Opinnäytetyö (AMK)

Tietojenkäsittely

Tietojärjestelmät

2010

Tomi Muurimaa

PROJEKTINHALLINTA-JÄRJESTELMÄ VERKKOKÄYTTÖLIITTYMÄLLÄ

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietojenkäsittely | Tietojärjestelmät

Marraskuu 2010 | Sivumäärä: 65

Ohjaaja: Anne Jumppanen

Tekijä: Tomi Muurimaa

PROJEKTINHALLINTAJÄRJESTELMÄ VERKKOKÄYTTÖLIITTYMÄLLÄ

Tässä opinnäytetyössä käsitellään projektinhallintajärjestelmän kehitystä. Järjestelmään toteutetaan raportointitoiminnallisuus projektien edistymisen seurantaan ja keskinäiseen vertailuun, sekä toiminnallisuus projektien luomiseen valmiista projektimalleista. Tavoitteena on luoda järjestelmä, joka yhdenmukaistaa yrityksen projekteista kerättyä dataa ja parantaa projektisuunnittelun resurssienhallintaa.

Työssä tutkitaan verkkokäyttöliittymän soveltuvuutta projektinhallintajärjestelmän tarpeisiin ja ketterien kehitysmenetelmien soveltamista yksin toteutettavaan projektiin.

Opinnäytetyö on toteutettu toimeksiantona Turussa ja Helsingissä toimivalle Medusaworks Oy:lle, joka on sisältöjä ja sisältöjärjestelmiä internetiin tuottava yritys.

Teoriaosuudessa käsitellään järjestelmän toteutuksessa käytettäviä työvälineitä ja ohjelmointikieliä. Lisäksi esitellään projektiin toteutettavien ominaisuuksien vaatimukset.

Käytännön osiossa esitellään projektinhallintajärjestelmään toteutetut näkymät ja käydään läpi järjestelmän toteutus ja toiminta.

ASIASANAT:

Tietojärjestelmät, PHP, JavaScript, MySQL, tietokannat, Ajax-ohjelmointi

BACHELOR´S THESIS | ABSTRACT

UNIVERSITY OF APPLIED SCIENCES

Information Technology | Information Systems

November 2010 | Total number of pages: 65

Instructor: Anne Jumppanen

Author: Tomi Muurimaa

PROJECT MANAGEMENT SYSTEM WITH A WEB USER INTERFACE

This thesis concerns the development of a project management system. The developed system includes the functionality for comparing projects and their shared tasks, and the creation of new projects from project templates. The aim of this study is to create a system for improving the resource handling of project planning through standardization of gathered project data.

The thesis addresses how a web user interface answers to the needs of a project management system and the application of agile software development principles to a one-man development project.

This work was a commission for Medusaworks Inc., a company that provides content on the internet, located in Turku and Helsinki.

The theoretical part focuses on the tools used for the development of the system, and the specified requirements of the system.

The empirical part presents the development and in-depth functionality of the project management system.

KEYWORDS:

Information systems, PHP, JavaScript, MySQL, database, Ajax-programming

SISÄLTÖ

KUVAT

KUVIOT

1 JOHDANTO

Tämä opinnäyte on toteutettu toimeksiantona Medusaworks Oy:lle. Medusaworks on Turussa ja Helsingissä toimiva verkkojulkaisuja tuottava yritys, joka vuonna 2010 työllisti seitsemän henkilöä. Yritys tuottaa monenlaisia projekteja useille asiakkaille; jotkin projektit toteutetaan alusta asti itse, jotkin osittain ulkopuolisia tekijöitä käyttäen. Uusia projekteja suunniteltaessa yritys käyttää aiemmista projekteista kerättyä dataa käytettävien resurssien arvioimiseen.

Ongelmana on projektien erilaisuus. Projektit saattavat olla hyvinkin erikokoisia. Vaikka useat projektit sisältävät samankaltaisia tehtäviä, yhtenäistä nimeämiskäytäntöä niille ei ole. Tästä johtuva tiedon hajanaisuus tekee tasapainoisen projektidatan keräämisen hankalaksi. Ilman luotettavaa projektidataa tulevien projektien resurssienkäytön suunnittelu on vaikeaa.

Medusaworks tarvitsi järjestelmän, jonka avulla projekteja on helpompi vertailla. Tarpeena oli esittää perusteltua tietoa tietynkokoisiin projekteihin käytettävistä resursseista esimerkiksi potentiaalisille asiakkaille.

Tässä opinnäytetyössä käsitellään projektinhallintajärjestelmän kehitysprosessia. Tavoitteena on dokumentoida järjestelmän toiminta ja selvittää, kuinka verkkokäyttöliittymätoteutus vastaa projektinhallintajärjestelmän tarpeisiin. Lisäksi tarkastellaan ketterien kehitysmenetelmien periaatteiden soveltumista yhden henkilön kehitysryhmän käyttöön.

Järjestelmä tulee sisältämään raportointitoiminnon projektien edistymisen seurantaan ja keskinäiseen vertailuun. Tavoitteena on luoda järjestelmä, joka parantaa yrityksen projekteista kerätyn datan laatua ja parantaa projektisuunnittelun resurssienhallintaa.

Järjestelmällä kerättävää dataa on tavoitteena yhdenmukaistaa, jolloin mahdollistetaan projektien vertailu. Tämän projektidatan pohjalta luotuja kaavioita voidaan käyttää hyödyksi uusien projektien suunnittelussa; toistuvasti

suunniteltua enemmän aikaa vieviin tehtäviin voidaan kiinnittää enemmän resursseja tai niitä voidaan tarkastella projektin toteutuksen aikana lähemmin.

Kaavioiden perusteella projektien kustannusarvioiden tekeminen nopeutuu, ja asiakkaille voidaan esittää mihin arviot perustuvat.

Toteutan järjestelmän kokonaan itsenäisesti toimeksiantajan toiveisiin ja omiin ideoihini perustuen.

Työn teoriaosuudessa esitellään järjestelmän kehityksessä käytettävät työvälineet ja ketterät menetelmät, joiden periaatteita järjestelmän kehityksessä käytetään. Käytännön osuudessa selvitetään lähtötilanne ja projektinhallintajärjestelmään toteutetut näkymät sekä niiden toiminta.

2 KETTERÄ OHJELMISTOKEHITYS

2.1 Ketterä manifesti

Ketterän ohjelmistokehityksen taustalla on Ketterä manifesti (Manifesto for Agile Software Development), jossa pyritään määrittelemään parempia tapoja ohjelmistokehittämiseen. Sen laati vuonna 2001 17 IT-alan henkilöä hiihtokeskuksessa käytyjen keskustelujen tuloksena (Highsmith 2001).

Manifestissa esitetään tiettyjen asioiden painottamista toisia enemmän ohjelmistokehityksessä. Sen neljä pääkohtaa ovat

- yksilöt ja vuorovaikutus prosessien ja työkalujen edellä
- toimiva ohjelmisto laajan dokumentaation edellä
- asiakasyhteistyö sopimusneuvottelujen edellä
- muutokseen reagoiminen suunnitelman noudattamisen edellä. (Beck ym. 2001a.)

2.2 Ketterän manifestin periaatteet

Manifesti perustuu kahteentoista periaatteeseen, joita ketterässä kehityksessä pyritään noudattamaan:

- Asiakkaan tyytyväisenä pitäminen ohjelmistoa aikaisessa vaiheessa ja jatkuvasti toimittamalla.
- Muutosten vastaan ottaminen jopa myöhäisessä kehitysvaiheessa.
- Toimivan ohjelmiston toimittaminen muutaman viikon tai kuukauden välein, mieluiten lyhyemmällä aikavälillä.
- Liiketoimintaa ymmärtävien henkilöiden ja kehittäjien yhteistyön ylläpitäminen projektin aikana.

- Projektien rakentaminen motivoituneiden henkilöiden ympärille ja heidän kykyihinsä luottaminen.
- Kasvokkain keskustelun käyttäminen tehokkaimpana tiedonvälityskeinona.
- Toimivan ohjelmiston pitäminen edistyksen mittarina.
- Kestävän ohjelmistokehityksen kannattaminen ja jatkuva tahdin ylläpitäminen.
- Jatkuva teknisen laadun huomioiminen ketteryyden parantajana.
- Yksinkertaisuuteen eli työmäärän vähentämiseen pyrkiminen.
- Parhaan arkkitehtuurin, vaatimusten ja suunnittelun toteuttaminen itsejärjestettyjä ryhmiä käyttäen.
- Työtapojen tarkastelu ja tarpeen vaatiessa muuttaminen tehokkuuden lisäämiseksi säännöllisin väliajoin. (Beck ym. 2001b.)

2.3 Kehitysmenetelmiä

Tässä osiossa esitellään kolme tunnettua ketterää menetelmää.

2.3.1 Scrum

Scrum on monimutkaisiin projekteihin tarkoitettu ketterä kehysjärjestelmä. Se on alkuperäisesti kehitetty ohjelmistokehitykseen, mutta toimii minkälaisessa projektissa tahansa. (Scrum Alliance 2010a.)

Scrumiin kuuluu kolme roolia: tuoteomistaja, joka on vastuussa projektin liikearvosta, Scrum-mestari, joka vastaa ryhmän toiminnasta ja tuottavuudesta sekä ryhmä, joka on itsejärjestyvä (Scrum Alliance 2010b). Tyypilliseen Scrum-ryhmään kuuluu viidestä yhdeksään henkilöä (Mountain Goat Software 2010).

Scrumissa kehitys etenee toteutuskierroksissa, joita kutsutaan sprinteiksi. Sprintin alussa kehitysryhmä valitsee asiakkaan vaatimusten pohjalta tehdystä työtehtävien tärkeyden mukaan järjestetystä työlistasta kierroksen aikana toteutettavat ominaisuudet. Kahdesta neljään viikkoa kestävän sprintin aikana ominaisuudet toteutetaan toimivaksi ohjelmistoksi. Kehitysryhmä kokoontuu päivittäin seuraamaan kehityksen edistymistä. Kierroksen päättyessä järjestetään sprintin katselmus. (Scrum Alliance 2010a.)

Katselmuksessa ryhmä esittää tuoteomistajalle sprintin aikana toteutetut ominaisuudet. Sen yhteydessä mietitään keinoja tuotteen ja prosessin parantamiseksi. (Scrum Alliance 2010b.)

Katselmuksen jälkeen ryhmä valitsee työlistasta seuraavalla kierroksella toteutettavat ominaisuudet. Toteutuskierrokset toistuvat kunnes työlistan kohdat on suoritettu, budjetti käytetty tai aikaraja tulee vastaan. (Scrum Alliance 2010a.)

2.3.2 Extreme programming

Extreme programming on Kent Beckin 1990-luvun lopulla kehittämä kevyt kehitysmenetelmä (Copeland 2001).

Extreme programming perustuu yksinkertaisuuteen, kommunikaatioon ja palautteen antamiseen. Kaikki projektin parissa työskentelevät ovat olennainen osa ryhmää. (Jeffries 2010.)

Suunnittelussa käytetään asiakkaan laatimia käyttäjäkertomuksia. Niissä määritellään ohjelmistoon toteutettavat ominaisuudet. (Wells 1999a.)

Projekti on jaettu iteraatioihin, ja jokainen iteraatio aloitetaan iteraatiosuunnittelulla. Asiakas valitsee käyttäjäkertomuksien perusteella luodusta julkaisusuunnitelmasta tärkeimmät toteutettavat tehtävät iteraatiota varten. Arviona mahdollisesta toteutettavasta määrästä käytetään edellisen iteraation aikana tehtyä työtä. (Wells 1999b.)

Iteraatiot kestävät yhdestä kolmeen viikkoa. Iteraation aikana pyritään suorittamaan asiakkaan iteraatiota varten valitsemat tehtävät. Iteraatioiden pituus pidetään samana koko projektin aikana, jotta tulevien iteraatioiden aikana suoritetun työn määrää pystytään arvioimaan (Wells 1999c.)

Extreme programmingissa kaikki tuotantokoodi kirjoitetaan pariohjelmointina, jossa toinen pareista ohjelmoi ja toinen tarkkailee tuotettua koodia virheiden varalta (Jeffries 1998).

Testauksessa käytetään etukäteen ohjelmoituja yksikkötestejä. Testit luodaan kaikelle koodille, ja koodin on läpäistävä kaikki testit ennen julkaisua. (Wells 1999d.)

Käyttäjäkertomusten pohjalta luodaan mustalaatikkotestejä hyväksymistesteiksi. Jokainen hyväksymistesti edustaa odotettua tulosta ohjelmistolta. Asiakas on vastuussa testien tulosten oikeellisuudesta ja siitä, mitkä epäonnistuneet testit ovat oleellisimpia. Hyväksymistestien tarkoituksena on taata että asiakkaan vaatimukset on täytetty ja että ohjelmisto on hyväksyttävä. (Wells 1999e.)

2.3.3 Agile Unified Process

Agile Unified Process on yksinkertaistettu versio Rational Unified Processista (Ambler 2009).

Rational Unified Process on ohjelmistonkehitysmenetelmä joka on iteratiivinen, arkkitehtuurikeskeinen ja käyttötapauksiin perustuva. Se tarjoaa muokattavissa olevan prosessikehysjärjestelmän ohjelmistokehitykseen. (Kroll & Kruchten 2004, 3.)

Agile Unified Process on nelivaiheinen:

- **Aloitusvaiheessa** tavoitteena on tunnistaa projektin laajuus ja mahdollinen arkkitehtuuri, sekä saada alkuvaiheen rahoitus ja osakkaiden hyväksyntä.
- **Tähdennysvaiheessa** tavoitteena on todentaa ohjelmiston arkkitehtuuri.

- **Rakennusvaiheessa** tavoitteena on rakentaa toimiva ohjelmisto inkrementaalisesti projektin osakkaiden tarpeiden mukaisesti.
- **Siirtymävaiheessa** ohjelmisto siirretään tuotantoympäristöön. (Ambler 2009).

Agile Unified Process sisältää seitsemän vaiheiden aikana iteratiivisesti suoritettavaa osa-aluetta:

- **Malli**, jossa tavoitteena on löytää ratkaisu projektin käsittelemään ongelmaan.
- **Implementointi**, jossa tavoitteena on muuttaa malli suoritettavaksi koodiksi ja suorittaa yksikkötestaus.
- **Testaus**, jossa tavoitteena on suorittaa laadunvalvontaa virheitä etsimällä, ohjelmiston toimintoja varmentamalla ja toimintaa vaatimuksiin vertaamalla.
- **Käyttöönotto**, jossa tavoitteena on ohjelmiston toimitus ja loppukäyttäjille käytettäväksi tekeminen.
- **Konfiguraationhallinta**, jossa tavoitteena on hallinnoida pääsyä projektiin liittyvään dokumentaatioon, ja dokumentteihin tehtäviin muutoksiin.
- **Projektinhallinta**, jossa tavoitteena on hallinnoida riskejä, ohjata ihmisiä ja toimia projektin laajuuden ulkopuolella olevien henkilöiden ja järjestelmien kanssa jotta varmistutaan aikataulussa ja budjetissa pysymisestä.
- **Ympäristö**, jossa tavoitteena on tukea muita osa-alueita varmistamalla että tarvittava ohjeistus ja työkalut on ryhmän saatavilla tarvittaessa. (Ambler 2009).

