Jukka Leinonen

PROJEKTINSEURANTASOVELLUS

Insinöörityö
Kajaanin ammattikorkeakoulu
Tekniikan ja liikenteen ala
Tietotekniikan koulutusohjelma
Kevät 2007

Kajaanin ammattikorkeakoulu

OPINNÄYTETYÖ
TIIVISTELMÄ

Koulutusala Tekniikan ja liikenteen ala	Koulutusohjelma Tietotekniikka
Tekijä(t) Jukka Leinonen	
Työn nimi Projektinseurantasovellus	
Vaihtoehtoiset ammattiopinnot	Ohjaaja(t) Raili Simanainen
	Toimeksiantaja Creabyte Oy
Aika Kevät 2007	Sivumäärä ja liitteet 30 + 3

Tämä insinöörityö tehtiin Creabyte Oy:n tilauksesta. Insinöörityön tavoitteena oli suunnitella ja toteuttaa Creabyte Oy:n ja sen sidosryhmien käyttöön helppokäyttöinen web-pohjainen sovellus projektien seurantaan. Yrityksen aiemmat kokemukset erilaisista saatavilla olevista projektisovelluksista eivät ole olleet kovin positiivisia, sovellukset ovat helposti liian laajoja tai niistä puuttuu jokin tarvittava ominaisuus.

Creabyte Oy on kesäkuussa 2004 perustettu pieni ohjelmistoalan yritys. Aiemmin avoimena yhtiönä toiminut yritys tekee pitkälle räätälöityjä sisällönhallintajärjestelmiä ja muita Internet-ohjelmointiprojekteja.

Sovelluksen suunnittelussa hyödynnettiin UML-mallinnuskieltä, ja sovelluksen toteutus tehtiin pääosin HTML- ja PHP-ohjelmointikielillä. Työn toisena tavoitteena oli tutkia proseduraalisten ja olio-pohjaisten ohjelmointitapojen eroja ja käyttökelpoisuutta tietokantakyselyissä Creabyte Oy:n kannalta.

Työn tuloksena saatiin toimiva sovellus, joka helpottaa Creabyte Oy:n ja sen sidosryhmien välistä viestintää.

Kieli	Suomi
Asiasanat	
Säilytyspaikka	☐ Kajaanin ammattikorkeakoulun Kaktus-tietokanta ☐ Kajaanin ammattikorkeakoulun kirjasto

Kajaanin ammattikorkeakoulu
University of Applied Sciences

THESIS
ABSTRACT

School	Degree Programme
School Of Engineering	Information Technology
Author(s) Jukka Leinonen	
Title A Project Following Application	
Optional Professional Studies	Instructor(s) Raili Simanainen
	Commissioned by Creabyte Oy
Date Spring 2007	Total Number of Pages and Appendices 30 plus 3 appendices

This Bachelor's thesis was made for Creabyte Oy. The purpose of the thesis was to design and implement a user friendly, web-based application for project following to Creabyte Oy and its interest groups. The company's experiences from the available project applications were not very positive, applications are too large or they do not have the required features.

Creabyte Oy was established in June 2004. Creabyte Oy produces tailor-made web site content management applications and other web-based programming projects.

The application was mainly programmed with HTML and PHP, and it uses MySQL as a database. The other purpose of this thesis was to examine the differences of procedural and object-oriented programming styles in the database queries.

The final outcome of the thesis was a complete application, which makes the communication easier between Creabyte Oy and its customers.

Language of Thesis	Finnish
Keywords	
Deposited at	☐ Kaktus Database at Kajaani University of Applied Sciences ☐ Library of Kajaani University of Applied Sciences

ALKUSANAT

Tämä Kajaanin ammattikorkeakoulun tietotekniikan insinöörityö on tehty Creabyte Oy:n tilauksesta.

Haluan kiittää Creabyte Oy:n toimitusjohtajaa Timo Lukkaria ja insinöörityön valvojaa lehtori Raili Simanaista Kajaanin ammattikorkeakoulusta. Työn kieliasun tarkastamisesta haluan kiittää lehtori Eero Soinista ja yliopettaja Kaisu Korhosta.

Jukka Leinonen

SISÄLLYS

LIITTEET

LYHENNELUETTELO

Ajax	Asynchronous JavaScript And Xml. Yleisnimitys JavaScript, XML, XHTML, CSS, XMLHTTPRequest ja DOM -tekniikoiden yhteiskäytölle.
ActiveX	Ohjelmistokomponentti, jolla Web-sivuihin, sovelluksiin ja ohjelmankehitys-työkaluihin voidaan lisätä erityistoimintoja.
CSS	Cascading Style Sheets. Tyylikieli, jolla HTML-elementteihin voidaan liittää erilaisia tyylimäärityksiä.
DOM	Document Object Model. Tekniikka, joka mahdollistaa (X)HTML –dokumenttien sisällön muokkauksen.
FTP	File Transfer Protocol. Tiedostojen siirtämismenetelmä.
HTML	HyperText Markup Language. Internetin sivunkuvauskieli.
JavaScript	Selaimessa suoritettava skriptauskieli.
MySQL	WWW-palvelimilla yleisimmin saatavilla oleva tietokantaohjelmisto.
PHP	Hypertext Preprocessor. Web-palvelimella suoritettava skriptauskieli.
SQL	Structured Query Language. Tietokannan kyselykieli.
XHTML	Extensible HyperText Markup Language. HTML:stä kehitetty sivunkuvauskieli, joka täyttää XML:n muotovaatimukset.
XML	Extended Markup Language. Metakieli, jolla määritellään rakenteellisia merkkauskieliä.
XMLHTTPRequest	JavaScript-objekti. Mahdollistaa palvelimen kutsun itsenäisesti, mikä mahdollistaa selaimen näkymän päivittämisen ilman, että ladataan uudestaan koko sivua.

1 JOHDANTO

Tämän insinöörityön tavoitteena oli suunnitella ja toteuttaa web-pohjainen projektinseurantasovellus Creabyte Oy:n sekä sen sidosryhmien käyttöön. Sovelluksen tarkoituksena on helpottaa yrityksen ja sen asiakkaiden, alihankkijoiden ja muiden projekteissa mukana olevien tahojen välistä viestintää.

Creabyte on kajaanilainen, vuonna 2000 perustettu yritys, ja alkuun avoimena yhtiönä toimineen yrityksen toiminta perustui tietokoneiden ja oheislaitteiden myyntiin. Yhtiömuoto vaihtui osakeyhtiöksi vuonna 2004, ja tietokonekauppa vaihtui erilaisten sisällönhallintajärjestelmien ja muiden Internet-ratkaisuiden tuottamiseen ja myymiseen. Yritys työllistää tällä hetkellä kaksi henkilöä ja suurin osa tämän hetkisestä asiakkaista ovat kainuulaisia yrityksiä tai yhteisöjä.

Monet saatavilla olevat projektinhallinta- tai seurantasovellukset ovat monimutkaisia ja vaikeaselkoisia tai liian laajoja. Yrityksen asiakkaita ajatellen järjestelmän tuli olla mahdollisimman yksinkertainen ja helppokäyttöinen. Lisäksi itse suunnittelemalla ja tekemällä saa omiin tarkoituksiin juuri sopivan järjestelmän.

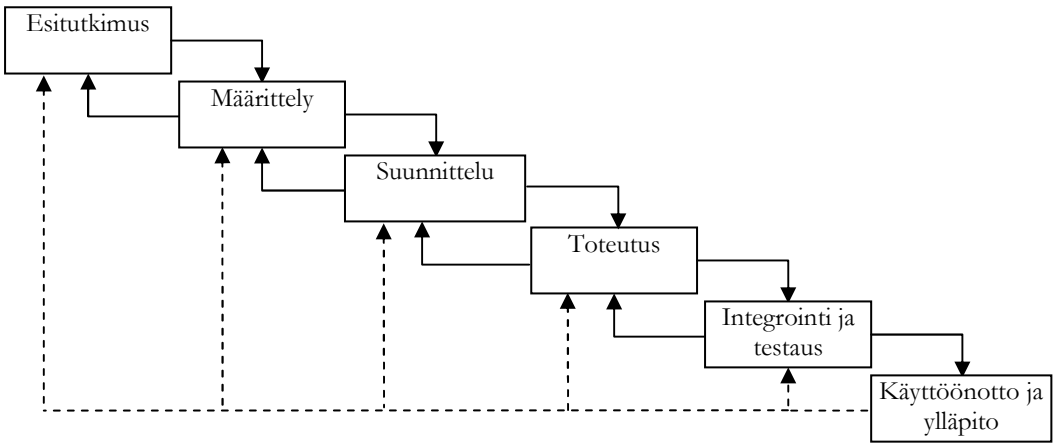
Työtä suunniteltaessa oli jo selvää, että sovellus tullaan tekemään MySQL-tietokantapohjaiseksi PHP- ja Ajax-tekniikoita yhdistäen, jotka ovat yritykselle tuttuja muistakin projekteista. Näin sovelluksen päivittäminen ja laajentaminen on jatkossakin vaivatonta. Sovellus sijoitettiin Creabyte Oy:n sivutilaan Woima-Wirta Oy:n palvelimelle.

2 OHJELMISTOTUOTANTO

Ohjelmistotuotannolla tarkoitetaan ohjelmistojen suunnittelua ja kehitystä järjestelmällisesti. Ohjelmistojen kehittäminen tapahtuu yleisimmin projektityöskentelynä ja projekti joudutaan usein jakamaan osaprojekteihin. Yleensä tuotekehitysprosessista voidaan erottaa määrittely-, suunnittelu-, ohjelmointi- ja testausvaiheet ja näitä vaiheita seuraavat yleensä ohjelmiston käyttöönotto ja ylläpito. Ohjelmistoprojektiin liittyy koko projektin ajan kestäviä tukitoimintoja, joista tärkeimmät ovat dokumentointi, laadunvarmistus ja tuotteenhallinta. [1.], [2.]

2.1 Ohjelmiston elinkaari ja elinkaaren vaiheet

Ohjelmiston kehittämisen aloittamisen ja sen käytöstä poistamisen välistä aikaa kutsutaan ohjelmiston elinkaareksi. Vaihejakomalli tarkoittaa tapaa, jolla ohjelmiston elinkaari tai kehitystyö jaetaan vaiheisiin. Projektin etenemistä seurataan katselmuksien avulla esimerkiksi kunkin vaiheen päätteeksi. Tavallisin vaihejakomalli on vesiputousmalli, joka on esitetty kuvassa 1. Kun vesiputousmalliin yhdistetään joka vaihetta vastaava testaus, saadaan niin kutsuttu testauksen V-malli, jonka vasen sakara kuvastaa ohjelmistoprosessin vaihetta ja oikea kunkin prosessin vaiheen testausta. Muita yleisesti käytettyjä elinkaarimalleja ovat muun muassa evoluutio-, prototyyppi- ja spiraalimallit. Evoluutiomalli muodostuu sarjasta toistuvia vesiputouksia, joista jokaisen tuloksena on uusi versio järjestelmästä, prototyyppimallissa taas luodaan ensin toteutettavan järjestelmän prototyyppi jonka jälkeen suunnitellaan jatkotoimenpiteet. Spiraalimalli yhdistää vesiputousmallin ja prototyyppimallin lisäksi riskianalyysin. Ohjelmistoille, joiden vaatimukset ovat epäselviä tai herkkiä muuttumaan, voidaan käyttää myös niin sanottua ketterää *agile* tai kevyttä *light* prosessimallia. Ominaista ketterille prosessimalleille on kevyt dokumentointi ja projektinhallinta, pienehköt tiimit ja pienet julkaisut tiheään tahtiin. Tunnetuin kevyt prosessimalli on Extreme Programming eli XP. [1.], [2.]



