Jussi Raitanen

SÄHKÖINEN OPASTUSTAULU

Tietotekniikan koulutusohjelma

Ohjelmistotekniikan suuntautumisvaihtoehto

2010

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

SÄHKÖINEN OPASTUSTAULU

Raitanen, Jussi
Satakunnan ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Marraskuu 2010
Ohjaaja: Kivi, Karri
Sivumäärä: 31
Liitteitä:

Asiasanat: Qt, C++, MySQL, ohjelmointi, ohjelmisto, relaatiotietokannat

---

Tämän opinnäytetyön tavoitteena oli suunnitella ja toteuttaa sähköinen opastusjärjestelmä erilaisiin tiloihin kuten liikkeisiin. Opastusjärjestelmä on interaktiivinen ja se toimii kosketusnäytöllä.

Järjestelmän ohjelmisto koostuu tietokannasta ja sen kanssa keskustelevista ohjelmista. Opastaja on tarkoitettu laitteeseen, jota asiakas käyttää kohteen etsimiseen. Muut ohjelmat ovat tarkoitettu järjestelmän hallintaan kuten karttojen piirtämiseen, ulkoasujen luomiseen ja asetuksien tallentamiseen.

Käyttöliittymät tehtiin C++ -ohjelmointikielellä Qt-ympäristössä.

Järjestelmän avulla kohteen löytää nopeammin ja vaivattomammin kuin itse etsimällä tai kysymällä apua henkilökunnalta. Kohteiden etsiminen onnistuu kahdella eri tavalla. Ensimmäinen on kategoriahaku, jossa käyttäjä etsii tuotteita kategorioiden perusteella. Toinen on vapaasanahaku, jossa käyttäjä kirjoittaa kohteen nimen tai osan siitä virtuaalisella näppäimistöllä.

AN ELECTRONIC GUIDANCE SYSTEM

Raitanen, Jussi
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in Information Technology
November 2010
Supervisor: Kivi, Karri
Number of pages: 31
Appendices:

Keywords: Qt, C++, MySQL, programming, software, relational databases

---

The purpose of this thesis was to design and implement an electronic guidance system for various locations such as shops. The system operates on an interactive touch screen.

The system software consists of a database and multiple smaller programs. Opastaja is the program that is used by customers and other users. The other programs are intended to manage the system, such as drawing maps, altering visual appearance and changing settings.

All of the graphical user interfaces were programmed with C++ using the Qt framework.

The system allows users to find objects faster and more easily than seeking themselves or asking the staff for help. Searching for items was carried out in two different ways. The first is a category search, in which the user searches for products based on categories. The second is a free text search where the user searches objects with keywords.

SISÄLLYS

Termiluettelo

CSS(*Cascading Style Sheet*) on erityisesti WWW-dokumenteille kehitetty tyyliohjeiden laji, mutta sitä voidaan käyttää myös muualla kuten Qt-kehitysympäristössä.

Relaatiotietokanta on tietotekniikassa käytetty termi tietovarastolle. Se on kokoelma tietoja, joilla on jokin yhteys toisiinsa. Tietokantojen koko voi vaihdella pienistä taulukoista miljooniin tietueisiin.

SQL(*Structured Query Language*) on IBM:n kehittämä standardoitu kyselykieli, jolla relaatiotietokantaan voi tehdä hakuja, muutoksia ja lisäyksiä.

MySQL on suosittu SQL-tietokantojen hallintajärjestelmä.

C++ on yksi tärkeimmistä kaupallisessa ohjelmistokehityksessä käytettävistä ohjelmointikielistä.

Subversion on version hallintajärjestelmä ja sen tarkoituksena on mahdollistaa ohjelmistojen muokkaamista hajautetusti niin, että kaikkien muokkaajien lokaalikopiot pysyvät ajan tasalla.

1 JOHDANTO

Kauppojen ja kauppakeskuksien koot ovat kasvaneet huimasti viime vuosina. Tällaiset suuret kaupat tai keskukset tarjoavat lähestulkoon kaikki tarpeelliset kulutustavarat kuluttajalle, mutta samalla vaikeuttavat yksittäisen tuotteen etsimistä. Suuren koon vuoksi katoissa olevat opastuskyltit eivät aina riitä löytämään tuotetta eikä työntekijöitä löydä opastamaan. Myös erillisen infopisteen puuttuminen vaikeuttaa tuotteen löytämistä.

Tuotteen löytämisen helpottamiseksi opinnäytetyönä lähdettiin kehittämään sähköistä opastustaulua, jonka avulla on helppoa löytää yksittäisiä tuotteita tai tuoteryhmiä kaupasta. Tällainen järjestelmä on ensimmäisiä Suomessa.

Tavoitteena oli kehittää ohjelmisto asiakkaan vaatimusten mukaan. Ohjelmisto tultaisiin asentamaan näkyvälle paikalle laitteeseen, joka voitaisiin sijoittaa eri tiloihin, joissa ihmisten on vaikea löytää haluttuun paikkaan. Esimerkkeinä voidaan mainita halutun kaupan löytäminen kauppakeskuksesta, hytin löytäminen risteilijältä sekä tuotteen löytäminen rautakaupasta.

Ohjelmiston ydinvaatimuksiksi asetettiin helppokäyttöisyys ja havainnollisuus, luotettavuus, ylläpidettävyys ja muokattavuus eri asiakkaiden tarpeisiin.

Ohjelmiston käyttötarkoituksena on ensisijaisesti henkilön ohjaaminen halutun kohteen luo, joka tapahtuu valitsemalla kohde kosketusnäytöltä. Kohde voi olla mikä tahansa ulkoisesta rakennuksesta kaupan hyllyllä olevaan tuotteeseen. Kohteen valitsemiseen on kaksi tapaa, tuotteen etsiminen kategorioiden perusteella tai käyttämällä vapaasanahakua. Vapaasanahakuun on toteutettu hieman muunneltu virtuaalinen QWERTY-näppäimistö, joten kirjoittamisen pitäisi olla vaivatonta.

Työ on ohjelmoitu kokonaan käyttäen ohjelmointikielenä C++:aa ja kehitysympäristönä Qt:ta. Tietokantana toimi MySQL, mutta ohjelmisto on kuitenkin suunniteltu niin, että sen laajentaminen tukemaan muitakin tietokantoja on helppoa.

2 TYÖVÄLINEET

2.1 QT-YMPÄRISTÖ

Qt on alun perin norjalaisen Trolltechin kehittämä alustariippumaton ohjelmistoalusta, nykyään sen omistaa Nokia. Qt on rakennettu C++:n päälle, mutta sitä voidaan käyttää kirjastojen avulla myös monella muulla ohjelmointikielellä kuten Python, Java, C#, Ruby, Perl, PHP. Olennainen osa Qt:tä on sen oma ohjelmistonkehitysympäristö, joka on vapaasti ladattavissa tuotteen web-sivustolta GPL- tai LGPL-lisensoinnilla. Kehitysympäristön asennuspaketti sisältää tarvittavat kirjastot, kääntäjät sekä työkalut /1/.

Qt ei rajoitu vain työpöydälle vaan sitä voidaan käyttää monen tyyppisiin alustoihin ja sovelluksiin sen alustariippumattomuuden vuoksi. Tuettuja alustoja ovat muun muassa Mac OS X, Windows, X11/Linux /2/ sekä Nokian kehittämä Maemo-mobiilialusta /3/. Vaikka enimmäkseen sitä käytetään graafisten työpöytäohjelmien tekemiseen, voidaan sitä käyttää myös ei-graafisten palvelinohjelmistojen ohjelmoimiseen. Nokian myötä Qt on laajentunut myös Symbian-alustalle. Qt siis sopii hyvin moneen paikkaan sekä ohjelmien siirrettävyys alustalta toiselle on nopeaa.

