OULUN SEUDUN
AMMATTIKORKEAKOULU

Kristina Zharnikova

TYÖNVASTAANOTTOJÄRJESTELMÄ

TYÖNVASTAANOTTOJÄRJESTELMÄ

Kristina Zharnikova
Opinnäytetyö
29.2.2012
Tietotekniikan koulutusohjelma
Oulun seudun ammattikorkeakoulu

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu
Tietotekniikan koulutusohjelma, Ohjelmistojen tuotanto

Tekijä: Kristina Zharnikova

Opinnäytetyön nimi: Työnvastaanottojärjestelmä

Työn ohjaajat: Jukka Häkkilä, Oulun Datapalvelut Oy;
Pekka Alaluukas, Oulun seudun ammattikorkeakoulu

Työn valmistumislukukausi ja vuosi: kevät 2012 Sivuja:32 Liitteitä:1

TIIVISTELMÄ

Tämän opinnäytetyön tarkoitus oli suunnitella ja toteuttaa työnvastaanottojärjestelmän kehittäminen Oulun Datapalvelut Oy:lle. Järjestelmän avulla yrityksen henkilökunta voi tallentaa asiakkaan tiedot, tallentaa tiedot asiakkaan huollon tuomasta koneesta ja tarvikkeista. Lisäksi järjestelmästä voidaan tulostaa käteiskuitti ja lasku. Järjestelmän avulla voi tallentaa kaikki vianselvitysvaiheet ja huollossa tehdyt toimenpiteet.

Sovellus toteutettiin Qt-ohjelmointiympäristössä, joka sisältää C++-luokkakirjaston ja graafiseen käyttöliittymään liittyvät luokat ja on helpottanut käyttöliittymän suunnittelua ja luomista. Tietokanta toteutettiin MySQL:n avulla.

Opinnäytetyön tuloksena syntyi työnvastaanottojärjestelmä, joka sisältää graafisen käyttöliittymää ja tietokannan, johon voidaan tallentaa asiakas- ja työntekijätietoja. Opinnäytetyön sovelluksen lähdekoodia ja tietokantaa yritys voi hyödyntää jatkokehityksessä.

SISÄLTÖ

1 JOHDANTO

Opinnäytetyön tavoite on suunnitella ja kehittää työnvastaanottojärjestelmän Oulun Datapalvelut Oy:lle. Työnvastaanottojärjestelmän tarkoitus on tallentaa yrityksen, henkilökunnan, asiakkaiden ja huoltoon vastaanotettavien laitteiden tiedot, kuvaukset viasta ja toimenpiteistä, tiedot laitteiden mukana tulleista ohjelmista ja tarvikkeista.

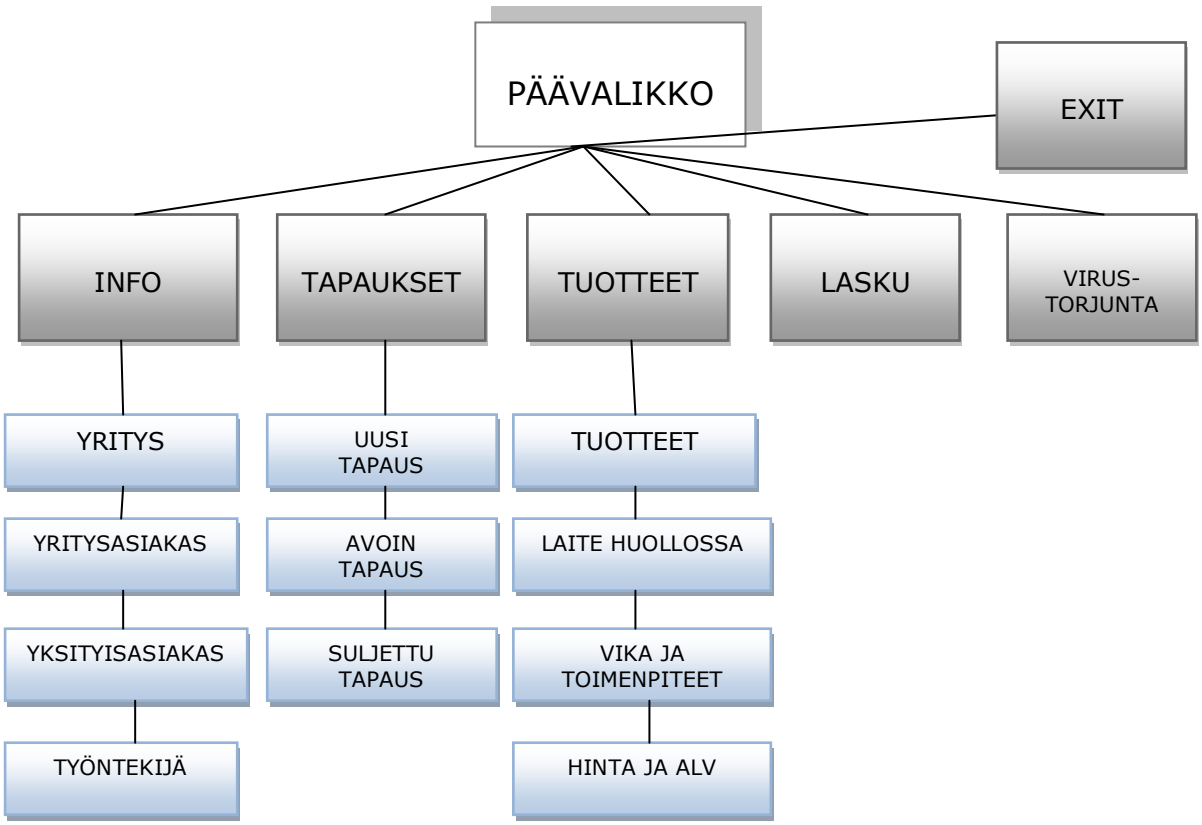
Käyttöliittymä on tehty mahdollisimman yksinkertaiseksi ja helppokäyttöiseksi. Ohjelmointiympäristöksi valittiin Qt-ohjelmointiympäristö, josta löytyy monipuolisen käyttöliittymän suunnittelutyökalut. Käyttöjärjestelmänä toimii Windows-käyttöjärjestelmä. Jatkokehityksessä on ajatus käyttää ohjelmaa myös Linux-käyttöjärjestelmässä. Tämä oli yksi syy, miksi Qt valittiin ohjelmointiympäristöksi.

