KEMI-TORNION AMMATTIKORKEAKOULU

WWW-pohjainen laiterekisterisovellus

Ruikkala, Aki

Tietojenkäsittelyn koulutusohjelman opinnäytetyö
Web-asiantuntijan suuntautumisvaihtoehto
Tradenomi

TORNIO 2009

SISÄLTÖ

TIIVISTELMÄ

Ruikkala, Aki 2009. WWW-pohjainen laiterekisteri. Opinnäytetyö. Kemi-Tornion ammattikorkeakoulu. Kaupan ja kulttuurin toimiala. Sivuja 27. Liitteet 1 – 3.

Opinnäytetyön tavoitteena on toteuttaa selainpohjainen laiterekisterisovellus, jolla voidaan hallita yrityksen tietoteknisiä laitteita. Työn toimeksiantajana toimii Verkkoasema Oy. Opinnäytetyö sisältää käyttöliittymän ja tietokannan suunnittelun sekä sovelluksen toteutuksen ja testauksen.

Työ toteutetaan Eclipse-ohjelmointiympäristössä, apuna käytetään EditPlus-tekstieditoria, GIMP-kuvankäsittelyohjelmaa sekä PuTTY-ssh-asiakasohjelmaa Apache-palvelinohjelmiston etäkäyttöön. Sovelluksen graafinen käyttöliittymä toteutetaan HTML- ja CSS-kuvauskielillä, toiminnallisuudet PHP-ohjelmointikielellä ja datan keskipisteenä käytetään MySQL-tietokantaa.

Www-pohjaisella laiterkisterisovelluksella hallitaan yrityksen tietoteknisiä laitteita. Laiterekisterisovelluksessa on kaksi käyttäjäryhmää: ylläpitäjät ja loppukäyttäjät. Kirjautumalla ylläpitäjän käyttäjätunnuksilla voidaan lisätä, muokata, poistaa ja selata laitetietoja sekä hallita käyttäjätilejä. Loppukäyttäjällä kirjautuminen mahdollistaa vain laitetietojen selauksen.

Asiasanat: PHP, MySQL, tietokanta, käytettävyys, käyttöliittymä, laiterekisteri.

ABSTRACT

Ruikkala, Aki. 2007. Web based database register for devices. Bachelor’s thesis. Kemi-Tornio University of Applied Sciences. The unit of Business and Data Processing. Tornio. 27 pages and appendices 1 – 3.

The main objective of the thesis is to design and create a web based register for devices. Company's devices are added to device register where user can manage their devices, software, licences etc. Thesis contains graphical user interface design, software's database design, building core functions and testing of the final product. Work is assigned by Verkkoasema Ltd.

Software will be produced with software development environment called Eclipse together with EditPlus text editor, GIMP image manipulation program and a terminal emulator application called PuTTY. Graphical user interface will be produced with HTML markup language and CSS style sheet language, coding the core functions will be used PHP scripting language which will use a MySQL database.

Logging in as an admin user to the device register user can manage company's hardware and software by adding, removing and modifying device information. With a normal privileges user can just browse devices and check current location of device or bonds between devices.

Keywords: PHP, MySQL, database, usability, user interface, device register.

1 JOHDANTO

1.1 Työn lähtökohdat ja tavoite

Opinnäytetyön tavoitteena on suunnitella ja toteuttaa www-pohjainen laiterekisterisovellus, jolla voidaan hallita yrityksen pöytäkoneita, puhelimia, ohjelmistoja, lisenssejä ja muuta tietoteknistä omaisuutta. Laiterekisterisovelluksen tarkoituksena on helpottaa yrityksen säännöllistä omaisuuden seurantaa ja pysyä ajan tasalla yrityksen omaisuudesta. Laiterekisterin pitäminen ajan tasalla helpottaa laitehankintoja, kun tiedetään yrityksen laitekannan tiedot ja mitä laitteita tulisi uusia. Työn toimeksiantajana toimii oululainen ohjelmistoyritys Verkkoasema Oy, jolla on toimipisteitä Oulussa ja Joensuussa. Tradenomin koulutukseen liittyvä työharjoittelu suoritettiin Verkkoasema Oy:ssä, Oulun päätoimipisteessä. Tarve sovellukselle löydettiin työharjoittelun aikana, josta idea sai alkunsa.

1.2 Työn toteutus ja menetelmät

Käyttöliittymä toteutetaan HTML- ja CSS-kuvauskielillä, joilla sovellus voidaan toteutettaa visuaalisesti yksinkertaiseksi, kevyeksi sekä käyttöjärjestelmä- ja selainriippumattomaksi. Sovelluksen toiminnallisuudet toteutetaan PHP:llä ja Javascript:llä, tietojen tallennuksesta huolehtii Apache-palvelinohjelmisto MySQL-tietokannalla. Sovellus tulee toimimaan yrityksen sisäverkossa, jolloin tietoturvariski pienenee.

Työn toteutus voidaan käytännössä jakaa kuuteen vaiheeseen: esitutkimukseen, määrittelyyn, suunnitteluun, toteutukseen, testaukseen ja käyttöönottoon. Esitutkimus aloitetaan tutustumalla muihin samantapaisiin sovelluksiin. Internetin hakukoneiden avustuksella saadaan hieman yleistä kuvaa muista vastaavista sovelluksista, niin toiminnoiltaan kuin käyttöliittymiltään. Alustava mielikuva kehitettävästä sovelluksesta on epäselvä, koska suurin osa vastaavien sovelluksien kuvankaappauksista ja esittelyistä on suppeita. Esitutkimuksessa saatuja tietoja käytetään hyväksi seuraavassa vaiheessa. Määrittely aloitetaan vaatimusanalyysillä, joka sisältää toimeksiantajan vaatimukset ja toiveet sovelluksen toiminnasta. Vaatimusanalyysin pohjalta tehdään ohjelmistosuunnitelma, jossa selvennetään sovelluksen toimintaa, ulkoasua ja

tietokantaa. Tietokannan suunnittelussa käytetään käsitteellisen mallintamisen ER-mallia ja käyttöliittymä toteutetaan GUIDe-menetelmällä. Toimeksiantajan hyväksyessä suunnitelmat, toteutusvaihe voidaan aloittaa.