Ohjelmisto toimitetaan tuotantoversioksi osissa; seuraava versio sisältää aina lisättyjä ominaisuuksia. Ensimmäisen tuotantoversion julkaisu kestää tyypillisesti kauemmin kuin sitä seuraavien. Tämä voi johtua esimerkiksi

järjestelmän perusominaisuuksien toteuttamisen vaikeudesta tai vasta muodostumassa olevasta ryhmädynamiikasta. (Ambler 2009).

Agile Unified Process noudattaa Ketterässä manifestissa esitettyjä periaatteita (Ambler 2009).

2.4 Opinnäytetyöhön soveltuminen

Järjestelmän kehittämiseen käytetään ketteriä menetelmiä. En kuitenkaan noudata tarkkaan mitään tiettyä menetelmää, vaan sovellan Ketterässä manifestissa esitettyjä periaatteita.

Perusteena tähän on se, että opinnäytetyöhön käyttämäni aika on rajallinen ja ketterät menetelmät mahdollistavat nopean ohjelmistokehityksen. Lisäksi hoidan itse järjestelmän toteutuksen yksin ja täydennän määriteltyjä vaatimuksia omilla ideoillani. Muutoksia voi ilmaantua yllättäen myös toimeksiantajan taholta, sillä kaikkia järjestelmän vaatimuksia ei tiedetä vielä projektin alkuvaiheessa.

Koska tarkkaan asetettuja vaatimuksia esimerkiksi ulkoasusta ei ole, saatan muuttaa valikoiden asetteluja tai väriteemaa ilman määrittelydokumenttien uudelleen kirjoittamista. Tämä sopii hyvin myös toimeksiantajalle, jolle dokumentteja tärkeämpää on toimiva ohjelmisto.

Ketterien kehitysmenetelmien käyttäminen sopii järjestelmän toteuttamiseen erityisen hyvin siksi, että koska olen työharjoittelussa sen kehittämisen aikana, olen jatkuvassa yhteydessä toimeksiantajaan.

Kehitystä seurataan lyhyissä kehityspalavereissa, joiden aikana esittelen kehityksen etenemisen ja otan vastaan uusia ideoita. Tarkkaa vaatimusmäärittelyä ei tehdä, sillä muutoksia saatetaan haluta tehdä nopeasti sitoutumatta valmiisiin suunnitelmiin. Lyhyitä palavereja pyritään pitämään ainakin noin viikon kestävien iteraatioiden välillä, tai aina kun sitä pidetään tarpeellisena.

3 JÄRJESTELMÄN KEHITYSTYÖKALUT

Järjestelmän kehittämisessä käytetään JavaScriptia selainpuolen ja PHP:ta palvelinpuolen ohjelmointiin. Kehitysympäristönä on Eclipse IDE ja testipalvelimena Apache. Testiympäristön pystyttämiseen käytetään XAMPP:ta.

3.1 PHP

PHP on laajasti käytössä oleva avoimen lähdekoodin yleiskäyttöinen skriptikieli. Se on erityisen sopiva web-kehitykseen ja sitä voidaan upottaa HTML:ään. PHP:tä kirjoitetaan <?php ja ?> -käskyjen sisään. PHP-koodi suoritetaan palvelimella, ja se luo selaimelle lähetettävän HTML:n. (Achour ym. 2010.)

Esimerkki PHP-koodista, joka tulostaa sivulle ”Hei maailma!”:

```
<html>
<head>
<title>PHP-esimerkki</title>
</head>
<body>
<?php echo ”Hei maailma!” ?>
</body>
</html>
```

3.1.1 Suorituskyky

PHP on suorituskyvyltään nopea. Yhdellä palvelimella voi palvella miljoonia osumia päivittäin. Se on myös hyvin skaalautuva ja soveltuu näin monenkokoisiin projekteihin. (Thomson & Welling 2009, 4.)

3.1.2 Tietokantaintegraatio

PHP sisältää valmiit yhteydet moniin tietokantajärjestelmiin. Näitä ovat MySQL:n lisäksi muiden muassa SQLite, PostgreSQL, Oracle, dbm, DB2 ja FilePro. (Thomson & Welling 2009, 4.)

3.1.3 Sisäänrakennetut kirjastot

Koska PHP on suunniteltu web-käyttöön, se sisältää useita valmiita toimintoja monipuolisten web-tehtävien tekemiseen. Näitä ovat esimerkiksi kuvien generoiminen, verkkopalveluihin yhteyden ottaminen, XML:n tulkitseminen, evästeiden käsittely, sähköpostin lähettäminen sekä PDF-dokumenttien luominen. Toimintojen käyttäminen edellyttää vain muutamaa riviä koodia. (Thomson & Welling 2009, 5.)

3.1.4 Helppous

PHP:n kielioppi perustuu muihin ohjelmointikieliin, pääasiassa C:hen ja Perliin. Tämän ansiosta kenen tahansa C++:aa, Javaa tai muuta C:n kaltaista kieltä osaavan on helppo oppia sitä. Tämä mahdollistaa PHP:n hyötykäytön aloittamisen lähes välittömästi. PHP tukee C++:n ja Javan tapaan oliopohjaista ohjelmointia. Näiden kielten ominaisuudet, kuten perintä, näkyvyysmääreet ja rajapinnat ovat myös PHP:ssa. (Thomson & Welling 2009, 5.)

3.1.5 Useat alustat

PHP on saatavilla monille eri käyttöjärjestelmille. PHP:tä voi kirjoittaa ilmaisilla Unix-pohjaisilla käyttöjärjestelmillä kuten Linuxilla ja FreeBSD:llä, maksullisilla Unix-versioilla, kuten Solariksella ja OS X:llä, tai Windowsin eri versioilla. Hyvin kirjoitettu koodi toimii tavallisesti muuntelemattomana eri järjestelmillä. (Thomson & Welling 2009, 5.)

3.1.6 Monipuolinen kehittäminen

PHP sallii joustavat lähestymistavat kehittämiseen. Yksinkertaiset tehtävät voi suorittaa yksinkertaisin ratkaisuin, ja suurempiin sovelluksiin voi soveltaa suunnittelumalleihin perustuvia kehysjärjestelmiä. (Thomson & Welling 2009, 5.)

3.2 JavaScript

JavaScript on tulkattu ohjelmointikieli, joka sisältää oliopohjaisen toiminnallisuuden. Syntaktisesti kieli muistuttaa C:tä, C++:aa ja Javaa. Tavallisimmin JavaScriptiä käytetään WWW-selaimissa. (Flanagan 2006, 1.)

JavaScriptin ilmestyttyä vuonna 1995 sen päätarkoitus oli toimia syötteenvalidoijana palvelinpuolen kielten tukena. Ennen JavaScriptiä syötteiden validointi tapahtui aina palvelimella. Tällä oli huomattavaa vaikutusta internetin käyttöön puhelinmodeemien aikaan, sillä tieto lomakkeen tyhjästä tai vääränmuotoista tietoa sisältävästä kentästä saatiin vasta hitaan palvelinpyynnön jälkeen. (Zakas 2009, 1.)

Näistä ajoista JavaScript on kasvanut kaikkien tunnettujen selainten tärkeäksi ominaisuudeksi. Sitä ei käytetä enää pelkkään validointiin, vaan se voi liittyä kiinteästi lähes kaikkeen mitä selainikkuna sisältää. JavaScript tunnustetaan täydeksi ohjelmointikieleksi, joka pystyy monimutkaisiin laskutoimituksiin ja vuorovaikutteisuuteen. JavaScriptistä on tullut niin tärkeä osa internetiä, että jopa mobiililaitteiden selaimet tukevat sitä. (Zakas 2009, 1.)

JavaScript on sekä hyvin yksinkertainen että hyvin monimutkainen kieli, jonka oppii minuuteissa, mutta jonka täydellinen hallinta vie vuosia. (Zakas 2009, 1.)

Esimerkki JavaScript-koodista, joka lukee HTML-tekstikentän sisällön ja ilmoittaa sen alert-ikkunassa:

```
<script type=”text/javascript”>

alert(document.getElementById(”tekstikentta”).value);

</script>
```

```
<!-- loppu on HTML:ää, jolla tekstikenttä luodaan -->

<input type=”text” id=”tekstikentta”>
```

3.3 JavaScriptin kirjastot

Kirjastot ovat luokista ja funktioista koostuvia kokoelmia, joita käytetään ohjelmistonkehityksessä.

JavaScriptin kirjastot tarjoavat yksinkertaistetun tien monimutkaisiin selaimen toimintoihin. Kirjastoja on kahdenlaisia: yleiskirjastot ja erikoiskirjastot. Yleiskirjastojen avulla saadaan pääsy yleiseen selaimen toiminnallisuuteen ja niitä voidaan käyttää internetsivujen tai -sovelluksen perustana; erikoiskirjastot suorittavat tiettyjä toimintoja ja niitä käytetään vain jossain internetsivujen tai -sovelluksen osassa. (Zakas 2009, 759.)

JavaScriptin yleiskirjastoja ovat esimerkiksi Yahoo! User Interface Library (YUI), Prototype, The Dojo Toolkit, MooTools ja jQuery (Zakas 2009, 759-760). Tässä opinnäytetyössä käytetään jQuery-kirjastoa, sillä työskentelin sen parissa työharjoittelun aikana ja se on helposti laajennettavissa tarpeen vaatiessa.

3.4 JQuery

JQuery on JavaScriptin kirjasto, joka yksinkertaistaa HTML-dokumentissa liikkumista, tapahtumankäsittelyä, animointia ja Ajaxin käyttöä (jQuery Project 2010a). JQueryn avulla on mahdollista helposti luoda esimerkiksi erilaisia animoituja valikoita tai kuvagallerioita, siirtää sivun elementtejä tai noutaa tietoa tietokannasta Ajaxia käyttämällä. JQueryyn on saatavilla lukuisia lisätoiminnallisuuksia mahdollistavia liitännäisiä, joita käyttäjät ovat itse kehittäneet.

3.4.1 Elementteihin viittaaminen

HTML-elementteihin viitataan jQueryssa syntaksilla $(”elementin_nimi”). Tämä mahdollistaa esimerkiksi sivulla sijaitsevan taulukon soluihin tai lomakekenttiin viittaamisen. Jos halutaan viitata yksittäiseen taulun soluun tai johonkin tiettyyn

elementtiin, voidaan käyttää syntaksia $(”#elementin_id”), kun elementille on määritelty id-atribuutti. Vastaavasti luokkaan voidaan viitata käyttämällä .-merkkiä #-merkin sijaan.

JQuerylla voi viitata elementteihin myös monella muulla tavalla. Se sisältää valitsijat muun muassa sivun ensimmäiselle ja viimeiselle elementille, nollasta alkavan indeksinumeron mukaan parittomille ja parillisille elementeille, animoiduille elementeille ja näkyville tai piilotetuille elementeille (W3Schools 2010). Valitsijoita voi myös olla useita.

```
$(”tr:even:not(:animated)”)
```

Yllä oleva täysin irrallinen esimerkki on viittaus indeksinumeron mukaan parillisiin tr-elementteihin, jotka eivät ole animoituina.

3.4.2 Elementtien manipulointi

Jquery sisältää monia funktioita sivun elementtien manipulointiin. Niillä voidaan muuttaa elementtien attribuutteja, vaihtaa tyyliasetuksia tai lisätä, kopioida ja poistaa kokonaisia elementtejä. Funktiot on nimetty loogisesti, ja osan toiminnallisuuden voi päätellä niiden nimestä. Funktioita ovat muiden muassa .addClass(), .append(), .attr(), .css(), .empty(), .hasClass(), .remove(), .text() ja .val() (jQuery Project 2010b).

Funktioita käytetään valitsijan perässä. Esimerkki:

```
$(”tr:even:not(:animated)”).empty();
```

Tämä lause tyhjentää indeksinumeron mukaan parilliset tr-elementit, jotka eivät ole animoituina.

Valitsijoiden tapaan myös funktioita voidaan käyttää useita kerrallaan. Esimerkki monimutkaisemmasta jQuery-lauseesta:

```
$(”tr:even:not(:animated)”).
removeClass(”vanhaLuokka”).insertAfter(”#uusi”);
```

Tämä lause poistaa parillisilta ei-animoiduilta tr-elementeiltä vanhaLuokka-nimisen luokan, ja siirtää ne sivulla uusi-id:llä varustetun elementin jälkeen.

3.4.3 Ajax

Ajax (Asynchronous Javascript and XML) on selain- ja palvelinpuolen ohjelmoinnin yhdistämistä niin, että tietoa saadaan haettua palvelimelta ilman koko sivun uudelleen lataamista. Selainpuolen ohjelmointi käyttää palvelinpuolen koodia hyväkseen päivittääkseen vain osaa selainikkunasta. (Thomson & Welling 2009, 856.) Tämä mahdollistaa entistä dynaamisempien ja interaktiivisempien käyttöliittymien luomisen. Jokin valikko voi esimerkiksi ladata sisältönsä tietyn elementin sisään ilman sivupäivitystä.

JQuery sisältää useita funktioita Ajaxiin liittyen. Ajax-pyyntö tehdään $.ajax()-funktiolla. Funktiolle voidaan antaa parametrina asetukset. (jQuery Project 2010c.) Esimerkki Ajaxin käytöstä jQueryllä:

```
$.ajax({ type: ”POST”,

url: ”tallennaTiedot.php”,

data: ”nimi=suunnittelu”});
```

Lause siirtää datan (nimi=suunnittelu) POST-metodilla määriteltyyn URL:ään tallennaTiedot.php. Siirrettyä dataa voidaan sen jälkeen käyttää kyseisessä tiedostossa, esimerkiksi tietokantaan tallentamisen yhteydessä.

3.5 SQL

SQL on tietokantojen hallintaan yleisesti käytetty ANSIn (American National Standards Institute) standardoima kieli. Sitä voidaan käyttää muun muassa

- tietokantakyselyjen tekemiseen
- tietueiden eli tietokantarivien tietokantaan lisäämiseen
- tietokannan tietueiden päivittämiseen
- tietokannan tietueiden poistamiseen

- uusien tietokantataulujen luomiseen
- tietokantataulujen käyttäjärajoitteiden asettamiseen. (Price 2007, 3-4.)

Standardista huolimatta SQL:stä on olemassa useita eri versioita. Suurin osa SQL:n peruslauseista toimii kuitenkin samalla tavalla kaikissa versioissa (Gunderloy ym. 2006, 103-104).

