JAVA-KÄYTTÖLIITTYMÄ ETÄHALLINTAAN

LAHDEN AMMATTIKORKEAKOULU
Tekniikan ala
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka
Opinnäytetyö
Kevät 2010
Henri Eskelinen

Lahden ammattikorkeakoulu
Tietotekniikan koulutusohjelma

ESKELINEN, HENRI: Java-käyttöliittymä etähallintaan

Ohjelmistotekniikan opinnäytetyö, 40 sivua

Kevät 2010

TIIVISTELMÄ

Tämä työ tehtiin Teknoware Oy:lle tarkoituksena tarjota lisenssivapaa etähallintasovellus turvavalojärjestelmälle. Turvavalojärjestelmän tarkoituksena on opastaa rakennuksessa olevat henkilöt uloskäynneille hätätilanteen sattuessa. Turvavalojärjestelmän kuntoa tulee tarkkailla testaamalla väliajoin sekä keskusten että valaisimien kuntoa. Näiden testien tulokset tulee tallentaa, jotta niistä voidaan laatia lakiasetusten mukaiset kirjanpidot.

Etähallintajärjestelmän graafinen käyttöliittymä toteutettiin Java-ohjelmointikielellä käyttäen sen Swing-kirjastoluokkaa. Swing-komponentit ovat käyttöliittymäriippumattomia, kuten sitä edeltäneen AWT-kirjaston komponentit. Swingin komponentit ovat kevyempiä käyttää, sekä niillä rakennetut sovellukset ovat ulkonäöltään samanlaisia, alla olevasta käyttöliittymästä riippumatta.

Graafiseen käyttöliittymään tahdottiin hahmottamista selventäviä puunäkymiä, joten työtä varten tutkittiin JTree-luokan ominaisuuksia ja käyttöä. JTree-luokka tarjoaa tavan esittää hierarkkista tietoa selkeässä muodossa. Sen komponentit eivät itsessään sisällä tietoa, vaan saavat esitettävän tiedon kysymällä sitä niiden tietomalleista.

Etähallintasovellus käyttää tietokantaa, joka toimii rajapintana sovelluksen moduulien välillä. Tietokantaan luotiin tauluja, jotka sisältävät valaisinjärjestelmään asennettujen keskusten ja valaisimien sekä graafisen käyttöliittymän vaatimat tiedot. Tietokantayhteyden muodostamiseen graafinen käyttöliittymä käyttään JDBC-ohjelmointirajapinta.

Työn tuloksena valmistui Java-ohjelmointikielellä tehdyn graafisen käyttöliittymäsovelluksen beta-versio, joka toimii osana ohjelmakokonaisuutta. Tätä voidaan käyttää turvavalokeskuksien etähallintaan.

Avainsanat: graafinen käyttöliittymä, Swing, Java, turvavalaistus

Lahti University of Applied Sciences
Degree Programme in Information Technology

ESKELINEN, HENRI: Java graphical user interface for remote control

Bachelor's Thesis inSoftware Engineering, 40 pages

Spring 2010

ABSTRACT

This thesis was made for Teknoware Oy in order to provide licence-free remote control software for building emergency light systems. The main purpose of the emergency light systems is to guide people out from the building when an emergency situantion occurs. From time to time the system's central and luminaire component must be tested to make sure that it is running correcly. The results of those tests must be stored for documenting purposes required by the laws.

The graphical user interface component of the remote control software was coded using Java programming language. Just like in AWT, predecessor, the components are platform independent. Swing's components are considered lightweight when compared to AWT, and they look exactly the same no matter what the underlying platform is.

Tree views were used in the graphical user interface to provide a clarifying overview of the system. For that reason the usage and properties of JTree class were studied in the thesis. The class provides a way to display hierarchical data. A JTree component does not contain data but gets it by querying its data model.

The remote control software contains a database that works as an interface between the modules. The database includes several tables that hold on the centrals and luminaires of the emergency light systems. Furthermore, the tables also keep crucial data on the graphical user interface. The JDBC application programming interface is used to create the connection between the database and the graphical user interface.

As a result of this work, a beta version of a graphical user interface coded with the Java programming language was created. It serves as a part of remote control software that can be used to remote control the emergency light systems inside buildings.

Key words: graphical user interface, Swing, Java, emergency lights

SISÄLLYS

1 JOHDANTO

Teknoware Oy on vuonna 1972 perustettu yritys, joka valmistaa ja myy ajoneuvojen valaisimia ja valaistusjärjestelmiä sekä kiinteistöjen turvavalaisimia ja turvavalaistusjärjestelmiä. Yrityksen erikoisosaamista on elektroninen muuttaja, joka toimii muuntajana syöttöjännitteen ja valolähteen välillä. (Teknowaren henkilöstökäsikirja 2008, 2.)

Yritys sijaitsee Ilmarisentiellä Lahdessa. Sen käytössä on neljä kiinteistöä, joista uusin 3600m²:n tehdaslaajennus otettiin käyttöön elokuussa 2009. Yritys työllistää noin 190 henkilöä, ja sillä on yksiköitä myös Venäjällä ja Kiinassa. (Teknowaren henkilöstökäsikirja 2008, 2.)

Teknowaren tuotteita viedään yli 40 maahan, ja viennin osuus on noin 70 % sen liikevaihdosta. Asiakkaita ovat ajoneuvoteollisuuden johtavat bussivalmistajat, junavalmistajat sekä turvavalaisinvalmistajat. Yrityksen laatujärjestelmä on sertifioitu ISO 9001:n mukaisesti vuodesta 1993 lähtien ja ympäristöjärjestelmä ISO 14001:n mukaisesti vuodesta 2003. Yrityksen valaisimia on käytetty valaisemaan muun muassa maailman suurinta risteilyalusta, Oasis of the Seasia.
(Teknowaren henkilöstökäsikirja 2008, 2.)

Työn tavoitteena on saada aikaan toimiva graafinen käyttöliittymäsovellus hätävalokeskusten etähallintakäyttöä varten. Työssä suunnitellaan tietokanta, joka toimii sovelluksen rajapintana järjestelmän muihin komponentteihin. Ohjelmalla pyritään pääsemään eroon lisenssöidystä hallintaohjelmasta.

Työ päädyttiin tekemään Java-ohjelmointikielellä, koska kiireisen aikataulun vuoksi ei ollut aikaa opetella uutta ojelmointikieltä sekä sen vaatimia työkaluja. Javan työkalut olivat moderneja ja avoimen lähdekoodin ohjelmia.

Tässä työssä luvussa 2 käydään läpi turvavalaisinjärjestelmän toimintaympäristöä. Luku 3 käsittelee Javan Swing-kirjastoluokkaa ja luku 4 sen JTree-ohjelmointirajapintaa. Lisäksi tutkitaan JDBC-kirjastoja sekä JDBC-ODBC-

tietokanta-ajurisiltaa. Nämä löytyvät luvuista 5 ja 6. Näitä komponentteja käytetään toteutettavassa Javalla tehdyssä graafisessa käyttöliittymäsovelluksessa.

2 VALAISINJÄRJESTELMÄN TOIMINTAYMPÄRISTÖ

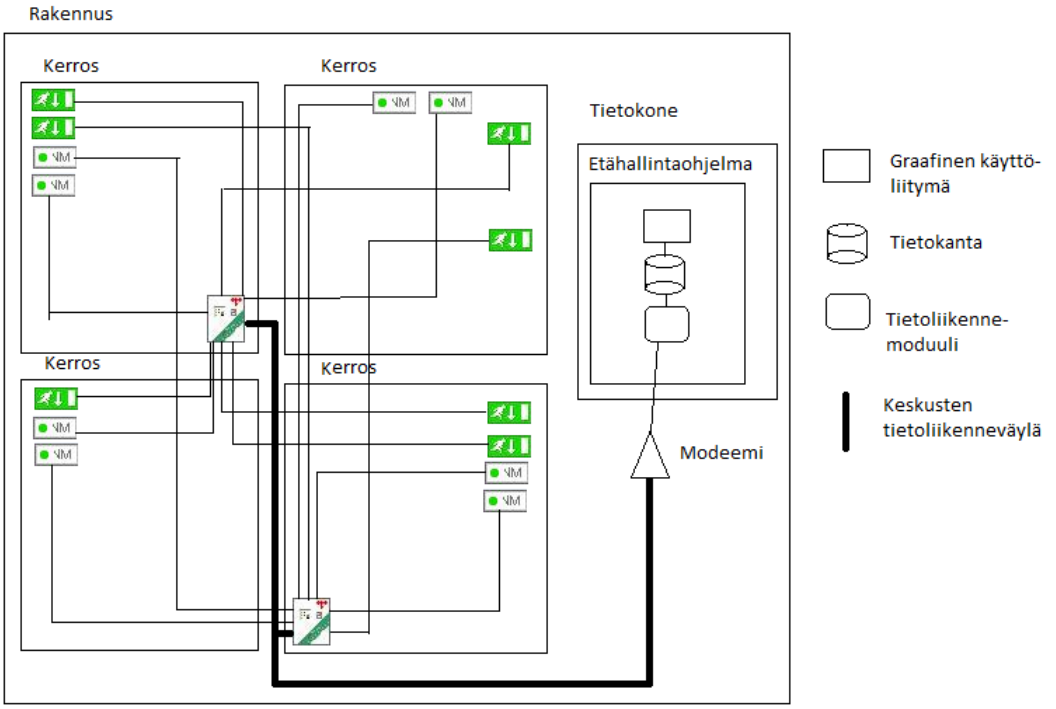
2.1 Valaisinjärjestelmän rakenne ja vaatimukset

Turvavalojärjestelmiä tarvitaan rakennuksissa osoittamaan uloskäyntejä ja valaisemaan poistumisreitit niiden luokse. Järjestelmän tulee toimia niin, että vikatilanteen sattuessa kaikki poistumisreitit pysyvät valaistuina vähintään yhden tunnin ajan vikatilanteen alkamisesta.

Jokaiselle uloskäynnille tulee asentaa vähintään kahdet valaisimet, jotka on kytketty eri keskuksiin. Tällä varmistetaan se, että yhden keskuksen vikaantuessa uloskäynti säilyy valaistuna. Valaistuksen tulee toimia myös, kun alueen normaalit valot lakkaavat toimimasta, esimerkiksi sähkökatkoksen aikana. Siksi keskukset vaativat normaalin virransyötön lisäksi akuston tukemaan virransaantia.

Suomen sisäasiainministeriön asetus koskien rakennusten poistumisreittejä ja valaisemista astui voimaan ensimmäinen päivä tammikuuta 2006. Siitä löytyy lakipykälät koskien turvavalaisemista ja sen kunnossapitoa. Sen mukaan poistumisopasteiden on oltava selkeitä, sekä ne on pystyttävä havaitsemaan, tunnistamaan ja ymmärtämään vaivatta. Valaisimien kunnossapitoa varten on laadittava kunnossapito-ohjelma, jossa selostetaan tarvittavat huoltotoimenpiteet. Tehdyt toimenpiteet merkitään joko kunnossapito-ohjelmaan tai erilliseen päiväkirjaan. (Sisäasiainministeriön asetus 2005.)

Käyttöliittymä tulee etähallintavälineeksi jossakin rakennuksessa sijaitsevaan turvavalojärjestelmään. Järjestelmään kuuluu yksi tai useampi turvavalokeskus, jotka on linkitetty toisiinsa sarjaliikenneväylän avulla. Keskukset sijoitetaan rakennuksessa eri kerroksiin tarpeen mukaan.