Sovelluksen toteutuksessa käytetään ensisijaisesti toimeksiantajan lisensoituja ohjelmia tai ilmaisia ohjelmistoja kuten PuTTY-ssh-asiakasohjelmaa ja GIMP-kuvankäsittelyohjelmaa. Sovellus toteutetaan toimeksiantajan tiloissa ja tämän työvälineillä. Käytettävät työvälineet käydään tarkemmin läpi luvussa 1.3. Laiterekisterisovelluksen ensimmäisten versioiden testauksessa ilmenevät virhetilanteet kirjataan ylös ja korjataan, korjausten jälkeen testaus voidaan aloittaa uudelleen. Testauksen jälkeen sovellus luovutetaan toimeksiantajalle, joka testaa sovelluksen, antaa palautetta ja päättää, täyttääkö sovellus vaaditut määritykset. Toimeksiantajan palautteen mukaan sovellus joko viimeistellään valmiiksi sovellukseksi käyttöönottovaihetta varten, tai tehdään tarvittaessa toimeksiantajan haluamat muutokset ja palataan toteutus- ja testausvaiheisiin.

Laiterekisterisovelluksen työvaiheet ja niiden tuotokset ovat seuraavat:

- Esitutkimus ja määrittely
    - Vaatimusanalyysi
    - Ohjelmistosuunnitelma
- Suunnittelu
    - Tietokannan suunnittelu
    - Käyttöliittymän suunnittelu
- Toteutus
- Testaus
- Viimeistely
- Käyttöönotto.

1.3 Työvälineet

Työ toteutetaan pääasiassa toimeksiantajan tiloissa ja sovellus pyritään toteuttamaan ilman ylimääräisiä ohjelmistokustannuksia. Kaikilla työssä käytettävillä ohjelmistoilla ja välineillä on jo valmiiksi oikeudet Verkkoasema Oy:llä tai ne ovat ilmaisia ja vapaasti käytettävissä.

Työssä käytetään seuraavia välineitä ja ohjelmistoja:

- EditPlus-tekstieditoria ohjelmointi- ja kuvauskielten kirjoittamiseen
- Eclipse-ohjemointiympäristöä ohjelmointi- ja kuvauskielten kirjoittamiseen
- Apache-palvelinohjelmisto
- MySQL-tietokantaa tietokannan hallintaan
- Adobe Photoshop CS2-kuvankäsittelyohjelmaa käyttöliittymän grafiikoiden tuottamiseen
- GIMP-kuvankäsittelyohjelmaa käyttöliittymän grafiikoiden tuottamiseen
- Microsoft Word-tekstinkäsittelyohjelmaa raportointiin
- Microsoft Access-relaatiotietokantaa, tietokannan suunnitteluun
- Mozilla Firefox-selainta, käyttöliittymän ja toiminnallisuuksien testaukseen
- Internet Explorer 6 ja Internet Explorer 7-selaimia, käyttöliittymän ja toiminnallisuuksien testaukseen

1.4 Konstruktiivinen tutkimusote

Opinnäytetyössä käytetään konstruktiivista tutkimusotetta. Konstruktiivinen tutkimusote on innovatiivisia konstruktioita tuottava metodologia, jolla pyritään ratkaisemaan aitoja reaalimaailman ongelmia ja tätä kautta tuottamaan kontribuutioita sille tieteenalalle, jossa sitä sovelletaan. Tiivis vuoropuhelu käytännön ja teorian välillä sekä tutkijan suorittamien interventioiden käyttö tutkimusmetodina ovat konstruktiiviselle tutkimusotteelle luonteenomaisia piirteitä. Konstruktiivinen tutkimusote on yksi case-tutkimuksen muoto. (Metodix 2007)

1.5 Keskeiset käsitteet

Apache — Apache HTTP Server on avoimeen lähdekoodiin perustuva HTTP-palvelinohjelma (Wikipedia 2007a).

CSS — CSS eli Cascading Style Sheets on erityisesti WWW-dokumenteille kehitetty tyyliohjeiden laji. CSS on kaskadinen tyyliohjejärjestelmä, eli

dokumentille voi määritellä useita tyyliohjeita, jotka yhdistetään tietyllä tavalla yhdeksi säännöstöksi (Wikipedia 2007b).

Eclipse — Eclipse on ohjelmointiympäristö, joka tukee esimerkiksi Java-, C/C++- ja PHP-ohjelmointikieliä (Wikipedia 2007c).

Eväste — Eväste on WWW-palvelimen ja WWW-selaimen välille määritelty tapa säilyttää tietoja. Keksien avulla voidaan muun muassa säilyttää tietoja käyttäjän kirjautumistiedoista. Eväste tunnetaan yleisesti nimillä cookie eli keksi, pipari tai eväste. (Heinisuo 2004, 150.)

Foreign key — Ks. viiteavain

GIMP — GIMP (The GNU Image Manipulation Program) on avoimeen lähdekoodiin perustuva monipuolinen kuvankäsittelyohjelma (Wikipedia 2007d).

GNU GPL — GNU General Public License(*GNU yleinen lisenssi*) on vapaa ohjelmistolisenssi, jonka alun perin loi Richard Stallman GNU-projektin tarpeisiin vuonna 1989. Lisenssistä käytetään yleisesti lyhenteitä GNU GPL tai GPL. (Wikipedia 2009b)

HTML — HTML eli Hypertext Markup Language on avoimesti standardoitu kuvauskieli, jolla voidaan kuvata hyperlinkkejä sisältävää tekstiä eli hypertekstiä. HTML tunnetaan erityisesti kielenä, josta webbisivut rakentuvat (Wikipedia 2007e).

Internet Explorer — Windows Internet Explorer (WIE) on Microsoftin www-selain, joka tulee jokaisen Windows-käyttöjärjestelmän mukana ja on maailman käytetyin selain (Wikipedia 2007f).

MD5 — MD5 on niin kutsuttu message-digest-algoritmi, jota käytetään mm. kryptografiassa. MD5-algoritmi tuottaa 128-bittisen tiivisteen, joka

tyypillisesti esitetään 32-merkkisenä heksakoodatussa muodossaan. (Wikipedia 2007g.)

Microsoft Access

Microsoft Access on tietokantojen hallintaohjelma (Wikipedia 2007h).