Kuva 1. Vesiputousmalli [1, s.25].

Esitutkimusvaiheessa asetetaan yleiset järjestelmätason vaatimukset. Nämä vaatimukset määrittelevät asiakkaan tarpeet, eli ne vastaavat kysymykseen miksi ohjelmisto tulisi tehdä. Esitutkimus voidaan usein ajatella myös osaksi määrittelyvaihetta, koska yleensä asiakastarpeiden analysointi ja tarkentaminen jatkuu koko määrittelyvaiheen ajan. [1.]

Määrittelyvaiheessa asiakkaan vaatimuksia analysoidaan ja ne johdetaan toteutettavan järjestelmän ohjelmistovaatimuksiksi. Määrittelyvaiheen tuloksena syntyvä määrittelydokumentti kuvaa ohjelmiston toiminnot, ei-toiminnalliset vaatimukset ja rajoitukset. Määrittelyvaiheessa siis kuvataan mitä järjestelmä tekee. [1.]

Suunnitteluvaihe vastaa kysymykseen miten järjestelmä tehtävänsä suorittaa, eli suunnitellaan määrittelyn kuvaamien toimintojen toteutus. Tämän vaiheen tuloksena syntyvä ohjelmistosuunnitelma kuvaa ohjelmiston tai ohjelman teknisen arkkitehtuurin tarkasti. Suunnittelu jaetaan usein kahteen tasoon. Arkkitehtuurisuunnittelussa järjestelmä jaetaan moduuleihin ja määritellään moduulien rajapinnat. Moduulisuunnittelussa vastaavasti suunnitellaan kunkin moduulin sisäinen rakenne. Yleisiä suunnitteluun liittyviä tavoitteita ovat selkeys ja ymmärrettävyys, tehokkuus, luotettavuus, ylläpidettävyys ja siirrettävyys. [1.]

Toteutusvaihe käsittää varsinaisen ohjelmoinnin valitulla ohjelmointikielellä. Mikäli edelliset vaiheet toteutettiin kunnolla, on toteutukseen menevä aika noin yksi tai kaksi kymmenesosaa koko ohjelmistoprojektiin käytetystä ajasta. [1.]

Testauksen tarkoituksena on löytää ohjelmistossa olevat virheet. Testausvaiheessa testaus suunnitellaan, luodaan testiympäristö, suoritetaan testit ja tarkastellaan testin tuloksia joiden mukaan ohjelmistoon tehdään tarvittavat muutokset. [1.]

Viimeisessä vaiheessa ohjelma otetaan käyttöön ja sitä varten ohjelmistosta valmistellaan kokonaisuus, joka suunnitellusti asennetaan määrätyille laitteille. Käyttöönottoon liittyy myös käyttäjien valmentaminen. Ylläpito sisältää ne toimenpiteet, mitä asiakkaat tarvitsevat ollakseen tyytyväisiä tuotteeseen. Se sisältää virheiden korjaamista, asiakkaan ongelmien ratkomista, ohjelman muuttamista ja uusien piirteiden lisäämistä. [1.], [2.]

2.2 Dokumentointi, laadunvarmistus ja tuotteenhallinta

Ohjelmistotuotannolle on ominaista projektin aikana kertyneen tiedon kirjaaminen dokumenttien muotoon. Projektiin liittyviä dokumentteja voi olla kymmeniä. Minimidokumentaatioon kuuluvat projektisuunnitelma, määrittelydokumentti, suunnitteludokumentti ja mielellään myös testaussuunnitelma. Dokumenttien määrä ja osin sisältökin riippuu projektin koosta ja monimutkaisuudesta. Projektidokumentaatiossa on olennaista, että ohjelmistoa korjattaessa tai muutettaessa on dokumentaatiokin päivitettävä, muuten dokumentit muuttuvat vähitellen hyödyttömiksi. [1.]

Laadunvarmistusasiat tarkoittavat kaikkia niitä keinoja, joilla pyritään varmistumaan tuotteen ja tuotetta tekevän prosessin laadusta. Ohjelmiston laadulla tarkoitetaan yleensä ohjelmistotuotteen kykyä täyttää käyttäjänsä kohtuulliset toiveet ja odotukset. Laatua voidaan tarkastella sekä toiminnanlaadun että tuotteen laadun kannalta. [1.]

Tuotteenhallinta on tavallisesti suunniteltava jokaisen tuotteen kohdalla erikseen. Yksinkertaisimmillaan tuotteenhallinta on tilanteessa, jossa tuotteesta on olemassa vain yksi konfiguraatio. Tällöin tuotteenhallinta on pääasiassa versionhallintaa, jonka tarkoituksena on helpottaa työskentelyn koordinointia tuotekehityksen aikana.[1.]

3 TIETOKANTAPOHJAISET VERKKOSOVELLUKSET

Verkkosovellukset ovat www-selaimella käytettäviä sovelluksia. Näitä ovat muun muassa erilaiset kauppa- ja pankkisovellukset, keskustelu- ja uutispalstat. Sovelluksen parametrit välitetään lomakkeilta, sivulta toiselle johtavissa linkeissä tai sessiotietojen avulla ja niiden tulosteet näkyvät selattavina sivuina. [3, s. 11]

WWW-sivut jaetaan staattisiin ja dynaamisiin sivuihin. Staattinen sivu on tiedosto palvelinkoneella ja sen päivitys tapahtuu tavallisesti muokkaamalla sivu jollain HTML-editorilla ja siirtämällä ftp-tiedonsiirto-ohjelman avulla päivitetty sivu palvelimelle. Dynaaminen sivu taas luodaan vasta selaimen sitä pyytäessä. Selaimen hakupyyntö käynnistää palvelimella toimintoja, joiden perusteella näytettävä verkkosivu muodostetaan. Lisäämällä tietokantasovellus dynaamiseen sivuun saadaan käyttäjälle mahdollisuus muuttaa sivun sisältöä suoraan selaimen kautta. [4.]

Verkkosovellukset tai verkkopalvelut voidaan jaotella kohderyhmien perusteella Internet-, ekstranet- ja intranet-palveluihin, joista Internet-palvelun käyttö on mahdollista kenelle tahansa. Yrityksien sisäisistä verkkopalveluista käytetään nimeä intranet ja rajatulle kohderyhmälle tarkoitetusta verkkopalvelusta ekstranet. Toinen tapa jaotella verkkopalveluita on jakaa ne karkeasti kahteen eri luokkaan, viestinnällisiin ja operatiivisiin verkkopalveluihin. Kun verkkoa hyödyntämällä tapahtuu jokin muutos Internet-järjestelmän ulkopuolella, puhutaan operatiivisista palveluista. Tällaisia muutoksia ovat esimerkiksi sähköisen varausjärjestelmän varaama istuin lentokoneesta tai pankissa rahan siirtyminen tililtä toiselle. Viestinnällisen verkkopalvelun hyöty käyttäjälle perustuu palvelussa oleviin teksteihin, kuviin, ääneen tai videoon. [5.]

Verkkosovellusten ohjelmoinnissa käytetyimpiä tekniikoita ovat ASP, ASP.NET, CGI, JSP ja PHP. Näistä CGI (Common Gateway Interface)–ohjelmissa on kysymys lähinnä keskustelutavasta, joten CGI-ohjelmia voidaan kirjoittaa melkein millä tahansa ohjelmointikielellä. PHP ja ASP tekniikoissa ohjelmointikielen komennot voidaan kirjoittaa suoraan HTML-sivujen sisään. [3, s. 11]

3.1 Relaatiotietokannat

Relaatiotietokanta on yleisimmin käytössä oleva tietokantamalli. Relaatiotietokanta koostuu useista tauluista, joita kutsutaan myös relaatioiksi. Yhtä riviä kutsutaan tietueeksi, ja taulun jokaisella rivillä on yhtä monta tietoa eli kenttää. Jokaisella rivillä täytyy olla yksikäsitteinen perusavain, joka vastaa jotakin reaalimaailman kohdetta. Kuhunkin kohteeseen liitetään vain siihen välittömästi liittyvät ominaisuudet. Relaatiotietokannasta tietoa voidaan hakea lukemattomin eri tavoin, mutta tiedon haku tapahtuu vain tiedon nimien ja arvojen perusteella, ei siis koskaan tiedon sijainnin tai järjestyksen mukaan. [6.]

Tietokannan hallintajärjestelmä pitää taulun järjestyksessä perusavaimen mukaan. Järjestetyn taulun käsitteleminen on huomattavasti tehokkaampaa kuin järjestämättömän. Taulun joitakin kenttiä voidaan määrittää toissijaisiksi avaimiksi. Toissijaiset avaimet nopeuttavat taulusta haettavan tiedon järjestämistä, joten toissijaisiksi avaimiksi määritellään sellaisia kenttiä, joiden mukaan tietoja halutaan mahdollisesti järjestellä. Taulun kentistä voidaan viitata jonkin toisen taulun perusavaimeen tai toissijaiseen avaimeen. Tällaista yhteyttä kutsutaan viite-eheydeksi ja kyseinen kenttä määritellään yleensä myös avaimeksi, jota kutsutaan viiteavaimeksi. Viite-eheys määrää, että jokaista viittaavassa taulussa esiintyvää viiteavaimen arvoa pitää vastata sama perusavaimen arvo viittauksen kohteena olevassa taulussa. Kuvassa 2 on esimerkki tietokannan henkilö- ja osastotauluista ja niiden välisestä viite-eheydestä. [6.]

Perusavain — Viiteavain

HenkiloID	Etunimi	Sukunimi	Osasto
4	Matti	Meikäläinen	2
5	Maija	Meikäläinen	1

Perusavain

OsastoID	Osasto
1	Markkinointi
2	Hallinto

Kuva 2. Henkilö- ja osastotaulut.

3.2 Työkalut

UML

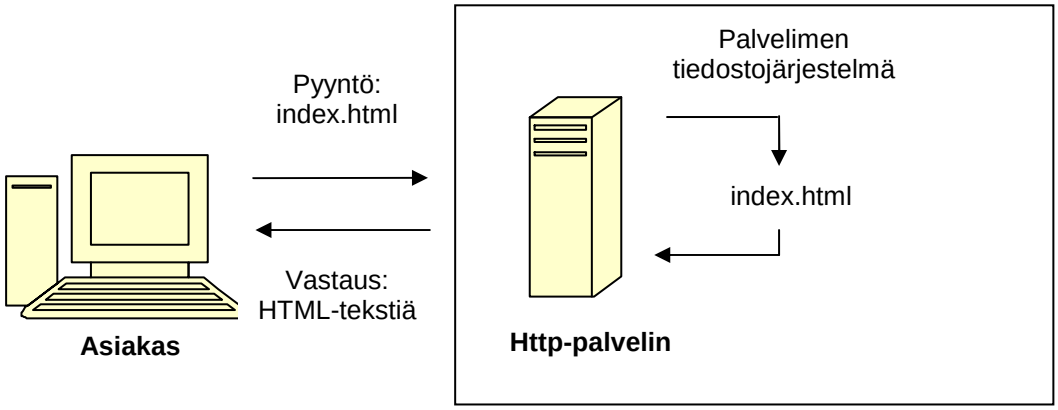
UML on standardoitu kuvauskieli oliorakenteiden ja järjestelmien esittämiseen. Uusin UML-standardi 2.0 sisältää 13 erilaista kaaviota. UML:ää käytetään ohjelmistosuunnittelun lisäksi muun muassa järjestelmien arkkitehtuurien kuvaukseen, ajettavan koodin luomiseen, työvirtojen kuvaukseen ja liiketoimintaprosessien mallintamiseen. [7.]

