KARELIA-AMMATTIKORKEAKOULU
Tietojenkäsittelyn koulutusohjelma

Jussi Alanen

WWW-SOVELLUSKEHYS – YII FRAMEWORKIN SOVELTAMINEN

Opinnäytetyö
Lokakuu 2013

OPINNÄYTETYÖ
Lokakuu 2013
Tietojenkäsittelyn koulutusohjelma

Liiketalouden ja tekniikan keskus
Karjalankatu 3
FI-80220 JOENSUU
p. (013) 260 6800

Tekijä
Alanen Jussi

Nimike
WWW-sovelluskehys – Yii Frameworkin soveltaminen

Toimeksiantaja
Karelia AMK, tietojenkäsittelyn koulutusohjelma

Tiivistelmä

Tässä opinnäytetyössä tarkastellaan WWW-sovelluskehyksiä yleisellä tasolla sekä tutkitaan Yii Frameworkia ja sen soveltamista. Työssä rakennetaan palvelimelle Yii-sovelluskehyksellä toimiva protosovellus sosiaalisen median palvelusta ammattikorkeakoulujen opiskelijoita sekä opettajia varten. WWW-sovelluskehys on verkossa oleva ohjelmistotuote, joka muodostaa valmiin rungon tulevaa web-sovellusta varten. WWW-sovelluskehys toimii ohjelmoijan apuvälineenä, ja se nopeuttaa uusien web-sovellusten valmistusta.

Työssä käydään läpi WWW-sovelluskehysten etuja ja haasteita, sovelluskehyksen rakennetta ja sen hallintaa sekä sitä, miten sovelluskehys käytännössä toimii verkossa. Työssä tarkastellaan myös tietokantojen ja tietojen automatisoinnin toimintaa eri sovelluskehyksissä. Lisäksi työssä on mitattu sovelluskehysten suorituskykyä esimerkiksi tiedon siirtonopeuden ja vasteaikojen perusteella. Opinnäytetyössä tarkastellaan lähemmin Yii Framework -sovelluskehystä: sen historiaa, suorituskykyä sekä loogista ja fyysistä rakennetta.

Opinnäytetyön lopussa syntyi toimiva Yii Frameworkilla toteutettu AMKLog-niminen sosiaalisen median palvelun protosovellus. AMKLogia on tarkoitus käyttää vain ammattikorkeakoulun opiskelijoita ja opettajia varten. Työ ei ole julkaistu julkisessa verkossa vaan sitä on tarkoitus kehittää lisää tulevaisuudessa.

Kieli
suomi

Sivuja 45
Liitteet 11
Liitesivumäärä 18

Asiasanat
verkkosovelluskehittäminen, sovelluskehys, ohjelmistokehys, PHP, web-ohjelmointi, tietokannat, automatisointi, arkkitehtuuri, palvelimet

THESIS
October 2013
Degree Programme in Business Information Technology

Centre for Business and Engineering
Karjalankatu 3
FIN 80220 JOENSUU
FINLAND
Tel. 358-13-260 6800

Author
Alanen Jussi

Title
Web application framework – Development using Yii Framework

Commissioned by
Karelia University of Applied Sciences, Degree Programme in Business Information Technology

Abstract

The purpose of this thesis is to examine Web application frameworks in general as well as explore Yii Framework and its implementation. In this thesis an application working with Yii Framework is built from a social media service for the students and teachers of University of Applied Sciences. A Web application framework is an online software product that forms a finished hull for an upcoming web application. A Web application framework is a programmer's tool, and it will speed up the production of new web applications.

The thesis presents the advantages and challenges of a web application framework, the structure and management of the application framework, and how the application framework practically works in online network. The thesis also examines the databases and data automation activities in various application frameworks. In addition, the performance of application frameworks such as data transfer and response times have been measured in this thesis. This thesis considers the Yii Framework application framework: history, performance and the logical and physical structure.

As a result of this thesis a functionally working social media service proto-application, implemented with Yii Framework, called AMKLog was created. The AMKLog application is to be used only for the students and teachers of University of Applied Sciences. The application has not been published on the public network, but it will be developed further in the future.

Language
Finnish

Pages 45
Appendices 11
Pages of Appendices 18

Keywords
web application development, software framework, application framework, PHP, web programming, databases, automation, architectures, servers

Sisältö

Liitteet

Käsite- ja lyhenneluettelo

CRUD Tulee englanninkielisestä sanoista ”Create, Read, Update, Delete”. Mahdollistaa automatisoitujen olioiden tietojen luomisen, palauttamisen, päivittämisen ja poistamisen.

DRY Tulee englanninkielisestä sanoista ”Dont Repeat Yourself” eli suomeksi ”älä toista itseäsi.” Tällä periaatteella pyritään vähentämään tietojen toistamista ohjelmistojen suunnittelussa. (Wikipedia 2013a, DRY.)

MVC Model-View-Controller (suomeksi malli, näkymä, ohjain). Tavoitteena on eriyttää liiketoiminnan logiikkaa käyttöliittymästä. Mallilla kuvataan tiedon tallentamista, ylläpitoa ja käsittelyä. (Wikipedia 2013b, MVC.)

ORM Object-Relational Mapping, ohjelmointitekniikka, joka muuntaa dataa yhteensopimattomien järjestelmien välillä olio-ohjelmoinissa. (Wikipedia 2013c, ORM.)

PHP Hypertext Preprocessor, käytetään web-palvelinympäristöissä dynaamisten verkkosivujen luonnissa. PHP on komentosarjakieli, jossa koodi tulkitaan vasta selaimen suoritusvaiheessa. (Wikipedia 2013d, PHP.)

SQL Structured Query Language, kyselykieli, jonka avulla voidaan tehdä erilaisia kyselyitä tietokantaan. Kyselyillä tietokantaan pystyy hakemaan, muokkaamaan, lisämään tai poistamaan tietoa. (Wikipedia 2013e, SQL.)

SQLite SQLite on relaatiotietokantajärjestelmä, joka on toteutettu pienenä C-kirjastona. SQLite-järjestelmä linkitetään johonkin käytettävään sovellukseen ja erillistä tietokannanhallintaohjelmaa tai tietokantapalvelinta ei tarvita. Tietokantaa voi pitää kokonaan tietokoneen muistissa tai tallentaa yhteen tiedostoon, joka lukitaan transaktioiden ajaksi. (Wikipedia 2013f, SQLite.)

WWW World Wide Web, Internet-verkossa hajautettu hypertekstijärjestelmä. (Wikipedia 2013g, WWW.)

1 Johdanto

Opinnäytetyön aiheena on sosiaalisen median palvelun toteuttaminen Karelia-ammattikorkeakoulua varten. Tavoitteena on luoda palvelu, joka toimii vain koulun verkossa. Työssä rakennetaan protosovellus palvelusta Yii Framework -sovelluskehystä käyttäen. Aihe on syntynyt keskusteluissa opettajan kanssa. Tälläiselle palvelulle on tarvetta, koska ammattikorkeakoulun verkossa ei tällä hetkellä ole kunnollista sosiaalisen median palvelua opiskelijoille ja opettajille.

Ammattikorkeakoulun verkossa toimii nykyään Kyvyt.fi-palvelu. Se on suurimmaksi osaksi blogikirjoituksia varten ja kurssien ylläpitoon, mutta opinnäytetyössä kehitetty palvelu sisältää enemmän sisältöä, toiminallisuuksia ja ominaisuuksia verrattuna Kyvyt.fi:hin.

Työssä käsitellään yleisesti sovelluskehyksiä ja tutkitaan WWW-sovelluskehystä ja sovelluskehyksen toiminnallisuutta. Lisäksi tarkastellaan lähemmin Yii Frameworkin sovelluskehystä ja sen rakennetta ja verrataan sitä muihin WWW-sovelluskehyksiin. Lisäksi tutkitaan sovelluksen rakenteen hallintaa ja tietokantaoperaatioiden automatisointia.

Sovelluskehyksen komponentteja on helppo käyttää ja hyödyntää erilaisissa asiakasprojekteissa ja niillä pystyy esimerkiksi rakentamaan helposti olio-orientoituneen verkkosovelluksen. Komponenttien avulla pystytään nopeuttamaan koodin kirjoittamista ja uudelleenkäytettävyyttä.

2 Yleistä WWW-sovelluskehyksistä

WWW on kehittynyt nopeasti omaksi tietoverkokseen. Alussa WWW-sivut olivat staattisia, mutta 1990-luvun lopussa niistä muodostui aktiivisemman näköisiä elementtien sekä palvelinpuolen ansiosta. Tällöin osattiin myös varastoida dataa tietokantaan. Tietokannan dataa pystyttiin lisäksi hakemaan, päivittämään ja muokkaamaan. Sovellukset oli pääsääntöisesti pystytty toteuttamaan ilman ohjelmistotuotannossa käytettyjä prosesseja. Ensimmäisiä oliomalleja kuten WOOMia (Web Object Oriented Model) pystyttiin hyödyntämään rakentamalla sisältöä ja navigaatiota. Mallin ansiosta työkalu sisälsi myös oman WOOM-luokkakirjaston, jossa oli WWW-sivujen kehittämiseen dokumentoituja kirjastoja. (Gellersen & Gaedke 1999.)

WWW-sovelluskehys tarkoittaa ohjelmistotuotetta, jonka avulla pystytään muodostamaan runko tulevalle verkkosovellukselle. Sovelluskehys toimii ohjelmoinnin apuvälineenä ja sen tarkoituksena on tehdä käyttäjäystävällinen, turvallinen, ammattimainen ja nopea ohjelmistotuote. Sovelluskehykset tarjoavat valmiiksi erilaisia komponentteja, joita ei tarvitse itse kirjoittaa käsin koodaten uudelleen – tämä nopeuttaa erityisesti ohjelman kirjoittamisen kehitystyötä. Komponenttien lisäksi WWW-sovelluskehykset rakentuvat luokista ja rajapinnoista, joista syntyy sovelluksen runko. Runko rakennetaan itse kirjoitetulla ohjelmakoodilla, joka liitetään sovelluskehykseen erityisten rajapintojen kautta. Rajapintojen yleisenä tekniikkana käytetään periytymistä, jossa ohjelmoijan luoma luokka perii sovelluskehyksestä olevan kantaluokan. Yiissa kuten muissakin sovelluskehyksissä pystytään perimään sovelluskehyksessä olevia luokkia, esimerkiksi ohjain-luokkia. (TTY Ohjelmistotekniikka 2009.)

Lopputuote verkkosovelluksesta saadaan rakentamalla vasta, kun uusi verkkosivusto on rakennettu kehyksen päälle. Sovelluskehykset on yleensä rakennettu oliopohjaisesti, joten ne tunnetaan olio-orientoituneina sovelluksina. Tunnettuja WWW-sovelluskehyksiä ovat mm. Yii Framework, Zend Framework, Symfony, Prado, CakePHP ja Codelgniter. Näistä suosituin WWW-sovelluskehys on Yii Framework. (Best Web-Frameworks 2013.)

2.1 Etuja ja haasteita

Sovelluskehyksien etuja ja haasteita on tutkittu vuosien varrella (Roihuvaara 2013). Etuja ovat jo edellä mainitut koodin uudelleenkäytettävyys ja ylläpidettävyys. Tutkija Mattsson on selvittänyt, että sovelluskehyksillä pystytään vähentämään runsasta työmäärää. Se tarkoittaa sitä, että koodin kirjoittamista pystytään huomattavasti nopeuttamaan, jos käytetään jotakin WWW-sovelluskehystä verkkosovellusta varten. (Mattsson 2000.) Sovelluskehyksen päivittämisen ansiosta pystytään käyttämään erilaisia ohjelmakirjastoja, joilla pyritään vähentämään sovelluksen ohjelmoinnin työmäärää.

Jos sovelluskehys ei noudata jonkin sovellusalueen vaatimuksia tai sovelluskehys ei sovellu tiettyyn ongelmanratkaisuun, on järkevämpää toteuttaa verkkosovellus jollain muulla sovelluskehyksellä tai olla käyttämättä jotain valmista sovelluskehystä. Sovelluskehyksen avoimeissa lähdekoodissa voi olla joskus puutteita. Ongelmia voi ilmetä esimerkiksi sovellusalueella tai arkkitehtuurin vaadittujen ominaisuuksien kanssa. Näissä tapauksissa ratkaisu on yleensä ollut tiedon muokkaaminen. Ongelmia voi seurata myös tiedon muokkauksen jälkeen, koska se saattaa rikkoa yhteensopivuutta uusimpien sovelluskehysversioiden kanssa. Mikä tahansa ohjelmisto muuttuu muokatessa koodia ja kehittyy muuttuvien vaatimusten mukana. Se aiheuttaa haasteita sovelluskehystä käyttävien verkkosovelluksien ylläpidettävyydelle. Kaikki muuttuvat sovelluskehyksen päivittämiset voivat aiheuttaa syvällisiä muutoksia käytettävässä verkkosovelluksessa. (Roihuvaara 2013.)

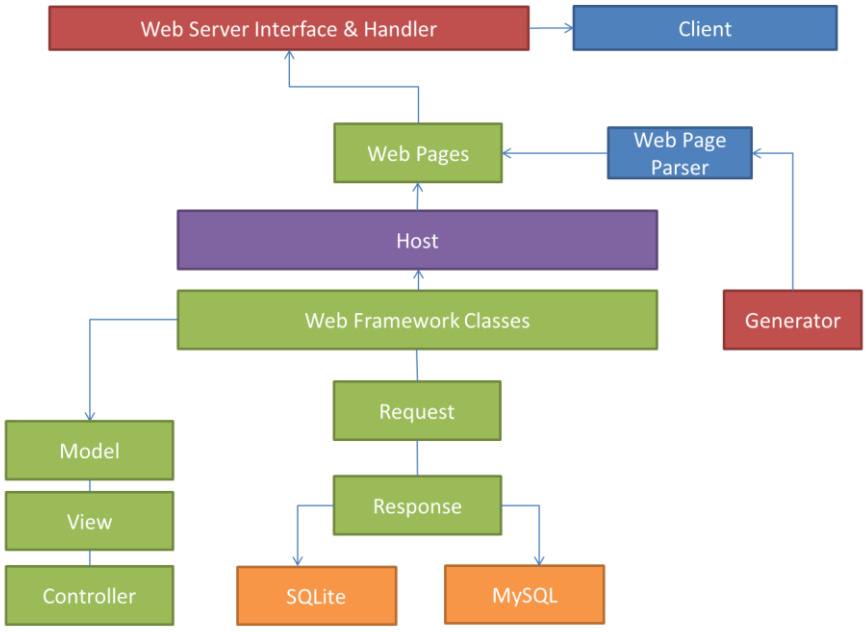
Kaikkiin sovelluskehyksiin tai tarvittaviin sovellusalueen tehtäviin on tarjolla erilaisia työkaluja ongelmien ratkaisuun. Sisäisen kehyksen logiikassa pystytään ongelmanratkaisuja hoitamaan oikeaoppisesti käyttämällä kehyksen julkisia rajapintoja ja luokkia. Ylimääräisen muokkaamisen välttäminen vähentää kriittisiä ongelmia verkkosovelluksen ja sovelluskehyksen välillä. (Roihuvaara 2013.)

2.1.1 Sovelluksen rakenteen pystyttäminen

Sovelluksen rakenteen pystyttäminen ei tarvitse suuria toimenpiteitä. Sovelluskehittäjällä on yleensä käytössä työkaluja, joilla pystyy valmiiksi generoimaan web-sivupohjan sovelluskehyksen esimerkiksi komentokehotteen kautta. Rakenteen pystyttäminen yleensä alkaa siitä, että sovelluskehyksellä on oltava selvä arkkitehtuurimalli, eli malli siitä, minkälaista runkoa sovelluskehyksessä käytetään. Sovelluskehyksen rakentaminen alkaa päähakemiston rakentamisesta. Päähakemistoon laitetaan käytettäviä alihakemistoja tulevaa sovellusta varten. Päähakemisto on tarkoitettu sovelluskehyksen luokkien lähdetiedostoille. Sovelluskehyksen rakentamiselle täytyy olla asetustiedosto, jolla on kyky ottaa sovelluskehyksen avulla tietoja sovelluskehyksen luokista ja tiedostoista tietokantayhteyden hyödyntämiseen ja tiedon varastointiin. Sovelluskehyksen ytimessä täytyy olla looginen rakenne ohjauksille, malleille ja näkymille, jotta näitä kaikkia pystyttäisiin hyödyntämään kun ajetaan sivustoa jollakin nettiselaimella. Sovelluksen pystyttämiselle on oltava eräänlainen hakemisto myös tiedon varastointia varten, jotta osa datasta menee välimuistiin (Cache), joka yrittää mahdollisesti sitten nopeuttaa sivuston pyyntöjä ja latausaikoja. Turvallisuuden kannalta sivusto ja sovelluskehys kannattaa suojata siten, että ne ovat salattujen hakemistojen takana, joille ei ole muilla pääsyä. Hakemistot yleensä yliajetaan palvelimien asetustiedostolla, johon laitetaan käyttöoikeudet hakemistoihin. Sovelluskehyksen tarvittavat luokat, rajapinnat ja metodit kannattaa laittaa päähakemistoon. Esimerkiksi action-metodeja käytetään sivuston avaamiseen ja jä näkymä-tiedostojen näyttämiseen.