Qt-kehityspaketin voi hankkia jollain kolmesta lisenssistä: GPL-lisenssi, LGPL-lisenssi ja suljettu kaupallinen lisenssi. Kehityspaketeista on ladattavissa versiot Windows-, Linux-, ja Mac OSX –käyttöjärjestelmille.

GPL- ja LGPL-lisensoidut versiot työkaluista soveltuvat avoimen lähdekoodin kehittämiseen. Ohjelmien, joita kehitetään näillä lisensseillä, on myös oltava lisensoitu samalla lisenssillä. Näin ollen kaupallisten ja ennen kaikkea maksullisten ohjelmistojen kehitys näillä kahdella linsensointimallilla on varsin haastavaa. Lisenssin mukaan ohjelman lähdekoodi on oltava saatavilla. Kaupallistaminen onnistuu tästä huolimatta, tosin yleensä maksullisten tukipalvelujen ja ohjelmiston mukauttamisen kautta.

LGPL-lisensoituja kirjastoja voi käyttää suljettujen ohjelmistojen osana, mikäli ne voidaan päivittää uudempaan versioon isäntäohjelmasta riippumatta. Näin sovellusten kehittäjien ei tarvitse hankkia maksullista vaihtoehtoa, jos haluttu

toiminnallisuus löytyy LGPL-lisensoituna. GPL-lisenssi kieltää tällaisen kytkemisen kaupallisiin ohjelmistoihin. On myös muistettava, että koko Qt-alusta on näissä tapauksissa avointa lähdekoodia. Mikäli ohjelmiston kehittäminen vaatii käytetyn Qt-version muokkaamista, on nämä muutokset tehtävä julkisiksi.

Kaupallinen lisensointi mahdollistaa suljettujen, maksullisten ohjelmistojen kehityksen. Sitä Qt:n versiota, jonka lisensoinnin yhteydessä on hankkinut, saa muokata omiin tarpeisiin soveltuvaksi, eikä näitä muutoksia tarvitse tehdä julkisiksi.

Tässä opinnäytetyössä käsiteltävä ohjelmisto on kehitetty LGPL-lisensoidulla ohjelmistonkehityspaketilla.

2.2 MYSQL-OHJELMISTO

MySQL on levinnein avoimen lähdekoodin tietokantaohjelmisto. Se on tunnettu nopeudestaan, luotettavuudestaan ja työkalujensa helppokäyttöisyydestä. Tietokantaohjelmisto toimii monissa eri käyttöjärjestelmissä, esimerkiksi Windows, Linux, UNIX, Solaris sekä eri BSD-variaatiot. Avoimen lähdekoodin tietokantapalvelin tunnetaan nimellä MySQL Community Server.

MySQL-tietokantaa voidaan hallinnoida, joko komentoriviltä tai graafisesti MySQL Administrator -ohjelmalla.

MySQL on saatavilla sekä GPL-lisenssillä että kaupallisella lisenssillä. Perustason kaupallinen lisenssi, MySQL Enterprise Basic sisältää MySQL Enterprise Server -ohjelmiston sekä tuotetuen.

2.3 TORTOISESVN

TortoiseSVN on vapaan lähdekoodin versionhallintaohjelma, joka perustuu Subversioniin. Erityisen hyvän tortoisesvn-versionhallintaohjelmasta tekee sen integroituminen Windowsin exploreriin. Tällöin sen käyttö on helppoa, koska sen kaikki toiminnot löytyvät resurssienhallinnasta ja se on täysin graafinen.

TortoiseSVN on myös hyvin dokumentoitu.

2.4 MYSQL WORKBENCH

MySQL Workbench on graafinen työkalu tietokantojen suunnitteluun. Se yhdistää tietokantojen kehittämisen, hallinnoin, suunnittelun, luomisen ja ylläpitämisen yhdeksi ympäristöksi. Tässä työssä käytettiin MySQL Workbenchin versiota 5.0, joka on vain Windows-ympäristöihin. Tätä uudemmat toimivat jo muissakin käyttöjärjestelmissä.

2.5 MYSQL QUERY BROWSER

MySQL Query Browser on graafinen työkalu tietokantojen tekemiseen ja muokkaamiseen. Työkalulla onnistuu helposti taulujen luonti sekä rivien lisääminen, muokkaaminen ja poistaminen. Työkalulla on myös helppo luoda ja poistaa sarakkeita taulusta.

3 RELAATIOTIETOKANNAT

Relaatiotietokanta on kokoelma tietoja, joilla on jokin yhteys toisiinsa ja jotka on koottu yhteen paikkaan. Relaatiotietokannassa kaikki tieto on sijoitettu tauluihin ja tauluissa olevat tiedot esitetään riveinä ja sarakkeina ja jokaisessa rivissä on yhtä monta saraketta. Rivin yksilöi taululle asetettu pääavain, jossa pääavain voi koostua yhdestä tai useammasta taulun sarakkeen tiedosta, mutta sen on oltava kuitenkin yksikäsitteinen. Relaatiotietokannasta voidaan hakea tietoa sarakkeen arvon perusteella.

Taulusta voidaan viitata oman taulun toiseen sarakkeeseen tai toisen taulun johonkin sarakkeeseen. Viittauksien tarkoitus on säilyttää tietokannan eheys tietokantaa muokattaessa. Esimerkiksi tietokannan rivejä poistaessa on tärkeää, että kaikki siihen viittaava tieto poistetaan myös.

Monet arkiset asiat voidaan luokitella tietokannoiksi, vaikkei niin usein tehdäkään. Esimerkiksi kalenteri ja puhelinluettelo ovat esimerkkejä arkisista tietokannoista.

Relaatiotietokannan perustietotyyppeihin kuuluvat numeeriset tietotyypit, aika ja päivämäärä, vaihtelevan mittaiset merkkijonot ja binääri.

Relaatiotietokannoissa käytetty SQL-kyselykieli on IBM:n kehittämä standardoitu kyselykieli. Sen käytetyimmät käskyt ovat CREATE, DROP, SELECT, UPDATE, INSERT ja DELETE. /4/

CREATE-käskyllä voidaan luoda uusia tietokantoja, tauluja, näkymiä ja indeksejä.

DROP-käskyllä voidaan poistaa kokonaisia tietokantoja, tauluja, näkymiä ja indeksejä.

SELECT-käskyllä voidaan hakea tietoa tietokannasta.

UPDATE-käskyllä voidaan päivittää tietokannan rivien tietoja.

INSERT-käskyllä voidaan lisätä rivejä tauluun.

DELETE-käskyllä voidaan poistaa rivejä taulusta.

Esimerkkejä SQL-lauseista:

CREATE TABLE taulu;

DROP TABLE taulu;

SELECT * FROM taulu;

UPDATE taulu SET sarake=arvo WHERE sarake=muu arvo;

INSERT INTO taulu (sarake1,sarake2) VALUES (arvo1,arvo2);

DELETE FROM taulu WHERE sarake=arvo;

3.1 KÄSITEANALYYSI JA KÄSITEMALLI

Tietokannan suunnittelu aloitetaan käsiteanalyysillä. Sen tarkoitus on selvittää tarvittavat tiedot, jotka halutaan tallentaa tietokantaan. Tässä vaiheessa määritellään myös tietojen väliset suhteet ja riippuvuudet.

Käsiteanalyysi on vain karkea malli toteutettavasta tietokannasta. Sen ei ole tarkoitus olla lopullinen, eikä sitä kannata jatkaa liian pitkään. Analyysissä ei myöskään vielä mietitä teknistä toteutusta vaan vasta suunnittelun myöhemmissä vaiheissa. Lopuksi käsiteanalyysistä saadaan käsitemalli, jota voidaan edelleen parantaa. /5/

Käsitemalli on yleensä graafinen kaavio, joka kuvaa miten käsiteanalyysissä saadut tiedot liittyvät toisiinsa.