Esimerkki SQL-lauseesta:

```
SELECT projekti FROM projektit WHERE id = 16;
```

Lause palauttaa projektit-taulusta projektin, jonka id-sarakkeessa on arvo 16. SQL-lauseessa voidaan käyttää myös *-merkkiä:

```
SELECT * FROM projektit;
```

Tämä lause palauttaa kaikki rivit projektit-nimisestä taulusta.

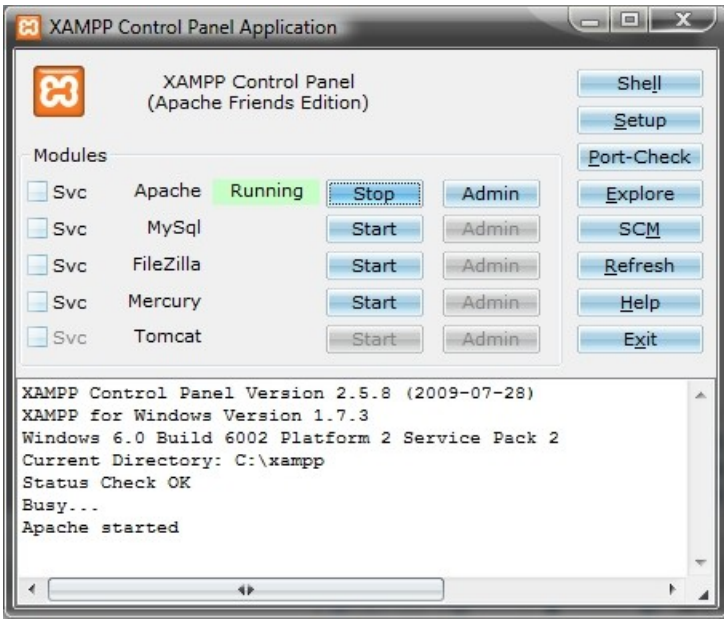
4 KEHITYSYMPÄRISTÖ

4.1 XAMPP

XAMPP on helposti asennettava ja ilmainen MySQL:n, PHP:n ja Perlin sisältävä Apache-jakelu (Seidler 2009). Apache HTTP Server Projectin kehittämä Apache on kaupallisen tason vaatimuksia vastaava, runsaasti ominaisuuksia sisältävä ilmainen HTTP-palvelin. Sen kehittämisestä vastaa joukko vapaaehtoisia ympäri maailman. (Apache Software Foundation 2009.)

Huomattavaa on, että XAMPP ei sisällä varsinaista ohjekirjaa, vaan dokumentointi on FAQ-listojen muodossa XAMPP:n kotisivuilla (Seidler 2010).

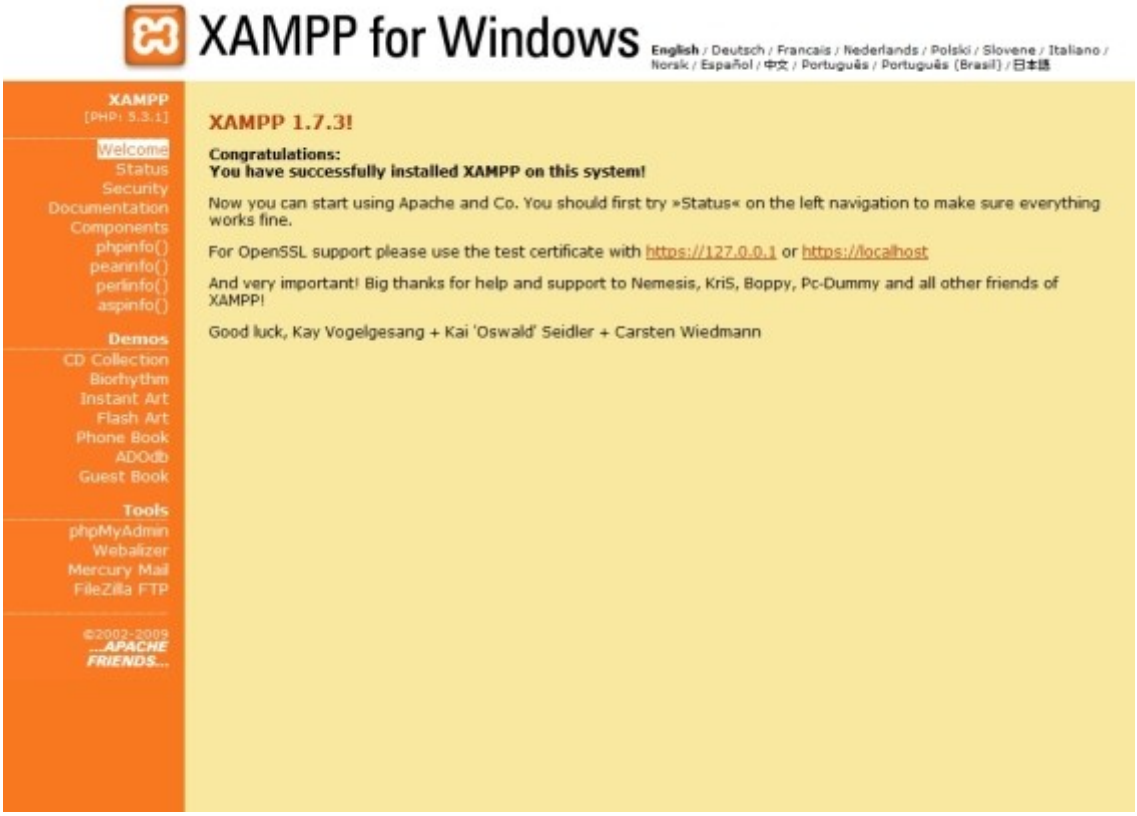
XAMPP sisältää graafisen käyttöliittymän palvelimen ja tietokannan helppoon käynnistämiseen.



Kuva 1: XAMPP:n ohjauspaneeli Apache käynnistettynä.

Palvelin on oletusarvoisesti konfiguroitu palvelemaan pyyntöjä vain paikallisesti, joten kehitysympäristönä se on tietoturvallinen.

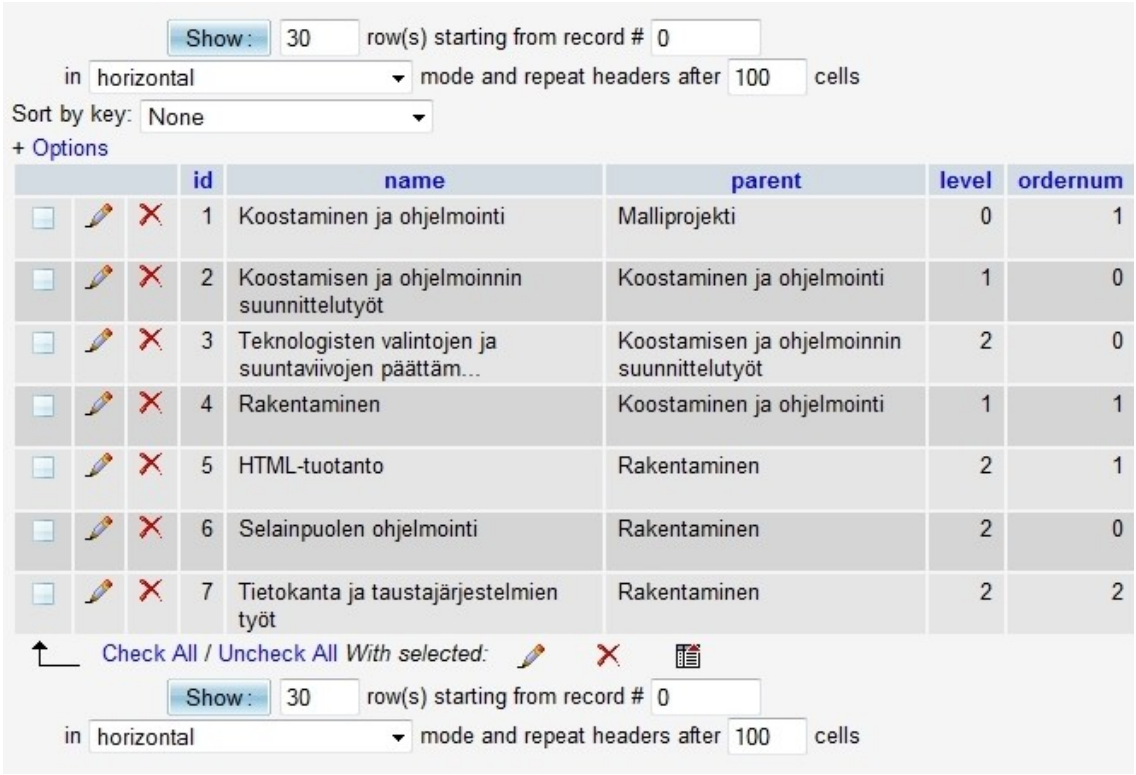
Palvelimen käynnistämisen jälkeen siirtymällä internet-selaimella osoitteeseen http://localhost saa ennen palvelimen konfigurointia näkyviin XAMPP:n esimerkkisivun, joka sisältää muun muassa tietoa järjestelmän tilasta ja esimerkkejä PHP:n käytöstä lähdekoodeineen.



Kuva 2: XAMPP:n esimerkkisivu.

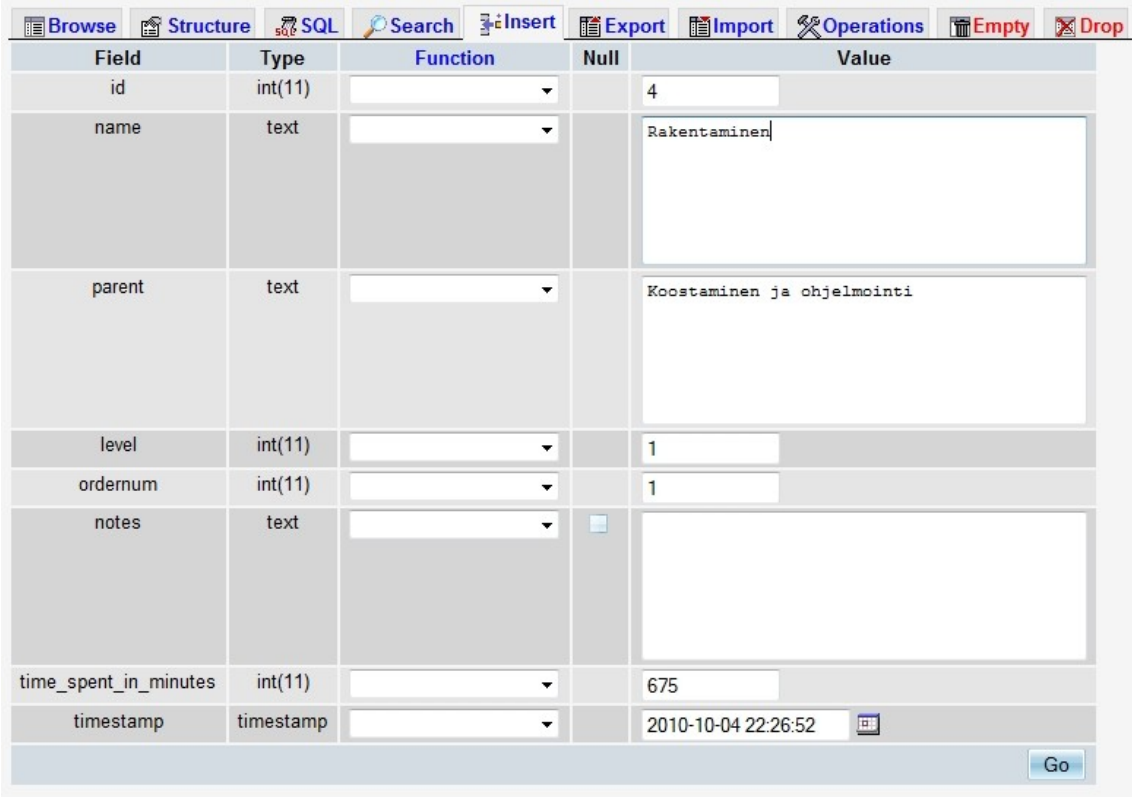
4.2 PhpMyAdmin

XAMPP sisältää PHP:llä kirjoitetun (PhpMyAdmin Developer Team 2010) phpMyAdmin-työkalun MySQL-tietokantojen hallintaan. PhpMyAdmin on graafinen käyttöliittymä tietokantaan. Sillä voi kuitenkin myös kirjoittaa SQL-lauseita.



Kuva 3: Osa phpMyAdminin taulunäkymästä, tiedot horisontaalisesti esitettyinä.

PhpMyAdminissa esitetyn tietokantataulun jokainen rivi sisältää painikkeen kyseisen rivin editointiin ja poistamiseen, sekä valintalaatikon mahdollistamaan useaa riviä koskevat operaatiot. Taulunäkymässä on mahdollista valita kerrallaan näytettävien rivien määrä ja tietojen horisontaalinen tai vertikaalinen esitystapa.



Kuva 4: Tietueen muokkaamista phpMyAdminilla.

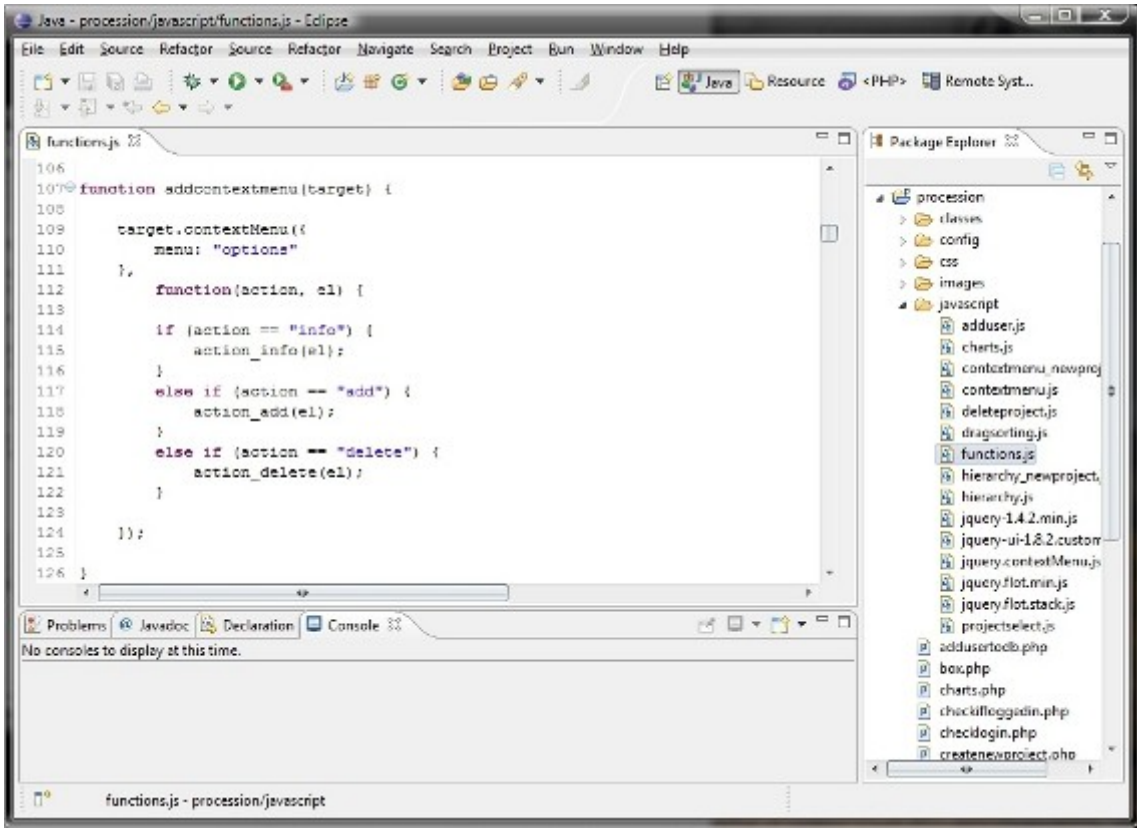
4.3 Eclipse IDE

Eclipse on integroitu kehitysympäristö (Integrated Development Environment), joka on tarkoitettu sovelluskehitykseen millä tahansa kielellä (Burnette 2005).