2.1.2 Rakenteen hallinta

Sovelluskehyksessä käytetään sivuston erilaisia näkymiä, ohjaimia, malleja, moduuleja ja liitännäisiä. Niillä pystytään hallinnoimaan jo laajemmin sovelluskehystä. Hallitseminen tarvitsee syvällisempää tuntemusta jostakin sovelluskehyksen arkkitehtuurimallinnuksesta tai sovelluskehyksien toiminnasta kuin myös niiden käytettävistä ominaisuuksista, jotta rakenteen muokkaaminen ja luominen olisi sovelluskehittäjälle helpompaa. Sovelluskehyksen rakenne pystytään tekemään sovelluksen ja tietokannan välisten funktioiden ja luokkien avulla ja tämä tarkoittaa sitä, että näitä samoja funktioita ja luokkia pystytään käyttämään tulevissa projekteissa. Sovelluskehyksen hallintaan voidaan lisätä rajapintoja laajentamaan sovellusta monipuolisemmaksi, jotta koodin kirjoittaminen olisi sovelluskehittäjälle helpompaa. Sovelluskehyksen rakenteen suunnittelussa ja toteutuksessa on oleellista tietokantaoperaatioiden automatisointi luokkien välillä, jota käytetään lähes kaikissa WWW-sovelluskehyksissä. Kuviossa 1 esitellään WWW-sovelluskehyksen rakennekaaviota.



Kuvio 1. WWW-sovelluskehyksen perinteinen rakennekaavio (Appweb Embedded Web Server 2013a).

2.2 Sovelluksen tietojen käsittelyn automatisointi

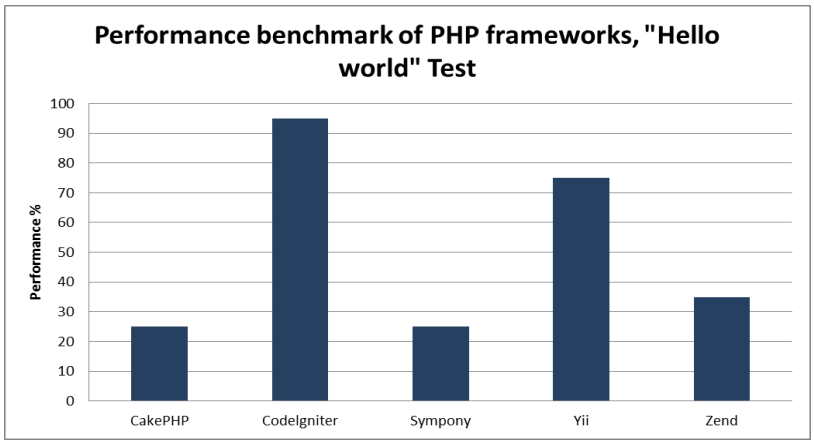
Sovelluskehys mahdollistaa tietokantalausekkeiden tekemisen selkeiksi ja erittäin helppokäyttöisiksi tulevissa verkkosivuprojekteissa. Sovelluskehys noudattaa ORM:ää ja käyttää automatisoitua CRUD-menetelmää, jossa pyritään luomaan olioista automatisoituja funktiota, ja luo tietoja tietokantaan helposti. Tietokannan automatisointi auttaa etenkin siinä, että käyttäjän ei tarvitse kirjoittaa pitkiä tietokantakyselyitä vaan kaikki toimii valmiiden funktioiden kautta. Sovelluskehyksellä on yli kymmeniä tietokantaan liittyviä funktioita, joilla pyritään keräämään ja tallentamaan tietoja turvallisemmin erillisten parametrien avulla sekä välttymään isoilta SQL-injektioilta. Sovelluskehittäjän ei tarvitse aina rakentaa itse olioiden tietojen tallentamismekanismeja vaan sovelluskehittäjä voi generoida tai kirjoittaa lyhyitä funktioita.

2.3 Sovelluskehyksien suorituskyvyn mittaaminen

Sovelluskehyksien suorituskyvyn mittaaminen tapahtuu palvelimen omalla mittaustyökalulla. Tämänhetkiset parhaat viisi www-sovelluskehystä ovat Codelgniter, Yii, CakePHP, Zend ja Symfony. (PHPFrameworks.com 2013.) Kun mitataan sovelluskehyksen suorituskykyä, ensiksi täytyy asentaa jokin olemassa oleva sovelluskehys esimerkiksi kotisivuhakemistolle, jossa ei ole käytössä muita sovelluskehyksiä. Kannattaa tarkistaa sitä ennen, millaisia ohjaimia, malleja ja näkymiä sovelluskehykset käyttävät sekä tarvitseeko sovelluskehykselle tehdä asetuksia ennen sen käyttöä selaimessa. Testisovellukselle kirjoitetaan jokin esimerkkiteksti, jota sovelluskehys hyödyntää ohjelmaa ajaessaan. Testaamiseen voi liittää myös tietokantoja tai palvelimen lisäominaisuuksia päälle. Sovelluskehyksen versiot, palvelimen versio sekä PHP-tulkin versiot vaikuttavat myös suorituskyvyn mittaamiseen. Palvelimen komponenttien on syytä olla nopea, ehjä ja toimiva, jotta suorituskyvyn mittaaminen onnistuisi ilman ongelmia.

Netistä pystytään lataamaan Apache-palvelimelle mittaustyökalusovellus, jolla mitataan sivuilta tulevia sovelluksen pyyntöjä ja viivenopeutta nettisivustoilta. Sovelluksen saa ladattua sivulta: http://httpd.apache.org/docs/2.2/programs/ab.html. Mittauksessa erityisesti otetaan huomioon se, minkälaista dataa sivusto sisältää ja kuinka paljon sovelluskehys käyttää välimuistia. Mittauksen aikana ei kannata olla varmuuden vuoksi mitään latauksia päällä, jos tuloksia mitataan kotipalvelimelta tai ulkoiselta palvelimelta. Sivuston suorituskyvyn mittaamisessa käytetään tekstiä, kuvia, videoita tai muuta materiaalia.

Sovelluskehyksissä mitattiin vasteaikaa, siirtonopeutta ja pyyntöjä Kapsi.fi:n palvelimelta. Palvelimessa täytyy olla asennettuna SSH-tunneli ja sille käyttäjätunnus sekä salasana, jotta päästään palvelimeen käsiksi ja kirjoittamaan komento, jolla pystytään mittaamaan eri sovelluskehyksien nopeuksia ns. pyyntöjä sekunnissa. Esimerkiksi komennolla ”ab –n 1000 –c 10 http://esimerkki.com/yiitest/index.php?r=say/hello” pystytään testamaan Yii-sovelluskehyksellä luotua esimerkkisivustoa, joka on valmiiksi generoitu. Ab tarkoittaa ApacheBench-sovellusta Apache-palvelimessa, -n-parametri tarkoittaa, kuinka monta pyyntöä halutaan sivuston hakevan. Tässä tapauksessa tehdään 1000 pyyntöä. Viimeinen mainittu parametri ”-c” tarkoittaa, että otetaan sovelluskehyksen sivuston evästeet mukaan. (liite 1.)



Kuvio 2. Vertailussa viisi parasta valittua sovelluskehystä tähän mennessä. Sovelluskehyksien tarkastelussa otettiin huomioon sovelluskehyksen viiveen nopeus ja tulevat pyynnöt sivustolta.

2.4 Yii Framework

Yii Framework on yksi suosituimmista WWW-sovelluskehyksistä. Sana ”Yii” lausutaan ”Jee”. Ilmaus on lyhenne sanoista ”Yes it is”, ”Kyllä se on”. Yiillä tarkoitetaan usein tarkkaa ja ytimekästä. Yiille tulee yleensä myös seuraavanlaisia kysymyksiä: ”Onko se nopea? Onko se turvallinen? Onko se ammattimaista? Voiko sitä käyttää seuraavissa projekteissa?”. Yllä oleviin kysymyksiin on vastaukseksi: ”Kyllä se on!”. (Yii Framework 2013a.)

Yiin perustaja on Qianq Xue, joka aloitti ensimmäisen Yii-hankkeen 1. tammikuuta 2008. Vuoden kestäneen kehitystyön tuloksena syntyi erittäin nopea, turvallinen ja ammattimainen sovelluskehys, joka on räätälöity vastaamaan Web 2.0 -sovelluksia. Ensimmäinen kehys saatiin valmiiksi 3. joulukuuta 2008. (Yii Framework 2013a.)

Julkiseen testaukseen on tullut Yii Frameworkin versio Yii 2.0. Yii Framework 2.0 -versiota kehitetään koko ajan aktiivisesti. Se on rakennettu version PHP 5.3.0+ päälle. Uudesta versiosta yritetään saada myös helppokäyttöinen ja siihen yritetään tuottaa uusia ominaisuuksia. (Yii Framework 2013a.)

2.4.1 Yleistä

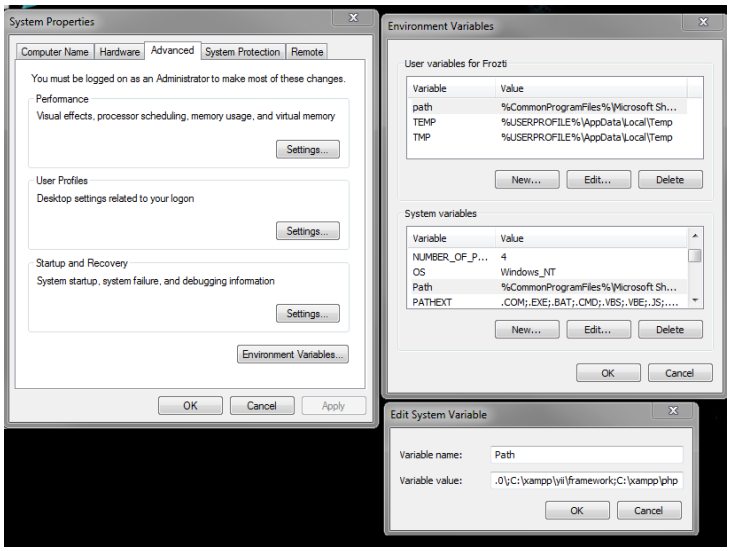
Yii tunnetaan ilmaisena avoimeen lähdekoodiin perustuvana web-sovellusten kehitysalustana, joka on kirjoitettu PHP5-versiolle. Edellytyksenä käytetään puhdasta koodikieltä ja DRY-suunnittelua ja kannustetaan tekemään nopeampaa verkkosovelluskehittämistä. Yii Frameworkin ansiosta verkkosivuista pystytään tekemään nopeampia, ammattimaisia, turvallisia ja monipuolisia. Yii on parhaimmillaan tehokas sovelluskehys ja tukee erityisesti Web 2.0 -sovelluksia. Yiillä toimii tehokas välimuistin tuki ja se on rakennettu toimimaan Ajaxin kanssa. Runsailla ominaisuuksilla saadaan aikaan helppokäyttöisiä verkkosovelluksia, jotka toimivat samalla periaatteella kuin muutkin sovelluskehykset. Yiin tärkeimmät ominaisuudet ovat MVC (Model-View-Controller), DAO (Data Access Object), ActiveRecord, Gii-työkalu, CRUD-menetelmä sekä roolipohjaisen sivuston luonti. Turvallisuudessa Yiin vakiovarusteena on toiminut sisääntulo-validointi, joka osaa tarkastaa, ovatko esimerkiksi salasanat oikein tai vaaditut kentät täytetty oikein. Yii tukee myös monipuolista web-palvelujen (Web-services) hallintaa sekä eri verkkosivujen pyyntöjä (Requests). Sovelluskehyksessä toimii integroituja jQuery-toimintoja ja se sisältää myös Ajax-pohjaisia widgettejä. Näillä pystytään esim. rakentamaan tietokannasta tuotuja tietoja dataruudukkoon (Data Grid), puunäkymä (Treeview), automaattinen täydennys tekstikenttään jne. (Yii Framework 2013b.)

Tietokantalausekkeiden ansioista Yii Frameworkilla pystytään sisäänrakennetuilla funktioilla vähentämään SQL-kyselyjen kirjoittamista ja jäsentämään SQL-kyselyä selkeämmin. Yiin suuunnittelumallin ansiosta pystytään erottamaan koodin logiikka verrattuna muihin verkkosovelluskehyksiin sekä tekemään kirjoitetusta koodista parempaa.

2.4.2 Yii Frameworkin käyttöönotto

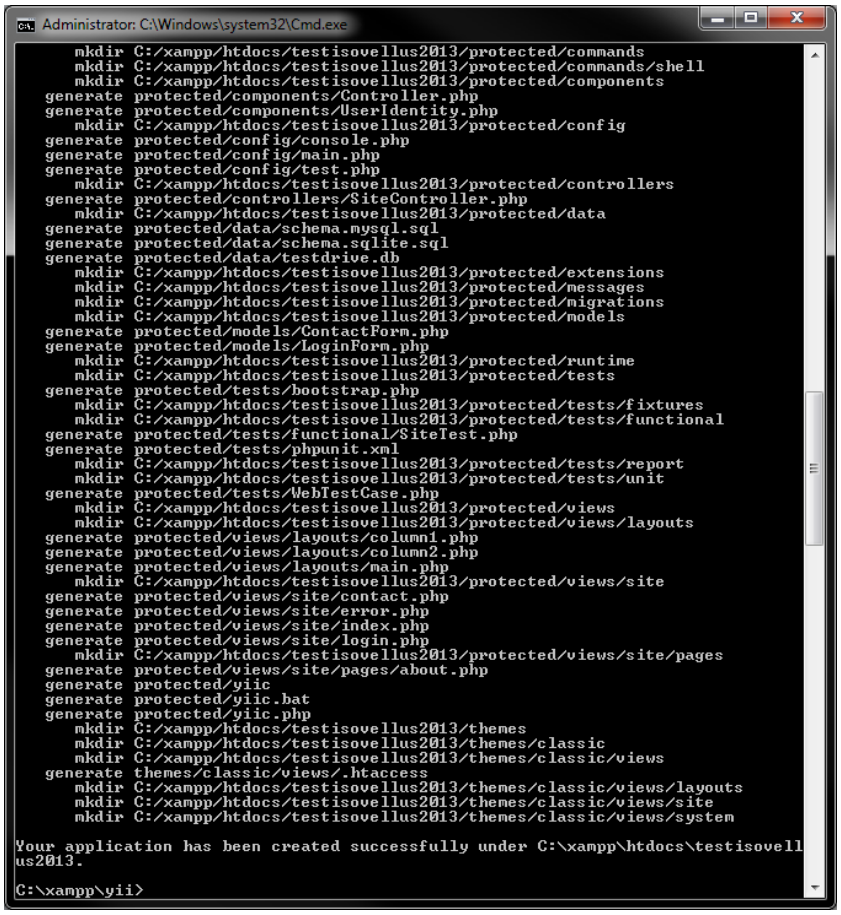
Yii-sovelluskehyksen viimeisimmän version voi ladata osoitteesta http://www.yiiframework.com/download/. Sovelluskehyksen voi ladata täysin ilmaiseksi eikä siihen tarvitse tehdä muita toiminpiteitä kuin käyttäjän rekistöröinti lataussivustolle. Paketti ladataan .tar.gz-muotona tai haluttaessa .zip-pakettina. Sivustolla on tällä hetkellä vielä ladattavissa versio 1.1.x, mutta vuoden 2013 aikana on tulossa versio 2.x.x. Puretun paketin sisältä löytyy *demos*-, *framework*,- ja *requirements*-hakemistot. Paketin mukana tulee myös versiomuutokset ja asennusohjeet.

Sovelluskehys asennetaan paikkaan, jossa on toimiva Apache HTTP-palvelin ja PHP-tulkki ja vapaavalintaisena myös MySQL-tietokanta. Jos pystyttämisessä käytetään paikallista palvelinta kuten XAMPP:ia, jonka saa osoitteesta: http://www.apachefriends.org/en/xampp-windows.html on asetustiedostot helppo konfiguroida ja asettaa. Purettu asennuskansio voidaan kopioida Xamppin päähakemiston juureen, jonne luodaan sovelluskehykselle oma päähakemisto. Jotta sovelluskehys toimii, täytyy ympäristömuuttujille määritellä polku (PATH) Yiin sovelluskehykseen (kuva 1). Tällöin sovelluskehys osaa hakea kaikki Yiin lähdekirjastot ja asetustiedostot.