4 KÄYTTÖYMPÄRISTÖ

4.1 OHJELMISTOYMPÄRISTÖ

Vaikka ohjelmisto on käännetty myös unix-varianteille, tässä työssä käsitellään vain Windows XP -käyttöjärjestelmässä kehitettyä ohjelmistoa.

Käyttöjärjestelmäksi valittiin Windows XP sen yleisen käytön sekä helppokäyttöisyyden vuoksi.

Ohjelmisto koostuu muutamista yksittäisistä helppokäyttöisyyttä lisäävistä ohjelmista, tietokannasta, asiakkaalle näkyvästä sovelluksesta sekä ylläpitäjille tarkoitetusta sovelluksesta, jolla on tarkoitus päivittää ja lisätä kohteita kartalle. Työn helpottamiseksi tehtiin pieni ohjelma, jolla voi tarkastella ja tehdä ulkoasuja asiakasohjelmaan. Toinen tärkeä helppokäyttöisyyttä lisäävä ohjelma oli asetusten muokkain. Ohjelmiston toteutuskieleksi valikoitui C++ ja kehitysympäristöksi Qt:n versio 4.5. Tietokantasovelluksena toimi MySQL 5. Versiohallinnassa käytettiin TortoiseSVN-nimistä ohjelmaa.

4.2 LAITEYMPÄRISTÖ

Ohjelmisto voi toimia kahdessa eri tilassa. Ensimmäinen tila on itsenäinen, jolloin laitteessa ei tarvitse olla verkkoyhteyttä, mutta kaikkien vaadittujen sovellusten on oltava asennettuna paikalliseen laitteeseen. Tätä tilaa voidaan käyttää missä vain ja sen hyviä puolia on tiedonhaun nopeus ja vähäisemmät laitevaatimukset. Haittapuolina voidaan pitää sen vaikeampaa ylläpitoa, jolloin saman alueen jokaiseen laitteeseen on erikseen tehtävä muutokset.

Toinen tila on päätetila, jolloin laitteeseen ei tarvitse asentaa muuta kuin asiakkaille tarkoitettu ohjelma. Tämä asettaa minimaaliset laitevaatimukset asiakaslaitteelle, mutta vastaavasti nostaa palvelimen laitevaatimuksia. Tämän tilan hyviä puolia on keskitetty hallinta, jolloin muutokset tarvitsee tehdä vain yhteen paikkaan ja jokainen siihen kytketty pääte päivittää automaattisesti itse tietonsa. Huonoina puolina voidaan pitää ylimääräisen liikenteen muodostumisen, jolloin verkkoyhteyksien pitää olla riittävän nopeat.

5 TIETOKANTALIITTYMÄ

Tietokantaliittymä tehtiin erillisellä apuluokalla WSql, joka pitää sisällään kaiken tietokantoihin ja sen käsittelyyn liittyvät tiedot. Kaikki tieto tietokannan ja ohjelmien välillä kulkee tämän luokan kautta.

Ennen kuin luokkaa voidaan käyttää kyselyiden tekemiseen, on tietokantayhteys avattava. Yhteys yritetään muodostaa seuraavalla funktiolla.

void createConnection(const char* server, const char* db, const char* user, const char* password);

- server, osoite palvelimeen, jossa tietokanta sijaitsee
- db, tietokannan nimi
- user, käyttäjätunnus tietokantaan
- password, salasana tietokantaan

Tämän jälkeen on vielä tarkistettava, että onnistuiko yhteyden avaaminen. Se tapahtuu bool isActive() – funktiolla.

Luokka on valmis käytettäväksi, jos isActive() palauttaa arvon true.

Tiedon päivittäminen muihin komponentteihin käy kätevästi requestUpdate() – funktiolla. Se lähettää signaalin eteenpäin, jos jotain päivitettävää löytyi, muutoin ei tarvitse tehdä mitään.

```
void WSql::requestUpdate() {
QString statement = "SELECT `time` FROM LAST_UPDATE;";
this->execute(statement);
this->query->seek(0);
if(firstUpdate) {
lastUpdate = this->query->value(0).toString();
firstUpdate = false;
return;
}
//päivitetään, jos kannassa on uudempaa tietota kuin itse ohjelmassa
if(lastUpdate < this->query->value(0).toString()) {
lastUpdate = this->query->value(0).toString();
updating = true;
emit updateSignal();
}
}
```

Koodi 1 requestUpdate-funktio

Vaikka kaikki tiedonhakeminen voitaisiin tehdä suoraan void execute(QString text) – funktion kautta, luokkaan laitettu joitakin projektispesifisiä funktioita, joita voidaan käyttää vain tässä yhteydessä ja niiden tarkoitus on helpottaa koodin luettavuutta.

```
33      void resetDatabase();
34
35      int getLastPointId();
36      int getLastPolygonId();
37      QMap<int, QMap<int, QPoint> > getPoints();
38      QMap<int, QMap<int, QList<int> > > getWalls(const QMap<int, QMap<int, QPoint> > points);
39      QMap<int, QMap<int, QList<int> > > getPolygons(const QMap<int, QMap<int, QPoint> > points);
40      QMap<int, QString> getPolygonTypes();
41      QMap<int, QMap<QString, QString> > getLocations();
42      QMap<int, QMap<QString, QString> > getCategories();
43      QMap<int, QMap<QString, QString> > getItems();
44      QMap<int, QMap<QString, QString> > getStaticItems();
45      QMap<int, QString> getBackgrounds();
```

Kuva 1 WSql:n projektispesifiset funktiot

Kuvassa 1 näkyy tietokantaliittymän projektispesifiset funktiot. Funktioiden nimistä näkee sen, mitä funktio tekee.

6 OHJELMISTOPROJEKTIN VAIHEET

6.1 SUUNNITTELU

6.1.1 SUUNNITTELUPERUSTEET

Suunniteltaessa ohjelmistoa oli otettava huomioon mukautuminen. Tämä tarkoitti sitä, että ohjelmistoa lähdettiin tekemään modulaariseksi, jolloin ominaisuuksien lisääminen, poistaminen ja muokkaaminen helpottuisivat.

Monikielisyyden tukea kehitettiin alusta asti, ettei se aiheuttaisi ongelmia lähdekooditasolla. Kuitenkin ohjelmistosta tehtiin vain englanninkielinen versio.

Ohjelman lähdekoodia oli dokumentoitava luettavuuden ja ylläpidettävyyden vuoksi.

Lähdekoodin oli oltava yksinkertaista, mutta riittävän tehokasta. Toiminnot ja luokat pyrittiin tekemään mahdollisimman yksinkertaisiksi ja havainnollisiksi. Luokkien ja funktioiden oli oltava nimetty sen mukaan, mitä se teki tai mitä se piti sisällään.

6.1.2 TIETOKANTASUUNNITTELU

Tietokannan suunnittelu on tärkeää, jotta tietokannasta tulisi hyvä ja käyttökelpoinen. Suunnittelun pääperiaatteina on tiedon toiston välttäminen, koska toistumiseen liittyy ongelmia kuten turhan tilan vieminen ja ylläpidon hankaloituminen. /6/ Tiedon toistumista vähennetään normalisoimalla tietokantaa. Normalisoinnilla tarkoitetaan taulujen pilkkomista pienempiin osiin ja yhteyksien lisäämistä näiden välille.