Integroidut kehitysympäristöt ovat sovelluskehityksessä auttavia työvälinekokoelmia. Useimmat integroidut kehitysympäristöt sisältävät työvälineet ainakin

- lähdekoodin kirjoittamiseen ja editoimiseen
- virheiden näkemiseen kirjoitettaessa
- koodisyntaksin korostamiseen
- toistuvien tehtävien automatisoimiseen
- koodin kääntämiseen

- luokkarakenteiden selaamiseen. (Nourie 2005.)



Kuva 5: Eclipsen tekstieditori.

Tämän opinnäytetyön koodi on kirjoitettu Eclipse for PHP Developers -pakkauksella, joka sisältää Web-kehittämiseen tarkoitetut työvälineet.

5 JÄRJESTELMÄN SUUNNITTELU

5.1 Aloituspalaveri

Projektin aluksi järjestettiin palaveri, jossa kartoitettiin lähtötilanne ja uuden järjestelmän vaatimukset. Uuden järjestelmän tuli tarjota ratkaisu samankaltaisten projektien sisältämien tehtävien vertailuun. Tavoitteena oli saada esitettyä projektidataa kaavioissa, joita voitaisiin hyödyntää tulevien projektien suunnittelussa ja asiakastapaamisissa resurssienkäytön suunnitteluun ja hintatarjousten perusteluun.

Lähtötilanteessa Medusaworksin käytössä oli tuntikirjausjärjestelmä, jonka avulla hallinnoidaan suoritettuja työtunteja ja niiden laskutusta. Järjestelmän puutteena oli projektidatan hajanaisuus. Asiakkaat, asiakkaiden projektit ja niiden sisältämät tehtävät esitetään järjestelmässä yhdessä hierarkkisessa listassa. Suhteita eri projektien sisäisillä samankaltaisilla tehtävillä ei ole, minkä vuoksi projektien vertailuun ei ole helppoa tapaa.

Suunnittelupalaverin tuloksena päädyttiin projektien tehtävien nimeämiskäytännön yhtenäistämiseen. Käyttämällä kaikissa projekteissa samoja tehtävien nimiä saadaan luotua niiden välille vertailun mahdollistava yhtymäkohta. Palaverin pohjalta laadittiin vaatimuslista järjestelmään toteutettavista ominaisuuksista.

5.2 Järjestelmän vaatimukset

Projektinhallintajärjestelmän vaatimuksiin katsottiin kuuluvan vertailutoiminnallisuuden lisäksi tuntikirjausjärjestelmään kuuluvat toiminnallisuudet.

Projekteihin tulee voida lisätä tehtäviä nopeasti ja helposti samassa näkymässä, jossa sen sisältämät tehtävät esitetään. Tehtävien järjestystä pitää pystyä muuttamaan, ja saman näkymän tulee mahdollistaa tuntien kirjaaminen tehtäviin.

Itse projektien luonnissa on tavoitteena käyttää valmiita projektimalleja. Uuden projektin luomisen tulee tapahtua omassa näkymässään. Projektimalleja tulee voida helposti muokata lisäämällä niihin tehtäviä tai jättämällä niitä pois.

Tärkeimpänä uutena ominaisuutena järjestelmän tulee mahdollistaa projektien keskinäinen vertailu. Tätä tarkoitusta varten pitää toteuttaa jonkinlainen raportointinäkymä.

Laajaa vaatimusmäärittelydokumenttia ei ketterän kehityksen periaatteita noudattaen laadittu, sillä äkkinäisiin muutoksiin oltiin varauduttu.

6 JÄRJESTELMÄN KEHITYS JA TOIMINTA

6.1 Testiympäristön pystyttäminen

6.1.1 XAMPP:n asennus

Ennen kehittämisen aloittamista pystytin testiympäristön. Asensin Apache-palvelimen sisältävän XAMPP for Windows -paketin kotikoneelleni. Asennus sujui ilman ongelmia; Windows-ympäristössä on mahdollista käyttää asennusohjelmaa, joka toimii kuten muut vastaavat ohjelmat.

6.1.2 Palvelimen konfigurointi

XAMPP:n asentamisen jälkeen konfiguroin palvelimen. Oletuksena Apachen httpd.conf-tiedoston DocumentRootiksi on asetettu XAMPP-hakemiston sisältämä htdocs-hakemisto. Vaihdoin arvoksi Eclipsen työtilan hakemiston, jolla osoitteen http://localhost sai osoittamaan käyttämääni työtilaan. Järjestely toimi hyvin, mutta useissa eri hakemistoissa sijaitsevien projektien varalta päätin alkaa käyttämään virtuaali-isäntiä.

Apachen httpd-vhosts.conf-tiedostoon (xampp\apache\conf\extra) voi lisätä useita virtuaali-isäntiä. Konfiguroinnin jälkeen tiedosto näytti tältä:

```
NameVirtualHost *

<VirtualHost *>

DocumentRoot "C:\xampp\htdocs"

ServerName localhost

</VirtualHost>

<VirtualHost *>

DocumentRoot "E:\Workspace\procession"

ServerName procession.local
```

```
<Directory "E:\Workspace\procession">

DirectoryIndex index.php

AllowOverride All

Order allow,deny

Allow from 127.0.0.1

</Directory>

</VirtualHost>
```

Tämän jälkeen lisäsin Windowsin hosts-tiedostoon (Windows\System32\drivers\etc) seuraavan rivin:

```
127.0.0.1       procession.local
```

Nyt osoite http://procession.local/ osoitti httpd-vhosts.conf-tiedostossa määriteltyyn Eclipsen työtilaani.

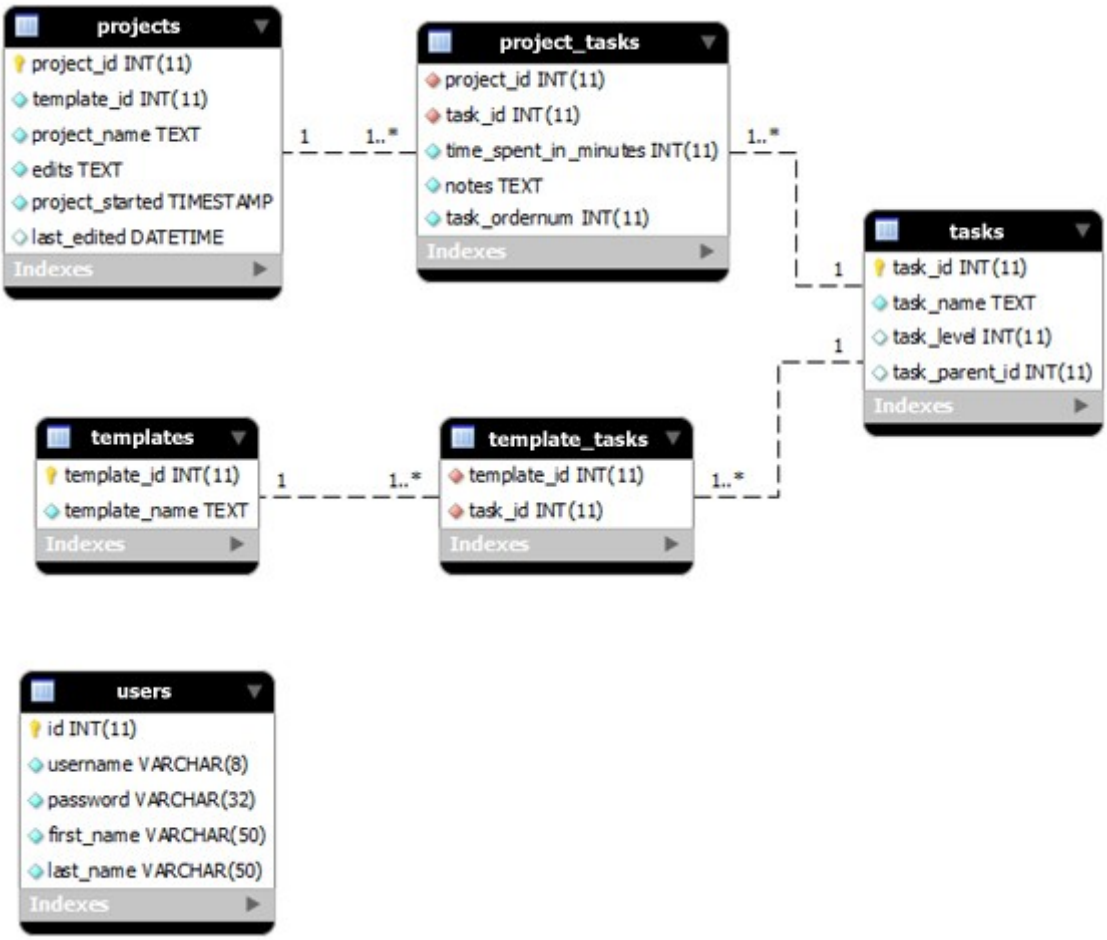
6.2 Tietokanta

Tietokannan suunnittelussa piti ottaa huomioon tehtävämallit. Niille ja niiden sisältämille tehtäville pitää olla omat taulunsa, mutta niiden ei tarvitse säilyttää tietoa käytetystä ajasta.

Tietokanta koostuu kuudesta taulusta:

- projects (projektit)
- templates (projektimallit)
- tasks (tehtävät)
- project_tasks (projektien sisältämät tehtävät)
- template_tasks (projektimallien sisältämät tehtävät)
- users.(käyttäjät)

Project_tasks ja template_tasks ovat liitostauluja, jotka yhdistävät tehtävät ja projektit tai tehtävät ja projektimallit toisiinsa.



Kuvio 1: Tietokannan relaatiomalli.

Projekteista tallennetaan id:n ja nimen lisäksi aloitusajankohta ja kaikki yksittäiset muokkaukset sekä viimeisimmän muokkauksen ajankohta raportteja varten. Projektien tehtävien ajat tallennetaan minuutteina, joista on helppo tehdä käännös tunneiksi ja minuuteiksi tarvittaessa.

Tietokantasuunnittelun yhteydessä kävi ilmi eräs ketterän kehityksen huonoista puolista. Nopean kehittämisen ja vähäisen suunnittelun vuoksi tietokanta piti kehityksen loppuvaiheessa suunnitella uudestaan. Tämä johtui heikosta

alkuperäisestä toteutuksesta, jota rakensin kehityksen edetessä ja näkymiä lisätessäni. Tämän kokoisessa projektissa menetetty aikamäärä ei kuitenkaan ollut kovin suuri. Se merkitsi kuitenkin noin viikon verran lisätyötä sovittaessani kaikki toiminnallisuudet uudelleen jokaiseen näkymään.

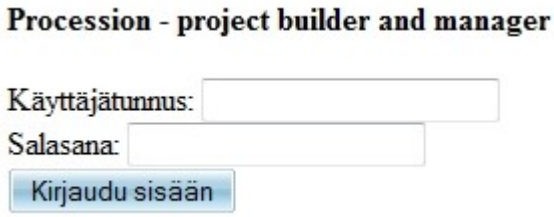
6.3 Kirjautuminen

Koska käyttäjät kuuluivat järjestelmän vaatimuksiin, ensimmäinen toteutettu näkymä oli kirjautumislomake. Käyttäjän painaessa kirjaudu sisään -painiketta lomake lähetetään checklogin.php-tiedostoon tarkistamista varten.

Checklogin.php:ssa syötetyt tiedot ajetaan mysql_real_escape_string-funktion läpi, minkä jälkeen tarkastetaan, löytyykö tietokannan users-taulusta syötettyä käyttäjänimeä ja salasanaa. Salasana on tallennettu tietokantaan md5-summana.

Jos käyttäjä ja salasana ovat olemassa, rekisteröidään istunto ja käyttäjä ohjataan index.php:hen eli pääsivulle. Muussa tapauksessa käyttäjä palautetaan takaisin kirjautumissivulle.

Jokainen kirjautuneena olemista vaativa sivu sisältää require-lauseen, joka sisällyttää checkifloggedin.php-tiedoston sivulle. Tämä tiedosto tarkastaa, onko käyttäjän sessio rekisteröity. Jos ei ole, se käyttäjä palautetaan kirjautumissivulle.



Kuva 6: Kirjautumislomake.

6.4 Tietolaatikko

Nopeaa navigointia varten jokaisen sivun vasempaan alalaitaan sisällytetään include-lauseella tietolaatikko, josta näkee tiedon ohjelmiston versionumerosta ja kirjautuneesta käyttäjästä. Se sisältää myös linkit etusivulle (projektit-näkymä) ja käyttäjät-sivulle sekä uloskirjautumislinkin.

Tietolaatikko sijaitsee erillisessä PHP-tiedostossa. Käyttäjänimi tulostetaan PHP:n $_SESSION-taulukosta. Laatikon sisältönä on lista, johon on helppo lisätä uusia linkkejä tarpeen vaatiessa tai uusia ominaisuuksia kehitettäessä.

Kuva 7:
Tietolaatikko.

6.5 Projektinäkymä

Järjestelmän aloitusnäkymä on projektinäkymä, joka listaa kaikki projektit ja niiden aloitusajankohdan, viimeisimmän muokkauksen ajankohdan sekä niiden käyttämän projektimallin. Tämä ja muut näkymät sisältävät tiedostoidensa alussa require-lauseet, joilla sisällytetään tietokannan konfigurointitiedosto ja tarvittavat JavaScript-tiedostot sivulle.