KUVIO 1. Esimerkki asennetusta järjestelmästä

Kuvio 1:ssä on esitys rakennuksesta, johon on sijoitettu 15 valaisinta. Nämä on kyketty kahteen keskukseen. Keskukset on kytketty toisiinsa sarjaliikenneväylällä, joka kytketään modeemiin, josta tiedot siirretään tietokoneeseen.

Jokainen keskus sisältää vähintään yhden ja maksimissaan 24 ryhmäkorttia mallista riippuen. Nämä ryhmäkortit voivat sisältää yhdestä 24 ryhmää. Ryhmien maksimivalaisimien määrä riippuu turvavalokeskuksen mallista. Koska rakennusten valaisintarve ja asennusmenetelmät vaihtelevat suuresti, voi järjestelmässä olla enemmän kuin kaksi keskusta, vaikka valaisinmäärä ei kokonaisuudessaan ylittäisi näiden maksimikapasiteettiä. Etähallinta on mahdollista Teknowaren itsetestaavissa Tapsa Control -turvavalojärjestelmissä.

Keskukset toimivat verkkovirralla, jonka lisäksi jokaiseen niistä on kytketty akku, joka pitää keskuksen toiminnassa sähkökatkoksen tai muun virransyöttövian aikana. Akun kuntoa tulee aika-ajoin testata keskuskohtaisesti ja näistä testeistä tulee tallettaa testitulokset, joita vaaditaan asetetuissa säädöksissä. Näissä testeissä simuloidaan vikatilannetta ja seurataan akun toimintaa. (Hongisto 2008.)

Myös keskuksessa olevat valaisimet tulee testata akkukäytöllä sekä normaalilla virralla. Valaisintestien tulokset tulee tallentaa usean vuoden ajalta, joita vaaditaan asetetuissa säädöksissä. Valaisimien kuntoa tarkkaillaan päivittäin, jotta niiden tiedetään olevan toimintakunnossa. Akkusyötön yhteydessä tarkarkastetaan valaisimien ja valaisinkilpien yleiskunto. (Hongisto 2008.)

2.2 Etähallintajärjestelmä

Valaisimien hallintajärjestelmä asennetaan erilliselle tietokoneelle. Tietokoneen ja keskuksien välillä on modeemi, joka kytketään tietokoneeseen USB-väylällä ja keskuksiin sarjaliikenneväylällä. Tätä pitkin keskusten lähettämät tiedot kulkevat tietoliikennemoduulille, joka tallentaa ne tietokantaan. Hallintajärjestelmän tulisi sisältää tarvittava tietokanta, joka sisältää taulut, joissa säilötään järjestelmän rakenne, testien historiatiedot sekä graafisen käyttöliittymän muodostamiseen tarvittavat osat.

Tietokannan tulisi toimia järjestelmän moduulien välisenä rajapintana, jotta tulevaisuudessa mahdollisten moduulien päivittäminen tai vaihtaminen onnistuisi helposti toteuttamalla kyseinen rajapinta. Itse graafisessa käyttöliittymässä tulisi olla mahdollisuus vaihtaa kielitietoja, jotta se saataisiin helpommin käytettäväksi erimaalaisten käyttäjien kesken.

Aikaisempien hallintajärjestelmien mukaan käyttöliittymässä tulisi olla havainnollistamista helpottavia puunäkymiä, erillinen karttapaneeli sekä tila, jossa keskusten mahdolliset virheelliset tilat näytetään käyttäjälle. Keskusten ja valojen testaustietoja tulisi myös pystyä selailemaan. Graafinen käyttöliitymä toteutetaan Javan Swing-kirjastoa hyväksikyttäen.

3 SWING

3.1 Historiaa

Internet Foundation Classes (IFC) luokkakirjasto oli alun perin kehitetty Netscape Communications Corporationin toimesta, ja se julkaistiin ensimmäisen kerran 16. joulukuuta 1996. Huhtikuun toinen päivä 1997, Sun Microsystems, nykyisin Oracle, ja Netscape Communications Corporation ilmoittivat aikeensa sulauttaa IFC muiden teknologioiden kanssa muodostaakseen Java Foundation Classes (JFC) luokkakirjaston. (Swing 2010.)

AWT (Abstract Window Toolkit) on ensimmäinen Javan käyttöliittymäriippumaton työkalu graafisten käyttöliittymäohjelmien luomiseen. Piirtäessä komponentteja se kutsuu suoraan alla olevan käyttöliittymän piirtorutiineja. Tämän vuoksi myös ohjelman ulkonäkö muuttuu riippuen siitä, missä käyttöjärjestelmän sitä ajetaan. Useat ohjelmoijat tahtoivat, että heidän tekemänsä sovellukset näyttäisivät samalta eri käyttöjärjestelmien päällä ajettaessa. Tämä johti siihen, että Swing-kirjasto syrjäytti AWT:n käytön ilmestyessään. (Abstract Window Toolkit 2010.)

Java Swing -kirjastoa jaettiin käyttäjille alun perin erillisenä ladattavana kirjastona, kunnes se liitettiin osaksi Java Standard Editionia sen 1.2. versiosta lähtien. Javan Swing on työkalu Java-ohjelmointikielellä toteutettavan graafisen käyttöliittymän toteuttamiseen (GUI). Swing-luokat ja komponentit on sijoitettu javax.swing-pakettiin. Swing sisältää GUI-vimpaimia (widget) kuten tekstilaatikoita, painikkeita, puunäkymän ja tauluja. (Swing 2010.)

3.2 Laajennettavuus

Swingin sovelluskehykset on ositettu tehokkaasti. Osittaminen mahdollistaa niiden mukautetun toteuttamisen. Sovelluskehyksen käyttäjä voi tarvittaessa käyttää itse tehtyä toteutusta ylikirjoittamalla sen oletustoteutuksen. Toinen tapa on periyttää alkuperäisen luokka ja lisätä siihen tarvittavat ominaisuudet. (Swing 2010.)

Swingin sovelluskehys rakentuu useista komponenteista. Ero objektin ja komponenttien välillä on hyvin pieni. Komponentti on objekti, jolla on tietty toimintatapa. Swingin objektit voivat eriaikaisesti laukaista tapahtumia, omata arvoja sekä vastaavat tunnettuihin komponenttikohtaisiin komentoihin. Swing-komponentit ovat Java Beans -komponentteja, jotka noudattavat Java Beans -komponenttiarkkitehtuuria (Swing 2010.)

3.3 Mukautettavuus

Koska komponentit omaavat Swingin sovelluskehyksen mallintamismahdollisuudet, voidaan niiden ulkonäköä muunnella hyvin vapaasti. Yleisesti ottaen komponentit omaavat perusominaisuudet, kuten esimerkiksi rajauksen (border), sisennyksen (insets) sekä koristelun (decoration). Lähtökohtaisesti ohjelmoija muokkaa ohjelmallisesti Swing-komponentin oletusmuotoa määrittämällä tietyt rajat, värit, näkyvyyden sekä muut ominaisuudet vastaamaan haluttua ulkonäköä. Sen jälkeen pohjalla oleva komponentti käyttää näitä määrityksiä tarkastellessaan, mitä renderöijiä sen tulee käyttää piirtäessään itseään näkyviin. Swingissä on myös mahdollista kehittää täysin uusia komponentteja yksilöllisin ulkonääin, jos valmiiden komponenttien joukosta ei löydy tilanteeseen sopivaa. (Swing 2010.)

3.4 Muokattavuus

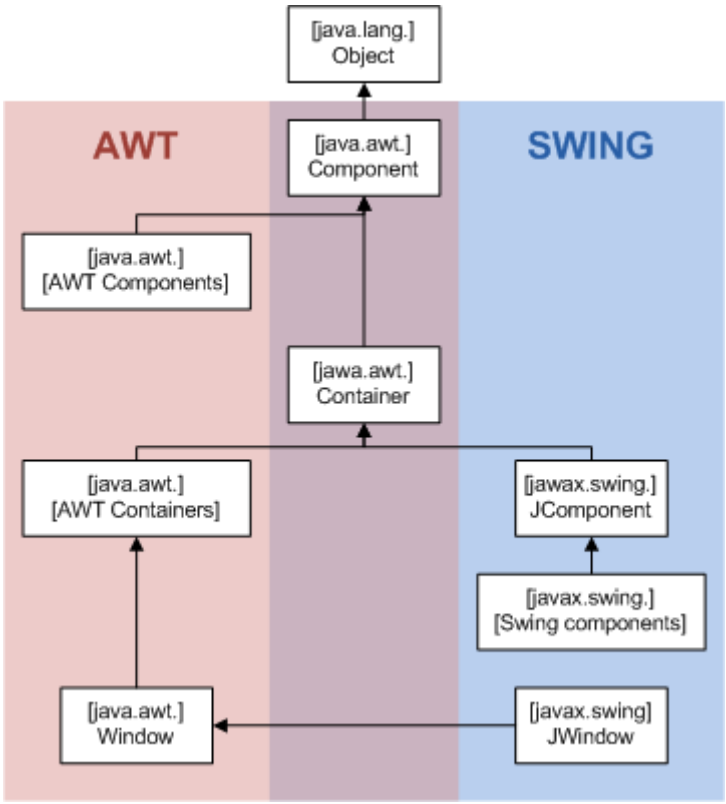
Swing mahdollistaa myös ajonaikaisen teeman (look and feel) muokkaamisen ilman, että käyttäjän täytyy tehdä muutoksia itse ohjelman lähdekoodiin. Tämä tarkoittaa sitä, että ohjelmaan voidaan tehdä esimerkiksi valikko eri ulkonköasetuksille, jotka valinnan jälkeen otetaan käyttöön. Ohjelmoija voi halutessaan myös luoda täysin uusia teemojen toteutuksia. (Swing 2010.)

3.5 Kevyt käyttöliittymä

Rakennetun ohjelman helppo muokattavuus on tulos siitä, että Swing-komponentti ei käytä pohjalla olevan käyttöjärjestelmän GUI-ohjaimia. Sen sijaan Swing piir-

tää näytettävät osat käyttäen Javan 2D-ohjelmointirajapintaa. Tästä syystä sen komponentit eivät omaa vastaavia kappaleita pohjalla olevasta käyttöliittymästä ja se on vapaa renderöimään komponentit alla olevan graafisen luokkakirjaston rajoissa. (Swing 2010.)

Pohjimmiltaan jokainen Swing-kirjaston komponentti luottaa Abstract Window Toolkitin pakettiin, sillä JComponentit perivät sen. Tämä mahdollistaa sen, että Swing voi kytkeytyä alla olevan käyttöjärjestelmän hallintasovelluskehykseen, joka sisältää laite- ja näyttömäärittelyt sekä käyttäjän tekemät toiminnat kuten hiiren liike tai näppäimistön painallukset. Luokkahierarkiaa on kuvattu tarkemmin kuviossa 2. Toisin kuin AWT-komponentit kutsuttaessa paint metodia, Swing ei käytä käyttöliittymän raskaita UI-komponentteja, vaan saa aikaan sen, että komponentit renderöivät itse itsensä. (Swing 2010.)