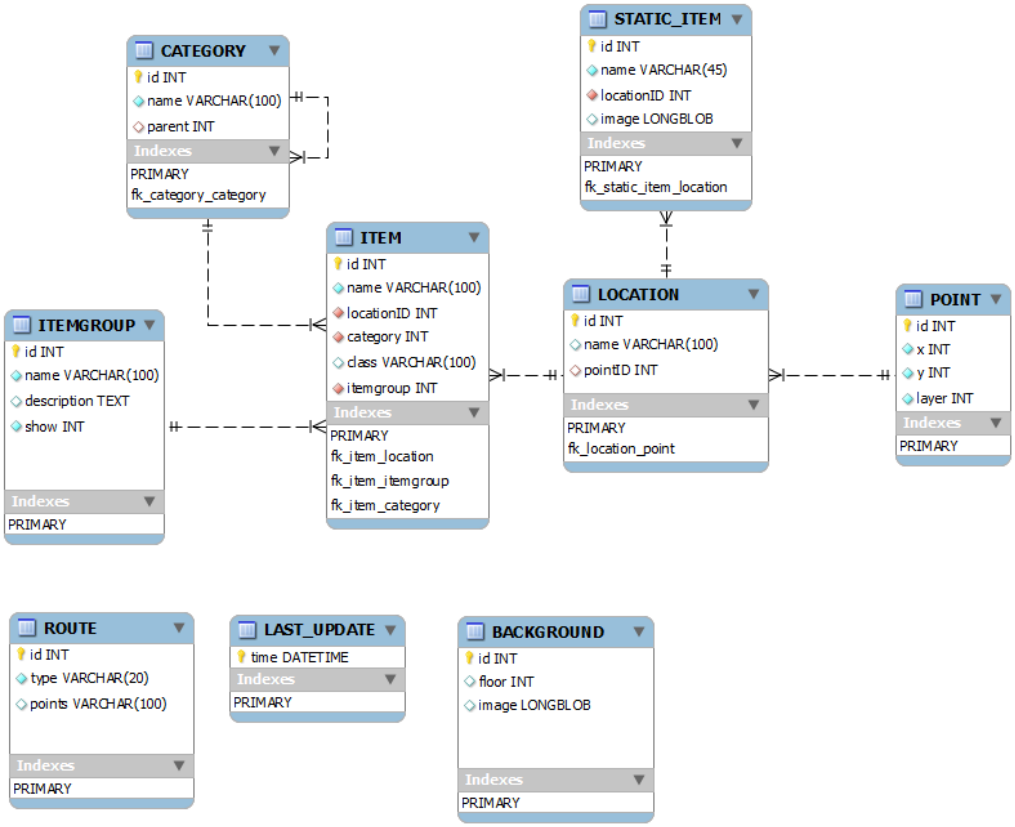
Tässä työssä tietokanta piti suunnitella hyvin, koska koko ohjelmisto käyttää sitä hyväkseen. Tietokanta suunniteltiin heti alusta asti laajennettavaksi, joten se mukautuu eri ympäristöihin helposti ja kätevästi.

Tiedot ovat jaettu omiin tauluihin redundanssin vähentämiseksi. Näin myös tiedon hakeminen on helpompaa.

6.2 TOTEUTUS

6.2.1 TIETOKANNAN TOTEUTUS

Alkuperäisen suunnitelman mukaan kaikki tieto olisi nojautunut pistetietoihin ja reittityyppeihin, mutta tämän lisäksi tietokantaan lisättiin uusi taulu, BACKGROUND, johon voitiin asettaa tietylle kerrokselle tietty taustakuva. Muuten tietokanta on pysynyt koko ohjelmiston kehityksen ajan samanlaisena.



Kuva 2 Infotaulun tietokantarakenne

Kuvassa 2 näkyy tietokannan rakenne. Kaikki taulujen väliset yhteydet ovat yhden-suhde-moneen. Esimerkiksi kohde (ITEM) voi olla vain yhdessä lokaatiossa kerrallaan, mutta yhdessä lokaatiossa voi olla monta kohdetta.

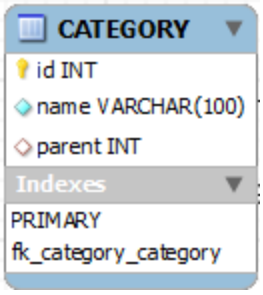
CATEGORY-taulu poikkeaa muista tauluista hieman siinä, että se viittaa itseensä. Kun taulu linkittää kaksi omaa sarakettaan toisiinsa, saadaan puumainen rakenne, jossa lapsielementit kuuluvat johonkin isäntään. Tästä on se hyöty, että sitä voidaan käyttää rekursiivisesti ja isäntää ei voi linkittää väärin.

Tietokanta luotiin ensin käyttäen MySQL Query Browseria, jolla myös asetettiin alkuarvot. Tämän jälkeen luoduista tauluista tehtiin SQL-kielen mukaiset tiedostot, jotta tietokanta ja taulut voitaisiin myöhemmin luoda ohjelmallisesti.

Alla esimerkki ITEM-taulun luomisesta SQL-kielellä.

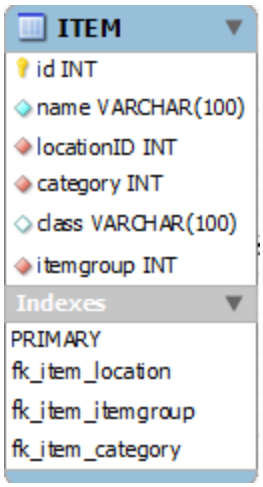
```
1   CREATE  TABLE IF NOT EXISTS `citidigi`.`ITEM` (
2     `id` INT NOT NULL AUTO_INCREMENT ,
3     `name` VARCHAR(100) NOT NULL ,
4     `locationID` INT NOT NULL ,
5     `category` INT NOT NULL ,
6     `class` VARCHAR(100) NULL ,
7     `itemgroup` INT NOT NULL ,
8     PRIMARY KEY (`id`) ,
9     INDEX `fk_item_location` (`locationID` ASC) ,
10    INDEX `fk_item_itemgroup` (`itemgroup` ASC) ,
11    INDEX `fk_item_category` (`category` ASC) ,
12    CONSTRAINT `fk_item_location`
13      FOREIGN KEY (`locationID` )
14      REFERENCES `citidigi`.`LOCATION` (`id` )
15      ON DELETE NO ACTION
16      ON UPDATE NO ACTION,
17    CONSTRAINT `fk_item_itemgroup`
18      FOREIGN KEY (`itemgroup` )
19      REFERENCES `citidigi`.`ITEMGROUP` (`id` )
20      ON DELETE NO ACTION
21      ON UPDATE NO ACTION,
22    CONSTRAINT `fk_item_category`
23      FOREIGN KEY (`category` )
24      REFERENCES `citidigi`.`CATEGORY` (`id` )
25      ON DELETE NO ACTION
26      ON UPDATE NO ACTION)
27  ENGINE = InnoDB;
```

Koodi 2 ITEM –taulun luonti MySQL-kielellä



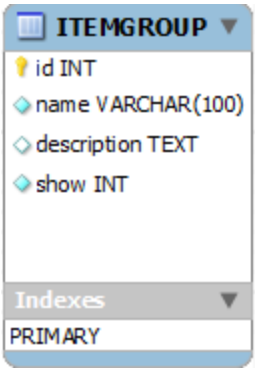
Kuva 3 CATEGORY-taulun rakenne

CATEGORY-taulu koostuu rivikohtaisesta id-numerosta, kategorian nimestä ja mahdollisesta isäntäkategoriasta. Taulussa on aina rivi, jonka id on nolla, nimi root ja jolla ei ole isäntää. Tätä riviä ei näytetä asiakaspäätteellä mitenkään, mutta karttoja tehdessä vähintään yhden kategorian on laitettava isännäkseen id-numeron nolla.