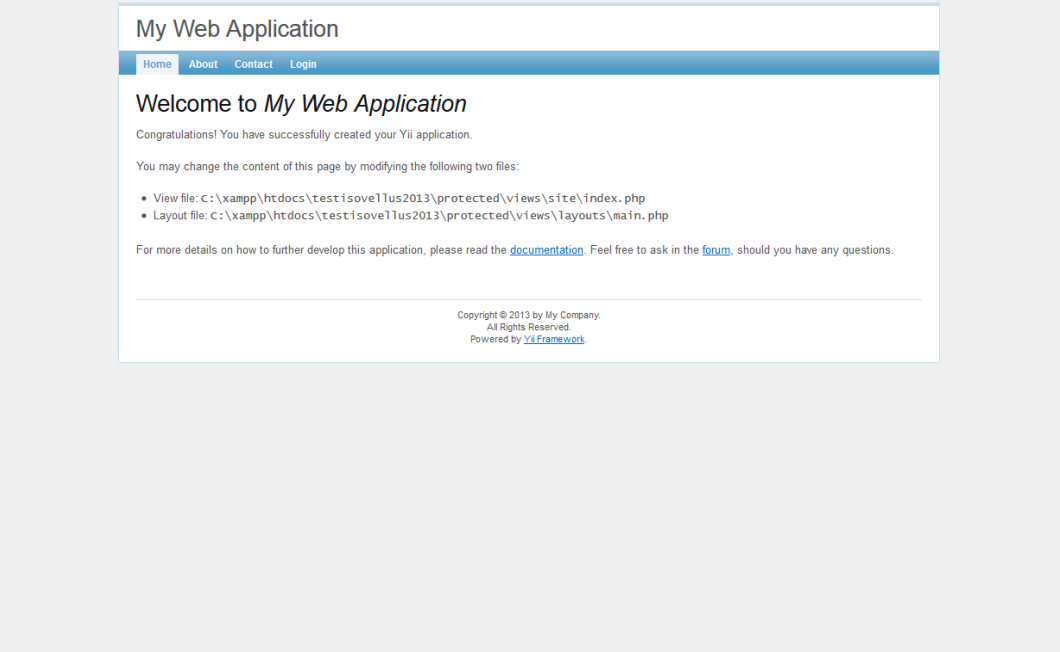


Kuva 1. Ympäristömuuttujalle täytyy antaa PATH-arvo Yii Frameworkin päähakemistossa, jotta Yii Framework toimisi kotisivupalvelimella.

Kun käyttäjä on kopioinut hakemiston oikeaan hakemistoon, tämän jälkeen pystytään komentorivikehotteesta luomaan Yiillä ensimmäinen testisovellus. Testiohjelmaa ei tarvitse itse kirjoittaa vaan kirjoitetaan komentoriviin komento, minkä jälkeen komentokehote ja Yii-lähdekirjasto generoivat kotisivuhakemistolle sivuston rakenteet, asetustiedostot jne. Komentoriville kirjoitetaan vain esimerkiksi *”yiic webapp C:\xampp\htdocs\testisovellus2013”*. Tällöin se luo testisovelluksen kotisivuhakemistolle, jossa kysytään sitä ennen, halutaanko luoda web-sovellus kotisivuhakemistolle vai ei. Kun valitaan kyllä, komentorivi suorittaa komennon, joka luo sovelluskehyksellä toimivan testisovelluksen kotisivuhakemistolle. Komentoa ajaessa komentokehotteessa nähdään loki, minkälaista tietoa luodaan Yii-sovellukselle (kuva 2). Kuvassa 3 nähdään, miltä generoitu sivusto näyttää selaimessa.



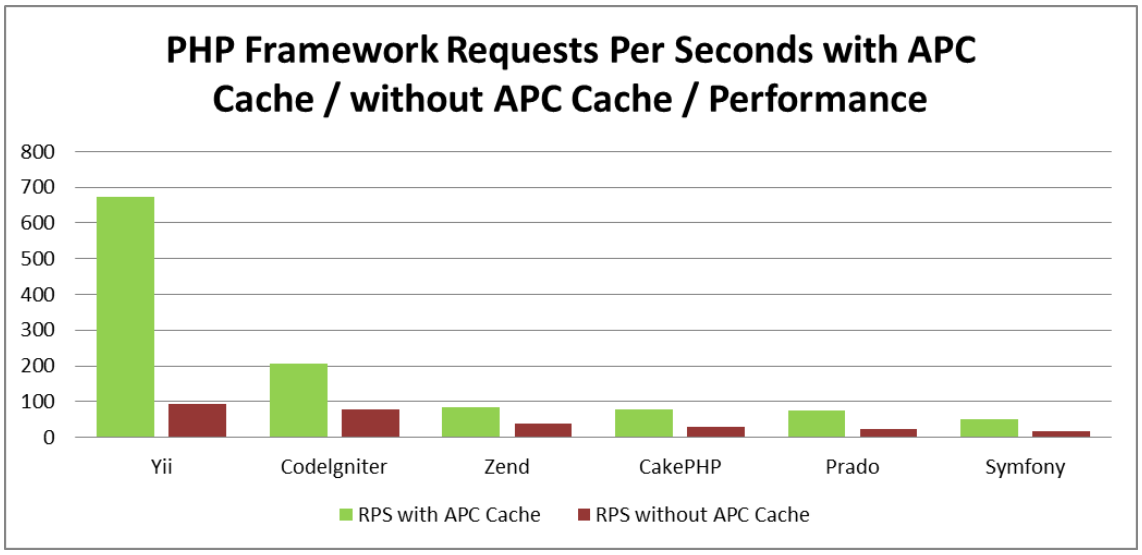
Kuva 2. Sovelluksen luonti komentokehotteessa.



Kuva 3. Komentorivin ajon jälkeen luotu testisovellus.

2.4.3 Suorituskyky

Yii on suorituskyvyltään kohtalaisen nopea. Kuviossa 2 esiteltiin yleinen katsaus erilaisten sovelluskehyksien nopeuksista, mutta kuviossa 3 mitataan ja tarkastellaan vielä Yiin nopeutta RPS:n avulla. RPS:lla (Request Per Second) tarkoitetaan suomeksi ”pyyntöjä sekunnissa”, joka kuvaa, kuinka monta eri prosessia voidaan käsitellä sekunnissa. Mitä suurempi luku on, sitä tehokkaammin kehys toimii. Yiin ja muiden sovelluskehysten suorituskykyn mittaaminen otetaan myös huomioon, jos APC (Alternative PHP Cache) laajennus on käytössä palvelimen asetuksissa. APC asettaa suorituskyvyn testauksessa palvelimen PHP:n välimuistit päälle. Suorituskyky ei kerro koko totuutta kehyksestä. Suorituskyvyn ajossa pitäisi käyttää lisäksi tavallista ja puhdasta HTML-tai PHP-koodia. Yii tarjoaa edelleen erittäin nopeat ja monipuoliset ominaisuudet, jotka merkittävästi parantavat verkkosovelluksen kehityksen tehokkuutta. (Yii Framework 2013c.)



Kuvio 3. Yiin vertailua erilaisiin sovelluskehyksiin. Yii voittaa nopean suorituskyvyn ansiosta. (Yii Framework 2013c.)

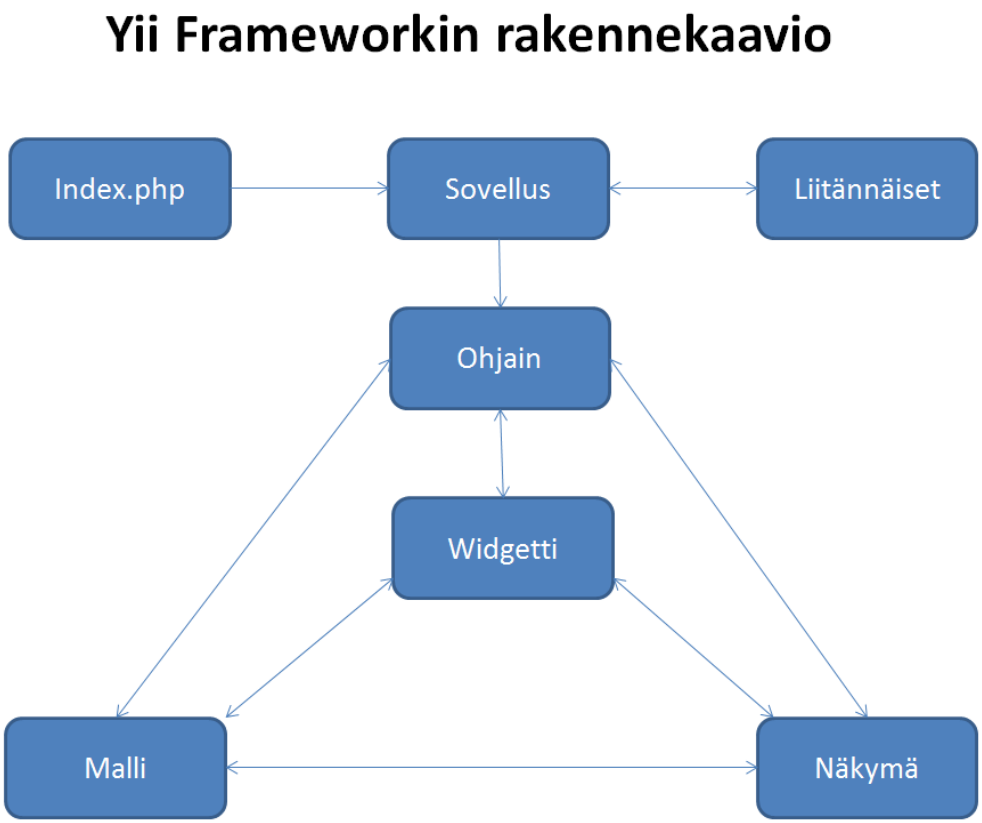
2.4.4 Keskeisimmät ominaisuudet

Luettelossa on Yii Framework -sovelluskehyksen keskeisimmät ominaisuudet:

- Yii käyttää, kuten monet muutkin sovelluskehykset, Model-View-Controller (MVC)-suunnittelumallia.
- Kansainvälistäminen ja lokalisointi (I18N ja L10N). Tukee viestien kääntämistä, päivämäärän ja kellonajan muotoilua, lukumuotoilua ja käyttöliittymän lokalisointia.
- Virheiden käsittely ja log-viestien käsittely.
- Turvatoimet ovat cross-site scriptingin (XSS) ehkäiseminen, cross-siten väärentäminen (CSRF) ennaltaehkäisy, evästejälkien ehkäisy jne.
- Yksikölliset ja toiminnalliset testaukset perustuvat PHPUnitiin ja Seleniumiin.
- Koodit pystytään generoimaan Yii-komponenteista ja komentorivityökaluista.
- Yii osaa hyödyntää XHTML-standardia.
- Zend Frameworkin tai Pearin olemassa olevaa koodia voidaan hyödyntää Yii-sovelluskehyksessä.

2.4.5 Looginen rakenne

Yii Framework toimii MVC-suunnittelumallilla kuten muutkin verkkosovelluskehykset. Tämä kyseinen Framework tarjoaa käytännöllisen ja toimivan MVC-arkkitehtuurimallin ja osaa myös hyödyntää erilaisia moduuleita ja komponentteja. Tässä tarkastellaan kuitenkin MVC-mallin ja muuta Yii-sovelluskehyksen toimintaa. Kuviossa 4 on esitetty Yii Framework -sovelluskehyksen rakennekaavio.



Kuvio 4**.** Yii Frameworkin perinteinen rakennekaavio, jossa näytetään, miten Yii toimii käytännössä (Media Infonet 2013).

Malli (M, model)

Malleilla pyritään hakemaan tietokannasta tietoa taulun avulla. Mallit luodaan luokaksi, joka toimii tietokannan omana tauluna. Mallin voi yleensä rakentaa itse tai generoida Gii-työkalulla. Malli-luokille rakenetaan apuattribuutit tietokantataulun tiedon hakua varten, jotta vältyttäisiin isoilta virheiltä ja ongelmantilanteilta. Jos malli on jo luotu valmiiksi, sitä ei tarvitse enää luoda vaan mallia pystytään hyödyntämään sen jälkeen suoritettavana luokkana esimerkiksi ohjaimessa. (Yii Framework 2013d.)

Näkymä (V, view)

Näkymällä pyritään tekemään käyttäjälle visuaalinen esitys mallista (Model). Näkymä ei tarkoita pelkkää rakennettua WWW-sivua vaan sillä voi myös tuottaa tiedostoja tai kuvia. Näkymän ulkoasu rakentuu HTML-sivuston elementeistä ja CSS-tyyleistä. Sivustolle voi tuoda myös käyttäjän tekemiä tai netistä ladattuja widgettejä, joilla pystytään rakentamaan sivusto nopeammin ja näyttäväksi. Malli (Model) tuodaan näkymälle (View), jotta tiedon haku olisi helpompaa ja vältytään ohjelmakoodin toistamiselta ohjaimessa. (Yii Framework 2013e.)

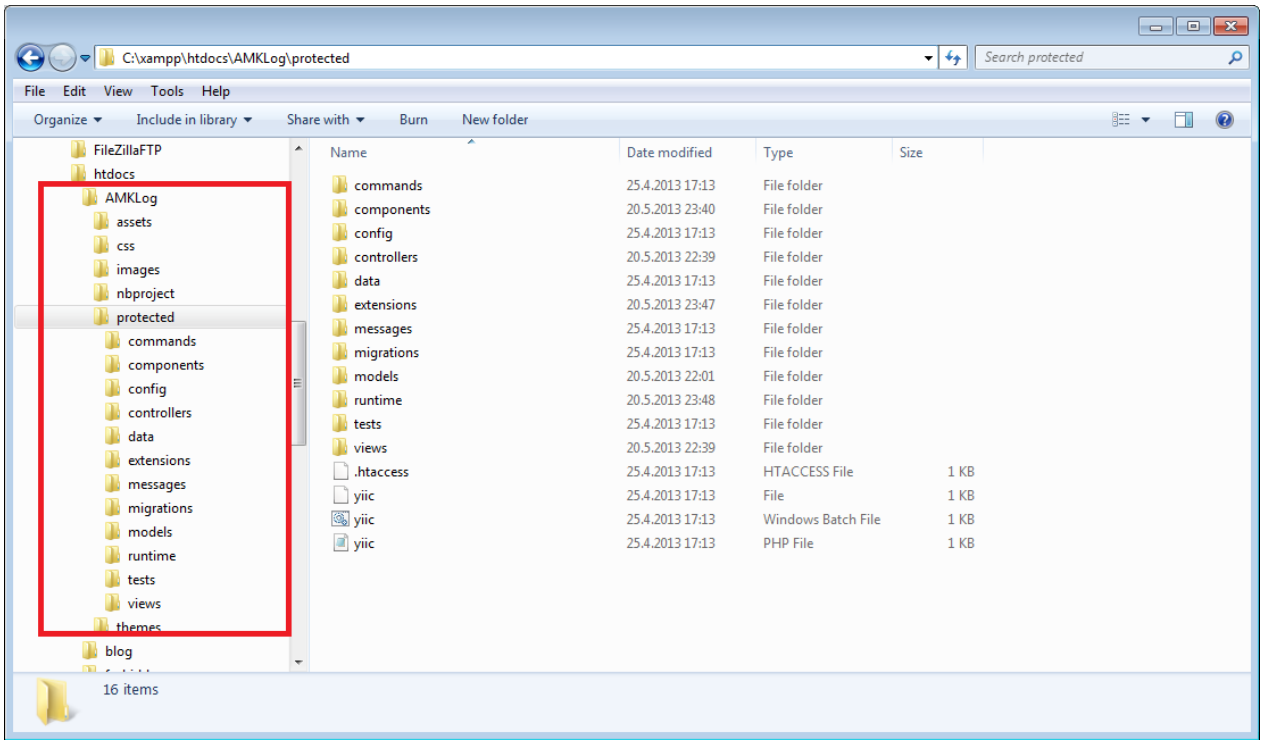
Ohjain (C, controller)

Yii Frameworkissa kuten muissakin sovelluskehyksissä käytetään kahdenlaista ohjainta. Käytössä on ensisijaisesti etuohjain *(engl. front controller)* ja toinen on toiminta-ohjain *(engl. action controller).* Etuohjaimen tarkoitus on huolehtia yksittäisistä toiminnoista ja toiminta-ohjaimella pyritään ohjaamaan erilaisiin nettisivuihin. Toiminta-ohjaimet pystytään itse luomaan tai generoimaan Gii-työkalulla. Toiminta-ohjauksissa perii yleensä pääohjainluokan nimeltä Controller-luokan Yiin kirjastosta. Tämän avulla toiminta-ohjaimessa koodin kirjoittaminen on helpompaa ja selkeyttää uudelleen käytettävää koodia, jolla saadaan helpommin suoritettua ja testattua luotuja nettisivustoja. (Yii Framework 2013f.)

2.4.6 Fyysinen rakenne ja sovelluskehyksen hakemistot

Sovellus jaetaan eri hakemistoihin, jotka muodostavat sovellukselle valmiin rungon. Kun generoidaan Yii Framework -generointiohjelmalla sivustoa, yleensä hakemistorakenne jakautuu viiteen eri hakemistoon, jotka ovat ”assets-”, ”css-”, ”images-”, ”protected-” ja ”themes”-hakemistot. Näistä sivuston ydinhakemisto on ”protected”. Projektin hakemistorakenne on esitetty kuvassa 4.

Hakemistorakenteen voi luoda käsinkin, mutta nopeimmin sen saa luomalla sivustorakenteen komentorivi-työkalun kautta. Tämän avulla pystytään välttämään ohjelmoijan tekemiä virheitä, esim. väärin kirjoitetut luokat.



Kuva 4. Yii Framework -sovelluksen päähakemiston rakenne.