Projektit esitetään listassa, joka täytetään tietokannan projects-taulun sisällöllä. Projektit järjestetään aakkosellisesti ja niiden listaelementin id-attribuutiksi asetetaan projektia luotaessa annettu id-numero, joka on myös tallennettu tietokantaan. Jokaisen projektin nimi on linkki tehtävänäkymään.

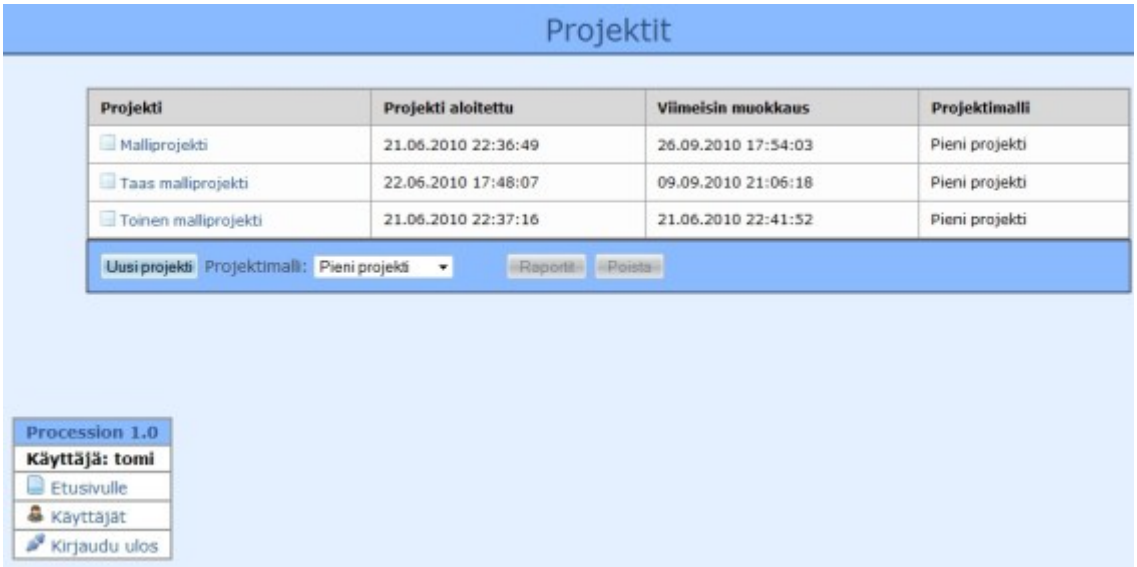
Projektinäkymä sisältää projektien alla painikkeen uuden projektin luomiseen alasvetovalikosta valitusta projektimallista. Näiden vieressä on raportit- ja poista -painikkeet, joiden käyttäminen vaatii haluttujen projektien vieressä olevan valintaruudun aktivoimista.

Raportit-painiketta painettaessa valintalaatikoin valituista projekteista kerätään JavaScript-taulukkoon valittujen projektien id-attribuutit. Kerätty taulukko asetetaan piilotettuun lomakkeeseen, joka lähetetään charts.php-tiedostoon, jonka toiminnallisuus on kuvattu tarkemmin raportointinäkymä-osiossa.

Poista-painikkeella poistetaan halutut projektit. Valittujen projektien id-attribuutit kerätään taulukkoon, joka lähetetään deleteproject.php-tiedostoon Ajax-kyselyllä. Tiedosto poistaa jokaisen valitun projektin rivin projects-taulusta ja kaikki projektiin liittyvät rivit project_tasks-taulusta.

Koska järjestelmän varsinainen käyttöönotto ei tapahdu opinnäytetyön kirjoittamisen aikana ja asiakkaista kerättyä uuteen järjestelmään soveltuvaa dataa ei ole saatavilla, loin havainnollistavia kuvia varten kolme esimerkkiprojektia.

Kuvassa 8 on esitetty kolme esimerkkiprojektia listaava projektit-näkymä.



Kuva 8: Projektit-näkymä.

6.6 Käyttäjät

Käyttäjät-näkymä listaa käyttäjät täyttämällä käyttäjälistan users-taulun sisällöllä ja sitä kautta voi lisätä uuden käyttäjän. Lisäyslomake sisältää JavaScript-tarkistuksen; kaikki kohdat ovat pakollisia. Lisäksi käyttäjänimessä saa olla vain alfanumeerisia merkkejä. Tämän tarkistamiseen käytetään säännöllistä lauseketta:

```
regexp = /^([a-zA-Z0-9]+)$/;
```

Lisää käyttäjä -painike lisää käyttäjän users-tauluun Ajax-kutsulla ja tulee aktiiviseksi vasta kun kaikkien kenttien sisältö vastaa säännöllistä lauseketta. Syötetyt tiedot lähetetään addusertodb.php-tiedostoon, jossa aluksi ajetaan lähetetyt tiedot mysql_real_escape_string-funktion läpi ja tarkastetaan, onko käyttäjänimi jo olemassa. Jos ei, suoritetaan tietokantapäivitys ja palautetaan HTML-muodossa käyttäjänimi, etunimi ja sukunimi:

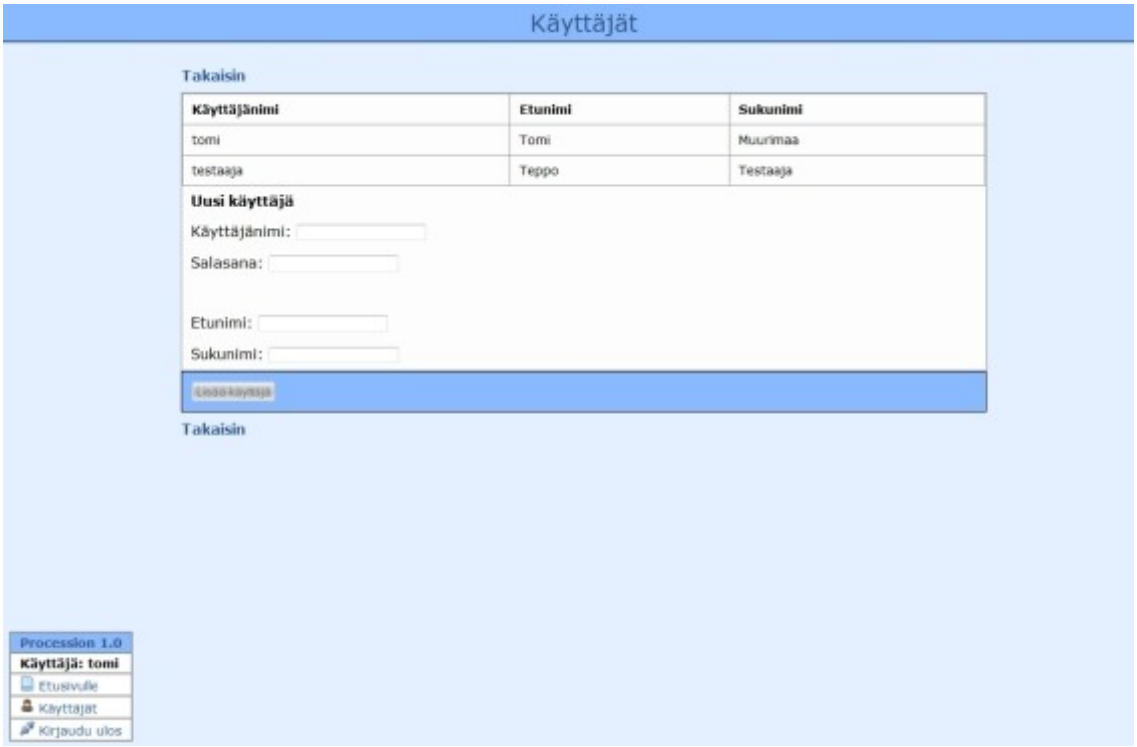
```
echo "<div id='newusername'>".$username."</div>";

echo "<div id='newfirstname'>".$first_name."</div>";

echo "<div id='newlastname'>".$last_name."</div>";
```

Tiedot päivitetään käyttäjälistaan lukemalla ne palautetusta datasta. Koko funktio käyttäjän lisäämiseen on esitetty alla.

```
$("#newuserbutton").click(function() {

            username = $("#uname").val();

            password = $("#pword").val();

            first_name = $("#fname").val();

            last_name = $("#lname").val();

            $.ajax({

                        type: "POST",

                        url: "addusertodb.php",
```

```
                    data: "username=" + username + "&password=" +
                    password + "&first_name=" + first_name +
                    "&last_name=" + last_name,

                    success: function(data) {
                              var $response=$(data);
                              var newusername =
                              $response.filter
                              ("#newusername").text();
                              var newfirstname =
                              $response.filter
                              ("#newfirstname").text();

                              var newlastname =
                              $response.filter
                              ("#newlastname").text();

                              $("#userinfotable >
                              tbody:last").append("<tr><td>" +
                              newusername + "</td><td>"  +
                              newfirstname + "</td><td>" +
                              newlastname + "</td></tr>");

                              alert("Käyttäjä " + newusername + "
                              lisätty.");

                    }

          });

});
```

Kuvassa 9 on esitetty käyttäjät-näkymä, kun lisättyjä käyttäjiä on kaksi.



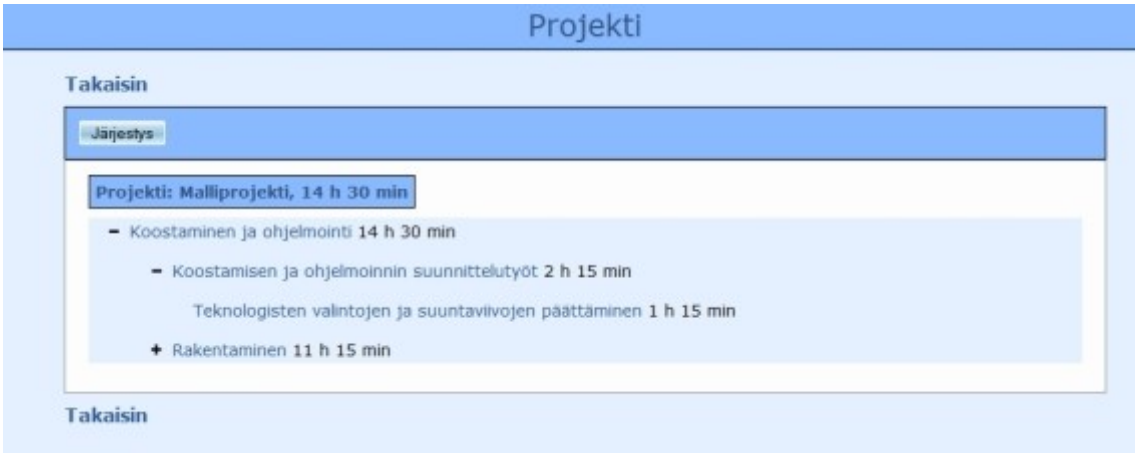
Kuva 9: Käyttäjät-näkymä.

6.7 Tehtävänäkymä

6.7.1 Rakenne

Tehtävänäkymä esittää yksittäisen projektin sisältämät tehtävät ja niihin käytetyt tuntimäärät. Jokaisen tehtävän alitehtävät asetetaan tietokantaan tallennettujen järjestysnumeroidensa mukaisesti ul-listan sisään, jolloin niitä voidaan käsitellä omina kokonaisuuksinaan. Tämä mahdollistaa järjestyksen muuttamisen ja tasojen piilottamisen jQueryä käyttäen.

Kuvassa 10 on esitetty projektin tehtävänäkymä. Miinusmerkkisten tehtävien alitehtävät ovat näkyvissä, plusmerkkisillä on piilotettuja alitehtäviä.



Kuva 10: Tehtävänäkymä.

Projektin kokonaistuntimäärä saadaan laskemalla ylimmän tason tehtävien yhteinen tuntimäärä, sillä jokaisen tehtävän alitehtävien tuntimäärät lisätään kyseisen tehtävän tuntimäärään. Esimerkiksi yllä olevan kuvan tilanteessa koostamisen ja ohjelmoinnin suunnittelutöihin on merkitty yksi tunti, mutta kuvassa esitettävä kokonaisaika on kaksi tuntia viisitoista minuuttia kun alitehtävän aika lasketaan mukaan.

Projektin tehtävät noudetaan näkymään SQL-lauseella

```
SELECT tasks.task_id, tasks.task_name, tasks.task_parent_id,
tasks.task_level, project_tasks.task_ordernum

FROM project_tasks

INNER JOIN tasks ON tasks.task_id = project_tasks.task_id

INNER JOIN projects ON projects.project_id = project_tasks.project_id

WHERE project_tasks.project_id = $project,
```

jossa $project-muuttuja on projektin nimi.

6.7.2 Tunnisteet

Tehtävien li-elementtien id-attribuutit ovat muotoa ”task_id-numero”, esimerkiksi ”task_3”. Vastaavan elementin alitehtävien sisältämän ul-listan nimi olisi muotoa ”ul_task_3”.

Id-numerot haetaan tietokannasta, jonne on tallennettu myös jokaisen tehtävän isäntätehtävän id-attribuutti, tehtävän taso ja tehtävän järjestysnumero. Näiden perusteella jokaiselle li-elementille annetaan myös luokat ”child_of_isäntäelementin id-numero”, ”level_level-numero” ja ”ordernum_järjestysnumero”. Kokonaisuudessaan yhden li-elementin attribuutit voisivat näyttää seuraavalta:

```
li #task_4 .level_3 child_of_3 ordernum_1 task.
```

Task-luokka lisätään automaattisesti li-elementeille. Sen lisäksi käytetään vielä luokkia hasChildren ja opened määrittelemään tehtävän edessä näytettävä plus- tai miinusmerkki.