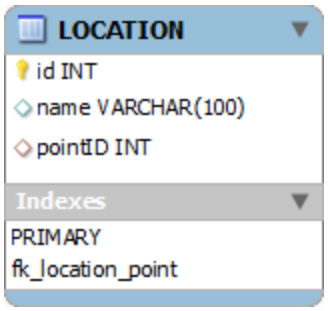
Kuva 4 ITEM-taulun rakenne

ITEM-taulu koostuu rivikohtaisesta id-numerosta, kohteen nimestä, lokaatiotiedosta, kategoriatiedosta, luokasta ja ryhmästä. Nimi on se tieto joka näytetään asiakkaalle, lokaatio on viittaus LOCATION-taulun pääavaimeen, joka kertoo millä alueella kyseinen kohde sijaitsee, kategoria ilmaisee sen mihin kategoriaan kohde kuuluu. Luokkaa ei käytetä missään, mutta se suunniteltiin tarjoamaan laajennusta ja yksilöivää tietoa kohteista, mikäli niin haluttaisiin. Ryhmän tarkoitus on tarjota helppo tapa toteuttaa kohteen näyttäminen ja piilottaminen asiakkaalta poistamatta kohdetta tietokannasta. Tätä toimintoa voidaan käyttää esimerkiksi sesonkiaikana sesonkituotteille.



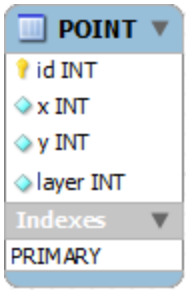
Kuva 5 ITEMGROUP-taulun rakenne

ITEMGROUP-taulu koostuu rivikohtaisesta id-numerosta, luettavammasta nimestä, ryhmän kuvauksesta ja näkyvyysalueesta. Ryhmälle annetaan nimi, jotta sitä olisi helpompi käyttää; käytössä vain kartanpiirto-ohjelmassa. Kuvaus kertoo tarkemmin miksi ryhmä on luotu, sitä ei ole kuitenkaan pakko käyttää. Ryhmän näkyvyysalue määritellään sarakkeella show, eikä se voi saada muita arvoja kuin 0 tai 1, jossa nolla piilottaa ryhmän. Asiakkaan kannalta tämä merkitsee samaa kuin kohteita ei olisi lainkaan, eikä näin ollen voi niitä näytöltä valita.



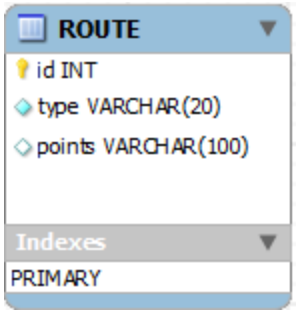
Kuva 6 Lokaatiotaulun rakenne

LOCATION-taulu koostuu rivikohtaisesta id-numerosta, lokaation nimestä ja yhdestä pisteestä. Lokaatiolle voidaan antaa nimi, jotta se löydettäisiin helpommin kartanpiirto-ohjelmassa. pointID-kenttä on viittaus POINT-taulun id-numeroon. Viittauksen avulla tiedetään mihin lokaatio piirretään koordinaatistossa.



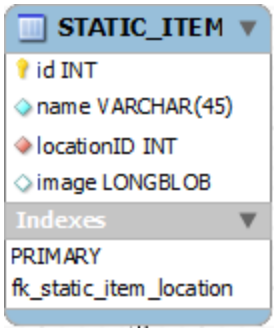
Kuva 7 POINT-taulun rakenne

POINT-taulu koostuu rivikohtaisesta id-numerosta, xy-koordinaatiston pistetiedoista ja kerroksesta. Tietokannassa olevat koordinaatit ovat raakatietoa, jotta koordinaatiston skaalaaminen ja transformointi onnistuisi kätevästi. Kerros ilmoittaa yksinkertaisesti mihin kerrokseen piste kuuluu.



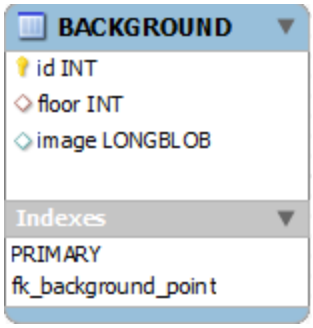
Kuva 8 ROUTE-taulun rakenne

ROUTE-taulu koostuu rivikohtaisesta id-numerosta, tyypistä ja pistetiedosta. Reitin tyyppi on pakollinen, mutta sen mahdolliset arvot eivät ole ennalta määrättyjä, koska ohjelmiston on käsiteltävä tieto aina eritavalla tyypistä riippuen. Tämä aiheuttaa muutoksia lähdekooditasolla. Työssä käytetyt reittityypit ovat wall eli seinät ja way eli reitit, joita pitkin voi kulkea. Points-kenttä ilmaisee mitkä pisteet kartalla kuuluvat samaan reittiin. Pisteet ovat erotettu toisistaan pilkuilla.



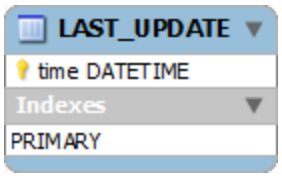
Kuva 9 STATIC_ITEM-taulun rakenne

STATIC_ITEM-taulu koostuu rivikohtaisesta id-numerosta, nimestä, lokaatiotiedosta ja kuvasta. Tauluun on laitettu nimikenttä pakolliseksi, jotta kuvia olisi helpompi etsiä, muokata ja poistaa, sillä ilman suuntaa antavaa nimeä toiminto olisi hyödytön jälkeenpäin. Lokaatiotieto taulussa ilmaisee sen mihin tiettyyn kohtaan kuva piirretään. Sarakkeessa image on kuvadata. Taulun tarkoitus on tarjota staattisia kuvia, jotka piirretään aina kartalle; näitä ovat esimerkiksi poistumistiet.



Kuva 10 BACKGROUND-taulun rakenne

BACKGROUND-taulu koostuu rivikohtaisesta id-numerosta, kerroksesta ja kuvasta. Kerros on viittaus POINT-taulun sarakkeeseen id. Viittaus kertoo sen mihin kerrokseen taustakuva ladataan. Sarakkeessa image on kuvadata. Taulun tarkoitus on pitää tieto kerroksien taustakuvista.



Kuva 11 LAST_UPDATE-taulun rakenne

LAST_UPDATE-taulu on tietokannan yksinkertaisin taulu. Se sisältää vain yhden kentän, time, jonka tarkoituksena on kertoa milloin viimeksi tietokantaan tehtiin muutoksia.

Koska ohjelmisto ei sisällä erillistä palvelinohjelmaa, joka tietäisi muutokset tietokannassa ja lähettäisi ne eteenpäin Opastajalle, päivitys oli hoidettava jotakin muuta kautta jolloin päädyttiin erillistauluun. Ilman erillistä päivitystaulua tietoa siirrettäisiin turhaan palvelimen ja asiakaspäätteen välillä, koska kummankaan pään tiedon eheydestä ei voisi olla varma.