Sovellus (Application)

Yii sisältää kaiken tarvittavan toiminnallisuuden protected-kansiosta. Yiin sovelluksen toiminnallisuus perustuu vain protected-kansion alla oleviin hakemistoihin. Päähakemisto ”protected” sisältää malleille, näkymille ja ohjaimille omat päähakemistot ja muut hakemistot ovat laajennuksia sovellukselle. Laajennuksille (Extensions) on oma hakemistonsa, johon käyttäjä voi tallentaa muiden käyttäjien tekemiä MVC-laajennuksia. Komponentit (Components) -kansioon pystytään lataamaan tai rakentamaan valmiita widgettejä sovellusta varten. Data-kansioon voidaan rakentaa itsenäinen tietokantatiedosto sovellusta varten, jos sovellus käyttää SQLiteä. Muutoin data-kansiota ei välttämättä tarvitse käyttää, jos tietokantana käytetään MySQL:ää. Runtime-kansiossa ajetaan sovelluksen Gii-työkalua, jossa käyttäjä voi luoda tietokantataulusta malleja, lomakkeita tai generoida CRUD-sivustoja. Config-hakemistossa konfiguroidaan verkkosivusto sopivaksi ja toimivaksi. Käyttäjä voi halutessaan myös kytkeä komponentteja ja moduuleita sovellukseen. (Yii Framework 2013g.)

Widgetit (Widgets)

Widgettejä käytetään esitysteknisiin tarkoituksiin. Widgetit ovat itsenäisiä ja pieniä MVC-sovelluksia. Widgetit voi luoda itse sovellukselle tai ladata Yii Frameworkin kotisivuilta. Widgetit luodaan omina luokkina ja tallennetaan komponentti-kansioon (Components). Widgetit pystytään näyttämään näkymän ohjelmakoodissa (Views). Widgettien käyttö helpottaa uudelleenkäytettävyyttä esim. käyttöliittymän rakentamisessa. Widget voi olla esimerkiksi kirjautumislomake tai uutisnauha etusivulla. (Yii Framework 2013d.)

Ulkoasu (Layout)

Ulkoasulla piirretään käyttöliittymää näkymää varten. Ulkoasu rakentuu yleensä perinteiseen tapaan HTML:n elementeistä ja CSS-tyylitiedoston sisällöstä. Yiissa voi tehdä omat teemat (Themes), joihin käyttäjä voi halutessaan luoda sivustolle oman ulkoasun. (Yii Framework 2013d.)

Moduulit (Modules)

Moduulit ovat itsenäisiä ohjelmia ja muistuttavat yhtenäistä sovellusta. Ne koostuvat malleista, näkymistä, ohjaimista ja muista vastaavista komponenteista. Moduulit ovat hyödyllisiä skenaarioita ja niistä pysytyään tekemään sovelluksista laajempia kokonaisuuksia. Moduuleja käytetään erityisesti esimerkiksi käyttäjähallinnan ja kommenttihallinnan luomisessa. (Yii Framework 2013i.)

Moduuleilla on sovelluksessa omat hakemistonsa, jotka toimivat siten, että niillä on yksilöllinen tunnus. Moduulin rakenne muistuttaa sovelluksen päähakemistoa. Jos sovelluksessa on käytössä esimerkiksi foorumi, niin sovellukselle luodaan Modules-hakemiston sisälle forum-hakemisto, johon sisällytetään kaikki foorumin ohjaimet (Controllers), mallit (Models), näkymät (Views), komponentit (Components) ja laajennukset (Extensions). Modules-hakemiston sisällä olevan forum-hakemiston sisällä täytyy olla vielä tiedosto, johon pystytään viittaamaan pääsovelluksessa. Config-tiedostossa täytyy muistaa viitata, millä tunnisteella moduuli halutaan tuntea pääsovelluksessa. (Yii Framework 2013i.)

Laajennukset (Extensions)

Laajennukset ovat hyödyllisiä sovelluskehittäjälle, joka haluaa valmiita lisätoimintoja sovellukseen. Laajennukset ovat itsenäisiä hakemistoja, jotka viitataan netistä ladattavaan Config-hakemistossa olevaan main.php-tiedostoon. Esimerkiksi kuvankäsittelylaajennus (EPhpThumb), jota ei tarvitse itse kirjoittaa, on helpoiten ladattavissa netistä. Kun tiedosto on ladattu, voidaan tiedostoon viitata main.php:ssä componets-taulussa. (Yii Framework 2013j.)

Auttajaluokat (Helpers)

Helpers on ohjelmoinnin tekniikka olio-ohjelmoinnissa. Helpers-luokkien etu on, että niiden avulla pystytään laajentamaan tai muokkaamaan olemassa olevia Yiin luokkia monipuolisemmaksi ja käyttämään toimintoja monessa eri luokassa. Kaikki apuluokat tehdään components-hakemistoon. Luokka täytyy määritellä itse, minkä jälkeen luokkaan sisään tehdään tarpeelliset toiminnot toisia luokkia varten. (Yii Framework 2013k.)

Ominaisuudet (Assets)

Assets-hakemistossa käsitellään muun muassa sovelluksen generoituja CSS-tiedostoja, Javascript-tiedostoja ja kuvatiedostoja. Luotuaan komponentin näkymälle sovellus tekee automaattisesti sille Assets-kansioon oman CSS-määrittelyn. Jos komponetti käyttää Javascript-koodia, se luo omanlaisensa Javaskript-koodin automaattisesti. Assets-hakemisto on ikään kuin sivuston välimuistitiedostoille tarkoitettu tallennushakemisto. (Yii Framework 2013l.)

Viestit (Messages)

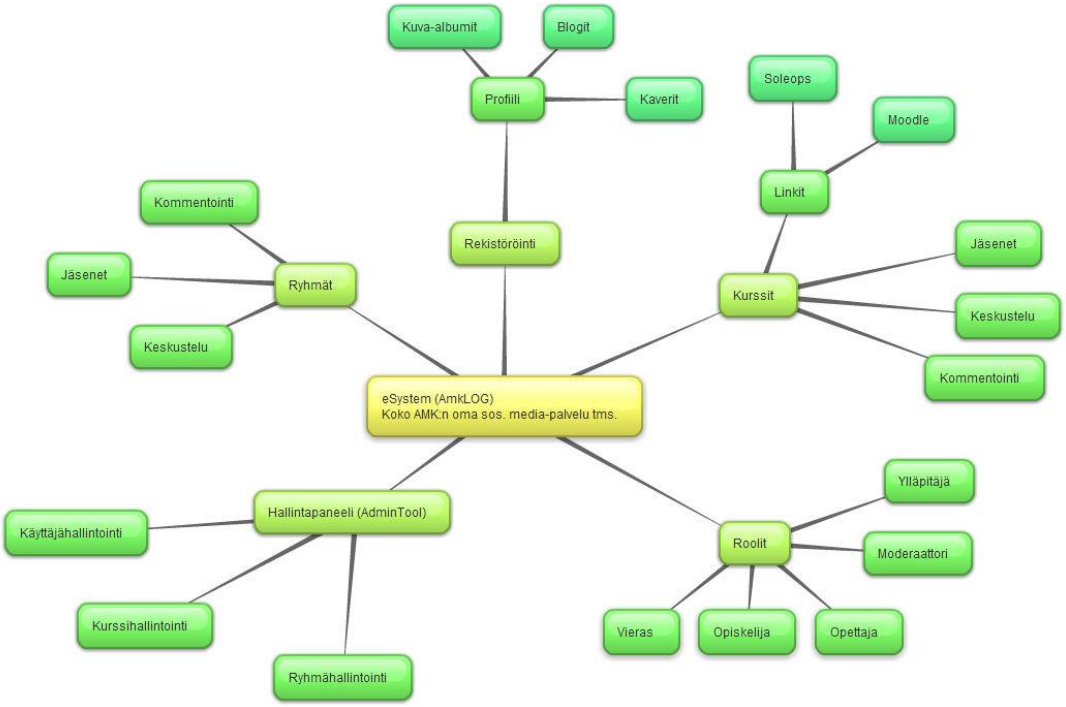
Messages on sovelluksen viestit-hakemisto, josta haetaan kaikille osioille tai teksteille käännökset. Viestit-hakemisto koskee lomakkeiden validointia ja virheilmoituksia. Viestit ovat sovellukselle ydin, josta se hakee yhdestä tiedostosta tietyille teksteille käännökset ja kääntää sitten mallissa tai näkymässä. Kaikissa kielissä toimii *coreMessages*-komponentti, joka etsii tietyt sanat, joihin viitataan malleissa tai näkymissä, ja kääntää kaikki viitatut sanat eri kielelle. (Yii Framework 2013m.)

Tiedot (Data)

Tietoja pystytään säilyttämään sovelluksen sisäpuolella Data-hakemistossa. Data-hakemistossa säilytetään muun muuassa paikallisia tietokantoja ja sovelluksesta tulevia tiedostoja, jos käyttäjä esimerkiksi haluaa ladata palvelimelle tiedostoja tai kuvia. Jos käytetään ulkopuolista tietokantaa, esimerkiksi MySQL:ää, ei data-hakemistoa silloin tarvita.

3 Sosiaalisen median palvelun rakentaminen Yii Frameworkilla

Sovelluksen rakentamisessa mietittiin ensin tarkkaan, mistä kannatti lähteä liikkeelle. Katsottiin myös, mitä kaikkea sovelluksella voisi tehdä, mitä vaatimuksia sovelluksella on ja mitä materiaaleja sovellus kaiken kaikkiaan tarvitsee. Sovelluksesta rakennettiin Facebook-tyylinen sosiaalisen median palvelusivusto, josta tehtiin ammattikorkeakoululle oma protoversio sekä testausta varten että jatkokehitystä ajatellen. Tämä rakennettu esimerkkisovellus on sosiaalisen median palvelu nimeltä AMKLog, ja se on suunnattu vain ammattikorkeakoulun opiskelijoille ja opettajille. Sana AMK tulee ammattikorkeakoulusta ja Log tulee sanasta Blog, koska on yhtenäistä pitää myös opiskelijoiden ja opettajien omia blogeja sivustolla ja jakaa niitä sosiaalisen median palvelussa muille opiskelijoille ja opettajille. Palveluun sisältyy perinteiseen tapaan käyttäjärekistöröinti, kirjautuminen, profiilisivuston ylläpitäminen, profiilikuvan lisääminen, kuva-albumien ylläpitäminen, blogien kirjoittaminen vapaa-ajasta tai opiskelusta, kurssikeskustelut ja ryhmäkeskustelut. Kuviossa 5 esitellään AMKLogin rakennekaaviota.



Kuvio 5. AMKLog-palvelun rakennekaavio.

3.1 Sovelluksen asetukset

Sovelluksen käyttämiä sovelluskehyksen asetuksia on helppo säätää sivustolle config-hakemistossa olevassa main.php-tiedostossa. Ensin pohdittiin, millaiset URL-polut sivusto tarvitsee turvallisuuden kannalta ja koodin kirjoittamisen nopeuden vuoksi sekä minkälaisia liitännäisiä sivustolle täytyy ladata, jotta sivustosta tulisi kattava. Huomioon otettiin se, haluaako käyttää paikallista tietokantayhteyttä vai ulkopuolista tietokantayhteyttä. Tärkeää on myös, että sivustolle määritellään heti ensimmäisenä otsikko ja toinen tieto kuten teeman lataaminen, joka tekee sivustosta näyttävämmän (kuva 5).

```
'basePath'=>dirname(__FILE__).DIRECTORY_SEPARATOR.'..',
'name'=>'AMKLog - Sosiaalinen media-palvelu ammattikorkeakoululle',
'theme' => 'amklogblack',
```

Kuva 5**.** Sovelluksen otsikointi ja teeman asettaminen. Teemalla täytyy olla oma hakemistonsa, johon viitataan sen jälkeen asetustiedostossa omalla nimellään.

URL-hallinnassa voidaan asettaa URL-osoitteille erilaisia sääntöjä (Rules), joilla pystytään saamaan sivujen URL-osoitteita käyttäjäystävälliseksi (liite 2, kuva a). Ensiksi annetaan parametreineen haluttu sivuston reitti (Route) ja sen jälkeen annetaan sivuston alkuperäinen URL-osoite parametreille. (Params). (Yii Framework 2013h.)

Sovellukseen on ladattu monipuolisia komponentteja, jotta sivusto toimisi paremmin ja kevyemmin. Komponenteilla pyritään myös vähentämään ja nopeuttamaan koodin kirjoittamista. Kuvan lataaminen tarvitsi esimerkiksi phpMyThumb-, CImageComponent- ja CFile-komponentit, jotta kuvista saadaan sopivan kokoisia ja jotta ne voidaan renderöidä siten, että ne vievät käyttäjäkohtaisilta hakemistoilta mahdollisimman vähän tilaa. CFile osaa tarkistaa, onko kyseinen kuva jo palvelimella ja tarvitseeko tiedostoa enää yliajaa. CImagecomponent osaa hoitaa kuvien rajaukset ja kuvien siirrot hakemistolle. (liite 2, kuva b.)

Asetuksissa pystyy myös viittaamaan julkisiin parametreihin, jos halutaan lisätä jokin komponentti usealle näkymälle, esimerkiksi lista käyttäjistä ja siihen sivunumerointi. Tässä tapauksessa täytyy vain viitata asetustiedostossa sivunumeroinnin nimeen ja hakutulosten määrään (liite 2, kuva c).

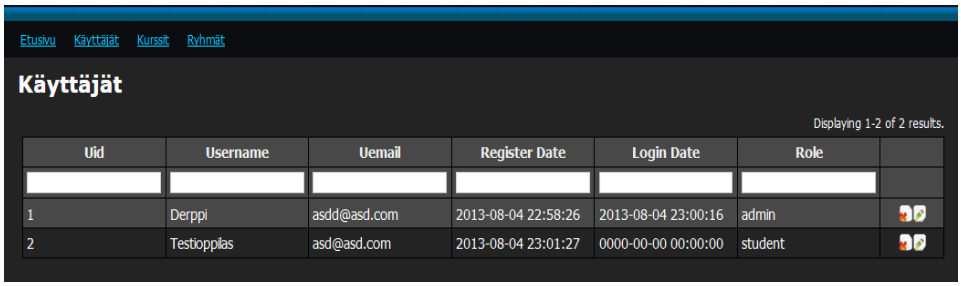
3.2 Sovelluksen tietokannan automatisointia

Tietokantalausekkeiden automatisointi ja yhteydenotto toimivat Yiillä monella tavalla. Ensin asetustiedostossa täytyy olla määritelty, käytetäänkö paikallista tietokantayhteyttä SQLiteä vai ulkopuolista MySQL-tietokantayhteyttä. (liite 2, kuva d.)

Tietokannan yhteyttä pystytään automaattisesti hakemaan kaikkiin tarvittaviin malleihin laittamalla sovelluksen asetustiedostoon palvelimen tietokantasetukset kuntoon. Jos käyttäjä käyttää malleihin liittyviä tietokantafunktioita, ei yhteysfunktiota luokkatiedostoille tarvitse erikseen hakea.

Kirjoitettavissa tietokantalausekkeissa tarvitaan tietokantayhteysfunktio. Tämän jälkeen ei tarvitse muuta kuin käyttää Yiin omia SQL:llä tehtyjä funktioita. Funktiot toimivat samalla tavalla kuin *SELECT*, *INSERT*, *UPDATE* ja *DELETE*, mutta Yiissä on valmiit funkiot mallin sisällä (liite 3, kuva a), jotka suorittavat itse SQL-komennot helpommin mallin omien funktioiden avulla. Toinen vaihtoehto on tietojen käsittelyn automatisointi niin, että annetaan lauseketta varten tietoja parametrien (*bindParam)* avulla (liite 3, kuva b). (Yii Framework 2013n.)

Yiin ansiosta on tarjolla valmiita komponentteja, joista tietoja haetaan suoraan mallilta ja joilla tietoja voidaan liittää suoraan komponenttiin. Yiillä on tarjolla kattava määrä erilaisia komponetteja tiedon tulostamista varten. Kuvassa 6 käytetään Yiin CGridViewiä, joka on Yiin oma komponentti, jossa näytetään ruudukossa olevia tietoja mallista (liite 3, kuva c). Ensiksi ohjaimessa täytyy viitata malliin, jota haetaan, minkä jälkeen näkymän lähdekoodissa kirjoitetaan, mihin komponentti halutaan sijoittaa (liite 3, kuva d).



Kuva 6. Asetettu malli CGridView-komponentille ja näkymälle. Tuloksena saadaan taulukko, joka hakee kaikki käyttäjät User-mallin avulla.

3.3 Käyttäjät ja roolit

AmkLOG-sovellus perustuu kokonaan rooliperustaiseen sovellukseen. Sovellukseen sisältyy opiskelija-, opettaja-, moderaattori- ja ylläpitäjä-roolit. Opiskelijalla on oikeus tehdä perusasiat palvelussa, mm. perustaa kuva-albumi, kommentoida käyttäjien kuvia ja blogeja, liittyä ryhmiin, perustaa ryhmä ja liittyä erilaisiin kursseihin. Opettajalla on enemmän oikeuksia muokata sivustoa, mm. perustaa erilaisia kursseja, joihin opiskelijat voivat liittyä. Moderaattoreilla on suurempi oikeus hallita käyttäjiä, ryhmiä ja kursseja. Ylläpitäjällä on palvelun kaikki oikeudet ja oikeus hallinnoida koko sovellusta. Ylläpitäjä voi lisätä, poistaa ja muokata käyttäjiä, kursseja ja ryhmiä.

