Tampereen ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Kimmo Havantola

Opinnäytetyö

Tuotehallintajärjestelmän suunnittelu ja toteutus

Case Proto Tool

Työn ohjaaja Petri Heliniemi
Työn tilaaja Flander Oy
Tampere 3/2010

Tekijä Kimmo Havantola
Työn nimi Tuotehallintajärjestelmän suunnittelu ja toteutus, Case Proto Tool
Sivumäärä 60
Valmistumisaika toukokuu 2010
Työn ohjaaja Petri Heliniemi
Työn tilaaja Flander Oy

Tiivistelmä

Opinnäytetyöni tarkoituksena on kuvata asiakkaan vaatimusten perusteella toteutetun tuotehallintajärjestelmän tietokannan ja käyttöliittymän suunnittelua ja toteuttamista. Asiakasyrityksen laajentumisen vuoksi käytössä ollut sovellus ei kyennyt vastaamaan asiakkaan tarpeisiin sen puuttuvien raportointiominaisuuksien ja käytettävyyden vuoksi.

Toimeksiannon lopputuloksena syntyi WWW-pohjainen tuotehallintajärjestelmä, joka mahdollisti pääsyn sovellukseen kaikkialta asiakkaan sisäverkosta. Tietokanta toteutettiin MySQL-ohjelmistolla ja sen käyttöliittymä PHP:n versiolla 5. Käyttöliittymä ohjelmoitiin proseduraalisen ohjelmoinnin perusteiden mukaisesti, ohjelmoijan vähäisen oliopohjaisten sovellusten kokemusten vuoksi.

Sovelluksen jatkokehitystä on suunniteltu kääntämällä proseduraalinen ohjelmointi oliopohjaiseen ohjelmointiin, sekä lisäämällä käyttäjän mahdollisuutta vaikuttaa omaan näkymäänsä. Jatkokehitys on tällä hetkellä suunnitteluvaiheessa ja lopullisen toteutuksen on laskettu valmistuvan vuoden 2010 loppuun mennessä.

Avainsanat MySQL, PHP, käyttöliittymä, tietokannan suunnittelu

Writer Kimmo Havantola
Thesis Design and implementation of Product Management System, Case Proto Tool
Pages 60
Graduation time May 2010
Thesis Supervisor Petri Heliniemi
Co-operating Company Flander Oy

Abstract

This thesis describes the designing and implementation of a database and user interface based on customers requirements. Due to customer growth, the application previously in use was not able to respond to the needs of a customer due to its missing reporting features and poor usability.

As a result, a web-based product management system was delivered, which allowed users in customers' network to access application. The database was created with the MySQLsoftware and the user interface with PHP 5. The user interface was programmed in accordance with the procedural programming criteria, because of programmer's poor knowledge about the object-oriented programming.

Application's further development is designed to turn the procedural programming to object-oriented programming and increase the users' ability to influence their view. The development is currently on a planning stage and the final implementation is estimated to be complete by the end of year 2010.

Keywords MySQL, PHP, user interface, database design

SISÄLLYSLUETTELO

Termit

DOM	Document Object Model on ohjelmointirajapinta, joka mahdollistaa WWW-dokumenttien sisällön muokkaamisen
DOMAIN	Internetin verkkotunnus. Kirjaimista koostuvia nimiä, joiden avulla verkkoon kytkettyihin koneisiin voidaan viitata paremmin muistettavalla tavalla verrattuna numeroista koostuviin IP-osoitteisiin
DTD	Document Type Definition on yksi SGML- ja XML- kielten yhteydessä käytetyistä rakennemäärittelytavoista. DTD määrittelee rakenteisen dokumentin sallitut ilmentymismuodot elementeille ja attribuuteille, muodostaen uuden merkintäkielen.
GPL-lisenssi	General Public Licence on vapaa ohjelmistolisenssi. Takaa käyttäjälleen oikeuden kopioida, muuttaa ja jakaa edelleen ohjelmia ja niiden lähdekoodia.
HOST	Internettiin tai muuhun verkkoon yhdistetty tietokone, joka hallinnoi ja tarjoaa verkkopalveluita. Suomenkielessä myös isäntä.
HTTP-protokolla	Lyhenne sanoista Hypertext Transfer Protocol on protokolla, jota selaimet ja WWW-palvelimet käyttävät tiedonsiirtoon.
ISAPI	Määritys, joka mahdollistaa Microsoftin ja muiden WWW-palvelinohjelmistojen valmistajien tuotteiden laajentamisen uusilla toiminnoilla.
JavaScript	Alun perin Netscape Communications Corporationin kehittämä pääasiassa WWW-ympäristössä käytettävä komentosarjakieli. Sen avulla on mahdollista lisätä WWW-sivuille dynaamista toiminnallisuutta.
NCSA	National Center of Supercomputing Applications on Illinoisissa toimivan yliopiston yksikkö, joka tarjoaa myös tehokkaita tietokoneitaan tutkijoiden tarpeisiin

Perl	Practical Extraction and Report Language. Larry Wallin kehittämä tulkattava produraalinen skriptimäinen ohjelmointikieli.
SGML	Standard Generalized Markup Language on metakieli, jonka avulla voidaan määritellä dokumenttien merkintäkieliä. Pohjautuu IBM:n Generalized Markup Languageen.
Skripti	Skripti on eräänlainen tietokoneohjelma, joka koostuu usein monista komentosarjoista muodostaen näin kokoelman komentoja.
SSL/TSL	Secure Socket Layer/ Transport Layer Security. Salausprotokolla, jonka avulla voidaan suojata Internet-sovellusten tietoliikenne IP-verkkojen yli. Nykyään SSL/TSL on yksi tavallisimpia tapoja suojata tietoliikennettä.
SQL	Structured Query Language on IBM:n kehittämä standardoitu kyselykieli, jonka avulla on mahdollista tehdä relaatiotietokantoihin erilaisia hakuja, muutoksia ja lisäyksiä.
Tag	Tag on HTML-komento, jonka avulla ohjataan sitä, miten tieto Internet-sivulle tulostuu. Suomenkielessä käytetään usein tagi tai tägi.
W3C	World Wide Web Consortium on kansainvälinen yhteisö, joka kehittää standardeja, joiden avulla Web voi jatkaa kasvuaan.
XML	eXtensible Markup Language. Merkintäkieli tai standardi, jolla tiedon merkitys on kuvattavissa sisällön sekaan. Kieltä käytetään sekä formaattina tiedonvälitykseen järjestelmien välillä että formaattina dokumenttien tallentamiseen. XML on rakenteellinen kuvauskieli, joka auttaa jäsentämään laajoja tietomassoja selkeämmin.
Zend Engine	Avoimen lähdekoodin komentosarjakielinen ydin, joka on tullut tutuksi erityisesti PHP:n ytimenä. Alkujaan Zendin kehittivät Andi Gutmans ja Zeev Suraski, joiden nimistä ydin sai nimensä.

1 Johdanto

WWW-pohjaisten sovellusten suunnittelu ja toteutus on lisääntynyt viime vuosikymmenen aikana merkittävästi parantuneiden verkkoyhteyksien, sekä yritysten globalisoitumisen myötä. Yhä useampi yritys laajentaa toimiaan ulkomaille ja tarvitsee tehtäviänsä varten reaaliaikaista informaatiota globalisoituneen tiimin kehityksestä sekä kompetenssista. Useimmilla suuremmilla yrityksillä on oma sisäverkkonsa, jonka avulla eri puolilla maailmaa sijaitsevat tiimit voivat olla yhteydessä toisiinsa turvallisesti sekä tehokkaasti. Oma sisäinen verkko mahdollistaa myös omien verkkosovellusten toteuttamisen riskittömämmin, kuin yrityksen käyttäessä asiointiinsa julkisia verkkoja.

Asiakasyritys kustomoi laitteitaan omien asiakkaidensa toiveiden mukaisesti, eli brandaa laitteen asiakkaiden määrittelyjen mukaisesti. Alkujaan ainoastaan yhdellä paikkakunnalla toiminut kustomointitiimi laajentui myös muihin maihin, jolloin useampiin käytössä olleisiin työkaluihin jouduttiin tekemään muutoksia. Tiimi käyttää työssään eri laitemalleja, joihin muutoksia tehdään. Ohjelmistototeutuksessa sekä testauksessa käytettävät laitteistot tilataan tehtailta suoraan, ja tiimi käyttää niitä yhtenäisesti eri työtehtäviä varten.

Jokaisella laitteella on myös omat kustannuksensa, joten tiimin sisällä liikkuvia laitteita, sekä muita työn tekemiseen tarvittavia laitteistoja seurattiin Excel-taulukoihin mallikohtaisesti tallennettujen tapahtumien avulla. Kaikki laitteet, jotka olivat tiimin jäsenten käytettävissä, sijaitsivat yhdessä varastohuoneessa, josta niitä haettiin tarpeen vaatiessa käyttöön. Samassa varastossa sijaitsi myös päätelaite, jonka avulla laitteita ja niiden käyttöä seurattiin.

Aikaisempi sovellus käytti tietokantanaan laitemallikohtaisesti nimettyjä Excel-taulukoita, joiden avulla kunkin laitteen viimeksi tallennettu tapahtuma voitiin tarkastaa. Käyttöliittymä oli toteutettu Visual Basic -tekniikan avulla siten, että käyttäjän valitessa laitteen mallin, käyttöliittymä avasi yhteyden vastaavaan Excel-tiedostoon, johon tietoja voitiin syöttää. Käyttäjän tehtävänä oli tallentaa käyttöönottamansa laitteen malli, laitteistotyyppi, valmistuserä, käyttötarkoitus sekä oma nimi tietokantaan, jotta voitiin seurata käyttäjillä käytössä olevia laitteita sekä käyttötarkoitusta. Käyttöliittymän kaikki kentät, paitsi mallinumeron valinta, olivat ns. vapaa kenttiä, eli käyttäjä kykeni syöttä-

mään niihin mitä tahansa merkkejä. Tämän lisäksi käyttäjää ei tunnistettu sovelluksessa mitenkään, vaan kuka tahansa kykeni muokkaamaan tietoja ilman, että siitä olisi jäänyt historiamerkintää. Vanha sovellus ei myöskään tarkastanut tuotteita lisätessä, oliko kyseiseen taulukkoon jo tallennettu tietoa yksilöivän tunniste-koodin perusteella, vaan käyttäjä pystyi tallentamaan samalla tunniste-koodilla useita laitteita tietokantaan.

Toimeksiannon yhtenä tavoitteena oli suunnitella asiakkaan tarpeisiin soveltuva relaatiotietokanta. Tietokannan tuli pitää sisällään tiedot jokaisesta asiakkaan tiimin käytössä olevasta laitteesta sekä muista tarvittavista apuvälineistä. Tietokannan oli pystyttävä mukautumaan asiakkaan muuttuvien olosuhteiden mukaisesti. Toisena tavoitteena oli suunnitella ja toteuttaa tietokantaa hyväksi käyttävä WWW-pohjainen käyttöliittymä, jotta eri puolilla maailmaa olevat tiimin jäsenet pystyisivät käyttämään sitä tehokkaasti myös muilta toimipisteiltä. Käyttöliittymän oli mahdollistettava laitteiden ja muiden välineiden tietojen selektiivinen hakeminen käyttäjän syötteen perusteella ja tarjottava käyttäjälle mahdollisuus muokata tiettyjä ennalta määritettyjä tietoja.

Sovelluksen oli myös kerättävä loki-tiedostoa kaikista käyttäjien tekemistä muutoksista ja mahdollistettava laitteiston ja muiden apuvälineiden jäljitettävyys mm. käyttäjän autentikoinnin avulla. Sovelluksen oli lisäksi tarjottava asiakkaan johdolle ajan tasalla olevaa informaatiota eri toimipisteiden laitemääristä ja malleista, sekä mahdollistettava Excel-raportin laatiminen valituilla hakuarvoilla.

Opinnäytetyön tavoitteena on kuvata sovelluksen suunnittelun ja toteutuksen vaiheet, sekä tekniikat, joiden avulla sovellusta kehitettiin. Käsittelen opinnäytetyössäni ainoastaan sovelluksen kannalta merkittävimpien ohjelmistojen käyttöä, jättäen mm. JavaScriptin esittelyn työn ulkopuolelle. Ensimmäisten lukujen tarkoitus on selventää lukijalle käytettyjen ohjelmistojen taustaa, sekä itse toteutuksessa käytettyjen ohjelmistoversioiden ominaisuuksia. Koska sovellus toteutettiin pitkälti omien taitojen sekä mieltymysten mukaisesti, unohtamatta asiakkaan vaatimuksia, itse suunnittelu- tai toteutusvaiheessa käytettyjen lähteiden merkitys on lopputuloksen kannalta merkityksetön. Erilaisia ohjelmointikeskeisiä sivustoja hyväksi käyttäen pyrin ratkaisemaan vain esiin nousseet ongelmat mahdollisimman nopeasti. Tällaisten sivustojen, kuten esimerkiksi www.ohjelmointiputka.net:n tarjoamat neuvot ja ratkaisumallit on erityisesti ohjelmistototeutuksessa muistettava ottaa käyttöön harkiten, sillä sivustoille neuvoja tarjoavat tahot julkaisevat harvoin tietoja itsestään. Tällöin ohjelmakoodin laadusta ja halutusta

toiminnallisuudesta ei ole takeita ja turvallisuusriski on suurempi. Sivustot toimivat kuitenkin hyvänä mallina esimerkiksi sille, miten tiettyä funktiota voidaan käyttää, ja mitä etuja tai haittoja sen käyttö saattaa aiheuttaa vaikkapa sovelluksen toiminnallisuuden kannalta.

Opinnäytetyön kannalta merkittävimmät lähteet olivat MySQL-tietokannan taustoja, sekä versio-ominaisuuksia tutkittaessa ohjelmiston omat verkkosivut. Ohjelmiston vankka suosio perustuu osaltaan juuri hyvin ylläpidettyihin tukisivuihin, joiden avulla käytössä olevasta MySQL-tuotteesta saadaan tarvittavat tiedot nopeasti ja luotettavasti, sekä mahdolliset kriittiset vikatilanteet kehittäjien tietoon nopeasti. Toteutuksessa käytetyn MySQL versio 5.1 -palvelimen oma referenssimateriaali osoittautui laadukkaasti toteutetuksi, jonka lisäksi ohjelmistojen kehittäjät päivittävät kaikkien MySQL-tuoteperheen ohjelmistoja koskevia julkaisuja hyvin säännöllisesti. Näin verkkolähteiden kannalta suurimmasta ongelmasta – tiedon vanhenemisesta ei tarvinnut kantaa huolta.

Puhuttaessa PHP-sovelluksista, yhdistetään niihin usein myös jokin tietoa varastoiva tietokanta. Niinpä PHP:n (ja osin MySQL:n) osalta opinnäytetyön painetuksi lähteeksi valikoitui Rami Heinisuon vuonna 2003 julkaistu uudistettu 2. painos kirjasta: *PHP ja MySQL Tietokantapohjaiset verkkopalvelut*. Teos kertoo käytännönläheisesti, miten parempia ja toimivampia verkkosovelluksia voidaan rakentaa hyödyntämällä tietokantojen ominaisuuksia. Vaikka kirja kuvaakin sovellusten toteuttamista käyttäen ohjelmistoina PHP:n ja MySQL:n 4. versioita, ovat siinä esitellyt käytännöt ja ominaisuudet hyvin samankaltaisia molempien ohjelmistojen uudemmissa versioissa. Kirjassaan Heinisuo kuvaa havainnollisesti myös sovellusten verkkoturvan merkitystä sekä sen toteutusta. Kirja soveltuu hyvin niin kokeneemmalle kehittäjälle kuin vasta-alkajallekin.

Esiteltäessä PHP:n kehitystä moderniksi ohjelmointikieleksi, ei voi sivuuttaa sen omille verkkosivuille koottua materiaalia. Aivan kuten MySQL:kin tapauksessa myös PHP:n omien verkkosivujen artikkeleja ja dokumentaatioita päivittävät sen kehittäjät, jolloin ajankohtaisimman tiedon saaminen muista lähteistä on käytännössä mahdotonta. Kielen kehittäjät ovat muodostaneet oman yhteisönsä, joka on vastuussa sivustolle julkaistavien dokumentaatioiden, ohjelmavirheiden ja muiden kieleen liittyvien kysymysten tarkastamisesta. Näin sivustoa voidaan pitää luotettavana verkkolähteenä, sen päivitystiheyden, sekä määriteltyjen vastuuhenkilöiden vuoksi.

Ohjelmistoesittelyiden jälkeen kuvaan tarkemmin varsinaista toteutusprosessia. Tietokannan suunnittelu ja toteutus kappaleet pyrkivät ilmentämään, miten sovelluksen tarpeisiin soveltuvaa tietokantaa suunniteltiin, sekä millä tekniikoilla se toteutettiin. Käyttöliittymän suunnittelussa käytettiin *protoilu-menetelmää*, jonka avulla saadaan nopeasti visuaalinen kuva suunnitellusta käyttöliittymästä sekä sen tarvitsemista toiminnallisuuksista.

Viimeisten lukujen tarkoituksena on kuvata lopullisen sovelluksen nykytilaa, sekä mahdollisia jatkokehityssuunnitelmia sovelluksen osalta. Nykytilaa käsittelevään kappaleeseen on myös kuvattu sovelluksen käyttäjien mielipiteitä sekä heidän kehitysehdotuksiaan. Yhteenveto kokoaa opinnäytetyöni osalta asetettujen tavoitteiden saavutukset sekä sen, miten sovellus täytti asiakkaan asettamat tavoitteet. Lisäksi kuvaan, mitä ongelmia havaitsin sovelluksen toteutusvaiheessa ja miten vastaavia ongelmia voidaan välttää tulevaisuuden projekteissa.