Kaaviotyypit jaetaan rakennekaavioihin ja käyttäytymiskaavioihin. Rakennekaaviot (luokka-, kooste-, komponentti-, sijoittelu-, olio- ja pakkauskaavio) kuvaavat järjestelmien staattista, ajasta riippumatonta rakennetta eri abstraktiotasoilla. Käyttäytymiskaaviot (aktiviteetti-, käyttötapaus- ja tilakaavio sekä vuorovaikutuskaaviot) kuvaavat eri näkökulmista järjestelmän ajonaikaista, ajasta riippuvaa toimintaa. Vuorovaikutuskaavioita ovat sekvenssi-, yhteistoiminta-, kokoava vuorovaikutus- ja ajoituskaavio. [8.]

PHP

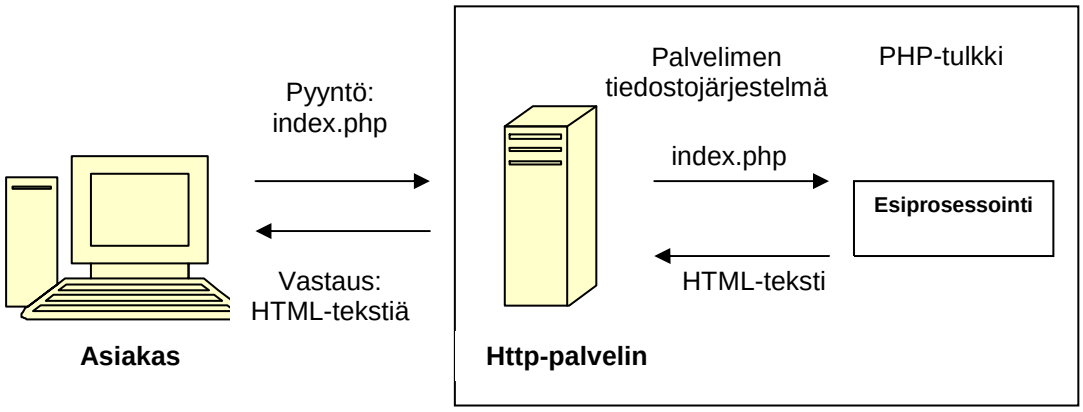
PHP:n kehitystyö alkoi vuonna 1994 ja sen kehitys on ollut nopeaa. Alun perin CGI-ohjelmien tekemistä helpottavasta komentokokoelmasta on tullut täysiverinen ohjelmointikieli ja sovelluspalvelinympäristö. PHP on niin sanottu *cross platform* -ohjelmisto eli se toimii laajasti sekä Unix- että Windows -pohjaisissa käyttöjärjestelmissä. [3.]

PHP:n syntaksi muistuttaa paljon C-kieltä ja Javaa ja se sisältää kaikki ohjelmointikielille tyypilliset rakenteet ja oliopohjaisuuden. PHP-koodi kirjoitetaan HTML-koodin sekaan ja se suoritetaan palvelimella, jolloin itse ohjelmakoodi ei näy selaimessa päinvastoin kuin esimerkiksi JavaScript, joka suoritetaan selaimella. Tavalliset HTML-sivut toimivat siten, kuva 3, että asiakkaan pyytäessä palvelimelta HTML-sivua, palvelin palauttaa asiakkaalle pyydetyn sivun. [3.]



Kuva 3. Http-palvelimen toiminta yksinkertaisimmillaan [3, s.14].

PHP-sivua kutsuttaessa, kuva 4, palvelin tunnistaa tiedoston päätteen perusteella että kyseessä on PHP-sivu, jolloin palvelin välittää PHP-tulkille pyynnön esiprosessoida pyydetty sivu. PHP-tulkki kääntää ja suorittaa tiedoston sisältämän PHP-koodin ja palauttaa palvelimelle pelkkää HTML-koodia, jonka palvelin taas palauttaa asiakkaalle. [9.]



Kuva 4. Http-palvelimen toiminta PHP-sivua kutsuttaessa. [9.]

MySQL

MySQL on ilmainen tietokantaohjelma, joka noudattaa asiakas-palvelin arkkitehtuuria jossa tietokannan käsittely tapahtuu aina palvelinohjelman kautta. MySQL-palvelimeen voidaan ottaa yhteyttä PHP:n lisäksi ainakin ODBC tai Java/JDBC, Perl-, Python- ja TCL-tekniikoin tai sen oman C-ohjelmointirajapinnan kautta. Myös oliopohjainen C++ -ohjelmointirajapinta on käytettävissä.

MySQL-tietokannan kyselykielenä on SQL, joka on standardoitu tietokantojen kyselykieli. MySQL ei noudata standardia kovinkaan perusteellisesti, mutta peruskomennot toimivat standardin mukaisesti. [3, s.33]

Ajax

Ajax-tekniikka yhdistää joukon olemassa olevia tekniikoita siten, että internetsivujen sisältöä voidaan muuttaa ilman koko sivun lataamista ja siten sivuille saadaan lisättyä vuorovaikutteisuutta, nopeutta ja käytettävyyttä. Ajax-termiä käytti ensimmäisen kerran Jesse James Garrett helmikuussa 2005, vaikkakin tällaista asynkronista sisällön lataamista voitiin toteuttaa jo aiemmin Microsoftin Internet Explorer 3 (1996) selaimen iframe-elementtiin ja Netscape 4 (1997) selaimen layer-elementtiin. [10.]

Ajax toimii useimmilla nykypäivän XML-tuetuilla selainohjelmilla ja sen käyttö onkin yleistynyt nopealla vauhdilla. Tunnettuja palveluita, joissa Ajax-teknologia on käytössä ovat muunmuassa Google Maps, Gmail, Flickr, Fonectan 020202 ja www.fi –hakukone. [11.]

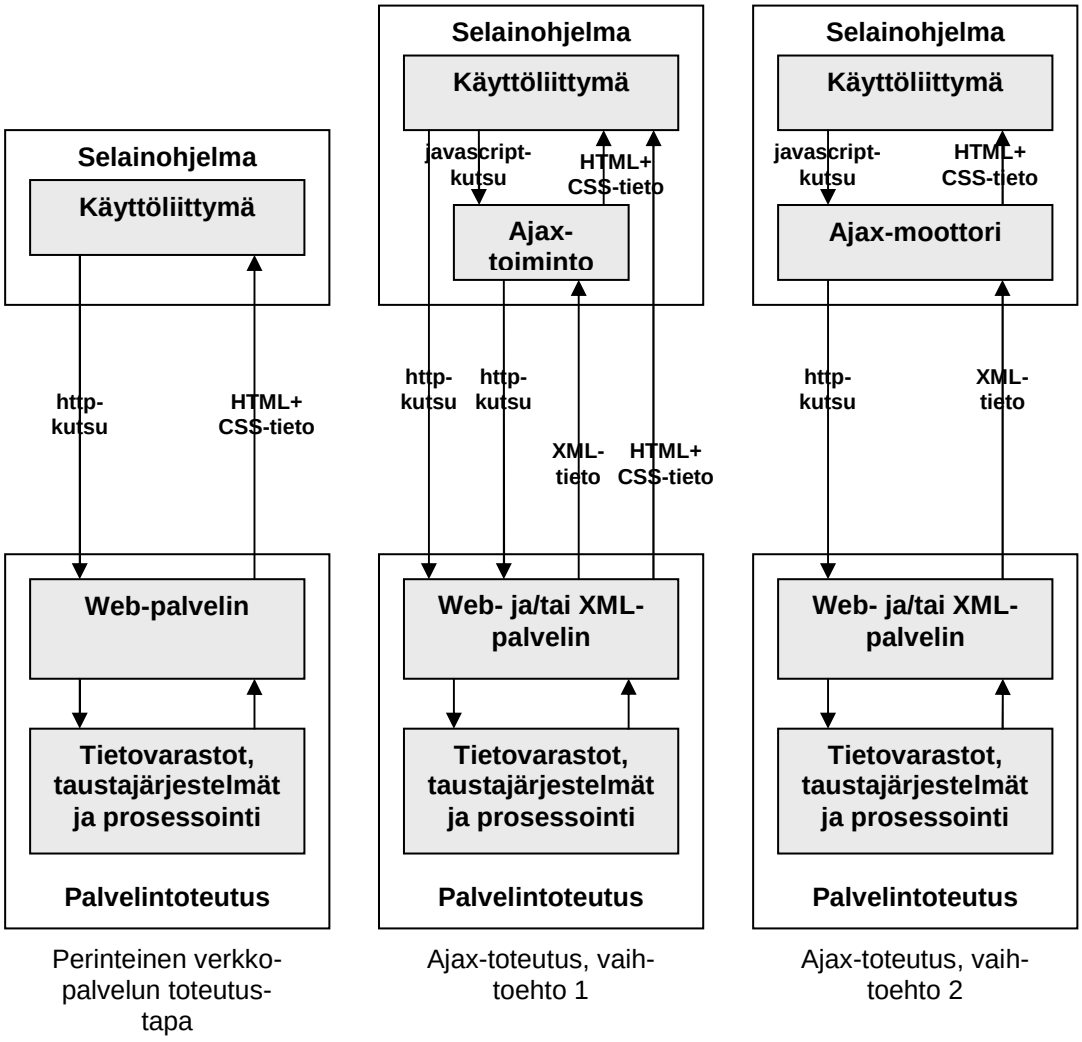
Teknisesti Ajax siis vain yhdistää eri tekniikoita. Nämä tekniikat ovat

- standardipohjaiset näyttötekniikat XHTML ja CSS,
- dynaamisen sisällönmuokkauksen mahdollistava Document Object Model,
- tiedon vaihtoon ja käsittelyyn XML ja XSLT,
- asynkroniseen tiedon noutamiseen XMLHTTPRequest,
- JavaScript edellä mainittujen yhteen sitomiseen.

Perinteinen verkkosovellus toimii siten, että sivun sisällön päivittämiseksi tehdään HTTP-kutsu joka käyttäjän tekemien valintojen tai syöttämien tietojen perusteella päivittää koko sivun sisällön. Ajax toteutuksessa tehdään JavaScript-kutsu erityiselle Ajax-kerrokselle tai Ajax-toiminnolle, joka tekee tiedon haun palvelimelta ja päivittää sivulta vain tietyn osan. [10.],[11.]

