SAVONIA

OPINNÄYTETYÖ - **AMMATTIKORKEAKOULUTUTKINTO**
TEKNIIKAN JA LIIKENTEEN ALA

PROJEKTINSEURANTA-SOVELLUS

TEKIJÄ/T: Joonas Mononen

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Koulutusohjelma Tietotekniikan koulutusohjelma	
Työn tekijä(t) Joonas Mononen	
Työn nimi Projektinseurantasovellus	
Päiväys 7.6.2017	Sivumäärä/Liitteet 27
Ohjaaja(t) Lehtori Keijo Kuosmanen, lehtori Jussi Koistinen	
Toimeksiantaja/Yhteistyökumppani(t) Mikko Pääkkönen, Savonia-AMK	

Tiivistelmä
Opinnäytetyön aiheena oli toteuttaa Savonia-ammattikorkeakoululle projektinseurantatyökalu oppilasprojekteja varten. Opettajilla on ohjauksessa useita oppilasprojekteja yhtäaikaisesti, eikä Savonialla ole juuri tähän tarkoitukseen soveltuvaa seurantatyökalua olemassa. Opettajien toiveiden pohjalta ryhdyttiin suunnittelemaan sovelluksen ominaisuuksia. Työn määrittely oli osin puutteellista, koska vaatimuksista ei vielä oltu täysin perillä projektin alkuvaiheessa. Osana opinnäytetyön tavoitetta olikin kartoittaa ja luoda selkeämpää kuvaa sovelluksen vaatimuksista seuraavaa versiota varten.

Sovellus toteutettiin selainpohjaisena sen monikäyttöisyyden takia. Back-end perustuu Slim 3 mikro -PHP-ohjelmistokehykseen ja front-endissä käytettiin AngularJS- sekä Bootstrap-ohjelmistokehyksiä. Sovelluksen käyttämän dynaamisen aikajanan toteuttamiseen käytettiin Vis.js JavaScript -kirjastoa.

Lopputuloksena saatiin kaikki alkuperäisen määrityksen mukaiset ominaisuudet sisältävä versio, mutta todellista käyttöä varten olisi vaadittu sovelluksen testaamista sekä myöhemmin määriteltyjen ominaisuuksien toteuttamista. Aikaansaadussa versiossa opettajat pystyvät luomaan sovellukseen kurssi-, opinnäytetyö- ja harjoittelukokonaisuuksia ja niihin kuuluvia ryhmiä. Ryhmien aiheet voivat perustua aiemmin sovellukseen lisättyihin projekti-ideoihin. Oppilaat voivat täyttää henkilökohtaisia viikkoraportteja, mikäli heidät on lisätty opiskelijaprojektiryhmään. Nämä oppilaiden viikkoraportit ohjaajat näkevät esitettynä aikajanalla. Opettajat voivat antaa opiskelijoiden viikkoraportteihin palautetta, jonka opiskelija näkee omalla sivullaan.

Avainsanat
AngularJS, Slim

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Information Technology			
Author(s) Joonas Mononen			
Title of Thesis Project Management Application			
Date	7 June 2017	Pages/Appendices	27
Supervisor(s) Mr. Keijo Kuosmanen, Lecturer, Mr. Jussi Koistinen, Lecturer			
Client Organisation /Partners Mikko Pääkkönen, Savonia University of Applied Sciences			

Abstract

The subject of this thesis was to implement a project management software for student projects. Software was made for Savonia University of Applied Sciences. Teachers can have multiple student projects under the supervision and Savonia does not have a project management tool to apply for this use. The features of the software were based on the wishes of the teachers. The specification was partially insufficient because the requirements were not quite clear in the beginning. Because of that, part of the thesis was to research and create a clearer image for the next version.

The software was developed as browser-based software because of its flexibility to run in almost all devices. The back-end is based on the Slim 3 PHP framework and the front-end was implemented with AngularJS and Bootstrap. The dynamic timeline was created with Vis.js JavaScript library.

As a result, every feature that was in the original specification was created, but for the real use there should have been more testing and bug fixing. In this version supervisors can create course, thesis or internship sets and groups attached on them. The project idea of a group can be based on an idea published in the idea section. Students can fill their personal weekly reports and supervisors can give comments on them which students see on their own reporting page.

Keywords
AngularJS, Slim

SISÄLTÖ

KÄSITTEET JA LYHENTEET 6
1 JOHDANTO 7
2 TEKNIIKAT JA TYÖKALUT 8
2.1 Slim 3 8
2.1.1 Toiminta 8
2.1.2 Väliohjelmisto (middleware) 9
2.1.3 Templatet 9
2.2 AngularJS 11
2.3 Bootstrap 12
2.4 Vis.js 12
3 TIETOKANTA 13
3.1 MySQL 13
3.2 phpMyAdmin 13
3.3 Projektinseurantasovelluksen tietokanta 13
4 SOVELLUS 15
4.1 Kirjautuminen 15
4.2 Kurssi- ja ryhmänhallinta 15
4.2.1 Aihetiedot 16
4.2.2 Ryhmät 16
4.3 Ideapankki 17
4.4 Seuranta 18
4.5 Oppilaan näkymät 19
5 SOVELLUKSEN TOTEUTUS 21
5.1 Back-end 22
5.2 Front-end 22
5.3 Näkymät 23
5.4 AngularJS:n ja Slimin yhteiskäyttö 24
6 JATKOKEHITYS 25
6.1 Henkilötiedot 25
6.2 Oppilasnäkymät 25
6.3 Etusivu 25

7 YHTEENVETO 26

LÄHTEET JA TUOTETUT AINEISTOT 27

KÄSITTEET JA LYHENTEET

HTML5	Hyper Text Markup Language, verkkosivujen sisällön ja rakenteen kuvauskieli
CSS	Cascading Style Sheets, www-sivujen ulkoasun määrittävä tyylikieli
JavaScript	Komentosarjakieli, mitä käytetään yleisesti web-ympäristöissä
Tietokanta	Tietovarasto, missä kaikki sovelluksen muuttuva tieto sijaitsee
MySQL	Relaatiotietokantaohjelmisto
Framework	Ohjelmistokehys, mikä muodostaa rungon sovellukselle
jQuery	JavaScript-kirjasto
PHP	Web-palvelinympäristön ohjelmointikieli
Front-end	Verkkoselaimessa ajettava koodi
Back-end	Palvelimella ajettava koodi
AD-tunnus	Windows-toimialueen käyttäjätietokannassa oleva käyttäjätunnus
JSON	JavaScript Object Notation, tiedostomuoto tiedonvälitykseen
AJAX	Asynchronous JavaScript And XML, tekniikka, jolla voidaan tehdä asynkronisia http-pyyntöjä
IIS	Internet Information Services, Microsoftin kehittämä palvelinohjauskokonaisuus
RESTful API	Ohjelmointirajapinta, joka käyttää HTTP-requesteja
DOM	Document Object Model, rakenteisen dokumentin ohjelmointirajapinta

1 JOHDANTO

Opinnäytetyön aiheena on suunnitella ja toteuttaa Savonia-ammattikorkeakoululle selainpohjainen sovellus oppilasprojektien seurantaan. Savonia AMK:ssa tietotekniikan opintoihin sisältyy 1 - 4 kurssia, joissa opiskelijat toteuttavat ryhmissä jonkin ohjelmistoprojektin. Yhdellä opettajalla voi olla useita projektikursseja ohjauksessa samanaikaisesti, joten projektien edistymisen seuranta ja ohjaus voi olla todella työlästä tai jopa mahdotonta ilman jatkuvaa dokumentaatiota.

Tämän opinnäytetyön tekohetkellä projektien seuranta tapahtuu lähinnä Moodleen ladattavien kuukausiraporttien ja projektitunneilla käytävien keskustelujen avulla. Kuukausiraportin ongelma on siinä, ettei se anna aina kuvaa siitä, kuka on tehnyt mitäkin ja mitä on tapahtunut esimerkiksi nykyhetken ja edellisen projektitunnin välillä. Tämän ongelman takia tarkoitus on kehittää sovellus, jonka avulla ohjaajat saisivat nopeasti kuvan projektin edistymisestä ja kunkin jäsenen antamasta työpanoksesta. Opiskelijat voivat täyttää viikkoraportteja, jotka opettajat näkevät visuaalisesti aikajanalla. Sovelluksen pitää mahdollistaa myös ryhmäkohtaisten tapahtumien luonti erilaisten merkkipaalujen lisäämistä varten.

Sovellus toteutetaan selainpohjaisena, millä mahdollistetaan sovelluksen käyttö useilla eri päätelaitteilla. Päätekniikoina ovat HTML5, JavaScript, CSS3 ja PHP. Käytössä on ohjelmistokehykset Slim 3, AngularJS ja Bootstrap.