2 MySQL

Erilaista tietoa on tallennettu jo useita vuosia erilaisin menetelmin. Ennen tietokoneiden yleistymistä, tieto tallennettiin kirjoittamalla tai painamalla se paperille ja tallentamalla nämä dokumentit muun muassa kansioihin tai laatikoihin. Tietokoneiden yleistyessä tallennettavan tiedon määrä kasvoi sekä monipuolistui, ja sitä voitiin tallentaa erilaisille ulkoisille medioille kuten 5 ¼- ja 3 ½ -tuumaisille levykkeille. Yhteisenä ongelmana niin kansioiden, kuin ulkoisten medioidenkin kohdalla oli kuitenkin tallennetun tiedon hakeminen. Yksittäisen dokumentin etsiminen lukuisten kansioiden tai levykkeiden seasta kulutti valtavasti resursseja. Tietokoneiden sisäiset tallennusmediat, pääsääntöisesti kiintolevyt, mahdollistivat tallennuskapasiteetin valjastamisen tietoa varastoivien tietokantojen käyttöön.

Useat yrityskäyttöön tarkoitetut tietokantaratkaisut olivat kuitenkin vain harvojen, taloudellisesti hyvin toimeentulevien yritysten ulottuvilla niiden korkeiden hankintakustannusten vuoksi. Ruotsalaisen MySQL Ab -nimisen konsultointiyrityksen alkujaan sisäiseen käyttöön suunniteltu ohjelmisto muutti tämän lopullisesti. Suomalaisen Michael Wideniuksen ja ruotsalaisen David Arxmanin vuonna 1995 suunnittelema tietokantasovellus noudattaa asiakas-palvelin-arkkitehtuuria, jossa tietokantaa käytetään palvelinohjelman rajapintojen kautta. MySQL-palvelin tukee useimpien ohjelmointikielten rajapintoja, mukaan lukien oliopohjaiset ohjelmointirajapinnat. Ensimmäinen versio julkistettiin vuotta myöhemmin (Wikipedia 2010a).

Ominaisuuksiltaan MySQL on kaupallisia kilpailijoitaan suvaitsevampi. Sen asennus on yksinkertainen, eikä sen ylläpito vaadi täysipäiväistä huolenpitoa. Ominaisuuksiensa vuoksi MySQL on vakiinnuttanut paikkansa niin pienten kuin keskisuurtenkin yritysten tietokantaohjelmistona. Huolimatta yksinkertaisesta käyttöönotosta, on MySQL ominaisuuksiltaan, kiitos jatkuvan kehitystyön, vakavasti harkittava vaihtoehto yrityksen tai verkkokaupan tietokannaksi. Yhdellä MySQL-palvelimella voi olla useita tietokantoja, jotka voivat sisältää useita tauluja. Sisäiset käyttöoikeudet määrittelemällä voidaan tietokannalle luoda käytännössä rajaton määrä eritasoisia käyttäjätunnuksia tietokantoihin ja tauluihin.

Standardisoitu kyselykieli SQL-92, eroaa MySQL-tietokannan käyttämästä kyselykielestä merkittävästi. Peruskomentojen noudattaessa standardia, on MySQL laajentanut omalta osaltaan SQL-92-komentokantaa, joten standardia SQL-kieltä ei varsinaisesti ole karsittu, ainoastaan muokattu. Standardisoidun SQL-92-komentokannan käyttö on siis mahdollista MySQL-tietokannan yhteydessä, mutta suurin hyöty MySQL-tietokanta-ohjelmistosta saadaan käytettäessä sen omaa komentokantaa. Komentokantojen eroavaisuuksia tarkastellessa, ensimmäinen havaittava ero on taulujen tietotyyppien poikkeavuus. Siinä missä SQL-92-standardi tunnistaa kokonaisluvun tietotyypeiksi määritelmät INT tai INTEGER, mahdollistaa MySQL-komentokanta kokonaisluvun määrittelemisen 42:lla eri tavalla (katso liite 1) (Heinisuo 2003 s, 37 - 50).

MySQL-ohjelmisto myytiin tammikuussa 2008 suurelle tietokantaohjelmistoja valmistavalle Sun Microsystems –yritykselle huikeaan miljardin dollarin kauppahintaan. Kaupan ansiosta Wideniuksen esikoistyttären My:n mukaan nimettyä ohjelmistoa pidetään yhtenä Suomen suurimpana IT-alan menestystarinana. Pitkään ei Sun Microsystems ehtinyt hyödyntämään MySQL-ohjelmiston potentiaalia, kun media ilmoitti ohjelmistojätti Oraclen ostaneen yrityksen 27. tammikuuta 2010.

2.1 MySQL versio 5.1

Jatkuvan kehitystyönsä avustuksella uudemmat MySQL-versiot saavat jatkuvasti lisää ominaisuuksia, jotka parantavat ja tehostavat tietokantaohjelmiston toimintaa. Yhtenä version 5.1 (julkaistu 27. 11. 2008) uusista ominaisuuksista, on *mysql_upgrade*-ohjelma, jonka avulla päivittäminen edellisestä, 5.0 -versiosta versioon 5.1 suoritetaan mahdollisimman automaattisesti. Ohjelma tarkastaa tietokannassa olevat taulut, sekä pyrkii korjaamaan mahdolliset erot versioiden välillä automaattisesti. Samanaikaisesti se päivittää tietokannan päätaulut (grant tables), jotta päivitetyn version uudet ominaisuudet ovat käytettävissä. Päivityksen aikana ohjelma käy läpi kaikkien tietokantojen taulut jolloin, riippuen tietokantojen koosta, päivitys saattaa kestää pitkään. Tuona aikana kaikki taulut ja tietokannat ovat lukittuina, eikä niihin voida muodostaa yhteyttä ennen kuin päivitys on suoritettu. Virhetilanteiden välttämiseksi version päivittäminen on tehtävä portaittain, päivittämällä aina ensin seuraava uusi versio.

Tietokantojen tietomäärät kasvavat jatkuvasti, jolloin tiedon hakeminen suuresta määrästä tietoa on vaarassa hidastua. SQL-standardi ei ota kantaa fyysisen tiedon tallentamiseen, vaan on tarkoitettu toimimaan itsenäisesti skeemojen, taulujen ja rivien määrittelemän tietokantarakenteen mukaisesti. Kehittyneimmät maksulliset tietokantaohjelmistot ovat osaltaan ottaneet käyttöön tallennettavien tietojen fyysisen tallennussijainnin. InnoDB-tietokantamoottori on tukenut pitkään ns. tauluavaruutta ja aikaisemmat MySQL-palvelinohjelmistot on pystytty konfiguroimaan tallentamaan eri tietokannat eri tiedostoihin käyttämällä *symbolisia linkkejä*. Osiointi, jonka tuki lisättiin version 5.1 ominaisuuksiin, lisäsi näitä mahdollisuuksia entisestään.

Osioinnin avulla on mahdollista määritellä yksittäisten taulujen eri osien tallentaminen uuteen tietokantatauluun käyttäjän määrittelemien asetusten mukaisesti. Käyttäjän määrittelemää asetusta kutsutaan osiointifunktioksi, joka MySQL:n tapauksessa voi verrata tietokannasta löytyviä vasteita joko itseisarvoon, yksinkertaiseen lista- tai arvoalueisiin, sisäiseen tarkistefunktioon tai lineaariseen tarkistefunktioon. Käytettävä funktio valitaan käyttäjän määrittelemän osiointityypin perusteella, ja se saa parametreinaan käyttäjän määrittelemän arvon. Arvo voi olla taulun sarakkeen kokonaisluku, tai funktio, joka käsittelee useampia sarakkeita ja palauttaa tuloksena kokonaisluvun. Arvon perusteella osiointifunktio palauttaa vasteen, joka vastaa osiointinumeroa, johon kyseinen tulos pitäisi olla tallennettuna. Osiointifunktio ei voi sisältää SQL-kyselyitä, mutta voi käyttää hyväkseen SQL-komentoja, jotka ovat MySQL-yhteensopivia. Funktion on palautettava aina joko arvo *NULL* tai jokin kokonaisluku toimiakseen. Osiointi otetaan käyttöön taulun määrittelyn tai sen ominaisuuksien muokkaamisen yhteydessä.

Osioinnin avulla voidaan siis luoda omien asetusten mukaisia tauluja nopeuttaaksemme tietokantahakujen suoritusta. Lisäksi osiointi mahdollistaa suuremman tietomäärän tallentamisen yhteen tietokantatauluun kuin mitä olisi mahdollista tallentaa kovalevylle tai tiedosto-osioon. Osioinnin merkittävin hyöty on käsitellä tarpeettomaksi muuttunutta tietoa. Sen sijaan, että jouduttaisiin poistamaan tietokantatauluista suuria määriä tarpeetonta tietoa, voidaan osioidusta taulusta poistaa tarpeettoman tiedon sisältämä osiointifunktio, jolloin tarpeetonta tietoa ei enää tallenneta. Vastavuoroisesti tarpeellisen tiedon lisääminen on yhtä helppoa määrittelemällä osiointifunktio käsittelemään tiettyä tietoa. (MySQL Reference Manual 2010 57 - 64 & 200 - 202 & 1986 - 2021)

2.2 Tietokantamoottori

Tietokannan toimintaa ohjaava tietokantamoottori määrittelee, miten tietoa käsitellään, sekä mitä mahdollisia erityisominaisuuksia tietokannalla on. MySQL mahdollistaa tietokanta-ammattilaisen valitsemaan juuri tarkoitukseensa sopivan tietokantamoottorin useasta eri vaihtoehdosta. MySQL tukee useita eri tietokantamoottoreita, joista seuraavissa kappaleissa esittelen kaksi yleisintä.

2.2.1 MyISAM

MySQL-tietokannan oletuksena (pois lukien Windows-järjestelmät) käyttämä MyISAM-tietokantamoottori perustuu aikaisempaan ISAM moottoriin, jota MySQL versio 5.1 ei enää tue. MyISAM-taulut tallentuvat kiintolevylle kolmeen eri tiedostoon, nimettynä taulun nimen mukaisesti sekä tiedostopäätteellä joka kuvastaa tiedoston tyyppiä. Taulujen sisältämät tiedot tallentuvat *.MYD*-, indeksit *.MYI*- ja taulun formaatit, eli rakenteet *.frm-päätteellä*.

Se, miten tietoa tallennetaan kiintolevyille, kuvaa MyISAMin eroja toiseen suosittuun tiekantamoottoriin InnoDB:n verrattuna. Kaikki numeeriset avainarvot tallennetaan suuruusjärjestyksessä, mahdollistaa tiiviimmän indeksoinnin. Suurempien (yli 63-bittisten) tiedostojen tuki riippuu käytettävästä käyttöjärjestelmästä, sekä tietovarastosta, johon tietoa tallennetaan. Oletusarvoisesti tietokantataulussa voi olla maksimissaan 2^{32} riviä, mutta taulun parametrina annettu *--with-big-tables*–optio lisää rivien määrää $(2^{32})^2$ riviin. Indeksoinnin on tarkoitus nopeuttaa tietokantahakujen suoritusaikaa, joten hyvin suunnitellut ja toteutetut indeksoinnit parantavat tietokannan suorituskykyä. Yksi MyISAM-taulu voi oletusarvoisesti sisältää maksimissaan 64 eri indeksiä, mutta indeksien määrää voidaan muuttaa option *-with-max-indexes=x* -avulla, jossa x:n arvo kuvastaa suurinta mahdollista indeksiarvoa ja voi olla pienempi tai yhtä suuri kuin 128. Indeksi voi sisältää enintään 16 saraketta.

MyISAM-moottori tukee yhden sarakkeen, tietotyypiltään *auto_increment*, käsittelyä päivittämällä automaattisesti sarakkeen arvon INSERT- ja UPDATE-komentojen aikana. Käytettyjä arvoja ei käytetä uudelleen, vaan ne poistuvat käytöstä kun rivi poistetaan tietokannasta. Tämä nopeuttaa *auto_increment*-sarakkeeseen kohdistuvia hakuja valmistajan antaman arvion mukaan jopa 10 %, mikä varsinkin suurissa tietokannoissa on

melkoisen merkittävä ero. MyISAM-moottorissa on myös tuki rinnakkaisille INSERT-lausekkeille monisäikeisyytensä vuoksi. Uusien rivien lisääminen samanaikaisesti on mahdollista, jos tietokannan rivien väleissä ei ole tyhjää tilaa, joka on voinut syntyä samanaikaisesti poistetusta tietorivistä.

MyISAM ei kuitenkaan ole niin luotettava ja varmatoiminen, että sitä voitaisiin käyttää kaikissa tietokantoja hyödyntävissä ohjelmistoissa tietokantamoottorina. Tietokannan pääohjelman *mysqld:n* kaatuessa jonkin järjestelmävirheen vuoksi, tietokoneen sammuessa, käytettävässä laitteistossa ilmenneen vian tai jonkin ulkoisen ohjelman muokatessa palvelimen kanssa yhtaikaisesti jotakin tietokannan taulua, voi lopputuloksena olla tietokannan taulujen korruptoituminen. Korruptoituminen voidaan havaita hakukyselyiden epäonnistuessa tietokantarivien noudossa, tai kun käyttäjälle tarjotaan virheilmoitus jossa kehotetaan korjaamaan väärä tauluavain. (MySQL Reference Manual 2010, s. 1059 - 1065.)

Yllä mainitut virhetilanteet ovat tietokannan ylläpitäjien arkipäivää, joten tiedon eheyden vuoksi MyISAM ei sovellu kriittisiin sovelluksiin, joiden sisältämä tieto on saatettava luotettavasti käyttäjälle ilman tietokannan korruptoitumista. Tämän vuoksi MySQL-ohjelmiston tietokantamoottoriksi on tarjolla vakaampi, virheistä paremmin toipuva vaihtoehto, InnoDB.

2.2.2 InnoDB

Usean käyttäjän tietokantamoottoriksi soveltuvan InnoDB:n eroavaisuudet ovat virheistä toipumisessa, sekä turvallisissa transaktioissa, jotka suojelevat tietokannan eheyttä. Sen rivikohtainen lukitus ja Oracle-tyyppinen lukitsemattomien rivien johdonmukainen lukeminen lisää suorituskykyä huomattavasti. Toisin kuin MyISAM, InnoDB tallentaa tietoa klusterisoituihin indekseihin vähentääkseen pääavaimiin kohdistuvien yleisimpien kyselyiden syöte- ja tulostekuormaa. Sen lisäksi se tukee tietokannan viite-eheyttä FOREIGN KEY -viiteavainten avulla. Viiteavaimia voidaan käyttää ainoastaan ns. pysyviin tauluihin ja vain, jos niiden tietokantamoottorina on InnoDB. Viiteavain, sekä sarakkeen, johon se viittaa, on oltava samaa tietotyyppiä ja samanpituinen, jotta niitä voidaan vertailla ilman tyyppimuunnoksia. Käytettäessä *string*-tyyppisiä viiteavaimia, on mahdollista vertailla avaimia määrittelemättä niiden pituutta.

Tietokannan eheyden säilyttämiseksi InnoDB estää kaikki INSERT- ja UPDATE-komennot, jotka yrittävät lisätä viiteavain-arvoa lapsitauluun, jos päätaulusta ei löydy vastaavaa arvoa. Näin voidaan varmistaa viiteavainten ja tietokannan eheys. Se, miten InnoDB käsittelee UPDATE- ja DELETE -komentoja, riippuu siitä miten käyttäjä on määritellyt viiteavaimen poiston (ON DELETE) ja päivityksen (ON UPDATE) parametrit. Usein käytetty CASCADE-parametri päivittää tai poistaa automaattisesti, päätaulun rivin mukaisesti, myös lapsitaulun viiteavaimen vastaavat rivit. Näin lapsi- tai aputaulut pysyvät päätaulujen mukaisesti ajan tasalla automaattisesti ilman, että käyttäjän tarvitsee niihin erikseen koskea.

Suuria tietomääriä sisältävien tietokantojen moottoriksi soveltuva InnoDB on suunniteltu tarjoamaan maksimaalista suorituskykyä, eikä mikään muu levypohjainen tietokantamoottori vedä vertoja sen tehokkuudelle. Tietokantamoottori ylläpitää omaa puskurimuistia välimuistissaan, joka sisältää tietokannan tiedon lisäksi myös indeksit ja tallentaa ne omaan tauluavaruuteen. Tämä mahdollistaa useista tiedostoista koostuvien laajojen, yli 2Gb:n kokoisten taulujen käsittelyn.

Tietokannan perusperiaatteena on tallentaa tärkeää tietoa siten, että se voidaan tarvittaessa hakea nopeasti käyttäjän saataville. Tietokannasta on syytä ottaa järjestelmällisesti varmuuskopioita mahdollisten järjestelmä- tai laitteistovikojen varalta. Monet tietokantamoottorit vaativat palvelimen sammuttamisen varmuuskopioinnin ajaksi. InnoDB:n *Hot Backup* mahdollistaa käynnissä olevan palvelimen tietojen kopioinnin, mukaan lukien MyISAM-taulut, vaikuttamalla minimaalisesti sen toimintaan ottamalla tietokannasta ns. snapshotin. Varmuuskopioinnin ollessa käynnissä, voidaan tauluihin kohdistuvat luvut ja kirjoittamiset suorittaa niitä keskeyttämättä. *Hot Backup* on Innobase Oy:n kaupallinen ohjelmisto, joten täysin ilmaista varmuuskopiointia sillä ei voida suorittaa. Vastaavalla periaatteella varmuuskopioinnin suorittavan *MySQL Administrator* -ohjelman avulla ajastetut varmuuskopiot voidaan luoda helposti graafisen käyttöliittymän avulla.