Työn tilaaja on Oulun Datapalvelut Oy. Oulun Datapalvelut Oy on pieni yritys, joka on toiminut vuodesta 2007 asti. Yritys tarjoaa asiakkaille mahdollisimman kaikenkattavaa ja henkilökohtaista palvelua: atk-korjaus ja huolto, tiedonpalautus ja tietoturva; laitteistojen, ohjelmistojen ja järjestelmien asennus sekä päivitys ja nettiongelmien ratkaisua. (Oulun Datapalvelut Oy.)

2 TYÖNVASTAANOTTOJÄRJESTELMÄ

Työnvastaanotto-ohjelman tarkoituksena on helpottaa henkilökunnan työn vastaanottoa sekä tuotteiden, asiakkaiden, yrityksen ja työntekijöiden tietojen käsittelyä ja työn laskutuksen hoitamista (kuva 1).

Työn vastaanotetuille tiedoille ja tapahtumille luodaan oma tapausnumero. Tapausnumeron perusteella tallennetaan koko työtapaushistoria. Niin voidaan seurata työn prosessia eli milloin työ on vastaanotettu, kuka on asiakas, mitä koneessa on vialla, minkälaiset toimenpiteet pitää suorittaa, mitkä ovat tuotteiden hinnat, milloin työn pitää olla valmis ja kuka on työn suorittaja. Kaikki nämä tiedot käyttäjä käyttöliittymän avulla tallentaa tietokantaan. Työtä suorittaessa työtapaus suljetaan ja kaikki työtapauksen tiedot liitetään laskuun ja tulostetaan käteiskuittina tai viitelaskuna.



KUVA 1. Ohjelman rakenne

3 OHJELMOINTIYMPÄRISTÖ

Ohjelmointi on tehty Qt-kehitysympäristöllä ja Qt Creator 2.0.1-työkalulla. Tietokanta on suunniteltu käyttämällä MySQL Workbench 5.2 CE-työkalua ja tietokantapalvelimena toimii MySQL Server 5.1.

3.1 MySQL

MySQL on hyvin suosittu avoimen lähdekoodin relaatiotietokannan hallintajärjestelmä, koska se on ilmainen ja saatavilla melkein kaikilla alustoilla. Se tarjoaa erittäin korkean suorituskyvyn, luotettavuuden ja helppokäyttöisyyden. MySQL kehitettiin hallitsemaan suuria tietomääriä hyvin suurella nopeudella ongelmien voittamiseksi. (MySQL. 2012.)

MySQL käyttää kommunikointiin muiden ohjelmien kanssa SQL-kieltä. SQL on lyhenne sanoista Structured Query Language eli kyselykieli. SQL on standardi, kun käsitellään ja haetaan tietoja relaatiotietokannasta. (MySQL. 2012.)

MySQL on ominaisuuksiltaan monipuolinen, joustava, nopea ja tehokas. MySQL sopii sekä pieniin että isoihin järjestelmiin. MySQL noudattaa niin sanottua asiakas-palvelin-arkkitehtuuria eli sovellus ei kommunikoi suoraan palvelimen kanssa vaan aina asiakkaan kautta. (MySQL Cookbook.)

MySQL eroaa monista tietokannanhallintajärjestelmistä myös siinä, että MySQL:ssä tietokannan ydin on erotettu tauluja käsittelevästä koodista. Tätä osaa voidaan kutsua tietokantamoottoriksi. MySQL:ään voi ladata useita eri tietokantamoottoreita, jotka vaikuttavat ratkaisevasti sen käyttäytymiseen.

Qt ei tue suoraan MySQL:ää. Tässä työssä tarvittiin MySQL Driver Plugin. Plugin luodaan GCC-kääntäjällä. Windowsille on olemassa

MinGW-niminen paketti (mingw-utils-o.3), joka sisältää mm. kyseisen kääntäjän. Kopioidaan mingw-utils-0.3/bin-hakemiston sisältö Qt/../qt/bin-hakemistoon. Avataan Qt:n pääte ja mennään hakemistoon Qt/../qt/lib/opt ja suoritetaan komento reimp libmysql.lib, jotta tuottaa liblibmysql.a-tiedosto. Tämä on tärkeä kirjaston tiedosto käytössä MinGW:n kanssa.

Sen jälkeen mennään Qt/../qt/src/plugins/sqldrivers/mysql-hakemistoon ja suoritetaan komento qmake -o Makefile "INCLUDEPATH+=C:\mysql\include" "LIBS+=C:\mysql\lib\opt\liblibmysql.a" mysql.pro. Tämän jälkeen suoritetaan komento mingw32-make, joka luo ajuritiedostot qsqlmysql4.dll, libqsqlmysqld4.dll, qsqlmysql4.a ja libqsqlmysqld4.a Qt/../qt/plugins/sqldrivers-hakemistoon. Lopussa kopioidaan libmysql.dll-tiedosto Qt/../qt/bin-hakemistoon.
Nyt liitännäisen pitäisi latautua kunnolla.

3.2 Qt

Qt on alun perin Trolltech Oy:n kehittämä alustariippumattomien ohjelmistojen kehitysympäristö, jonka Nokia osti vuonna 2008. Qt:lla voi luoda alustariippumattomia graafisia ja tekstipohjaisia käyttöliittymiä. Qt:n siirrettävyys on mahdollistanut sen saatavuuden useille alustoille, kuten Windows 95/98/NT/2000, Linux, Maemo, Mac OS X, Sun Solaris, HP-UX, Digital Unix, ja monet muut. (Qt.)

Qt:n oliopohjainen malli tarjoaa tehokkaan ja nopean sovelluksen kehittämisen sekä rajoittamattomia mahdollisuuksia ja nopean nykyaikaistamisen. Qt rikastaa C++-kieltä helppokäyttöisten funktioiden lisäksi myös tehokkailla makroilla eli Meta-Object Compilereilla. Meta-Object Compiler on Qt:n taustalla toimiva sovellus, joka huolehtii Qt:n sisäänrakennnetuista C++:n lisäyksistä. (The Meta-Object System.)

Sovelluksessa käytetään QtCore-luokkakirjastoa, jonka kaikki Qt-sovellukset tarvitsevat. Mitkään QtCore-kirjaston luokista eivät vaadi graafista käyttöliittymää, mikä mahdollistaa käyttöliittymän erottamisen liiketoimintalogiikasta. (QtCore Module.)

Qt:ssä on automaattinen muistihallinta, joka vaatii, että luokka periytyy QObject-luokasta. Mekanismi toimii niin, että kaikki lapsioliot poistetaan ja niiden muisti vapautetaan automaattisesti, kun vanhempi poistetaan muistista. Tämä mekanismi helpottaa ohjelmoijan työtä. (QObject Class Reference.)