3.4 Sovelluksen sisältö

Sovelluksen rakenne jakautuu yhdeksään eri osaan: rekisteröinti, kirjautuminen, profiili, kuva-albumit, kuvat, blogit, ryhmät, kurssit ja hallintapaneeli. Kaikille näille on määritelty muutamat mallit ja jokaiselle näille on omat ohjaimet.

3.4.1 Rekisteröinti

Uudella vieraalla käyttäjällä on mahdollisuus rekisteröityä sivustolle ilmaiseksi rekisteröidy-lomakkeen kautta. Käyttäjä täyttää vaaditut kentät, mm. käyttäjätunnus, salasana ja sähköposti. Halutessaan käyttäjä voi kirjoittaa enemmän profiilitietoja rekisteröintivaiheessa tai vasta myöhemmin Asetukset-sivulla. Lomake tarkistaa Yiin mukana olevalla Ajaxilla, onko täytetyt kentät kirjoitettu oikein vai ei. Jos ei, käyttäjä saa virheilmoituksen Yiihin luodun malliluokan kautta. Rekisteröintisivulle on kirjoitettu, mitä käyttäjältä vaaditaan.

3.4.2 Kirjautuminen

Kirjautuminen tapahtuu joko etusivun kautta tai virallisen kirjautumissivun kautta. Kirjautuminen käyttää mallissa kahdenlaista tyyppiä *Front* ja *Back*. *Front*-tyypillä tarkoitetaan, että käyttäjäkirjautuminen tapahtuu etusivustojen kautta ja *Back*-tyyppi tarkoittaa, että käyttäjäkirjautuminen on tarkoitettu vain ylläpitäjille hallinnointisivustolle. *Front*- ja *Back*-tyypit huomataan vasta *LoginForm*-mallin metodissa. Jos *Frontiin* on viitattu *UserIdenty*-komponentti luokassa, kirjaudutaan etusivun kautta. Jos käytössä on *Back*, kirjaudutaan hallintapaneelin kautta (liite 4, kuva a). Jos käyttäjä onnistui kirjautumisessa, käyttäjä ohjataan etusivulle, jos ei, niin käyttäjää pyydetään antamaan vaaditut kirjautumisen kohdat. Sovelluksessa on pyritty siihen, että käyttäjä kirjautuu aina ensimmäisenä etusivun kautta (liite 4, kuva b).

3.4.3 Profiili

Käyttäjä esittelee profiilisivustolla itsensä ja kertoo henkilötietojen avulla, millainen opiskelija tai opettaja on kyseessä. Käyttäjällä on mahdollisuus rajata itsensä vain kavereille tai julkiseksi henkilöksi. Profiilisivustolla käyttäjällä on mahdollista ilmaista itseään lataamalla profiilisivustolle kuvan, joka myös linkittyy kuva-albumiin muiden nähtäväksi. Profiilisivustolla esitetään myös käyttäjään liitetyt kurssit ja ryhmät. Käyttäjä voi halutessaan myös käyttää toimintatyökalua, joka toimii widgettinä lisäämään itselleen blogiin kirjoituksen tai kuvan.

3.4.4 Kuva-albumit

Käyttäjä voi profiilisivustolla tuoda useita kuvia profiilille. Käyttäjän on mahdollista luoda albumeja omalle profiilille ja ladata albumin sisään uusia kuvia. Albumit ovat tällä hetkellä kaikille julkiset, mutta jatkokehityksessä on mahdollista luoda albumeita, jotka ovat nähtävillä vain yksityisesti tai kavereille. Kuvat skaalautuvat kuvia ladatessa mahdollisemman sopivaan kokoon, ettei palvelimelta mene paljon tilaa. Ne tallennetaan aina .jpg-formaattina. Kuvista tehdään komponentin avulla thumbnail-kuva eli esikatselukuva isompaa kuvaa varten. Kuvat tallennetaan käyttäjäkohtaiseen kansioon ja tietokantaan käyttäjän ID:n perusteella ja kuvan ID:n perusteella Kuvien ID:t viittaavat yleensä albumeihin ja albumit viittaavat kuviin sekä käyttäjiin (liite 5, kuva a).

3.4.5 Kuvat

Käyttäjän kuvat löytyvät käyttäjän albumeista. Käyttäjällä on heti rekisteröinnin jälkeen valmiina oletusalbumi, minne on valmiiksi ladattu oletuskuva. Käyttäjä voi milloin tahansa vaihtaa oletuskuvansa, kun hän on ladanut kuvia albumiin.

3.4.6 Blogit

Blogi on kaikkien rekisteröityjen käyttäjien käytössä. Blogiin pystytään erityisesti kirjoittamaan käyttäjäkohtaisia blogikirjoituksia liittyen joko vapaa-aikaan tai kouluun. Opiskelijalla on mahdollisuus kirjoittaa esimerkiksi kurssiin liittyvistä tehtävistä tai itsearvioinnista. Sekä opiskelija että opettaja pystyvät samalla tavalla kirjoittamaan blogiin ja jakamaan sen yksityisesti, vain kavereille tai julkisesti. Kaikkia käyttäjien kirjoittamia blogeja voi kommentoida kuka vain paitsi, jos se on rajattu tietyille käyttäjille tai rooleille.

3.4.7 Ryhmät

Kaikkien käyttäjien on mahdollista tutustua erilaisiin ryhmiin, liittyä ryhmiin tai jopa perustaa oma ryhmä. Käyttäjä voi myös kirjoittaa ryhmään keskusteluaiheita ja tarkastella, ketä kaikkia jäseniä on mukana ryhmässä. Ryhmässä näkyy etusivulla, minkälaisiin kategorioihin ryhmät on jaettu, montako ryhmää kategoria sisältää ja vielä, kuinka paljon jäseniä kussakin ryhmässä on. Ryhmässä näkyy enemmän tietoja vasta, kun kirjaudutaan ryhmään sisälle vasemmassa laidassa.

3.4.8 Kurssit

Kurssit ovat kaikille avoimia. Jos opiskelija tai opettaja kuuluu koulutusohjelmaan, jonka kurssista on kyse, hän voi liittyä kurssiin. Kursseille on pääsy vain silloin, jos opiskelija tietää salasanan. Tätä ominaisuutta ei ole tällä hetkellä palvelussa, mutta jatkokehityksessä se voi olla tarpeen luoda. Kurssilla ilmoitetaan kurssin ylläpitäjä, kurssin kuvaus sekä Moodle- ja Soleops-linkki kurssille. Kurssi-osiossa näkyy myös, ketkä kurssille ovat osallistuneet. Kurssilla pystytään keskustelemaan kurssiin liittyvistä asioista tai muuten vain kommentoimaan kurssisivun etusivulla.

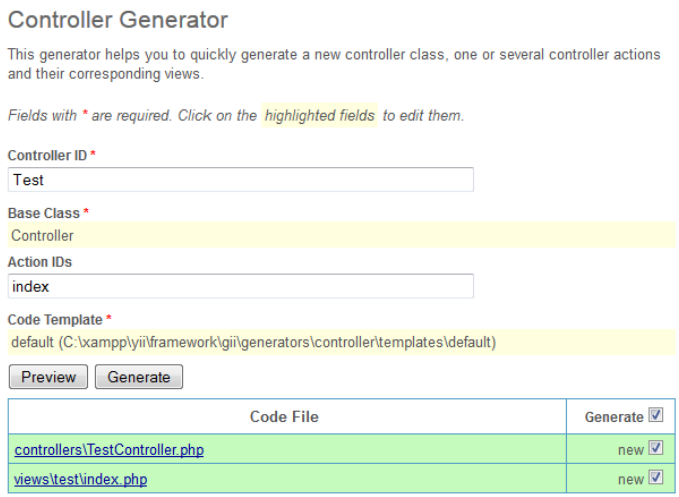
3.4.9 Hallintapaneeli

Sivuston ylläpitoon hallinnointisivu on tärkeä. Ylläpitäjillä ja moderaattoreilla on oikeus ylläpitää sivustossa mm. käyttäjiä, ryhmiä, kursseja ja myös kuvia. Hallintapaneeli on kuin valvontatyökalu ylläpitäjille. Hallintapaneelissa oikeudet rajoittuvat siten, että ylläpitäjillä on täydet oikeudet sivuston ylläpitäämiseen ja moderaattorilla on vain tietyt oikeudet. Esimerkiksi ohjaimessa on säädetty moderaattorille, että se ei voi poistaa mitään tietokannasta tai sovelluksesta. Sen voi tehdä ainoastaan ylläpitäjä. Moderaattori pystyy vain muokkaamaan ja lisäämään tietoja, kun taas ylläpitäjät pystyvät ohjauksen määrittelyssä tekemään kaikki lisäykset, muokkaukset ja poistot (liite 6, kuva a).

3.5 Sovelluksen ohjaimet

Sovellus käyttää yhdeksää eri ohjainta: SiteController, UserController, GroupsController, CoursesController, BrowserController, BlogsController, AlbumsController, AdminController ja ApiController. Näistä ApiController on ainoa ohjain, joka ei käytä mitään näkymää tällä hetkellä. SiteControlleria käytetään etusivua, rekisteröintisivua ja kirjautumissivua varten. UserControlleria käytetään käyttäjäsivuston hallinnointiin eli profiilisivujen ja kuvien ja albumien ohjaukseen. GroupsControlloria käytetään ryhmäsivujen ohjaukseen ja sama periaate on myös CourseControllersissa, joka ohjaa kurssin sivuja. BrowserController on selausohjaus-sivusto, jossa näytetään uusimmat luodut käyttäjät, uudet kuvat ja uudet blogikirjoitukset. BlogsController on blogeille tarkoitettu ohjaustiedosto, jota ei ole tässä työssä hyödynnetty ollenkaan vaan kaikki blogien toiminta tapahtuu UserControllersissa. AdminController vastaa sovelluksen hallintasivustoa. Tässä ohjaimessa on määritelty käyttäjille, kursseille ja ryhmille omat ohjauksensa. ApiController on sovelluksen toimintaohjain, jossa tapahtuu kaikki sivuston mekaaninen toiminta. Sen funktioihin on sisälletty muun muassa tiedon lisääminen, tiedon poistaminen ja tiedon muokkaaminen.

Ohjaimen lisääminen onnistuu parhaiten Gii-työkalulla. Asetuksissa täytyy vain määritellä, käyttääkö sovellus paikallista vai ulkopuolista tietokantaa. Kuvassa 7 näkyy, kuinka ohjain pystytään määrittelemään esimerkkisovellukseen.



Kuva 7. Ohjaimen generointi on helpointa Gii-työkalun avulla. Tässä tehdään testi-ohjain, joka luodaan antamalla ensimmäisenä kontrollin nimi tekstikenttään, jonka jälkeen kontrolli luodaan painamalla Generate-nappia.

3.5.1 Etusivu-ohjain

Etusivu-ohjaimella toimii kaikki olennainen pääsivulla. Index-ohjain on koko sovelluksen etusivu. Index-ohjaimen lisäksi toimii Kirjautuminen-ohjain ja Rekisteröinti-ohjain, jotka kuuluvat etusivulle oletuksena. Etusivu-ohjaimelle on syytä lisätä sellaisia ohjaimia, jotka toimivat parhaiten käyttäjän näkökulmasta. Käyttäjän on esimerkiksi aina parasta kirjautua etusivun kautta nopeuden ja helpon käytettävyyden vuoksi.

Etusivu-ohjaimen koodiesimerkissä selvennetään, miten Index-ohjaimesta haetaan tulostettava sivu ja miten se tapahtuu myös muissa sivuissa (liite 7, kuva a). Etusivu-ohjain ei käytä pelkästään Index-ohjainta vaan myös muita ohjaimia. Kaikki esimerkiksi action-alkuiset ohjaimet ovat saman pääohjaimen aliohjaimia. Etusivulle on luotu aliohjaimeksi actionLogin, jolla pystytään hyödyntämään kirjautumista etusivun kautta tai login-sivun kautta (liite 7, kuva b). ActionLogin-ohjaimelle välittyvät käyttäjän tiedot. Sieltä ne menevät LoginForm-mallille, jossa käsitellään käyttäjän kirjautumistietoja ja tarkistetaan käyttäjätunnus sekä salasana (liite 7, kuva c).

3.5.2 Käyttäjät-ohjain

Käyttäjät-ohjain sisältää kaikki käyttäjäkohtaiset toiminnot. Käyttäjän profiilin esittäminen tarvitsee oman ohjaimen näyttämään mallista haetun tiedon parametrin avulla. Oma ohjain tarvitaan myös siihen, että käyttäjä-parametrin avulla pystytään hakemaan käyttäjälle tietoa, joka näkyy myös ulkopuolisille. Parametrille syötetään Users-mallissa ensiksi oleva tietueen arvo. Tämän jälkeen parametrissa olevat tiedot välittyy Users-ohjaimelle ja siitä näkymälle. Erilaisia määrityksiä voi tehdä ohjaukselle *accessRulesin* kautta tai sitten ohjaimen sisään voi laittaa esimerkiksi erilaisia poikkeuksia käyttäjän ohjauksien rajaamiseksi. (liite 7, kuva d).

3.5.3 Ryhmät ja kurssit -ohjain

Ryhmät ja kurssit -ohjain toimii samalla tavalla kuin Käyttäjät-ohjain. Ohjaimelle on rajattu pääsy *accessRulesin* avulla, ja vain tietyt ryhmät pääsevät näkemään tämän sivun. Ryhmäohjaimelle välitetään samalla tavalla kuin parametrejä mallin avulla ja tulostetaan ne näkymälle sen jälkeen. Muutamissa ohjauksissa käytetään redirect-toimintoa, joka ohjaa käyttäjää edelliselle sivustolle. Tämä toiminto on hyödyllinen silloin, kun käyttäjä kirjoittaa esimerkiksi ryhmän foorumisivustolle (liite 8, kuva a).

Ohjain sallii vain opettajien, moderaattoreiden ja ylläpitäjien luoda esimerkiksi kurssi opiskelijoita varten. Kurssit-ohjain toimii samalla tavalla kuin ryhmät-ohjain. Access-Rules-määrittelyssä jouduttiin käyttämään lauseketta *(engl. expression)* eli ns. poikkeustapaa taulukon sisällä, jotta roolien tunnistautuminen toimisi paremmin ohjaimien kanssa. (Mental Issues 2013.) Roolimäärittelyssä esiintyy joskus häiriöitä, niin että sovellus ei osaa ottaa oikeaa roolia tietokannasta UserIdentity-luokan kautta. UserIdentity-luokassa tapahtuu käyttäjän tunnistautuminen ja siinä rekisteröidään käyttäjän tietoja istunnon ajaksi.

3.5.4 Selaa-ohjain

Sosiaalisen median käyttäjien, blogien ja kuvien kannalta on tärkeää, että hakutoiminto on nopea ja hyödyllinen. Yii Frameworkin ansiosta on tarjolla eräänlainen komponentti, joka osaa mallin mukaan sivuttaa käyttäjämääriä sivumäärän mukaisesti. CPagnation-luokan avulla saadaan selville tietojen määrä sivua kohti, hakulauseke ja yhden sivun koko eli se, kuinka paljon tuloksia mahtuu sivulle. Jokaisen ohjaimen sisään on rakennettu sivustusfunktio, joka hakee noin yhdeksän tulosta sivua kohti (liite 9, kuva a). Se ottaa parametrilla config-hakemiston main.php-tiedostosta myös tiedon, montako sivua halutaan näyttää näkymällä.

3.5.5 Ylläpito-ohjain

Ylläpito-ohjain sisältää käyttäjien, ryhmien ja kurssien tarvittavat ohjaimet. Niiden avulla pystytään vaikuttamaan tiedon lisäämiseen, muokkaamiseen ja poistamiseen. Ylläpito-ohjaimelle on valmiiksi generoitu CRUD-menetelmä (Create, Read, Update and Delete), jolla saadaan helpommin tehtyä ylläpito-ohjaimelle tarvittavat toiminnot, jotta käsin koodaamista voitaisiin välttää. CRUD-menetelmän ansiosta on saatu liitettyä näkymälle valmiita komponentteja, esimerkiksi valmiita lomakkeita tiedon täydennystä ja tulosten listausta varten ja valmiiksi luotuja nappeja tiedon poistamista ja muokkaamista varten. Tiedon poistamista ja muokkaamista varten on rakennettu Ajaxille pohja, josta pystytään välittämään tietoja Api-ohjaimelle. Ylläpito-ohjain suorittaa lisäämisen ja päivittämisen toimintoja. Lisäämiseen ja muokkaukseen on CRUDilla luotu valmiita lomakkeita, joita ei tarvitse käsin koodata. Ainoastaan silloin, kun lomake tarvitsee lisätietoja toisesta mallista, pitää käyttää virtuaalimuuttujia. Virtuaalimuuttujilla täytyy vielä viitata mallin rules-taulukossa oleviin tietoihin. Käyttäjä voi vapaasti valita, ovatko jotkut tiedot esimerkiksi lomakkeessa pakollisia vai eivät. Käyttäjä voi määrittää myös, voidaanko jotain virtuaalimuuttujaa etsiä jostakin mallista.