Kuvassa 5 on esitetty perinteisen ja Ajax -toteutuksien erot. Kuvan toteutusvaihtoehto 1 on hyvä vaihtoehto jos koko sovellusta ei ole tarkoitus tehdä Ajax-pohjaiseksi tai jos jollekin jo

olemassa olevaan sovellukseen halutaan lisätä Ajax-toimintoja. Toteutusvaihtoehto 2 taas on täysin Ajax-pohjainen sovellus ja on jo haastavampi toteuttaa. [10.],[11.]



Kuva 5. Perinteisen verkkosovelluksen ja Ajax-toteutuksien erot. [4.]

Selaimien Ajax-tukeen liittyvät ongelmat ovat ainoa tekijä, jota voi pitää ainoana merkittävänä syynä olla käyttämättä Ajax-toteutuksia. Jos suuri osa sivuston käyttäjistä käyttää vanhoja selaimia, Ajax-tekniikat eivät toimi kunnolla. Lisäksi Ajax-toteutusta suunniteltaessa on syytä huomioida myös käytettävyys, esimerkiksi sivun sisällön dynaaminen päivittäminen saattaa jäädä koko sivun päivittymiseen tottuneilta täysin huomaamatta. Ajax-tekniikalla toteutettu sivu ei myöskään muokkaa sivun osoitekentästä löytyvää osoitetta, joten halutun sivun lisääminen kirjanmerkkeihin tai linkin lähettäminen ystävälle ei ole mahdollista.[12.]

4 PROSE-SOVELLUKSEN VAATIMUSMÄÄRITTELY

4.1 Yleiskuvaus

Prose -projektinseurantasovellus on www-pohjainen sovellus Creabyte Oy:n ja sen sidosryhmien käyttöön. Sovelluksen päätehtävänä on helpottaa Creabyte Oy:n ja sen sidosryhmien välistä viestintää.

Tämä määrittely on tarkoitettu kattamaan sovelluksen toteutus. Määrittelydokumentti määrittelee järjestelmän toiminnalliset, ei-toiminnalliset ja muut vaatimukset. Lisäksi dokumentti määrittelee järjestelmän ongelmat ja rajoitukset, tiedot ja tietokannat sekä rajoitukset suunnittelulle ja toteutukselle.

4.2 Ongelmat ja rajoitukset

Aikaisemmin yrityksen viestintä hoitui lähinnä sähköpostin välityksellä ja lisäksi kiireellisemmät muutos- tai korjauspyynnöt hoituivat puhelimen välityksellä. Ongelmaksi muodostui turhat sähköpostit, sillä yhden pienenkin muutoksen tai korjauksen tekeminen saattoi vaatia useita sähköpostiviestejä ja usein nämä sähköpostit kiersivät vielä usean eri henkilön kautta.

Kokemukset erilaisista projektinhallinta-sovelluksista ovat olleet vähemmän toimivia. Sovellukset ovat olleet liian monimutkaisia, liian laajoja tai yksinkertaisesti liian vaikeakäyttöisiä, jotta ne olisivat vakiinnuttaneet paikkansa käytännössä.

Järjestelmä voidaan asentaa mille tahansa web-palvelimelle jossa on PHP versio 5 ja MySQL tietokannasta vähintään versio 4.1.3. Tämä sovellus tehtiin Woima-Wirta Oy:n webhotellipalveluun, jossa on käytössä PHP:n versio 5.1.2 ja MySQL:n versio 5.0.32.

4.3 Toiminnalliset vaatimukset

Järjestelmällä pystytään seuraamaan meneillään olevia projekteja. Projektit ovat pääasiassa erilaisia www-sivustoja, intranet- ja ekstranet-palveluja tai muita www-sovelluksia.

Järjestelmän kannalta käyttäjiä on kahta tyyppiä, pääkäyttäjät ja tavalliset käyttäjät.

Tavallinen käyttäjä voi

- seurata projektien etenemistä
- päivittää projektin tietoja,
- lisätä ja poistaa tiedostoja,
- lisätä ja poistaa kalenterimerkintöjä,
- lisätä ja poistaa virheilmoituksia ja seurata niiden tilaa.

Pääkäyttäjä voi yllä olevien toimintojen lisäksi

- lisätä ja poistaa projekteja,
- lisätä ja poistaa käyttäjiä ja käyttäjäryhmiä,
- liittää ja poistaa projektiin.

Käyttötapauskaavio

Käyttötapaukset kuvaavat järjestelmän ja sen käyttäjien välistä vuorovaikutusta. Kuvassa 6 on projektinseurantasovelluksen käyttötapauskaavio, josta käyttötapausten kuvauksissa (Liite 1) esiintyy jäsenneltynä jokainen varsinainen toiminto sekä poikkeustilanteet osana järjestelmän kokonaistoimintaa.

Kuva 6. Projektinseurantasovelluksen käyttötapauskaavio.

4.4 Ei-toiminnalliset vaatimukset

Verkkosovellusten suunnittelussa valitut palvelinratkaisut ohjaavat myös sovelluksen teknistä toteutusta ja siten myös käytettäviä ohjelmointikieliä. Tämän sovelluksen varsinainen ohjelmointikieli on PHP ja tietokantakyselyiden kieli on SQL. Käyttöliittymän ohjelmointikieliä ovat HTML, XML, JavaScript ja Ajax.

Tämän sovelluksen käyttöliittymän tuli olla sen tulevia käyttäjiä ajatellen selkeä ja helppokäyttöinen. Sovelluksen tuli toimia yleisimpien web-selaimien uudemmilla versioilla joista PC:llä ainakin Internet Explorer, Mozilla Firefox ja Netscape Navigator. Mac:lla vastaavasti Safari ja Firefox.

Sovelluksen tiedonsiirrossa käytetään ainakin alkuun HTTP-siirtoprotokollaa. Tietoturvan lisäämiseksi jatkossa sovelluksen tiedon siirtoon voidaan käyttää myös HTTPS–protokollaa jossa tiedot salataan ennen lähettämistä SSL-protokollan tai TLS-protokollan avulla. Käyttä-

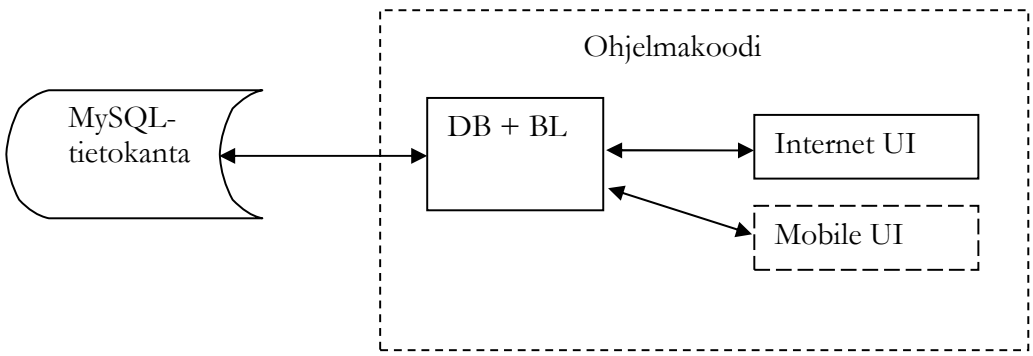
jien autentikointi perustuu käyttäjätunnuksiin ja salasanoihin ja sovellukseen kirjautuminen tapahtuu PHP:n HTTP-autentikaatiolla. HTTP-autentikaation huonona puolena voidaan pitää sitä, että sovelluksesta ulos kirjautuminen onnistuu ainoastaan sulkemalla selain.

Sovelluksen on toimittava luotettavasti normaalitilanteissa. Sovelluksen sisältämien tietojen varmuuskopiointi voidaan suorittaa vedostamalla tietokannasta tarvittaessa. Uusien käyttäjien lisääminen on järjestelmän pääkäyttäjien vastuulla ja ohjelmakoodin päivittäminen ja muutosten tekeminen jatkossa työn tilaajan oman harkinnan mukaan. Lisäksi ohjelmakoodin ylläpidettävyyteen ja modulaarisuuteen on kiinnitettävä huomiota, jotta sovelluksen kehittäminen ja laajentaminen on jatkossakin mahdollista.

4.5 Muut vaatimukset

Sovelluksen suorituskyky riippuu suuresti palvelimen ominaisuuksista, mutta sovelluksen tulee toimia riittävän nopeasti modeemiyhteydelläkin. Käyttäjien syöttämän sisällön suuret tiedostot voivat kuitenkin kasvattaa latautumisaikoja merkittävästi, joten niiden näyttäminen on suunniteltava siten, että käyttäjä voi itse valita lataako isot tiedostot vai ei.

Sovelluksen arkkitehtuuri koostuu DB- (Database), BL- (Business Logic) ja erilaisista UI- (User Interface) kerroksista. Tässä toteutuksessa DB ja BL kerrokset on yhdistetty. Mobile UI on ehkä myöhemmin toteutettava lisätoiminto. MySQL-tietokanta toimii tiedon varastona.



Kuva 7. Sovelluksen arkkitehtuurikuvaus.

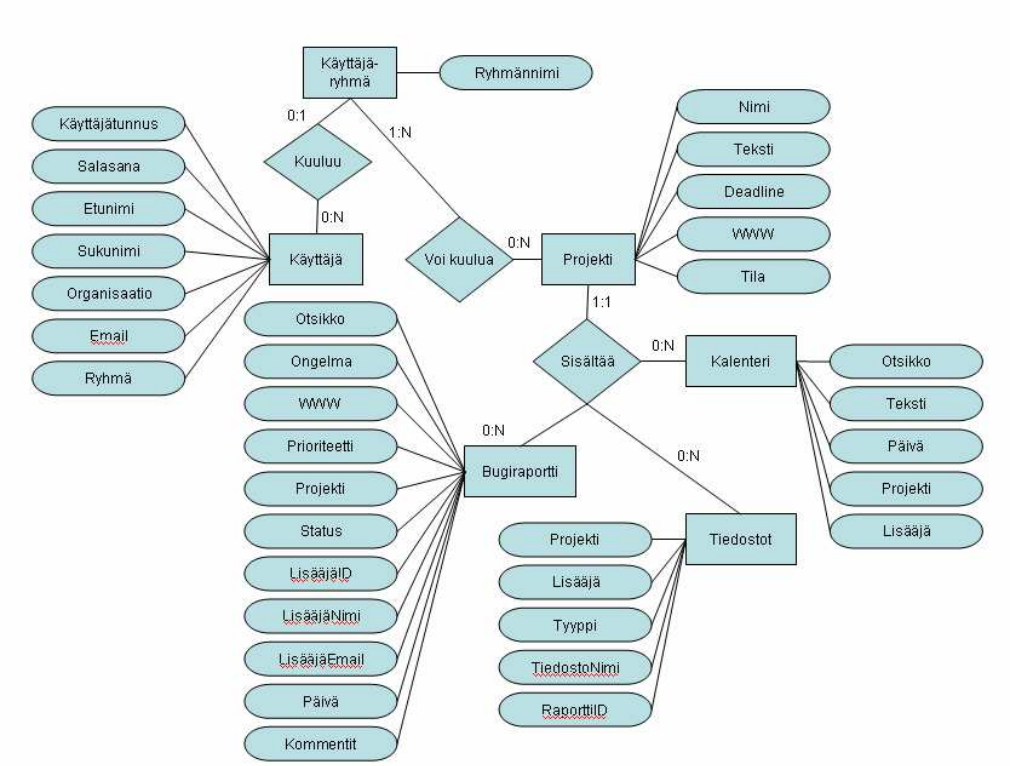
Käytettävät rajapinnat määräytyvät saatavilla olevien palvelinohjelmien mukaan. Sovellus käyttää PHP:n 5-version tarjoamia rajapintoja tietokantajärjestelmien käyttöön. PHP 5:n mysqli –rajapinta mahdollistaa olio-pohjaisen ja perinteisen proseduraalisen ohjelmointitavan tietokantakyselyihin.