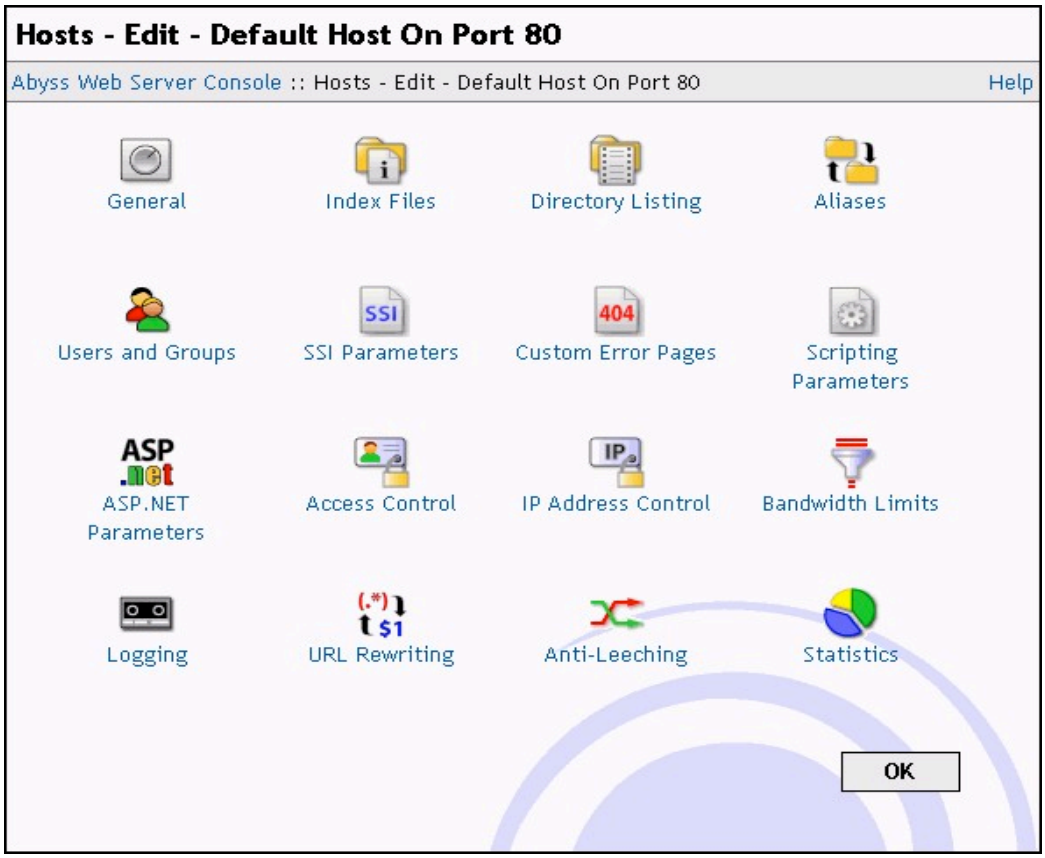
InnoDB:n käyttöön liittyy myös rajoitteita, samoin kuin MyISAM-moottorinkin käyttöön. Yhden taulun sarakkeiden määrä on InnoDB:ssa rajoitettu 1000 sarakkeeseen, joka monessa hyvin suunnitellussa tietokannassa riittää mainiosti. Avaimen maksimipituus on 3500 bittiä, mutta MySQL rajoittaa sen 3072 bittiin. MyISAM rajoittaa tiedostojen koon maksimissaan 2Gb:n kuten yllä mainittiin, mutta InnoDB:ssa on mahdollista

määritellä tiedoston kooksi suurempikin raja-arvo, luomalla useita pienempiä tiedostoja yhden suuren tiedoston sijaan. (MySQL Reference Manual, s. 1065 - 1144.)

3 Abyss Web Server

Abyss Web Server on *Aprelium*-nimisen yrityksen toteuttama sovellus, jonka tavoitteena on tarjota suorituskykyinen WWW-palvelin niin yksityisille kuin yrityskäyttäjillekin. Yritys on perustettu vuonna 2001 ja sen pääkonttori sijaitsee Tunisiassa. Yritys on erikoistunut tuottamaan verkkosovelluksia sekä implementoimaan standardeja yhteysprotokollia. Vuonna 2002 yritys esitteli ensimmäistä kertaa Abyss Web Serverin, jonka yksityiskäyttöön tarkoitettua X1-versiota on ladattu miljoonia kopioita. Yritystarkoitukseen tarkoitettu X2-version lisenssejä on myyty yli 50 eri maahan.

Abyss Web Server on asennusvaatimuksiltaan pienikokoinen WWW-palvelin, ja sitä on saatavilla lähes jokaiselle käyttöjärjestelmälle. Huolimatta pienestä koostaan, se tukee salattuja SSL/TLS-yhteyksiä, salasanasuojausta sekä monia muita suuremman palvelimen ominaisuuksia. Eräs sen vahvuuksista on monikielinen etähallintakonsoli, joka mahdollistaa palvelimen konfiguroinnin toiselta verkossa olevalta päätelaitteelta (Kuva 1).



Kuva 1. Abyss Web Serverin etähallintakonsoli (Abyss Web Server 2010)

WWW-palvelimena toimivan Abyss Web Serverin merkittävimpiin ominaisuuksiin kuuluu muun muassa *virtuaalinen hostaus* (Virtual Hosting). Tämä mahdollistaa usean sivuston ylläpidon saman palvelimen avulla sillä ohjelman avulla voidaan määritellä jokaiselle isännälle omat konfiguraatiot ja mahdollistaa niiden toimiminen täysin itsenäisesti. Virtual Hosting -tuki on mukana ohjelmiston maksullisessa X2-versiossa.

Kevyemmät WWW-palvelimet saattavat muodostaa tietoturva-aukkoja, joiden avulla haittaohjelmat saattavat päästä käsiksi palvelimen sisältöön tai häiritä sen toimintaa. Abyss Web Server pitää sisällään tehokkaan URL-dekoodausmoottorin, jonka avulla se estää haittaohjelmien sekä epäilyttävien palvelupyyntöjen pääsyn palvelimelle. Lisäksi palvelimessa on tehokas hakkerointiesto-järjestelmä, joka tunnistaa hakkerointi- ja DoS-hyökkäykset (Denial of Service) varhaisessa vaiheessa sulkemalla IP-osoitteet joista hyökkäykset tehtiin.

Salattujen yhteyksien tuki mahdollistaa Abyss Web Serverin käytön myös verkkokauppasovellusten palvelimena. Sen vahvan 256-bittisen salauksen avulla pystytään varmistamaan yhteydet ja käyttäjien arkaluontoisten tietojen suojaaminen, noudattamalla alan korkeimpia standardeja. Näitä standardeja ovat muun muassa korttimaksamisen turvallisuutta parantavat Verified by Visa- ja MasterCard SecureCode-todentamispalvelut.

Vikatilanteiden, kuten automaattisten päivitysten varalta, Abyss Web Server voidaan konfiguroida käynnistymään Windows System -palveluna, MacOS X –käynnistyslohkona tai Unix-pohjaisissa järjestelmissä virtuaalisena asemana, *daemonina*. Tämän avulla palvelinta ei tarvitse manuaalisesti käynnistää uudelleen tietokoneen uudelleen käynnistyessä. Palvelimen konfigurointiin ei vaadita erityisosaamista, vaan se onnistuu peruskäyttäjältä varsin helposti hyvin toteutetun hallintapaneelin avulla. (Abyss Web Server 2010.)

4 PHP

PHP on palvelinpuolella suoritettava, tulkattava komentosarjakieli. PHP-sivujen sisältö käsitellään siis WWW-palvelimella ennen kuin sivu ladataan käyttäjän selaimeen. PHP sai alkunsa tanskalais-grönlantilaisen Rasmus Lerdorfin tarpeesta tutkia hänen WWW-sivujensa käyttäjämääriä. Tätä tarkoitusta varten Lerdorf kirjoitti pienen kokoelman Perl-skriptejä, jonka hän nimesi *Personal Home Page Tools*:ksi.

4.1 PHP/FI

Käyttötarkoituksen kasvaessa Lerdorf kirjoitti laajemman C-kieleen pohjautuvan kokoelman, joka kykeni kommunikoimaan tietokantojen kanssa. Se mahdollisti yksinkertaisten dynaamisten WWW-sivujen toteuttamisen. Vuonna 1995 hän julkisti lähdekoodin GPL-lisenssillä, jotta muut voisivat kehittää sitä edelleen, ja nimesi sen PHP/FI:ksi (*Personal Home Page / Forms Interpreter*). PHP/FI sisälsi joukon toiminnallisuuksia, jotka ovat käytössä myös PHP:n uudemmissa versioissa. Syntaksi oli hyvin Perlin kaltainen, joskin rajoitetumpi, yksinkertaisempi ja epäjohdonmukaisempi.

PHP/FI:n versio 2.0 oli vuonna 1997 käytössä jo 50 000 domainissa, eli nimipalvelujärjestelmässä, kattaen n. 1 % koko internetin nimipalvelimista. Vaikka sitä oli kehittämässä useita ihmisiä, oli vastuu pääsääntöisesti edelleen Lerdorfilla. Virallisesti PHP/FI 2.0 julkaistiin joulukuussa 1997 useiden beeta-julkaisujen jälkeen, mutta se korvattiin nopeasti PHP 3 -julkaisulla.

4.2 PHP 3

PHP 3 oli ensimmäinen versio, joka läheisesti muistuttaa PHP:ta jota nykyään käytetään. Se kirjoitettiin Andi Gutmansin ja Zeev Suraskin toimesta vuonna 1997 kokonaan uusiksi, sillä PHP/FI 2.0 oli riittämätön heidän tarkoitukseensa kehittää sähköisen kaupankäynnin sovellusta yliopistoprojektille. Tavoitteenaan yhdistää ja lisätä PHP/FI 2.0:n käyttäjäkuntaa, Gutmans, Lerdorf ja Suraski päättivät tehdä yhteistyötä ja julkaisivat PHP 3:n virallisena seuraajana PHP/FI 2.0:lle, jonka kehitys pysähtyi.

Yksi PHP 3:n vahvuuksista oli voimakkaasti laajennettavissa olevat ominaisuudet. Sen lisäksi, että loppukäyttäjille tarjottiin kiinteitä infrastruktuureita useaan tietokantaan, protokolliin ja ohjelmointirajapintoihin, houkuttelivat PHP 3:n laajennettavat ominaisuudet mukaan useita muita kehittäjiä, jotka loivat siihen lisämoduuleita. Tämä oli avain PHP 3:n valtavaan menestykseen. Muita PHP 3:ssa esiteltyjä ominaisuuksia olivat oliopohjaisen ohjelmoinnin tuki sekä tehokkaampi ja johdonmukaisempi syntaksi. Kokonaan uudelleen kirjoitettu kieli julkaistiin myös uudella nimellä poistamalla siitä ainoastaan henkilökohtaiseen käyttöön viittaavan lisäyksen, joka PHP/FI 2.0:ssa oli. Kieli nimettiin PHP:ksi, joka on rekursiivinen lyhenne sanoista *Hypertext Preprocessor*.

Vuoden 1998 loppuun mennessä PHP:lla oli kymmeniä tuhansia käyttäjiä ja se oli asennettuna satoihin tuhansiin sivustoihin. Enimmillään PHP 3 kattoi noin 10 % koko Internetin WWW-palvelimista. PHP 3 julkaistiin virallisesti kesäkuussa 1998, vietettyään 9 kuukautta julkisessa testikäytössä.

4.3 PHP 4

Pian PHP 3:n julkaisun jälkeen, talvella vuonna 1998, Gutmans ja Suraski alkoivat kirjoittaa PHP:n ydintä uudelleen. Suunnittelun tavoitteena oli parantaa monimutkaisten sovellusten suorituskykyä ja parantaa PHP:n ytimen koodin modulaarisuutta. PHP 3 mahdollisti uudet ominaisuudet ja laajan tuen useille kolmannen osapuolen ohjelmointirajapinnoille sekä tietokannoille, mutta sitä ei ollut suunniteltu käsittelemään näitä monimutkaisia sovelluksia tehokkaasti.

Uusi *Zend-moottori* (lyhennetty kehittäjiensä etunimistä, Zeev ja Andi) vastasi näitä tavoitteita ja esiteltiin ensimmäisen kerran vuoden 1999 puolessa välissä. PHP 4 perustuu tähän moottoriin ja yhdessä lukuisten lisäominaisuuksiensa kanssa julkaistiin virallisesti toukokuussa 2000, miltei kaksi vuotta edeltäjänsä PHP 3:n jälkeen. Parantuneen suorituskyvyn lisäksi PHP 4 sisälsi myös muita keskeisiä ominaisuuksia, kuten tuen usealle eri WWW-palvelimelle, HTTP-istunnot, tulostuksen puskurointi, turvallisempia tapoja käsitellä käyttäjän syötteitä, sekä useiden muiden kielien rakenteita.

4.4 PHP 5

Vaikka jo aiemmat versiot PHP:sta ovat tukeneet olio-ohjelmointia, on niiden käyttö ollut hyvin rajoittunutta ja puutteellista. Vuonna 2005 julkaistun PHP:n version 5 mukana tuoma täysi olio-ohjelmointituki mahdollisti kielen käytön myös oliopohjaisissa ratkaisuissa. Olio-ohjelmoinnin lisäksi PHP:n versio 5 sisältää sisäänrakennetun tietokantamoottorin (SQLite), jonka avulla tietokantapohjaisten sovellusten toteuttaminen on mahdollista ilman erillisen tietokantasovelluksen asentamista. SQLite on kuitenkin karsittu versio ja soveltuu ainoastaan pienimuotoisten tietokantasovellusten tietokannaksi. Uusi versio paransi myös suosittujen XML-tiedostojen käsittelyä ja lisäsi tukea mm. SOAP-palveluille. Viimeisin versio 5.3.2 on julkaistu 04.03 2010 ja tuki PHP:n 4. versiolle päättyi elokuussa 2008. (PHP.net 2010.)

PHP:n versio 6 on kehitteillä samanaikaisesti kun versio 5:n ominaisuuksia parannellaan. Uuden version odotetaan parantavan aikaisempien versioiden tietoturva-aukkoja poistamalla muun muassa käytöstä *register_globals*-muuttujat. Muuttujien avulla lomakkeelta lähetettyjen tietojen sekä kyselylausekkeiden muuttujat olivat käytössä koko ohjelmassa mahdollistaen näin yksinkertaisten skriptien kirjoittamisen nopeasti. Muuttujien ollessa yleisiä, huomasivat hakkerit mahdollisuuden käyttää näitä muuttujia omien ohjelmiensa käynnistämiseen ja sivuston tietojen tallentamiseen. Toinen muutos uuteen versioon tulee olemaan ei ASCII-merkkien tuki luokkien, funktioiden ja merkkijonojen nimeämisessä. Virallista julkaisupäivämäärää ei ole tiedotettu. (Wikipedia 2010b)

5 XHTML JA CSS

Kaikilla julkaisuilla, dokumenteilla ja asiakirjoilla on oma rakenteensa, joka jakaa sisällön eri sivuille, otsikoihin sekä kappaleisiin. Internet-sivustojen rakenne toteutetaan joko HTML- tai XHTML-kielellä ja sen ulkoasua muokataan pääosin ulkoisilla tyylitiedostoilla (CSS).

5.1 XHTML

Vuonna 1989 kansainvälisen energiafysiikan tutkimuskeskuksen CERNin työntekijät Tim Berners-Lee ja Robert Caillau hahmottelivat linkitetyn tiedon järjestelmää, joka mahdollistaisi haluttuun tiedostoon pääsemisen tutkimuskeskuksen miltä tahansa koneelta. Käytössä oli tuolloin monia eri kieliä, kuten PostScript, komentopohjainen asiakirjojen valmistelujärjestelmä TEX ja SGML-kieli. Berners-Lee koki kuitenkin tarpeelliseksi rakentaa paljon yksinkertaisemman järjestelmän, joka mahdollistaisi erilaisten koneiden välisen tiedon kulun. Tuohon aikaan vielä yksinkertaiseen HTTP-protokollaan sovitettiin yksinkertainen HTML-kieli, joka mahdollisti omien selainten ja palvelinten kehittämisen. Myöhemmin sekä HTTP-protokolla että HTML-kieli laajenivat hyvin monimutkaisiksi kokonaisuuksiksi.

Web, myöhemmin Internet, käynnistettiin vuonna 1991 Cernin toimesta julkaisemalla *www-talk*-niminen keskustelulista. Samaan aikaan mukaan liittyi useita muita saman intressin omaavia ihmisiä, jotka pystyttivät omia WWW-sivustojaan sekä kehittelivät omia selaimiaan. Varsinainen läpimurto tapahtui, kun NCSA rohkaisi Marc Andersenia ja Eric Binaa kehittämään Mosaic-selainta, josta myöhemmin tuli varsinainen menestystarina.

HTML-kielen ajatus kuvata WWW-sivuston rakennetta muuttui sivujen laatijoiden halutessa vaikuttaa enemmän myös varsinaiseen ulkoasuun. Selainten valmistajat määrittelivät tähän tarkoitukseen suunniteltuja elementtejä, joita ei kuulunut alkuperäiseen määritykseen. Myöhemmin nämä elementit jouduttiin ottamaan mukaan viralliseen HTML-kielen määritykseen niiden suuren suosion vuoksi. HTML-kieli kärsi standardisoinnin vaikeudesta, joka huomataan tarkastelemalla sen kehityskaarta. Kielen kehitys

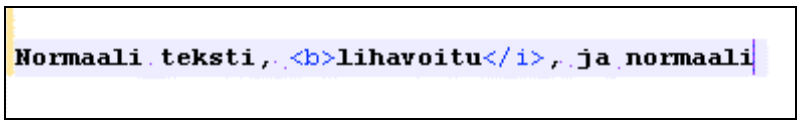
ei ollut lineaarista, versioista 1.0 ja 2.0 hypättiin suoraan versioon 3.2, josta edelleen versioon 4.0. Tämä kehitys johtuu epästandardien ominaisuuksien nopeasta kehittymisestä ja kulminoituu versioon 3.0, joka oli jo luonnosvaiheessaan käytännössä vanhentunut.

