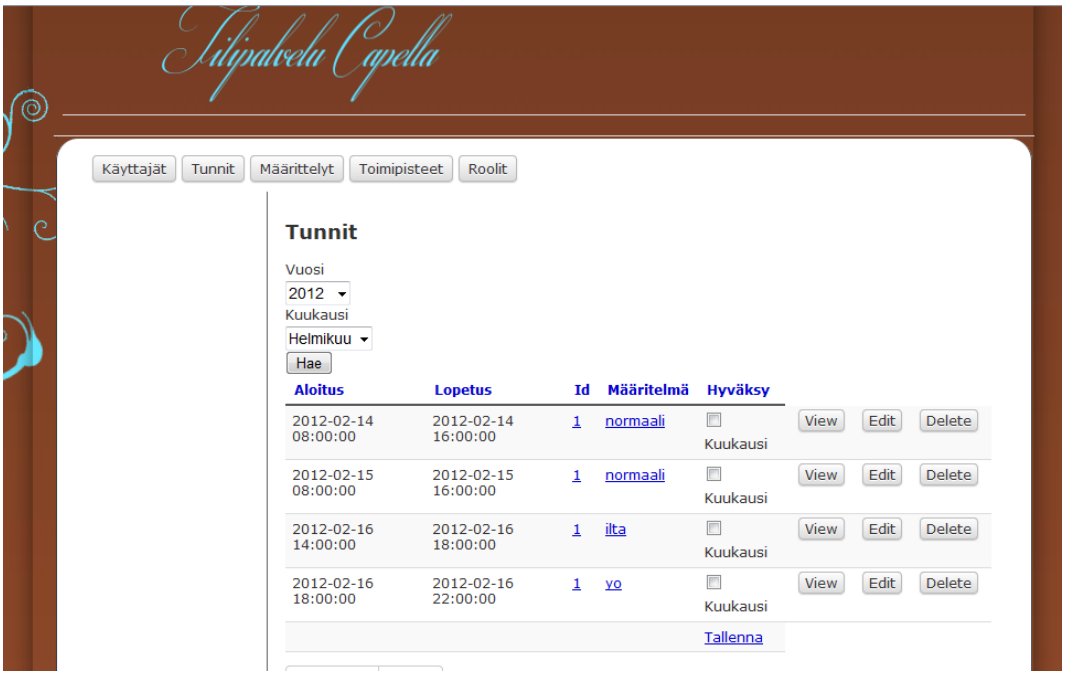
Listaa tunnit on user-oikeuksilla kuvion 66. mukainen. Otsikoista pystyy lajittelemaan tietueita. Lisää uusi -painikkeella pystyy lisäämään työaikoja, edit-

painikkeella pystyy muokkamaan työtuntia ja delete-painikkeella poistamaan työtunnin.



KUVIO 66. Näkymä listaa tunnit

Listaa tunnit-näkymä admin-oikeuksilla on kuvion 67. mukainen. Admin-käyttäjälle näkyy linkit tietojen ylläpitoon ja pystyy hyväksymään tunteja.



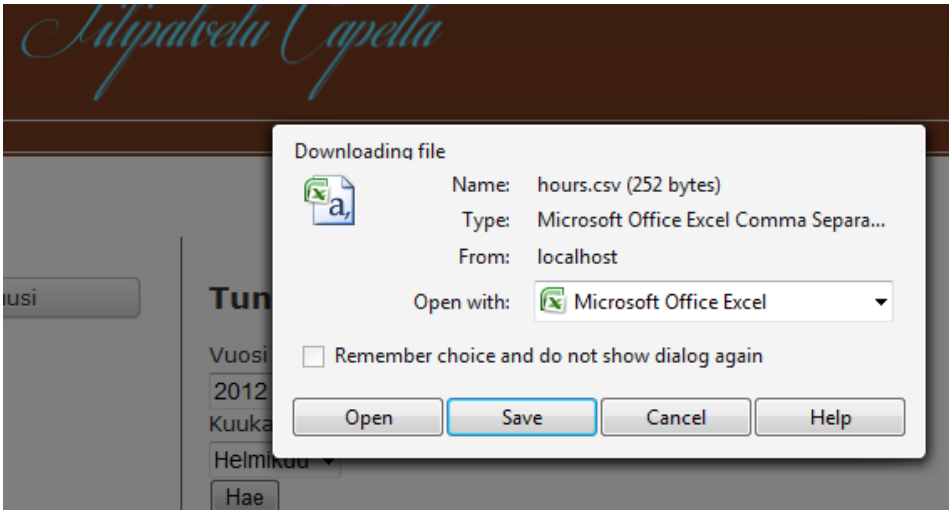
KUVIO 67. Näkymä listaa tunnit

Tunnit pitää pystyä viemään raporttina Exceliin, koska palkanlaskentaohjelmat ovat usein Excelin kanssa yhteensopivia, kuten toimeksiantajan tapauksessa on. Tätä varten on kirjoitettu oma exportcsv-funktio, josta kuviossa 68 näkyy osa koodista.