2 TEKNIIKAT JA TYÖKALUT

Sovelluksen toteutustekniikoita valittaessa keskeistä oli uuden oppiminen sekä positiiviset kokemukset aiemmista projekteista. Sovellus toteutettiin selainpohjaisena, mikä tekee siitä laajasti käytettävän useilla eri päätelaitteilla. Back-end kirjoitettiin PHP:llä hyödyntäen Slim-ohjelmistokehystä, joka valikoitui käyttöön sen suhteellisen loivan oppimiskäyrän takia. Front-end-toteutukseen käytettiin kieliä HTML5, CSS3 ja JavaScript, lisäksi hyödynnettiin JavaScript-kirjastoja ja ohjelmistokehyksiä, joista tärkeimpänä AngularJS. Tietokantaohjelmistona käytettiin MySQL-tietokantaa.

2.1 Slim 3

Slim on kevyt ja mahdollisimman yksinkertaiseksi tehty PHP-ohjelmistokehys, joka sopii erityisen hyvin pienehköihin projekteihin. Slimillä voidaan toteuttaa joko rajapintoja tai kokonaisia web-sovelluksia. Lyhyesti kuvailtuna Slim ottaa vastaan HTTP-pyynnön, prosessoi sen ja suorittaa halutut toimenpiteet, minkä jälkeen se palauttaa HTTP-vastauksen. Tarvittaessa Slimin ominaisuuksia voidaan laajentaa PHP-komponenteilla. Komponentteja voidaan asentaa esimerkiksi Composer riippuvuushallintatyökalulla. (Slim Framework Team, n.d.)

2.1.1 Toiminta

Slim tarvitsee toimiakseen web-palvelimen kuten Apachen. Web-palvelin konfiguroidaan lähettämään kaikki kyseiseen hakemistoon kohdistuvat pyynnöt – osoitteesta huolimatta – yhteen PHP-tiedostoon, mikä käynnistää Slim-sovelluksen. Näin kaikki mitä pyynnön osoitteessa lukee varsinaisen verkko-osoitteen jälkeen, jää Slimin käsiteltäväksi. Slim parsii osoitteesta sen osat ja näiden osien avulla sovellukseen voidaan määrittää koodissa reitit, jotka sitten suorittavat halutun toimenpiteen. (Slim Framework Team, n.d.) Kuvassa 1 määritetään GET-metodilla toimiva reitti /hello/, mikä ottaa vastaan yhden parametrin osoitteen mukana. Tätä reittiä voitaisiin kutsua kutsua esimerkiksi osoitteella ”www.slimapp.com/hello/matti”, jolloin vastauksena saataisiin viesti ”Hello matti”.

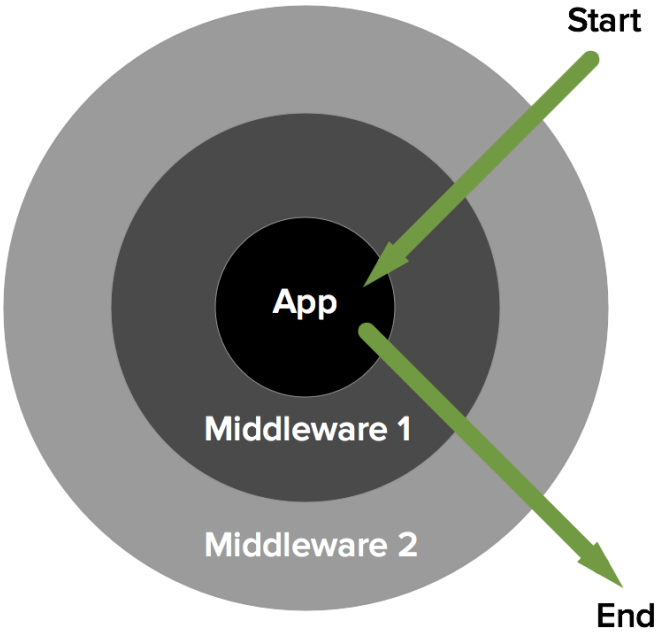
```php
<?php
// Create and configure Slim app
$config = ['settings' => [
    'addContentLengthHeader' => false,
]];
$app = new \Slim\App($config);

// Define app routes
$app->get('/hello/{name}', function ($request, $response, $args) {
    return $response->write("Hello " . $args['name']);
});

// Run app
$app->run();
```

KUVA 1 Yksinkertainen Slim-sovellus (Slim Framework Team, n.d.)

Jokainen määritetty reitti ottaa vastaan Request- ja Response-objektit. Nämä objektit noudattavat PSR7 rajapintaa (interface). PSR7 (PHP Stantards Recommendation) on HTTP-viesteille kehitetty stantardi. Slimissä jokaisen reitin täytyy aina palauttaa jokin PSR-7:n rajapintaan perustuva objekti. (O'Phinney, 2015)

2.1.2 Väliohjelmisto (middleware)

Väliohjelmistoksi kutsutaan koodia, joka suoritetaan ennen ja jälkeen varsinaisen pyynnön käsittelyn. Väliohjelmisto sopii Slimissä käytettynä esimerkiksi käyttäjän tunnistamiseen, minkä jälkeen voidaan tarkistaa, onko hänellä oikeus kyseisen sivun katsomiseen. Jokainen lisätty väliohjelmisto sijoittuu edellisen väliohjelmiston ympärille (kuva 2). Kun Slim-sovellus suoritetaan, kulkee suoritusjärjestys ulkoa sisälle päin ja sisältä takaisin päinvastaisessa järjestyksessä. (Slim Framework Team, n.d.)



KUVA 2 Middlewaren toimintaperiaate (Slim Framework Team, n.d.)

2.1.3 Templatet

Twig-view komponentilla voidaan hyödyntää Slimissä templateja. Templatet ovat eräänlaisia HTML-sivupohjia, joihin varsinainen muuttuva sisältö kirjoitetaan sille merkitylle paikalle. Kuvassa 3 rekisteröidään jo asennettu Twig-view komponentti käyttöön.

```php
<?php
// Create app
$app = new \Slim\App();

// Get container
$container = $app->getContainer();

// Register component on container
$container['view'] = function ($container) {
    $view = new \Slim\Views\Twig('path/to/templates', [
        'cache' => 'path/to/cache'
    ]);

    // Instantiate and add Slim specific extension
    $basePath = rtrim(str_ireplace('index.php', '', $container['request']->getUri(
    $view->addExtension(new Slim\Views\TwigExtension($container['router'], $basePa

    return $view;
};
```

KUVA 3 Komponentin rekisteröinti sovellukseen (Slim Framework Team, n.d.)

Nyt reitissä ”/hello” (kuva 4) voidaan palauttaa pelkän viestin sijaan näkymä ”profile.html”. Tähän näkymään voidaan syöttää tietoa, kuten nimi, joka on saapunut osoitteen mukana. Näkymään syötettävä data voisi olla peräisin mistä tahansa, esimerkiksi sovelluksen omasta tietokannasta. (Slim Framework Team, n.d.)

```php
// Render Twig template in route
$app->get('/hello/{name}', function ($request, $response, $args) {
    return $this->view->render($response, 'profile.html', [
        'name' => $args['name']
    ]);
})->setName('profile');

// Run app
$app->run();
```

KUVA 4 profile.html sivun sisällyttäminen HTTP-vastaukseen (Slim Framework Team, n.d.)

2.2 AngularJS

AngularJS on Googlen ylläpitämä MVC-ajatukseen nojautuva JavaScript-pohjainen ohjelmistokehys, jolla voidaan luoda vuorovaikutteisia web-sovelluksia. Angularin ensimmäinen versio julkaistiin vuonna 2010 ja huomattavasti ensimmäisestä versiosta poikkeava versio 2.0 julkaistiin vuonna 2016 (Wikipedia, 2017). Alla oleva teoria on AngularJS:n ensimmäisestä versiosta ja kattaa lähinnä tässä sovelluksessa käytetyn osion AngularJS:stä.

AngularJS käyttää HTML:ää kuvauskielenä laajentaen HTML:n omaa syntaksia, jolloin AngularJS kehitys on selkeää ja yksinkertaista. Kaikki Angularin koodi suoritetaan selaimessa, joten se ei ole sidottu mihinkään back-end ratkaisuun. HTML:ää laajentavia attribuutteja kutsutaan direktiiveiksi (directive), joita ovat mm. tupla-aaltosulkeet ({{ }}) , joita käytetään datan kiinnittämiseen näkymään, toistot (ng-repeat), elementin esittäminen (ng-show), elementin piilottaminen (ng-hide), datan sitominen HTML-kontrolliin (ng-model). Direktiivit ei rajoitu vain olemassa oleviin vaan niitä voi luoda myös itse directive-funktiolla (kuva 5). (w3schools, n.d.)

```
<body ng-app="myApp">

<w3-test-directive></w3-test-directive>

<script>
var app = angular.module("myApp", []);
app.directive("w3TestDirective", function() {
    return {
        template : "<h1>Made by a directive!</h1>"
    };
});
</script>

</body>
```

KUVA 5 Oman direktiivin luominen (w3schools, n.d.)

Yksi AngularJS:n keskeisistä asioista on kontrolleri (controller), joka nimensä mukaisesti kontrolloi AngularJS sovelluksen datavirtaa. Kontrolleri liitetään DOM:iin ng-controller -direktiivillä. Kontrolleri on JavaScript objekti, joka koostuu lähinnä scopen attribuuteista ja funktioista, riippuen sovelluksen toiminallisuudesta. Kaikki osaksi scopea liitetyt attribuutit ja funktiot ovat myös käytettävissä käyttöliittymästä. Scope on objekti, joka voidaan mieltää liimaksi sovelluksen käyttöliittymän ja kontrollerin välillä. Kun jokin scopeen liitetty attribuutti muuttuu, tulee direktiiveille siitä ilmoitus, ja DOMiin päivittyy automaattisesti näkyviin uusi arvo. (Google, n.d.)