KUVIO 2. AWT:n ja Swingin luokkahierarkia (Swing 2010)

Javan ensimmäisistä versioista lähtien on AWT tarjonnut alustariippumattoman ohjelmointirajapinnan käyttöliittymä komponenteille. AWT:ssa jokainen komponentti renderöidään ja ohjataan pohjalla olevan käyttöliittymän mukaan. Swingin komponentteja usein kuvataan kevyinä, koska niiden käyttäminen ei vaadi alla olevan käyttöliittymän ohjainten sisällyttämistä niihin. (Swing 2010.)

Swingin tarjoamat komponentit ovat ominaisuuksiltaan hienostuneempia kuin aikaisemman AWT:n tarjoamat vimpaimet. Suurimpana etuna on niiden toteutus. Koska kaikki komponentit on rakennettu puhtaalla Java-kielellä, ne näkyvät käytössä samanlaisina, riippumatta siitä mikä käyttöjärjestelmä on käytössä. (Swing 2010.)

KUVIO 3. Swing- ja AWT-komponentin ulkonäkö Windows-käyttöjärjestelmässä

3.6 Debuggaus

Swing-sovelluksen debuggaus on vaikeampaa verrattuna normaaleihin ei-graafisiin sovelluksiin, sillä niitä ei voida helposti ajaa askel kerrallaan. Pääsyy tähän on se, että Swing suorittaa piirtämisen ensiksi muistissa sijaitsevaan puskuriin, josta se siirtää lopullisen tuloksen ruudulle näkyviin. Järjestelmä käyttää puskuria, jotta käyttäjä ei näe piirtoprosessia, joka ilmenisi näytöllä sen vilkkumisena. Tästä syystä on vaikea seurata yksittäisten graafisten operaatioiden vaikutusta käyttöliittymään käyttämällä yleiseen tarkoitukseen suunniteltua Java debuggeria. Tätä ongelmaa voidaan yrittää kiertää asettamalla kaksoispuskurointi pois päältä ja käyttämällä DebugGraphic-olioa. Tämän jälkeen piirto-operaatioita voidaan tarkkailla niiden rakentuessa näytölle. (Swing 2010.)

Myös piirtämisen suorittavaan säikeeseen liittyy muutamia yleisiä ongelmia. Swing käyttää AWT:n tapahtuman käsittelijäsäiettä komponenttien piirtämiseen.

Yhtälailla Swingin Standardiin kuuluu se, että kaikki komponentit täytyy olla sekä luotu että käsitelty AWT-tapahtuman käsittelijästä. Jos sovelluksen toiminta ei noudata tätä sääntöä, saattaa lopputuloksena syntyä hallitsemattomia tapahtumia. Swingin tarjoama käyttöliittymäriippumaton ulkoasu auttaa tämän ongelman kanssa, sillä se ei käsittele komponentteja, jotka eivät käytä tapahtumankäsittelijää. Näin ollen sitä käytettäessä myös virheelliset tapaukset voidaan helpommin paikantaa. (Swing 2010.)

Jos AWT-tapahtumankäsittelijäsäikeellä ajetaan pitkään kestäviä operaatioita, muiden käyttöliittymän Swing-komponenttien piirtäminen viivästyy, mikä aiheuttaa sen, että käyttöliittymä jää jumiin. Tällaiset operaatiot tulee sijoittaa ajettavaksi toiseen säikeeseen, jotta käyttöliittymä saadaan toimimaan siten, että se kykenee vastaamaan käyttäjän toimintoihin kohtuullisessa ajassa. (Swing 2010.)

4 JTREE

JTree-luokan avulla voidaan esittää hierarkkista tietoa selkeässä muodossa. JTree-komponentit eivät itsessään sisällä tietoa, se vain tarjoaa näkymän siihen. Muiden Swing-komponenttien tavoin, puunäkymä saa esittämänsä tiedon kysymällä sitä tietomallistaan. (Java Tutorial 2010.)

JTree-olio esittää saamansa tiedon pystysuorasti. Jokainen näytettävä rivi sisältää yhden tiedon, jota kutsutaan solmuksi. Jokainen puunäkymä sisältää yhden juurisolmun, josta muut solmut polveutuvat. JTreen-olion perusasetuksissa juurisolmu on asetettu näytettäväksi, mutta sen voi tarvittaessa piilottaa. Jokaista solmua, jotka voivat sisältää lapsisolmuja, kutsutaan oksiksi, vaikka niillä ei olisi lapsia. Solmuja, jotka eivät voi sisältää lapsisolmuja, kutsutaan lehdiksi. (Java Tutorial 2010.)

Oksasolmujen lapsimäärää ei ole rajoitettu, ja tavallisimmissa toteutuksissa käyttäjä voi avata tai sulkea oksasolmun alle sijoittuvan näkymän. Yleensä alkutilanteessa kaikki oksasolmut, juurisolmua lukuun ottamatta, ovat suljettuina. Ohjelma huomaa käyttäjän tekemät muutokset oksasolmujen tilassa kuuntelemalla Tree Expansion- tai TreeWillExpand-tapahtumaa. (Java Tutorial 2010.)

Tietty puun solmu voidaan löytää puusta joko TreePathin, solmun ja sen korkeammat tasot sisältävän objektin, tai sen näyttörivin perusteella. Avattu solmu ei ole lehtisolmu, ja se näyttää kaikki lapsisolmut, kun sen sisältävä rakenne on avattuna. Suljettu solmu on solmu, joka ei näytä sen alla olevaa puurakennetta. Piilossa oleva solmu on sellainen, joka sijaitsee suljetun solmun piilossa olevassa rakenteessa. (Java Tutorial 2010.)

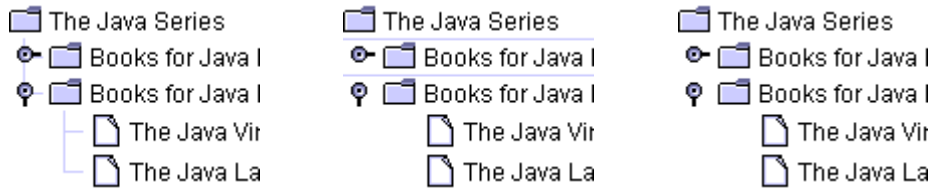
```
1  //Where instance variables are declared:
2  private JTree tree;
3  ...
4  public TreeDemo() {
5      ...
6      DefaultMutableTreeNode top =
7          new DefaultMutableTreeNode("The Java Series");
8      createNodes(top);
9      tree = new JTree(top);
10     ...
11     JScrollPane treeView = new JScrollPane(tree);
12     ...
13 }
```

KUVIO 3. JTreen luonti (Java Tutorial 2010)

Kuvion 3 esimerkkissä luodaan DefaultMutableTreenode-instanssi, joka palvelee puun juurisolmuja. Sen jälkeen se luo muut puun sisältämät somut. Rivillä 9 luodaan itse puu siten, että aikaisemmin tehty juurisolmu välitetään arvoksi JTreen muodostimelle. Lopuksi puu sijoitetaan JScrollPane-komponenttiin. Tämä on yleinen käytäntö, sillä täysin avattu puu voi viedä paljon tilaa näytöltä, ja tämän vuoksi tarvitaan vierityspalkit. (Java Tutorial 2010.)

4.1 JTreen ulkonäön muokkaaminen

Tavanomaisesti puunäkymässä näkyy solmun ikoni sekä tekstirivi solmua kohden. Puu myös toteuttaa teema kohtaisia piirtomenetelmiä kuvaamaan solmujen keskenäistä suhdetta. Jos käytössä on Javan oma teema asetus, piirtää puu oletusasetuksilla haarautuvat viivat solmujen välille. Muuttamalla JTree.lineStyleasetuksen arvoja voidaan tätä erottelutapaa muokata. (Java Tutorial 2010.)



`"Angled"` (default) `"Horizontal"` `"None"`

KUVIO 4. Kolme Javan tapaa kuvata solmujen suhdetta (Java Tutorial 2010)

Perusaseteuksilla puunäkymän solmujen ikonien ulkonäkö määräytyy sen mukaan, onko kyseessä avattu solmu tai lehti. Kuviossa 4 oksasolmuja kuvataan kansioilla ja lehtisolmut näyttävät paperiarkeilta. Näitä ikoneja päästään muuttamaan luomalla DefaultTreeCellRenderer-instassi ja määrittelemällä siihen halutut kuvat setLeafIcon-, setOpenIcon- ja setClosedIcon-toiminnoilla. Tämän jälkeen, Default-TreeCellRenderer asetetaan puun käyttöön asettamalle se setCellRenderetoiminnon arvoksi. CellRenderer voi vain piirtää ikoneja, ei suorittaa tapahtumia. Näitä varten tulee puuhun rekisteröidä erillinen tapahtumankäsittelijä. (Java Tutorial 2010.)

4.2 Tietomallin luominen

Jos puunäkymää varten tarvitaan perusmallista poikkeavaa tietomallia, täytyy se kirjoittaa kokonaan uudestaan. Tietomallin tulee toteuttaa TreeModel-rajapinta. TreeModel tarjoaa seuraavat toiminnot:

- Sen avulla puusta löydetään tietty solmu sekä tietyn solmun lapset.
- Saadaan tieto siitä, onko solmu lehti.
- Saadaan ilmoitus mallille, jos puu muuttuu.
- Se mahdollistaa puumallin kuutelijoiden lisäämisen sekä poistamisen.

(Java Tutorial 2010.)

TreeModelrajapinta hyväksyy kaikenlaisia objekteja puun solmuiksi. Solmujen ei täydy toteuttaa TreeNoderajapintaa tai edustaa DefautMutableTreeNodeobjektia.

Tämän vuoksi jo valmiina olevat hierarkkiset tietorakenteet voidaan esittää ilman niiden kahdentamista tai TreeNodemalliin sovittamista. (Java Tutorial 2010.)

5 JDBC

JDBC (Java Database Connectivity rajapinta) mahdollistaa Java-sovellusten tietokantayhteyden. Yhdeyden muodostamiseksi se tarvitsee JDBC-ajurin, joka muodostaa yhteyden tietokantaan ja toteuttaa protokollan, jolla tietokantakyselyt sekä tulokset välitetään sovelluksen ja tietokannan välillä. Sun Microsystems (nykyisin Oracle) julkaisi JDBC:n osana JDK 1.1 19. helmikuuta 1997 ja se on siitä lähtien ollut osana Java Standard Editionia. JDBC:n luokat löytyvät java.sql -paketista (JDBC 2010).

5.1 Toiminnallisuus

JDBC sallii useamman erilaisen toteutuksen yhden ohjelman sisällä. Rajapinta tarjoaa toiminnan dynaamiseen Java-pakettien lataamiseen ja rekisteröimiseen JDBC Driver Manageriin. Driver Manageria käytetään rakentajana luotaessa JDBC yhteyksiä. (JDBC 2010.)