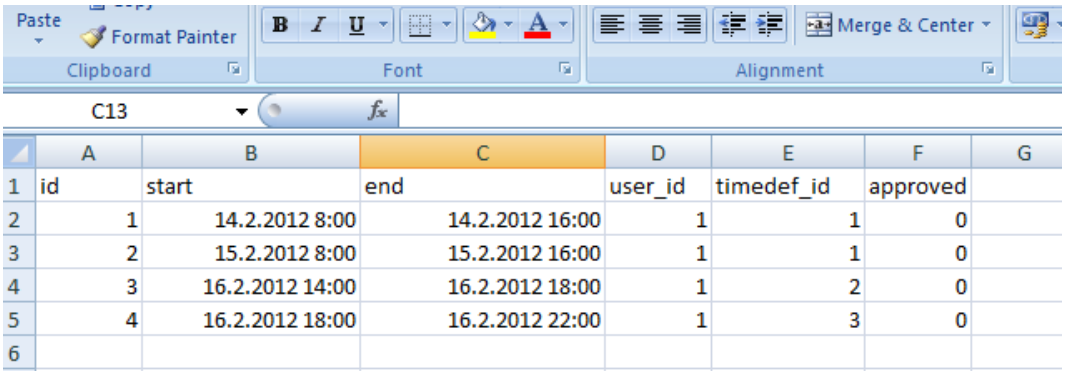
```
69      public function exportCSV(Model &$Model ) {
70          if( !ini_get('safe_mode') && ini_get('max_execution_time') < $this->
71              set_time_limit($this->settings[$Model->alias]['max_execution_tim
72          }
73
74          $Model->recursive = -1;
75          $records = $Model->find('all');
76          if ( !empty($records ) ) {
77              $this->ensureTmp();
78
79              $tmpFilename = TMP . $this->tmpDir . DS .  strtolower( Inflector
80              $fp = fopen($tmpFilename, 'w');
81              if ( $this->settings[$Model->alias]['encoding'] == 'utf8' ) {
82                  fwrite($fp, "\xEF\xBB\xBF");
83                  fputcsv( $fp, $this->arrayToUtf8( array_keys($records[0][$Mo
84              } else {
85                  fputcsv( $fp, array_keys($records[0][$Model->alias]), $this-
86              }
87
88              foreach( $records as $record ) {
89                  if ( $this->settings[$Model->alias]['encoding'] == 'utf8' )
90                      fputcsv($fp, $this->arrayToUtf8( $record[$Model->alias]
91                  } else {
92                      fputcsv($fp, $record[$Model->alias], $this->settings[$Mo
93                  }
94              }
```

KUVIO 68. Exportcsv-funktio

Kun näkymästä kutsutaan funktiota, avautuu Excel kuvion 69. mukaisesti. Kuviossa 70 näkyy tiedot Excelissä.



KUVIO 69. Excelin avautuminen



	A	B	C	D	E	F	G
1	id	start	end	user_id	timedef_id	approved	
2	1	14.2.2012 8:00	14.2.2012 16:00	1	1	0	
3	2	15.2.2012 8:00	15.2.2012 16:00	1	1	0	
4	3	16.2.2012 14:00	16.2.2012 18:00	1	2	0	
5	4	16.2.2012 18:00	16.2.2012 22:00	1	3	0	
6							

KUVIO 70. Tiedot Excelissä

Excelistä tiedot pystyy siirtämään kirjanpito-ohjelmaan.

6 YHTEENVETO

Työn tavoitteena oli tehdä tuntikirjanpito-ohjelma ohjelmointikehyksen CakePHP avulla. Työn toteuttamiseen meni aikaa noin neljä kuukautta. Ohjelmointikehyksen opetteleminen vei aikaa, eikä kaikkia kehyksen ominaisuuksia tullut testattua. Mutta kun ohjelmointikehys on tullut tutuksi, websovelluksen tekeminen sen avulla on nopeampaa kuin koodata kaikki alustapitäen. Tietokannan suunnitteluun kannattaa käyttää aikaa, jotta se olisi mahdollisimman täydellinen alusta asti. Jos tietokannan tauluihin pitää myöhemmin lisätä kenttiä, työ on melko iso, koska myöhemmin ei kannata enää mallia, ohjainta tai näkymää tehdä uudelleen konsolin avulla.

Ohjelmistokehyksien käyttö websivustojen tekemisessä on järkevää. Koodi pysyy paremmin hallinnassa, kun sivustolla on tietty rakenne. CakePHP:n nimeämiskäytännöt ovat loogisia. Useampi kehittäjä voi tehdä samaa sivustoa, kun kaikkien kehittäjien pitää seurata samaa nimeämiskäytäntöä. Myöskin sivuston ylläpitäjän vaihtaminen on helpompaa, jos edellinen sivuston ylläpitäjä siirtyy muihin tehtäviin.