Mozilla Firefox

Mozilla Firefox on ilmainen avoimen lähdekoodin www-selain (Wikipedia 2007i).

MySQL MySQL on monipuolinen, joustava ja suorituskykyinen relaatiotietokanta (Heinisuo 2004, 34).

Perusavain Perusavain eli primary key on yksilöllinen avain (unique key), jonka avulla löydetään yksi rivi tietokantataulusta (MySQL Control Center 2009a).

PHP PHP (lyhenne sanoista PHP: Hypertext Preprocessor) on Perlin kaltainen ohjelmointikieli, jota käytetään erityisesti Web-palvelinympäristöissä dynaamisten web-sivujen luonnissa (Wikipedia 2007j).

Primary key Ks. perusavain

PuTTY PuTTY on suosittu vapaa avoimen lähdekoodin telnet- ja ssh-asiakasohjelma ja pääte-emulaattori. Sitä käytetään etenkin Windows-käyttöjärjestelmissä Unix-koneiden merkkipohjaiseen etäkäyttöön (Wikipedia 2007k).

Tietokanta Tietokanta(Database, DB) on loogisesti yhteenkuuluvien, tallennettujen tietojen joukko, jota voidaan helposti käsitellä tietokantakielellä kuten SQL. Tietokannan tietoja hallinnoi erityinen ohjelmisto, tietokannan hallintajärjestelmä(Database Management System). Tunnettuja hallintajärjestelmiä ovat muun muassa Microsoft SQL Server, MySQL ja Access. (Hovi & Huotari & Lahdenmäki 2005, 4.)

Viiteavain — Viiteavain eli foreign key. Viiteavaimia tarvitaan varsinkin useampia tauluja sisältävissä tietokannoissa. Viiteavaimet eivät yleensä ole yksilöllisiä ja avaimet koostuvat usein useasta taulun sarakkeesta. (MySQL Control Center 2009b)

Viite-eheys — Taulun kentistä voidaan viitata jonkin toisen taulun perusavaimeen tai toissijaiseen avaimeen. Tällaista yhteyttä kutsutaan viite-eheydeksi ja kyseinen kenttä määritellään yleensä myös avaimeksi, jota kutsutaan viiteavaimeksi. (Ekonoja & Lahtonen & Mäntylä 2009)

2 LAITEREKISTERISOVELLUKSEN SUUNNITTELU

2.1 Käyttöliittymä ja ulkoasu

Käyttöliittymän suunnittelussa käytetään Nielsenin heuristisia sääntöjä, joiden avulla parannetaan sovelluksen käytettävyyttä. Käytettävyys (engl. *usability*) on apuvälineen tai muun valmistetun esineen, palvelun tai ympäristön helppokäyttöisyyttä tietyn tavoitteen saavuttamiseksi (Wikipedia 2009). Käytettävyydeltään hyvä sovellus on siis käyttöliittymältään looginen, käyttäjä löytää tarvittavat toiminnot ja tiedot helposti omilta paikoiltaan ilman suurempaa vaivaa. Virhetilanteiden ja toimintojen raportoinnilla lisätään vuorovaikutusta sovelluksen ja käyttäjän välille, jolloin käyttäjä on selvillä sovelluksen toiminnoista ja vaiheista.

ISO 9241-11 -standardi määrittelee käytettävyyden seuraavalla tavalla: "Se vaikuttavuus, tehokkuus ja tyytyväisyys, jolla tietyt määritellyt käyttäjät saavuttavat määritellyt tavoitteet tietyssä ympäristössä" (Wikipedia 2009). Hyvällä käytettävyydellä kasvatetaan siis työtehoa kaikilla mittareilla: se luo sovellukselle arvokkuutta selkeänä ohjelmistona, se lisää työtehokkuutta mielekkäästä käyttöliittymästä ja toiminnoista sekä jättää käyttäjälle onnistumisen tunteen, jolloin kynnys sovelluksen uudelleenkäytöstä laskee.

Nielsenin heuristiset säännöt listattuna:

1. Käytä yksinkertaista ja luonnollista dialogia
2. Käytä käyttäjien omaa kieltä
3. Minimoi käyttäjien muistikuorma
4. Tee käyttöliittymästä kauttaaltaan yhdenmukainen
5. Anna käyttäjälle palautetta toiminnoista
6. Anna selkeä poistumistapa eri tilanteista ja toiminnoista
7. Anna käyttäjille mahdollisuus käyttää oikopolkuja
8. Anna virhetilanteissa selkeät virheilmoitukset
9. Vältä virhetilanteita
10. Anna riittävä ja selkeä apu ja dokumentaatio

2.2 Tietokanta

2.2.1 Tietokannan valinta

Nykyiset SQL-pohjaiset tietokantojen hallintajärjestelmät ovat valtaosin relaatiotietokantoja, koska ne ovat helpompia käyttää ja muuttaa kuin perinteisemmät tietokantamallit. Relaatiotietokannat perustuvat IBM:n tutkijan E. F. Coddin vuonna 1980 julkaisemaan relaatiomalliin, joka määrittelee relaatiotietokannan teoreettisen pohjan. Coddin määrittelemään relaatiomalliin perustuvat relaatiotietokantatuotteet ovat syrjäyttäneet aiemmin käytetyt hierarkkiset ja verkkomalliset tietokantatyypit. SQL onkin standardoitunut lähes ainoaksi tietokantakieleksi. (Hovi ym. 2005, 5-8.)

MySQL-tietokanta on hyvin suosittu ja tehokas tietokanta web-palveluiden käytössä, se on saatavissa vapaalla GNU GPL-lisenssillä. Toimeksiantajan LAMP-ympäristö suosii MySQL-tietokantaa, joten laiterekisterisovelluksen tietokantavalinta oli helppo.

2.2.2 Tietokannan suunnittelu

Tietokannan suunnittelu aloitetaan käsiteanalyysillä, jonka tavoitteena on määrittää ja kuvata havainnollisella kaaviolla tietokantaan talletettavia tietoja. Käsiteanalyysissä kuvataan sitä reaalimaailmasta rajattua osaa eli kohdealuetta, jota on tarkoitus kuvata tietokannassa. Lopputuloksena syntyy käsitemalli, joka kuvaa kohdealuetta ja määrittelee pohjan tietokannan fyysiselle rakenteelle. Liitteessä 3. käsitemalli kuvataan graafisena käsitekaaviona UML-kuvauskielellä. (Hovi ym. 2005, 32.)