Ohjelman käyttöliittymä pidetään alussa mahdollisimman yksinkertaisena, siten sovelluksen ohjelmointi- ja testausvaiheessa pystytään keskittymään sovelluksen toiminnan kannalta olennaisiin seikkoihin.

Kuva 8. Periaatekuva ohjelman käyttöliittymästä.

4.6 Tiedot ja tietokannat

Tietokantana toimii MySQL-tietokanta, jonka hallintaan käytetään phpMyAdmin–MySQL tietokantojen hallintatyökalua. Tietokanta sisältää kuusi taulua joiden rakenne, lukumääräsuhteet ja taulujen väliset yhteydet näkyvät kuvassa 9. Lukumääräsuhteet eli kardinaliteetit ovat muotoa ”n:m” joista selviää kuinka monta oliota voi vähintään (n) ja enintään (m) liittyä tiettyyn olioon yhteyden toisessa päässä. Esimerkiksi yhteyden *Kuuluu* yläpuolella oleva merkintä 0:1 luetaan niin, että voi olla käyttäjiä jotka eivät kuulu yhteenkään käyttäjäryhmään ja toisaalta käyttäjä voi kuulua vain yhteen käyttäjäryhmään. Saman yhteyden alapuolella oleva merkintä 0:N taas kertoo että käyttäjäryhmässä ei tarvitse olla ainuttakaan käyttäjää ja toisaalta taas käyttäjäryhmässä voi olla useita käyttäjiä.[1, s.104]



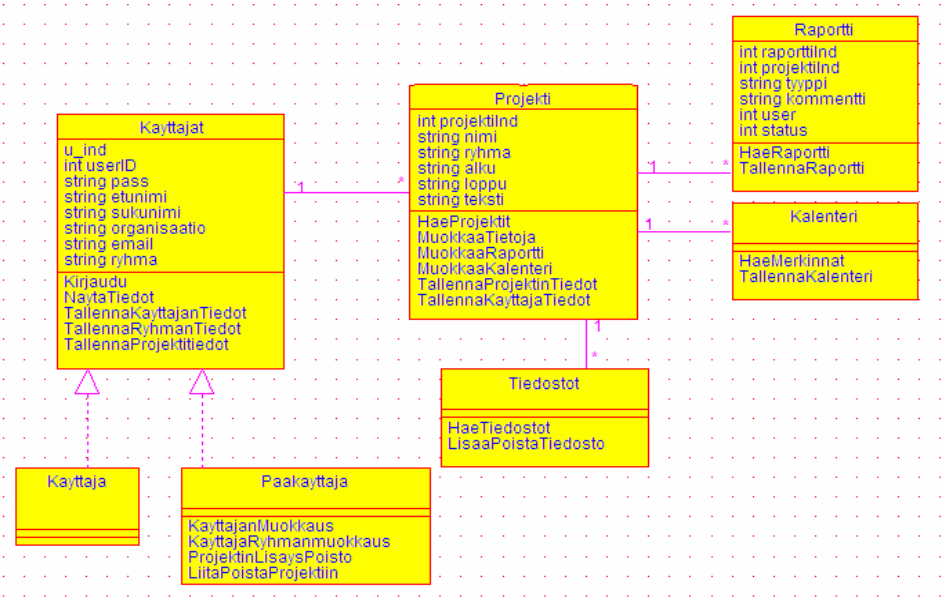
Kuva 9. ER-kuvaus sovelluksen tietokannan tauluista.

5 PROJEKTINSEURANTASOVELLUKSEN SUUNNITTELU

Sovelluksen suunnittelun pohjana käytettiin vaatimusmäärittelyssä päätettyjä toimintoja ja kokonaisuuksia. Suunnittelun tarkoituksena on muuntaa tarpeiden mukaan tehty määrittely tekniselle kielelle, eli järjestelmän toteutuksen kuvaukseksi. [1, s.67]

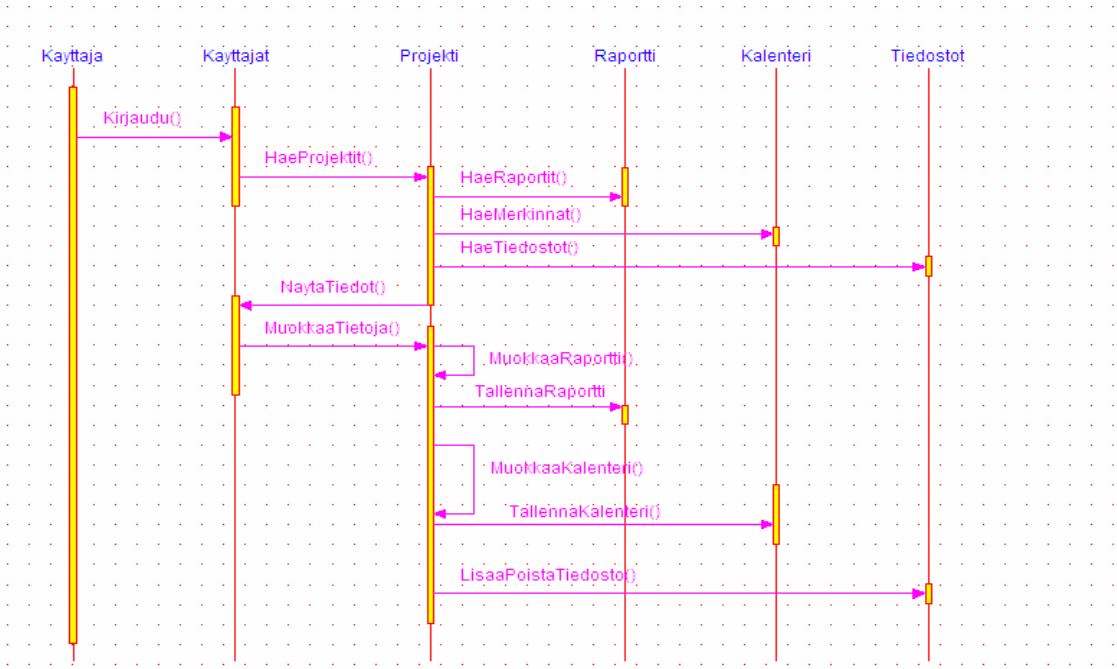
5.1 Luokka- ja oliokaaviot

UML:n luokkakaaviolla kuvataan ohjelmassa esiintyvät attribuutit, niiden toteuttamat toiminnot eli operaatiot sekä luokkien väliset suhteet, assosiaatiot. Prose–sovelluksen luokkakaavio, kuva 10, kuvaa sovellusta hieman yleisemmällä tasolla. Vaikka PHP:n syntaksi on lähelle C–ohjelmointikielen syntaksia ja se sisältää oliopohjaisuuden, niin ohjelmakoodin generoimisella ei nähty saavutettavan suurempaa hyötyä sovelluksen ohjelmoimiseen.

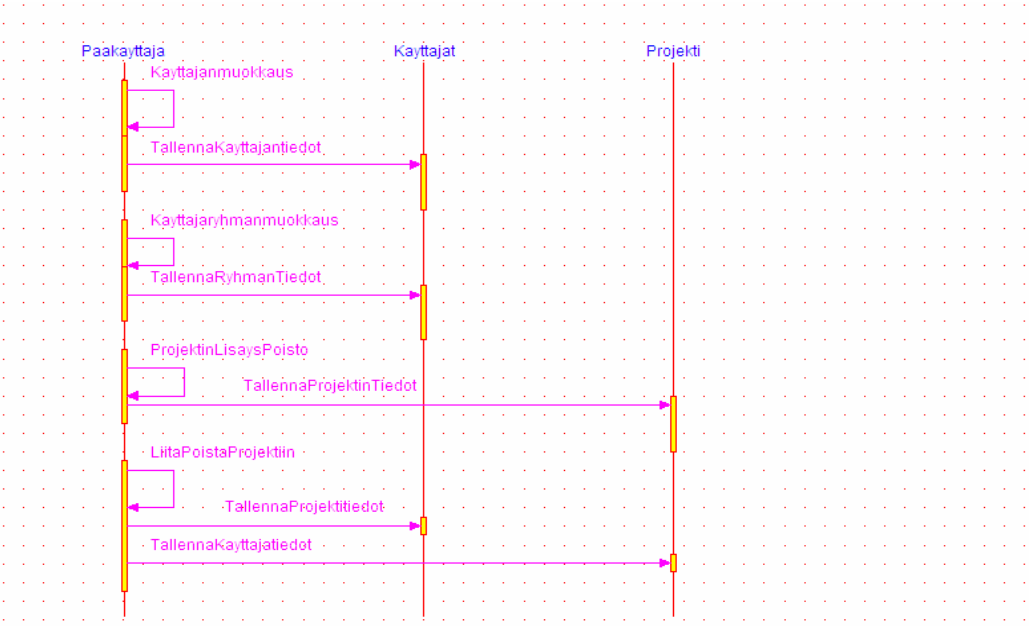


Kuva 10. Prose-sovelluksen luokkakaavio.

Sekvenssikaaviolla kuvataan UML-mallinnuksessa olioiden välistä vuorovaikutusta. Kuvan 11 sekvenssikaavio kuvaa pääkäyttäjän ja tavallisen käyttäjän toimintoja Prose–sovelluksessa ja kuva 12 puolestaan pääkäyttäjän lisätoimintoja.

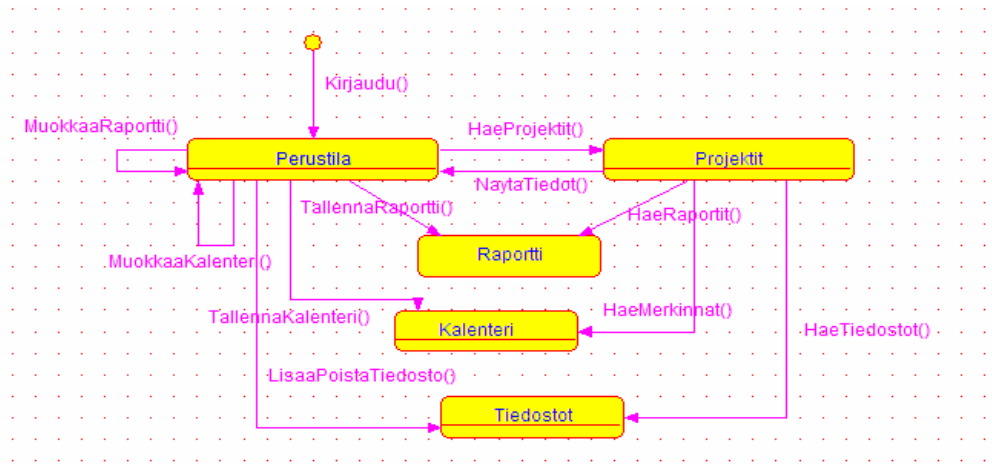


Kuva 11. Sekvenssikaavio tavallisen käyttäjän toiminnoista.