6.2. 1 OPASTAJAN TOTEUTUS

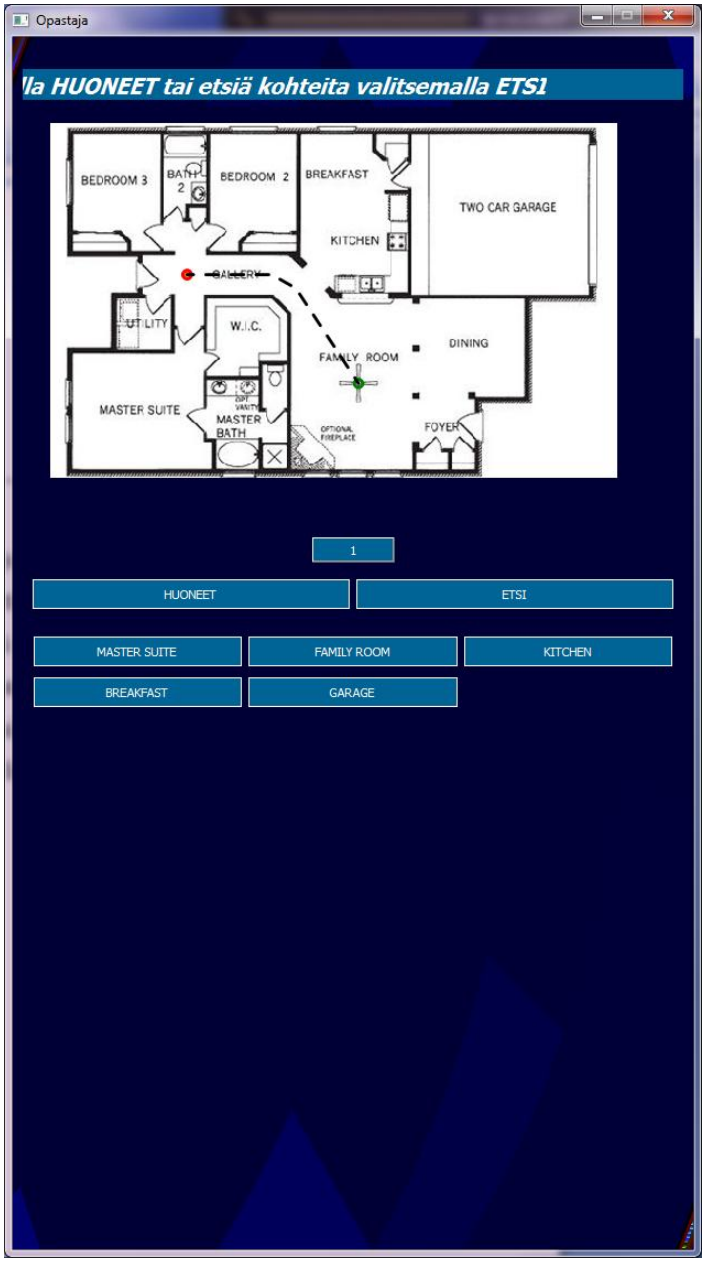
Opastaja on asiakaspään ohjelma, joka yhdistää tietokantaan, lataa uusimmat paikkatiedot, piirtää kartan ruudulle ja tarvittaessa ohjaa halutun kohteen luokse.

Opastajan ulkoasu on muuten täysin muokattavissa, mutta painikkeiden järjestystä ei voi muuttaa. Ulkoasut luodaan omalla ohjelmistoon kuuluvalla tyylieditorilla. Se käyttää teemoihin omaa Opastajalle tarkoitettua style-formaattia, joka muunnetaan ajoaikana CSS:ksi.

```
class=WButton
object=Item
state=pressed
description=Item button, pressed
background-color=darkgray
border-style=outset
border-width=0px
border-color=none
color=white
width=150
height=25
```

Koodi 3 Painikkeen tyylimäärittely

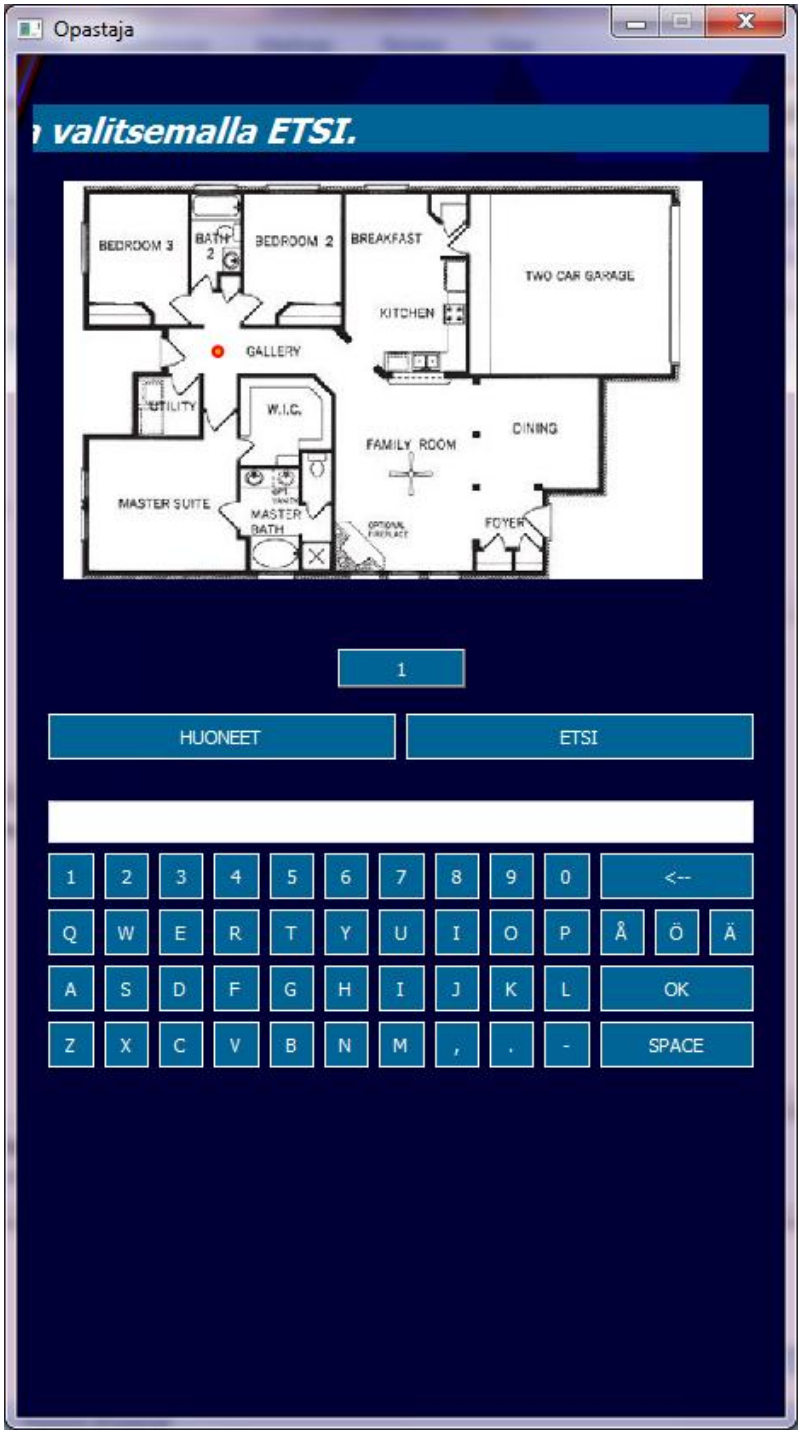
Koodissa 3 näkyy miten tyylitiedosto on kirjoitettu. ”class” määrittää mille C++ -luokalle tyylittely asetetaan. ”object” määrittää minkä nimisille olioille tyylittelyt asetetaan. ”state” määrittää sen, milloin tyyli otetaan käyttöön. Koodissa 3 painiketta painaessa sen väri muuttuu tummanharmaaksi, painikkeen palautuessa normaaliin tilaan sen väri myös vaihtuu takaisin. ”description” on lyhyt kuvaus, mitä kyseinen tyylitiedosto tekee. Loput tyylitiedostosta muunnetaan CSS:ksi.



Kuva 12 Kuvankaappaus Opastajasta

Opastajan ulkoasu koostuu taustakuvasta, marquee-tekstistä, kartasta, kerrospainikkeista, hakupainikkeista (HUONEET ja ETSI) ja kohteista (MASTER SUITE, FAMILY ROOM, KITCHEN, BREAKFAST, GARAGE).

Kuvassa 12 näkyy Opastaja toiminnassa. Kuvassa on ensiksi haettu huoneita ja sen jälkeen painettu painiketta FAMILY ROOM, jolloin käyttäjä on ohjattu sen sijaitsemaan paikkaan. Punainen piste kartalla on lähtöpiste ja vihreä piste kohde. Katkoviivat liikkuvat laitteen sijainnista kohteeseen joten epäselvyyttä ei synny minne pitäisi lähteä.