JDBC tukee SQL:n lauseita, kuten CREATE, INSERT, UPDATE ja DELETE tai SELECT. Lisäksi myös tallennettujen proseduurien ajaminen on mahdollista. JDBC esittää lauseita käyttäen jotain seuraavista luokista:

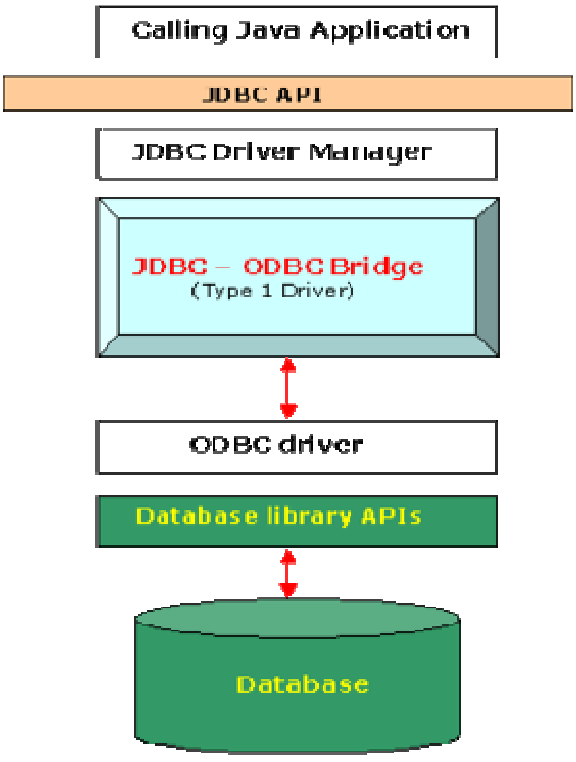
- Statement - lause lähetetään tietokantaserverille joka kerta.
- PreparedStatement - lause tallennetaan välimuistiin, ja sen suorituspolku on ennalta määritelty tietokantaserverillä, josta se voidaan suorittaa useampaan kertaan säästäen käytössä olevia resursseja.
- CallableStatement - tätä käytetään ajettaessa tallenettuja proseduureja tietokantaan.

Päivityslauseet, kuten INSERT, UPDATE ja DELETE, palauttavat suorituksen jälkeen tieton siitä, kuinka monta riviä ajettu lause päivitti. Nämä lauseet eivät tämän lisäksi palauta muuta tietoa käyttäjälle. (JDBC 2010.)

Kyselylauseet palauttavat JDBC-tulosjoukon. Näistä voidaan hakea yksittäisiä sarakkeita joko nimen tai sarakenumeron perusteella. Tulosjoukossa olevien rivien maksimilukumäärää ei ole rajattu. Se myös sisältää kuvaustiedon riveistä, joka kuvaa sarakkeiden nimen sekä sen, minkä tyyppistä tietoa ne sisältävät. (JDBC 2010.)

5.2 JDBC-ODBC-silta

JDBC-ODBC silta on tietokanta-ajuri, joka käyttää ODBC-ajuria yhdistymään tietokantaan. Ajuri muuntaa JDBC-metodikutsut ODBC-funktiokutsuiksi. Itse ajuri on alustariippuvainen, koska se käyttää hyväksi ODBC:tä, joka vuorostaan on riippuvainen alla olevan käyttöliittymän kirjastoista. (JDBC driver 2010.)



KUVIO 5. JDBC-ODBC silta (JDBC driver 2010.)

Tämä ajuri tuo mukanaan myös muita vaadittavia asennustarpeita. ODBC-ajuri tulee olla asennettuna tietokoneeseen, jossa sitä käytetään, sekä käytössä, olevan tietokannan tulee tukea ODBC-ajureita. Tästä syystä JDBC-ODBC-sillan käyttöä

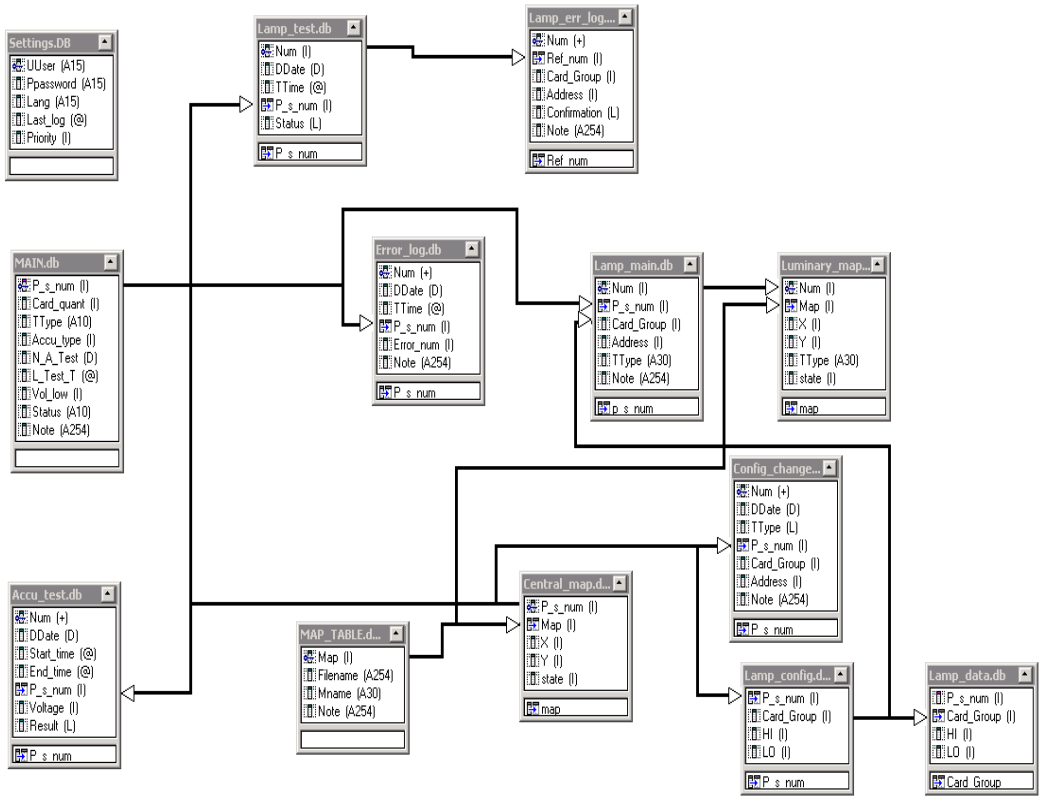
ei suositella, jos on olemassa käyttöön sopiva puhdas Java-ajuri. Toinen seuraus ajurin käyttämisestä on se, että ohjelma ei enää toimi alustariippumattomasti, koska siitä tulee ODBC riippuvainen. Koska ajuri tulee olla asennettuna käytössä olevaan laitteeseen, ei se myöskään sovi pienoissovellusten käyttöön. (JDBC driver 2010.)

6 TIETOKANTA

Kun työssä toteutettavan järjestelmän rakenne selvitettiin, alkoi sen toteuttaminen tietokannan suunnittelulla. Tietokannan toimiessa rajapintana komponenttien välillä oli tärkeää saada tietokanta valmiiksi sellaiseen muotoon, että siihen ei tulisi myöhemmin enää suuria muutoksia. Jos näin kävisi, voisi se pahimmassa tapauksessa johtaa siihen, että kaikki tietokantaa käyttävät osat tulisi ohjelmoida uudestaan vastaamaan muutoksia.

6.1 Tietokannan rakenne

Lopulliseen versioon tietokannasta sisältyi kolmetoista taulua. Osa tauluista on yksinomaan graafisen käyttöliittymän käytössä. Näihin tauluihin tehdään SQL-kyselyitä, joista saaduilla tiedoilla rakennetaan järjestelmän näkymä.



KUVIO 6. Tietokannan rakenne

6.2 Tietoa tauluista

Seuraavassa osiossa käydään läpi tietokannassa olevia tauluja sekä niiden ominaisuuksia ja käyttötarkoitusta.

6.2.1 Accu_test

Accu_test-taulu on tehty pitämään tallessa järjestelmässä olevien keskusten akkutestien tietoja. Koska tietojen täytyy olla tallessa usean vuoden ajalta, tähän tauluun lisätään jokaisen tapahtuneen testauksen jälkeen uusi rivi.

Itse taulu sisältää Num-, DDate-, Start_time-, End_time-, P_s_num-, Voltage- ja Result- kentät. Numkenttä on kasvava numerointi suoritetuista testeistä. DDate-, Start_time-, End_time -kentät sisältävät kyseisellä kerralla suoritetun testin päiväyksen sekä testin aloitus- ja lopetuskellonajan. P_s_num toimii avaimena MAIN-taulua vastaan ja pitää sisällään testin suorittaneen keskuksen numeron. Voltage ilmaisee akun jännitteen testin jälkeen. Resultkenttä on boolean typpinen, ja se ilmoittaa, onko testi onnistunut vai ei.

6.2.2 Central_map

Central_map-taulu on pelkästään graafisen käyttöliittymän käytössä, ja se sisältää tiedot keskuksen kuvan paikasta karttanäkymässä. Se sisältää P_s_numkentän, joka toimii avaimena MAIN-tauluun ja ilmaisee samalla kysessä olevan keskuksen numeron. Sen lisäksi taulussa on Map-, x- ja y -kentät. Nämä pitävät sisällään tietoa siitä, mihin kerrokseen keskus on asetettu näkyviin, sekä x- ja y -koordinaatit kyseisellä pohjalla. Mapkenttä toimii myös avaimena Map_table-tauluun. Viimeisenä taulussa on statetieto, joka ilmaisee, onko keskuksella aktiivinen vika. Jos tämä tieto on 1, karttanäkymässä näkyy punainen laatikko keskuksen ikonin ympärillä ilmaisten virhetilannetta.

6.2.3 Config_change

Config_change-taulu on vain tietoliikennemoduulin käyttämä, ja sitä käytetään, jos järjestelmässä ajetaan muutostietoja sisään. Avaimena taulussa on P_s_numkenttä MAIN-tauluun, joka kertoo keskuksen numeron, jota muutokset koskevat. Taulussa on myös Num-kenttä, joka sisältää kasvavan numeroinnin indeksoiden kaikki tapahtuneet muutokset.

6.2.4 Error_log

Error_log-taulu sisältää tiedot virheellisistä akkutesteistä. Niihin voidaan viitata Accu_test-taulun Numkentällä. Testeistä tallennetaan päiväys, kellonaika, keskuksen numero sekä virheen numero.

6.2.5 Lamp_config ja Lamp_data

Lamp_config- ja Lamp_data -taulut ovat vain tietoliikennemoduulin käyttämiä, ja niitä käytetään tallentamaan tieto siitä, kuinka monta valaisinta keskuksessa on asennettuna sekä miten ne ovat asettuneet sen sisältämille korteille. Lamp_configilla on P_s_numkenttä avaimena MAIN-tauluun ja Lamp_data-taulun avaimena toimii Card_Group Lamp_config-tauluun.

6.2.6 Lamp_err_log

Lamp_err_log-taulu sisältää järjestelmässä ajettujen valotestien tietoja. Valotesteistä pitää olla usean vuoden tiedot tallessa, joten taulussa on indeksoiva Num-kenttä, joka kasvaa jokaisen testin myötä. Ref_numkenttä toimii viiteavaimena Lamp_test-taulun Numkenttään, jolla vastaava testi löydetään. Card_groupkenttään jää tieto virheellisen tiedon antaneesta korttiryhmästä, ja Address kertoo, missä kortin osoitteessa vika ilmeni. Confirmation on boolean tyylinen kenttä, joka näyttää, onko vika huomattu ja vahvistettu. Taulussa on myös Notekenttä, johon mahtuu 255 merkkiä pitkä, vapaamuotoinen tieto kuvaamaan tapahtunutta.