Qt:n GUI on laaja valikoima komponentteja. Kaikki GUI-luokat on peritty QWidget-luokasta, joka puolestaan on peritty luokasta QObject. QWidget-luokassa on paljon hyödyllistä widgettejä valmiina. Ulkoasun suunnittelussa käytetään Qt Designer-työkalua. (QObject Class Reference.)

Qt:n tapahtumankäsittely on viestinvälitystekniikka sovelluksen sisäisten olioiden välillä ja kahden olion ei tarvitse tietää toisistaan yhtään mitään. Siinä käytetään signals/slot-mekanismia, jonka käyttö vaatii Q_OBJECT-makroa. Slotit toteutetaan luokan jäsenmetodina. Signaalit toteutetaan esikääntäjän generoimana. Jokainen olio pitää omaa listaa yhteyksistään. Slotin yhteydet signaaleihin rakennetaan connect()-metodilla, jonka määrittely tehdään Q_OBJECT-makrolla. (QObject Class Reference.)

4 TIETOKANTA

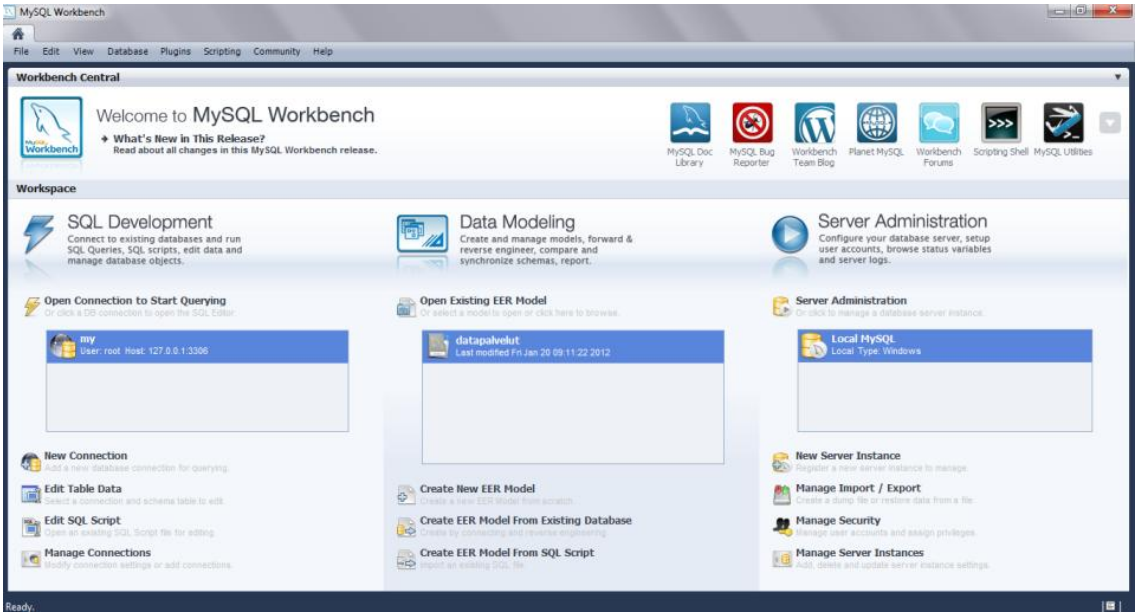
Tietokantapalvelin on MySQL Server 5.1. Tietokannan suunnitteluun ja hallinnointiin on käytetty MySQL Workbench 5.2 CE-työkalua. Tietokanta on luotu helpottamaan järjestelmän kaikkien tietojen käsittelyä.

Tietokantaa voi muokata MySQL Workbench 5.2 CE- ja phpMyAdmin-työkalujen avulla. Järjestelmän tietokannan tietojen hakemisen ja muokkaamisen pystyy tekemään ohjelman käyttöliittymän avulla.

4.1 MySQL Workbench

MySQL Workbench (kuva 2) on Oraclen kehittämä graafinen hallintatyökalu MySQL-tietokantoja varten. MySQL Workbench-työkalun ominaisuudet:

- *SQL Development* mahdollistaa yhteyksien ja istuntojen määrittelyn, SQL-kyselyjen sekä taulujen rakenteen ja sisällön muokkaamisen.
- *Data modeling* tarjoaa kehittyneitä työkaluja tietokannan mallintamiseen eli tuen EER-kaavioille (enhanced entityrelationship diagram).
- *Server Administration* on koottu tietokannan yleisten asetusten hallinnointi. Sen avulla voi hallinnoida mm. käyttäjätilejä, instansseja, tietoturvaa, lokitiedostoja ja tietojen siirtämistä eri tietokantojen välillä. (MySQL Workbench 5.2.)



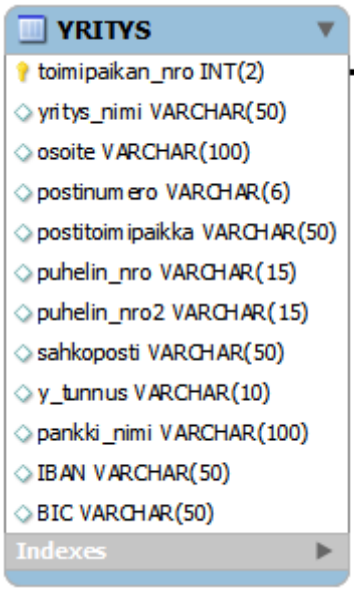
KUVA 2. Näkymä MySQL Workbenchistä

4.2 Tietokannan taulut

Ohjelman tietovarastona käytetään MySQL-tietokantaa, jonne varastoidaan tarvittavat tiedot. Tietokanta sisältää 17 taulua. Tietokannan EER-kaavio löytyy liitteestä 1. Tietokannan taulujen sarakkeiden tietotyypit on nimitetty teksti- tai numerotiedon mukaan. NOT NULL-arvo on annettu sen mukaan, että sarakkeen arvo ei voi olla puuttuva tieto eli NULL-arvo.

Yritys-taulu

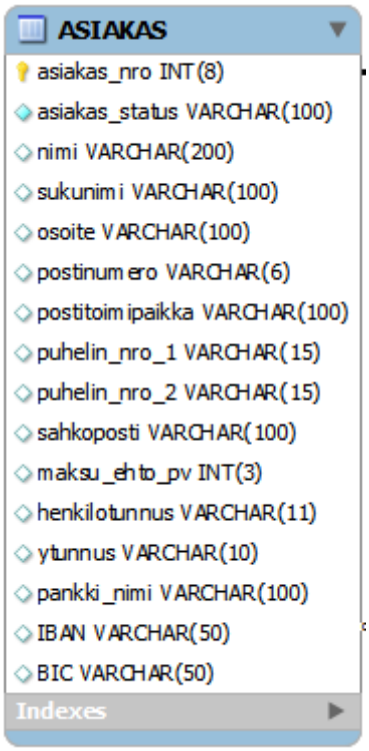
Yritys-taulu sisältää yrityksen (Oulun Datapalvelut Oy) tiedot. Taulussa on 12 saraketta: toimipaikan_nro, yritys_nimi, osoite, postinumero, postitoimipaikka, puhelin_nro, puhelin_nro2, sahkoposti, y-tunnus, pankki_nimi, IBAN ja BIC (kuva 3).