Tietokannan suunnitteluvaiheessa käytetään pääasiassa Microsoft Access-tietokantaohjelmaa, jolla voidaan luoda sovelluksen tietokannan rakenne. Rakenteen suunnittelussa täytyy huomioida laiterekisteritiedoissa käytettävien kenttien nimeämisperusteet ja niiden tietotyypit. Käytettävät tietotyypit tulevat olemaan pääasiassa numeerista tietoa, tekstiä, näiden kahden sekoituksia ja tosi/epätosi-arvoja.

Laiterekisterisovelluksen tietokannan suunnittelun yhteydessä tietokannan taulut normalisoidaan, jolla tietokannan rakenne saadaan parhaiten tukemaan tietojen ehjää tallennusta ja tiedon tehokasta saatavuutta. Normalisoinnilla vähennetään tallennettavan tiedon monistusta, jolloin sama tieto tallennetaan useampaan eri tietokannan tauluun.

Normalisoinnin tuloksena tietokanta jakautuu useampaan pienempään tauluun, joiden välille muodostuu relaatioita.

Taulun kentistä voidaan viitata jonkin toisen taulun perusavaimeen tai toissijaiseen avaimeen. Tällaista yhteyttä kutsutaan viite-eheydeksi ja kyseinen kenttä määritellään yleensä myös avaimeksi, jota kutsutaan viiteavaimeksi. Viite-eheys määrää, että jokaista viittaavassa taulussa esiintyvää viiteavaimen arvoa pitää vastata sama perusavaimen arvo viittauksen kohteena olevassa taulussa. Yritettäessä syöttää Työntekijä-tauluun arvoja, joita vastaavia ei ole Osasto-taulussa, viite-eheys pakottaa syöttämään jotain kelvollista ennen kuin syöttö voidaan hyväksyä. Sama tapahtuu, jos Projekti-taulun projektipäällikkö-kenttään yritetään syöttää arvoa jota ei löydy Työntekijä-taulun työntekijäID-kentästä. (Ekonoja & Lahtonen & Mäntylä 2009)

Esimerkkinä viite-eheydestä on devices-taulun perusavain device_id, jolla viitataan device_links-taulun viiteavaimeen device1_id:hen. Devices-taulussa määritellään laitteen tiedot ja device_links-taulussa laitteen viitteet toisiin laitteisiin. Määrittämällä laitteelle uniikki tunnistenumero, esimerkiksi tuotekoodi, sitä voidaan linkittää ristiin toisten laitteiden ja ohjelmistojen kanssa. Viitteet voi tässä tapauksessa olla esimerkiksi ohjelmistolisenssin kuulumisen tiettyyn pöytäkoneeseen.

Suunnittelun lopputuloksena Microsoft Access-tietokantaohjelmasta saadaan tietokannan rakenne selkeässä graafisessa muodossa. Tietokannan rakenne taulukoissa 1-5.

Taulukko 1. Laiterekisteri-tietokannan devices-taulu

devices		
device_id	bigint(20)	Primary key, auto_increment
label	varchar(255)	not null
model	varchar(255)	not null
type_id	tinyint(10)	foreign key, not null
serial	varchar(255)	
identifier_id	varchar(255)	
location	varchar(255)	
description	varchar(255)	

purchase_date	varchar(255)	
purchase_info	varchar(255)	
leasing_info	varchar(255)	

Taulukko 2. Laiterekisteri-tietokannan device_links-taulu

device_links		
link_id	bigint(20)	Primary key, auto_increment
device1_id	bigint(20)	foreign key, not null
device2_id	bigint(20)	foreign key, not null

Taulukko 3. Laiterekisteri-tietokannan users-taulu

users		
user_id	bigint(20)	Primary key, auto_increment
username	varchar(255)	not null
password	varchar(255)	not null
user_right_id	tinyint(10)	foreign key, not null, default ’1’

Taulukko 4. Laiterekisteri-tietokannan user_rights-taulu

user_rights		
right_id	tinyint(10)	Primary key, not null
name	varchar(255)	not null

Taulukko 5. Laiterekisteri-tietokannan device_types-taulu

device_types		
type_id	tinyint(10)	Primary key, not null
name	varchar(255)	not null

3 TOTEUTUS JA TESTAUS

Sovelluksen tekninen toteutus aloitetaan luomalla sovelluksen tietokanta. Tietokanta on sovelluksen ydin, jonka ympärille toiminnallisuudet kehitetään. Rekisterin tietokannan taulut tulee luoda ylläpidon ja jatkokehityksen kannalta loogisesti sekä nimetä kuvaavasti. Kun tietokantarakenne on luotu, ohjelmoidaan sovelluksen tietokantafunktiot, jotka mahdollistavat loppukäyttöpuolen laitteiden ja käyttäjien hallinnan.

Tietokantafunktioiden ohjelmointi on isossa osassa projektin kokonaiskuvaa, koska funktiot yhdessä tietokantataulujen kanssa muodostavat itse sovelluksen ja loppukäyttöpuoli toimii pelkkänä käyttöliittymänä käyttäjälle. Käyttöliittymä on helpommin muokattavissa, toisin kuin sovelluksen toimintaan liittyvät funktiot. Tämän takia tietokantafunktiot testataan huolella toteutuksen yhteydessä, jotta sovellukseen liittyvät virheet ja ongelmat saadaan minimoitua jo ohjelmointivaiheessa.

Ohjelmoinnin ja funktioiden testauksen jälkeen tulee toteuttaa sovellukselle käytettävyydeltään hyvä käyttöliittymä. Hyvään käyttöliittymään liittyy Jakob Nielsenin(2000) mukaan mm. kieleltään selkeä, yhdenmukainen ja käyttäjää ohjeistavat toiminnot. Käyttöliittymään tehdään omat osiot omille toiminnoilleen. Osiot jaetaan sovelluksen etusivuun, laitehakuun, laitteen lisäykseen ja käyttäjähallintaan. Etusivulle lisätään sovelluksen esittely, lyhyet käyttöohjeet sekä yhteystietoja mahdollisia virhetilanteita varten.