6.2.7 Lamp_main

Lamp_main sisältää tietoja yksittäisestä lampusta, ja sitä käytetään rakennettaessa puunäkymää keskuksista ja valoista. Tietokenttinä taulussa on Num, P_s_num, Card_group, Address, TType sekä Note. Numkenttä kertoo, mikä kyseisen lampun numero järjestelmässä on. Se myös toimii avaimena Luminary_map-tauluun. P_s_num osoittaa sen, mihin keskukseen valaisin kuuluu ja toimii samalla avaimena MAIN-tauluun. Card_group ilmaisee, mihin korttiin kyseissä keskuksessa valaisin on kytketty. Address kertoo osoitteen kyseisellä kortilla, missä valaisin on kiinni. TType ilmaisee, minkä mallinen valaisin on kyseessä. Kenttä myös antaa valaisimelle sitä vastaavan ikonin puunäkymässä. Note on maksimissaan 255 merkkiä pitkä vapaamuotoinen tieto valaisimesta.

6.2.8 Lamp_test

Lamp_test-taulu sisältää tiedot ajettujen valaisintestien viittausnumerosta, päivämääristä, kellonajoista ja keskuksesta, jota kyseinen testi koskee. Talteen otetaan myös tieto siitä, onko testi onnistunut ilman virheitä. Virheellisten testien tiedot haetaan viittausnumeron avulla Lamp_err_log-taulusta.

6.2.9 Luminary_map

Luminary_map-taulu on vain graafisen käyttöliittymän käytössä ja sitä käytetään rakennettaessa puunäkymiä sekä asetettaessa valaisimien ikoneita karttanäkymään. Taulussa on Numkenttä, joka toimii avaimena Lamp_main-tauluun. Kenttä myös ilmoittaa, mikä valaisin on kyseessä. Map, x ja y ilmaisevat, mihin kerrospohjaan valaisin on asetettu sekä mitkä ovat sen koordinaatit karttanäkymässä. TTypekenttä ilmaisee, minkälainen valaisin on kyseessä, ja sen avulla saadaan oikeanlainen ikoni näkyviin. Statekenttä ilmoittaa, onko kyseinen valaisin virhetilassa. Arvo 1 tässä kentässä aiheuttaa ikonin ympärille punaisen laatikon karttanäkymässä.

6.2.10 MAIN

MAIN-taulu sisältää tietoja järjestelmässä kiinniolevista keskuksista. Keskusnäkymä pääikkunassa rakennetaan pääosin tämän taulun perusteella. Taulu sisältää P_s_numkentän, joka kertoo keskuksen numeron. Tämä kenttä toimii avaimena kaikkiin keskusta käsitteleviin tauluihin. Card_quant ilmaisee, montako korttia keskukseen on kytkettynä. TTypekenttä kertoo keskuksen mallin ja Accu_typekenttä keskuksessa kiinni olevan akun tyypin. N_A_TEST- ja L_TEST_T -kentät pitävät sisällään tietoa siitä, koska keskuksen seuraava akkutesti suoritetaan ja milloin valaisintesti on viimeksi suoritettu.

6.2.11 Map_table- ja settings -taulut

Map_table-taulu on vain graafisen käyttöliittymän käytössä, ja sen tarkoituksena on pitää tallessa lisätyt rakennuksen kerrokset, niiden nimet sekä niitä vastaavien kuvatiedostojen nimet.

Settings-taulu pitää sisällään graafisen liittymän käyttäjätiedot sekä niiden salasanat, tasot ja viimeisimmät kirjautumisajankohdat. Taulu on irrallaan muista, ja sitä käytetään vain graafisen käyttöliittymän toimesta.

6.3 Ongelmatilanteita

Tietokantojen osalta tuli vastaan useita ongelmatilanteita. Ensimmäinen suuri ongelma oli saada tietokanta toimimaan niin, että ohjelmat eivät lukitsisi kantoja ja täten estäisi niiden lukemista tai kirjoittamista. Käyttöliittymän tehdessä SQL-kyselyjä kantaan, lukitsi se kaikki taulut, joita kysely koski. Se ei vapauttanut tauluja luettavaksi tai kirjoitettavaksi, kunnes SQL-lauseen tulosjoukko sai uudet arvot tai tietokantayhteys suljettaisiin. Ratkaisua etsittäessä huomattiin, että tietokanta-ajurit eivät tue non-blocking-kyselyjä eikä yhteyden sulkeminen ja käynnistäminen vaikuttanut toimivalta ratkaisulta.

Ongelma saatiin ratkaistua niin, että jokaisen SQL-kyselyn jälkeen käyttöliittymä tekisi myös toisen kyselyn, kun vanhaa tulosjoukkoa ei enää tarvittaisi. Tämän kyselyn kohdetauluna toimi vain käyttöliittymän käytössä oleva Settings-taulu. Taulu jäi lukkoon, mutta sillä siihen ei tarvinnut päästä muualta kiinni, se ei enää haitannut.

Taulujen ja kenttien nimeäminen oli myös hieman ongelmallista, sillä alussa annetut kentät toimivat vain niiltä osin, joiden nimet olivat tarpeeksi lyhyitä. Tämä ongelma saatiin korjattua muokkaamalla niiden nimet sopivan mittaisiksi. Myös muutamat varatut sanat tarvitsivat muokkaamista, joten niihin lisättiin ylimääräinen merkki tämän ohittamiseksi.

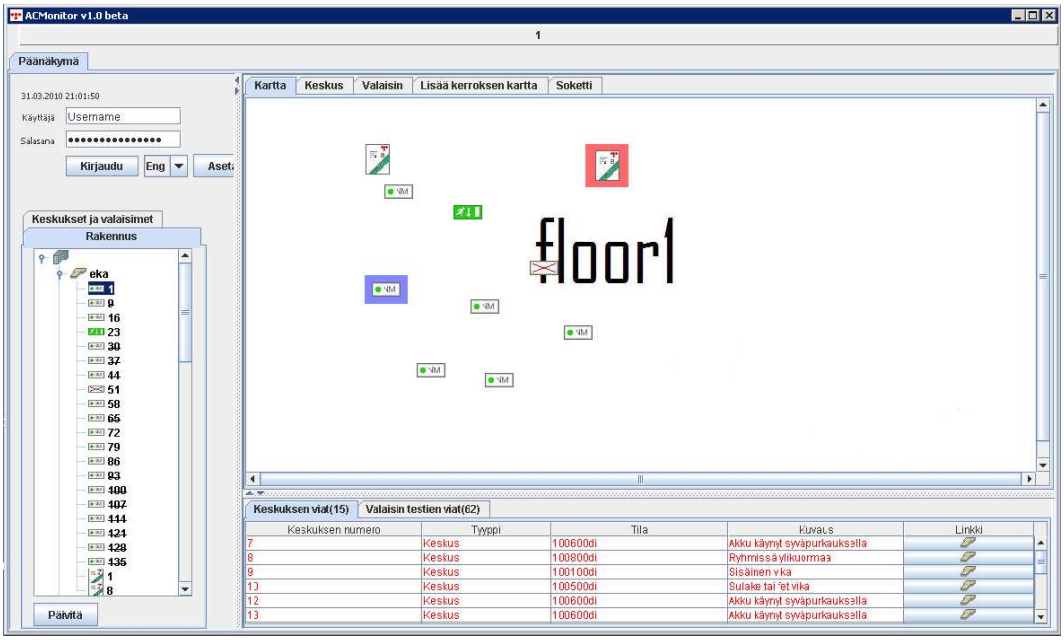
Eräs suurempi ongelma tietokantojen osalta tuli vastaan, kun käyttöliittymää testattiin suurella määrällä tietoa. Ohjelman rakentaessa puunäkymiä tekee se suuren määrän kyselyitä tauluihin ja nämä aiheuttivat virhetapahtuman, jonka takia tietokanta yhteys sulkeutui, eikä se enää toiminut normaalisti edes uuden tietokantayhteyden muodostamisen jälkeen, vaan vaati koko käyttöliittymän uudelleenkäynnistämisen. Aluksi virheen syytä ei tiedetty, joten vikaa koetettiin korjata keventämällä tarvittavien kyselyjen määrää. Tämä auttoi tilanteeseen hieman, mutta lopulta samaan virheeseen päädyttiin, kun ohjelmaa ajettiin tarpeeksi kauan. Tietoliikennemoduulia testattaessa havaittiin, että virhetilanne tapahtui dBASE-ajureissa olevan muistivuodon takia. Ajettaessa ohjelmaa tietokanta-ajurin viemä muistimäärää alkoi kasvaa, kunnes se lopulta lakkasi toimimasta. Tästä syystä päädyimme vaihtamaan tietokannan taulujen formaatin Paradox-tauluihin. Vaihdon jälkeen ei muistivuotoa enää syntynyt.

7 PÄÄNÄKYMÄ

Käyttöliittymän oikeassa yläosassa sijaitsee kansiokomponentti, jonka välilehdiltä löytyy useita eri toimintoja kuten karttojen lisääminen tai keskuksen tietojen selailu. Oikeaan yläreunaan on sijoitettu kirjautumisvalikko, jonka kautta saadaan avattua asetukset välilehti.

Vasenpaan alareunaan on sijoitettu puunäkymät. Alareunassa on vikailmoitusruutu valaisimille ja keskuksille. Sekä puunäkymät että vikailmoitukset on sijoitettuna kansiokomponenttiin, joten molemmista komponenteista voi kerralla olla näkyvissä vain toinen osio.

Näkymään piirretyt komponentit ovat muokattuja JToggleButtoneita, sillä niihin saa lisättyä tapahtumakuuntelijan, jonka avulla tarvittavat toiminnot saadaan toteutettua. Näytettävän kerroksen vaihtuessa pohja tyhjennetään kaikista komponenteista, minkä jälkeen siihen lisätään SQL-kyselyn perusteella saadut uudet painikkeet ja pohja piirretään uudelleen.



KUVIO 7. Käyttöliittymän päänäkymä

7.1 Kartta-välilehti

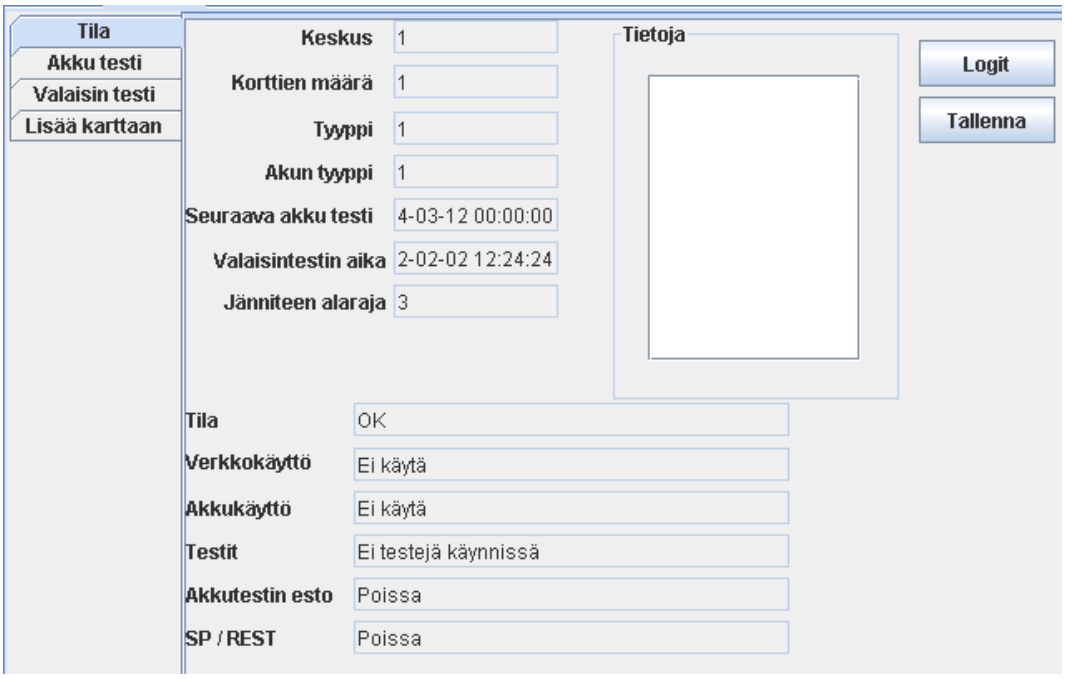
Kartta välilehdellä esitetään rakennuksen karttanäkymä. Kartassa näkyy kulloinkin valittuna olevan kartan kuva sekä mahdolliset siihen lisätyt valaisimet. Karttanäkymässä näkyy myös valittuna oleva valaisin tai keskus. Valittuna oleva kohde erottuu muista siten että, sen reunat on ympäröitynä sinisellä kehyksellä. Jos jokin osanen on ympäröitynä punaisella tarkoittaa se sitä, että kyseinen systeemin osa on viallinen. Keltainen väri ilmoittaa, että vika on kuitattu mutta ei vielä korjattu.