2.3 Bootstrap

Bootstrap on ilmainen avoimen lähdekoodin HTML, CSS ja JavaScript ohjelmistokehys responsiivisten web-sivujen toteuttamiseen (Wikipedia, 2017). Responssiivisella toteutuksella sivusto saadaan skaalautumaan erikokoisille näytöille, joten sivuston ulkoasua ei tarvitse erikseen räätälöidä mobiililaitteille. Bootstrap sisältää paljon valmiita ulkoasumäärityksiä, joten silmää miellyttävän ulkoasun tekeminen ei vaadi suuria ponnisteluja. Negatiivisena puolena ilman omia ulkoasumäärityksiä, on suhteellisen geneerinen ulkoasu, mikä ei varsinkaan omaperäistä ilmettä hakeviin sivustoihin sovi.

Sivun rakenne jaetaan riveihin ja sarakkeisiin (kuva 6). Sarakkeet yhdistyvät halutun määrityksen perusteella tai siirtyvät omille riveilleen näyttökoon muuttuessa. Näin sivustolla ei lähtökohtaisesti tarvita sivusuuntaista vieritystä laisinkaan. Muita Bootstrapin ominaisuuksia, kuten valmiiksi tyylitellyjä taulukkoja, nappeja ja listoja käytetään lisäämällä HTML-elementteihin luokkia.

```
<div class="row">
  <div class="col-xs-9 col-md-7">.col-xs-9 .col-md-7</div>
  <div class="col-xs-3 col-md-5">.col-xs-3 .col-md-5</div>
</div>

<div class="row">
  <div class="col-xs-6 col-md-10">.col-xs-6 .col-md-10</div>
  <div class="col-xs-6 col-md-2">.col-xs-6 .col-md-2</div>
</div>

<div class="row">
  <div class="col-xs-6">.col-xs-6</div>
  <div class="col-xs-6">.col-xs-6</div>
</div>
```

KUVA 6 Bootstrapilla toteutettu responsiivinen sivustorakenne (w3schools, n.d.)

AngularJS:n kanssa Bootstrappia voidaan käyttäjä Bootstrap UI:n avulla, missä kaikki Bootstrap-komponentit on kirjoitettu AngularJS direktiiveinä, jolloin pelkkä Bootstrapin CSS-tiedoston sisällyttäminen ohjelmaan riittää. (ui-bootstrap contributors, n.d.)

2.4 Vis.js

Vis.js on Almenden kehittämä avoimen lähdekoodin JavaScript-kirjasto, jolla voidaan luoda monipuolisesti interaktiivista grafiikkaa selaimessa esitettäväksi. Tässä projektissa hyödynnettiin Vis.js:ää aikajanaan, jossa esitetään viikkoraportteja ja tapahtumia. Vis.js tukee dynaamista kohteiden lisäämistä, muokkaamista ja poistamista. Aikajana on täysin kustomoitavissa CSS-tyyliohjeilla. Muita Vis.js:n tukemia ominaisuuksia ovat verkosto, sekä erilaiset 2d- ja 3d-kaaviot. (Almende B.V., 2015) Useita eri kirjastoja valikoidessa Vis.js soveltui parhaiten tähän tarkoitukseen monipuolisuuden, ilmaisuuden ja suhteellisen helpon muokattavuuden takia.

3 TIETOKANTA

Keskeinen osa sovellusta on sen tietokanta, mihin kaikki muuttuva tieto säilötään. Tässä luvussa esitellään projektissa toteutettu tietokanta ja käytetyt tekniikat.

3.1 MySQL

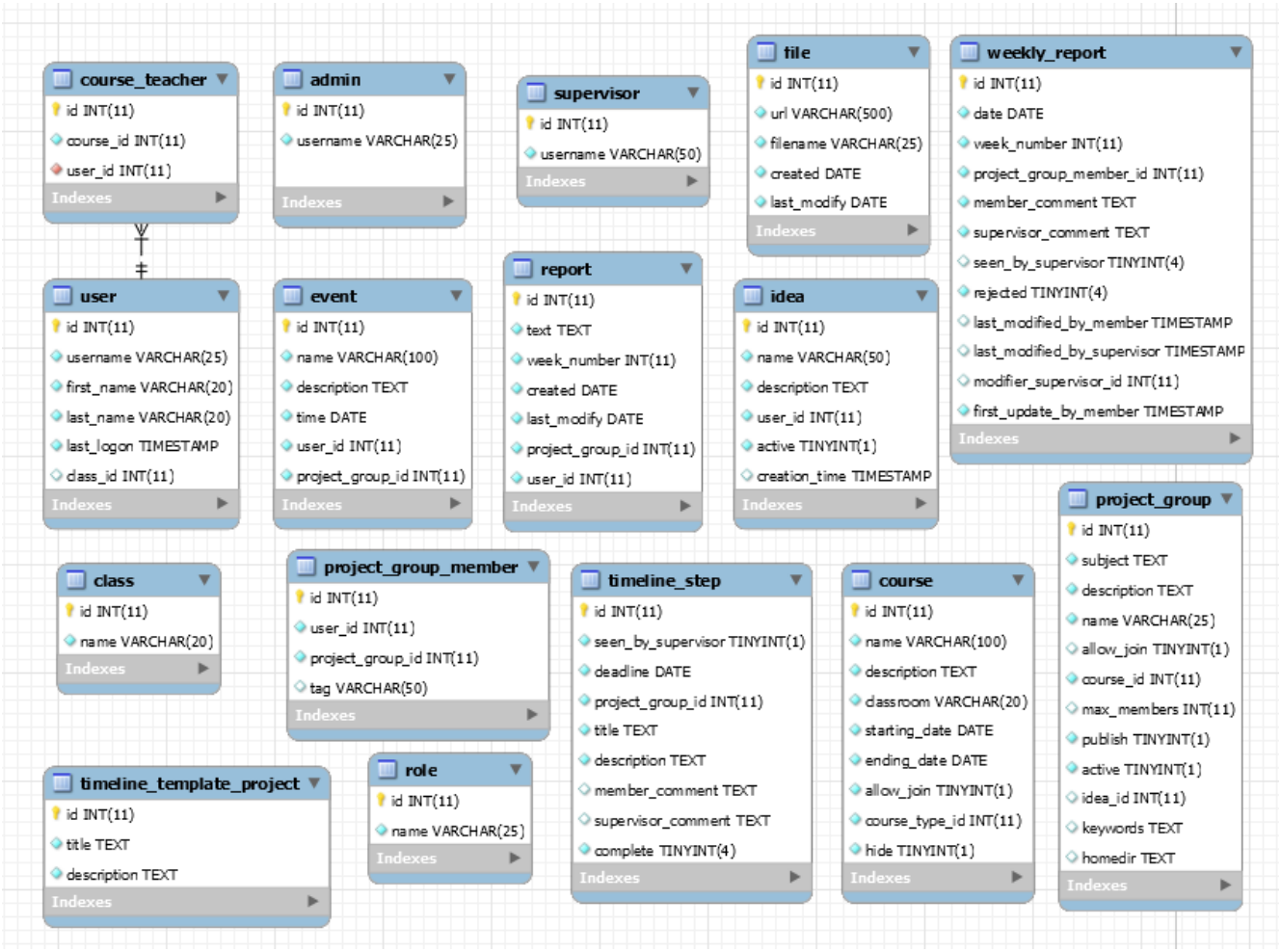
MySQL on Oracle Corporationin kehittämä, jakama ja tukema avoimen lähdekoodin relaatiotietokantaohjelmisto. Relaatiotietokannassa tieto on jaettu tauluihin, joiden välille voidaan luoda erilaisia relaatioita eli yhteyksiä. MySQL on suosittu etenkin web-sovellusten tietokantana, jota käyttävät mm. Facebook, Twitter ja YouTube. (Oracle Corporation, 2017)

3.2 phpMyAdmin

phpMyAdmin on selainpohjainen PHP:llä kirjoitettu MySQL-tietokannan hallintatyökalu. phpMyAdminin avulla voidaan luoda tietokantoja, tauluja, taulujen sarakkeita, relaatioita, indeksejä, käyttäjiä ja käyttöoikeuksia. Nämä kaikki toiminnot ovat saatavilla graafisen käyttöliittymän kautta, joten varsinaisia SQL-komentoja ei tarvitse käyttää, joskin sekin on mahdollista. (phpMyAdmin contributors, 2017)

3.3 Projektinseurantasovelluksen tietokanta

Tietokannasta luotiin alustava suunnitelma millainen sen rakenteen tulisi olla. Projektin ja määrittelyn etenemisen myötä kantaan lisäiltiin tarvittavia tauluja ja sarakkeita. Tietokanta ei tullut täysin valmiiksi, sillä siitä puuttuu lähes kaikki relaatiot, mitä tarvitaan tietokannan luotettavuuden ylläpitämiseen. Relaatiot estävät, ettei tietokanta sisällä rivejä, joissa viitataan toiseen olemattomaan riviin. Lisäksi erilaiset poistot voivat tapahtua relaatioiden kautta automaattisesti, joten näitä ei tarvitse huomioida omassa SQL-lausekkeessa. Kuvassa 7 on esitetty sovelluksen käyttämän tietokannan rakenne. Tietokannassa on osittain huomioitu myös vielä toteuttamattomia ominaisuuksia, kuten tiedoston tallennus.