3.5.6 Api-ohjain

Api-ohjain on sovelluksen päätoimintojen ja pääfunktioiden ohjausluokka. Api-ohjaimessa toimii suurin osa sovelluksessa suoritettavista toiminnoista, joita käytetään vain näkymissä Ajaxien kanssa. Lomakkeelta syötetyt tiedot välitetään painamalla Lähetä-nappia, joka lähettää malliin lomakkeelta välitetyt tiedot. Täytyy huomioida vielä, että näkymään pitää luoda tietoja lähetettävä Javaskript-funktio, joka toimii Ajaxin yhteydessä. Api-ohjaimessa olevia sisäänrakennettuja $_POST- tai $_GET-metodeja hyödynnetään vasta, kun tietoja lomakkeelta on välitetty näkymältä Ajaxin avulla Api-ohjaimeen.

4 Tulokset ja johtopäätökset

Opinnäytetyön aihe oli vaativa ja haasteellinen. Työn toteuttaminen vaati laajaa perehtymistä sovelluskehyksiin ja niiden perusteisiin, toimintaan ja rakenteeseen.

4.1 Edut ja haitat

Yiillä on tällä hetkellä selkeäkieliset dokumentit ja helppokäyttöiset käyttöohjeet sovelluskehykseen perehtymistä ja sen testaamista ja pystyttämistä varten. Työssä selvitettiin, mitä etuja ja haasteita erilaisilla sovelluskehyksillä on olemassa. Sekä normaalissa koodauksessa että sovelluskehyksen käyttämisessä on sekä etuja että haasteita. Sovelluskehyksen käyttämisestä on kuitenkin selkeästi hyötyä. Sovelluskehyksen ansiosta voi hyödyntää monenlaisia apukirjastoja ja koodeja paljon paremmin. Sovelluskehyksen projektissa olevaa koodia voidaan käyttää uudelleen jopa uusissa tulevissa projekteissa. Sovelluskehyksien avulla käytetään monenlaisia työkaluja, jotka auttavat sovelluskehittäjiä sovelluksen luomisessa.

Tietokantalausekkeita oli paljon parempi automatisoida käyttämällä sovelluskehystä kuin käsin kirjoitettua PHP-koodia. Teoriassa tutkin myös, millä tavalla tietojen automatisointi tapahtuu sovelluksen ja sovelluskehyksen ympärillä. Sovelluskehyksen mallin tietokantafunktiot nopeuttavat työtä käsin tehtyyn koodiin verrattuna. On suositeltavaa käyttää sovelluskehyksen kaikkia luokan funktiota, jotka nopeuttavat muun muuassa sovelluksen olio-malleissa tietokantahakuja.

4.2 Suorituskyvyn mittaus yleisesti

Suorituskyvyn mittauksessa huomasin monien sovelluskehyksien nopeuserot, kun ladataan jotain esimerkkisivustoa, jossa on vain pelkkää leipätekstiä. Huomasin myös sen, että Yii ei ole tällä hetkellä nopein sovelluskehys vaikka näin oli puhetta nettifoorumien keskustelupalstoilla. Yii päätyi omien tietojeni perusteella toiseksi, mutta se ei käytännössä eroa paljon muista sovelluskehyksistä. Käytössäni oli apacheBench-sovellus, jota käytin Kapsi.fi:n palvelimella. Asensin kaikki viisi toisessa luvussa mainitsemaani WWW-sovelluskehystä palvelimelle ja testasin kaikki ajamalla ApacheBenchissa. Jos tämä olisi tehty esimerkiksi paikallisella palvelimella, niin tuloksena olisi saatu nopeampia aikoja, koska ulkopuoliselta palvelimelta kuormitetaan enemmän kuin paikallisella palvelimella (Localhost). Sovelluskehys, joka sai sivuston pyynnöissä suurimmat luvut, vasteajassa pienimmän luvun ja tiedonsiirrossa nopeimman siirron oli tuloksissa paras. Mittaustuloksia otettiin myös selaimen kautta eikä pelkästään komentokehotteen kautta. Selaimella pystyttiin mittaamaan, miten nopeasti sivusto, tietokantalauseke tms. suoriutuu Ajaxin kanssa. Tässä hyödynsin esimerkiksi Firefoxin tai Chromen liitännäistä Firebugia. Näkymälle piti ensiksi koodata vielä Ajaxilla toiminallinen funktio tiedon siirtoja varten. Selaimessa Firebug-liitännäinen kannattaa laittaa päälle ennen kun alkaa lähettää tietoja ohjaimelle, minkä jälkeen Firebug kerää nopeustietoja Ajaxin ja Api-ohjaimen välillä. Oman kokemuksen perusteella parhaiten mittaustuloksia saa komentokehotteen kautta mittaamalla ApacheBench-ohjelmalla, koska siinä kerrotaan selkeästi enemmän mittaustietoja kun taas selaimen kautta nähdään vain nopeustulos sekunneissa.

4.3 Sovelluskehyksen rakenteen tutkiminen

Sovelluskehyksen rakenteen tutkiminen antoi parhaat ohjeistukset itselleni sovelluskehyksen arkkitehtuurimalleista ja periaatteista. Selvisi myös, kuinka moni sovelluskehystä käyttää. Monet sovelluskehittäjät käyttävät jopa eri sovelluskehyksiä. Siksi itsekin päädyin käyttämään suosittua Yii Frameworkia. Aloin käyttää sitä ensiksi nopeuden mittaamiseeen ja proto-sovelluksen rakentamiseen monipuolisen tarjonnan takia. Loogisen rakenteen tutkiminen selkeytti, mitä Yii tarvitsee toimiakseen. Fyysisen rakenteen kohdalla oli helppo erotella, mitä tiedostoja, luokkia, rajapintoja ja hakemistoja pysytään erottamaan sovelluskehyksen rakenteesta. Kaikilla oli selvät hakemistorakenteet, joihin tallennetaan omanlaisina tiedostoina. Malleista tehdään oliomaisia sekä toiminnallisia luokkia, jotka hakevat tietokannasta tietoa ja mallien sisään upotetaan toiminallisia funktiota, jotta niitä pystytään hyödyntämään ohjaimissa ja näkymissä. Ohjaimille on tarvetta ohjata sivuja ja näyttää ne näkymälle. Näkymällä näytetään tulostettavaa tietoa selaimelle.

4.4 Tekninen toteutus

Opinnäytetyön teknisen toetutuksen tuloksena syntyi toimiva protoversio. Versiota voitaisiin sanoa alphaksi, koska siitä joutui ottamaan muutamia ominaisuuksia pois. Facebookin tyylistä protosovellusta varten täytyi ensiksi suunnitella raakaversio koko sovelluksen rakenteesta (liite 10.). Mietin jo suunnittelurakenteen piirtämisessä, että siitä ei tule aivan samanlaista kuin Facebook. Sovellus sisältää asioita, joita ammattikorkeakoulun sosiaalisen median palvelu oikeasti tarvitsee. Suunnittelukaavio jakaantui käyttäjiin, rooleihin, kursseihin, ryhmiin ja ylläpitoon. Yii Frameworkin Gii-työkalu oli hyödyllisin ja parhain tapa luoda koodillisesti helppokäyttöinen sosiaalisen median palvelun runko, jossa oli valmiiksi luokat ja rajapinnat sovellusta varten. Valmiin generoidun sovelluksen rungon ansiosta oli helppo koodata sivustoa eteenpäin. Käyttöön sai heti generoinnin jälkeen muuassa paljon erilaisia valmiita liitännäisiä ja widgettejä sivuston koodaamisen nopeuttamiseksi.

Hyvä ja monipuolinen esimerkki Yii-sovelluskehyksessä oli mallien yhdistäminen lomakkeissa. Jos yksi malli käyttää yhtä tietokannan taulua, niin ensisijaista mallia voi käyttää myös toisessa mallissa, joilloin se hyödyntää kahta tietokantataulua. Tämä tarkoittaa sitä, että mallille täytyy olla määritelty virtuaalimuuttujat toisesta mallista, jotta se toimisi. Ohjaimille oli yksinkertaisesta järjestää jokaiselle oma pääohjaimensa, esimerkiksi etusivustolle, hallintatyökalulle, käyttäjille, kursseille ja ryhmille oamat toimivat pääohjaimet. Näkymien luonti oli kuin staattisen WWW-sivun luontia.

Sovelluksen tietokantalausekkeiden automatisointi teknisessä toteutuksessa tehtiin kolmella eri tavalla. Lausekkeet toimivat malleissa ja jopa ohjaimissa. Ohjaimille välitetään esimerkiksi lomakkeella tuleva tieto, minkä jälkeen tietokantafunktio käsittelee sen ohjaimen sisällä. Automatisointi tehdään mallista, jossa toimivat lausekkeen funktiot. Niistä muodustuu yhdenlainen tietokantalauseke. Sivustoa on rakennettu ulkoasullisesti kaikille sivuston näkymille, jotta se on visuaalisesti mahdollisimman hyvän näköinen. (liite 11, kuva a). Profiili-sivustolle joutui luomaan sivuston sisälle erilaisia widgettejä, jotta koodin selkeyttäminen oli helpompaa. Ensiksi kirjoitetaan Components-kansioon uusi luokka-widgetti, minkä jälkeen tuodaan valmiiksi kirjoitettu widgetti näkymälle. Jos kaikki koodi olisi kirjoitettu pelkälle näkymälle, olisi koodista tullut liian spagettimaista. Profiilisivusta tuli paljon näyttävämmän näköinen visuaalisesti ja teknisesti widgettien avulla, koska widgettejä voidaan jatkossa käyttää muissakin sovelluksen sivustoilla eikä pelkästään profiilisivustolla. (liite 11, kuva b). Tällä hetkellä widgettejä käytetään myös profiilisivuston lisäksi etusivulla, kurssi-sivuilla ja ryhmä-sivuilla.

4.5 Mikä Yiillä onnistuu? Mikä ei onnistu?

Yiillä onnistui parhaiten MVC:n hyödyntäminen eli mallien, ohjaimien ja näkymien luonti sovelluksen ympärille, uusien widgettien luonti ja käyttö näkymissä, tietokantalausekkeiden automatisointi malleissa ja ohjaimissa, teeman käyttö asetus-tiedostossa ja laajennuksien lataaminen netistä ja käyttö sovelluksessa. Gii-työkalun ansiosta kaikkea ei tarvinnut koodata käsin. Lomakkeiden generointi on kaikille sovelluskehittäjille suuri etu. Lomakkeiden generaattori osaa luoda valmiiksi Ajaxilla toimivat validoitavat tekstikentät ja tuoda ohjaimelle suoraan tiedot valmiista mallista.

Erilaisia sovelluskehyksiä oli paras tapa lähteä tarkastelemaan erilaisilta nettisivustoilta ja keskustelupalstoilta. Erilaiset WWW-sovelluskehysgallupit ja vuosien aikana mitatut mittaustulokset internetissä antoivat minulle eniten apua siihen, mitä sovelluskehystä kannattaa lähteä soveltamaan. Erityisesti rakenteen kannalta sovelluskehysten valmiiden luokkien ja rajapintojen avulla pystytään jo sovellusta viemään eteenpäin niin, että koodin toistoa ei tarvitse lainkaan. Rakenteen avulla saadaan jokaiselle tulevalle sovelluskehyksen projektille runko ja valmiita koodeja helpottamaan koodin kirjoitusta. Jos sovelluskehyksellä ei olisi mitään valmiita käytettäviä komponentteja tai koodigeneraattoria valmiin koodin luomiseen, se olisi ajan hukkaa.

Sovelluskehyksien mittauksien kannalta parhaiten onnistui selvittää, millaista sovelluskehystä kannattaa käyttää jatkossa. Nopeuksien mittaaminen ei ollut vaikeaa, vaikka niin luulin. Mittaamista varten täytyi lukea ApacheBenchin dokumentteja ja seurata, mitä kyseinen komento tekee tietyillä parametreillä ja mikä niiden merkitys on ajettaessa komentoa palvelimella. Suorituskyvyn mittauksessa sain Yiin kohdalla selkeät nopeuserot verrattuna muihin sovelluskehyksiin. Codegniter olisi ollut toinen vaihtoehto myös käyttää sovelluksissa nopeuden takia, mutta Yii voitti monipuolisuuden takia. Yii antoi sen lisäksi nopeampia tuloksia verrattuna muihin tunnettuihin WWW-sovelluskehyksiin vaikka oli palvelimelta lisäasetuksia laitettu päälle mm. APC (Alternative PHP Cache). Yiissä täytyi huomioida ennen mittausta, että täytyi olla debug-tilassa sekä tehty esimerkkisivusto esimerksi ”Hello World”-tekstillä, jotta välttyi virheilmoituksilta.

Suuria epäonnistumisia ei ollut tutkimisen sekä teknisen toteutuksen kohdalla. Aliohjelmien luonti pääohjaimille sekä roolien asettaminen käyttäjille epäonnistuivat Yiillä, koska roolien haku ei toimi kunnolla toiminta-ohjaimissa. Yleinen syy oli tähän, että tietokannasta haetut roolit käyttäyvät eri tavalla kuin Yiin omat roolit. Ylläpito-ohjaimelle olisi voinut luoda erillisen aliohjaimen käyttäjä-ohjainta, ryhmä-ohjainta sekä kurssi-ohjainta varten. Nopeamman ja helpomman tavan kautta tein Ylläpito-ohjaimelle niin että loin ensimmäisenä Ylläpito-ohjainta varten alihakemistot käyttäjähallinnalle, kurssihallinnalle ja ryhmähallinnalle, jotta Ylläpito-ohjaimen kautta muita ohjaimia olisi helpompi löytää.

5 Pohdinta

Työn aikana opin paljon uusia ja hyödyllisiä asioita sovellukehyksien periaatteista ja niiden soveltamisesta. Työssäni opin ensisijaisesti MVC (Model-View-Controller) -arkkitehtuurimallin hyödyntämistä. Siihen tutustuminen ja perehtyminen vaati paljon aikaa. Sovelluskehyksen arkkitehtuurimallin ymmärtäminen, oppiminen ja käyttöönotto vaati paljon itsekuria, jotta sitä myös osasi käyttää syvällisemmin jopa Yii Frameworkissakin. Vaikka Yii on alussa vaikea käyttää, niin helpoiten ja parhaiten saa ohjeistukset foorumeilta sekä Yiin kotisivuilta, kun lukee dokumentaatiot tarkasti. Sovelluskehyksiin perehtyminen ja Yii Frameworkin käyttö antoivat paljon työkaluja monipuolisen sosiaalisen median palvelun rakentamiseen. Ladattavat liittännäiset, mallit ja ohjaimet tekivät sovelluksesta paljon näyttävämmän ja koodiltaan selkeän kokonaisuuden luokkien ja rajapintojen ansiosta. Jos sovellus olisi rakennettu pelkällä PHP-koodilla, tuloksena olisi syntynyt hieman huonompi kokonaisuus. Olisi ollut hankalaa tehdä web-sovellusta ilman sovelluskehystä, mutta sovelluskehys antoi valmiit komponentit, jotka nopeuttivat koodin kirjoittamista sovelluksessa. Sosiaalisen median palvelu toimisi tulevaisuudessa suuressa AMK-verkostossa parhaiten, jos AMKLog saisi paljon ytikemästä jatko-kehitystä jossakin verkkosovellustiimissä tai jossain muussa työorganisaatiossa. Yii Framework oli minulle aivan uusi menetelmä ja tapa tehdä tästä sovelluskehyksestä korkeamman asteen sovelluksia.

Lähteet

Appweb Embedded Web Server 2013a.
http://appwebserver.org/products/appweb/server-side-javascript.html. 25.8.2013.

Best Web-Frameworks. 2013. Compare PHP Frameworks.
http://www.bestwebframeworks.com/compare-web-frameworks/php/. 20.5.2013.

Gellersen H, Gaedke M. 1999. S&R Technology Group.
http://www.srtechgroup.com/oo_web.pdf. 18.5.2013.

Mattsson, M. 2000. Evolution and Composition of Object-Oriented Frameworks.
http://www.bth.se/faculty/mma/papers/michael.mattsson.ph.d.thesis.pdf. 20.5.2013.

Media Infonet. 2008. MVC Design Pattern.
http://www.mediainfonet.com/blog/yii-mvc-design-pattern/. 11.5.2013.

Mental Issues: Yii Tips: Add user role to accessRules() in controller
http://mentalissue.blogspot.fi/2012/10/yii-tips-add-user-role-to-accessrules.html. 25.8.2013.

PHPFrameworks.com. 2013. TOP-10 Ranking PHP Frameworks.
http://www.phpframeworks.com/top-10-php-frameworks/. 4.10.2013.

Roihuvaara, J. 2013. Kehysriippuvaisen PHP-ohjelmiston päivitettävyyden arviointi
http://tutkielmat.uta.fi/pdf/gradu06608.pdf. 18.5.2013.

TTY Ohjelmistotekniikka. 2009. Ohjelmistoarkkitehtuurit.
http://www.cs.tut.fi/~ohar/luennot/luennot2009/Ohar13%20Kehykset.pdf. 11.5.2013.