Kuviossa 7 on näkyvillä karttapohja, jota edustaa kuvatiedosto, jossa on teksti floor 1. Valmiissa järjestelmässä sen tilalla voisi olla esimerkiksi kerroksen pohjapiirros, jonka päälle valaisimet asetetaan siten, että se vastaa niiden fyysistä sijaintia. Karttanäkymässä ei ole näkyvillä keskusten tai valaisimien numeroita, mutta liikkuttamalla kursori halutun kohteen päälle, avautuu infolaatikko, joka sisältää nämä tiedot.

7.2 Keskus-välilehti

Keskus-välilehti koostuu uudesta kansiokomponentista, jonka välilehdiltä pystyy tarkkailemaan valittuna olevan keskuksen tietoja. Tietokentät näkyvät tyhjinä, jos keskusta ei ole valittuna.

7.2.1 Tila-välilehti



KUVIO 8. Keskuksen tila-välilehti

Ensimmäinen välilehti antaa keskuksesta yleistä tietoa. Ensimmäisellä rivillä näytetään valittuna olevan keskuksen numero. Toinen rivi ilmoittaa, kuinka monta korttia kyseisessä keskuksessa on kytkettynä. Kolmas rivi kertoo, minkä tyyppinen keskus on kyseessä. Neljäs rivi ilmoittaa, minkälainen akku keskuksessa on käytössä, ja viides rivi ilmoittaa akun seuraavan testauksen ajankohdan. Kuudes rivi näyttää, mikä on tämän keskuksen valaisintestauksen aloitusaika.

Välilehdellä on myös testikenttä, jossa näkyy keskukselle mahdollisesti lisätty lyhyt kuvaus. Tämän lisäksi siellä on kaksi painiketta, logit sekä tallenna. Logi-painike avaa uuden ikkunan, jossa voi selailla valaisintestien vikalogeja. Tallenna-painike tallentaa tekstikentässä olevan tekstin keskuksen tietoihin.

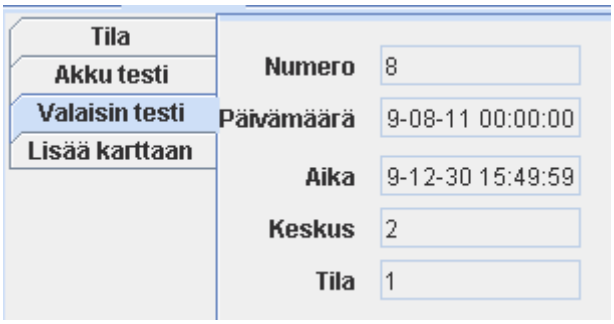
Tähän näkymään päästään myös kahden oikotien kautta. Painamalla joko karttanäkymässä tai puunäkymässä kaksi kertaa tiettyä keskuksen kuvaa ohjelma avaa välilehden näkyviin siten, että kyseisen keskuksen tiedot ovat näkyvissä.

7.2.2 Akkutesti-välilehti

KUVIO 9. Akkutesti-välilehti

Akkutesti-välilehti kertoo tietoja valittuna olevan keskuksen akkutestauksesta. Ensimmäinen kenttä kertoo testin numeron. Päivämääräkenttä ilmaisee, koska akkua on viimeksi testattu. Aloitusaika kertoo, koska testi aloitettiin, ja lopetusaika, koska testin suoritus loppui. Keskuskenttä ilmaisee numerolla, minkä keskuksen tiedot ovat näkyvissä. Jännitekenttä ilmoittaa keskuksen akun jännitteen. Tuloskenttä ilmaisee, onko viimeisin ajettu testi antanut virheitä.

7.2.3 Valaisintesti-välilehti



KUVIO 10. Valaisintesti-välilehti

Valaisintesti-välilehden tarkoituksena on antaa tietoja valittuna olevan keskuksen valotestin suorituksesta. Ensimmäisellä rivillä näkyy valaisintestin numero. Toisella rivillä näkyy viimeisimmän valaisintestin päivämäärä. Aikarivillä ilmoite-

taan, mihin aikaan testi alkoi. Keskus ilmoittaa tarkastelussa olevan keskuksen numeron. Tila kenttä ilmaisee, onko viimeisin ajettu testi antanut virheitä.

7.2.4 Lisää karttaan -välilehti

KUVIO 11. Keskusten lisää karttaan -välilehti

Sen jälkeen kun keskukset on asennettu ja niiden tiedot on lisätty tietokantaan, voidaan niitä alkaa lisätä rakennuspuunäkymään. Lisääminen tapahtuu niin, että keskuskenttään syötetään lisättävän keskuksen numero ja karttakenttään kartan numero, johon keskus kuuluu (lisää kartta -välilehti). Kun tiedot on annettu, painetaan muokkaa-painiketta, joka siirtää tiedot tietokantaan. Välilehdellä oikealla olevassa listassa näkyy kaikki kerroksiin lisätyt keskukset ja vasemmanpuoleisessa listassa näkyy kaikki lisäämättömät. Sen jälkeen kun järjestelmä on syötetty käyttökuntoon, tulisi vasemmanpuoleisen lista olla tyhjä.

7.3 Valaisin-välilehti

Valaisi-välilehti koostuu uudesta kansiokomponentista, jonka välilehdiltä pystyy tarkkailemaan valittuna olevan valaisimen tietoja. Tietokentät näkyvät tyhjinä, jos mitään valaisinta ei ole valittuna.

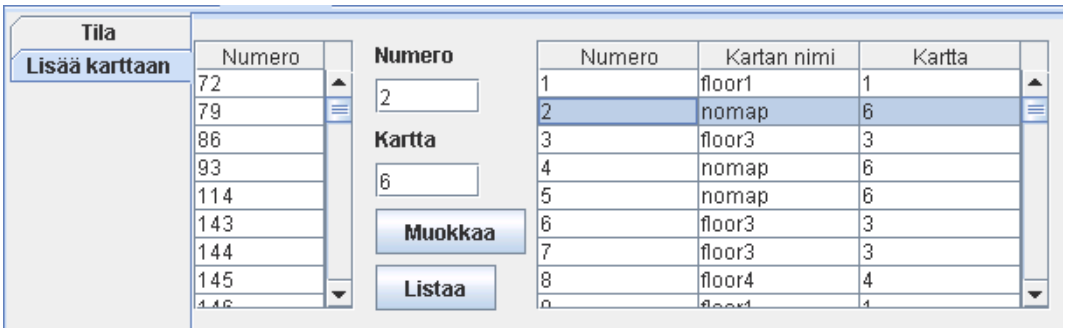
7.3.1 Tila-välilehti

KUVIO 12. Valojen tila-välilehti

Ensimmäinen välilehti näyttää tietoja valaisimesta. Keskus rivi ilmaisee, mihin järjestelmän keskukseen valaisin on kytketty. Kortti rivi kertoo, mihin korttiin kyseisessä keskuksessa valaisin kuuluu. Numerorivi ilmoittaa valaisimen numeron. Tyyppirivi ilmaisee, minkä mallinen valaisin on kyseessä. Se myös sisältää painikkeen, jolla valaisintyypin kuvausta voidaan vaihtaa.

Välilehdellä on myös testikenttä, jossa näkyy valaisimelle mahdollisesti lisätty lyhyt kuvaus. Tämän lisäksi siellä on kaksi painiketta, logit sekä tallenna. Logi-painike avaa uuden ikkunan, jossa voi selailla valaisintestien vikalogeja. Tallenna-painike tallentaa tekstikentässä olevan tekstin valaisimen tietoihin. Tähän näkymään päästää myös kahden oikotien kautta. Painamalla joko karttanäkymässä tai puunäkymässä kaksi kertaa tiettyä valaisimen kuvaa, ohjelma avaa välilehden näkyviin siten, että kyseisen valaisimen tiedot ovat näkyvissä.

7.3.2 Lisää karttaan -välilehti



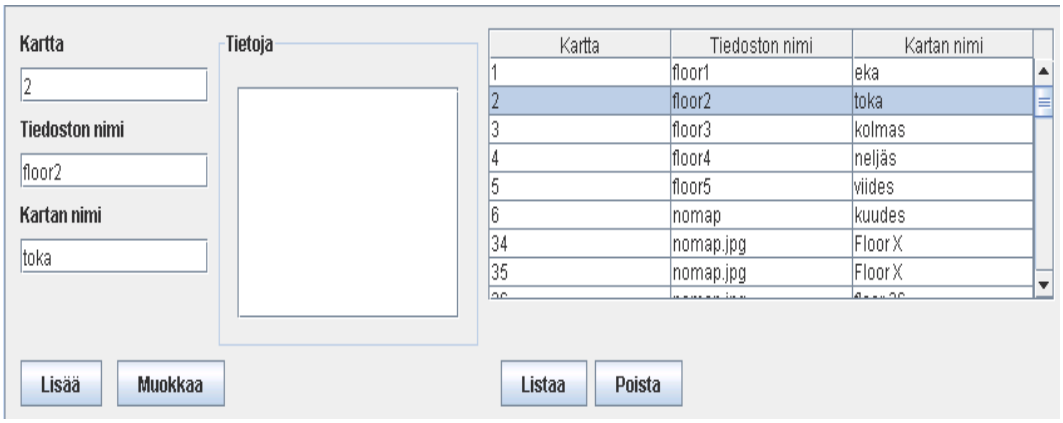
KUVIO 13. Valojen lisää karttaan -välilehti

Lisää karttaan -välilehden tarkoitus on lisätä valaisimia Rakennuspuunäkymään. Valaisimia voidaan alkaa lisätä, kun tiedot systeemin kuuluvista valaisimista on syötetty tietokantaan.

Puunäkymään lisääminen tapahtuu siten, että numerokenttään kirjoitetaan lisättävän valaisimen numero ja karttakenttään kirjoitetaan kohdekerroksen numero. Tämän jälkeen painetaan muokkaa-painiketta, joka suorittaa lisäämisen. Jos valaisin syötetään vahingossa väärään kerrokseen, voidaan se siirtää samoin kuin lisättäessä. Tämä voidaan myös tehdä painamalla jo syötettyä kerrosta listasta, jolloin ohjelma esitäyttää syöttökentät käyttäjän puolesta.