CakePHP:n scaffolding-toiminnolla voi testata tietokannan yhteyksien toimivuuden ennen kuin varsinaista websivustoa aletaan tehdä. Kun tietokanta on luotu ja testattu, varsinaisen sivuston tekeminen on sen jälkeen vaivattomampaa. Scaffolding-toimintoa voi käyttää yleisestikin tietokantojen toimivuuden testaamiseen, vaikkei sivustoa CakePHP:llä toteuttaisikaan.

Tätä ohjelmaa voisi kehittää vielä toiminnallisemmaksi, jotta se sopisi yleisempään käyttöön. Tällä hetkellä se on räätälöity vain tilipalvelun tarpeisiin ja pääpaino on ollut tuntien saaminen helposti Exceliin. Tulevaisuudessa ohjelmassa voisi olla toimintoja, jotka laskisi työtunteja yhteen ja palkan sen perusteella, onko kyseessä iltatyötä vai normaalia työaikaa. Ohjelma voisi myös tehdä erilaisia raportteja.

LÄHTEET

Apache documentation 2012, [viitattu 12.2.2012]. Saatavissa: http://apache.org

CakePHP documentation 2012a, Associations [viitattu 23.2.2012]. Saatavissa: http://book.cakephp.org/2.0/en/models/associations-linking-models-together.html

CakePHP documentation 2012b, Controllers [viitattu 22.2.2012]. Saatavissa: http://book.cakephp.org/2.0/en/controllers.html

CakePHP documentation 2012 c, Core libraries [viitattu 7.3.2012]. Saatavissa: http://book.cakephp.org/2.0/en/core-utility-libraries/xml.html

CakePHP documentation 2010d, CSRF [viitattu 13.3.2012]. Saatavissa: http://bakery.cakephp.org/articles/T0aD/2008/06/11/protect-your-website-against-csrf-attack

CakePHP documentation 2012 e, Email [viitattu 9.3.2012]. Saatavissa: http://book.cakephp.org/2.0/en/core-utility-libraries/email.html

CakePHP documentation 2012f, JS-helper. [viitattu 8.3.2012]. Saatavissa: http://book.cakephp.org/2.0/en/core-libraries/helpers/js.html

CakePHP documentation 2012g, Models [viitattu 21.2.2012]. Saatavissa: http://book.cakephp.org/2.0/en/models.html

CakePHP documentation 2012h, Text-helper. [viitattu 8.3.2012]. Saatavissa: http://book.cakephp.org/2.0/en/core-libraries/helpers/text.html

CakePHP documentation 2012i, Views [viitattu 22.2.2012]. Saatavissa: http://book.cakephp.org/2.0/en/views.html

CakePHP readme-tiedosto. 2012.

Fowler, M 2012. Active Record [viitattu 29.3.2012]. Saatavissa: http://www.martinfowler.com/eaaCatalog/activeRecord.html

Golding, D. 2008. Beginning CakePHP. New York: Apress.

Jquery 2012, Jquery [viitattu 9.3.2012]. Saatavissa: http://jquery.com/

JSON 2012, JSON [viitattu 9.3.2012]. Saatavissa: http://www.json.org/

Korpela, J. 2012, Form methods [viitattu 10.3.2012]. Saatavissa: http://www.cs.tut.fi/~jkorpela/forms/methods.html

PHP documentation 2012, PHP [viitattu 24.2.2012]. Saatavissa: http://www.php.net/

Rapa, V. 2008. ProGradu. Kerroksittainen ja malli-näkymä-ohjain - ohjelmistoarkkitehtuurityyli.

Vanhatupa, J-M 2011. Luento-materiaali, graafisen käyttöliittymän ohjelmointi, suunnittelumallit, Tampereen teknillinen korkeakoulu.

Wikipedia 2012a, Active record pattern. [viitattu 15.2.2012]. Saatavissa: http://en.wikipedia.org/wiki/Active_record_pattern

Wikipedia 2012b, Ajax [viitattu 9.3.2012]. Saatavissa: http://fi.wikipedia.org/wiki/Ajax_(ohjelmointi)

Wikipedia 2012c, Data mapping [viitattu 15.2.2012]. Saatavissa: http://en.wikipedia.org/wiki/Data_mapping

Wikipedia 2012d, Cross site scripting. [viitattu 13.3.2012]. Saatavissa: http://fi.wikipedia.org/wiki/Cross_site_scripting

Wikipedia 2012e, HTML [viitattu 9.3.2012]. Saatavissa: http://fi.wikipedia.org/wiki/HTML

Wikipedia 2012f, Javascript [viitattu 9.3.2012]. Saatavissa: http://fi.wikipedia.org/wiki/JavaScript

Wikipedia 2012g, MVC-arkkitehtuuri [viitattu 28.2.2012]. Saatavissa: http://fi.wikipedia.org/wiki/MVC-arkkitehtuuri

Wikipedia 2012h, XML [viitattu 9.3.2012]. Saatavissa: http://fi.wikipedia.org/wiki/XML