KUVA 3. Yritys-taulun rakenne

Asiakas-taulu

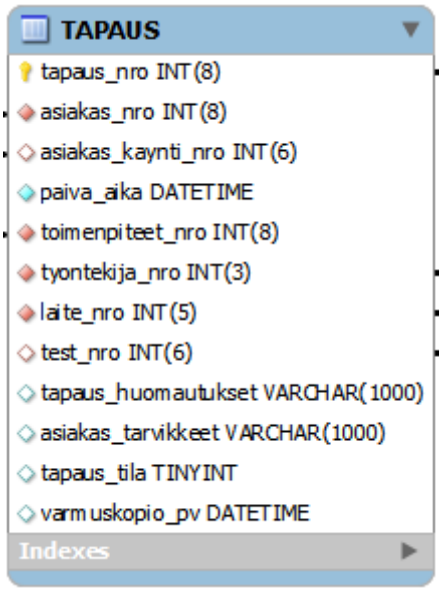
Asiakas-taulu sisältää yrityksen asiakkaiden tiedot. Taulussa on 12 saraketta: asiakas_nro, asiakas_status, nimi, sukunimi, osoite, postinumero, postitoimipaikka, puhelin_nro_1, puhelin_nro_2, sahkoposti, maksu_ehto, henkilotunnus, ytunnus, pankki_nimi, IBAN ja BIC (kuva 4).



KUVA 4. Asiakas-taulun rakenne

Tapaus-taulu

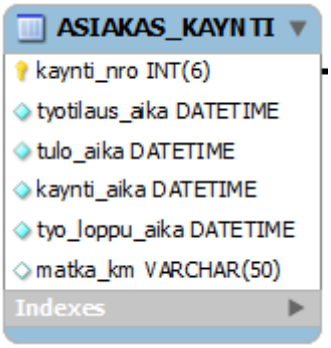
Tapaus-taulu sisältää työtapauksien tiedot. Taulussa on 11 saraketta: tapaus_nro, asiakas_nro, asiakas_kaynti, paiva_aika, toimenpiteet_nro, tyontekija_nro, laite_nro, test_nro, tapaus_huomautukset, asiakas_tarvikkeet, tapaus_tila ja varmuuskopio_pv (kuva 5).



KUVA 5. Tapaus-taulun rakenne

Asiakas_kaynti-taulu

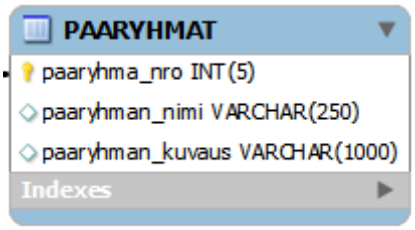
Asiakas_kaynti-taulu sisältää tiedot yrityksen tehdyistä asiakaskäynneistä. Taulussa on 6 saraketta: kaynti_nro, tyotilaus_aika, tulo_aika, kaynti_aika, tyo_loppu_aika ja matka_km (kuva 6).



KUVA 6. Asiakas_kaynti-taulun rakenne

Paaryhmat-taulu

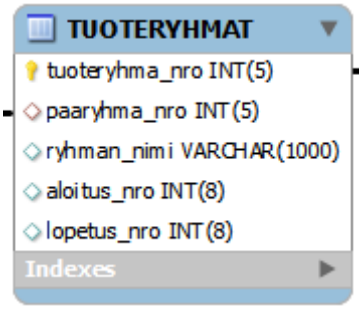
Paaryhmat-taulu sisältää tuotteiden pääryhmien tiedot. Taulussa on 3 saraketta: paaryhman_nro, paaryhman_nimi ja paaryhman_kuvaus (kuva 7).



KUVA 7. Paaryhmat-taulun rakenne

Tuoteryhmat-taulu

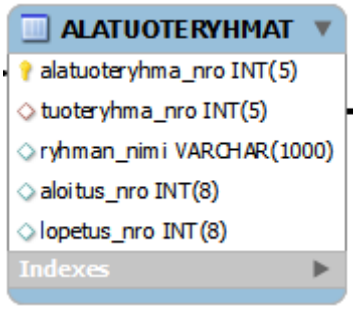
Tuoteryhmat-taulu sisältää tuoteryhmien tiedot. Taulussa on 5 saraketta: tyoteryhma_nro, paaryhma_nro, ryhman_nimi, aloitus_nro ja lopetus_nro (kuva 8).



KUVA 8. Tuoteryhmat-taulun rakenne

Alatuoteryhmat-taulu

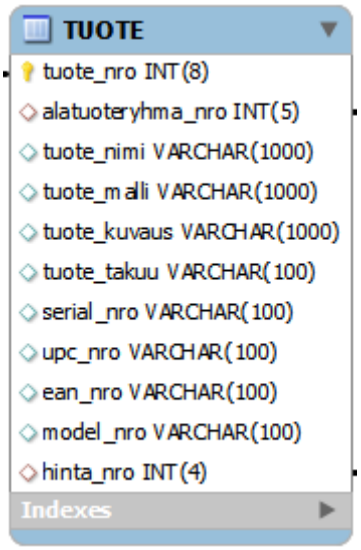
Alatuoteryhmat-taulu sisältää alatuoteryhmien tiedot. Taulussa on 5 saraketta: alatuoteryhma_nro, tyoteryhma_nro, ryhman_nimi, aloitus_nro ja lopetus_nro (kuva 9).



KUVA 9. Alatuoteryhmat-taulun rakenne

Tuote-taulu

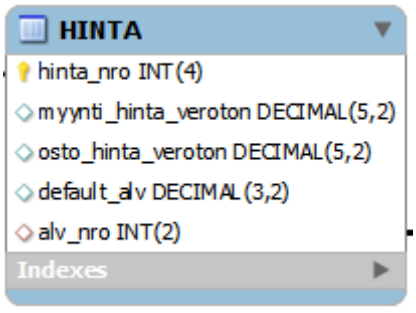
Tuote-taulu sisältää tuotteiden tiedot. Taulussa on 11 saraketta: tuote_nro, alatuoteryhma_nro, tuote_nimi, tuote_malli, tuote_kuvaus, tuote_takuu, serial_nro, upc_nro, ean_nro, model_nro ja hinta_nro (kuva 10).