6.7.3 Tehtävien näyttäminen ja piilottaminen

Task-luokkaan on sidottu jQueryn click-funktio, jonka perusteella ul-listojen sisältämät tehtävät näytetään tai piilotetaan.

```
$(".task").click(function(e) {
            $(this).children("ul").toggle(200);
            if ($(this).hasClass("hasChildren opened")) {
                        $(this).removeClass("opened");
            }
            else if ($(this).hasClass("hasChildren")) {
                        $(this).addClass("opened");
            }
            e.preventDefault();
```

```
            e.stopPropagation();
});
```

6.7.4 Kontekstivalikko

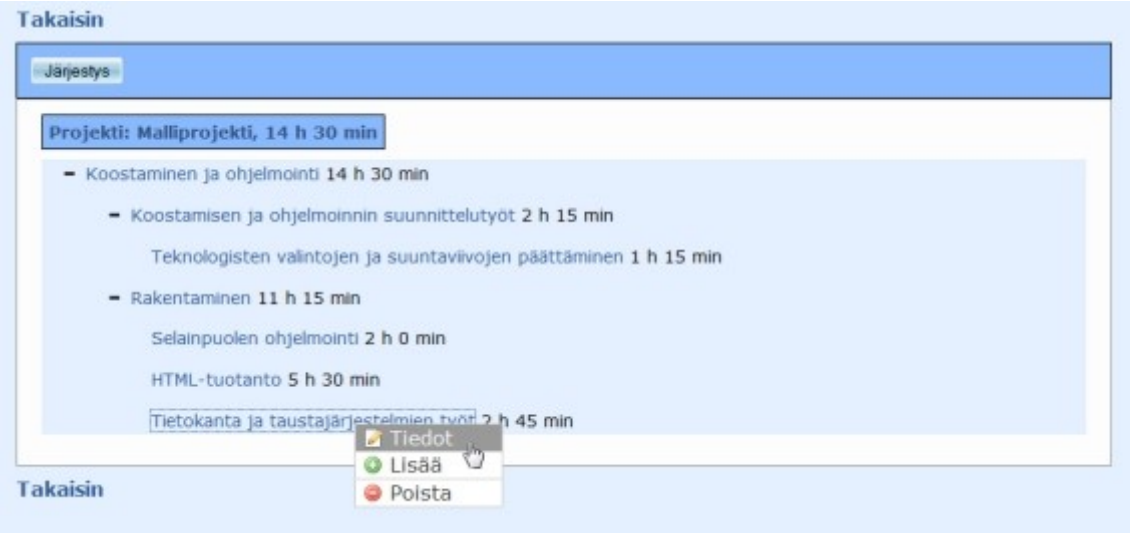
Tehtävänäkymän toiminnallisuudet ovat kontekstivalikossa. Valikko on toteutettu jQueryn Context Menu -liitännäisellä. Kontekstivalikko on itsessään HTML-lista, joka kirjoitetaan sitä käyttävälle sivulle. Kontekstivalikon lisäävä funktio:

```
function addcontextmenu(target) {
            target.contextMenu({
                        menu: "options"
            },
            function(action, el) {
                        if (action == "info") {
                                    action_info(el);
                        }
                        else if (action == "add") {
                                    action_add(el);
                        }
                        else if (action == "delete") {
                                    action_delete(el);
                        }
             });
            }
```

Funktiossa target on elementti, johon kontekstivalikko sisällytetään. Menu on HTML-listan id-attribuutti.

Toiminnallisuuksien sisällyttäminen kontekstivalikkoon säästää tilaa näytöllä, kun esimerkiksi erillistä valikkoa ei tarvitse tulostaa erikseen jokaisen tehtävän yhteyteen. Toteutuksesta käy esiin jQueryn laajennettavuus; valikkoliitännäinen oli valmiiksi olemassa ja se täytyi vain sisällyttää sitä käyttävälle sivulle.

Kuvassa 11 on tehtävänäkymän kontekstivalikko. Valikko ilmestyy valitun tehtävän kohdalla.



Kuva 11: Tehtävänäkymän kontekstivalikko.

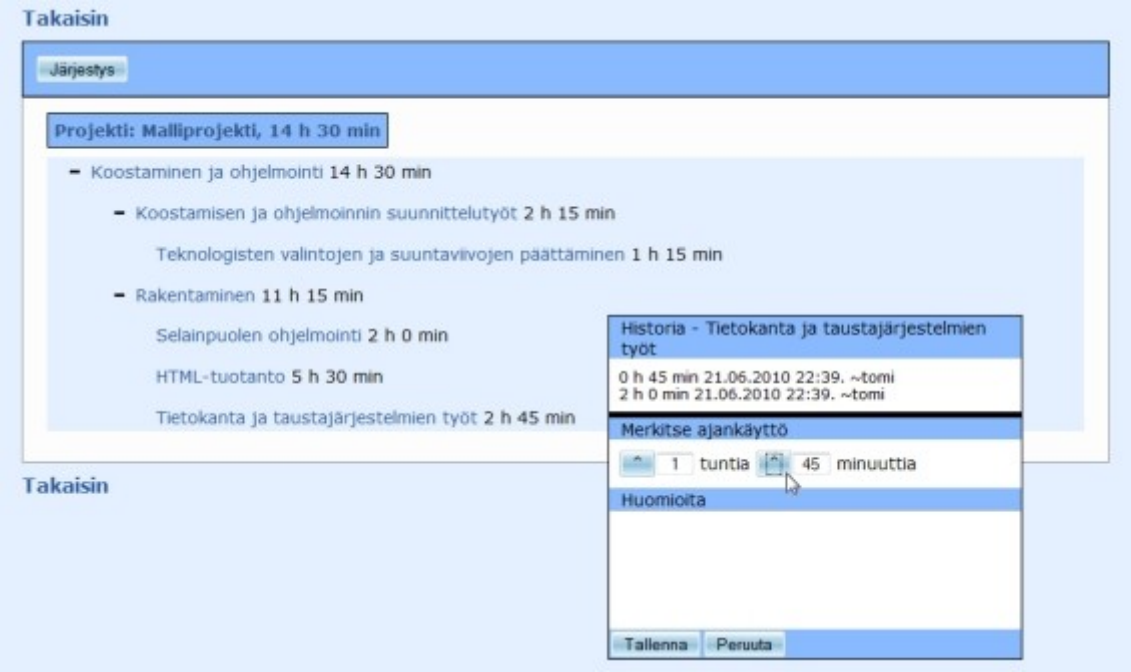
6.7.5 Tuntien kirjaaminen

Tuntikirjausikkuna aukeaa valitsemalla halutun tehtävän kontekstivalikosta ”Tiedot”. Aika merkitään painikkeilla. Minuutteja lisätään 15 kerrallaan, 45:n jälkeen kentän arvo palaa nollaan. Painikkeihin sidotut funktiot on esitetty alla.

```
$("#hours_up_button").click(function() {

            if ( !$("#hours_input").val() ) {

                        $("#hours_input").val(0);
```

```
            }

            $("#hours_input").val( parseInt($("#hours_input").val())+1);

});

$("#minutes_up_button").click(function() {

             if ( !$("#minutes_input").val() ) {

                        $("#minutes_input").val(0);

             }



            $("#minutes_input").val( parseInt($
            ("#minutes_input").val())+15);

            if ( $("#minutes_input").val() == 60 ) {

                        $("#minutes_input").val(0);

            }

});
```

Kuvassa 12 on esitetty tuntien kirjaaminen tuntikirjausikkunassa. Ikkunan yläosassa näkyy aiemmin tehtyjen merkintöjen tuntimäärä ja ajankohta.



Kuva 12: Tuntien kirjaaminen.

Tallenna-painiketta painaessa suoritetaan Ajax-kutsu. Aikakentistä kerätyt arvot lähetetään savenotes.php-tiedostoon, jossa tehdään tietokannan päivitys. Tuntimäärä kerrotaan 60:lla ja lisätään minuuttien määrään. Tämä arvo päivitetään tietokantaan lisäämällä se aiempaan minuuttimäärään.

Lisäksi päivitetään sama aikamäärä kyseisen tehtävän isäntätehtävien riveihin hakemalla for-silmukassa tehtävän parent_id siltä riviltä, joka on nykyisen tehtävän id. For-silmukka, jota toistetaan tehtävän tasojen mukainen määrä:

```
for ($i=0; $i<$currentTaskLevel; $i++) {

            $db->query("

            UPDATE project_tasks

            SET time_spent_in_minutes = time_spent_in_minutes +
            '$totaltime'

            WHERE project_id = '$currentProject'

            AND task_id = '$currentTaskParent'

            ;") or die(mysql_error());
```

```
            if (!in_array($currentTaskParent, $ancestors) &&
            $currentTaskParent != "") {

                        array_push($ancestors, $currentTaskParent);

            }


            //hae seuraavan isäntätehtävän id

            $db->query("

            SELECT task_parent_id

            FROM tasks

            WHERE task_id = '$currentTaskParent'

            ") or die(mysql_error());


            while ($line = $db->fetchNextObject()) {

                        $currentTaskParent = $line->task_parent_id;

            }
}
```

$ancestors-taulukko kerää tehtävien id:t, jotta ne voidaan palauttaa sivulle. Niiden perusteella päivitetään tehtävien ajat dynaamisesti selainpuolella.

Lopuksi päivitetään projektitaulun kyseisen projektin rivin edits-sarakkeeseen tapahtuman aikaleima raporttien käyttöä varten.

6.7.6 Tehtävän lisääminen

Tehtävä lisätään kontekstivalikon lisää-kohdasta. Valinta avaa lisäämisikkunan, johon uuden tehtävän nimi syötetään. Kuvassa 13 on lisäämisikkuna, johon on kirjoitettu lisättävän tehtävän nimi, ”Testi”.

Kuva 13: Tehtävän lisääminen.

Tallenna-painiketta painettaessa suoritetaan Ajax-kutsu. Newtask.php-tiedostoon lähetetään kyseisen JavaScript-muuttujaan projektia valitessa tallennettu projektin id, valitun tehtävän id, tehtävän taso ja järjestysnumero li-elementin tunnisteista sekä lisättävän tehtävän nimi.

Palvelinpuolella päivitetään tehtävätauluun uuden tehtävän nimi, tehtävän isäntätehtävän id ja tehtävän tasonumero. Tehtävä saa oman id:n automaattisesti; se saadaan mysql_insert_id-funktiolla ja tulostetaan newtaskid-nimisen div-elementin sisään, josta se luetaan jQuerylla. Id asetetaan uuden li-elementin id:ksi tehtäessä selainpuolen dynaaminen päivitys.

6.7.7 Tehtävän poistaminen

Tehtävä poistetaan kontekstivalikon poista-kohdasta. Kohdan valinta avaa varmistusikkunan. Jos tehtävällä on alitehtäviä, sitä ei voida poistaa. Kuvassa 14 esitetään Testi-nimisen tehtävän poistaminen.

Kuva 14: Tehtävän poistaminen.

Kun poistaminen on varmistettu, tehdään Ajax-kutsu deletetask.php-tiedostoon. Tietokantaan päivitetään muuttuneet järjestysnumerot ja tehtävä poistetaan. SQL-kysely, joka päivittää järjestysnumerot on esitetty alla.

```
$db->query("

UPDATE project_tasks, tasks

SET project_tasks.task_ordernum = project_tasks.task_ordernum - 1

WHERE tasks.task_parent_id = $taskParent

AND project_tasks.task_id = tasks.task_id

AND project_id = $currentProject

AND task_ordernum > $removedOrdernum

") or die(mysql_error());
```

Kyselyssä $taskParent on tehtävän isäntätehtävä, $currentProject nykyinen projekti ja $removedOrdernum poistettavan tehtävän järjestysnumero.

6.7.8 Tehtävien järjestyksen muuttaminen

Tehtävien järjestyksen muuttaminen oli yksi toiminnallisista vaatimuksista. Perusteena oli kronologian säilyminen; tehtävälistaa luetaan ylhäältä alas ja joskus on tarpeen lisätä tehtävä listan keskivaiheille. Uusi tehtävä lisätään automaattisesti listan loppuun.

Tehtävien järjestystä muutetaan tehtävänäkymän järjestys-painikkeesta. Järjestystä voi muuttaa vain tasoittain, toisin sanoen siirrettävän tehtävän isäntätehtävä pysyy samana. Järjestämiseen käytetään jQueryn sortable-liitännäistä.

Kuvassa 15 järjestys-painiketta on painettu, ja liikuteltavissa olevat tehtävätasot on merkitty reunaefektillä.

Kuva 15: Järjestyksen muuttaminen.

Painiketta painettaessa kaikista ul-elementeistä tehdään järjestettäviä, ja niihin sidotut muut toiminnallisuudet poistetaan käytöstä. Jokainen lista saa myös reunaefektin. Aina kun tehtävää on siirretty, uudet järjestysnumerot päivitetään selainpuolella asettamalla järjestetyn listan elementtien järjestysnumeroksi niiden indeksinumero. Tämän jälkeen tehdään Ajax-kutsu, ja lähetetään kyseisen listan kaikkien tehtävien id-attribuutit ja uudet järjestysnumerot

updateordernums.php-tiedostoon. Id:t ja järjestysnumerot kerätään omiin taulukoihinsa ja päivitetään tietokantaan for-silmukassa:

```
for ($i=0;$i<count($tasks);$i++) {

            $db->query("

            UPDATE project_tasks

            SET task_ordernum = '$ordernums[$i]'

            WHERE project_id = '$projectid'

            AND task_id = '$tasks[$i]'

            ;") or die(mysql_error());

}
```

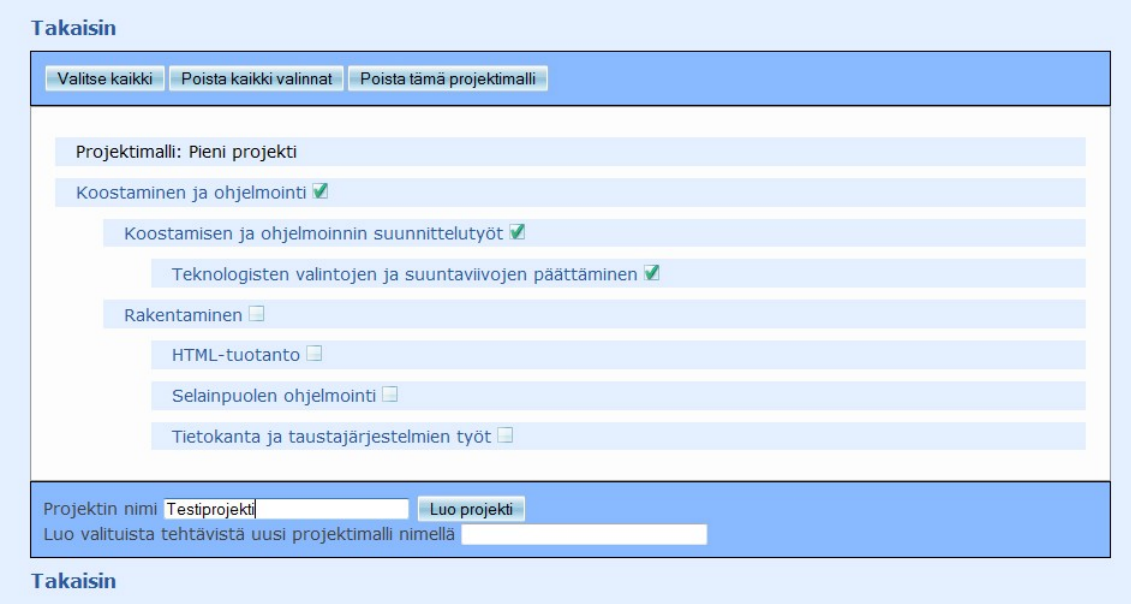
Järjestäminen lopetetaan painamalla painiketta uudestaan, jolloin reunaefekti poistetaan ja kontekstivalikko sidotaan tehtäviin uudestaan.