Tietokannan hakumoottorin ja sen listauksen käyttöliittymän suunnittelussa tulee ottaa huomioon haussa käytettävät kentät, mahdolliset sovelluksen virheilmoitukset ja listausnäkymän muutokset.

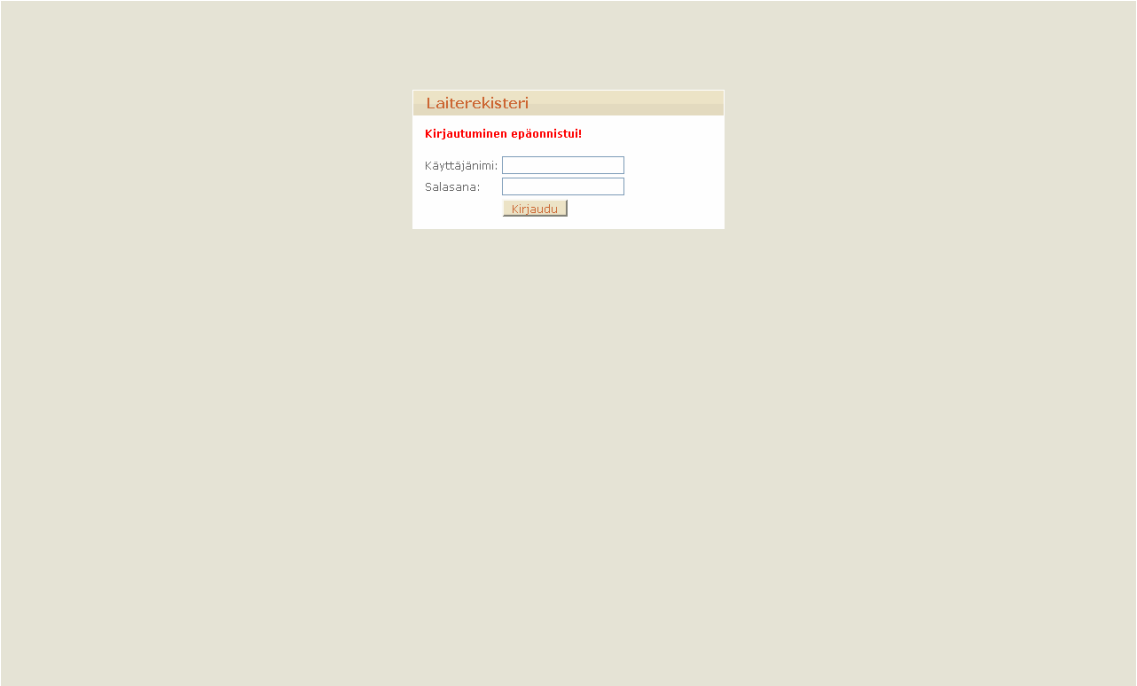
3.1 Sovelluksen toteutus

3.1.1 Kirjautuminen

Sovelluksen käyttöä suojataan käyttäjätunnuksilla ja kirjautumisella. Kirjautumista ja käyttäjän oikeuksia ohjataan käyttäjätileillä, jotka jaetaan kahteen ryhmään: pääkäyttäjät ja loppukäyttäjät. Pääkäyttäjän käyttäjätunnuksilla voidaan lisätä, muokata, poistaa ja selata laitetietoja sekä hallita käyttäjätilejä. Loppukäyttäjällä kirjautuminen mahdollistaa vain laitetietojen selauksen.

Sovelluksen pääkäyttäjä hallinnoi käyttäjätilejä ja uusien tilien luonti tapahtuu pääkäyttäjän kautta. Käyttäjätunnusten luotua käyttäjät voivat kirjautua sovellukseen. Kirjauduttaessa sovellukseen, tietokannasta haetaan käyttäjätunnusta vastaavat tiedot ja verrataan niitä käyttäjän antamiin tietoihin. Mikäli tiedot vastaavat toisiaan, käyttäjä ohjataan sovellukseen. Kirjautumisen virhetilanteissa käyttäjälle ilmoitetaan ongelmasta ja ohjataan takaisin kirjautumislomakkeelle. Onnistuneessa kirjautumisessa käyttäjän kirjautumistiedot tallennetaan selaimeen tunnin mittaiseen evästeeseen, jolloin käyttäjän kirjautumistiedot tunnistetaan automaattisesti seuraavalla käyttökerralla. Kirjautumistietoihin kuuluu käyttäjätunnus, käyttäjäryhmä ja salasana MD5-kryptattuna.

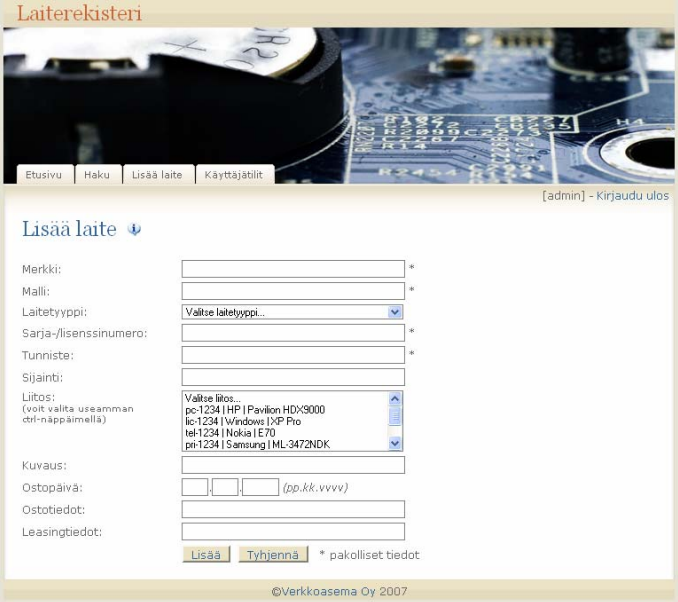
Käyttäjän kirjauduttua ulos sovelluksesta eväste poistetaan selaimen muistista, muissa tapauksissa eväste säilytetään selaimen muistissa kolme tuntia, jonka jälkeen se poistetaan ja sovellukseen automaattinen kirjautuminen poistuu käytöstä. Aikarajoituksella vältetään sovelluksen väärinkäyttöä ja parannetaan sovelluksen käytettävyyttä. Käytettävyyden parantamisella tarkoitetaan tässä tapauksessa automaattista kirjautumista sovellukseen, joka vähentää sovellukseen käytettävää aikaa ja vaivaa. Aikarajoituksella minimoidaan myös ulkopuolisia uhkia tapauksissa, joissa sovellus unohdetaan tietokoneelle auki, mutta kirjautumistiedot ovat vanhentuneet. Tällöin sovellus on jätetty auki kirjautuneena käyttäjänä, mutta jokainen tehty sivupäivitys tarkastaa käyttäjätiedot tietokannasta ja selaimen evästeistä. Kuvassa 1. on sovelluksen kirjautumisikkuna ja epäonnistuneesta kirjautumisesta tulostettu virheilmoitus.