Listaa-painike hakee tietokannasta tiedot molemmilla puolilla oleviin listoihin. Kun järjestelmä on täysin asennettu, ei numero listassa tulisi olla enää yhtään valaisimia vaan niiden tulisi löytyä sijoittettuna johonkin karttaan. Tilanteessa, jossa valaisin tahdotaan palauttaa takaisin valittaviin, tulee se lisätä karttaan numero 0.

7.4 Lisää kerroksen kartta -välilehti



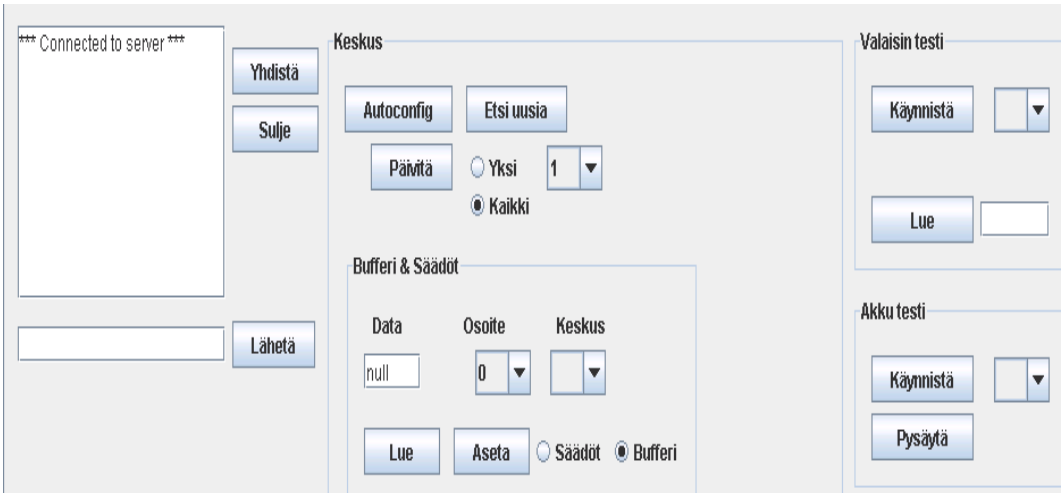
KUVIO 14. Lisää kerroksen kartta -välilehti

Lisää kerroksen kartta -välilehti on tarkoitettu rakennuspuunäkymän rakenteen muodostamiseen. Jokainen lisätty kerros tekee puuhun yhden kerrostason, jonne voidaan myöhemmin lisätä keskuksia sekä valaisimia.

Ensimmäinen syöttörivi on tarkoitettu kerroksen numerolle, jonka mukaan ne asettuvat puunäkymään. Jos rakennuksessa on esimerkiksi kellarikerroksia ja ne halutaan näkymään puun ylimmäisenä, tulisi numeroiminen aloittaa niistä. Tiedoston nimikenttä määrittää karttapohjaan tulevan kuvatiedoston nimen. Tässä versiossa tiedoston nimi tulee syöttää ilman päätettä ja se käyttää vain JPG-formaatissa olevia kuvia. Kartan nimikenttään kirjoitetaan nimi, joka tulee näkyviin puunäkymässä, esimerkiksi kellari tai kerros 6. Välilehden oikeassa reunassa oleva näkymä ilmoittaa riveittäin syötettyjen kerrosten tiedot näyttäen kaikki edellä mainitut arvot.

Kerrostietoja voidaan muokata jo käytössä olevan kerrosnumeron avulla syöttämällä se karttakenttään ja antamalla uudet tiedot muihin kenttiin ja painamalla muokkaa-painiketta. Tämä voidaan myös tehdä painamalla jo syötettyä kerrosta listasta, jolloin ohjelma esitäyttää syöttökentät käyttäjän puolesta. Kerroksen poistaminen kokonaan tapahtuu samalla periaatteella, mutta lopuksi painetaan muokkaa napin sijasta poista-nappia.

7.5 Soketti



KUVIO 15. Soketti-välilehti

Soketti-välilehti sisältää kommunikointivälineen käyttöliittymän ja tietoliikenne-moduulin välille. Sen kautta moduulille voidaan antaa komentoja, joko valmiilla funktioilla tai kirjoittamalla käskyt itse.

Valmiit fuktiot on toteutettu niin, että ne syöttävät parametrien mukaisen komennon tekstikenttään ja laukaisevat lähetä painikkeen toiminnon. Vasemmalla puolella oleva tekstialue näyttää lähetetyt käskyt ja takaisin tulleet kuittaukset. Näiden lisäksi siinä näkyy tietoja moduulien välisen yhteyden tilasta.

8 PUUNÄKYMÄT

Käyttöliittymään tehtiin kaksi erilaista puunäkymää helpottamaan järjestelmän havainnollistamista ja selaamista. Juurta lukuunottamatta kaikki puiden solmut ovat objekteja, jotka pitävät sisällään tiedon tekstistä sekä näytettävästä ikonista.

KUVIO 16. Rakennus- sekä keskukset ja valaisimet -puunäkymät

8.1 Rakennus-puunäkymä

Rakennusnäkymä rakennettiin toiminnollisesti samanlaiseksi kuin aikaisemmassa hallintaohjelmassa ollut puunäkymä on. Kuviossa 16 nähtävissä oleva vasemmanpuoleinenpuu. Näkymässä juurena toimii rakennus, johon järjestelmä on asennettu. Rakennuksen alle aukeaa kerrostasot, jotka taas koostuvat niissä sijaitsevista keskuksista sekä valoista. Valot ja keskukset asettuvat kerroksen alle niin, että

kaikki kerrokseen kuuluvat valot tulevat numerojärjestyksessä ensiksi ja sen jälkeen kyseisessä kerroksessa mahdollisesti sijaitsevat keskuksen samaan tapaan.

Jos valo- tai keskussolmun teksi on yliviivattu, tarkoittaa se sitä, että kyseistä osaa ei vielä ole sijoitettu karttanäkymään sen oikealle paikalle. Tämä näkymä mahdollistaa valaisimen poistamisen kartalta, ikonin muuttamisen sekä siirtämisen toinseen kerrokseen. Kaikki nämä toiminnot saadaan esille painamalla hiiren oikealla painikkeella halutun valaisimen päältä.

8.2 Rakennuspuun muodostus

Itse puu muodostetaan niin, että SQL-lauseella haetaan tietokannasta ensin rakennuksen kerrosten kokonaismäärä, minkä jälkeen jokaista kerrosta kohden haetaan siihen kuuluvat valot ja liitetään ne puuhun. Tämän jälkeen sama tehdään keskuksille. Koska jokainen kerros vaatii useamman kyselyn, on tämän puunäkymän muodostaminen raskasta ajettavalle säikeelle.

8.3 Keskukset ja valaisimet -puunäkymä

Keskukset ja valaisimet -näkymä näyttää kyseisessä kohteessa olevat keskukset sekä niihin kuuluvat valaisimet siten, että kunkin keskuksen alle aukeaa siihen kytkettynä olevat valaisimet. Rakenne muodostuu niin, että järjestelmässä kiinni olevat keskukset ja niiden alle sijoitetut valaisimet listataan numerojärjestyksessä.

8.4 Keskukset ja valaisimet -puun muodostus

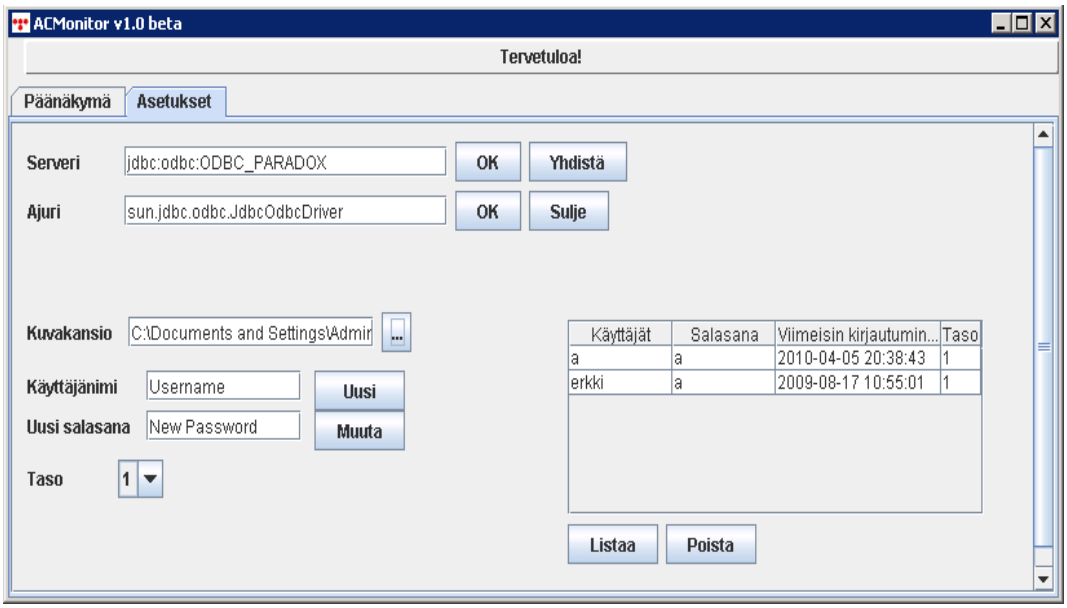
Puu muodostetaan käyttäen SQL-lauseita, joilla aluksi etsitään keskuksien määrä. Tämän jälkeen keskus lisätään solmuksi ja tehdään uusi SQL-kysely, josta saadaan siihen kuuluvat valaisimet. Tätä toistetaan, kunnes kaikki keskukset on käyty läpi.

9 KIRJAUTUMINEN

Käyttöliittymään on tehty mahdollisuus kirjautumiseen. Kirjautuminen avaa uuden välilehden käyttöön pääikkunatasossa. Tämä asetus-välilehti mahdollistaa serveri- ja ajuritietojen muuttamisen. Nämä tiedot eivät beta-versiossa tallennu asetustiedostoon, sillä käytössä oleva serveri ja ajuri oli aina sama.

Välilehti sisältää tiedostopolun kansioon, jossa käyttöliittymän karttatiedostoja säilytetään. Polkua muutettaessa avautuu esiin kansioselain, josta haluttu kansio saadaan helposti etsittyä. Kun polku on valittu, tallentuu se asetustiedostoon, josta ohjelma sen lukee seuraavilla käynnistyskerroilla.

Välilehdellä voidaan lisätä järjestelmään uusia käyttäjiä. Se tapahtuu kirjoittamalla tiedotkenttiin uusi käyttäjätunnus sekä salasana, minkä jälkeen painetaan uusi-painiketta, joka tallentaa tunnuksen tietokantaan. Käyttäjille voidaan asettaa tasoja välillä 1-3. Beta-versiossa tasoilla ei ole erityistä merkitystä, mutta ne on helppo ottaa käyttöön jatkoa ajatellen.