KUVA 10. Tuote-taulun rakenne

Hinta-taulu

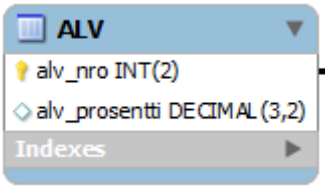
Hinta-taulu sisältää tuotteiden ja palveluiden hinnaston tiedot. Taulussa on 5 saraketta: hinta_nro, myynti_hinta_veroton, osto_hinta_veroton, default_alv ja alv_nro (kuva 11).



KUVA 11. Hinta-taulun rakenne

Alv-taulu

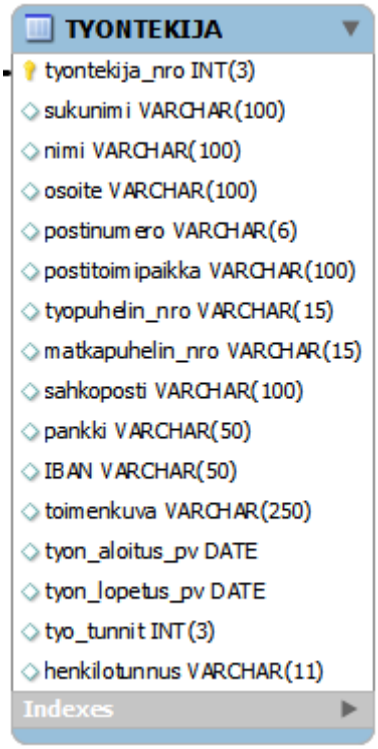
Alv-taulu sisältää alv-prosenttien tiedot. Taulussa on 2 saraketta: alv_nro ja alv_prosentti (kuva 12).



KUVA 12. Alv-taulun rakenne

Tyontekija-taulu

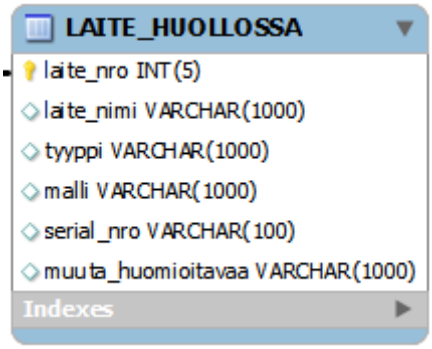
Tyontekija-taulu sisältää yrityksen työntekijöiden tiedot. Taulussa on 14 saraketta: tyontekija_nro, sukunimi, etunimi, osoite, postinumero, postittoimipaikka, tyo_puhelin, matkapuhelin_nro, sahkoposti, pankki, IBAN, toimenkuva, tyon_aloitus_pv, tyon_lopetus_pv ja tyo_tunnit (kuva 13).



KUVA 13. Tyontekija-taulun rakenne

Laite_huollossa-taulu

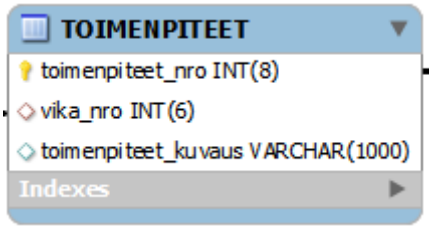
Laite_huollossa-taulu sisältää tiedot huollossa olevista laitteista. Taulussa on 7 saraketta: laite_nro, tapaus_nro, laite_nimi, tyyppi, malli, serial_nro ja muuta_huomiotavaa (kuva 14).



KUVA 14. Laite_huollossa-taulun rakenne

Toimenpiteet-taulu

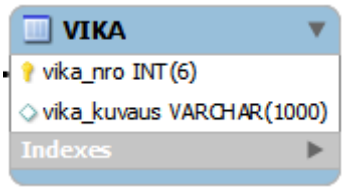
Toimenpiteet-taulu sisältää huoltotyön toimenpiteiden tiedot. Taulussa on 3 saraketta: toimenpiteet_nro, vika_nro ja toimenpiteet_kuvaus (kuva 15).



KUVA 15. Toimenpiteet-taulun rakenne

Vika-taulu

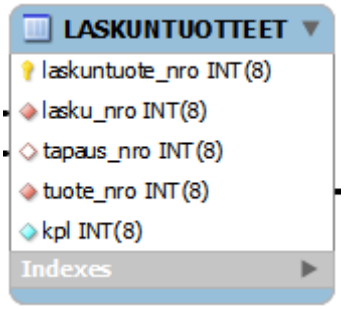
Vika-taulu sisältää laitteiden mahdollisten vikojen tiedot. Taulussa on 2 saraketta: vika_nro ja vika_kuvaus (kuva 16).



KUVA 16. Vika-taulun rakenne

Laskuntuotteet-taulu

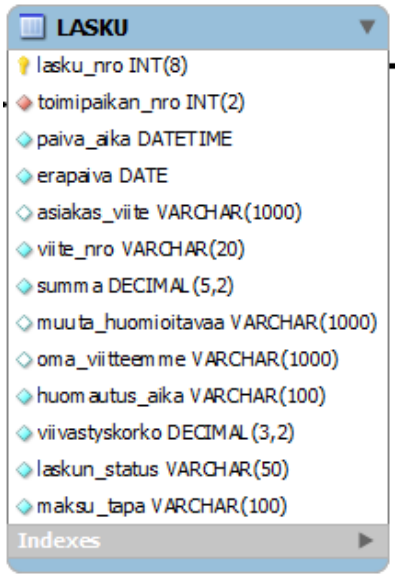
Laskuntuotteet-taulu sisältää laskun tuotteiden tiedot. Taulussa on 5 saraketta: laskuntuote_nro, lasku_nro, tapaus_nro, tuote_nro ja kpl (kuva 17).



KUVA 17. Laskuntuotteet-taulun rakenne

Lasku-taulu

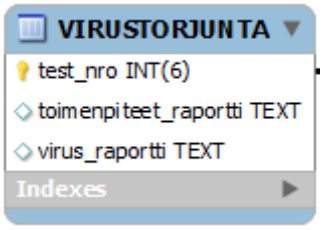
Lasku-taulu sisältää kokonaislaskujen tiedot. Taulussa on 13 saraketta: lasku_nro, toimipaikan_nro, paiva_aika, erapaiva, asiakas_viite, viite_nro, summa, muuta_huomioitavaa, oma_viitteemme, huomautus_aika, viivastyskorko, laskun_status ja maksu_tapa (kuva 18).