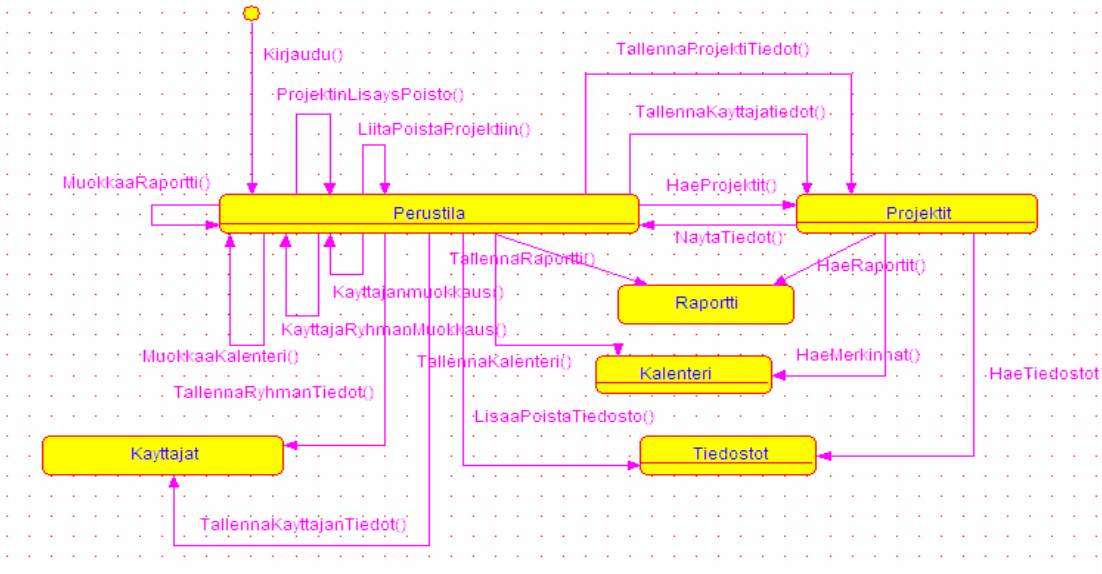


Kuva 12. Sekvenssikaavio järjestelmän pääkäyttäjän lisätoiminnoista.

Tilakaaviolla kuvataan luokan tilakäyttäytymistä, ja sen kannalta olennaisia käsitteitä ovat tilat, tilasiirtymät ja niihin liittyvät toiminnot. UML käyttää David Harelin tilakaavioita, joka on Haikalan ja Märijärven [1, s.122] mukaan notaatioltaan rikkaampi kuin aikaisemmat vastaavat. Kuvissa 13 ja 14 on Prose–sovelluksen tilakaaviot käyttäjän ja pääkäyttäjän toiminnoista.



Kuva 13. Tilakaavio tavallisen käyttäjän toiminnoista



Kuva 14. Tilakaavio pääkäyttäjän toiminnoista

6 SOVELLUKSEN TOTEUTUS JA TESTAUS

Suurin osa sovelluksen toiminnoista toteutettiin PHP:lla, tiedon näyttäminen pääosin HTML –kielellä ja tyyliohjeet määriteltiin CSS:llä. Toteutuksen tietokantakyselyt tehtiin PHP:n versio 5:n mukana tulleen ext/mysqli -laajennuksen mahdollistaman oliopohjaisen ohjelmointitavan mukaisesti. PHP:n 5 -versio on julkaistu heinäkuussa 2004 ja se on hiljalleen alkanut yleistymään kaupallisten webhotellien ja muiden web-sivutilaa tarjoavien keskuudessa. Sovelluksen kalenteri-osio toteutettiin Ajax-tekniikkaa hyväksikäyttäen, jolloin esimerkiksi kuukauden vaihto tai merkinnän näyttö voidaan toteuttaa päivittämällä ainoastaan kalenteri-osio sivulla.

6.1 Tietokannan toteutus

Sovelluksen tietokantana käytettiin yrityksen jo olemassa olevaa tietokantaa, jonne tehtiin tarvittavat tietokantataulut vaatimusmäärittelydokumentin mukaisesti. Tietokannan hallintaa helpotti selaimella käytettävä phpMyAdmin - hallintatyökalu MySQL tietokannoille. Kuvassa 15 on esimerkkinä sovelluksen tietokantataulu tiedostojen tietoja varten, itse tiedoston siirto palvelimelle tapahtuu PHP:n move_upload_file -funktiolla. Kuvan perusteella taulun yksilöivänä tietona eli perusavaimena on ind –sarake. Viiteavaimia taulussa on kaksi, projektiid– ja raportID –sarakkeet, joista ensin mainittu viittaa projektitaulun perusavaimeen ja jälkimmäisellä voidaan viitata tarvittaessa raporttitauluun. Tiedosto siis liittyy aina johonkin tiettyyn projektiin ja se voidaan tarvittaessa myös liittää johonkin raporttiin.

Kuva 15. Sovelluksen tietokantataulu tiedostoja varten.

6.2 Ohjelmointi

Sovelluksen käyttöliittymä toteutettiin HTML:llä ja se pidettiin tarkoituksella mahdollisimman yksinkertaisena, jotta testausvaiheessa testaajien huomio pysyisi sovelluksen toiminnan kannalta keskeisissä asioissa eikä niinkään ulkonäöllisissä seikoissa niitä yhtään väheksymättä. Ulkoasu tulee jatkossa muuttumaan, mutta sovelluksen hieno ulkonäkö ei ollut tämän työn tavoitteena eikä tarkoituksena.

Tietokannan kyselyt, tietojen näyttäminen ja muokkaaminen toteutettiin PHP-kielellä oliopohjaista ohjelmointitapaa käyttäen. Taulukossa 1 on esitetty tietokantakyselyn tekeminen perinteisellä proseduraalisella ohjelmointitavalla ja taulukossa 2 on sama kysely oliopohjaisesti toteutettuna. Kumpikin kyselyistä hakee käyttäjien etu- ja sukunimet users–taulusta ja ne tuottavat täsmälleen samanlaisen tuloksen.

Taulukko 1. Tietokantakysely proseduraalisella ohjelmointitavalla.

```
<?php
$yhteys = mysqli_connect("localhost", "kayttajatunnus", "salasana", "tietokanta");
/* Tarkistetaan tietokantayhteys */
if (!$yhteys)
{
    echo "Tietokantaan yhdistäminen epäonnistui ".mysqli_connect_error();
    exit();
}

/* Kysely, joka hakee kaikki käyttäjien etu- ja sukunimet users -taulusta */
$kysely = "SELECT etunimi, sukunimi FROM users";
if ($tulos = mysqli_query($yhteys, $kysely))
{
/* Käydään tulosjoukko läpi. Fetch_assoc -funktio palauttaa tulosjoukon seuraavan rivin
assosiatiivisena taulukkona, jolloin sen indekseihin voidaan viitata suoraan nimellä */
    while ($rivi = mysqli_fetch_assoc($tulos))
    {
        echo $rivi['etunimi']." ".$rivi['sukunimi']."<br>";
    }
/* Vapautetaan tulos muistista */
    mysqli_free_result($tulos);
}
/* Suljetaan tietokantayhteys */
mysqli_close($yhteys);
?>
```

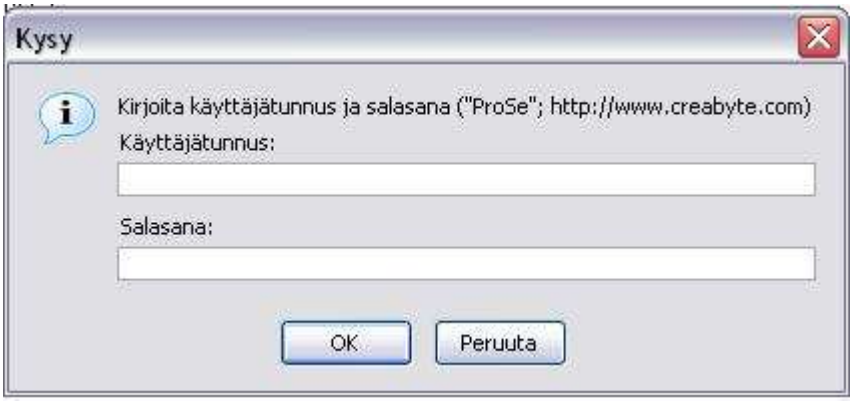
Taulukko 2. Tietokantakysely oliopohjaisella ohjelmointitavalla.

```
<?php
$mysqli = new mysqli("localhost", "kayttajatunnus", "salasana", "tietokanta");
/* Tarkistetaan tietokantayhteys */
if (mysqli_connect_error())
{
    echo "Tietokantaan yhdistäminen epäonnistui ".mysqli_connect_error();
    exit();
}

/* Kysely, joka hakee kaikki käyttäjien etu- ja sukunimet users -taulusta */
$kysely = "SELECT etunimi, sukunimi FROM users";
if ($tulos = $mysqli->query($kysely))
{
/* Käydään tulosjoukko läpi. Fetch_assoc -funktio palauttaa tulosjoukon seuraavan rivin
assosiatiivisena taulukkona, jolloin sen indekseihin voidaan viitata suoraan nimellä */
    while ($rivi = $tulos->fetch_assoc())
    {
        echo $rivi['etunimi']." ".$rivi['sukunimi']."<br>";
    }
/* Vapautetaan tulos muistista */
    $tulos->close();
}
/* Suljetaan tietokantayhteys */
$mysqli->close();
?>
```

6.3 Sisäänkirjautuminen

Sovelluksen jokaisen käyttäjän on kirjauduttava järjestelmään. Autentikaatio toteutettiin PHP:n HTTP-autentikaatiolla, jolloin selain kysyy käyttäjältä käyttäjätunnusta ja salasanaa kuvan 16 mukaisesti.



Kuva 16. Sisäänkirjautuminen.

Käyttäjän syöttämää käyttäjätunnusta ja salasanaa verrataan tietokannan users-taulun tietoihin. Jos käyttäjä löytyy taulusta, niin taulusta haetaan käyttäjän ID, käyttäjätaso, etu- ja sukunimi. Haetut tiedot tallennetaan istunto- eli sessiotietoihin, jolloin tiedot säilyvät siihen saakka kunnes käyttäjä lopettaa session tai sulkee selaimen. Sovellus päättää sessiotietojen perusteella ne toiminnot ja tiedot jotka kyseiselle käyttäjälle on määritetty. Sessiotiedot tallennetaan PHP:ssä muuttujaan $_SESSION ja keskeisenä erona esimerkiksi evästeisiin on se, että sessiotiedot säilytetään palvelimella eikä niitä välitetä lainkaan selaimelle.

Sovelluksen kannalta on kaksi käyttäjätasoa, tavalliset käyttäjät ja pääkäyttäjät. Pääkäyttäjä luo tavalliset käyttäjät ja antaa oikeudet projekteihin vaatimusmäärittelyn mukaisesti. Käyttäjän kirjautuminen varmistetaan sovelluksen jokaisen tiedoston alussa.