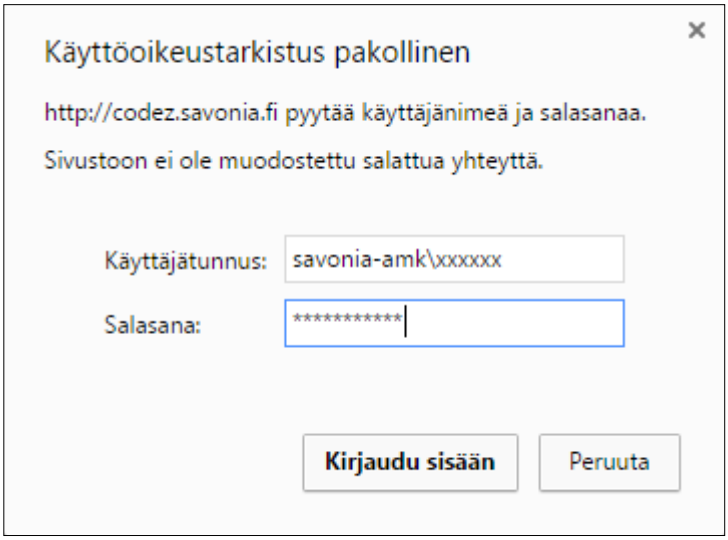


KUVA 7 Projektiseurantasovelluksen tietokantarakenne (Joonas Mononen, 2017)

4 SOVELLUS

4.1 Kirjautuminen

Koska sovelluksen käyttäjämäärä voi olla suuri sekä pääsy ulkopuolisilta haluttiin estää, päätettiin kirjautumisessa hyödyntää Savonian AD-tunnuksia. Näin vältytään kokonaan käyttäjätunnusten ylläpitämisen vaivalta. Kirjautuminen on toteutettu IIS:n windows authentication -menetelmällä, eikä sovellus itsessään näin hallinnoi käyttäjätietoja. Sivustolle mentäessä ensin avautuu kirjautumisikkuna, mihin käyttäjä syöttää Savonia ympäristön tunnukset (kuva 8).



KUVA 8 Kirjautuminen sovellukseen (Joonas Mononen, 2017)

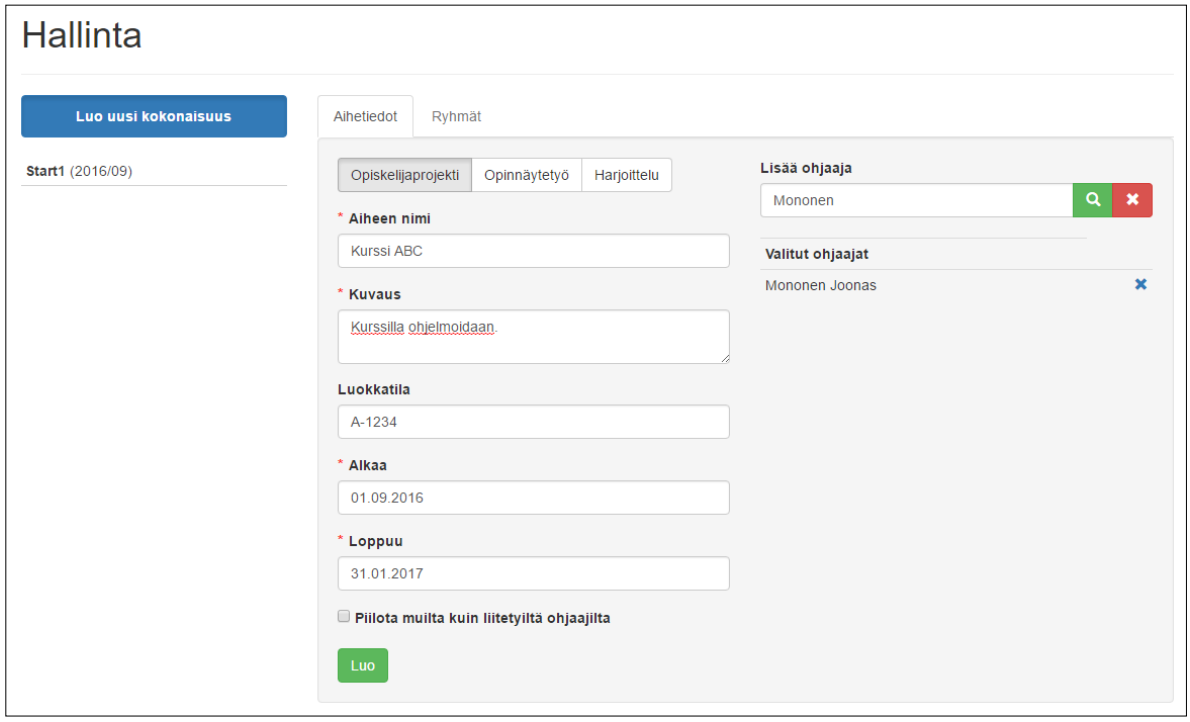
Kirjautuminen antaa sovelluksen tietoon vain kirjautuneen käyttäjän käyttäjätunnuksen sekä verkkotunnuksen (domain). Kirjautumisen yhteydessä tarkistetaan, löytyykö käyttäjän käyttäjätunnus sovelluksen omasta tietokannasta, josta haetaan käyttäjän rooli ja tallennetaan se istuntoon (session). Roolin perusteella käyttäjälle saadaan esitettyä oikeat näkymät ja rajattua käyttöoikeudet. Mikäli käyttäjä ei ole aiemmin kirjautunut sovellukseen, pyydetään kirjautumisen yhteydessä syöttämään henkilötiedot. Uudet käyttäjät asettuvat oletuksena opiskelijarooliin, josta tietokannan hallintaoikeuden omistavat henkilöt voivat korottaa heidät ohjaajiksi.

4.2 Kurssi- ja ryhmänhallinta

Kurssi- ja ryhmähallinta -sivun näkevät vain käyttäjät joilla on ohjaajarooli. Tämän sivun kautta ohjaajat luovat kursseja tai muita opiskelijaryhmiä tai yksilöitä sisältäviä kokonaisuuksia. Kaikki sivustolla tapahtuvat luomiset, päivitykset, poistot ja haut tapahtuvat AJAX-kutsuilla, joiden on tarkoitus tehdä sivustosta sulavakäyttöisempi, kun vain tarvittava tieto ladataan palvelimelta uudestaan.