Wikipedia. 2013a. Dont_Repeat_Yourself.
http://en.wikipedia.org/wiki/Don%27t_repeat_yourself. 16.5.2013.

Wikipedia. 2013b. Model-View-Controller.
http://fi.wikipedia.org/wiki/MVC. 13.5.2013.

Wikipedia 2013c. ORM.
http://en.wikipedia.org/wiki/Object-relational_mapping 9.9.2013.

Wikipedia. 2013d. PHP.
http://fi.wikipedia.org/wiki/PHP. 13.5.2013.

Wikipedia. 2013e. SQL.
http://fi.wikipedia.org/wiki/SQL. 13.5.2013.

Wikipedia. 2013f. SQLite.
http://fi.wikipedia.org/wiki/SQLite. 16.5.2013.

Wikipedia. 2013g. WWW.
http://fi.wikipedia.org/wiki/WWW. 13.5.2013.

Yii Framework. 2013a. About.
http://www.yiiframework.com/about/. 13.5.2013.

Yii Framework .2013b. Features.
http://www.yiiframework.com/features/. 13.5.2013.

Yii Framework. 2013c. Performance.
http://www.yiiframework.com/performance/. 13.5.2013.

Yii Framework. 2013d. Fundamentals: Model.
http://www.yiiframework.com/doc/guide/1.1/en/basics.model. 13.5.2013.

Yii Framework. 2013e. Fundamentals: View.

http://www.yiiframework.com/doc/guide/1.1/en/basics.view. 13.5.2013.
Yii Framework. 2013f. Fundamentals: Controller.
http://www.yiiframework.com/doc/guide/1.1/en/basics.controller.
13.5.2013.
Yii Framework. 2013g. Fundamentals: Application.
http://www.yiiframework.com/doc/guide/1.1/en/basics.application.
13.5.2013.
Yii Framework. 2013h. Special Topics: URL Management.
http://www.yiiframework.com/doc/guide/1.1/en/topics.url. 13.5.2013.
Yii Framework 2013i. Fundamentals: Module.
http://www.yiiframework.com/doc/guide/1.1/en/basics.module.
25.8.2013.
Yii Framework 2013j. Extending Yii: Creating Extensions.
http://www.yiiframework.com/doc/guide/1.1/en/extension.create.
25.8.2013.
Yii Framework 2013k. Understanding Autoloading, Helpers Classes and Helper Functions.
http://www.yiiframework.com/wiki/165/understanding-autoloading-helper-classes-and-helper-functions/.
25.8.2013.
Yii Framework 2013l. Understanding “Assets”.
http://www.yiiframework.com/wiki/148/understanding-assets/. 25.8.2013.
Yii Framework 2013m. How to customize Yii core messages?
http://www.yiiframework.com/wiki/18/how-to-customize-yii-core-messages/. 25.8.2013.
Yii Framework 2013n. Working with Databases: Database Access Objects.
http://www.yiiframework.com/doc/guide/1.1/en/database.dao.
25.8.2013.

Sovelluskehyksen suorituskyvyn mittaaminen

```
tribuutti@lakka:~$ ab -n 2000 -c 10 http://tribuutti.kapsi.fi/benchmarks/yiitest/index.php?r=say/hello
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking tribuutti.kapsi.fi (be patient)
Completed 200 requests
Completed 400 requests
Completed 600 requests
Completed 800 requests
Completed 1000 requests
Completed 1200 requests
Completed 1400 requests
Completed 1600 requests
Completed 1800 requests
Completed 2000 requests
Finished 2000 requests


Server Software:        Apache/2.2
Server Hostname:        tribuutti.kapsi.fi
Server Port:            80

Document Path:          /benchmarks/yiitest/index.php?r=say/hello
Document Length:        1967 bytes

Concurrency Level:      10
Time taken for tests:   7.947 seconds
Complete requests:      2000
Failed requests:        2
   (Connect: 0, Receive: 0, Length: 2, Exceptions: 0)
Write errors:           0
Non-2xx responses:      2
Total transferred:      4676946 bytes
HTML transferred:       3931286 bytes
Requests per second:    251.67 [#/sec] (mean)
Time per request:       39.735 [ms] (mean)
Time per request:       3.974 [ms] (mean, across all concurrent requests)
Transfer rate:          574.72 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    1   2.0      1      57
Processing:    12   39  95.0     29    1860
Waiting:       12   38  95.0     29    1859
Total:         13   40  95.4     30    1870

Percentage of the requests served within a certain time (ms)
  50%     30
  66%     35
  75%     38
  80%     41
  90%     47
  95%     59
  98%    107
  99%    206
 100%   1870 (longest request)
tribuutti@lakka:~$
```

Kuva a. Ennen mittausta kirjaudutaan palvelimeen shellin (SSH) kautta, minkä jälkeen ajetaan WWW-sovelluskehystä varten suorituskykyajon komento.

Koodiesimerkit 1: Asetustiedoston konfigurointi

```
// uncomment the following to enable URLs in path-format
'urlManager'=>array(
        'urlFormat'=>'path',
        'showScriptName'=>false,
        'caseSensitive'=>false,
        'rules'=>array(

                // API PATTERNS
                //array('api/setuserdefaultimage', 'pattern'=>'api/<model:\w+>', 'verb'=>'POST'),
                //array('api/sendgroupcomment', 'pattern'=>'api/<model:\w+>', 'verb'=>'POST'),
                // API PATTERNS

                // CONTROLLER SETTINGS
                //'<controller:\w+>/<id:\d+>' => '/view',
                //'<controller:\w+>/<action:\w+>/<id:\d+>'=>'<controller>/<action>',
                //'<controller:\w+>/<action:\w+>'=>'<controller>/<action>',

                // SITE SETTINGS
                '/'=>'site/index',
                'login'=>'site/login',
                'logout'=>'site/logout',
                'contact'=>'site/contact',
                'gii'=>'gii/default/login',
                'register'=>'site/register',
```

Kuva a. URL-osoitteiden hallinta. URL-osoitteita pystytään asettamaan käyttäjäystävälliseksi.

```
// application components
'components'=>array(
        'file'=>array(
            'class'=>'application.extensions.file.CFile',
        ),
            'image'=>array(
            'class'=>'application.extensions.image.CImageComponent',
            // GD or ImageMagick
            'driver'=>'GD',
            // ImageMagick setup path
            'params'=>array('directory'=>'/opt/local/bin'),
        ),
        'user'=>array(
                // enable cookie-based authentication
                'allowAutoLogin'=>true,
        ),

        'phpThumb'=>array(
            'class'=>'ext.EPhpThumb.EPhpThumb',
        ),
        'authManager'=>array(
          'class'=>'PhpAuthManager',
          'defaultRoles' => array('guest'),
        ),
        // uncomment the following to enable URLs in path-format
        'urlManager'=>array(
                'urlFormat'=>'path',
                'showScriptName'=>false,
                'caseSensitive'=>false,
                'rules'=>array(

                        // API PATTERNS
                        //array('api/setuserdefaultimage', 'pattern'=>'api/<
                        //array('api/sendgroupcomment', 'pattern'=>'api/<mod
                        // API PATTERNS
```

Kuva b. Komponentit ovat hyödyllisiä sovelluksessa, jos koodin kirjoittamisesta halutaan nopeampaa ja yksinkertaistaa. Komponenteille täytyy tehdä viittaus config-hakemistossa olevaan main.php-tiedostoon. Tällöin pystytään jakamaan komponentteja kaikissa malleissa tai ohjaimissa.

```
// application-level parameters that can be accessed
// using Yii::app()->params['paramName']
'params'=>array(
        // this is used in contact page
        'adminEmail'=>'webmaster@example.com',
        'listPerPage'=> 9, // <-- insert this line with the number you prefer
),
```

Kuva c. Asetustiedostossa olevilla ”params” eli parametreilla pystytään asettamaan tieto luoduille malleille.

```
/*
'db'=>array(
        'connectionString' => 'sqlite:'.dirname(__FILE__).'/../data/testdrive.db',
),
 *
 */
// uncomment the following to use a MySQL database

'db'=>array(
        'connectionString' => 'mysql:host=localhost;dbname=testikanta',
        'emulatePrepare' => true,
        'username' => 'root',
        'password' => '1234',
        'charset' => 'utf8',
),
```

Kuva d. Tietokannan yhteysasetuksen laittaminen asetustiedostossa on yksinkertaista. Tämän jälkeen ei tarvitse erikseen muissa tiedostoissa ottaa yhteyden ottoa. Ylempi (kommenttien alla) käyttää paikallista yhteyttä ja alla oleva käyttää ulkopuolista tietokantayheyttä.

Koodiesimerkit 2: Tietokannan automatisointia

```
$model->attributes=$_POST['Course'];
if($model->validate())
{
    // form inputs are valid, do something here
    $sday                = $model->sday;
    $smonth              = $model->smonth;
    $syear               = $model->syear;
    $model->start_date   = $syear . "-" . $smonth . "-" . $sday;

    $eday                = $model->eday;
    $emonth              = $model->emonth;
    $eyear               = $model->eyear;
    $model->end_date    = $eyear . "-" . $emonth . "-" . $eday;

    $model->save(false);
    $lastID = $model->primaryKey;

    $cm = new CourseMember;
    $ct = new CourseTeacher;

    $cm->cid = $lastID;
    $cm->uid = Yii::app()->user->id;

    $cm->save();

    $ct->cid = $lastID;
    $ct->uid = Yii::app()->user->id;

    $ct->save();
}
```

Kuva a. *INSERT*-tyylinen funktio Yiissä. Mallille on tuotu tietoja lomakkeelta, minkä jälkeen välitetään tietoja mallin attribuuteille ja sen jälkeen tallennetaan tietokantaan save-funktiolla.

```
public function getUserImage($image_id)
{
    $dbC = Yii::app()->db->createCommand();
    $dbC->setFetchMode(PDO::FETCH_OBJ);
    $dbC->select()->from('images')->where('image_id = :image_id');
    $dbC->bindParam(":image_id", $image_id);

    $username = $_GET['username'];

    foreach ($dbC->queryAll() as $row)
    {
        echo CHtml::image(Yii::app()->baseUrl . "/images/profileimages/{$username}/" . $row->image_url);
    }
}
```

Kuva b. Ennen createCommand-funktiota haetaan tietokantayhteys, minkä jälkeen pysytytään luomaan tietokantalauseke. Automatisoinnissa toimii tietojen haussa kätevästi Yii:n omat funktiot: SELECT, FROM ja WHERE, jotka viittaavat samalla tavalla kuin SQL:n lausekeessa.

```php
public function getUserFriends($uid)
{
    $dbC = Yii::app()->db->createCommand();
    $dbC->setFetchMode(PDO::FETCH_OBJ);
    $dbC->select("u.username, i.image_id, i.thumbnail_url")
            ->from("friends AS f")->join("users AS u","u.uid = f.friend_uid")
            ->join("images AS i", "i.uid = f.friend_uid")
            ->where("f.user_uid = :uid")
            ->andWhere("f.friend_request = 'false'")
            ->group("f.friend_uid");
    $dbC->bindParam(":uid", $uid);

    $results = $dbC->queryAll();
    $count = count($results);

    if($count!=0)
    {
        foreach ($dbC->queryAll() as $row)
        {
            echo "<div class='userblock'><a href='" . Yii::app()->baseUrl . "/u
        }
    }
    else
    {
            echo "<p>Sinulla ei ole yhtään kaveria.</p>";
    }
}
```

Kuva c. Tietokannan automatisointia parametrien avulla sekä Yiin valmiilla SQL-funktioilla. Parametrillä (*bindParam*) ilmoitetaan, mistä ja minkälaista tietoa halutaan hakea lausekkeen avulla.

```php
public function actionUsers()
{
    $model=new User('search');
    $model->unsetAttributes();  // clear any default values
    if(isset($_GET['User']))
            $model->attributes=$_GET['User'];

    $this->render('users/index',array(
            'model'=>$model,
    ));
    //$this->render("users/index");
}
```

Kuva d. Ohjaimessa täytyy määritellä parametrien avulla malli näkymää varten, jos halutaan tuoda tietoja komponentille. Tässä tapauksessa halutaan hakea User-mallista kaikki tiedot.

```php
<?php $this->widget('zii.widgets.grid.CGridView', array(
	'id'=>'users-grid',
	'dataProvider'=>$model->search(),
	'filter'=>$model,
	'ajaxUpdate'=>false,
	'columns'=>array(
		'uid',
		'username',
		//'password',
		'uemail',
		'register_date',
		'login_date',
		'role',
		array(
			'class'=>'CButtonColumn',
			'template'=>'{delete}{update}',
			'deleteConfirmation'=>"js:'Oletko varma, että ha
			'afterDelete'=>'function(link,success,data){ if(
			'buttons'=>array(
				'update'=>array(
					'url'=>'$this->grid->control
					),
				'delete'=>array(
					'url'=>'$this->grid->controll
					),
				),
		),
	),
)); ?>
```

Kuva a. CGridView-komponentti ja ohjaimen kautta tuotu User-malli.

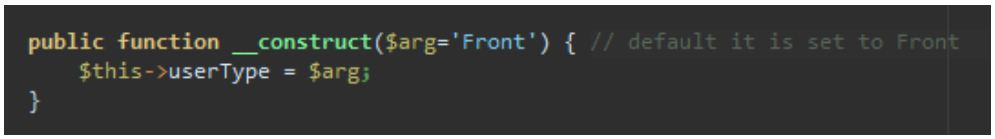
Koodiesimerkit 3: Kirjautumisen toiminallisuudet

```
class UserIdentity extends CUserIdentity
{
        /**
         * Authenticates a user.
         * The example implementation makes sure if the username and password
         * are both 'demo'.
         * In practical applications, this should be changed to authenticate
         * against some persistent user identity storage (e.g. database).
         * @return boolean whether authentication succeeds.
         */
        public $userType = 'Front';
        private $_id;

        public function authenticate()
        {
            if($this->userType=='Front') // This is front login
            {
                $username=strtolower($this->username);
                $user = User::model()->find('LOWER(username)=?',array($username));

                if($user == null)
                {
                    //echo "Väärät nimi";
                    $this->errorCode=self::ERROR_USERNAME_INVALID;
                }
                else if($this->username != $user->username)
                {
                    echo "Väärät tunnukset";
                    $this->errorCode=self::ERROR_USERNAME_INVALID;
                }
                else if(md5($this->password) != $user->password)
                {
                    echo "Väärät salasanat!";
                    $this->errorCode=self::ERROR_PASSWORD_INVALID;
                }
                else
                {
                    $this->_id=$user->uid;
                    $this->username=$user->username;
```

Kuva a. UserIdentity komponentti-luokassa esitellään käyttäjäntyypin kirjautumista. Front on etusivulle kirjautumista varten ja Back on hallintapaneelille kirjautumista varten.



```
public function __construct($arg='Front') { // default it is set to Front
    $this->userType = $arg;
}
```

Kuva b. Oletuksena on asetettu LoginForm-malliluokassa Front, jolloin käyttäjä aina kirjautuu etusivuston kautta.

```php
<?php
$model = new LoginForm('Front');
?>
<div id="left">
        <div class="bar"></div>
        <div class="content">
                <?php $this->widget('UserLogin'); ?>
        </div>

        <div class="bar"></div>
        <div class="content">
                <?php $this->widget('TopGroups'); ?>
        </div>

        <div class="bar"></div>
        <div class="content">
                <?php $this->widget('TopCourses'); ?>
        </div>
</div>

<div id="right">
        <div class="bar"></div>
        <div class="content">
                <h2>Tervetuloa AMKLogiin!</h2>
                <p>Lorem Ipsum on yksinkertaisesti testaus
        </div>

        <div class="bar"></div>
        <div class="content">
            <?php $this->widget('listUsers'); ?>
        </div>

</div>
```

Kuva c. Sivuston pääsivustolle on asetettu oliomalliluokka LoginForm, joka käyttää kirjautumistyyppiä Front, jolloin kirjaudutaan etusivun kautta.