Vuodesta 1986 lähtien SGML on ollut kansainvälinen standardi, joka määrittelee, mitä tieto on. Kuvaamisen kannalta on merkittävää, mitkä alueet asiakirjasta on määritelty esimerkiksi otsikoiksi, ja mitkä kappaleiksi. DocBook-, TEI- ja MIL-STD-kielet hyödyntävät SGML-standardia ja niiden avulla pystytään kuvaamaan monimutkaisia ja epämääräisiä asiakirjoja. SGML:n avulla voidaan tekstistä erotella jokainen merkittävää informaatiota sisältävä palanen helposti muotoilua varten. Standardin noudattaminen vaati WWW-sivujen tekijöiltä entistä suurempaa ponnistelua, sekä muutti sivustojen lähdekoodit monimutkaisiksi ja hankaliksi. SGML:n noudattaminen vaati kokonaan uuden kielen opettelemisen, johon vain harvat olivat innokkaita paneutumaan.

SGML-dokumentit vaativat myös yksityiskohtaiset määrittelyt (*Document Type Definition*) jokaista selainta varten, jotta ne pystyisivät tulkitsemaan dokumentin tageja oikein. Tämän lisäksi jokainen SGML:n pohjautuva kieli vaatisi oman DTD:n, joka osoittautui epäkäytännölliseksi.

HTML mahdollisti epätarkan kuvaamisen, jolloin selaimilta vaadittiin ylimääräistä työtä selvittääkseen dokumentin varsinainen sisältö. HTML-kieltä voidaankin pitää SGML-sovelluksena, sillä jokaisen tagin säännöt on toteutettu DTD:ssä SGML-sääntöjen mukaisesti, ja ne ovat valmiiksi upotettu selaimeen.

Mosaic-selaimen julkaisu vaikutti WWW:n räjähdysmäiseen kasvuun, sillä sen avulla kuka tahansa pystyi tekemään omia sivustoja. Mosaic ei noudattanut perinteisten SGML-sovellusten lähestymistapaa tulkatessaan kuvaamista, vaan käsitteli tageja komentoina, joilla rakenteen muoto toteutettiin. Kuvaamisen syntaksi oli myös Mosaic-selaimessa hyvin erikoinen. Käytettäessä esimerkiksi lihavointi-tagia (<b>) aloitusmerkkinä, lopetti mikä tahansa korostukseen liittyvä tagi korostuksen (Kuva 2).



Kuva 2 Mosaic-selaimen esimerkki (Boumphrey, Greer, Raggett, Raggett, Schnitzenbaumer, Wugofski 2003, 26)

Mosaic-selaimen suvaitsevaisuus oli suurimpana syynä siihen, että ihmisten oli helppoa laatia omia sivustojaan välittämättä oikeasta syntaksista. Kopioimalla toisten sivujen lähdekoodit ja muokkaamalla niitä omien tarpeiden mukaan, saatiin omat sivut verkkoon hetkessä. Selainten kehitys lisäsi HTML:n käyttöä ja selainvalmistajat esittelivät omia tagejaan, jolloin huonon HTML:n kirjoittamisen osuus kasvoi entisestään.

Uusien laitteiden kehittyneet resurssit muodostivat kuitenkin pullonkaulan näiden huonojen HTML sivujen tulostamisessa. Suorituskykyisemmät, mutta tarkemmin tehon kulutusta seuraavat laitteet, eivät suostuneet näyttämään hankalasti luettavien HTML-sivujen sisältöä niiden runsaan tehonkulutuksen vuoksi. Sivustojen määrien kasvaessa hakukoneet kuluttivat entistä enemmän aikaa etsimällä yksittäisiä sanoja sivustoviidakosta. Nämä ongelmat saivat ihmiset kehittämään järjestystä Webiin suunnittelemalla kielen, jonka avulla sivustoja voitaisiin kuvata selittävillä tageilla rakenteellisten tagien sijaan.

Ratkaisuna ongelmaan oli uuden kielen kirjoittaminen. Kielen tuli sisältää suurin osa SGML:n kuvaamisominaisuuksista ilman sen mukana tuomaa monimutkaisuutta. Lisäksi kieltä tulisi voida käyttää ilman DTD-sääntöjä. Tim Brayn ja Jon Bosakin johdolla perustettu W3C-komitea löysikin ratkaisun, *eXtensible Markup Language* (XML).

XHTML ei ole varsinaisesti uusi kieli, vaan siihen on yhdistetty XML:n ja HTML:n parhaat puolet. Se käyttää HTML-kielen sanastoa yhdistettynä XML-kielen kieliopilla. Ensimmäinen julkaistu XHTML-standardi, XHTML 1.0, julkaistiin W3C-komitean toimesta vuonna 2000. Nykyään käytössä oleva XHTML 1.1 -standardi julkaistiin vain vuotta myöhemmin. Uutta XHTML 2.0 -standardia kehitellään jatkuvasti, ja sen on sanottu eroavan edeltäjistään merkittävästi. Se ei tule olemaan yhteensopiva aiempien XHTML-version kanssa, eikä se jatka HTML-kielen seuraamista. Suurimpana syynä

uuden standardin kehittämiselle on päätelaitteiden jatkuva kehitys ja sen mukana tuomat vaatimukset.(Boumphrey ym. 2003, 7 - 40.)

XHTML suunniteltiin kuvaamaan sivuston rakennetta, joten muotoilu ja ulkoasu voidaan toteuttaa erillisellä kaskadisella CSS-tyylitiedostolla (Cascading Style Sheets), jonka määrittelystä vastaa myös W3C-konsortio.

5.2 CSS

CSS, eli tyyliohjeet, määrittelevät sivuston tai dokumentin ulkoasun. Alkujaan jo SGML-kielen kanssa käytettyjä tyyliohjeita tarkennettiin HTML:n yleistymisen vuoksi. WWW-sivustojen ulkoasun määrittelyä varten W3C esitteli yhdeksän tyyliohjetta, joista kaksi (CHSS ja SSP) muodostivat pohjan CSS:n synnylle. Tyyliohjeiden järjestelmän määritteleminen eteni kuitenkin hyvin hitaasti verrattuna selainvalmistajien HTML:n lisäämiin ulkoasun säätelyyn piirteisiin.

Virallisesti CSS määrittely käynnistyi W3C:n toimesta vuonna 1995 ja jo vuotta myöhemmin julkaistiin ensimmäinen CSS 1 -suositus. Määrittelyn toteutus oli kuitenkin hyvin hidasta, vaikka vuonna 1996 julkaistussa IE:n versiossa 3 olikin jonkinlainen CSS-tuki. CSS:n käytön lisääntyminen ja mutkistuminen lisäsi myös tulevien selainten vajavaista tukea tyylitiedostoille, joten useimmat WWW-sivujen kehittäjät jatkoivat ulkoasun muokkaamista HTML:n piirteiden avulla. CSS 1 -määrittelyä korjattiin vuonna 1999.

CSS 2 -suositus julkaistiin vuonna 1998, jolloin edellisen version toteutukset olivat vielä kehitysasteella. Selainten valmistajat eivät keskittyneet hankalasti implementoitavan tyyliohjeen lisäämistä täysin selaimiin, vaan keskittyivät luomaan siltä väliltä olevia ratkaisuja. Selaimista vasta IE 6 ja Netscape 6 (tai 7) omasivat laajemman tuen CCS 2 -ohjeelle. Kaikkia osioita ei näihinkään selaimiin oltu toteutettu, mutta suurimmat virheet oli pystytty korjaamaan.

Huolimatta selainten heikosta tuesta CSS:ä kohtaan, yleistyi sen käyttö kuitenkin siten, että vuonna 2001 tyylitiedostoja käyttämättömät sivustot olivat jo harvinaisempia. Tähän oli syynä CSS:n monimuotoisuus, joka mahdollisti toteuttamaan asioita, joita

HTML:n avulla toteutettuihin ulkoasupiirteisiin ei pystytty tekemään. CSS:n kuvaus on laadittu tiiviiksi ja vaikealukuiseksi, joka on johtanut siihen, etteivät selainten valmistajat, eivätkä sivujen tekijätkään, voineet olla varmoja onko määrittely vielä voimassa, vai onko sitä muutettu jo vuosia sitten.

Huolimatta CSS 2:n puutteista, siihen ei koskaan laadittu korjausta, vaan vuonna 1998 luonnosteltiin realistisempi versio CSS 2.1. Sen tarkoituksena oli kuvata ne osajoukot, jotka CSS 2:ssa on toteutettu kaikilla parhaimmilla yleiseen käyttöön tarkoitetuilla selaimilla. Samoin siitä poistettiin niitä toteuttamatta jääneitä ominaisuuksia, joita vain harvat selaimet tukivat. Virallinen määrittely on edelleen CSS 2.0, mutta jopa W3C-konsortio on hylännyt sen ja tällä hetkellä CSS versiota 2.1 pidetään käytännön standardina.

CSS 3 -versio on ollut jo pitkään kehitteillä, ja sen julkistamista ovat mutkistaneet selainvalmistajien toteuttamat omat laajennukset. Uusi versio koostuu moduuleista, jotka mahdollistavat tiettyjen ongelmakohtien kehittämisen riippumatta muiden moduuleiden kehitysasteesta. Selaimet tukevat uutta versiota vielä vajavaisesti, joten toistaiseksi pitäytyminen CSS 2.1 -määrittelyssä takaa paremman toimivuuden useimmilla eri selaimilla. (Korpela 2008, s 53 - 60.)

CSS:n käyttöä WWW-sivujen kehittämisessä puoltaa mahdollisuus muokata sivujen tyyliä ainoastaan muuttamalla tyylitiedoston sisältöä. Kuitenkin sivustoja kehittäessä on otettava huomioon käytössä olevien selainten suuri vaihtelevuus tyyliohjeiden tukemiseen ja usein eri selaimille on laadittava omat, selaimen tuen mukaiset tyylitiedostot.

6 Arkkitehtuuri

Proto Toolin arkkitehtuuri kuvastaa asiakas–palvelin-pohjaista ratkaisumallia, jossa käyttäjät lähettävät sovelluksen käyttöliittymän kautta erilaisia ohjelmapyyntöjä tietokantapalvelimelle, joka vastaa lähettämällä ohjelmapyyntöjen tulokset takaisin käyttäjän päätelaitteelle. Tässä luvussa tarkastellaan arkkitehtuuriin kuuluvien ohjelmistojen kokoonpanoja, sekä toimintaa normaalissa käyttäjätilanteessa.

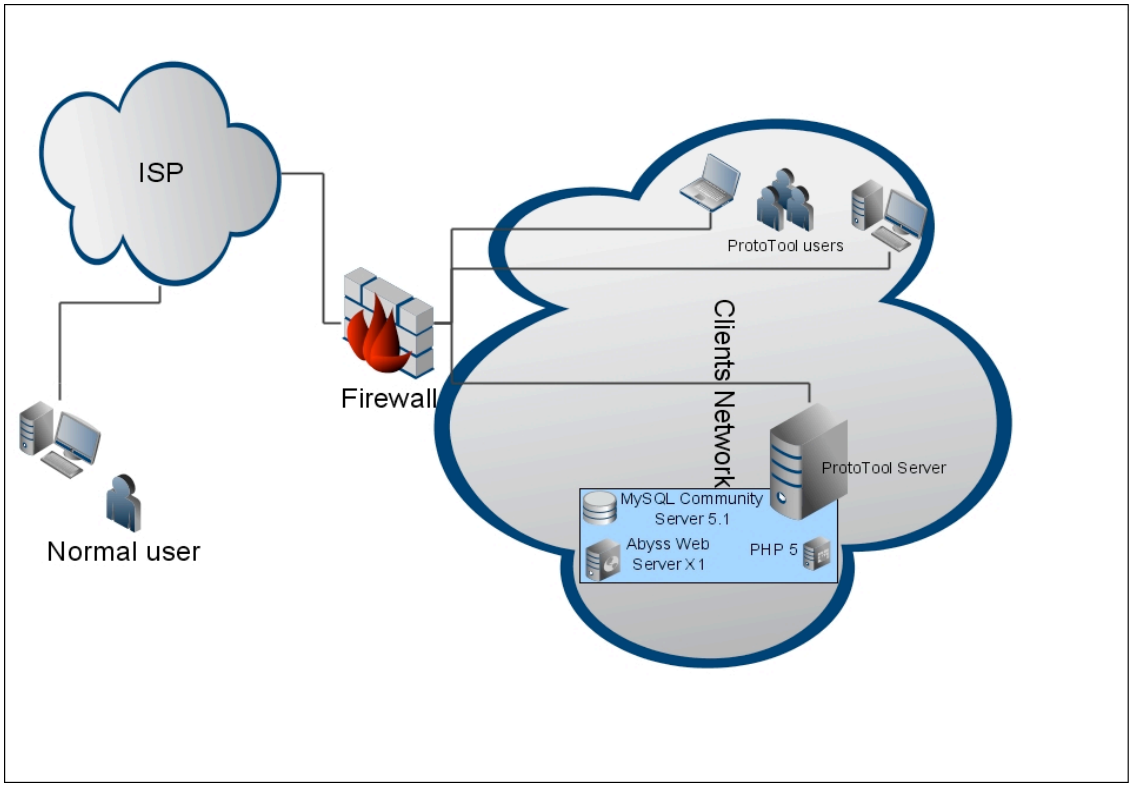
Asiakkaan palomuuri estää sisäverkon ulkopuoliset yhteydet, mikä parantaa samalla sovelluksen tietoturvaa, kuten kuvassa 3 oleva arkkitehtuuri osoittaa. Proto Toolin palvelin koostuu Windows XP -käyttöjärjestelmällä varustetusta tietokoneesta, jonka muistin ja kiintolevyn koon määrää kasvatettiin vastaamaan paremmin palvelimena toimivan laitteen vaatimuksia. Alkuperäisessä kokoonpanossaan tietokone sisälsi sisäistä muistia 1 Gb:n verran, sekä sen kiintolevy oli varsin pieni, vain 40 Gt:a, mutta päivityksen jälkeen molemmat kaksinkertaistuivat. Palvelimen käyttöjärjestelmänä Windows XP ei ole optimaalisin, sillä sen resurssien kulutus on suurempi, kuin esimerkiksi Unix-pohjaisen Gentoo- Linuxin, joka optimoituu automaattisesti laitteiston mukaisesti. Asiakas ei alkuperäisen määrittelyn mukaisesti budjetoinut sovelluksen laitteistoihin ja ohjelmistoihin, joten kompromissina käytettiin niitä, joita oli saatavilla.

Palvelimelle asennettiin kolme sovelluksen kannalta merkittävää ohjelmistoa; MySQL-tietokanta-, Abyss Web Server -, sekä PHP 5 -palvelin. WWW-sovellusten tuottamiseen on tarjolla useita valmiita ohjelmistoja, joissa sekä WWW-palvelin, että sovelluskehitysympäristö on integroitu samaan asennuspakettiin. Tällä voidaan välttää ulkoisten sovellusten käynnistäminen itse sovellusta käynnistettäessä. Integroiduissa palvelupaketeista huolimatta Proto Toolin arkkitehtuuriin katsottiin sopivan paremmin erilliset palvelinohjelmistot eri tarpeisiin. Ratkaisu mahdollistaa yksittäisen ohjelmiston muuttamisen tai päivittämisen ilman, että sovelluksen toiminta heikkenee.

Palvelinohjelmistot vastaavat käyttäjien palvelupyyntöihin ennalta määritetyissä porteissa. Porttien numerot ovat hyvin pitkälle standardisoituja, mutta tarvittaessa niitä on mahdollisuus määritellä myös käsin. Useimmat selaimet on määritelty ottamaan yhteyttä porttiin 80, joka on WWW-palvelimen oletusarvoinen portti. Tämä oletusarvo säilytettiin myös konfiguroidessa Abyss Web Serverin asetuksia. Palvelin tukee useita eri

ohjelmointikieliä, mutta niiden tiedostopäätteet, sekä tulkki on määritettävä palvelinta käyttöönotettaessa.

Ennen WWW-palvelimen asennusta, asennettiin palvelinkoneeseen ohjelmointikielen PHP 5 -tulkki, jotta kieltä voidaan käyttää käyttöliittymän toteutuksessa. Asennus suoritettiin oletusarvoisesti, ainoastaan *php.ini*-tiedostoon tehtiin tarvittavat muutokset muun muassa puskurimuistin koon lisäämiseksi. Tulkin asennuksen jälkeen konfiguroitiin Abyss Web Server tukemaan .php-päätteisiä tiedostoja määrittelemällä Scripting parametres -valikkoon käytettävän tulkin sijainti ja tiedostopääte, sekä luomalla uusi index-tiedosto *index.php*. Palvelimen uudelleenkäynnistyksen jälkeen se osaa lukea ja käsitellä myös PHP-tiedostoja. Käyttäjälle palvelimen PHP-tuki ei varsinaisesti näy, vaan hänen suorittaman HTML-muotoisen palvelupyynnön käsittelevä WWW-palvelin tunnistaa pyynnön vastaanottaessaan ohjelmakoodin tiedostomuodon, jonka jälkeen pyynnön mukainen PHP-ohjelmakoodi suoritetaan ja käyttäjän selaimeen tulostetaan koodin mukainen HTML-tuloste.



Kuva 3. Proto Tool-sovelluksen arkkitehtuuri. Palvelin on yhdistetty asiakkaan omaan sisäiseen verkkoon, joka mahdollistaa kaikkien sisäverkossa olevien käyttäjien pääsyn palvelimelle. Ulkopuolisen yhteyden palvelimeen estää asiakkaan oma palomuuri.

7 Tietokannan suunnittelu ja toteutus

Sovelluksen tietokantaohjelmistoksi valikoitui MySQL Community Server versio 5.1, jonka ominaisuuksia ja kehityskaarta esittelin opinnäytetyöni tietokantoja käsittelevässä luvussa. Ohjelmiston valintaan vaikuttivat sen laaja tuki eri käyttöjärjestelmille, kattavat dokumentaatiot sekä pienet kustannukset. Ohjelmisto on saatavissa ilmaiseksi MySQL:n kotisivuilta, josta tarvittaessa on saatavilla myös apua virhetilanteisiin. Huolimatta siitä, että ohjelmisto on ilmainen, kykenee se toimimaan luotettavasti myös suurempia tietokantoja käsitellessä. Tästä esimerkkinä ohjelmiston kehittämän MySQL Ab:n tietokanta, joka pitää sisällään yli 10 000 taulua, joista yli 500:aan on tallennettu yli 7 miljoona riviä (Heinisuo 2003, 38).