KUVA 18. Lasku-taulun rakenne

Virustorjunta-taulu

Virustorjunta-taulu sisältää virustorjuntaohjelman toimenpiteen tiedot. Taulussa on 3 saraketta: test_nro, toimenpiteet_raportti ja virus_raportti (kuva 19).



KUVA 19. Virustorjunta-taulun rakenne

4.3 Tietokantayhteyden luominen sovelluksessa

Tietokantayhteyttä varten projektissa luodaan luokka DataBase. Funktiossa createConnection() (kuva 20) kutsutaan metodia addDatabase() ohjaamaan ajuria, jota käytetään sovelluksessa.

```
//create database connection
bool DataBase::createConnection()
{
    QSqlDatabase db = QSqlDatabase::addDatabase( "QMYSQL" );
    db.setDatabaseName( "datapalvelut" );
    db.setUserName( "root" );
    db.setPassword( "test" );
    db.setHostName( "localhost" );
    db.setPort(3306);
    if (!db.open()) {
        QMessageBox::information( 0, "Cannot open database.",
                db.lastError().databaseText() + "\n");
        return false;
    }
    return true;

}
```

KUVA 20. Tietokantayhteys

Funktion createConnection() luomisen jälkeen lisätään DataBase-objekti main.cpp-tiedostoon. Objektia käytetään tietokantayhteyden oletuksena (kuva 21).

```
//create database connection
DataBase db;
db.createConnection();
```

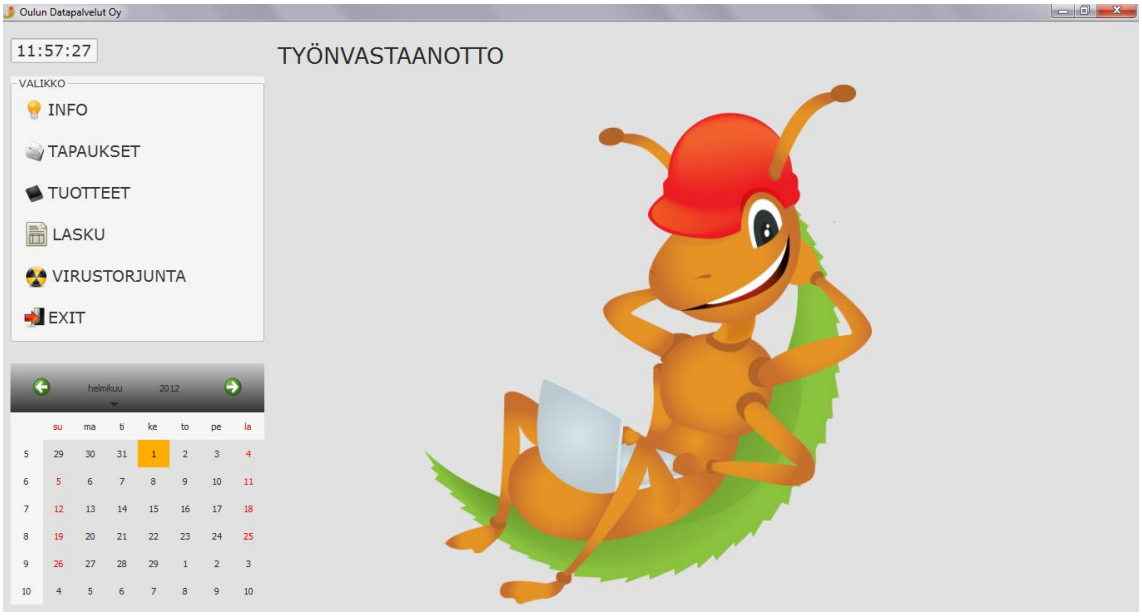
KUVA 21. Objektin lisääminen main-tiedostossa

5 KÄYTTÖLIITTYMÄN SUUNNITTELU

Ohjelman graafisen ulkoasun ja käyttöliittymän suunnitteluun on käytetty Qt Designeria. Qt Designer - käyttöliittymämuokkain sisältää paljon hyödyllisiä ominaisuuksia. Sen avulla dialogien tai muiden ikkunoiden suunnittelu on helppoa ja yksinkertaista. Komponentteja on valmiiksi kattava valikoima. Komponenttien sijoittelu on helppoa ja suoraviivaista. Myös on mahdollista laittaa automaattisia asetteluja, joiden avulla voi nopeasti asetella komponentit paikoilleen.

5.1 Pääikkuna

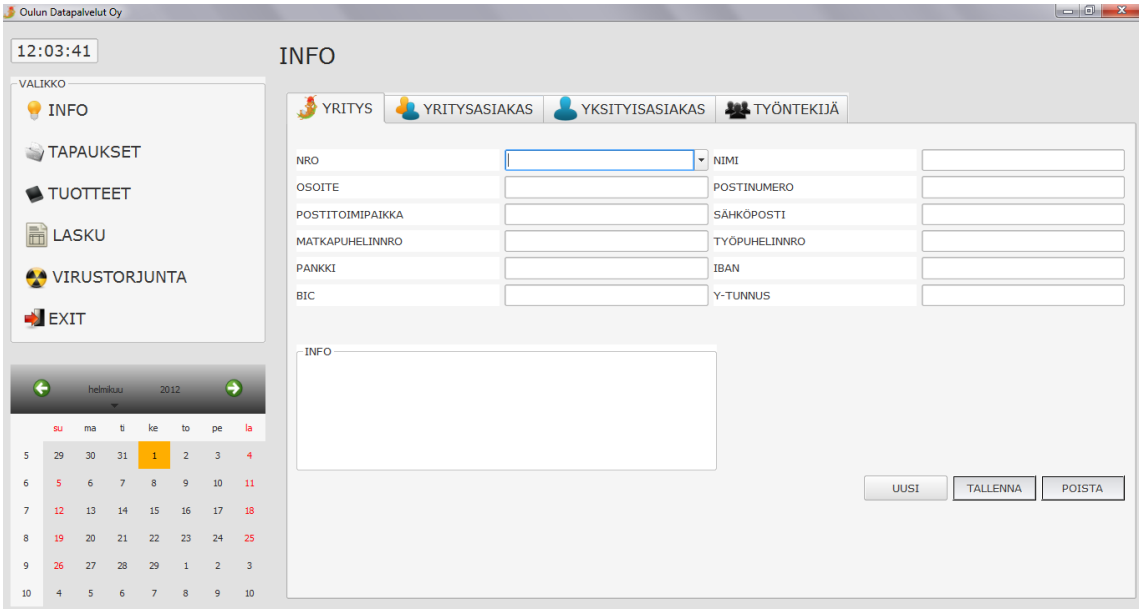
Aloitin suunnittelun sitä, kuinka monta ikkunaa ohjelmassa tarvitaan ja miten niitä sijoitetaan. Tulin päätökseen luoda pääikkuna (kuva 22), jonka sisältää muut alaikkunat: info-, tapaus-, tuote-, lasku- ja virustorjuntaikkuna. Alaikkunat aukeavat pääikkunan sisälle valittuja näppäimiä painettaessa, joissa selvästi lukee valinnan teksti. Pääikkunan lisäominaisuuksia ovat kello ja kalenteri, jotka näyttävät nykyisen ajan. Näitä ominaisuuksia voi kehittää ohjelman jatkokehityksessä. Esimerkiksi, kalenterin päivää näpäyttämällä voidaan tarkistaa työtapaukset päivämäärän mukaan.