Kuva 1. Kirjautumisikkuna ja kirjautumisen virheilmoitus

3.1.2 Laitteen lisäys

Laiterekisteriin voidaan lisätä laitteita monipuolisesti, koska kentät ovat yleisluontoisia. Laite lisätään syöttämällä lomakkeeseen tiedot, jotka tallennetaan tietokantaan. Laitteille syötettävät pakolliset tiedot ovat merkki, malli, laitetyyppi, sarja-/lisenssinumero ja tunniste. Pakolliset laitetiedot on syötettävä, jotta laitetietojen haku ja selaus olisi mahdollista. Vapaaehtoisia laitetietoja ovat sijainti, liitos, kuvaus, ostopäivä, ostotiedot ja leasingtiedot. Laitteen lisäyslomake kuvassa 2.



Kuva 2. Laitteen lisäyslomake

3.1.3 Laitetietojen haku

Laiterekisterisovelluksen hakutoiminto toteutetaan HTML-lomakkeella, joka sisältää hakuvaihtoehtoina vapaan sanahaun tai kategorian valinnan. Käyttäjä voi syöttää sanahakuun vapaamuotoisen hakusanan, jota verrataan tietokannan tietoihin. Vapaa sanahaku vertaa annettua hakusanaa jokaiseen tietokannan tietueen tietokenttään ja tulostaa vastaavat hakusanaa vastaavat tietueet. Haettaviin tietokenttiin kuuluu: merkki, malli, laitetyyppi, sarja-/lisenssinumero, tunniste, sijainti, kuvaus, ostotiedot ja leasingtiedot. Hakua vastaavat tietokannan hakutulokset tulostetaan hakulomakkeen alapuolelle HTML-taulukkoon niin, että jokaisesta hakutuloksesta tulostetaan yksi rivi, joka sisältää laitteen tunnisteen, merkin ja mallin. Ylläpitokäyttäjille listataan jokaisen laitteen perään muokkaa ja poista-toiminnot, jotka toimivat linkkeinä laitteiden ylläpitotietoihin.

3.1.4 Laitetietojen muokkaus

Yksittäisiä laitetietoja voidaan muokata hakutoiminnon kautta. Laitteita haetaan halutulla hakusanalla tai kategorialla, jolloin hakutulokset tulostetaan taulukkoon. Taulukosta valitaan kyseisen laitteen perästä kohdalta muokkaa-toiminto, joka avaa

erillisen laitteen muokkausnäkymän. Muokkausnäkymä sisältää samankaltaisen lomakkeen kuin laitteen lisäyksessäkin, erona vain tallenna-painike. Lomakkeen kenttiä voidaan muokata halutulla tavalla, jonka jälkeen tiedot tallennetaan tietokantaan tallenna-painikkeella. Virhetilanteissa käyttäjälle ilmoitetaan virheestä lomakkeen yläpuolelle näkyvällä punaisella varoitustekstillä. Virheilmoituksien punainen väri toimii varoituksissa hyvin, koska punainen symbolisoi vaaraa tai varoitusta.

3.1.5 Käyttäjätilit

Käyttäjätilit koostuvat kahdesta käyttäjätasosta: ylläpitäjä ja loppukäyttäjä. Ylläpitäjät vastaavat sovelluksen käyttäjähallinnasta, jolloin sillä on oikeus luoda, poistaa ja muokata käyttäjätilejä. Käyttäjätunnuksien luominen tapahtuu sovelluksen ylläpitäjän kautta, joka luo käyttäjille omat käyttäjätunnukset väärinkäytöksien välttämiseksi. Omilla käyttäjätunnuksilla tehdyt laitetietojen muutokset tallentuvat järjestelmään ja tällöin yksittäisen laitteen tietomuutoksia on mahdollista seurata järjestelmästä. Käyttäjät ovat siis tietoisia kuka laitetta on viimeksi päivittänyt ja ongelmatapauksissa voidaan kääntyä oikean käyttäjän puoleen.

3.2 Testaus

Sovelluksen testauksen tarkoituksena on löytää ohjelmistossa olevat virheet. Virheitä löytyy toteutuksen huolellisuudesta riippumatta käytännössä kaikista ohjelmistoista. Esimerkiksi yritykselle on erittäin tärkeää löytää virheet (ja korjata ne) itse sen sijaan, että asiakas löytää ne ja kertoo sitten huonoista kokemuksistaan muille. (Wikipedia 2009c)

Testaus on siis todella tärkeä osa ohjelmistokehitystä, mutta se ei kuitenkaan takaa laadukasta lopputulosta. Ohjelmiston korkeaa laatua tavoitellessa tulee kiinnittää huomiota heti suunnitteluvaiheessa asettamalla ohjelmiston vaatimukset tarpeeksi korkealle.

Testaajan tehtävä on löytää sovelluksesta mahdollisimman paljon virheitä ja raportoida ne ohjelmistokehittäjille. Virheiden löytäminen sovelluksesta on hankalampaa mitä

luulisi ja testaajalta vaaditaan kärsivällisyyttä sekä rautaisia hermoja. Testaajan ja ohjelmistokehittäjän välinen kommunikointi on tärkeää, jotta raportoidut virheet ymmärretään kokonaisuudessaan ja ohjelmistokehittäjän tekemät korjaukset kohdistuvat oikeaan paikkaan. Vaihtoehdoista helpoin on toimia itse niin testaajana kuin ohjelmistokehittäjänä, jolloin tiedetään tarkkaan virheellisien toimintojen ilmeneminen ja sovelluksen korjaamien on huomattavasti helpompaa. Paras vaihtoehto on ulkoistaa testaus eri henkilölle tai osastolle, jolloin säilytetään objektiivinen näkökanta sovellukseen. Objektiivista näkökantaa tarvitaan käsittelemään sovellusta niin kuin loppukäyttäjä tulisi sitä käsittelemään. Testaajan tulee huomioida, että käyttäjät voivat omata heikot atk-taidot, jolloin yksinkertaistenkin toimintojen suorittaminen voi olla hankalaa. Testaajan tulisi ottaa huomioon koko käyttäjäkanta ja testata sovellusta sen mukaan.