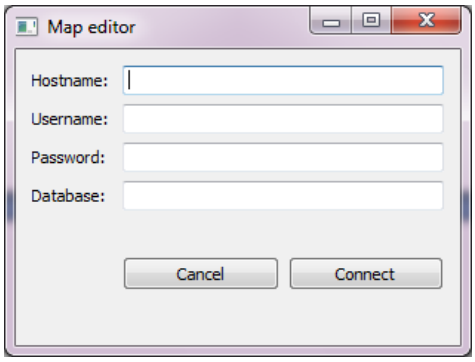


Kuva 13 Muokattu QWERTY-näppäimistö

Kuvassa 13 näkyy ohjelmaan paremmin sopivaksi muokattu QWERTY-näppäimistö.

6.2.1 KARTTAEDITORIN TOTEUTUS

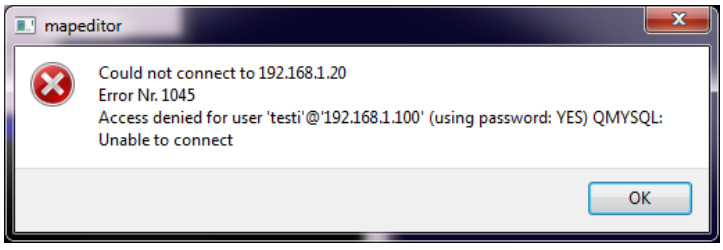
Ohjelman käynnistyessä ensimmäiseksi kirjaudutaan tietokantapalvelimelle (Kuva 14). Käyttäjä kirjautuu suoraan omilla MySQL-tunnuksillaan.



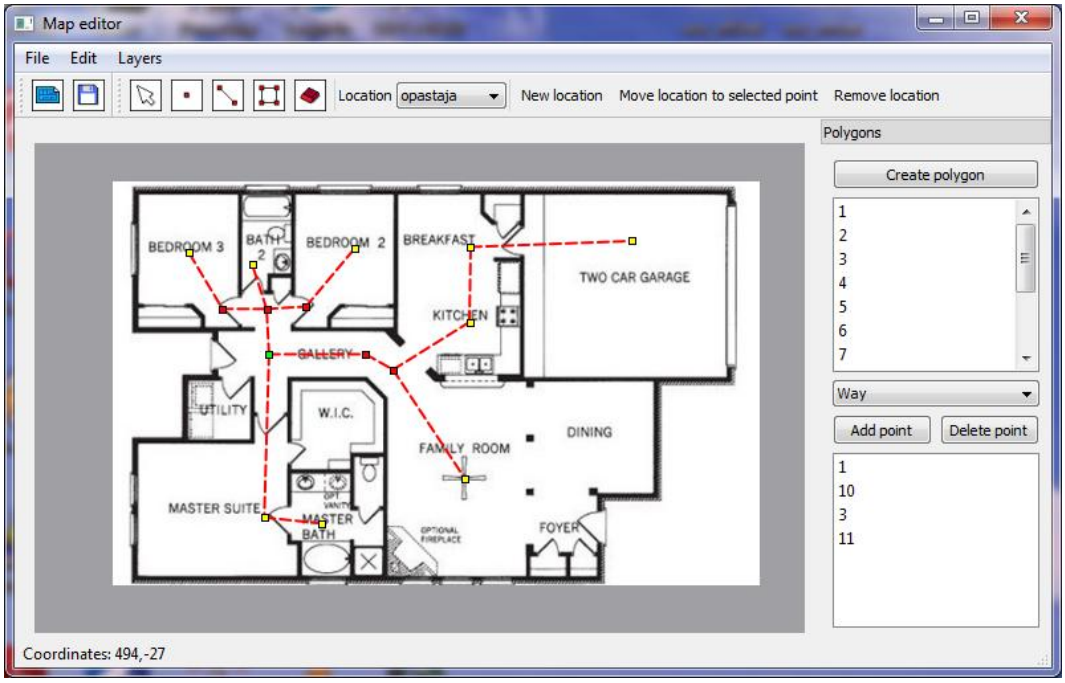
Kuva 14 Karttaeditorin kirjautumisikkuna

Kuvassa 14 näkyy karttaeditorin kirjautumisikkuna. Kirjautumisen yhteydessä myös tarkistetaan onko tietokanta ehjä vai ei. Mikäli tietokanta on rikki, pyydetään käyttäjältä lupa ylikirjoittaa perusasetukset tietokantaan.

Kirjautumisen epäonnistuessa näytetään kuvan 15 mukainen virheilmoitus.



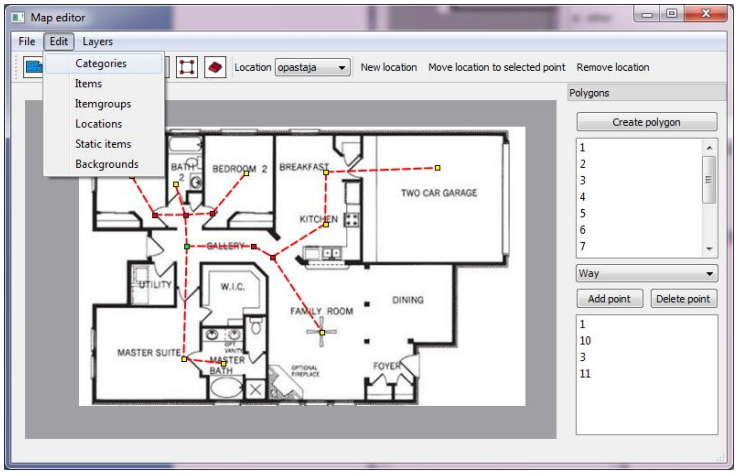
Kuva 15 Epäonnistunut kirjautuminen



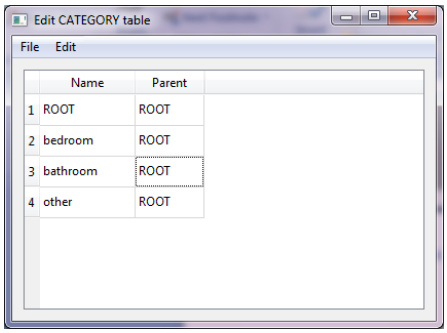
Kuva 16 Kuvankaappaus karttaeditorista

Onnistuneen kirjautumisen jälkeen näytetään kuvan 16 mukainen perusnäkymä karttaeditorista.

Tietoja voidaan alkaa syöttää Edit-valikosta (Kuva 17). Toimiakseen järjestelmä vaatii ainakin yhden itemgroupin, johon tietty kohteet lisätään ja pisteen johon lokaatio kiinnitetään. Kartan yhden pisteen lokaatio on oltava sama kuin opastajaan konfiguroitu lokaatio, jolloin Opastaja kiinnittää itsensä siihen.

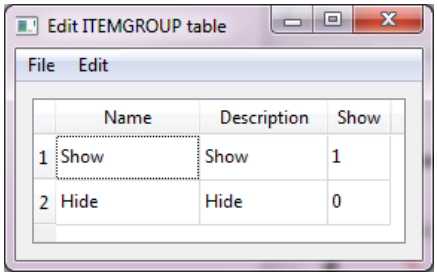


Kuva 17 Editointi-ikkunan valitseminen



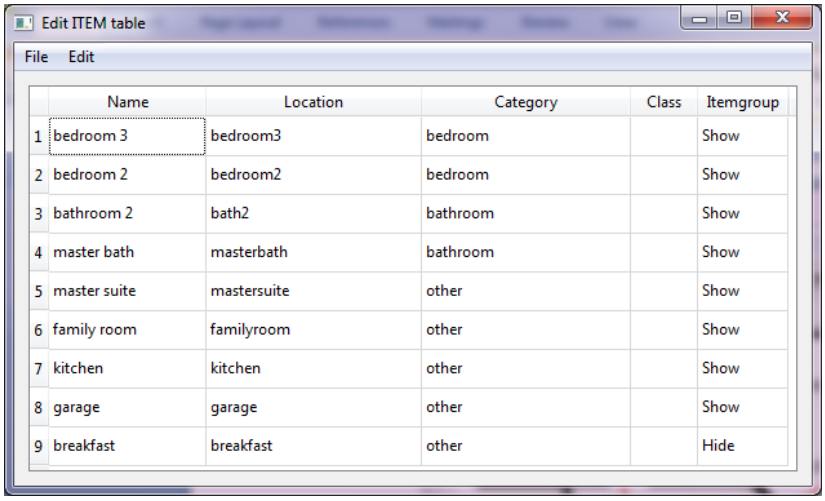
Kuva 18 Kategorioiden muokkausikkuna

Kuvassa 18 on lisätty 3 kategoriaa, bedroom, bathroom ja other, joihin voidaan lisätä kohteita.