4.2.1 Aihetiedot

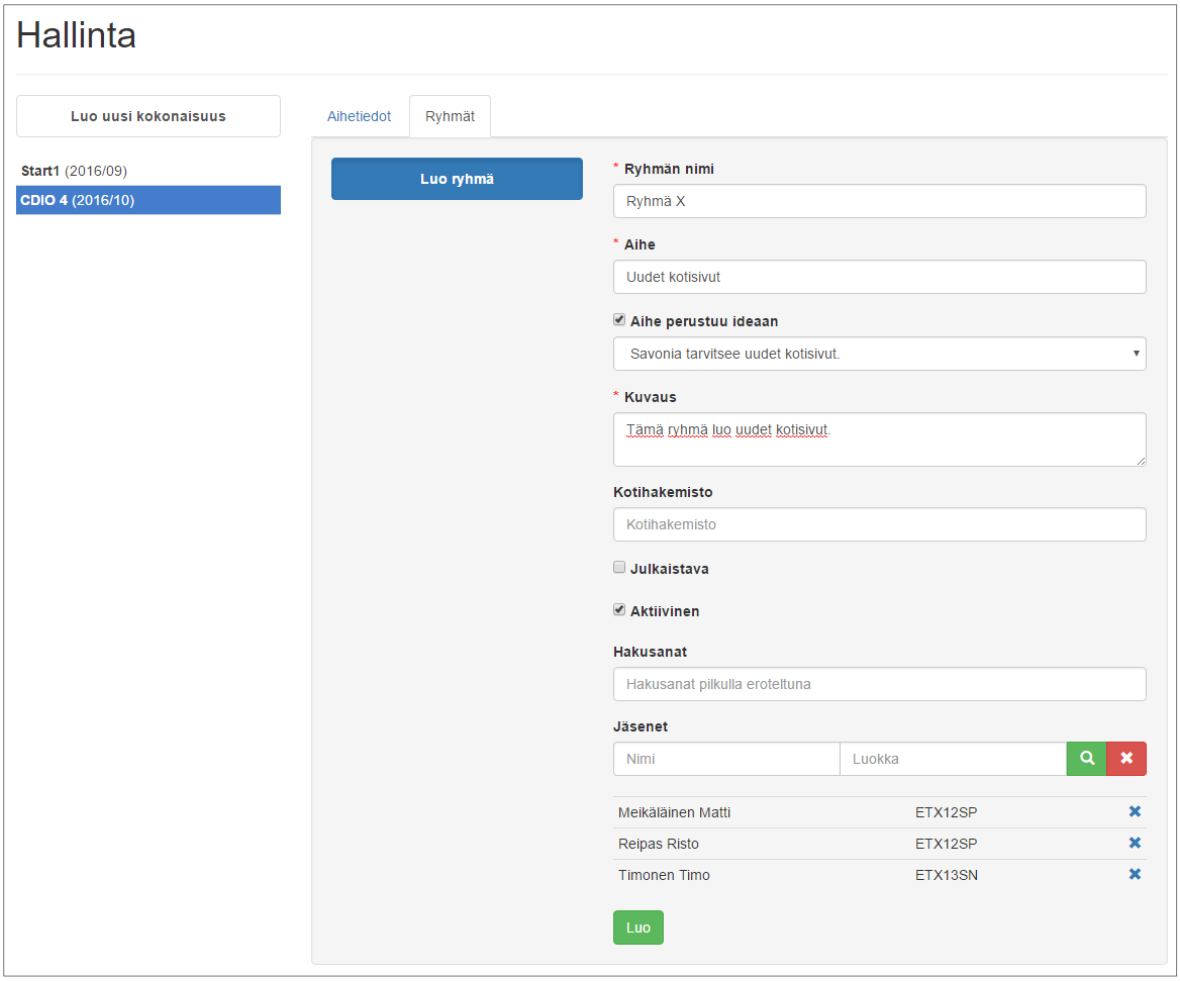
Kaikki kokonaisuuksien ja ryhmien luonnit, poistot ja päivitykset tapahtuvat hallintapaneelin kautta (kuva 9). Aihetyyppejä on kolme: opiskelijaprojekti, harjoittelu ja opinnäytetyö. Ohjaaja täydentää aihekokonaisuuden kuvauksen, ajankohdan ja ohjaajat. Ohjaajalla on mahdollisuus piilottaa kokonaisuus näkyvistä hallintapaneelissa muilta kuin liitetyiltä ohjaajilta ja täten estää tämän muokkaaminen ja tarkastelu ulkopuolisilta.



KUVA 9 Aihetietojen ylläpito (Joonas Mononen, 2017)

4.2.2 Ryhmät

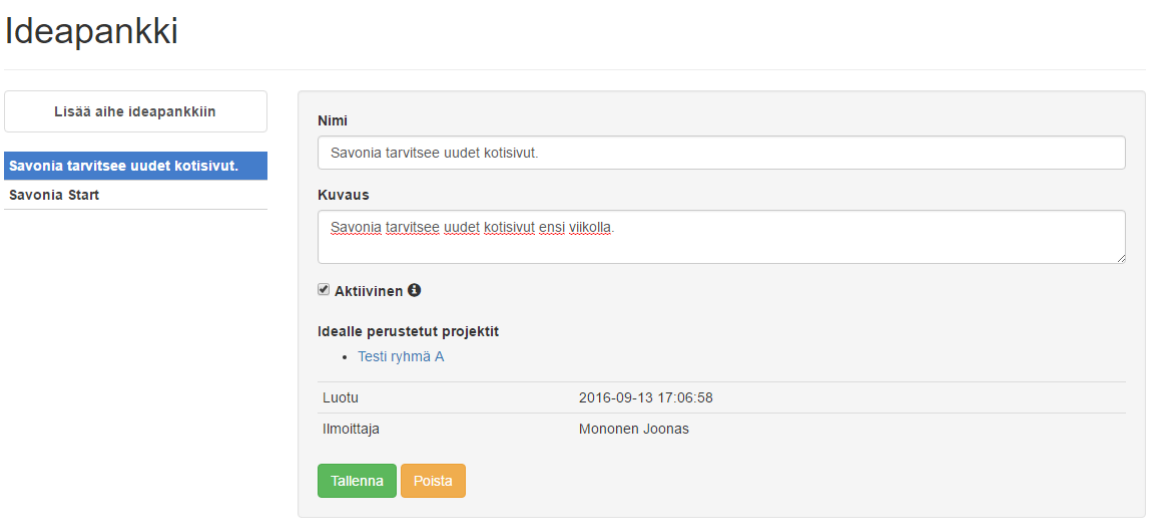
Kokonaisuuden luonnin jälkeen käyttäjälle avautuu mahdollisuus ryhmät-välilehteen (kuva 10). Jokaiseen kokonaisuuteen voidaan luoda rajoittamaton määrä ryhmiä. Ryhmälle täytetään ryhmää sekä ryhmän aihetta koskevat tiedot. Ohjaaja voi merkitä ryhmän aiheen perustuvan ideaan, joka on lisätty ideapankki-sivun kautta. Jäsenet voidaan hakea ryhmään nimen ja/tai luokkatunnuksen perusteella. Mikäli aihekokonaisuuden tyypiksi on valittu opiskelijaprojekti, generoituu ryhmän luontivaiheessa jokaiselle jäsenelle kurssin keston ajalle viikkoraporttipohjat. Jäsenien lisäys ja poisto luonnin jälkeen on mahdollista, mutta opiskelijaprojektien tapauksessa viikkoraporttipohjat jäävät generoitumatta.



KUVA 10 Ryhmien ylläpito (Joonas Mononen, 2017)

4.3 Ideapankki

Ideapankki-sivulla (kuva 11) kaikki ohjaajat voivat lisätä ideoita, joita voitaisiin käyttää oppilasprojekteissa. Nämä ideat näytetään kurssi- ja ryhmänhallinta sivustolla ryhmää hallittaessa. Mikäli idealle on perustettu ryhmä, tämä näkyy listattuna idean kohdalla. Vanhentuneet ideat voidaan poistaa käytöstä poistamalla aktiivinen-valintaruutu.

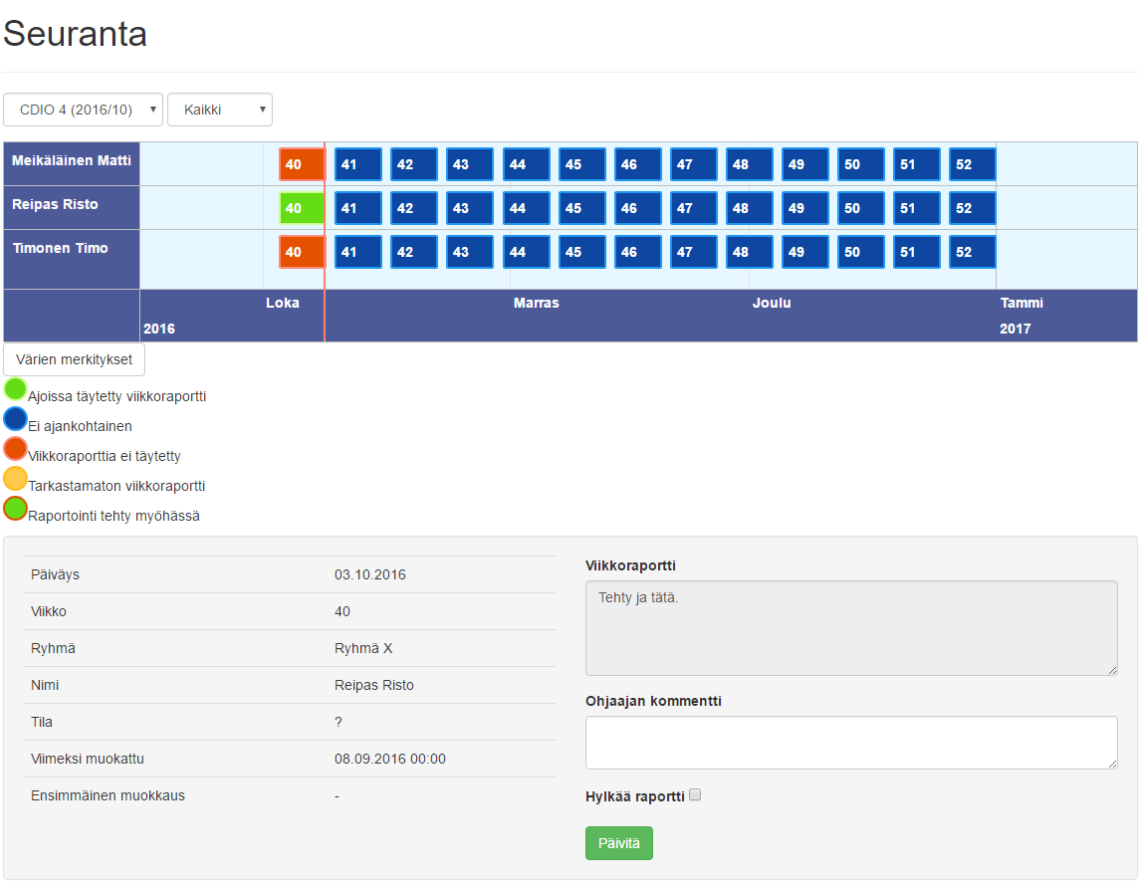


KUVA 11 Ideapankki (Joonas Mononen, 2017)

4.4 Seuranta

Ohjaajien käytössä olevalta seuranta-sivulta (kuva 12) löytyvät kaikki raportit ja tapahtumat. Käyttäjä valitsee pudotusvalikosta kurssin tai muun luodun kokonaisuuden, jolloin alla olevalle aikajanalle tulee näkyviin ryhmien tapahtumat. Ryhmille voidaan luoda uusia tapahtumia yksitellen tai kaikille kerralla. Ohjaaja näkee tapahtuman tiedot, ryhmän kommentin ja ajankohdat. Mikäli kyseessä on opiskelijaprojekti, on luotujen ryhmien jäsenille muodostunut viikkoraporttipohjat. Kun ryhmän valinta -pudotusvalikosta valitsee ryhmän, tulee aikajanalle näkyviin viikkoraportit ryhmäjäsenittäin. Valitsemalla viikkoraportin aikajanalta päivittyy sitä koskevat tiedot näkyviin alla olevaan lomakkeeseen. Aikajana on skaalattava eli siihen saa kerralla näkyviin hyvin laajan tai lyhyen ajanjakson.