Käytännössä testaus aloitetaan sovelluksen ensimmäisestä versiosta, jonka virheet raportoidaan ja korjataan. Testaus- ja korjauskierroksia jatketaan kunnes uusia virheitä ei enää sovelluksesta löydy. Testausvaiheen jälkeen sovelluksesta voidaan halutessa julkaista niin sanottu beta-versio, jolla tarkoitetaan sovelluksen julkista testausversiota. Betatestauksessa sovellus julkaistaan isommalle käyttäjäkunnalle, jolloin viimeisten virheiden ilmenemisen todennäköisyys kasvaa. Betatestausta käytetään enimmäkseen suurissa ohjelmistoissa, joissa käyttäjäkunta on yleensä maailmanlaajuinen.

4 YHTEENVETO

Opinnäytetyön tavoitteena oli suunnitella ja toteuttaa helppokäyttöinen ja kevyt selaimessa toimiva laiterekisterisovellus Verkkoasema Oy:lle. Sovelluksen tuli toimia toimeksiantajan palvelimella, jonne sovelluksen tiedot myös talletettiin. Suunnitteluvaiheessa toimeksiantaja määritteli sovelluksen vaatimukset ja laitetietoihin tarvittavat kentät. Työn eri toteutus- ja testausvaiheissa kävimme vuoropuhelua toimeksiantajan kanssa ja mietimme yhdessä mahdollisia jatkokehitykseen liittyviä toimintoja. Sovellus haluttiin kuitenkin pitää vielä suhteellisen yksinkertaisena, joten isommille jatkotoimenpiteille ei tässä vaiheessa ollut tarvetta.

Käsitys toimeksiantajan visioimasta sovelluksesta oli epäselvä, joten sovelluksen suunnittelu oli aluksi todella vaikeaa. Vaihe vaiheelta sovelluksen toiminta alkoi hahmottua ja kiinnostus työhön kasvaa. Toimeksiantajan rooli olikin suunnitteluvaiheessa todella tärkeä ja se toimi hyvänä ponnistuslautana sovelluksen kehitykseen. Sovellusta kehittäessä työstimme Verkkoasema Oy:n muita projekteja, joista saatiin apuja ohjelmointiin ja tietokannan hallintaan. Nämä oheisprojektit tukivat laiterekisterisovelluksen kehitystä, vaikka söivätkin resursseja ja aikaa opinnäytetyöltä.

Työ vei huomattavasti enemmän aikaa kuin odotettiin, varsinkin suunnittelu. Ohjelmointiin varattiin aikaa reilusti, mutta todennäköisesti suunnitellut työtunnit eivät pitäneet paikkaansa. Suunnittelua laiminlyötiin ja se kostautui jälkeenpäin ohjelmoitaessa sovelluksen toimintoja. Jotkut toiminnot ja painikkeet olisi voitu sijoittaa käytettävyyden kannalta eri paikkoihin, mutta ovat kyllä löydettävissä nykyisiltäkin paikoilta.

Työn tavoite siis saavutettiin: sovelluksesta saatiin kevyt ja selainpohjainen sovellus, jonka ylläpitäminen on helppoa. Ylläpitosovelluksen lomakkeet noudattavat samaa kaavaa, joka on käytettävyyden kannalta hyvä asia. Parantamisen varaa käyttöliittymään toki jäi, mutta täytyy olla tyytyväinen työn tuloksiin tämän hetkisellä tietotaidolla ja osaamisella. Yleisfiilis työstä on positiivinen ja sovelluksen kehityksen mukana saatu kokemus, niin hyvässä kuin pahassa, on todella arvokasta.

Jatkokehityksen osalta sovelluksessa käytettävät funktiot olisi voitu nimetä loogisemmin ja käyttää enemmän olio-ohjelmointia, jolloin toimintojen lisäys olisi

huomattavasti vaivattomampaa. Tietokannan rakenne on hyvä, joka on helposti laajennettavissa tarpeellisilla tauluilla.

LÄHTEET

Painetut

Heinisuo R. 2004. PHP ja MySQL: Tietokantapohjaiset verkkopalvelut. 3. uudistettu painos. Talentum, Helsinki.

Hovi, A. & Huotari, J. & Lahdenmäki, T. 2005. Tietokantojen suunnittelu & indeksointi. Docendo Finland Oy, Jyväskylä.

Nielsen J. 2000. WWW-suunnittelu. Oy Edita Ab.

Painamattomat

Ekonoja, Antti & Lahtonen, Tommi & Mäntylä, Jukka 2009. Relaatiotietokantojen peruskäsitteet. Luettu 8.12.2009. <http://appro.mit.jyu.fi/doc/tiedonhallinta/tietokannat/index2.html>

Metodix 2007. Konstruktiivinen tutkimusote. Luettu 27.11.2007. <http://www.metodix.com/fi/sisallys/04_virtuaalikirjasto/dokumentit/aineistot/konstruktiivinentutkimusote>

MySQL Control Center 2009a. Perusavain. Luettu 8.12.2009. <http://edu.phkk.fi/Opiskelu/Internet-ohjelmointi/PHP_ja_MySQL/MySQLCC.htm>

MySQL Control Center 2009b. Viiteavain. Luettu 8.12.2009. <http://edu.phkk.fi/Opiskelu/Internet-ohjelmointi/PHP_ja_MySQL/MySQLCC.htm>

Ratol 2009. Foreign Key. Luettu 7.12.2009. <http://www.ratol.fi/opensource/mysql/avaimet.htm#fk>

Wikipedia 2007a. Apache palvelinohjelma. Luettu 19.11.2007. <http://fi.wikipedia.org/wiki/Apache_HTTP_Server>