KUVA 22. Näkymä pääikkunasta

5.2 Infoikkuna

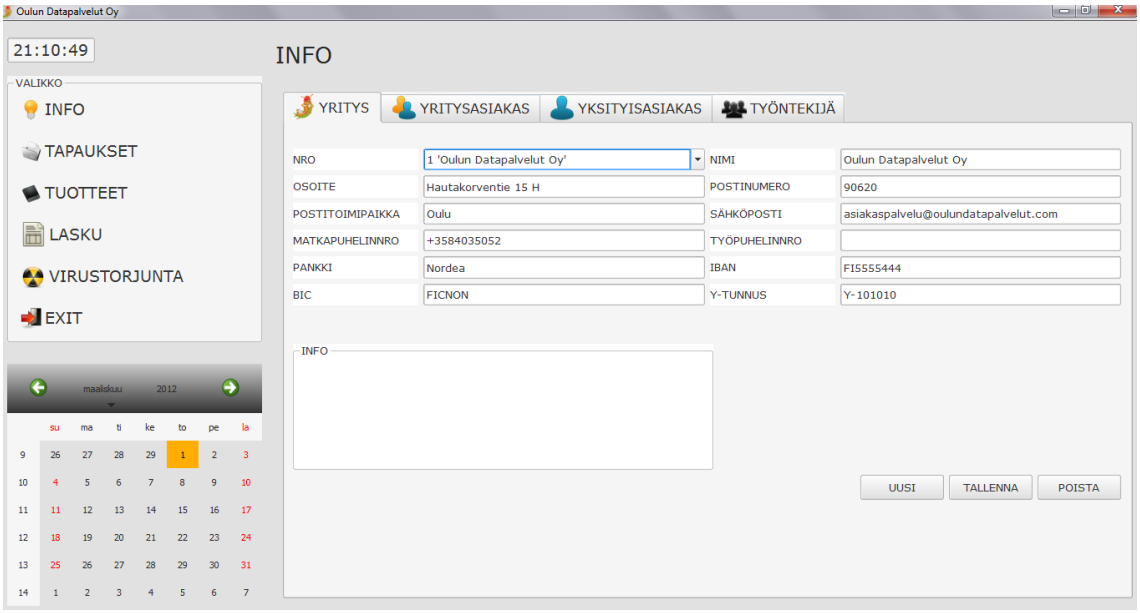
Infoikkunan rakenne (kuva 23) on yksinkertainen. Käyttäjä voi helposti selata alaikkunat välilehtiä painettaessa.



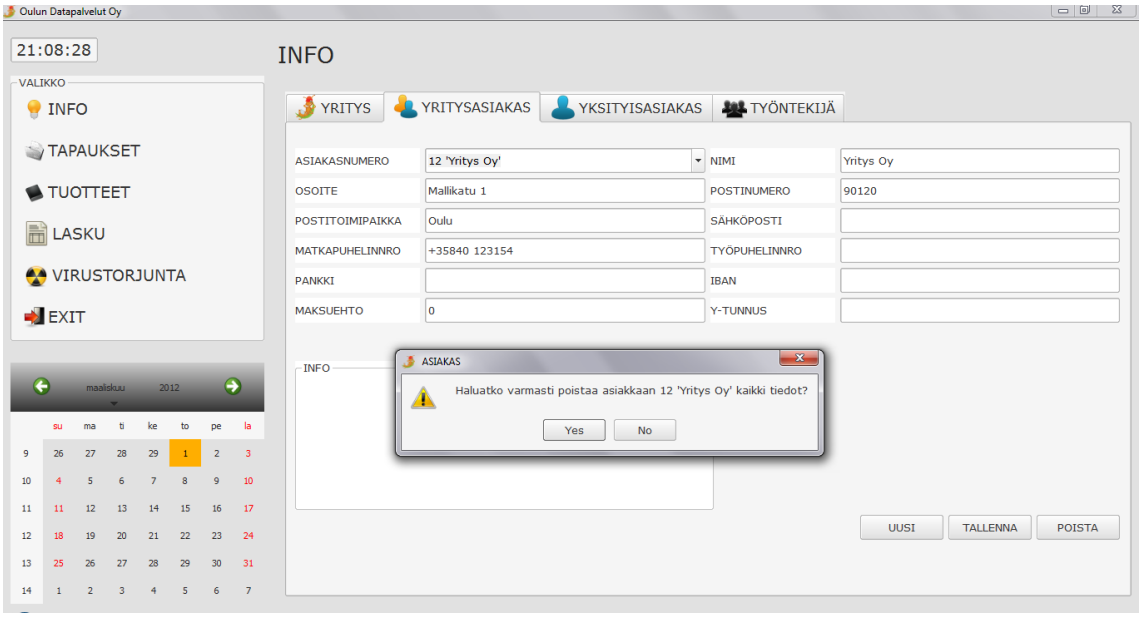
KUVA 23. Näkymä infoikkunasta

Infoikkuna sisältää 4 välilehteä: yritys, yritysasiakas, yksityisasiakas ja työntekijä. Jokaisen välilehden ikkuna näyttää tiedot yrityksestä,