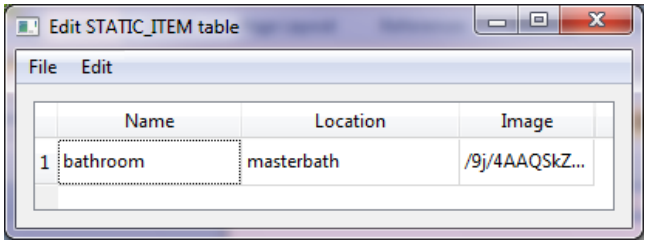
Kuva 19 Ryhmien lisääminen

Kuvassa 19 on lisätty kaksi ryhmää, Show ja Hide, jotka voidaan asettaa kohteille. Name on ryhmän nimi, joka näkyy kun kohteille asetetaan näkyvyysalue. Description on kuvaus ryhmästä ja Show kertoo näkyvyyden. Luku 1 näyttää kohteen ja luku 0 piilottaa kohteen kartalta.



Kuva 20 Järjestelmän sisältämät kohteet

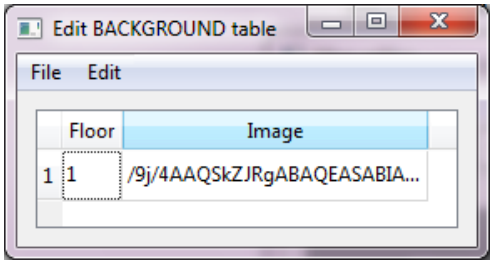
Kuvassa 20 näkyy järjestelmän sisältämät kohteet. Name on kohta joka näkyy käyttäjälle kohteiden selaamisvaiheessa. Location määrittää sen mihin pisteeseen kohde on kiinnitetty. Category määrittää sen, minkä kategorian alla kohde näytetään. Saraketta Class ei käytetty työssä, mutta sitä voidaan käyttää tulevaisuudessa kohteen tarkempaan määrittelyyn. Itemgroup määrittää kohteen näkyvyysalueen.



Kuva 21 Staattiset kuvat

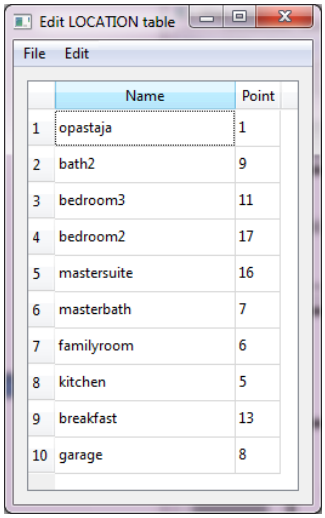
Kuvassa 21 on asetettu staattinen kuva lokaatioon masterbath. Staattiset kuvat ovat kuvia, jotka piirretään kartalle aina. Staattisilla kuvilla saadaan kartalle enemmän grafiikkaa jolloin kartasta tulee näyttävämpi ja helpommin lähestyttävä.

Name-kenttä on tarkoitettu kuvaamaan sijoituspaikkaa ja Location-kenttä määrittää kuvan sijainnin. Image-kenttä on itse kuva. Kuva lisätään tuplaklikkaamalla image-kenttää, jolloin avautuu ikkuna, josta voi ladata kuvan palvelimelle.



Kuva 22 Kerroksen taustakuva

Kuvassa 22 on asetettu kerrokseen yksi taustakuva. Sen lisääminen tapahtuu samalla lailla kuin staattisen kuvan lisääminen, tuplaklikataan Image-kenttää, jolloin aukeaa ikkuna, josta voi ladata kuvan palvelimelle.



Kuva 23 Järjestelmän lokaatiot

Kuvassa 23 näkyy tietokantaan lisätyt lokaatiot. Name-kenttä on lokaation nimi, joka näkyy kaikissa muissa muokkausikkunoissa, joissa käytetään lokaatiotietoa hyväksi.

Muokkausikkunaa käytetään lähinnä lokaatioiden poistamiseen tai nimen muokkaamiseen. Lokaatioiden lisääminen on mahdollista, mutta hankalaa, koska pitäisi tietää mihin pisteeseen lokaatio kiinnitetään.

6.3 TESTAUS

Ohjelmiston testauksen tarkoitus on löytää virheitä ohjelmistosta. Käytännössä jokaisesta ohjelmistosta löytyy virheitä ja niiden havaitseminen on hankalaa. Koska ohjelmiston kehittäjät tietävät, miten ohjelmiston tulee toimia, he eivät testaa ohjelmistoa samoin kuin ulkopuoliset testaajat. Testaamisen tulee olla hajoittava, jolloin virheitä voidaan korjata, tämä taas voi aiheuttaa lisää virheitä. Tämän vuoksi testaus pitäisi suorittaa lopullisella laitteistolla ja erillisen testausryhmän avulla.

Ohjelmistoa testattiin suljetussa ympäristössä, ja vain kehittäjät toimivat testaajina. Kriittisiä virheitä ei löytynyt.

Testaamista ei voitu suorittaa loppuun asti, koska työn tilaajalla ei ollut tarvittavia laitteita.

7 YHTEENVETO

Ohjelmistosta tuli paljon laajempi kuin alun perin oli suunniteltu, mutta oli järkevämpää toteuttaa ohjelmistoon erilliset komponentit, joita voidaan laajentaa helposti tukemaan myös uusia asioita. Erilliset ohjelmat asetustietojen ja ulkoasun muokkaamiseen nähtiin järkevänä ratkaisuna, koska nämä ohjelmat käsittelevät täysin eri asioita.

Ohjelmistosta saatiin tehtyä helppokäyttöinen ja yksinkertainen. Helppokäyttöisyyttä lisäävät myös kuvaavat ikonit toiminnoille.

Jatkokehityksen kannalta ohjelmisto saatiin tehtyä hyvään pisteeseen. Ohjelmisto toimii hyvin ja sitä voidaan käyttää, mutta siitä puuttuu joitakin hyödyllisiä ominaisuuksia. Näitä ominaisuuksia ovat mm. automaattinen seinien tunnistus ja parhaan reitin laskenta-algoritmiin painoarvot. Painoarvoilla saataisiin hieman mielekkäämmät reitit vaikka se ei olisikaan lyhin reitti kohteeseen.

Ohjelmiston testaamista ei voitu suorittaa sen lopullisessa ympäristössä, koska työn tilaajalla ei ollut tarvittavia laitteita eikä resursseja.

LÄHDELUETTELO

1. Qt – Downloads [Viitattu 28.03.2010] http://qt.nokia.com/downloads
2. Qt Cross-Platform Application Framework [Viitattu 28.03.2010] http://qt.nokia.com/products/platform/files/pdf/qt-cross-platform-application-framework-datasheet
3. Qt - Maemo [Viitattu 28.03.2010] http://qt.nokia.com/products/platform/maemo
4. SQL [Viitattu 14.06.2010] http://fi.wikipedia.org/wiki/SQL
5. Tietokantojen suunnittelu [Viitattu 10.10.2010] http://student.labranet.jamk.fi/~huojo/opetus/IIZO3020/IIZO3020m3.pdf
6. Tietokantasuunnittelusta [Viitattu 10.10.2010] https://www.cs.helsinki.fi/u/tkujala/tikapeS2003/kalvot/pdf/tksu.PDF