Tietokannan suunnittelu aloitettiin pohtimalla, mitä tietoja tietokantaan tullaan tallentamaan. Sovelluksen tehtävänä oli tarjota informaatiota laitteiden tiedoista, niiden käyttäjistä sekä muista tiimin tarvitsemista välineistä. Tarvittavat välineet ja käyttäjät muodostivat ns. päätaulut, joissa toiminnan kannalta oleellisimmat tiedot tullaan säilyttämään. Päätaulujen kartoittamisen jälkeen määriteltiin niiden tietoja tukevat taulut, jotka sisältäisivät lisätietoa kyseisen taulun sarakkeesta. Käyttäjien kohdalla näitä olivat muun muassa käyttäjän sijainti sekä sähköpostiosoite.

Tietokantataulujen määrittelyjen jälkeen suunniteltiin yksittäisten taulujen sarakkeiden nimet, tietotyypit, sekä pää- ja viittaus-avaimet. Jokaisella laitteella on oma yksilöllinen tunniste-koodinsa, jonka avulla kyseinen laite voidaan yksilöidä. Tunniste-koodi soveltui siis taulun pääavaimeksi, sillä toista samanlaista tunniste-koodia ei kahdella laitteella voinut olla. Tunniste-koodi koostuu ainoastaan numeroista ja sen pituus on määritelty 15 merkkiin. Sarakkeen tietotyypiksi valittiin Integer, eli kokonaisluku, ja kooksi 20. Tämä sen vuoksi, että myöhemmin uusien tuotteiden kohdalla tunniste-koodin pituutta saatetaan muuttaa.

Poikkeuksen tietotyypin valintaan aiheuttivat ne sarakkeet, joiden avulla voitiin tarvittaessa hakea tarkennettuja tietoja, kuten käyttäjän sijainti ja sähköpostiosoite. Kahden saman arvon tallentamista tietokantaan pyritään hyvässä tietokantasuunnittelussa välttämään, joten tämä ratkaistiin käyttämällä yksilöivää tunnistetta (ID), jonka avulla tar-

kennettuihin tietoihin päästään käsiksi. Käyttäjien sijainti määrittyi asiakkaan tiimin sijainnin perusteella, joten käyttäjä-taulun aputauluksi luotiin sijainti-taulu.
Sijainti-taulun pääavaimena toimi yksilöivä tunniste (ID), joka tallennettiin käyttäjän sijainnin perusteella myös käyttäjä-tauluun. Näin voidaan etsiä lisätietoa käyttäjästä ja hänen sijainnistaan tunnisteen avulla. Tämä nopeuttaa tietokantahaun suorittamista. Kaikkien aputaulujen pääavaimet luotiin samalla periaatteella, joka mahdollisti samantyyppisten funktioiden käytön käyttöliittymää toteutettaessa.

7.1 Tietokannan normalisointi

Esisuunnittelumallin valmistuttua pyrittiin suunniteltu tietokanta normalisoimaan ylläpidon ja tehokkuuden parantamiseksi. Normalisointiin käytetään viisi-portaista tarkastuslistaa, jossa ensimmäinen taso kuvastaa alinta mahdollista normalisoinnin tasoa ja viides ylintä. Yleisesti tietokannat normalisoidaan toiselle tai kolmannelle tasolle ja vain harvoin niistä ylemmille. Normalisoidun tietokannan avulla pystytään suorittamaan monimutkaisiakin tietokantakyselyitä kohtalaisen yksinkertaisin kyselylausekkein ja sen avulla vältytään tietojen päällekkäisyyksiltä, sekä voidaan varmistaa tiedon ajantasaisuus. Yksi tietokantasuunnittelun huomioon otettavista seikoista on myös tietokannan skaalautuvuus, eli mahdollisuus mukautua tulevaisuuden kasvun mukaisesti.

Normalisoinnissa tietokannan jokaista taulua ajatellaan yhtenä kokonaisuutena, jonka ilmentymiä taulun rivit ovat. Ensimmäisen normalisointisäännön mukaisesti jokaisella taululla on oltava pääavain, joka koostuu mahdollisimman pienestä määrästä kenttiä. Atomisuudella tarkoitetaan kentän koostumista ainoastaan yhdestä arvosta. Proto Toolin tietokannan taulut käytiin läpi tarkastaen, että jokaiselle taululle oli määritelty pääavain ja että tallennettava tieto oli atominen. Toimipisteiden osoite-sarake oli alkujaan määritelty kattamaan koko osoite, mutta noudattamalla ensimmäisen normaalisäännön ohjeita, jaettiin se kolmeen sarakkeeseen: osoite, paikkakunta ja maa. Muutoksen avulla varmistettiin tietojen päivittämisen helpottaminen. Osoitetiedon muuttuessa vain yhden sarakkeen arvoa joudutaan muuttamaan.

Toisen normalisointisäännön mukaisesti tietokannan on täytettävä ensimmäisen säännön vaatimukset, sekä jokaisen pääavaimettoman sarakkeen on oltava täysin riippuvainen taulun pääavaimesta. Sarakkeessa useasti toistuvat arvot ovat pääavaimesta riippu-

mattomia, kuten Proto Toolin laitteen tila-sarakkeessa (statustype). Useasti toistuvat arvot on toisen normalisointisäännön mukaisesti tallennettava omaan tauluunsa, jolloin päällekkäisyyksiltä vältytään. Käytössä olevien tila-arvojen muuttuessa tarvitsee tietokannan pääkäyttäjän päivittää ainoastaan yhden taulun arvot, jolloin ne automaattisesti päivittyvät kaikkiin tietokannan ilmentymiin.

Kolmannen normalisointisäännön mukaisesti tietokannan on täytettävä edellisten tasojen vaatimukset, eikä siinä saa olla siirtyviä riippuvuuksia. Siirtyvällä riippuvuudella tarkoitetaan tietokannan, ei pää- eikä viittausavainten, sarakkeita, joiden arvo määräytyy toisen, 'avaimettoman' arvon perusteella. Proto Toolin tietokantasuunnitelmasta ei löytynyt siirtyviä riippuvuuksia, joten tietokannan todettiin täyttävän normalisoinnin kolmannen tason vaatimukset.

7.2 Tietokannan toteutus

Tietokanta toteutettiin käsin, eli valmiita graafisia tietokantasuunnitteluun tarkoitettuja ohjelmistoja ei käytetty, vaan taulut, sarakkeet ja relaatiot luotiin Notepad++:lla, joka on tunnettu, monia eri ohjelmointikielten syntakseja tunnistava tekstieditori. Verkosta on saatavissa graafisella käyttöliittymällä varustettuja, tietokannan suunnitteluun ja toteutukseen tarkoitettuja sovelluksia, mutta manuaalinen toteuttaminen koettiin tilanteessa tehokkaimmaksi.

Proto Toolin relaatiotietokantaan luotiin yhteensä 16 taulua ja 24 niitä yhdistävää riippuvuutta. Tietokannan tauluissa käytetään ainoastaan InnoDB-tietokantamoottoria, sillä sen avulla voidaan varmistaa datan viite-eheyden säilyminen, mikä on hallinnan kannalta välttämätöntä.

Kaikista taulujen välisistä riippuvuuksista laadittiin omat testitapaukset, joita käytettiin tietokannan toiminnallisuuden testaamiseen ennen käyttöliittymän toteutusta. Testitapauksien tulokset raportoitiin ja mahdolliset virheet korjattiin. Kun tietokannan kaikki relaatiot oli testattu ja ne tuottivat testitapauksen mukaisen lopputuloksen, siirryttiin käyttöliittymän toteutukseen. (ensimmäisen testikierroksen tulokset ovat liitteessä 2).

8 Käyttöliittymä

Käyttöliittymän suunnittelussa on huomioitava varsinainen käyttötarkoitus, eli mitä tehtäviä sovelluksen käyttöliittymällä on tarkoitus suorittaa. Proto Tool:n käyttöliittymän oli ensisijaisesti palveltava käyttäjäänsä tarjoamalla hänen haluamansa tiedot helposti ja vaivattomasti päätelaitteelle.

Sivuston asettelulla ja värimaailmalla pyrittiin yhtenäiseen ja hillittyyn lopputulokseen, sekä helppokäyttöisyyteen käyttämällä väreinä asiakkaan tunnusvärien eri sävyjä. Lisäksi sivustolla käytettiin käyttäjäryhmän tuntemaa termistöä, jonka käytöllä minimoidaan näin uuden käyttöliittymän oppimiseen kuluvaa aikaa (Kuva 4).

Kuva 4. Proto Tool käyttöliittymä

Asiakkaan käyttäjäryhmä ei ole pakotettu käyttämään ainoastaan tiettyä selainta, vaan heidän käytössään oli ainakin Safari, Mozilla Firefox sekä Internet Explorer (IE) eri versioineen. Näistä varsinkin IE vaati täysin oman CSS-tyylitiedostonsa yhtenäisen sivustorakenteen saavuttamiseksi. Sivustolle vaadittiin selaimen tunnistava koodipätkä, jotta kyettiin erottamaan IE-selaimet muista ja jonka avulla määriteltiin mitä tyylitiedostoa selaimen tulisi käyttää. Kuvassa 5 on IE-hack, jonka avulla voidaan määritellä IE-selainten käyttämä tyylitiedosto.

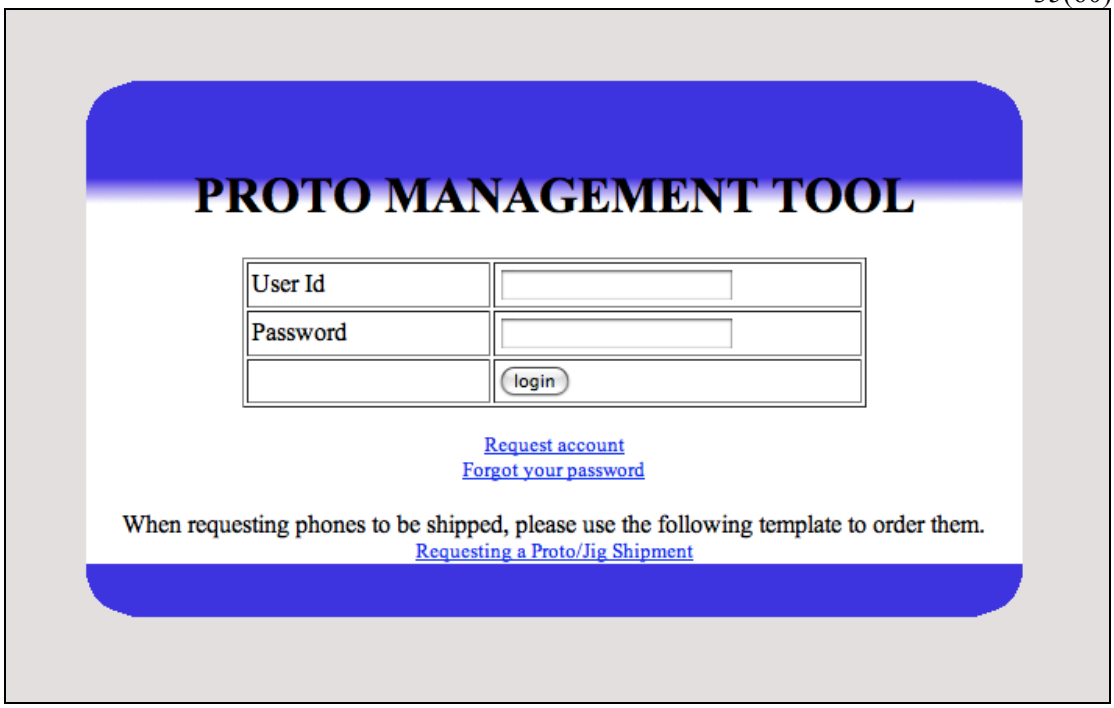
```
<head>
<link rel="stylesheet" type="text/css" href="../css/flyout.css"/>
<!--[if IE]>
        <link rel="stylesheet" href="../css/flyout_ie.css" type="text/css" />
 <![endif]-->
<title>Proto Management Tool</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
```

Kuva 5. IE-hack (Muokattu Loweryn 2005, s. 57 mallista)

8.1 Kirjautuminen

Proto Tool:n luotiin käyttäjätunnukset ja salasanat vanhan järjestelmän tietojen perusteella. Näin voitiin varmistaa vanhan järjestelmän tietokannan siirto uuteen järjestelmään ilman tietojen häviämistä. Uusia käyttäjiä varten luotiin *requestaccount.php*-tiedoston avulla lomake, jonka kautta he voivat anoa käyttäjäoikeuksia sovelluksen pääkäyttäjiltä.

Ulkoasultaan kirjautumisikkunan tuli olla selkeä, sekä helppokäyttöinen ja tarjota käyttäjälle mahdollisuus tunnusten hakemisen lisäksi myös pyytää automaattista salasanan resetointia. Kirjautumisikkuna määrittelee koko sovelluksen ulkoasun, lukuun ottamatta muutamia asiakkaan pyytämiä raportointisivuja (Kuva 6).



Kuva 6. Kirjautumisikkuna

Sisäänkirjautuminen toteutettiin sovellukseen *login.php*-tiedoston avulla, johon luotiin HTML-pohjainen lomake käyttäjätunnuksen ja salasanan syöttämistä varten. Käyttäjän painaessa Login-painiketta, tallentaa lomake syötetyt tiedot (käyttäjätunnus ja salasana) muuttujiin $userid ja $password. Ennen käsittelyä *login.php* tarkistaa, ettei kumpikaan muuttuja ole jäänyt tyhjäksi.

Tiedosto hallinnoi tietokantayhteyttä käyttämällä *config.php*-, *opendb.php*- ja *closedb.php*-tiedostoja ja suorittaa tietokantahaun käyttäjätunnuksen ja salasanan perusteella. Mikäli tietokannasta löytyy vastaava tunnus ja salasanapari, luodaan käyttäjälle cookie eli keksi, jonka avulla käyttäjä tunnistetaan hänen liikkuessaan sivustolla, sekä ladataan sovelluksen etusivu. Tietokantavasteen puuttuessa ilmoitetaan käyttäjälle virheestä ja ladataan kirjautumis-ikkuna uudelleen. Kuvassa 7 on koodi, jolla luodaan käyttäjälle cookie.

```
if (mysql_num_rows($result) == 1)
{
    // the user id and password match,
    // set the session
    $_SESSION['db_is_logged_in'] = true;
    // create ptmcookie
      setcookie("ptmcookie", $userid,  time()+3600);
    // after login we move to the index page
    header('Location: index.php');
    exit;
}
else
{
    $errormessage = 'Sorry, wrong user id / password';
}
```

Kuva 7. Cookien luonti

Käyttäjän kirjautuessa ulos sovelluksesta varmistaa *logout.php*-tiedosto käyttäjän cookien voimassaolon, jonka jälkeen cookie tuhotaan, nollataan ja kirjautumisikkuna ladataan selaimeen. (Kuva 8). Sovelluksesta uloskirjautuminen on toteutettu käyttöliittymään käyttämällä html-linkkiä, joka ohjataan *logout.php*-tiedostoon.

```
session_start();
// if the user is logged in, unset the session
if (isset($_SESSION['db_is_logged_in'])) {
   unset($_SESSION['db_is_logged_in']);
}
// now that the user is logged out,
// go to login page
// set the expiration date to one hour ago
setcookie("ptmcookie", "", time()-3600);
header('Location: login.php');
?>
```

Kuva 8. Cookien nollaus

8.2 Autentikointi

Käyttäjän autentikoinnilla varmistetaan, että hän on saapunut sivustolle kirjautumisikkunan kautta, ja että istunnolla on voimassa oleva cookie. Proto Toolissa tämä suoritetaan *userauthentication.php*-tiedostolla ainoastaan palvelimella, käyttäjälle tiedosto on

”näkymätön”. Kirjautumisikkuna lähettää parametrina cookien arvon, jonka perusteella voidaan määritellä, onko käyttäjän istunto eli sessio edelleen voimassa. Cookien avulla skripti hakee muuttujiinsa muut käyttäjän tiedot, kuten esim. roolin, sekä henkilön nimen, joita käytetään mm. tulostettaessa käyttäjän tiedot sovelluksen info-kenttään. Cookien vanhenemisaika asetetaan käyttäjän roolista riippuen. Normaalikäyttäjän cookie on voimassa yhden tunnin, kun taas pääkäyttäjän cookie on voimassa kolme tuntia (Kuva 9).

```
if (isset( $_COOKIE['ptmcookie']))
{
if (!isset($_SESSION['db_is_logged_in']) || $_SESSION['db_is_logged_in'] !== true)
{
        // not logged in, move to login page
        header('Location: login.php');
        exit;
}
else
{
   $userid = $_COOKIE["ptmcookie"];
   setcookie("ptmcookie", $userid,  time()+3600);
   include 'lib/config.php';
   include 'lib/opendb.php';
   // get user info from database
   $sql = "SELECT
   firstname,
   lastname,
   company,
   role
   FROM usertype
   WHERE
   userid = '$userid'
   AND
   exist=0";
   $result = mysql_query($sql) or die('Query failed. ' . mysql_error());

        $firstname = mysql_result($result,0,"firstname");
        $lastname = mysql_result($result,0,"lastname");
        $company = mysql_result($result,0,"company");
        $role = mysql_result($result,0,"role");
        if ($role == 'administrator')
        {
        setcookie("ptmcookie", $userid,  time()+10800); // Time that cookie is valid //
        }
        else
        {
        setcookie("ptmcookie", $userid,  time()+3600);
        }
```

Kuva 9. Käyttäjän autentikointi

Tiedosto *userauthentication.php* lisätään jokaiselle sovelluksen sivulle varmistamaan, etteivät kirjautumattomat käyttäjät pääse näkemään sisältöä. PHP:ssa voidaan kätevästi sulauttaa ulkoisia tiedostoja toiminnallisuuteen käyttämällä include-funktiota. Se eroaa toisesta PHP:n tunnetusta require-funktiosta virheiden käsittelyssä. Kohdatessaan virheen, include-funktio luo virheilmoituksen keskeyttämättä kuitenkaan skriptin ajamista, kun taas require-funktio luo vakavan virheilmoituksen ja keskeyttää skriptin ajamisen.