KUVIO 17. Asetus-välilehti

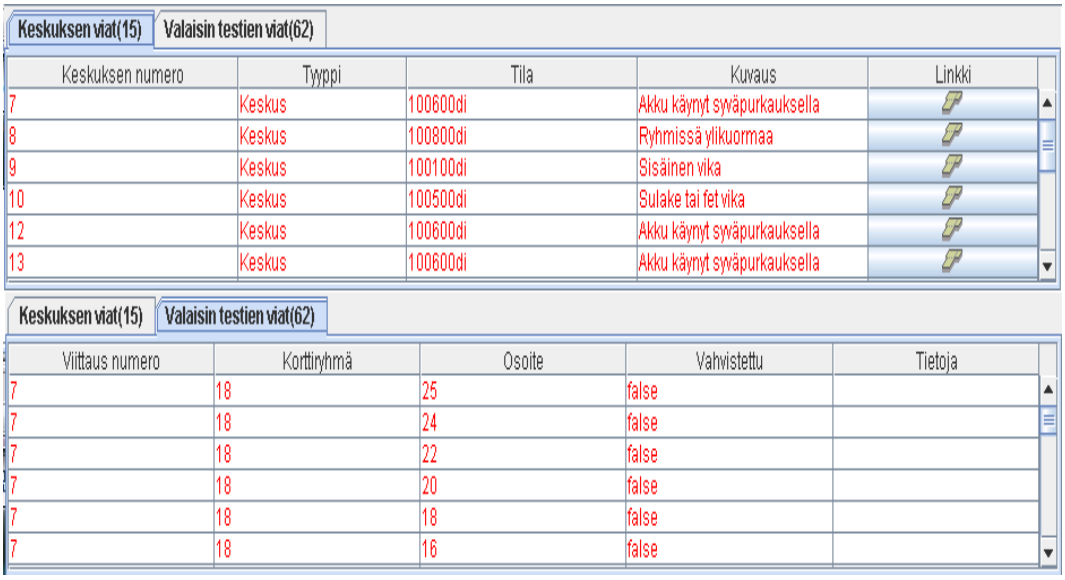
Kuviossa 17 oikealla näkyvässä listassa näkyy kaikki käyttäjät, salasanat, viimeisin kirjautuminen ja tunnuksen taso. Nämä eivät ole oletuksena esillä, mutta ne

saadaan ladattua tietokannasta painettaessa listaa painiketta. Beta-versiossa salasanat tallentuvat tietokantaan selkokielisenä.

10 VIKAILMOITUSRUUTU

Käyttöliittymän alimpana näkymänä on ruutu, jossa näkyvät kaikki järjestelmässä olevat aktiiviset viat. Vikailmoitukset on jaettu kahteen eri osioon. Ensimmäinen osio näyttää kaikki keskuksissa olevat viat ilmoittaen viallisen keskuksen numeron, vian tilan järjestelmäkoodina sekä vian kuvauksen selkokielisenä, sillä kielellä, joka on asetuksista valittuna. Vikaikkuna näyttää vain yhden vian keskusta kohden. Jos keskuksessa on useampi vika, tulevat ne näkyviin erillisen prioriteetin mukaan.

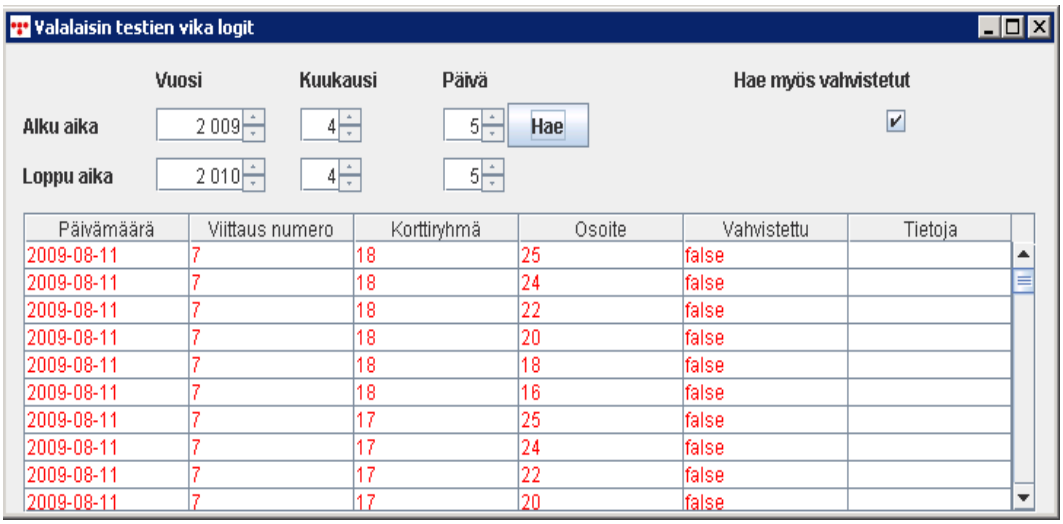
Vikailmoitusrivillä on myös linkkipainike ja se on sijoitettu linkki sarakkakeeseen. Linkkipainikkeen tarkoitus on helpottaa viallisen keskuksen paikallistamista siten, että sitä painettaessa päänäkymä siirtyy karttanäkymään asettaen näkyviin sen kerroksen, jossa viallinen keskus sijaitsee sekä asettaen keskuksen valituksi.



KUVIO 18. Vikailmoitus-näkymät

Toinen osio näyttää kaikki järjestelmän epäonnistuneet valaisintestit. Valaisintestien vikatiedoissa on näkyvissä testin viittausnumero, korttiryhmä, osoite kortilla, sekä se onko vikaa kuitattu, ja mahdolliset lisätiedot valaisimesta.

Molempien välilehtien nimitiedon perässä on sulkuihin sijoitettuna järjestelmässä sillä hetkellä olevien vikojen määrää.



KUVIO 19. Valaisintestien vikalogi

Valaisintestien logitietoja voidaan selata erilisestä ikkunasta, joka avautuu luvuissa 6.2.1 ja 6.3.1 mainituista logit-painikkeista. Ikkunan avautuessa ohjelma asettaa haettavat päivämäärät automaattisesti nykyhetkeen. Näitä aikoja muokkaamalla saadaan haettua tiedot eri aikaväleiltä. Oletuksena käyttöliittymä ei tuo näkyviin jo vahvistettuja vikoja, mutta nämä saadaan haettua asetettaessa niiden hakuehto valituksi.

11 MONIKIELISYYS

Koska Teknoware vie tuotteita myös ulkomaille, olisi ohjelmassakin hyvä olla mahdollisuus vaihtaa kieltä. Alun perin suunnitteilla oli, että kieliasetus sidottaisiin käyttäjätunnukseen, joka tallennettaisiin settings-tauluun omana sarakkeenaan. Kirjautumisen yhteydessä sovellus sitten lukisi käyttäjälle määritellyn maatunnuksen ja lataisi kielen sen mukaan.

Kielimuutos toteutettiin niin, että ohjelman jokainen sana otettiin ylös ja korvattiin erillisellä tunnuksella. Jokaista tunnusta kohden varataan oma muuttuja. Sanat kirjoitettiin lang.csv-tiedostoon siten, että jokainen rivi sisältää yhden kielivaihtoehdon sanat, sillä lisäyksellä, että tiedoston ensimmäinen sarake on varattu kielitunnukselle ja juuri tämä sarake näkyy käyttäjälle alasvetovalikossa päänäkymässä.

Beta-versiossa on mahdollisuus maksimissaan yhteensä 15 kielen valintaan, tätä asetusta on kuitenkin helppo laajentaa muuttamalla vain muutamia arvoja lähdekoodissa. Käytännössä kieliä on mahdollista vaihtaa mielivaltaisesti muokkaamalla lang.csv-tiedostoa tarvittavaan muotoon. Ensimmäisellä ajokerralla ohjelma latautuu englanninkielisenä, mutta kun kieliasetus vaihdetaan, se tallentuu erilliseen asetustiedostoon ja jää voimaan seuraaville käynnistyskerroille.

12 YHTEENVETO

Työssä saatiin toteutettua graafinen käyttöliittymä, joka sisälsi komponentit, joita siihen alun perin tahdottiin sisällyttää. Kiireisen aikataulun takia sitä ei kuitenkaan ehditty testata tarpeeksi ja sen viimeisin versio jäi sen osalta vaillinaiseksi.

Sovelluksen toimintaa voisi vielä parannella lisäämällä enemmän toimintoja, jotka olisivat salasanan takana. Näiksi toiminnoiksi sopisivat esimerkiksi järjestelmän rakenteen muokkaaminen käyttöliittymässä, sen jälkeen kun se on saatu asennettua loppuun. Ilman kirjautumista nämä ominaisuudet voisivat olla kokonaan poissa näkyvistä.

Myös sovelluskokonaisuutta voisi muunnella niin, että moduulit eivät keskustelisi suoraan keskenään, vaan sitä varten lisättäisiin tietokantaan lisää tauluja, joita käyttämällä käskyjä voitaisiin välittää molempiin suuntiin. Tämä menetelmä eheyttäisi alkuperäisen idean siitä, että tietokanta toimisi rajapintana moduulien välillä. Tietokantaformaattiakin kannattaisi ehkä vaihtaa Paradoxista esimerkiksi Oracleen, sillä sen kunnollinen tukeminen ei ole enää voimassa ja sille ei ole julkaistu mitään uutta kuluvan vuosituhannen aikana.

Käyttöliittymä jäi kaipaamaan useamman säikeen käyttöä, sillä esimerkiksi puunäkymien rakentaminen isoilla tietomäärillä saa pääsäikeen jäämään hetkellisesti jumiin, jolloin se ei vastaa käyttäjän toimintoihin. Tämä myös saisi itse ohjelman käynnistymään nopeammin ja mahdollistaisi esimerkiksi aloituskuvan käytön ohjelman avautuessa.

Kun käyttöliittymä on asennettu valmiiksi vastaamaan jonkin tietyn rakennuksen järjestelmää, voisi sen puurakenteet serialisoida, jonka jälkeen ei ohjelman käynnistämisen yhteydessä tarvitsisi tehdä lukuisia kyselyitä kantaan vaan puut saataisiin ladattua takaisin suoraan tiedostosta.

Käyttöliittymä on tällä hetkellä ylläpidossa ja ohjelman versio on 1.1.

LÄHTEET

Abstract Window Toolkit. 2010 Wikipedia [viitattu 11.4.2010]. Saatavissa: http://en.wikipedia.org/wiki/Abstract_Window_Toolkit

Hongisto, P. 2008. Käsikirja turvavalaistukseen [viitattu 12.4.2010]. Saatavissa: http://www.teknoware.fi/index.php?lang=1&nav=98

Java Tutorial 2010. Oracle Corporation [viitattu 30.3.2010]. Saatavissa: http://java.sun.com/docs/books/tutorial/uiswing/components/tree.html

JDBC 2010. Wikipedia [viitattu 30.3.2010]. Saatavissa: http://en.wikipedia.org/wiki/Jdbc

JDBC driver 2010. Wikipedia [viitattu 30.3.2010]. Saatavissa: http://en.wikipedia.org/wiki/JDBC_driver

Sisäasiainministeriön asetus. 2005 [viitattu 12.4.2010]. Saatavissa: http://www.finlex.fi/fi/laki/alkup/2005/20050805

Swing. 2010. Wikipedia [viitattu 30.3.2010] Saatavissa: http://en.wikipedia.org/wiki/Swing_%28Java%29

Teknowaren henkilöstökäsikirja. 2008. 5 painos. Lahti: Ei tiedossa.