Koodiesimerkit 4: Kuvan lataus komponentin avulla

```
$thumbnail_image = "./images/profileimages/$username/thumb" . $filename;
$original_image  =  $filepath;

$thumb=Yii::app()->phpThumb->create($filepath);
$thumb->resize(150,150);
$thumb->save($thumbnail_image);

$thumb=Yii::app()->phpThumb->create($filepath);
$thumb->resize(500,500);
$thumb->save($original_image);

// USER INFORMATION
$uid = Yii::app()->user->id;
$album_id = $model->album_id = $_POST['Images']['album_id'];
$is_default_image            = $_POST['Images']['is_default_image'];
// USER INFORMATION

$connection=Yii::app()->db;
$sql="INSERT INTO images (album_id, uid, thumbnail_url, image_url) VALUES(:album_id, :uid, :thumbnail_url, :image_url)";
$command=$connection->createCommand($sql);
$command->bindParam(":album_id",$album_id);
$command->bindParam(":uid",$uid);
$command->bindParam(":thumbnail_url",$thumbFile);
$command->bindParam(":image_url",$originalFile);
$command->execute();
```

Kuva a. Kuvan lataaminen tarvitsee tarkastelun siitä, onko kuva liian iso vai liian pieni latautuakseen. Kuva muotoillaan sopivan kokoiseksi phpThumb-lisäosan avulla ja liitetään sen jälkeen tietokantaan

Koodiesimerkit 5: Hallintapaneelin roolien käyttöoikeudet

```php
/**
 * Specifies the access control rules.
 * This method is used by the 'accessControl' filter.
 * @return array access control rules
 */
public function accessRules()
{
	return array(
		array('allow',  // allow all users to perform 'index' and 'view' actions
			'actions'=>array('index','users', 'settings', 'userdelete','courses', 'settingscourse', 'addcourse', 'coursedelete',
			'expression' => function(){return Yii::app()->user->getState("roles") == 'admin';},
			'users'=>array('@'),
		),
		array('allow',  // allow all users to perform 'index' and 'view' actions
			'actions'=>array('index','users', 'settings','courses', 'settingscourse', 'addcourse', 'groups', 'addgroup', 'settin
			'expression' => function(){return Yii::app()->user->getState("roles") == 'moderator';},
			'users'=>array('@'),
		),
		array('deny',  // deny all users
			'users'=>array('*'),
		),
	);
}
```

Kuva a. Hallintapaneeliin voi aina miettiä, mitä sääntöjä haluaa jakaa rooleille. Tässä koodiesimerkissä haluttiin, että moderaattorit pystyvät kaiken lisäkysen ja muokkauksen tekemään, mutta ylläpitäjällä on täydet oikeudet.

Koodiesimerkit 6: Sovelluksen ohjaimet

```
/**
 * This is the default 'index' action that is invoked
 * when an action is not explicitly requested by users.
 */
public function actionIndex()
{
        // renders the view file 'protected/views/site/index.php'
        // using the default layout 'protected/views/layouts/main.php'
        $this->render('index');
}

/**
 * This is the action to handle external exceptions.
```

Kuva a. Perinteisesti välitetään sivusto render-funktion avulla näkymälle. Näkymä muotoilee ja lähettää sen käyttäjän selaimelle.

```
/**
 * Displays the login page
 */
public function actionLogin()
{
        $model=new LoginForm;

        // if it is ajax validation request
        if(isset($_POST['ajax']) && $_POST['ajax']==='login-form')
        {
                echo CActiveForm::validate($model);
                Yii::app()->end();
        }

        // collect user input data
        if(isset($_POST['LoginForm']))
        {
                $model->attributes=$_POST['LoginForm'];
                // validate user input and redirect to the previous page if valid
                if($model->validate() && $model->login())
                        $this->redirect(Yii::app()->user->returnUrl);
        }
        // display the login form
        $this->render('login',array('model'=>$model));
}
```

Kuva b. Kirjautumis-ohjaimelle tuodaan LoginForm-malli, joka sisältää Kirjautumislomakkeen täydennetyt tiedot näkymältä. Mallin tiedot välitetään sen jälkeen validoitavaksi ja siitä mallin kirjautumis-funktiolle, jossa tarkistetaan käyttäjän nimimerkki ja salasana.

```
*/
public function login()
{
        if($this->_identity===null)
        {
                $this->_identity=new UserIdentity($this->username,$this->password);
                $this->_identity->authenticate();
        }
        if($this->_identity->errorCode===UserIdentity::ERROR_NONE)
        {
                $duration=$this->rememberMe ? 3600*24*30 : 0; // 30 days
                Yii::app()->user->login($this->_identity,$duration);
                return true;
        }
        else
                return false;
}
```

Kuva c. Kirjautumis-ohjaimelta tuodaan käyttäjätiedot LoginFormin login-funktiolle. UserIdentityn-luokassa tapahtuu tunnuksen ja salasanan tarkastus. Sieltä tulee myös paluuviesti käyttäjälle.

```php
public function actionView($username)
{
    $model = User::model()->find("username = '" . $username . "'");
    if($model==NULL)
    {
        throw new CHttpException(404,'The specified post cannot be found.');
    }
    $this->render('view',array('model'=>$model));
}

public function actionAlbums($username)
{
    $model = User::model()->find("username = '" . $username . "'");
    if($model==NULL)
    {
        throw new CHttpException(404,'The specified post cannot be found.');
    }
    $this->render('albums',array('model'=>$model));
}

public function actionBlogs($username)
{
    $model = User::model()->find("username = '" . $username . "'");
    if($model==NULL)
    {
        throw new CHttpException(404,'The specified post cannot be found.');
    }
    $this->render('blogs',array('model'=>$model));
}

public function actionAlbum($username, $album_id)
{
    $user = User::model()->find("username='". $username ."'");
    if($user==NULL)
    {
        throw new CHttpException(404,'The specified post cannot be found.');
    }
    else
    {
        $uid = $user->uid;
    }

    $model = Images::model()->find("uid = '" . $uid . "'");
    if($model==NULL)
    {
        throw new CHttpException(404,'The specified post cannot be found.');
    }
    $this->render('album',array('model'=>$model));
}
```

Kuva d. Käyttäjä-ohjaimen ohjaimia parametrien kanssa. Tieto tulee mallista, joka sen jälkeen välitetään näkymälle näyttäväksi tiedoksi.

Koodiesimerkit 7: Hallintapaneelin roolien käyttöoikeudet

```
public function accessRules()
{
        return array(
                array('allow',  // allow all users to perform 'index' and 'view' actions
                        'actions'=>array('index','degree_programme','course','forums','forum','addtopic','post','members'),
                        'users'=>array('@'),
                ),
                array('allow',  // allow all users to perform 'index' and 'view' actions
                        'actions'=>array('addcourse'),
                        'expression' => function(){return Yii::app()->user->getState("roles") == 'teacher';},
                        'users'=>array('@'),
                ),
                array('allow',  // allow all users to perform 'index' and 'view' actions
                        'actions'=>array('addcourse'),
                        'expression' => function(){return Yii::app()->user->getState("roles") == 'admin';},
                        'users'=>array('@'),
                ),
                array('allow',  // allow all users to perform 'index' and 'view' actions
                        'actions'=>array('addcourse'),
                        'expression' => function(){return Yii::app()->user->getState("roles") == 'moderator';},
                        'users'=>array('@'),
                ),
                array('deny',  // deny all users
                        'users'=>array('*'),
                ),
        );
}
```

Kuva a. *accessRulesilla* on hyödyllisin tapa rajata käyttäjiä kurssiohjaimen ja ryhmäohjaimen olevia verkkosivustoja. Kurssissa vain opettajat, ylläpitäjät ja moderaattorit pystyvät esimerkiksi lisäämään kursseja.

Koodiesimerkit 8: Selaa-ohjauksen sivunumerointi komponentin avulla

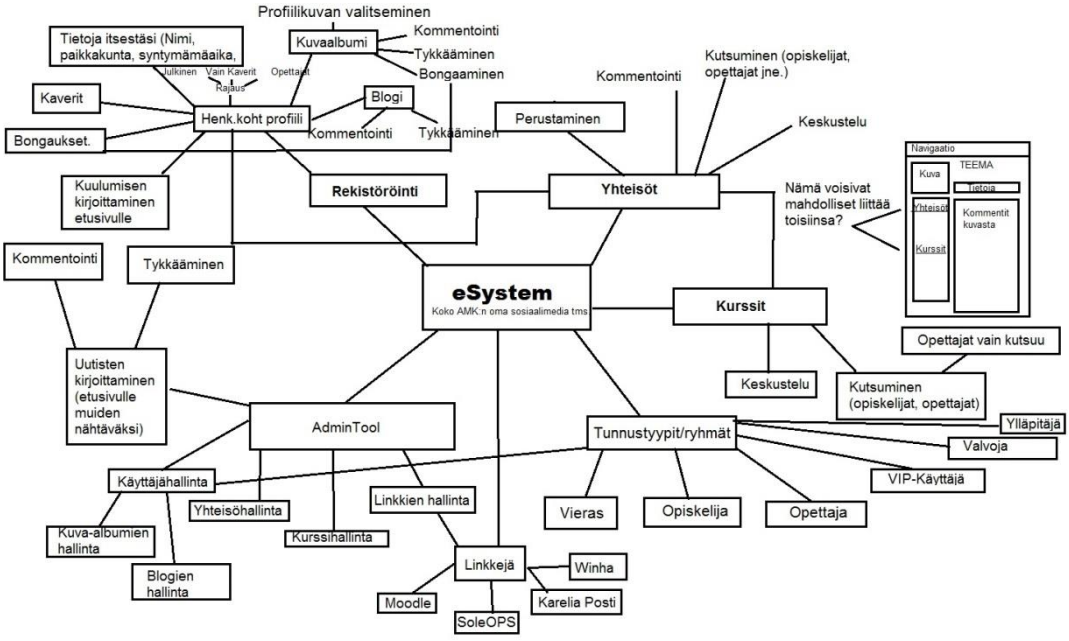
```
//////////////////////////////
// Site Browser Methods
//////////////////////////////
public function actionUsers()
{
        $this->layout="browser_page";
        $criteria = new CDbCriteria();
        $item_count = User::model()->count($criteria);

        $pages = new CPagination($item_count);
        $pages->setPageSize(Yii::app()->params['listPerPage']);
        $pages->applyLimit($criteria);  // the trick is here!

        $this->render('users',array(
                    'model'=> User::model()->findAll($criteria), // must be the same as $item_count
            'item_count'=>$item_count,
            'page_size'=>Yii::app()->params['listPerPage'],
            'items_count'=>$item_count,
            'pages'=>$pages,
        ));
}
```

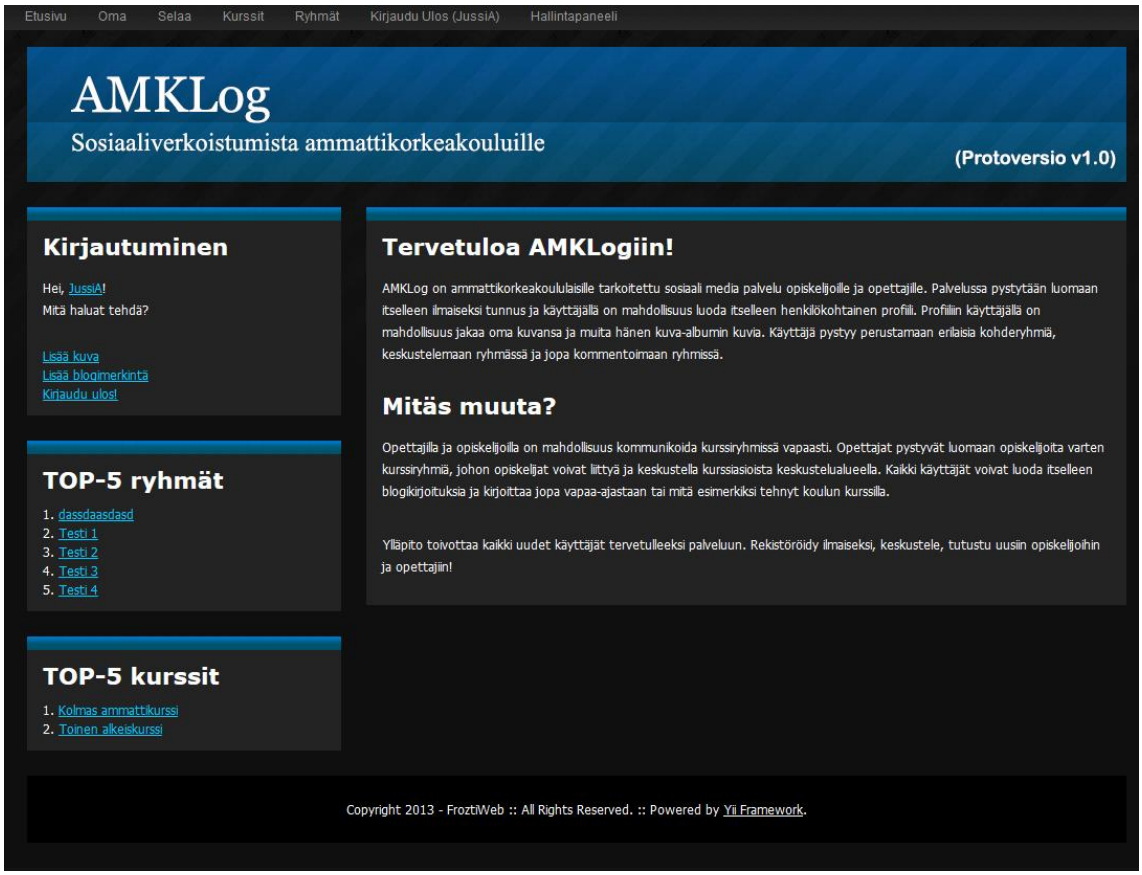
Kuva a. Selauksessa on tarvetta sivunumeroinille ja näyttää rajallinen määrä saatuja hakutuloksia. Parametrien avulla saadaan selville mallista ainakin tuotu sivumäärä ja tulostemäärä per sivu.

Sosiaalisen median palvelun rakennekaavio

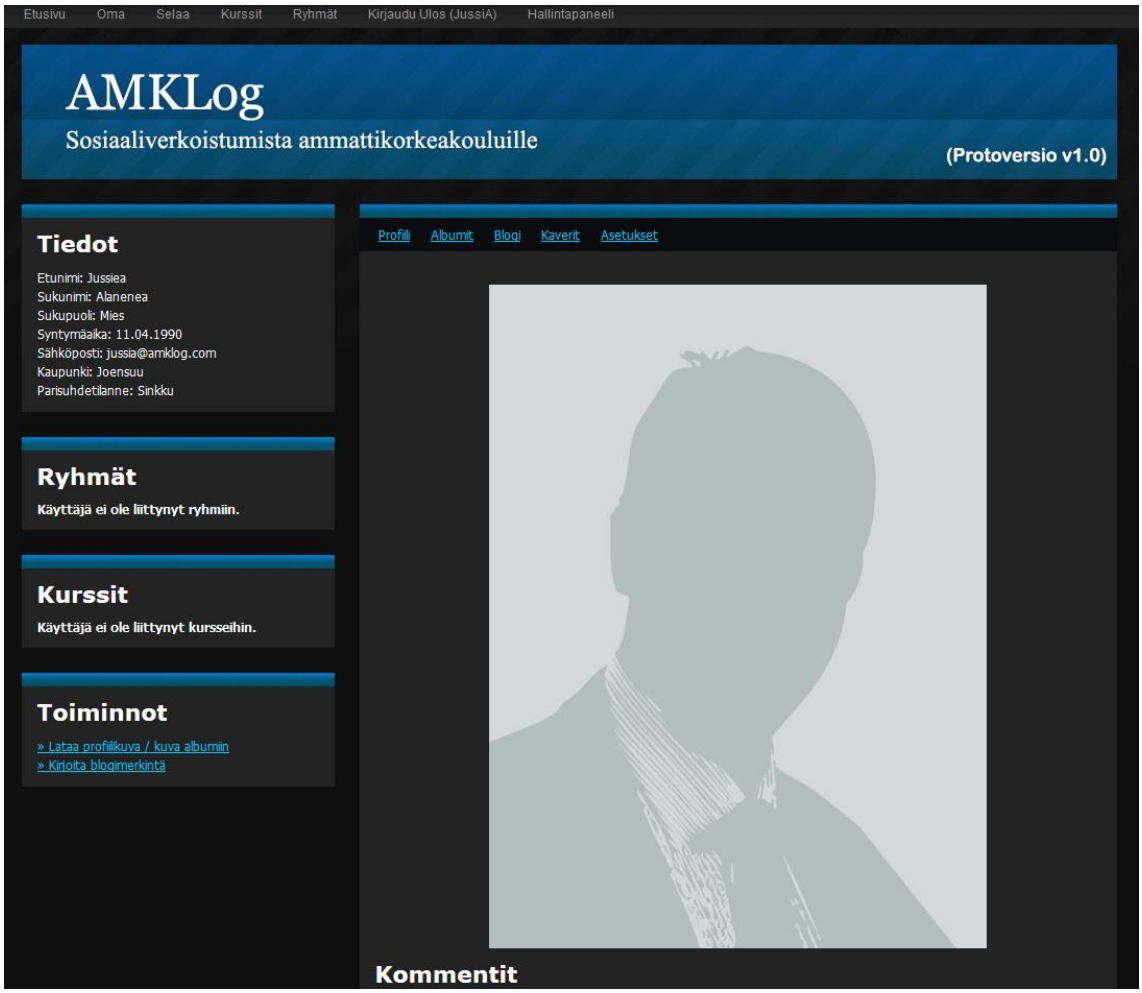


Kaavio a. Ensimmäinen rakennekaavion raakaversio koulun sosiaalisen median palvelusta.

Protosovelluksen kuvakaappaukset



Kuva a. Protosovelluksen etusivu. Vasemmalla on käyttäjän kirjautumisboksi, missä näkyy käyttäjätunnus ja mitä käyttäjä voi halutessa tehdä. Vasemmalla puolella näkyvät TOP-5 ryhmät, TOP-5 kurssit ja oikealla on esittelyteksti sivustolle.



Kuva b. Profiilisivu, käyttäjä voi kommentoida toisen käyttäjän kuvia, tuoda omia kuvia kuva-albumeihin ja kirjoittaa blogeja.