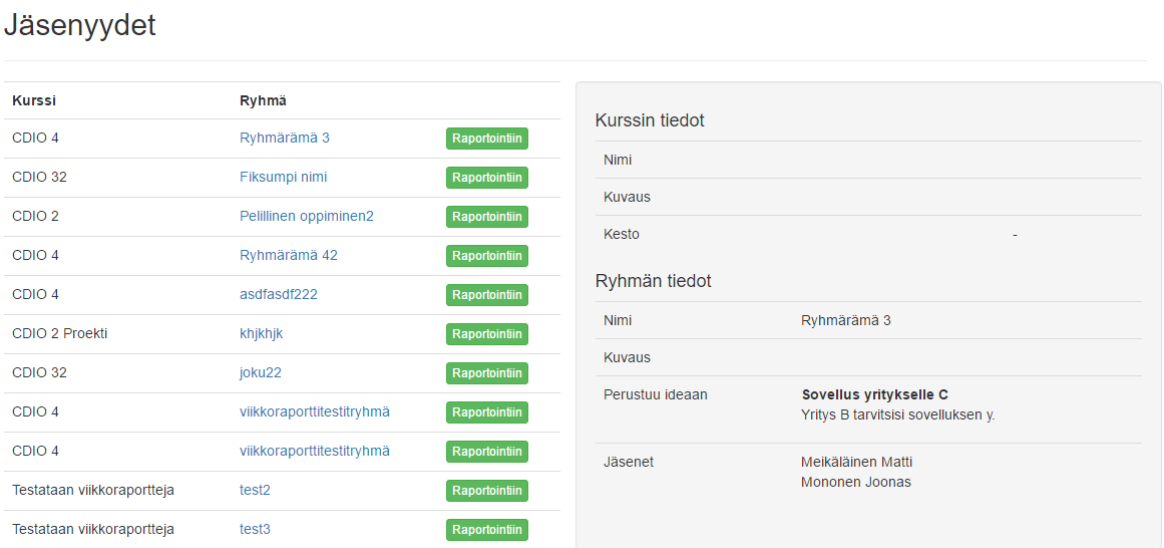
Kaikki tapahtumat ja viikkoraportit, jotka sisältävät uuden opiskelijan kommentin esiintyvät oranssina aikajanalla. Kun ohjaaja on valinnut kommentin, se merkitään tarkistetuksi ja viikkoraportti saa vihreän tai punareunaisen vihreän värin, riippuen onko merkintä tehty myöhässä. Tapahtumat ja raportit on mahdollista hylätä ohjaajan toimesta, jolloin ne merkitään punaisella. Sinisellä merkityt raportit ovat tulevien viikkojen raportteja. Näin kurssin lopussa jokaiselle opiskelijalle on muodostunut rivi väreillä merkittyjä raportteja, joka kertoo nopealla vilkaisulla jäsenen aktiivisuudesta ja suoriutumisesta projektissa.



KUVA 12 Ryhmän viikkoraporttien tarkastelu (Joonas Mononen, 2017)

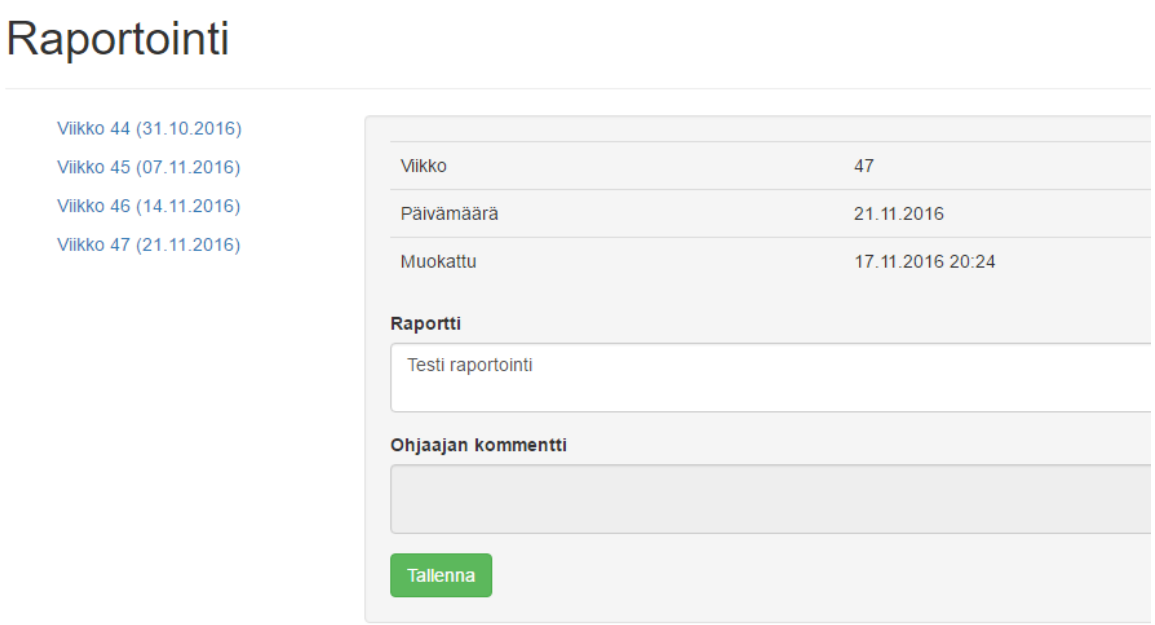
4.5 Oppilaan näkymät

Oppilailla on kaksi sivua, minne navigoida; koti ja jäsenyydet. Koti-sivulla ei näy sisältöä, mutta se on luotu tulevaisuutta varten, jolloin siihen voidaan sisällyttää esimerkiksi ajankohtaiset täytettävät raportit. Jäsenyydet-sivulla (kuva 13) käyttäjä näkee kaikki ryhmät, joiden jäseneksi hänet on liitetty. Ryhmän nimeä painamalla käyttäjä voi tarkastella ryhmän tietoja viereisessä olevassa lomakkeessa. Raportointiin päästän painamalla ryhmän vierestä raportointiin-painiketta.



KUVA 13 Jäsenyydet (Joonas Mononen, 2017)