8.3 Käyttäjän salasanan vaihtaminen

Vanhan sovelluksen käyttäjille luotiin automaattisesti käyttäjätunnukset ja salasanat, kuten opinnäytetyöni luvussa 8.1 mainittiin. WWW-sovelluksen on kuitenkin tarjottava käyttäjälleen mahdollisuus muuttaa omia käyttäjätietojaan tarvittaessa. Proto Tool:ssa käyttäjän on mahdollisuus muuttaa kirjautumissalasanaansa sovelluksen pääikkunasta löytyvän *Change password* -lomakkeen kautta.

Lomake lähettää syötetyn, käytössä olevan salasanan lisäksi uuden salasanan, sekä sen vahvistuksen parametreina tiedostolle *passwordchange.php* käyttäen PHP:n POST-funktiota. Funktiota käytetään lomakkeen arvojen keräämiseen, kun niitä ei haluta näyttää selaimen osoiterivillä. Arvojen kokoa ei ole rajoitettu, mutta käytettäessä POST-funktiota on otettava huomioon, että maksimioletusarvo on 8Mb. POST eroaa toisesta vastaavasta funktiosta GET juuri näkymättömyytensä vuoksi, ja on tällöin suositeltavampi, joskaan ei turvallisin vaihtoehto yksityiskohtaisten tietojen siirtämiselle.

Tiedosto vertailee uutta ja vahvistettua salasanaa keskenään poistaakseen käyttäjän mahdolliset kirjoitusvirheet. Salasanojen ollessa identtiset tarkistaa tiedosto, että kirjautuneen käyttäjän käyttäjätunnus, sekä salasanapari löytyvät tietokannasta. Tämän jälkeen suoritetaan tietokantaan päivitys, päivittämällä käyttäjän vanha salasana uudella ja ilmoitetaan onnistuneesta päivityksestä käyttäjälle. Vaihdettavan salasanan pituus on rajattu minimissään viiteen merkkiin, joka tarkastetaan käyttämällä PHP:n strlen-funktiota. Strlen-funktio palauttaa muuttujan merkkien määrän, eli käytännössä laskee montako merkkiä pitkä muuttuja on. Salasanan vaihdosta luodaan samalla tapahtumalokiin merkintä.

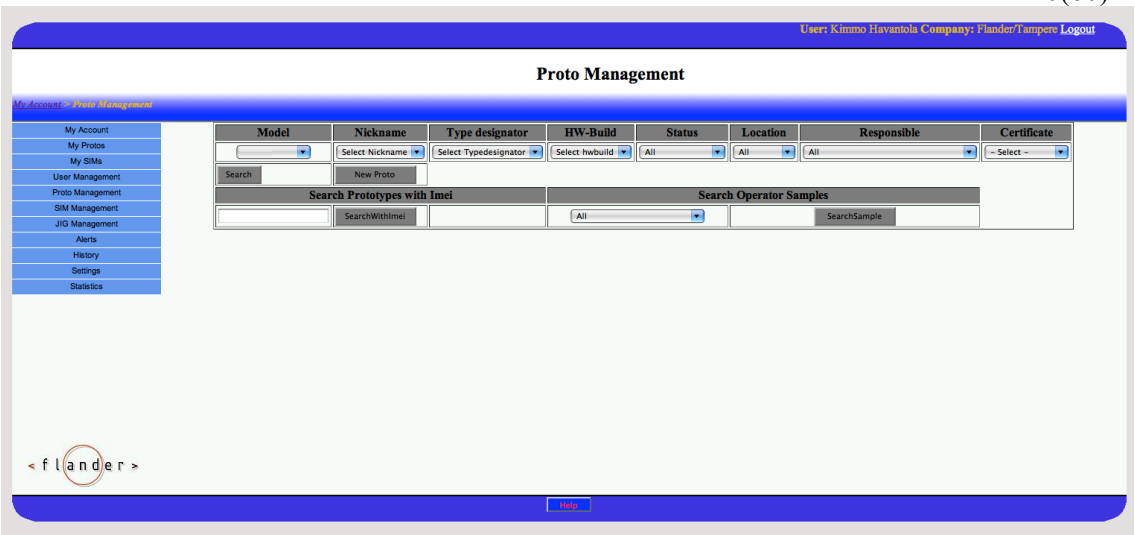
8.4 Tiedon hakeminen tietokannasta

Edellisessä alaluvussa mainittiin tietokantayhteyttä avattaessa kolme tiedostoa, joita käytettiin yhteyden luomiseen: config.php, opendb.php ja closedb.php. Näistä config.php sisältää tietokannan ns. perustiedot, eli käyttäjätunnuksen, salasanan, tietokannan nimen, sekä yhdistettävän palvelimen, eli hostin. Opendb.php-tiedosto käyttää config.php-tiedostoon määriteltyjä muuttujien arvoja hyväkseen pyrkiessään luomaan yhteyden tietokantaan. Yhteyden luonti tapahtuu PHP:n oman *mysql_connect*-funktion avulla, joka saa parametreikseen tietokannan palvelimen, käyttäjän ja salasanan. Yhteyden luonnin onnistuessa valitaan käytettävä tietokanta funktiolla *mysql_select_db*, jonka parametrina on tietokannan nimi. Tietokantayhteys suljetaan funktiolla *mysql_close*.

Proto Tool:n päämääränä on tarjota tietoa erityisesti laitteiden liikkeistä tiimien sisällä, sekä eri laitteistoversioiden määristä. Tietoa on voitava hakea käyttäjän määrittelemin hakuehdoin ja se on pystyttävä tallentamaan tarvittaessa Excel-muotoon. Seuraavissa luvuissa käsitellään eri tietokantatapahtumia, sekä niiden toteutettuja ratkaisuja.

8.4.1 Hae laitteita

Hakulomakkeella on mahdollista suorittaa kolmea eri tarkoituksen omaavaa hakutoimintoa. Kaikkien kolmen hakupainikkeen sijaitessa saman lomakkeen sisällä, on niiden tunnisteet pystyttävä erottamaan toisistaan pystyäksemme määrittelemään, minkä haun käyttäjä on halunnut suorittaa. Käsittelen opinnäytetyössäni ainoastaan pudotusvalikoiden avulla suoritettavaa hakua, muita hakuvaihtoja ovat laitetietojen hakeminen tunniste-koodin tai sen osan perusteella, sekä asiakkaille lähetettyjen laitteiden haku. Tietokantakyselyt on toteutettu kaikissa hakuvaihtoehdoissa samankaltaiseksi, ainoastaan haun yksilöivä kenttä vaihtuu (Kuva 10).



Kuva 10. Tietojen hakeminen -perusnäkymä

Laitteista tarvitaan tietoa mm. tuotenimikkeen, mallinumeron, laitetyypin sekä laitteistoversion perusteella. Hakuehtojen toiminnallisuus on toteutettu *searchprotos.php*-tiedoston avulla, jossa käytön helpottamiseksi on luotu mainituille hakuehdoille oma pudotusvalikkonsa. Eri laitetyyppien laajan määrän vuoksi asiakas halusi pudotusvalikkojen olevan ns. päivittyviä, eli kun yksi yllämainituista valintaehdoista määritellään, päivittyvät seuraavista valikoista löytyvät vaihtoehdot valinnan mukaisesti.

Toteutuksessa ei voitu käyttää ainoastaan PHP:ta, sillä käyttäjän selaimessaan tekemät toiminnot eivät suoriudu palvelimella, joten pudotusvalikot toteutettiin JavaScriptillä, joka mahdollistaa käyttäjän selaimessa tapahtuvien toiminnallisuuksien toteutuksen. JavaScriptin avulla luotiin lomaketta päivittävä funktio, jonka sisältöä ohjataan PHP:n avulla. Funktion tarkoituksena on ladata sivu käyttäjälle uudestaan kunkin valinnan jälkeen ja päivittää jäljellä olevat valikot sen mukaisesti (Kuva 11).

```
function reload4(form)
{
var val=form.model.options[form.model.options.selectedIndex].value;
var val2=form.nickname.options[form.nickname.options.selectedIndex].value;
var val3=form.typedesignator.options[form.typedesignator.selectedIndex].value;
self.location='searchprotos.php?model=' + val + '&nickname=' + val2 + '&typedesignator=' + val3;
}
```

Kuva 11. JavaScript-funktiot

Funktio tallentaa kolmeen muuttujaansa käyttäjän tallentamat arvot, jonka jälkeen ne lähetetään uudelleen *searchprotos.php*-tiedostolle käyttämällä selaimen osoiteriviä. Lähetettyjen muuttujien perusteella *searchprotodropdown.php*-tiedosto suorittaa tietokantahaun.

8.4.2 Tulosta tiedot

Laitteiden haun jälkeen tietokannasta saadut tulokset tulostetaan käyttäjän näyttöpäätteelle. Tulostuksen hallinnasta vastaa searchprotodropdown.php-tiedosto, jonka avulla määritellään muun muassa yhdellä sivulla näkyvien tulosten määrä. Mikäli haun tulos ylittää 20 kappaletta, jaetaan tietokantahaun vasteet eri tulossivuille, ettei käyttäjän tarvitse selata sivuaan loputtomasti alaspäin ja jotta hakulomake olisi jatkuvasti näkyvissä.

Tiedosto tarkistaa ensimmäisenä, mitä hakutoimintoa käyttäjä on käyttänyt suorittaessaan hakua. Tarkastus toteutettiin hyväksikäyttämällä hakupainikkeiden tunnistimia, sekä POST-funktiota, joka esiteltiin salasanan vaihtoa käsittelevässä luvussa (Kuva 12). Muuttujan vastatessa hakupainikkeen tunnistinta, muodostetaan tietokantakyselylauseke käyttäjän antamien muuttujien mukaisesti, joka ajetaan tietokantaan *mysql_query*-funktiolla.

```
<td align="center">
<input id="SearchButton" type="submit" value="Search" name="SearchButton" />
</td>
<?php
if ($_POST["SearchButton"] == "SearchButton"){
    ... koodia
}
else{
    ... koodia
}
?>
```

Kuva 12. Valitun hakupainikkeen vertailu

Tietokannasta saadun vasteen ollessa suurempi kuin nolla, tulostetaan saadut tulokset käyttäjälle for-ehtolauseen avulla. Muussa tapauksessa käyttäjälle näytetään ilmoitus nollatuloksesta. Lausekkeen iteraattorina toimii muuttuja nimeltään $recordnumber , joka alustetaan arvolla 0. Lausekkeen sisällä oleva tulostus suoritetaan niin monta kertaa

kun iteraattori on pienempi kuin tietokannan vasteen rivien määrä. Tulostettavat muuttujat selvitetään määrittelemällä *mysql_result*-funktion parametreiksi suoritettu kysely ($result), iteraattori, sekä tietokannan vastaava sarake. Tämän jälkeen muuttujat tulostetaan echo-komennolla HTML-taulukkoon (Kuva 13).

```
for($recordnumber = 0; $recordnumber < mysql_numrows($result); $recordnumber++) {
    $modeltype= mysql_result($result,$recordnumber,"model");
    $nick= mysql_result($result,$recordnumber,"nickname");
    $rm= mysql_result($result,$recordnumber,"typedesignator");
    $hw= mysql_result($result,$recordnumber,"hwbuild");
    $status= mysql_result($result,$recordnumber,"statusname");
    $loc= mysql_result($result,$recordnumber,"location");
    $response= mysql_result($result,$recordnumber,"responsible");
    $lastname= mysql_result ($result,$recordnumber,"lastname");
    $firstname= mysql_result ($result,$recordnumber,"firstname");
    $imeicode= mysql_result($result,$recordnumber,"imei");
    echo '<tr>';
        echo '<td>'. $modeltype .'</td>';
        echo '<td>' . $nick . '</td>';
        echo '<td>' . $rm . '</td>';
        echo '<td>' . $hw . '</td>';
        echo '<td>' . $status . '</td>';
        echo '<td>' . $loc . '</td>';
        echo '<td>' . $lastname . ' ' . $firstname;'</td>';
        echo '<td>'. $imeicode .'</td>';
```

Kuva 13. Tietojen tulostaminen

Asiakkaan vaatimusten mukaan tietokantahaun tulokset oli myös pystyttävä tallentamaan Excel-muotoon mahdollisten raporttien luomiseksi. Ratkaisuna tähän määritettiin tiedostoon omat muuttujansa raportointia varten. Muuttujat lähetetään *protoreportexcel.php*-tiedostolle käyttämällä JavaScriptin funktiota *onClick*. Funktion parametreihin määritellään ensin tiedosto, jolle tulostettavat tiedot lähetetään ja tämän jälkeen käyttäjän hakukriteerit.

Tiedosto vastaanottaa muuttujat käyttämällä GET-funktiota, jonka jälkeen suoritetaan tietokantakysely käyttäen muuttujien arvoja. Näin vähennetään selaimen puskurin tallennettavan tiedon määrää ja nopeutetaan sovelluksen toimintaa. Poiketen päätelaitteeseen tulostamisesta, protoreportexcel.php kerää saadut tiedot yhteen muuttujaan ($data). Header-funktion avulla voidaan ohjata selain haluttuun tiedostoon tai sovellukseen (Kuva 14). Käytettäessä header-funktiota on muistettava olla tulostamatta mitään ruudulle ennen headereiden asettamista (echo-komennon sijainti).

```
header("Content-type: application/vnd.ms-excel");
header("Content-Disposition: attachment; filename=\"" . $reportname . "\"" );
header("Expires: 0");
header("Cache-Control: must-revalidate, post-check=0, pre-check=0");
echo $data;
```

Kuva 14. Headerien asettaminen

PHP sisältää joukon valmiita MySQL-funktioita, joista *mysql_num_rows*:a käytettiin selvittääksemme tietokantahaun tuloksen rivien määrän. Tulostussivua luotaessa lasketaan rivien määrä funktion avulla, jotta saadaan selville tarvittavien tulossivujen määrän. Tarvittavien sivujen määrä saadaan selville jakamalla rivit määritetyllä raja-arvolla, tässä tapauksessa 20. Käyttämällä % -merkkiä lausekkeessa tavallisen /-merkin sijaan, saadaan selville mahdollinen jakojäännös. Tämä on otettava huomioon sivujen määrässä lisäämällä yksi sivu, sillä muuten viimeiset tulokset jäisivät tulostamatta.

8.5 Tiedon päivittäminen tietokantaan

Tiedon päivittäminen on toinen sovelluksen merkittävimmistä toiminnoista. Käytön kannalta päivittämisen on oltava nopeaa ja helppoa, sekä mahdollisten virheiden määrää on pyrittävä vähentämään. Päivityslomakkeen ulkoasussa ja toiminnallisuudessa pyrittiin käytettävyyteen, sekä yksinkertaisuuteen tarjoamalla käyttäjälle mahdollisuus muuttaa vain sovelluksen kannalta merkittäviä tietoja ja vähentämällä käyttäjältä vaadittavan syötteen määrää mahdollisimman paljon.

Sovelluksen tarkoituksena on seurata projektissa tarvittavien laitteiden tilaa, lokaatiota ja käyttötarvetta. Sen on myös tarjottava tietoa siitä, kenellä projektin jäsenellä mikäkin laite on käytössä. Tämän vuoksi päivittäminen rajoitettiin normaalikäyttäjien osalta mahdolliseksi ainoastaan, kun käyttäjä on itse vastuuhenkilönä, tai kun laite on käytettävissä ja sijaitsee varastossa. Näin toinen käyttäjä ei voi siirtää laitetta omiin nimiinsä ilman, että siitä informoidaan alkuperäistä omistajaa. Päivitettäessä laitteiden tietoja, on käyttäjän mahdollista muuttaa ainoastaan käyttötarkoitusta, vastuuhenkilöä sekä lokaatiota.

8.5.1 Laitetietojen päivittäminen

Tietokannassa jo olevia tietoja voidaan joutua muuttamaan toisinaan useamman laitteen osalta. Tällöin tietojen päivittäminen yksitellen veisi turhaa aikaa. Siksi pääkäyttäjälle luotiin mahdollisuus päivittää useamman laitteen tiedot kerralla. Vaatimusmäärittelyssä ei rajoitettu tapaa, jolla tietoja tulisi tietokantaan päivittää, joten ratkaisussa käytettiin hyväksi PHP 5:n valmista DOM-luokkaa. Luokka on ns. superglobal, eli muuttujia ei tarvitse erikseen määritellä julkisiksi, vaan ne ovat käytössä automaattisesti sovelluksessa luokkaa käytettäessä. Käyttäjä päivittää *updateProto.xml*-tiedostoon tarvittavat tiedot (Kuva 15), jonka jälkeen tiedosto tallennetaan käyttäjän valitsemaan hakemistoon. Tiedoston käsittely aloitetaan protoupdate.php-tiedoston avulla, jossa käyttäjä valitsee päivitettävän tiedoston html-lomakkeen avulla (Kuva 16).

	A	B	C	D
1	**imei**	**statusname**	**responsible**	**location**
2				
3				
4				

Kuva 15. Päivittämiseen käytettävä xml-tiedosto

```
<form enctype="multipart/form-data" action="import.php" method="post">
  <input type="hidden" name="MAX_FILE_SIZE" value="2000000" />
  <table width="600">
    <tr>
        <td>Note !!, Format must be xml</td>
    </tr>
    <tr>
        <td>Update file:</td>
        <td><input type="file" name="file" /></td>
        <td><input type="submit" value="Upload" /></td>
    </tr>
  </table>
</form>
```

Kuva 16. Laitteiden päivittäseen käytetty HTML-lomake

PHP 5:n sisäänrakennetut tiedostojen lähetysfunktiot mahdollistavat tiedostojen lähettämisen suoraan käyttäjän selaimen kautta. Lomakkeen salaustyypin on oltava useimpia tiedostomuotoja käytettäessä *multipart/formdata,* muuten tiedoston lähettäminen epäonnistuu. Tiedoston maksimikoko määritellään *name* ja *value* attribuuttien avulla. Toi-

miakseen PHP vaatii, että tiedoston maksimikoko määritellään lähetyslomakkeessa ennen varsinaisen tiedoston lähettämistä. Samalla tarkistetaan lähetettävän tiedoston koko heti lähetyksen alussa, jolloin liian suuren tiedoston ollessa kyseessä, saa käyttäjä tästä tiedon ja vältytään turhalta odottamiselta.