6.8 Projektin luominen

Projektit luodaan projektimallien pohjalta. Projektimallien taustalla on ajatus käyttää samankaltaisten projektien pohjina samaa projektimallia, jolloin saadaan kerättyä vertailukelpoista dataa.

Uuden projektin luominen aloitetaan projektinäkymän uusi projekti -painikkeella. Alasvetovalikosta valitun projektimallin id-attribuutti lähetetään newproject.php-tiedostoon ja sen sisältämät tehtävät avataan luontinäkymään. Näkymä muistuttaa projektinäkymää, mutta kaikki tasot ovat koko ajan näkyvillä valintojen helpottamiseksi.

Kuvassa 16 esitetään Testiprojekti-nimisen projektin luonti. Kolme tehtävää on valittuna.



Kuva 16: Uuden projektin luominen.

Projektiin halutut tehtävät valitaan valintalaatikoita käyttäen. Vaikka olisi suositeltavaa käyttää aina muokkaamattomia projektimalleja valitsemalla kaikki tehtävät valitse kaikki -painikkeesta, joskus voi olla tarpeen käyttää lähes samanlaiseen projektiin osittain muokattua projektimallipohjaa. Tästä syystä päädyin mahdollistamaan vapaammin valittavat tehtävät myös saman projektimallin sisällä.

Toinen syy valintalaatikoin valittaviin tehtäviin on uuden projektimallin luonti, joka tehdään samassa näkymässä. Uutta projektia luotaessa voidaan luoda myös uusi projektimalli valituista tehtävistä kirjoittamalla sen nimi tekstikenttään.

Luo projekti -painiketta painettaessa kerätään tehtävien id-attribuutit taulukkoon käyttämällä jQueryn :checked-valitsijaa. Tehtävät lähetetään Ajax-pyynnöllä createnewproject.php-tiedostoon, joka lisää projektin projektitauluun yksinkertaisella SQL-kyselyllä.

Projektin tehtävien lisääminen projektitehtävätauluun on monimutkaisempaa, sillä niistä tallennetaan myös järjestysnumero, jota projektimallinäkymässä ei ole saatavilla. Tämä johtuu siitä, että listat voivat sisältää vaihtelevan määrän tehtäviä.

Tehtävien id:t on tallennettu $taskids-taulukkoon ja niitä vastaavat isäntätehtävät $taskparentids-taulukkoon vastaaviin indekseihin. Kunkin tehtävän oikean järjestysnumeron määrittämiseen käytetään aputaulukoita, joista toinen sisältää lisättyjen tehtävien isäntätehtävien id:t ja toinen niiden esiintymiskerrat. Alla on esitetty kommentoitu koodi, joka suorittaa järjestysnumeroiden laskemisen ja tehtävien tietokantaan lisäämisen.

```
$ordernum = 1; //oletuksena listan ensimmäinen tehtävä

$parents = array(); //tulee sisältämään tehtävien isäntätehtävien id:t

$times = array(); //tulee sisältämään isäntätehtävien esiintymiskerrat

for ($i=0; $i<count($taskids); $i++) { //suoritetaan tehtävien
            lukumäärän verran

            if (!in_array($taskparentids[$i], $parents)) {
            //tarkastetaan, onko läpikäytävän tehtävän isäntätehtävä
            lisätty taulukkoon, tässä tapauksessa ei

                        $ordernum = 1; //isäntätehtävän ensimmäisen
                                    alitehtävän järjestysnumero on 1

                        array_push($parents, $taskparentids[$i]);
                        //lisätään $parents-taulukkoon läpikäytävän
                        tehtävän isännän id

                        array_push($times, $taskparentids[$i]);
                        //lisätään $times-taulukkoon läpikäytävän
                        tehtävän isännän id...

                        array_push($times, 1); //...ja sen seuraavaan
                        alkioon esiintymiskerrat, tässä tapauksessa 1

            }

            else {

                        $key = array_search($taskparentids[$i], $times) +
                        1; //etsii $times-taulukosta annetun
                        isäntätehtävän id:n ja palauttaa sen indeksin+1
                        (kyseisen tehtävän esiintymiskerrat)
```

```
                $times[$key] = $times[$key]+1; //päivittää
                esiintymiskerrat

                $ordernum = $times[$key]; //asettaa uuden
                järjestysnumeron arvon

        }

        //lisää tehtävän tauluun

        $db->query("

        INSERT INTO project_tasks (project_id, task_id,
        time_spent_in_minutes, notes, task_ordernum)

        VALUES ($newprojectid, $taskids[$i], 0, '', $ordernum)

        ;") or die(mysql_error());

}
```

Tämän jälkeen tietokantaan lisätään projektimalli, mikäli sen nimi on annettu. Lisääminen on huomattavasti yksinkertaisempaa, sillä järjestysnumeroita ei tarvitse ottaa huomioon:

```
if ($createtemplate) {

        $db->query("

        INSERT INTO templates (template_name)

        VALUES ('$newtemplatename')

        ;") or die(mysql_error());


        $newtemplatetid = mysql_insert_id();


        for ($i=0; $i<count($taskids); $i++) {

                $db->query("

                INSERT INTO template_tasks (template_id, task_id)
```

```
                    VALUES ($newtemplatetid, $taskids[$i])

                    ;") or die(mysql_error());

            }

}
```

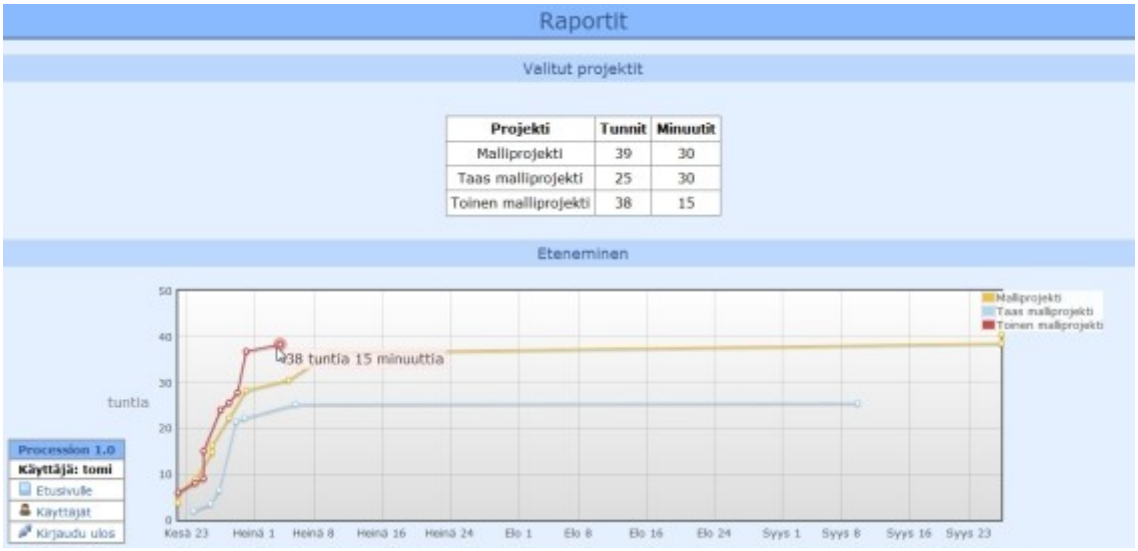
6.9 Raportointinäkymä

Raportointinäkymää käytetään ensisijaisesti samankaltaisten projektien vertailuun. Perusajatuksena on samannimisten tehtävien vertailu. Raportointinäkymä ei aseta rajoitteita vertailtavien projektien määrälle. Näkymä esittää kuitenkin käytetyn ajan ja etenemiskäyrän vain yhdestäkin valitusta projektista.

Kuvaajien luomiseen käytetään jQueryn flot-kirjastoa. Näkymä esittää projekteihin käytetyn kokonaisajan taulukossa ja pylväsdiagrammina, etenemiskäyrän, eri projektien samojen tehtävien keskimääräisen käytetyn ajan ja eri projektien tehtäviin käytetyt ajat vierekkäisinä ja päällekkäisinä pylväsdiagrammeina. Näkymää avattaessa siihen lähetetään valittujen projektien id:t, joiden perusteella tehdään tietokantahaut. Projektien id:t tallennetaan myös JavaScript-taulukkoon selainpuolen ohjelmointia ja kuvaajien luontia varten.

6.9.1 Etenemiskäyrä

Etenemiskäyrä esittää pystyakselilla projektiin käytetyn ajan ja vaaka-akselilla yksittäisten muokkausten ajankohdat. Taulukossa projektin yläpuolella on esitetty valitut projektit ja niihin käytetyt kokonaisajat. Mikäli projekteja on valittu vain yksi, näkymä esittää vain projektien ajat ja ensimmäisen kuvaajan.



Kuva 17: Projekteihin käytetyt ajat ja etenemiskäyrä.

Aikatiedot saadaan tietokannasta projektien time_spent_in_minutes-riveiltä. Minuuttiajat muutetaan tunneiksi ja minuuteiksi ennen taulukkoon tulostamista.

Kuvaajaa varten projektitaulun edits-sarakkeeseen on tallennettu jokaisen muokkauksen yhteydessä aikaleima. Jokaista projektia kohden näistä tiedoista luodaan PHP:lla dynaamisesti JavaScript-muuttuja:

```
echo "<script type='text/javascript'>";

            echo "var edits_".$i." = '".str_replace("'", "\'",
            $edits)."';";

echo "</script>";
```

Selainpuolella jokaisen projektin luodut taulukot käydään läpi ja niiden sisältö asetetaan niille luotavan div-elementin sisään. Tähän käytetään eval()-funktiota:

```
curredits = eval("edits_" + j);

//lisää tiedot edits_x diviin

$("#edits_"+j).html(curredits);
```

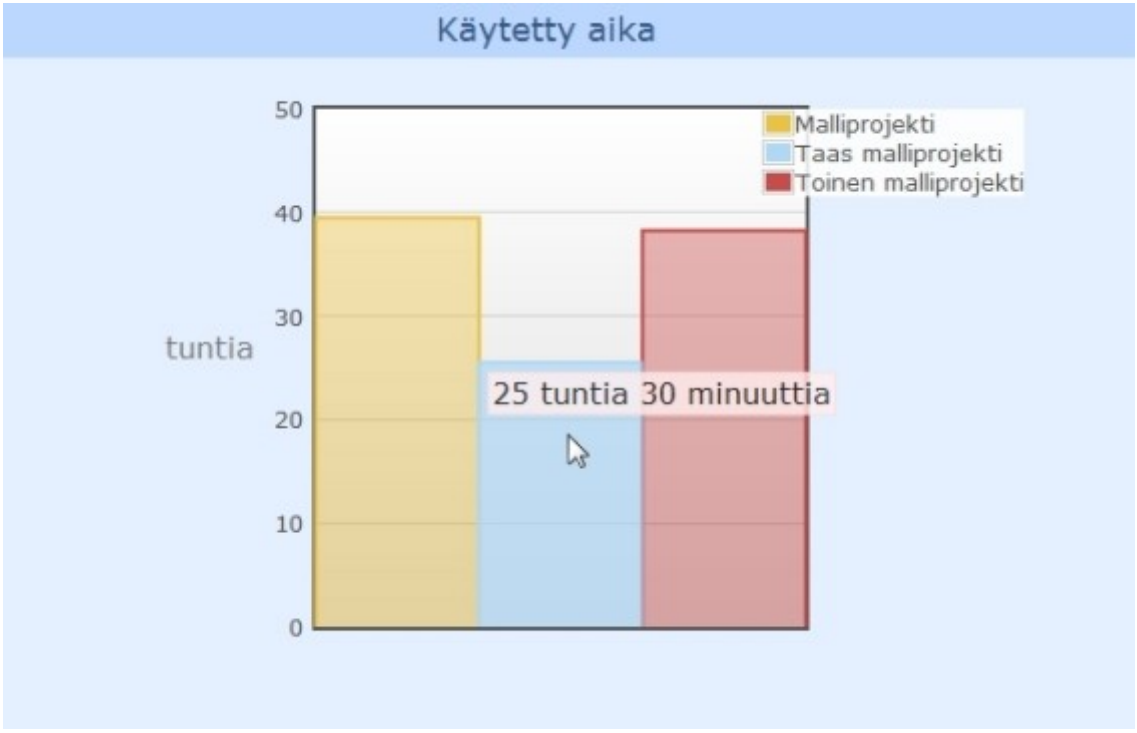
Edits-sarakkeeseen tallennetaan aikaleiman lisäksi siihen asti käytetty aika minuutteina span-elementin sisään. Aikaleima ja käytetty aika erotellaan

toisistaan tildellä(~). Tämän ansiosta eri muokkauskerrat voidaan erottaa toisistaan elementin perusteella.

Aikaleiman ja käytetyn ajan saa erilleen JavaScriptin split()-funktiolla. Nämä arvot syötetään jokaista projektia kohden for-silmukan sisällä flotille muodossa [[aikaleima, kulunut aika], [aikaleima, kulunut aika], [aikaleima, kulunut aika]]. Asteikot skaalautuvat automaattisesti.

6.9.2 Käytetty aika

Käytetyn ajan kuvaaja esittää projektit ja niihin käytetyn ajan pylväsdiagrammissa.



Kuva 18: Käytetyn ajan pylväsdiagrammi.

Jokaisen projektin kokonaisajat tallennetaan PHP-tiedostossa JavaScript-taulukkoon:

```
echo "<script type='text/javascript'>"; $edits)."';";
```

```
            echo "totaltime[".$i."] = '".$totaltime[$i]."';";
echo "</script>";
```

Flotin käyttämä datamerkkijono muodostetaan for-silmukassa:

```
for(i=0;i<numberofprojects;i++) {
            data2 += "{label: '" + projectnames[i] + "',  data:    [[" +
            i + "," + totaltime[i]/60 + "]]},";
}
```

Pylväisiin sidotaan flotin sisäänrakennettu plothover-funktio, jolloin näytetään kyseisen projektin tarkka kokonaisaika liikutettaessa hiiren osoitin pylvään päälle. Aika muutetaan datamerkkijonoon kirjoitetuista tunneista tunneiksi ja minuuteiksi.

6.9.3 Tehtävien keskimääräiset ajat

Tehtävien keskimääräiset ajat listataan taulukossa. Ajat ovat valittujen projektien sisältämien samojen tehtävien keskimääräisiä aikoja.

Jaettujen tehtävien vertailu

Jaetut tehtävät:

Tehtävän nimi	Keskimääräinen käytetty aika
Koostaminen ja ohjelmointi	12 tuntia 40 minuuttia
Koostamisen ja ohjelmoinnin suunnittelutyöt	2 tuntia 0 minuuttia
Teknologisten valintojen ja suuntaviivojen päättäminen	1 tunti 25 minuuttia
Rakentaminen	9 tuntia 35 minuuttia
HTML-tuotanto	5 tuntia 10 minuuttia
Selainpuolen ohjelmointi	1 tunti 20 minuuttia
Tietokanta ja taustajärjestelmien työt	2 tuntia 15 minuuttia

Kuva 19: Jaettujen tehtävien vertailutaulukko.