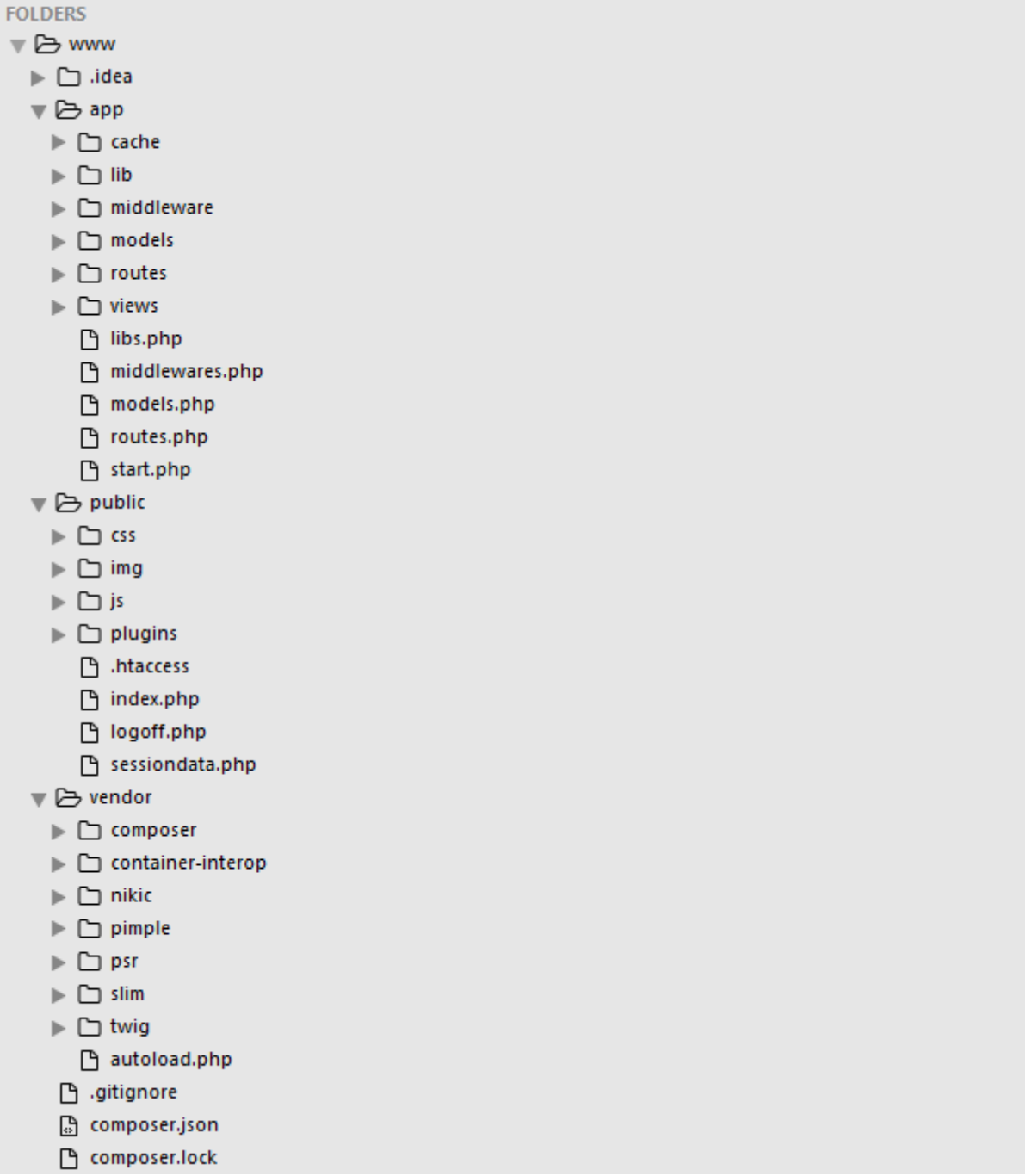
Oppilas näkee valitun projektin viikkoraportit kuluvalle viikolle asti. Kaikkiin näkyviin raportteihin voidaan lisätä tekstiä, mutta mikäli oppilas täyttää yli viikon vanhan raportin, luokitellaan se myöhästyneeksi opettajan näkymässä. Mikäli opettaja on antanut raportille kommentin, näkyy se myös lomakkeella. Kuvassa 14 on esitetty opiskelijan raportointinäkymä.



KUVA 14 Viikkoraporttien täyttö (Joonas Mononen, 2017)

5 SOVELLUKSEN TOTEUTUS

Slim-applikaation rakenne voidaan toteuttaa lukemattomilla eri tavoilla. Tässä projektissa on käytetty erästä GitHubista löydettyä projektipohjaa, joka muokkaantui omanlaiseksi kehityksen aikana. Projektin kansiorakenne on esitetty kuvassa 15. Sovellus sisältää kaksi päähakemistoa, app ja public. App-kansiossa sijaitsee sovelluksen back-end eli palvelimella suoritettava osuus ohjelmasta. Public-kansiossa sen sijaan sijaitsee front-end eli selaimessa ajettava osuus. Front-end koostuu pääsääntöisesti AngularJS:ää toteuttavista JavaScript-tiedostoista sekä käytetyistä JavaScript-kirjastoista.



KUVA 15 Projektin kansiorakenne (Joonas Mononen, 2017)

5.1 Back-end

Sovelluksen käynnistyessä ensin täytyy kirjautua IIS:n Windows Authentication´in kautta (kuva 9) Savonia-amk:n Windows-ympäristön tunnuksilla, mikä tallettaa käyttäjän käyttäjänimen PHP:n globaaliin muuttujaan $_SERVER. Sovelluksen suorittaminen lähtee liikkeelle index.php-scriptistä, mikä alustaa sovelluksen kutsumalla start.php-scriptiä, minkä jälkeen itse toimintaprosessi käynnistyy. Sovellukseen on lisätty kaksi middlewarea, mitkä suoritetaan aina ennen kuin varsinaista polkuun sidottua prosessia suoritetaan. Nämä ovat CheckLogin.php ja PageRights.php. CheckLogin.php tarkistaa, että käyttäjä on rekisteröitynyt sovellukseen. Mikäli käyttäjää ei löydy tietokannasta, ohjautuu sovellus rekisteröintisivulle, missä käyttäjä täyttää käyttäjänimeen liitettävät henkilötietonsa. PageRights.php tarkistaa käyttäjän rooliin perustuen, onko käyttäjällä oikeus päästä kohteena olevalle sivulle. Mikäli käyttöoikeutta ei ole, käyttäjä ohjataan sovelluksen aloitusnäkymään.

App\Routes-kansion alle määritetään reitit ja niitä vastaavat toiminnot. Reitti palauttaa joko näkymän eli HTML5-tiedoston tai JSON-dataa. Kuvassa 16 palautetaan näkymä tracking_teacher.php, johon on liitetty kirjautuneen ohjaajan kurssit. Reitit käyttävät tiedonhakuun models-kansiossa sijaitsevia luokkia, jotka hoitavat tietokantakyselyt ja muut tarvittavat tiedon parsimiset (kuva 17). Tietokannan käsittely tehdään PHP:n PDO:n (PHP Data Objects) avulla, mikä huolehtii mm. SQL:n turvallisuudesta.

```php
$app->get('/tracking', function ($request, $response, $args) {

    $oCourse = new \models\Course();
    $userId = $_SESSION['USERID'];
    $myCourses = $oCourse->getCourseNames($userId);

    $params = ['courses' => $myCourses];

    return $this->view->render($response, 'tracking_teacher.php', $params);

})->setName('tracking');
```

KUVA 16 Seuranta-sivun reitti (Joonas Mononen, 2017)

```php
public function getCourseNames($userId) {

    $sql = "SELECT name, id, CONCAT(' (', DATE_FORMAT(starting_date, '%Y/%m'), ')') as 'date', hide FROM course
            WHERE hide = 0 OR hide = 1 AND id IN (SELECT course_id FROM course_teacher WHERE user_id =:user_id)";
    $stmt = $this->core->dbh->prepare($sql);

    $stmt->bindParam(':user_id', $userId, PDO::PARAM_INT);

    if ($stmt->execute()) {
        $return = $stmt->fetchAll();
    } else {
        $return = 0;
    }

    return $return;
}
```

KUVA 17 Course-luokan getCourseNames-metodi (Joonas Mononen, 2017)

5.2 Front-end

Public-kansion alla olevassa js-kansiossa sijaitsee AngularJS-applikaatiot ja muut JavaScript-kirjastot. Jokaiselle sovelluksessa olevalle toiminnallisuutta sisältävälle sivulle on luotu oma moduuli, mikä sisältää koko sivun toiminallisuuden. Yleisesti käytössä olevista funktioista luotiin oma luokka, mitä voidaan kutsua tarvittaessa kaikkialta.

AngularJS:n moduulit koostuvat controllerista, mikä pitää sisällään funktioita, joista osasta on tehty $scope:n ominaisuuksia (property). Esimerkiksi kuvassa 18 on käyttöliittymästä kutsuttavissa oleva funktio deleteCourse. Controlleriin voidaan liittää palveluja (service), jotka hallitsevat tietyn osa-alueen. Tärkeä sovelluksessa käytetty palvelu on $http, joka tekee AJAX:in avulla pyynnön (request) palvelimelle ja palauttaa vastauksen (response). Kaikki sovelluksessa AngularJS:n avulla tehdyt pyynnöt noudattavat samaa kaavaa. Määritetään mihin osoitteeseen kysely tehdään, sen sisältämä data ja headerit. Kun palvelin vastaa – palauttaen yleensä jotain dataa – määritellään tehtävät toiminnot, kuten latausindikaattorien piilottaminen.

```
$scope.deleteCourse = function(id, name) {
    courseHttpStarted();
    $scope.courseLoaderSpinner = true;

    if (confirm('Poistetaanko kurssi ' + name + '?')) {

        var postData = { 'id': id }

        $http({
            method: 'POST',
            url: 'course/delete',
            headers: {'Content-Type': "application/json"},
            data: postData

        }).success(function (data, status) {

            if (data['status'] == 'ok') {
                getCourses();
                $scope.courseMsg = 'Kurssi poistettu';
            }

        }).finally(function () {

            newCourse();
            courseHttpDone();
            $scope.courseLoaderSpinner = false;
        });
    }
}
```

KUVA 18 Scopen deleteCourse -funktio (Joonas Mononen, 2017)