Wikipedia 2007b. CSS. Luettu 19.11.2007. <http://fi.wikipedia.org/wiki/Css>

Wikipedia 2007c. Eclipse. Luettu 19.11.2007. <http://fi.wikipedia.org/wiki/Eclipse_%28IDE%29>

Wikipedia 2007d. GIMP. Luettu 19.11.2007. <http://fi.wikipedia.org/wiki/Gimp>

Wikipedia 2007e. HTML. Luettu 19.11.2007. <http://fi.wikipedia.org/wiki/Html>

Wikipedia 2007f. Internet Explorer. Luettu 19.11.2007. <http://fi.wikipedia.org/wiki/Internet_explorer>

Wikipedia 2007g. MD5. Luettu 21.11.2007.
<http://fi.wikipedia.org/wiki/MD5>

Wikipedia 2007h. Microsoft Access. Luettu 19.11.2007.
<http://fi.wikipedia.org/wiki/Microsoft_Access>

Wikipedia 2007i. Firefox. Luettu 19.11.2007. <http://fi.wikipedia.org/wiki/Firefox>

Wikipedia 2007j. PHP. Luettu 19.11.2007. <http://fi.wikipedia.org/wiki/Php>

Wikipedia 2007k. PuTTY. Luettu 19.11.2007. <http://fi.wikipedia.org/wiki/Putty>

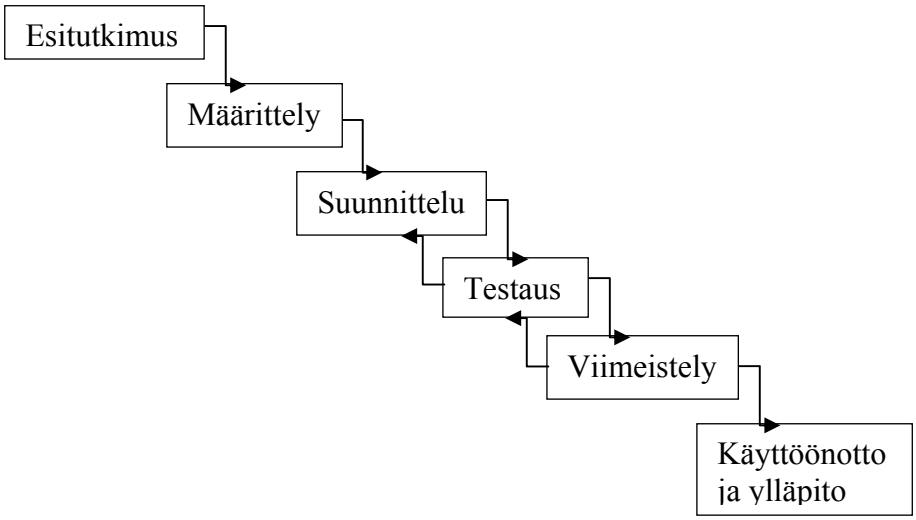
Wikipedia 2009a. Käytettävyys. Luettu 7.12.2009.
<http://fi.wikipedia.org/wiki/K%C3%A4ytett%C3%A4vyys>

Wikipedia 2009b. GNU GPL. Luettu 8.12.2009.
<http://fi.wikipedia.org/wiki/GNU_GPL>

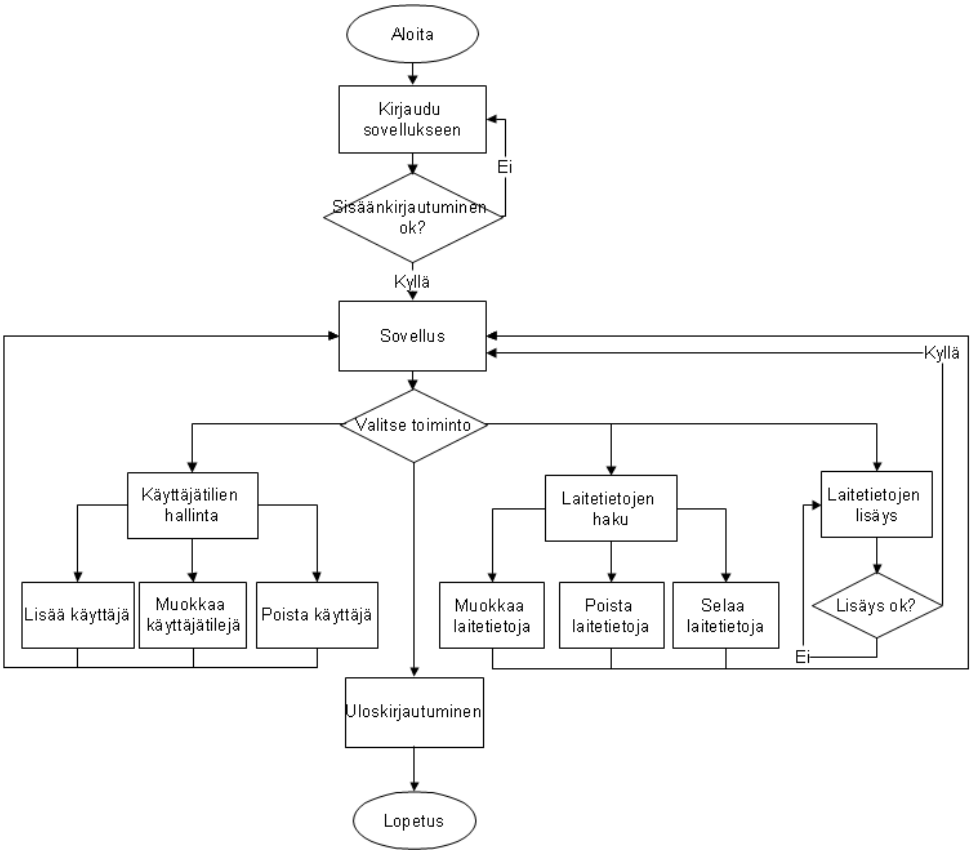
Wikipedia 2009c. Ohjelmistotuotanto. Luettu 10.12.2009.
<http://fi.wikipedia.org/wiki/Ohjelmistotuotanto>

LIITTEET

Vesiputousmalli Liite 1



Laiterekisterisovelluksen vuokaavio Liite 2



Laiterekisterisovelluksen käsitekaavio Liite 3