6.4 Sovelluksen toimintojen toteutus

Käyttäjän kirjauduttua sovellukseen ladataan index.php-sivu, jossa näytetään uusimmat projektit, raportit, kalenterimerkinnät ja tiedostot. Sovellus koostuu useasta www-sivusta joiden käyttöoikeus määräytyy kirjautumisessa muodostettujen sessiotietojen perusteella. Periaate kaikissa toimintosivuissa on sama, tietojen päivitystä varten on toiminnosta riippuen sovel-

luksessa oma sivunsa. Nämä sivut sisältävät HTML:n form- eli lomake-elementin, jossa on kentät päivitettäviä tietoja varten. Lomakkeelta tiedot lähetetään erityiselle sivulle joka huolehtii tietojen lisäyksestä tai päivittämisestä tietokantaan.

Hallintatunnusten toiminnot haluttiin pitää erillään tavallisten käyttäjän toiminnoista, joten niitä varten luotiin admin-hakemisto. Hakemistossa on toimintosivut projektien lisäykseen ja päivitykseen sekä käyttäjien ja käyttäjäryhmien päivitykseen liittyen. Hakemistoon on käyttöoikeus ainoastaan hallinta-tunnuksilla.

6.5 Kalenterin toteutus Ajax-tekniikalla

Sovelluksen kalenterikomponentti toteutettiin Ajax –tekniikalla. Siten kuukausien vaihto ja merkintöjen näyttäminen onnistuu niin, ettei koko sivua tarvitse ladata uudelleen vaan ainoastaan kalenterin sisältö päivittyy. Toteutuksessa on omat ongelmansa esimerkiksi selaimien eroavaisuuksista johtuen.

Ajax käyttää XMLHttpRequest-objektia datan vaihtamiseen asynkronisesti palvelimen kanssa. XMLHttpRequest ei ole W3C:n hyväksymä standardi, joten objektin voi luoda kahdella tavalla. Internet Explorer –selaimella XMLHttpRequest toteutetaan ActiveX-komponenttina, kun taas muut selaimet toteuttavat sen sisäisenä JavaScript-oliona. Taulukossa 3 on esitetty XMLHttpRequest –objektin ilmentymän luominen selainriippumattomalla tavalla. [12, s.25]

Taulukko 3. XMLHttpRequest -objektin ilmentymän luominen. [12, s.26]

```
var XmlHttp:

function createXMLHttpRequest()
{
   if(window.ActiveXObject)
   {
/* Tarkistetaan, tukeeko selain ActiveX –komponentteja. Jos tukee, niin luodaan
    HMLHttpRequest –objekti ActiveX –komponentin avulla */
      xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
   }
   else if (window.XMLHttpRequest)
   {
/* Muuten luodaan HMLHttpRequest –objekti aivan kuten mikä tahansa olio */
      xmlHttp = new XMLHttpRequest();
   }
}
```

Kalenterikomponentti koostuu HTML:n div-tageista eli elementeistä, jotka määrittävät jonkun tietyn lohkon HTML-dokumentissa. Taulukko 4 kuvaa kalenterin rakenteen, johon kuuluu kolme div-elementtiä. Ensimmäinen div-elementti, jonka id on kalenteri, on itse kalenterinäkymää varten, toinen on merkinnän tekstin asemointia varten ja kolmas itse merkinnän tekstiä varten. Itse toiminta on yksinkertaistettuna sellainen, että ensin luodaan XMLHttpRequest-objekti ja haetaan sisältö palvelimelta tekstimuotoisena responseText-ominaisuutta käyttäen tai XML-oliona responseXml:n avulla. Sen jälkeen päivitetään div-elementin sisältö sijoittamalla div-elementin innerHTML-arvoksi XMLHttpRequest-objektin responseText-ominaisuuden arvo. Taulukossa näkyvät myös kalenterimerkintöjen piilottamiseen liittyvät hiiren tapahtumakäsittelijät.

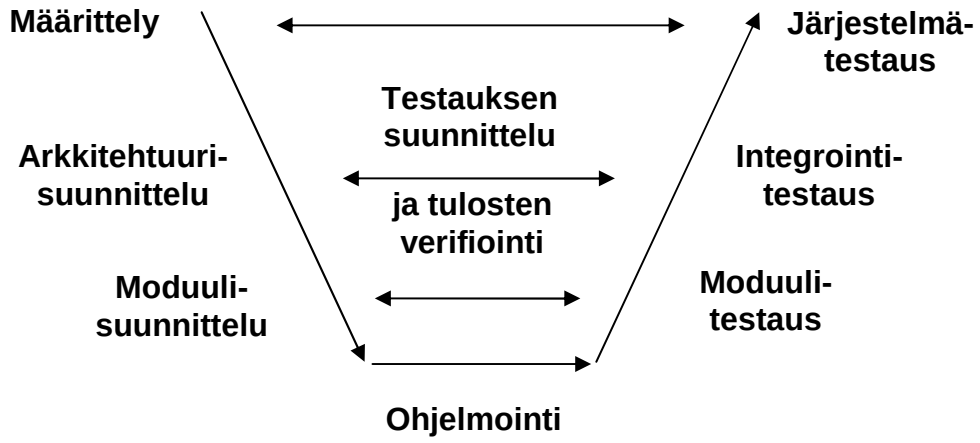
Taulukko 4. Kalenterin rakenne.

```
<div id="kalenteri"  style="position: relative;" onmouseover="cflag=1;" onmouseout="cflag=0;
closeCalendar();"></div>
<div id="naytto" style="position: absolute; z-index:99; width:291;"  onmouseover="cflag=1;"
onmouseout="cflag=0; closeCalendar();">
   <div id="teksti" onmouseover="cflag=1;" onmouseout="cflag=0;"></div>
</div>
```

Kalenteri vaatii että käyttäjä sallii JavaScriptin ajamisen selaimessaan ja tämä tulee ohjeistaa sovelluksen käyttäjille. XMLHttpRequest-objektin pyytämän resurssin on sijaittava saman verkkotunnuksen sisällä kuin missä skriptin suoritus on aloitettu, joten käyttäjän kannalta Ajax-tekniikka on suhteellisen turvallista jos käyttäjä liikkuu yleisesti tunnetuilla tai luotetuilla sivustoilla.

6.6 Testaus

Testaus suoritettiin V-mallin mukaisesti, joka on esitetty kuvassa 17. Testitapaukset suunniteltiin vaatimusmäärittelyn mukaisesti. Moduulitestauksessa testattavana on yksittäinen moduuli, ja se käsittää pienimpien ohjelmayksiköiden, esimerkiksi funktioiden, testauksen. Integrointitestauksessa moduuleita tai moduuliryhmiä yhdistellään ja testaus keskittyy moduulien välisten rajapintojen toimivuuden tutkimiseen. Järjestelmätestauksen kohteena on koko järjestelmä ja tässä vaiheessa tuloksia verrataan määrittelydokumentaatioon. [1.], [13]



Kuva 17. Testauksen V-malli [1, s.263].

Prose-sovelluksen moduulitestaus suoritettiin ohjelmointivaiheessa. Suurin osa virheistä löytyi tässä vaiheessa. Löydetyt virheet olivat tyypiltään pääasiassa käännöksenaikaisia virheitä, eli puolipisteiden ja sulkujen puuttumisesta johtuvia syntaksivirheitä. Tietokantakyselyistä löytyi myös muutama looginen virhe, jotka havaittiin kyselyn tuloksia phpMyAdmin-ohjelman avulla tietokannasta tarkasteltaessa. Moduulitestauksessa löydetyt virheet korjattiin välittömästi ja suoritettiin testitapaus uudelleen niin monta kertaa että virhe saatiin korjattua.

Integrointitestaus suoritettiin osin jo moduulitestausvaiheessa. Integrointi tapahtui kokoavasti, *bottom up,* eli alimman tason moduuleista ylöspäin. Alimman tason moduuleita ovat sellaiset moduulit, jotka eivät käytä tai kutsu muita moduuleja. Tässä vaiheessa sovellusta testattiin myös eri selaimilla, ja huomattiin että kalenterikomponentin toiminta eri selaimilla oli kovinkin kirjavaa. Suurin työ tässä vaiheessa olikin saada kalenteri toimimaan selainriippumattomasti.

Lopuksi suoritettiin järjestelmätestaus vaatimusmäärittelyyn pohjautuvan testaussuunnitelman testitapausten mukaisesti, liite 2. Tässä vaiheessa sovelluksesta löytyi yksi suurempi virhe, sovelluksen hallintaosion käyttöoikeudet eivät toimineet suunnitellusti.

7 TYÖN ANALYSOINTIA

Sovellukselle asetettu tavoite helppokäyttöisyydestä toteutui työssä hyvin ja työn läpivieminen onnistui ilman suurempia hankaluuksia. Työn edetessä alkuperäisiin suunnitelmiin tehtiin muutamia tarkennuksia lähinnä tietokannan taulujen osalta.

Sovelluksessa toteutettu kalenteri aiheutti toteutusvaiheessa eniten työtä. Selaimien erilaiset tulkinnat Ajaxilla toteutetusta kalenterista olisi voitu kiertää toteuttamalla kalenteri iframe-elementtiin, mutta kalenteri haluttiin saada toimivaksi koska samantapainen kalenteri oli tehtävä toiseen projektiin, jossa iframe-toteutusta ei voitu käyttää.

Työn yhtenä tavoitteena oli tutkia PHP 5:n ext/mysqli–laajennusta. Niinpä kaikki työn tietokantakyselyt toteutettiin laajennuksen mahdollistaman olio-pohjaisen ohjelmointitavan mukaisesti. Laajennuksen ja uusien MySQL:n versioiden myötä tietokantakyselyt ovat aiempaa nopeampia ja tietoturvaominaisuuksiltaan parempia esimerkiksi salasanojen käsittelyn osalta. Käytännössä lisääntynyttä nopeutta ei kuitenkaan huomaa, ja joidenkin tutkimusten mukaan vanhemmat versiot toimivat tietynlaisissa tietokantakyselyissä yhtä nopeasti tai jopa nopeammin kuin uusi. Aiempaan proseduraaliseen ohjelmointitapaan tottuneena oliopohjaisuuteen tottuminen vei oman aikansa, mutta sen jälkeen kokemukset olivat pelkästään positiivisia. Työn ohjelmoimisessa käytetty PHP-editori ei osannut oliopohjaisen ohjelmointitavan syntaksia, joten uudemman editorin käyttö on suositeltavaa. Laajennus mahdollistaa myös muista relaatiotietokantajärjestelmistä tutut valmistellut kyselyt, näille ei tämän työn puitteissa ollut kuitenkaan käyttöä. Lopputuloksena on, että ext/mysqli-laajennuksen käyttö on suositeltavaa missä se vain on mahdollista.