5.3 Näkymät

Näkymät ovat näkyvä osuus sivusta, mikä palautetaan käyttäjälle sivulatauksen yhteydessä. Näkymät ovat HTML5:en, AngularJS:n ja Slim-view'in syntaksin sekoitusta. Jokaista näkymää ei tarvitse kirjoittaa täysimääräisenä, vaan toistuvia elementtejä sivuilla voidaan hyödyntää käyttämällä extends-merkintää. Kuvassa 19 on näkymä, joka luodaan student_base.php-tiedoston pohjalle. Student-base.php-tiedostossa määritetään lohkoilla, mihin kohtaan sivua liitettävät osat sidotaan. Vastaavan nimiset lohkot löytyvät liitettävästä näkymästä (kuva 20). Sovelluksessa toistuvia elementtejä ovat mm. navigaatio, footer ja erinäisten scriptien ja tyylitiedostojen sisällyttäminen.

```
{% extends "base/student_base.php" %}

{% block title %}
Koti
{% endblock %}

{% block head %}
{{ parent() }}
{% endblock %}

{% block content %}
<div class="page-header">
    <h1>Projektinseuranta</h1>
</div>
<p>Olet kirjautunut oppilaana projektinseurantaan.</p>
{% endblock %}
```

KUVA 19 Sovelluksen etusivun näkymä (Joonas Mononen, 2017)

```
<!DOCTYPE html>
<html>
    <head>
        {% block head %}
            <link rel="stylesheet" href="style.css" />
            <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css" />
            <meta charset="UTF-8">
            <title>{% block title %}{% endblock %}</title>
        {% endblock %}
    </head>
    <body>
        <nav class="navbar navbar-default">
            <a href="#" class="navbar-brand">Projektinseuranta</a>
            <div class="container-fluid">
            </div>

        </nav>
        <div class="container">
            {% block content %}
            {% endblock %}
        </div>

        <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
        <!-- Include all compiled plugins (below), or include individual files as needed -->
        <script src="js/bootstrap.min.js"></script>
    </body>
</html>
```

KUVA 20 Esimerkki pohjanäkymä (Joonas Mononen, 2017)

5.4 AngularJS:n ja Slimin yhteiskäyttö

AngularJS sekä Slim mahdollistavat molemmat näkymien käytön. Tässä projektissa näkymät on toteutettu Slim-view´in avulla ja sivun sisällä tapahtuvat toiminnot AngularJS:n avulla. Jälkeenpäin tarkasteltuna tämä ei osoittaunut kaikista järkevimmäksi tavaksi sovelluksen toteuttamiseen. Ongelmia muodostuu etenkin siinä vaiheessa, kun sovelluksessa on tarkoitus navigoida sivulta toiselle ja osoittaa jokin tarkempi polku sivun sisällä, esimerkiksi avata tietty projektiryhmä projektihallinnassa. Slim olisi kannattanut jättää vain RESTful API:n rooliin, sillä silloin projektin back-end ja font-end eivät näin olisi mitenkään riippuvaisia toisistaan, sekä tarkempi ohjaus sivuston sisällä olisi ollut helpompaa. Projektin muutenkin tiukan aikataulun takia tätä ongelmaa ei ehditty selvittää tämän enempää.

Näkymät sisältävät kahta erilaista syntaksia datan sitomiseen, joista toinen tapahtuu jo palvelimella ja käyttäjälle palautetaan näkymä, missä data on jo sisällytetty näkymään. AngularJS puolestaan lataa datan AJAXin avulla ja vasta sen jälkeen sisällyttää sen osaksi näkymää. Tämä aiheuttaa yleensä erinäisten latausindikaattorien näkymistä ruudulla.

6 JATKOKEHITYS

Sovellus saatiin siihen pisteeseen, että sen käyttäminen on mahdollista aloittaa. Paljon suunniteltuja ominaisuuksia jäi toteuttamatta ajan puutteen takia. Sovellus on lisäksi täysin testaamaton todellisessa käytössä, eli paljon tässä luvussa mainitsemattomia muutoksia todennäköisesti tehdään. Sovellus luovutettiin myöhemmin jatkokehitettäväksi toiselle henkilölle.

6.1 Henkilötiedot

Nykyisellä toteutustavalla sovellus saa tietoonsa vain käyttäjän käyttäjätunnuksen. Tämän takia käyttäjät joutuvat ensikirjautumisen yhteydessä täyttämään omat henkilötietonsa itse, mikä altistaa virheille ja väärinkäytölle.

Ratkaisuksi tähän ongelmaan on se, että henkilötiedot haettaisiin ensikirjautumisen yhteydessä jostain jo olemassa olevasta järjestelmästä, esimerkiksi Wilmasta. Wilma sisältää web service -rajapinnan, josta voidaan hakea tietoa SOAP (Simple Object Access Protocol) -tietoliikenneprotokollan avulla. Tätäkin aihetta tutkittiin hieman opinnäytetyön aikana, mutta todettiin, ettei se aikataulusyistä mahdu mukaan

6.2 Oppilasnäkymät

Projektissa keskityttiin pääasiassa ohjaajien toimintoihin, joten opiskelijalle näkyvät käyttöliittymät jäivät vajavaisiksi ja osa uupuu kokonaan. Ulkoasua ja käytettävyyttä ei ole mietitty nykyisten opiskelijan toiminnollisuuksien osalta.

Opiskelijat pystyvät nyt täyttämään ja katselemaan nykyhetkeen asti olevia omia viikkoraportteja. Jokaiselle ryhmälle olisi hyvä pystyä asettamaan erikseen voiko kyseisen ryhmän jäsenet nähdä toistensa viikkoraportit. Lisäksi jokaisella ryhmällä täytyisi olla ryhmäkohtainen viikkoraportti, jossa projektin edistymistä voisi seurata yleisellä tasolla. Lisäksi opiskelijat eivät pysty näkemään eivätkä kommentoimaan ryhmän tapahtumia, vaikka opettajat pystyvät niitä luomaan.

6.3 Etusivu

Etusivua ei hyödynnetty projektin tässä vaiheessa mitenkään. Siihen voitaisiin laittaa ilmoitusluontoisesti ajankohtaisia tapahtumia, esimerkiksi oppilaat voisivat nähdä suoraan kyseisen viikon täyttämättömät raportit, joita pääsisi siten täyttämään yhdellä klikkauksella. Lisäksi opettajien antamat uuden kommentit voitaisiin esittää tässä, muuten ne jäävät helposti huomaamatta.

7 YHTEENVETO

Opinnäytetyön aiheena oli toteuttaa Savonia-ammattikorkeakoululle selainpohjainen sovellus oppilasprojektien seurantaan. Sovelluksen toteutuksessa merkittävimmät ohjelmistokehykset olivat Slim 3, AngularJS ja Bootstrap. Sovelluksen tietokantaohjelmisto oli MySQL.

Lopputuloksena aikaansaatiin sovellus, jossa ohjaajat pystyvät luomaan sovellukseen kursseja ja ryhmiä sekä liittämään niihin opiskelijoita. Seuranta-sivulla ohjaajat pystyvät selaamaan ryhmittäin oppilaiden tuottamia viikkoraportteja ja luomaan ryhmille tapahtumia, eräänlaisia merkkipaaluja projektissa. Oppilaat pystyvät selaamaan omia ryhmiään sekä täyttämään niihin viikkoraportteja.

Sovelluksen kehitys eteni hitaasti ja jakaantui noin puolen vuoden aikajaksolle. Toiminnollisuuksien määrittely oli puutteellista, mikä aiheutti jatkuvaa suunnittelutarvetta projektin edetessä. Lisäksi erinäiset projektista riippumattomat tekijät hidastivat kehitystä.

Sovellus vaatii vielä jatkokehitystä, eikä tallaisenaan välttämättä sovi vielä sellaisenaan todelliseen käyttöön. Kaikki tärkeimmät toiminnot on toteutettu, mutta perusteellinen testaaminen jäi viivästyneen aikataulun takia kokonaan tekemättä.

LÄHTEET JA TUOTETUT AINEISTOT

Almende B.V. (2015). *vis.js*. Haettu 7. 5 2017 osoitteesta http://visjs.org/

Google. (ei pvm). *AngularJS: Developers guide*. Haettu 14. 5 2017 osoitteesta https://docs.angularjs.org/guide/controller

O'Phinney, M. W. (2015). *PSR-7 And Middleware*. Haettu 7. 5 2017 osoitteesta http://weierophinney.github.io/2015-10-20-PSR-7-and-Middleware/#/1

Oracle Corporation. (26. Maaliskuu 2017). *MySql*. Noudettu osoitteesta https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html

phpMyAdmin contributors. (26. Maaliskuu 2017). *phpMyAdmin*. Noudettu osoitteesta https://www.phpmyadmin.net/

Slim Framework Team. (ei pvm). *Slim*. Haettu 7. 5 2017 osoitteesta https://www.slimframework.com/docs/concepts/middleware.html

ui-bootstrap contributors. (ei pvm). *UI Bootstrap*. Haettu 7. 5 2017 osoitteesta https://angular-ui.github.io/bootstrap/#!#buttons

w3schools. (ei pvm). *w3schools*. Haettu 7. 5 2017 osoitteesta https://www.w3schools.com/bootstrap/bootstrap_grid_examples.asp

w3schools. (ei pvm). *w3schools*. Haettu 14. 5 2017 osoitteesta https://www.w3schools.com/angular/angular_directives.asp

Bootstrap, Wikipedia. (2017). Haettu 15. 5 2017 osoitteesta Wikipedia: https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework)

AngularJS, Wikipedia. (2017). *Wikipedia*. Haettu 15. 5 2017 osoitteesta AngularJS: https://en.wikipedia.org/wiki/AngularJS