Lähetetty tiedosto ladataan muuttujaansa *DOMDocument::load*-funktiolla, joka saa parametreinaan tiedostopolun, sekä selaimen luoman väliaikaisen tiedostonimen. Tiedoston sisältämät rivit tallennetaan DOM-luokan funktiolla *getElementsByTagName*, joka saa parametrinaan xml-tiedoston elementin arvon (tässä tapauksessa Row). Rivien ja solujen sisältö käydään läpi sisäkkäisillä foreach-ehtolausekkeilla rivin jokainen solu kerrallaan. Solun sisältämä arvo tallennetaan muuttujaansa DOM-funktion palauttaman *nodeValuen* avulla (Kuva 17).

```
if ( $_FILES['file']['tmp_name'] ){
    $dom = DOMDocument::load( $_FILES['file']['tmp_name'] );
    $rows = $dom->getElementsByTagName( 'Row' );
    $first_row = true;
    foreach ($rows as $row){
        if ( !$first_row ){
            $imei = "";
            $statusname = "";
            $responsible = "";
            $location = "";

            $index = 1;
            $cells = $row->getElementsByTagName( 'Cell' );
            foreach( $cells as $cell ){
                $ind = $cell->getAttribute( 'Index' );
                if ( $ind != null ) $index = $ind;
                if ( $index == 1 ) $imei = $cell->nodeValue;
                if ( $index == 2 ) $statusname = $cell->nodeValue;
                if ( $index == 3 ) $responsible = $cell->nodeValue;
                if ( $index == 4 ) $location = $cell->nodeValue;
                $index += 1;
            }
        }
        if ($imei != "" ){
            upd_proto( $imei, $statusname, $responsible, $location);
            $count=$count+1;
        }
  $first_row = false;
  }
  echo  $count . ' Prototypes updated to the Proto Management Tool';

}
```

Kuva 17. DOM-funktion käyttö luettaessa xml-tiedostoa

Funktion upd_Proto avulla päivitetään xml-tiedostosta kerätyt muuttujat tietokantaan, sekä luodaan tapahtumamerkintä prototypehistory-tauluun. Funktio saa parametreikseen samat muuttujat, jotka käyttäjä täytti xml-tiedostoon. Funktio avaa tietokantayhteyden, sekä suorittaa UPDATE-kyselyn tietokantaan käyttämällä tunniste-koodia tunnisteena ja päivittämällä muuttuneet tila-, vastuu-, sekä lokaatiotiedot. Päivityksen aikana tapahtuvasta virheestä (esim. tietokannasta löytymättömästä tunniste-koodista) annetaan käyttäjälle virheilmoitus päivityksen epäonnistumisesta (Kuva 18). Onnistuneesta päivityksestä informoidaan käyttäjää tulostamalla näyttöpäätteeseen viesti päivitettyjen laitteiden kokonaismäärästä.

```
function upd_proto( $imei,$statusname, $responsible, $location ){
    $userid = $_COOKIE["ptmcookie"];
    include '../../lib/config.php';
    include '../../lib/opendb.php';
    $sql = "update prototype
        set
        statusname='$statusname',
        responsible='$responsible',
        location='$location'
        where
        imei='$imei'";
        $result = mysql_query($sql) or die('Prototypes were not updated, plese check the details. ');
        $historyInsert = "INSERT INTO prototypehistory
        (imei,loginfo)
        VALUES
        ('$imei','$userid updated $imei and changed location to $location, responsible to $responsible and status to $statusname ')";
        $result = mysql_query($historyInsert) or die('Historyinfo not added,but phones was,please check the details.' . mysql_error());

    include '../../lib/closedb.php';
}
```

Kuva 18. Päivitysfunktio

8.5.2 Yksittäisen laitteen tietojen päivittäminen

Käyttäjän ottaessa laitteen käyttöönsä, on hänen päivitettävä laitteen tila, vastuuhenkilö sekä lokaatio sovelluksen avulla. Käyttäjän suorittaman haun tuloksien lisäksi laitteen tietojen perään tulostetaan Edit-painike, jonka avulla tietoja voidaan päivittää. Painike lähettää laitteesta olevat tiedot *protoedit.php*-tiedostolle, joka avautuu käyttäjälle uuteen ikkunaan. Käyttäjälle näytetään mahdolliset muutettavat tiedot pudotusvalikoissa, sekä niiden alapuolella tämänhetkiset tiedot.

Käyttäjän tekemät muutokset saadaan selville isset-funktion avulla. Funktio tutkii, onko määriteltyjä muuttujia olemassa, eli onko käyttäjä muuttanut käyttötarkoitusta, vastuuhenkilöä tai lokaatiota. Tämän lisäksi funktio tarkastaa, onko Tallenna-painikkeella muuttuja-arvoa, eli onko käyttäjä tallentamassa tekemänsä muutokset.

Ehtojen ollessa tosi, eli käyttäjän muutettua jotain yllämainituista arvoista, uuden muuttujan sisältö tallennetaan tietokantaan UPDATE-kyselyllä sekä tallennetaan kyseinen tapahtuma tietokannan lokitauluun. Tiedot päivitetään käyttämällä aktiivisen, päivitettävissä olevan tuotteen uniikkia tunniste-koodia, joka saadaan lomakkeen muuttujasta $imei. Lokitiedostoon tallennetaan samanaikaisesti käyttäjän suorittama tapahtuma lisäämällä *prototypehistory*-tauluun uusi tapahtuma. Tapahtuma tallentuu päivitettävän tunniste-koodin avulla, mutta lokitiedoston tarkastelua varten viesti muodostetaan sen tiedon perusteella, mitä tietoja käyttäjä kyseisen laitteen kohdalla päivitti. Alla olevassa

kuvassa 19 luodaan lokitiedostoon tallennettava tapahtuma, muuttuja $message, sekä päivitetään kyseisen laitteen tiedot tietokannan *prototype*-tauluun.

```
$message = $userid . " changed ";
if ($newstatus != $statusname)
  $message .= " status ";
if ($newuser != $responsible)
  $message .= " responsible ";
if ($newlocation != $location)
  $message .= " location ";
if ($newoperator != $operator)
  $message .= " operator ";
if ($newrecipient != $recipient)
  $message .= " recipient ";

$sql = "update prototype
set
statusname = '$newstatus',
responsible = '$newuser',
location = '$newlocation',
operatorsample = '$newoperator',
samplerecipient = '$newrecipient'
where
imei = '$imei'";
$sqll = "INSERT INTO prototypehistory
          (imei,
          loginfo)
          VALUES
          ('$imei',
          '$message')";
$result = mysql_query($sql) or die('Update failed. ' . mysql_error());
$resultl = mysql_query($sqll) or die('Update failed. ' . mysql_error());
```

Kuva 19. Tietokannan päivitys ja historiaviestin luominen sekä lisääminen tietokantaan

8.6 Tiedon poistaminen tietokannasta

Sovellusta määriteltäessä ei asiakas kokenut tarpeelliseksi poistaa sovellukseen talletettavia tietoja, joten poistamisen sijaan määriteltiin tietojen aktiivisuus. Laitteille ei määritelty aktiivisuuskenttää, joten ne ovat aina aktiivisia, mutta käyttäjiä hallinnoivassa usertype-taulussa on sarake ’active’.

Käyttäjien poistuessa ei käyttäjää siis poisteta tietokannasta kokonaan, vaan määritellään hänen aktiivisuutensa. Oletusarvoisena käyttäjän anoessa tunnuksia sovellukseen

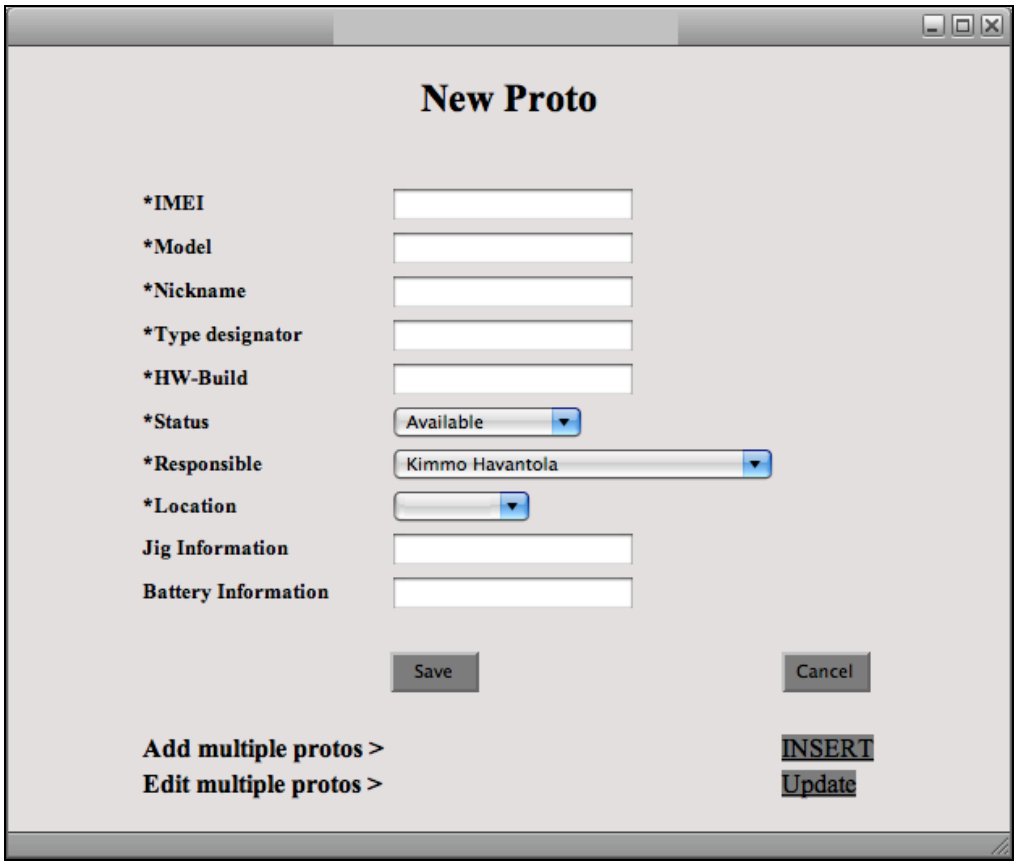
määritellään hänen aktiivisuutensa 1:ksi, eli inaktiiviseksi. Pääkäyttäjän hyväksyttyä käyttäjän tunnukset, päivitetään aktiivisuus 0:ksi. Näin voidaan helposti poistaa käyttäjiä sovelluksen näkyvistä poistamatta heitä kuitenkaan täydellisesti järjestelmästä. Sovelluksen toiminnoissa näkyvät ainoastaan ne käyttäjät, jotka on luokiteltu aktiivisiksi. Käyttäjätietojen päivittämisestä vastaa *useredit.php*-tiedosto. Se saa parametrinaan käyttäjätunnuksen, jonka avulla tulostetaan taulussa olemassa olevat tiedot käyttäjän nähtäville. Parametri lähetetään Edit-painikkeen avulla, kuten tietojen tulostus-kappaleessa aikaisemmin esiteltiin. Aktiivisuus voidaan ajatella määriteltäväksi Boolean-logiikan mukaisesti. Tällöin muuttujalla voi olla vain arvot Tosi (True) tai Epätosi (False). Aktiivisuutta muutetaan lomakkeesta löytyvästä checkboksista. Merkittynä käyttäjä on inaktiivinen ja tyhjä laatikko osoittaa käyttäjän olevan aktiivinen.

8.7 Tiedon lisääminen tietokantaan

Uusia tuotteita on pystyttävä lisäämään tietokantaan sekä yksittäin että suuremmissa erissä. Tietoja lisättäessä on jo ennen tietokannan yhteyden muodostamista tarkastettava, että kaikki lomakkeessa vaaditut kentät ovat täytetty. Seuraavissa alaluvuissa käsittelen yksittäisen tiedon sekä tietoryhmän lisäämistä tietokantaan, sekä mitä ratkaisuja toteutuksessa käytettiin. Tietojen lisäämiseen on oikeus ainoastaan sovelluksen pääkäyttäjillä.

8.7.1 Laitteen lisääminen tietokantaan

Yksittäisen laitteen lisäämisestä tietokantaan vastaa *newproto.php*-tiedosto. Pakollisten kenttien tarkastaminen on toteutettu jo edellä mainitun isset-funktion avulla. Laitteen pakolliset tiedot ovat yksilöllinen tunniste-koodi, mallinumero, tuotenimike, laitetyyppi, laitteistoversio, tila, vastuuhenkilö sekä lokaatio eli missä kyseinen laite sijaitsee. Vaihtoehtoisia kenttiä lomakkeessa ovat soveltuva akkutyyppi, sekä tuotteen ohjelmointiin soveltuva adapteri (Kuva 20).



Kuva 20. Uuden laitteen lisääminen

Lisättäessä mallia jota tietokannassa ei vielä ole, määritellään tuotteelle raja-arvo, jonka alitettuaan tuotteen *vähäisyydestä* informoidaan pääkäyttäjille. Raja-arvo määritellään tietokannassa tilassa *Available* olevien tuotteiden mukaisesti. Raja-arvon alittuessa, eli tuotteen saatavuuden ollessa sitä alempi, luodaan tietokannan tauluun protoalert uusi ilmoitus tapahtuneesta. Hälytys poistuu pääkäyttäjän näkymästä vasta, kun tuotteen *Available*-määrä on raja-arvoa suurempi.

Manuaalisesti tietoja syötettäessä on aina mahdollisuus virheeseen, etenkin tunniste-koodin kohdalla saattaa numeroita olla liikaa tai liian vähän. Validi tunniste-koodi koostuu viidestätoista numerosta väliltä 0-9, eikä siinä saa olla muita merkkejä. Syöte tarkastetaan lomakkeesta käyttämällä PHP:n valmiita funktioita strlen, joka esiteltiin jo salasanan vaihtoa käsittelevässä luvussa, sekä preg_match. Funktioon määritellään käyttäen säännöllisten lausekkeiden ehtoja, sekä määritellään parametrina muuttuja, johon ehtoja verrataan. Kuvan 21 esimerkissä kaikki merkit väliltä 0-9 ovat hyväksyttyjä ja niitä saa olla 15 peräkkäin. Toteutuessaan funktio palauttaa arvon 1, muuten palautettava arvo on 0.

```
if ((strlen($imei) < 15) || (preg_match('/[0-9]{15}/', $imei) == 0))
{
    echo "<script>alert(\"Sorry, check the imei lenght or that imei is valid! Proto not added\");</script>";
    $correct=1;
}
```

Kuva 21. Säännöllisten lausekkeiden esimerkki

Lomakkeessa on kolme kenttää, joihin käyttäjä ei voi manuaalisesti syöttää tietoja, vaan tiedot haetaan jo olemassa olevien tietojen perusteella tietokannasta. Näille kolmelle kentälle; tila, vastuuhenkilö sekä lokaatio, toteutettiin omat funktionsa joiden toiminnallisuus on täysin samanlainen. Tilan pudotusvalikko luodaan funktiolla statusdropdownlist. Funktion avulla noudetaan tietokannasta jokainen mahdollinen tila ja esitetään ne pudotusvalikossa. Funktio saa parametrina valitun tilan, joka esiteltävän lomakkeen tapauksessa on oletusarvoisesti tyhjä.

Kyselyn tulokset käydään läpi for-ehtolausekkeella, johon määritellään muuttuja tietokannan taulun sarakkeen mukaisesti mysql_result-funktion avulla. Saadut tulokset tulostetaan pudotusvalikkoon, joka valittaessa palauttaa arvon *status*. Saatu arvo tallennetaan newproto.php-tiedostossa muuttujaan $statusname POST-funktiolla. Kuvassa 22 statusdropdownlist-funktio.

```
function statusdropdownlist($selectedstatus){
    if ($selectedstatus == ""){
        include 'lib/config.php';
        include 'lib/opendb.php';
        //get all available statuses from database
        $sql = "SELECT
            statusname
            FROM
            statustype";
        $result = mysql_query($sql) or die('Query failed. ' . mysql_error());
        echo '<select name="status">';
        for($recordnumber = 0; $recordnumber < mysql_numrows($result); $recordnumber++) {
            $statusname = mysql_result($result,$recordnumber,"statusname");
                echo '<option value="' . $statusname . '">';
                echo $statusname;
                echo '</option>';
        }
        echo '</select>';
        include 'lib/closedb.php';
    }
    else{
        include 'lib/config.php';
        include 'lib/opendb.php';
        //get all available statuses from database
        $sql = "SELECT
            statusname
            FROM
            statustype";
        $result = mysql_query($sql) or die('Query failed. ' . mysql_error());
        echo '<select name="status">';
        for($recordnumber = 0; $recordnumber < mysql_numrows($result); $recordnumber++) {
            $statusname = mysql_result($result,$recordnumber,"statusname");
            if ($statusname == $selectedstatus){
                echo '<option value="' . $statusname . '" selected>';
                echo $statusname;
                echo '</option>';
            }
            else{
                echo '<option value="' . $statusname . '">';
                echo $statusname;
                echo '</option>';
            }
        }
        echo '</select>';
        include 'lib/closedb.php';
    }

}
```

Kuva 22. Funktio statusdropdownlist, jolla haetaan tietokannan eri tilat pudotusvalikkoon

8.7.2 Laitteiden lisääminen tietokantaan

Asiakas tilaa tarvittavat tuotteet suoraan tehtailtaan, jolloin saapuvat määrät ovat yleensä runsaita. Näiden tietojen päivittäminen yksitellen tietokantaan veisi tarpeettoman paljon aikaa, joten sovelluksen tietokantaan lisääminen toteutettiin samoin kuin useampien laitetietojen päivittäminenkin. Sekä käytettävään xml-tiedostoon, että päivittävään funktioon tehtiin tarvittavat muutokset vaadittujen tietojen osalta. Lisätessä laitteita tietokantaan, tarvitaan tunniste-koodin, tilan eli statuksen, vastuuhenkilön sekä lokaation lisäksi myös muita yksilöivämpiä tietoja.