Projektin läpivieminen pysyi alkuperäisessä aikataulussa, ja sen lopputuloksena saatiin toimiva sovellus. Sovellus olisi sellaisenaan valmis käyttöönottoon, mutta sitä päätettiin siirtää tulevaan ajankohtaan muutaman lisätoiminnon toteutuksen tarpeesta. Sovelluksen ulkonäkö on tarkoitus tehdä uusiksi ja sovellukseen on tarkoitus tehdä pääkäyttäjille oma muistilapputoiminto. Toteutettavia lisätoimintoja ovat myös automaattisesti päivittyvä sivu, jolla seurataan uusimpia sovelluksen tietoja ja tapahtumia sekä toiminto, jolla sovellukseen voidaan lisätä raportteja suoraan meneillään olevasta projektista. Sovelluksen tiedonsiirto on myös jatkossa tarkoitus suojata HTTPS-protokollan avulla.

8 YHTEENVETO

Työn tavoitteena oli luoda toimiva ja helppokäyttöinen sovellus projektien seurantaan. Sen tuli helpottaa yrityksen ja sen asiakkaiden sekä muiden sidosryhmien välistä viestintää. Nämä tavoitteet saavutettiin.

Tässä työssä käytiin läpi perinteisen ohjelmistotuotannon prosessia pääpiirteittäin, jonka mukaisesti tämäkin projekti vietiin läpi. Esiteltiin verkkosovellusten tekemiseen liittyviä tekniikoita ja työkaluja kuitenkin niin, että pääpaino oli tässä projektissa käytetyissä tai siihen olennaisesti liittyvissä menetelmissä. Sovelluksen suunnittelu- ja toteutusvaiheet esitettiin yleisellä tasolla, tarkemman ohjelmakoodin läpikäymisen ei nähty olevan mielekästä työn eikä etenkään työn tilaajan kannalta katsottuna.

Internetiin liittyvä ohjelmistokehitys on nopeatempoista, vaatimukset muuttuvat jatkuvasti ja sovellusten toteuttamiseen ei usein ole kovin paljon aikaa. Niinpä tällekin sovellukselle tuli projektin edetessä toiveita lisätoiminnoista, joten sovelluksen lopullinen käyttöönotto siirtyy myöhempään ajankohtaan ja sovelluksen ulkonäkö tulee vielä muuttumaan. Lisätoiminnot ovat kuitenkin helppo toteuttaa, koska tämän työn tuloksena saatu sovellus suunniteltiin siten, että sitä voidaan tarpeen mukaan laajentaa.

Sovellus tehtiin todelliseen tarpeeseen. Sen lopullisen hyödyn ja käytettävyyden mittarina toimii se, että miten sovellus onnistuu vakiinnuttamaan paikkansa yrityksen ja sen asiakkaiden viestinnän välineenä.

LÄHTEET

1. Haikala, Ilkka - Märijärvi, Jukka. Ohjelmistotuotanto. Jyväskylä, Gummerus Kirjapaino Oy, 1998. ISBN 951-762-696-7

2. Laine, Harri – Paakki, Jukka. Ohjelmistotuotanto. 2003, luettu 4.4.2007 [PDF-dokumentti]
   http://www.cs.helsinki.fi/u/paakki/ohtuk03-luento2-bw.pdf

3. Heinisuo, Rami. PHP ja MySQL. Jyväskylä, Talentum Media Oy, 2001, ISBN 951-762-770-X

4. Wikipedia. Verkkosivu, viimeksi muutettu 5.2.2007, luettu 9.4.2007, [WWW-dokumentti]
   http://fi.wikipedia.org/wiki/Verkkosivu

5. Wikipedia. Verkkopalvelu, viimeksi muutettu 13.2.2007, luettu 9.4.2007, [WWW-dokumentti]
   http://fi.wikipedia.org/wiki/Verkkopalvelu

6. Lahtonen, Tommi. Relaatiotietokannat, 2004, luettu 9.4.2007, [WWW-dokumentti]
   http://appro.mit.jyu.fi/doc/tiedonhallinta/tietokannat/index1.html

7. Wikipedia. UML-mallinnus, viimeksi muutettu 21.3.2007, luettu 9.4.2007, [WWW-dokumentti]
   http://fi.wikipedia.org/wiki/Uml

8. Koskimies, Kai – Koskinen, Johannes – Maunumaa, Mika – Peltonen, Jari – Selonen, Petri – Siikarla, Mika – Systä, Tarja. UML työvälineenä ja tutkimuskohteena, luettu 9.4.2007 [PDF-dokumentti]
   http://tkts.fi/lehti/a21/uml.pdf

9. Kollannus, Sami. Johdatus PHP-kieleen, 2004, luettu 13.3.2007, [WWW-dokumentti]
   http://www.cs.jyu.fi/~kolli/ITK215/PHP/

10. Wikipedia. Ajax (programming), viimeksi muutettu 13.3.2007, luettu 13.3.2007, [WWW-dokumentti]
http://en.wikipedia.org/wiki/Ajax_%28programming%29

11. Kettunen, Sami – Tapper, Teemu – Lehtinen Sami. Ajaxilla sujuvuutta verkkopalveluiden käytettävyyteen. 2006, luettu 13.3.2007 [PDF-dokumentti]
http://www.samcom.fi/Miten_AJAXia_voi_hyodyntaa_verkkopalveluissa.pdf

12. Asleson, Ryan - Schutta, Nathaniel T. Ajax tehokas hallinta. Jyväskylä, Gummerus Kirjapaino Oy, 2007. ISBN 952-5655-08-3

13. Kautto, Tuomas. Ohjelmistotestaus ja siinä käytettävät työkalut, luettu 2.4.2007, [WWW-dokumentti]
http://www.mit.jyu.fi/opiskelu/seminaarit/ohjelmistotekniikka/testaus/

LIITTEIDEN LUETTELO

KÄYTTÖTAPAUS:	Kirjautuminen
YHTEENVETO:	Käyttäjä tai pääkäyttäjä kirjautuu järjestelmään
TOIMIJAT:	Käyttäjä tai pääkäyttäjä
EHDOT:	Toimijalla on tunnukset järjestelmään
KUVAUS:	Toimija syöttää kirjautumisikkunaan käyttäjätunnuksensa ja salasanansa. Järjestelmä tarkistaa tiedot tietokannasta ja päättää päästääkö toimijan sisään järjestelmään.
POIKKEUKSET:	Jos internet-yhteys katkeaa, tai palvelimelle ei saada yhteyttä
LOPPUTULOS:	Toimija pääsee sisään järjestelmään

KÄYTTÖTAPAUS:	Projektin etenemisen seuranta
YHTEENVETO:	Käyttäjä tai pääkäyttäjä voi selata projektien tietoja
TOIMIJAT:	Käyttäjä tai pääkäyttäjä
EHDOT:	Kirjautuminen onnistunut ja käyttäjällä on oikeudet selata kyseisen projektin tietoja.
KUVAUS:	Toimija voi selata projektien tietoja
POIKKEUKSET:	Jos internet-yhteys katkeaa, tai palvelimelle ei saada yhteyttä
LOPPUTULOS:	Toimija selaa projektin tietoja

KÄYTTÖTAPAUS:	Päivitä projektin tietoja
YHTEENVETO:	Käyttäjä tai pääkäyttäjä voi päivittää projektien tietoja
TOIMIJAT:	Käyttäjä tai pääkäyttäjä
EHDOT:	Kirjautuminen onnistunut ja käyttäjällä on oikeudet kyseiseen projektiin.
KUVAUS:	Toimija voi lisätä tai poistaa virheilmoituksia, kalenterimerkintöjä sekä tiedostoja projektiin liittyen
POIKKEUKSET:	Jos internet-yhteys katkeaa, tai palvelimelle ei saada yhteyttä
LOPPUTULOS:	Toimija päivittää projektin tietoja

KÄYTTÖTAPAUS:	Lisää, päivitä tai poista projekti
YHTEENVETO:	Pääkäyttäjä voi lisätä, päivittää tai poistaa projektin järjestelmästä
TOIMIJAT:	Pääkäyttäjä
EHDOT:	Kirjautuminen onnistunut.
KUVAUS:	Toimija voi lisätä, päivittää tai poistaa projektin tietoja, liittää käyttäjiä tai käyttäjäryhmiä projektille käyttäjiksi.
POIKKEUKSET:	Jos internet-yhteys katkeaa, tai palvelimelle ei saada yhteyttä
LOPPUTULOS:	Toimija päivittää projektin tietoja

KÄYTTÖTAPAUS:	Käyttäjän lisäys, tietojen päivitys ja poisto
YHTEENVETO:	Pääkäyttäjä voi lisätä, päivittää tai poistaa käyttäjiä järjestelmästä
TOIMIJAT:	Pääkäyttäjä
EHDOT:	Kirjautuminen onnistunut.
KUVAUS:	Toimija voi lisätä, päivittää tai poistaa käyttäjän tietoja, lisätä, päivittää ja poistaa käyttäjäryhmiä. Lisäksi käyttäjä voidaan liittää johonkin käyttäjäryhmään.
POIKKEUKSET:	Jos internet-yhteys katkeaa, tai palvelimelle ei saada yhteyttä
LOPPUTULOS:	Toimija päivittää projektin tietoja

Nro	Toiminto	Testitapaus	Hyväksymiskriteerit
1	Kirjautuminen	Kirjautuminen järjestelmään pääkäyttäjän tunnuksella.	Kirjautuminen onnistuu, käyttäjätason tunnistus onnistuu
2	Kirjautuminen	Kirjautuminen tavallisen käyttäjän tunnuksella	Kirjautuminen onnistuu, käyttäjätason tunnistus onnistuu
3	Kirjautuminen	Kirjautuminen väärällä tunnuksella	Kirjautuminen epäonnistuu
4	Projektin etenemisen seuranta	Tavallisen käyttäjän projektien näyttäminen siten että käyttäjä näkee vain projektit joihin on oikeus	Käyttäjälle näkyy vain projektit joihin on oikeudet
5	Projektin etenemisen seuranta	Pääkäyttäjän tulee nähdä kaikki projektit	Kaikkien projektien seuraaminen onnistuu
6	Päivitä projektin tietoja	Kalenterimerkintöjen lisäys, poisto ja päivitys, bugiraportointi ja tiedostojen lisäys ja poisto niin, että tavallinen käyttäjä voi tehdä merkintöjä vain omiin projekteihinsa.	Projektien tietojen päivitys onnistuu
7	Päivitä projektin tietoja	Kalenterimerkintöjen lisäys, poisto ja päivitys, bugiraportointi ja tiedostojen lisäys ja poisto niin, että pääkäyttäjä voi tehdä merkintöjä kaikkiin projekteihin.	Projektien tietojen päivitys onnistuu
8	Lisää, päivitä tai poista projekti	Projektien tietojen päivitys vain pääkäyttäjätunnuksilla	Projektien tietojen päivitys onnistuu
9	Käyttäjän lisäys, päivitys ja poisto	Käyttäjien ja käyttäjäryhmien päivitys vain pääkäyttäjätunnuksilla	Käyttäjien tietojen päivitys onnistuu

Nro	OK/FAIL	Kuvaus (jos Fail)
1	FAIL, 3.2.2007, JL	Kirjautuminen yleiselle puolelle onnistuu, hallintapuoli ei edes kysy tunnuksia.
2	OK, 3.2.2007, JL	
3	OK, 3.2.2007, JL	
4	OK, 3.2.2007, JL	
5	OK, 3.2.2007, JL	
6	OK, 3.2.2007, JL	
7	OK, 3.2.2007, JL	
8	OK, 3.2.2007, JL	
9	OK, 3.2.2007, JL	