Koska projektissa käytetty MySQL-tietokanta ei tue intersect-komentoa, jolla saisi haettua eri projektien sisältämät samat tehtävät melko helposti, projektien jaettujen tehtävien määrittäminen on ratkaistu eri tavalla.

Ratkaisun keksiminen vei kauimmin yksittäisistä järjestelmän kehityksessä kohdatuista ongelmista. Tietokannan muuttamisen jälkeen yhteisten tehtävien määrittelyn ongelma piti ratkaista uudestaan.

Jokaisen projektin jokaisen tehtävän id tallennetaan yhteen taulukkoon, jota sen jälkeen tarkastellaan. Jos tehtävän id ilmenee taulukossa yhtä monta kertaa kuin projekteja on valittu vertailuun, se löytyy kaikista projekteista ja lisätään jaettujen tehtävien taulukkoon. Kommentoitu PHP-koodi tämän toteuttamiseen:

```
if (count($projects)>1) { //jos projekteja on yli yksi

            $tasks = Array(); //kaikki tehtävät

            $sharedtasks = Array(); //jaetut tehtävät

            for ($i=0;$i<count($projects);$i++) {

                        //hakee kaikkien projektien kaikkien tehtävien
                        id:t

                        $db->query("

                        SELECT DISTINCT task_id

                        FROM project_tasks

                        WHERE project_tasks.project_id = '$projects[$i]'

                        ");

                        //lisää id:t tehtävätaulukkoon

                        while ($line = $db->fetchNextObject()) {

                                    array_push($tasks, $line-
                                    >task_id);

                        }

            }

            //käy läpi tehtävätaulukon

            for ($i=0;$i<count($tasks);$i++) {
```

```
                    //taulukko, joka sisältää tehtävien
                    esiintymiskerrat

                    $acv = array_count_values($tasks);

                    $index = $tasks[$i];

                    //tarkistaa onko tehtävä taulukossa projektien
                    lukumäärän verran; jos on, se on jaettu

                    if ($acv[$index]==count($projects)) {

                              //lisää tehtävä jaettujen tehtävien
                              taulukkoon

                              array_push($sharedtasks, $tasks[$i]);

                    }

          }

}
```

6.9.4 Tehtävien pylväsdiagrammit

Tehtävien pylväsdiagrammit esittävät kunkin projektin jaettuihin tehtäviin käytetyt ajat vierekkäin ja päällekkäin.

Kuva 20: Tehtävien pylväsdiagrammit.

Tarvittava data kerätään PHP-tiedostossa luoduista span-elementeistä, joihin lisätään pilkulla erotettuna tehtävän nimi ja siihen käytetty aika. Elementtien luominen on esitetty alla.

```
for($i=0; $i<count($projects);$i++) {

            $db->query("SELECT projects.project_name,
            project_tasks.time_spent_in_minutes, tasks.task_name

            FROM projects

            JOIN project_tasks ON project_tasks.project_id =
            projects.project_id

            JOIN tasks ON tasks.task_id = project_tasks.task_id

            WHERE project_tasks.project_id = '$projects[$i]'

            AND (project_tasks.task_id = '$sharedtasks')

            ;") or die(mysql_error());

            $j=0;

            echo "<div id='sharedtaks_".$i."'>";

            while ($line = $db->fetchNextObject()) {


                        $currname[$j] = $line->task_name;


                        $timespent[$j] = $line->time_spent_in_minutes;


                        echo "<span class='sharedtask'>".
                        $currname[$j].", ".$timespent[$j]."</span><br
                        />";

                        $j++;

            }

            echo "</div>";
```

```
}
```

Luoduista elementeistä luetaan arvot jQueryllä ja niistä muodostetaan datamerkkijono flotin käyttöön. Samaa dataa käytetään molempien diagrammien muodostamiseen.

Molempiin diagrammeihin lisätään alapuolelle tehtävien nimet. Tämä tehdään ajamalla jQueryn append()-funktiota silmukassa tehtävien määrän verran:

```
$("#multibarchart").append("<td
class='chartxlabel'>"+multichartlabeltext+"</td>");
```

Kunkin selite-elementin oikea leveys lasketaan lukemalla jQueryllä sivun latauduttua luodun kaavion leveys muuttujaan ja jakamalla muuttuja tehtävien lukumäärällä.

```
chartwidth = $("#multibarchart").width();

divwidth = Math.floor(chartwidth/numoftasks)+"px";
```

Näin saatua lukua käytetään yksittäisten diagrammin alle lisättävien div-elementtien leveytenä, jolloin jokainen seliteteksti vie yhtäläisen verran tilaa.

```
$(".chartxlabel").css({"width":divwidth});
```

7 TUTKIMUKSEN TULOKSIA

Tämän tutkimuksen tavoitteena oli toteuttaa projektinhallintajärjestelmä verkkokäyttöliittymällä. Järjestelmä toteutettiin toimivaksi asti. Sitä tullaan kuitenkin kehittämään jatkossa ennen varsinaista käyttöönottoa; tämä johtuu siitä, että projektin edetessä yhä enemmän toimeksiantajan tarvitsemia toiminnallisuuksia tuli ilmi ja opinnäytetyön puitteissa toteutettavaa projektia oli rajattava.

Järjestelmän kehityksen aikana huomattiin, että nykyaikaisen projektinhallintajärjestelmän toteutus verkkopohjaisena helpottaa käyttöliittymän suunnittelua ja toteutusta PHP:hen ja JavaScriptiin perehtyneelle ohjelmoijalle.

Verkkopohjaisuus tarjoaa etunaan Ajaxin käyttämisen mukanaan tuoman dynaamisuuden; tehdyt muutokset voidaan päivittää ilman koko näkymän uudelleen lataamista. PHP mahdollistaa helpon tietokantojen käsittelyn.

JavaScriptin jQuery-kirjasto yksinkertaistaa sivun elementtien käsittelyä, mikä mahdollistaa esimerkiksi tehosteiden luomisen muutamalla koodirivillä. JQueryn liitännäiset soveltuvat monipuolisten käyttöliittymien luomiseen; kontekstivalikko on hyvä esimerkki toiminnallisuuksien piilottamisesta silloin, kun niitä ei tarvita.

Raportointijärjestelmälle oleellisten kuvaajien luonti onnistuu tehokkaasti JavaScriptin kirjastoilla. Tässä työssä käytettiin jQueryn flot-kirjastoa, joka mahdollistaa monipuolisten kuvaajien luomisen syötetystä datasta kohtuullisen vähällä vaivalla.

Ketterän ohjelmistokehityksen periaatteiden soveltaminen aiheutti projektissa jonkin verran ongelmia. Ongelmat liittyivät lähinnä suunnittelun puutteellisuuteen. Kehitystyö oli hyvin nopeaa ilman jatkuvaan dokumentointiin ja suunnitteluun kuluvaa aikaa. Osa tästä ajasta tosin kului nyt virheiden korjaamiseen.

Koin kuitenkin ketterien menetelmien soveltamisen minulle hyväksi työtavaksi, mahdollisesti siksi, että itse tekemäni virheet opettavat minua parhaiten

välttämään niitä jatkossa. Liiallisten suunnitelmien puute antaa myös tilaa luovuudelle, ja jatkuva yhteys toimeksiantajaan selkeyttää tavoitteita.

8 JOHTOPÄÄTÖKSET

8.1 Kriittinen arviointi

Järjestelmän toteutusta jälkeenpäin tarkasteltaessa tulee todenneeksi, että monta asiaa olisi voinut tehdä toisin.

Suunnitteluun olisi kannattanut käyttää enemmän aikaa. Yhden palaverin ja lyhyen rakenteellisen suunnittelun jälkeen pikaisesti toimeen ryhtyminen ja toiminnallisuuksien lisääminen yksi kerrallaan aiheutti sen, että joitakin toiminnallisuuksia piti sovitella jälkeenpäin toisiinsa, jotta ne sai toimimaan edellytetyllä tavalla.

Näkymien rakenteen paremmalla suunnittelulla olisi säästänyt paljon vaivaa ja järjestelmän rakenteesta olisi tullut siistimpi ja helpommin ylläpidettävä.

PHP:n olio-ominaisuuksien hyödyntäminen jäi tietokantaluokan käyttämiseen. Yksi ennen käyttöönottoa tehtävistä toimista onkin järjestelmän muuttaminen oliopohjaiseksi hyödyntämällä osittain Medusaworksin omia luokkakirjastoja.

8.2 Tutkimuksen hyödynnettävyys

Tutkimuksen tuloksena luodun järjestelmän toiminnallisuuksien kuvaukset selittävät projektinhallintajärjestelmän toimintaa. Toiminnallisuuksien kuvauksista on hyötyä jatkokehityksessä, kun on tarpeen ymmärtää järjestelmän toimintaa. Lisäksi kuka tahansa vastaavaa järjestelmää kehittävä voinee käyttää järjestelmän toiminnallisuuden kuvauksia hyödykseen suunnittelussa.

Medusaworks Oy:n edustajan Tommi Lundbergin mukaan vertailutoiminnallisuuden sisältävän projektinhallintajärjestelmän yritykselle tuomia hyötyjä ovat kustannusarvioinnin tarkkuuden paraneminen, tarjouspyyntöjen tekemisen nopeutuminen, informaation saaminen

ulkoistuspäätösten tueksi ja modulaarisen tuotteistamisen helpottaminen (Lundberg 1.11.2010).

LÄHTEET

Achour, M.; Betz, F.; Dovgal, A.; Lopes, N.; Magnusson, H.; Richter, G.; Seguy, D. & Vrana, J. 2010. PHP Manual. What is PHP?. Viitattu 26.9.2010 http://www.php.net/manual/en/intro-whatis.php.

Ambler, S. W. 2009. The Agile Unified Process (AUP). Viitattu 14.11.2010 http://www.ambysoft.com/unifiedprocess/agileUP.html.

Apache Software Foundation 2009. What IS the Apache HTTP Server Project?. Viitattu 7.10.2010 http://httpd.apache.org/ABOUT_APACHE.html.

Beck, K.; Beedle, M.; Bennekum, A.; Cockburn, A.; Cunningham, W.; Fowler, M.; Grenning, J.; Highsmith, J.; Hunt, A.; Jeffries, R.; Kern, J.; Marick, B.; Martin, R.C.; Mellor, S.; Schwaber, K.; Sutherland, J. & Thomas. D. 2001a. Manifesto for Agile Software Development. Viitattu 13.11.2010 http://agilemanifesto.org/.

Beck, K.; Beedle, M.; Bennekum, A.; Cockburn, A.; Cunningham, W.; Fowler, M.; Grenning, J.; Highsmith, J.; Hunt, A.; Jeffries, R.; Kern, J.; Marick, B.; Martin, R.C.; Mellor, S.; Schwaber, K.; Sutherland, J. & Thomas. D. 2001b. Manifesto for Agile Software Development. Viitattu 13.11.2010 http://agilemanifesto.org/principles.html.

Burnette, E. 2005. Eclipse IDE: Pocket Guide. Sebastopol: O'Reilly Media, Inc.

Copeland, L. 2001. Extreme Programming. Viitattu 13.11.2010 http://www.computerworld.com/s/article/66192/Extreme_Programming?taxonomyId=063.

Flanagan, D. 2006. JavaScript: The Definitive Guide. 5. painos. Sebastopol: O'Reilly Media, Inc.

Gunderloy, M.; Jorden, J. L. & Tschanz, D. W. 2006. Mastering Microsoft SQL Server 2005. Indianapolis: Wiley Publishing, Inc.

Highsmith, J. 2001. History: The Agile Manifesto. Viitattu 13.11.2010 http://agilemanifesto.org/history.html.

Jeffries, R. E. 1998. Pair Programming. Viitattu 13.11.2010 http://xprogramming.com/Practices/PracPairs.html

Jeffries, R. E. 2010. What is Extreme Programming?. Viitattu 13.11.2010 http://xprogramming.com/xpmag/whatisxp.

JQuery Project 2010a. jQuery is a new kind of JavaScript library. Viitattu 2.10.2010 http://jquery.com/.

JQuery Project 2010b. jQuery API reference. Manipulation. Viitattu 3.10.2010 http://api.jquery.com/category/manipulation/.

JQuery Project 2010c. jQuery API reference. Ajax. Viitattu 5.10.2010 http://api.jquery.com/jQuery.ajax/.

Kroll, P. & Kruchten, P. 2004. The Rational Unified Process Made Easy. A Practicioner's Guide to the RUP. 4. painos. Boston: Pearson Education, Inc.

Mountain Goat Software 2010. The Scrum Team. Viitattu 13.11.2010 http://www.mountaingoatsoftware.com/scrum/team.

Nourie, D. 2005. Getting Started with an Integrated Development Environment (IDE). Viitattu 6.10.2010 http://java.sun.com/developer/technicalArticles/tools/intro.html.

PhpMyAdmin Developer Team 2010. About phpMyAdmin. Viitattu 14.10.2010 http://www.phpmyadmin.net/home_page/.

Price, J. 2007. Oracle Database 11g SQL. New York: The McGraw-Hill Companies, Inc.

Scrum Alliance 2010a. Scrum Is an Innovative Approach to Getting Work Done. Viitattu 13.11.2010 http://www.scrumalliance.org/learn_about_scrum.

Scrum Alliance 2010b. Scrum Terminology. Viitattu 13.11.2010 http://www.scrumalliance.org/learn_about_scrum.

Seidler, K. 2009. XAMPP. Viitattu 6.10.2010 http://www.apachefriends.org/en/xampp.html.

Thomson, L. & Welling, L. 2009. PHP and MySQL Web Development. 4. painos. Lontoo: Addison-Wesley.

W3Schools 2010. jQuery Selectors. Viitattu 2.10.2010 http://www.w3schools.com/jquery/jquery_ref_selectors.asp.

Wells, D. 1999a. User Stories. Viitattu 13.11.2010 http://www.extremeprogramming.org/rules/userstories.html.

Wells, D. 1999b. Iteration Planning. Viitattu 13.11.2010 http://www.extremeprogramming.org/rules/iterationplanning.html.

Wells, D. 1999c. Iterative Development. Viitattu 13.11.2010 http://www.extremeprogramming.org/rules/iterative.html.

Wells, D. 1999d. The Rules of Extreme Programming. Viitattu 13.11.2010 http://www.extremeprogramming.org/rules.html.

Wells, D. 1999e. Acceptance Tests. Viitattu 13.11.2010 http://www.extremeprogramming.org/rules/functionaltests.html.

Zakas, N.C. 2009. Professional JavaScript for Web Developers. 2. painos. Indianapolis: Wiley Publishing, Inc.