8.8 Varmuuskopiointi

Sovelluksen palvelimena toimivan tietokoneen kovalevylle on tallennettuna sekä jatkuvasti päivittyvä tietokanta, että ohjelmakoodit joilla käyttöliittymän toiminnallisuuksia ohjataan. Proto Toolia käytetään useasta eri maanosasta, joten käyttäjiä sovelluksella on eri vuorokaudenaikoihin. Palvelin on käynnissä ympäri vuorokauden lukuun ottamatta pakollisia tietoturvapäivityksiä, jolloin se käytetään alhaalla päivitysten asentamiseksi. Toimintakyvytön palvelin on virhetilanteen sattuessa pystyttävä palauttamaan mahdollisimman nopeasti toimintakuntoon ilman suurempia tiedon menetyksiä.

Käytettävä tietokantamoottori asettaa varmuuskopiointiohjelmistolle omat vaatimuksensa. InnoDB-moottorin mukana tuoma viite-eheyden ylläpitäminen osoittautui varmuuskopioinnin osalta ongelmaksi. Useat Internet-sivustoilta löytyneet valmiit ohjelmakoodit eivät tukeneet InnoDB-moottoria, eivätkä osanneet lukita kopioitavia tauluja suorituksen ajaksi. MySQL:n oma hallinnointisovellus MySQL Administration osoittautui niin käytettävyydeltään, ulkonäöltään, että ominaisuuksiltaan soveltuvan tarkoitukseen erinomaisesti. Graafisen käyttöliittymän avulla käyttäjä kykenee hallinnoimaan tietokantapalvelimen replikointia, varmuuskopiointia sekä käyttäjiä vaivattomasti ilman aikaisempaa tietokantaosaamista.

Tietokannan tauluista ja niiden tiedoista tallennetaan varmuuskopiot asiakkaan verkkolevyllä sijaitsevaan suojattuun kansioon. Tietokannasta tallennetaan kahta eri varmennustiedostoa. Päivittäin kello 23:00 ajettava kopiointi varmistaa, että pahimmillaan me-

netetään ainoastaan yhden päivän aikana laite-, käyttäjä- tai historiatauluihin tallentuneet merkinnät, sekä sovelluksen nopean toimintakunnon palauttamisen. Sovelluksen toiminnallisuuteen vaikuttavat PHP-, XHTML- ja CSS-koodit varmistetaan käyttämällä Windowsin ajastettuja tehtäviä. Tehtävien avulla varmuuskopiot tallennetaan niille varattuun kansioon verkkolevylle päivittäin, sekä viikoittaiset varmuuskopiot niille varattuun kansioon. Samanaikaisesti ajettavat tehtävät saattavat hidastaa palvelimen toimintaa, joten tehtävät ajastettiin alkavaksi tunnin välein. Varmuustiedostoja säilytetään kolme kuukautta verkkolevyllä, jonka jälkeen ne poistetaan palvelimelta.

9 Nykytila ja Tulevaisuus

Proto Tool -sovellus on ollut asiakkaan käytössä nyt noin 1,5 vuotta. Käyttäjämäärä on kasvanut alkujaan arvioidusta n. 200 henkilöstä hieman yli kuuteensataan henkilöön ja käyttäjiä on tällä hetkellä usealla eri mantereella. Sovellusta käytetään myös yhä enemmän asiakkaan johdon tehtäviin sen tarjoamien raporttien, sekä ajan tasalla olevien tilannekatsausten vuoksi. Sovelluksen elinkaaren aikana siihen on toteutettu useita uusia toiminnallisuuksia, joita asiakas on huomannut tarvitsevansa pystyäkseen suunnittelemaan tulevia allokointeja aikaisempaa tehokkaammin. Sovellusta on myös pienin muutoksin otettu käyttöön asiakkaan muiden tiimien projekteihin sen helpon käytettävyyden ja sopeutuvuuden vuoksi.

Käytön aikana ilmeni tarve pystyä poistamaan myös laitteita sovelluksen tietokannasta, joten toiminnallisuus lisättiin pääkäyttäjän näkymään. Toteutuksessa käytettiin vastaavalla menetelmällä DOM-luokkaa kuin mitä opinnäytetyöni Laitteiden tietojen päivittämistä käsittelevässä luvussa esittelin tietojen lukemista xml-tiedostosta. Ennen poistamista funktio check_proto tarkastaa, löytyvätkö kaikki käyttäjän tallentamat tunniste-koodit tietokannasta. Tietokantavasteen puuttuessa muotoillaan käyttäjän antamaa tunniste-koodia poistamalla siitä kuusi ensimmäistä merkkiä. Näin korjattua tunniste-koodia verrataan uudestaan tietokannan vasteisiin jokerihaun avulla. Jokerihaku mahdollistaa tietokannassa määritellyn merkkijonon etsimisen suuremman merkkijonon sisältä. Ennen varsinaista poistamista sovellus kerää jokaisen laitteen historia- ja tapahtumatiedot HTML- tiedostoon, joka nimetään tapahtuman päivämäärän mukaisesti. Kaikkien samanaikaisesti poistettavien laitteiden tiedot tallennetaan samaan tiedostoon ja onnistuneen tiedoston kirjoituksen jälkeen kaikki tiedot poistetaan tietokannasta.

Tarkasteltaessa sovelluksen käyttäjämäärien kehitystä, voidaan todeta sovelluksen käyttöasteen kasvavan tasaisesti. Samoin sen tarjoamien raporttien, sekä seurannan merkitys on korostunut asiakkaan organisaatiossa jatkuvasti ja sovellusta on suunniteltu otettavaksi käyttöön myös yrityksen muiden laitteistojen kanssa toimivien tiimien toimesta. Voidakseen vastata jatkuvasti kasvavaan käyttöasteeseen, sekä tarjotakseen jatkossakin käyttökelpoista tietoa laitteiden liikkeistä ja tarpeesta eri toimipisteillä, on sovellukseen tehtävä joitain muutoksia. Alla on kuvattuna lähitulevaisuuden kannalta merkittävimmät muutoksen kohteet, sekä muutamia ratkaisumalleja.

Tietokannan hallinnoima tietomäärä kasvaa kuukausittain uusien laitteiden ja uusien käyttäjien toimesta. Tällä hetkellä palvelimena toimiva tavallinen pöytäkone ja sen tallennuskapasiteetti eivät tule jatkossa riittämään, joten palvelinta on päivitettävä. Uuden palvelimen olisi hyvä tukea RAID-5 levynohjausta, jolloin voitaisiin varmistaa tiedon säilyminen mahdollisten kiintolevyrikkojen varalta. Neljä 1T:n kiintolevyä mahdollistaisivat tietokannan peilaamisen kahdelle 1T:n levylle, jolloin yhden kiintolevyn rikkoutuessa palvelin olisi yhä käyttökuntoinen. Tietojen peilaus kahdelle muulle levylle ei kuitenkaan kestäisi kahden kiintolevyn rikkoutumista samanaikaisesti, vaan tällöin kaikki olemassa oleva tieto menetettäisiin. Palvelimen suorituskykyä voidaan lisäksi parantaa päivittämällä sen sisäinen muisti 4Gb:n, sekä vaihtamalla tehokkaampi prosessori nykyisen tilalle.

Sovelluksen käyttöliittymää toteutettaessa ei katsottu tarpeelliseksi toteuttaa sitä olio-ohjelmoinnin perusteiden mukaisesti, vaan ns. proseduraalisen ohjelmoinnin perusteella, jossa toiminnallisuudet toteutetaan kuhunkin sivuun erikseen, jolloin samoja funktiota voidaan joutua kirjoittamaan useampaan kertaan sovelluksen eri sivuille. Alati kasvava tarve eri raportteihin, sekä uusiin toiminnallisuuksiin on kuitenkin johtanut sovelluksen hankalaan ylläpitoon. Koska useat eri sivut käyttävät samoja, uudelleen kirjoitettuja funktiota, joudutaan päivitettäessä käymään läpi kaikki ne sovelluksen näkymät, joissa päivitettävää funktiota käytetään.

Valmiin sovelluksen ”kääntäminen” oliopohjaiseksi vaatii paljon esisuunnittelua, sekä muita valmisteluja toteutuakseen, mutta valmistuttuaan takaa sovelluksen käyttöliittymän päivitettävyyden, koodin selkeyden sekä paremman suorituskyvyn. Alustava suunnitelma sovelluksen ohjelmointitekniikan muuttamiselle on laadittu, mutta mitään konkreettista asiakkaan kanssa ei ole vielä tätä kirjoitettaessa sovittu.

Pienempien näyttöjen, kuten matkapuhelinten ja kämmentietokoneiden, lisääntynyt käyttö on myös herättänyt tulevaisuuden kannalta uusia suunnitelmia. Mikäli sovellusta ryhdytään kääntämään oliopohjaiseksi, päivitetään käyttöliittymä samalla skaalautumaan myös pienempiin näyttöpäätteisiin, mahdollistaen tällä tavoin sovelluksen käytön myös muilta laitteilta. Tällä hetkellä sovelluksen käyttöliittymä skaalautuu ainoastaan 640 x 480 kokoisiin ja sitä suurempiin näyttöihin.

Toimeksiannon tavoitteena oli suunnitella ja toteuttaa asiakkaan tarpeita ja määrityksiä vastaava relaatiotietokanta, sekä sitä hyödyntävä käyttöliittymä. Toteutettu tietokanta on osoittanut skaalautuvansa asiakkaan tarpeisiin loistavasti, eikä sen toiminnallisuudessa ole havaittu käytön aikana vakavia virheitä. Tietokanta pystyy käsittelemään suuren määrän tietoa nopeasti ja luotettavasti, sekä tarjoamaan luotettavan sijainnin asiakkaan tarvitsemille tiedoille.

Käyttöliittymän suunnittelun ja toteutuksen tavoitteena oli toteuttaa selkeä ja helppokäyttöinen käyttöliittymä, joka tarjoaa käyttäjälleen työkalut hakea tietokantaan tallennettuja tietoja käyttäjän valitsemien hakukriteerien perusteella. Käyttöliittymään voidaan tarvittaessa toteuttaa asiakkaan tarvitsemia lisätoimintoja ilman ulkoasun rikkoutumista. Sovelluksen tarjoamat mahdollisuudet tulevat käyttöliittymän valikkorakennetta tarkastelemalla hyvin esiin ja käyttöliittymän terminologia on käyttäjiensä mukainen. Käyttöliittymä sisältää normaalinäkymän lisäksi pääkäyttäjän näkymän, jota tässä opinnäytetyössä ei esitelty tarkemmin, mutta joka mahdollistaa pääkäyttäjän tarvitsemien tehtävien toteuttamisen nopeasti ja tehokkaasti.

Toiminnallisuudelle asetetut tavoitteet saavutettiin ja asiakkaalle tarjottiin heidän tarkoituksiinsa soveltuva sovellus, jonka avulla saadaan ajankohtaista tietoa kunkin toimipisteen valmiuksista, sekä voidaan seurata laitteiden liikkeitä tiimin sisällä. Sovellus on lisäksi muokattavissa muita vastaavia tarkoituksia varten ja sen käyttöönotto, sekä ylläpito muutamaa poikkeusta lukuun ottamatta on helppoa.

Opinnäytetyöni tavoitteeni oli kuvata sovelluksen suunnittelu- sekä toteutusvaiheita, sekä esitellä niissä käytettyjä ratkaisuja. Henkilökohtaisella tasolla tavoitteenani oli oppia käyttämään PHP-ohjelmointikieltä yhdessä tietokantasovellusten kanssa ja ratkaisemaan käytettävyyden kannalta merkittävämpiä ongelmia. Opinnäytetyön avulla sain arvokasta kokemusta pienimuotoisesta ohjelmistosuunnittelusta, sekä ohjelmistokehityksestä. Opin myös näkemään ja sisäistämään proseduraalisen ja oliopohjaisen ohjelmoinnin edut. Kuvaamani suunnittelu- ja toteutustekniikat auttavat ohjelmoinnin ensikertalaisia tunnistamaan verkkosovellusten kannalta merkittävimmät asiat, sekä tarjoavat niihin tässä toteutuksessa käytettyjen ratkaisujen tarkat kuvaukset.

Lähteet

Abyss Web Server
[www-sivu]. [viitattu 28.3.2010]
Saatavissa: http://www.aprelium.com/abyssws.html

Boumphrey, F.; Greer, C.; Raggett, D.; Ragget, J.; Schnitzenbaumer, S.; Wugofski, T.;
Kolehmainen, K. (kääntäjä) 2003. Inside XTML: ohjelmoijan käsikirja.
Helsinki: Edita. ISBN 951-826-228-4

Heinisuo, R. 2003. PHP ja MySQL tietokantapohjaiset verkkopalvelut. 2. painos.
Jyväskylä: Gummerus. ISBN 951-762-833-1

Korpela, K J. 2008. CSS verkkosivujen muotoilussa.
Jyväskylä: WSOY.
ISBN 978-951-0-34044-8

Lowery, W J. 2005. CSS Hacks & Filters.
Indianapolis: Wiley Publishing.
ISBN 0-7645-7985-1

MySQL Reference Manual.
[online] [viitattu 12.04.2010]
Saatavissa: http://dev.mysql.com/doc/refman/5.1/en/index.html

PHP.net.
[www-sivu]. [viitattu 06.04.2010]
Saatavissa: http://www.php.net/manual/en/history.php.php

Schlossnagle, G. 2004. Advanced PHP Programming.
Indianapolis: Sams Publishing.
ISBN 0-672-32561-6

Wikipedia 2010a.
[www-sivu]. [viitattu 12.04.2010]
Saatavissa: http://fi.wikipedia.org/wiki/MySQL

Wikipedia 2010b.
[www-sivu]. [viitattu 14.04.2010]
Saatavissa: http://en.wikipedia.org/wiki/PHP

Liitteet

Liite 1. Tietotyyppien erot SQL-92- ja MySQL:n välillä

SYNTAKSI	SQL-92	MySQL 5.1
TINYINT		X
TINYINT (N)		X
TINYINT UNSIGNED		X
TINYINT (N) UNSIGNED		X
TINYINT UNSIGNED ZEROFILL		X
TINYINT (N) UNSIGNED ZEROFILL		X
TINYINT ZEROFILL		X
TINYINT (N) ZEROFILL		X
SMALLINT		X
SMALLINT (N)		X
SMALLINT UNSIGNED		X
SMALLINT (N) UNSIGNED		X
SMALLINT UNSIGNED ZEROFILL		X
SMALLINT (N) UNSIGNED ZEROFILL		X
SMALLINT ZEROFILL		X
SMALLINT (N) ZEROFILL		X
MEDIUMINT		X
MEDIUMINT (N)		X
MEDIUMINT UNSIGNED		X
MEDIUMINT (N) UNSIGNED		X
MEDIUMINT UNSIGNED ZEROFILL		X
MEDIUMINT (N) UNSIGNED ZEROFILL		X
MEDIUMINT ZEROFILL		X
MEDIUMINT (N) ZEROFILL		X
INT	X	X
INT (N)		X
INT UNSIGNED		X
INT (N) UNSIGNED		X
INTEGER	X	X
INTEGER (N)		X
INTEGER UNSIGNED		X
INTEGER (N) UNSIGNED		X
INTEGER UNSIGNED ZEROFILL		X
INTEGER (N) UNSIGNED ZEROFILL		X
INTEGER ZEROFILL		X
INTEGER (N) ZEROFILL		X
BIGINT		X
BIGINT (N)		X
BIGINT UNSIGNED		X
BIGINT (N) UNSIGNED		X

Liite 2. Tietokannan testitapaukset

Test case id	Test case name	Priority	Reference	Result
6	Add company location	1		**PASS**
18	Add location	3		**PASS**
1	Add new alert	3		**PASS**
3	Add new company	1		**PASS**
9	Add new historystatus	2		**PASS**
11	Add new jig	2		**PASS**
15	Add new jighistory	3		**PASS**
29	Add new SIM	3		**PASS**
45	Add new user	3		**PASS**
43	Add new user history	3		**PASS**
40	Add new useralert	3		**PASS**
21	Add prototype	3		**PASS**
25	Add prototype history	3		**PASS**
35	Add SIM history	3		**PASS**
37	Add status	3		**PASS**
17	Change jigstatus	3		**PASS**
28	Change prototype status	3		**PASS**
49	Change SIM status	3		**PASS**
2	Confirm alert	3		**PASS**
41	Confirm useralert	3		**PASS**
4	Edit company	1		**PASS**
7	Edit company location	1		
12	Edit jig	2		
19	Edit location	3		
22	Edit prototype	3		
30	Edit SIM	2		
38	Edit status	3		
46	Edit user	3		**PASS**
5	Remove company	1		**PASS**
8	Remove company location	1		
13	Remove jig	1		
20	Remove location	3		
23	Remove prototype	2		**PASS**
31	Remove SIM	2		
39	Remove status	3		**PASS**
47	Remove user	3		**PASS**
10	Search historystatus	3		
16	Search jighistory	3		**PASS**
14	Search jigs	3		
24	Search prototype	3		
26	Search prototype history	3		
27	Search prototype status	3		
32	Search SIM	3		
33	Search SIM history	3		
34	Search SIM status	3		
36	Search status history	3		
48	Search user	3		**PASS**
44	Search user history	3		**PASS**
42	Search useralert	3		**PASS**