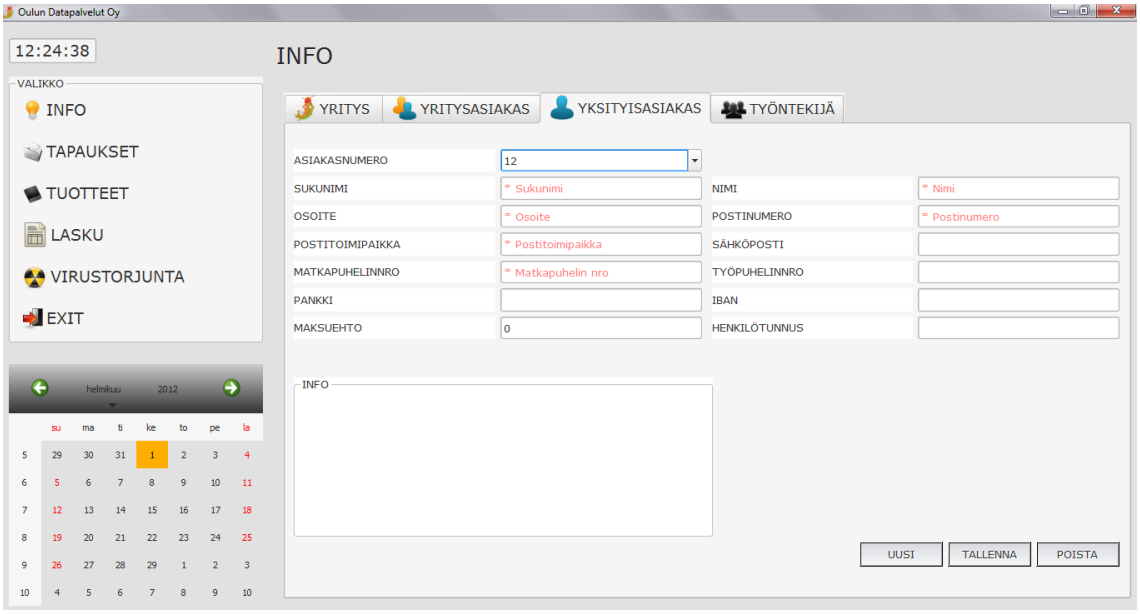
yritysasiakkaasta, yksityisasiakkaasta tai työntekijästä (kuva 24), jonka käyttäjä voi poistaa (kuva 25), luoda uudet (kuva 26) ja muokata (kuva 27). Tätä varten jokaisen välilehden ikkunassa on luotu painikkeet uusi, tallenna ja poista.



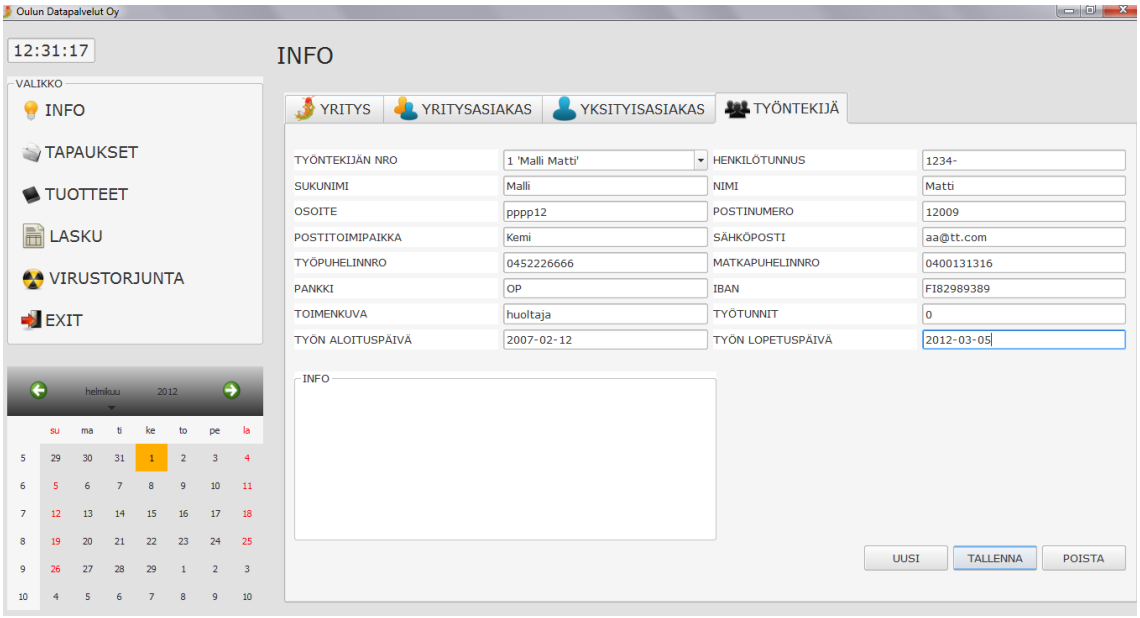
KUVA 24. Näkymä yrityksen tiedoista



KUVA 25. Näkymä yritysasiakkaan tietojen poistamisesta



KUVA 26. Näkymä uuden yksityisasiakkaan tietojen syöttämisestä



KUVA 27. Näkymä työntekijän tietojen muokkaamisesta

6 SOVELLUS

Sovelluksen teossa oli tarkoituksena suunnitella mahdollisimman modulaarinen sovelluskokonaisuus, jotta voidaan helposti laajentaa myöhemmin.

Sovelluksen kehittämisessä käytetyistä Qt:n luokista voidaan mainita käyttöliittymään liittyvät luokat QMainWindow, QWidget, QPushButton, QGroupBox, QLabel, QLineEdit, QComboBox, QStackedWidget ja QCalendarWidget.

Sovelluksen toimintojen toteutukseen käytettiin hyväksi Qt:n QDebug-, QSqlDatabase-, QSqlTableModel-, QSqlQuery-, QSqlQueryModel-, QSqlRecord-, QSqlError-, QString-, QMessageBox-, QTime-, QDate-, QTextCharFormat-, QMap-, QEvent-, QObject-, MouseEvent-, QVariant-, QPalette-luokkaa jne. Myös käytettiin itse luodut luokat MainWindow, InfoWindow, ClientWidget, CompanyClientWidget, CompanyWidget, EmployeeWidget, DataBase ja FormValidator.

Sovelluksen teossa oli asioita, joita joskus olen käyttänyt sekä jotain oli aivan uutta ja ratkaisut pitäisi etsiä joistakin lähteistä tai forumeista.

Ohjelman käytettävissä on useita editoreja. Niitä tarvitaan tietojen poimimiseen, tallentamiseen, muokkaamiseen ja poistamiseen tietokannasta. Loin FormValidator-luokan, jotta voisin välttää toistuvan editorin luettelemisen ja vähentää koodin kirjoittamista. Tässä tapauksessa käytetään QMap-tietorakennetta eli tallennetaan avain, jonka avulla itse tallennettava arvo haetaan. Avaimen tietotyyppi on QLineEdit-luokka ja tallennettavan arvon tietotyyppi on QString-luokka (kuva 28).

```
//provides line edits list
m_editFields    << ui->NimiLineEdit
                << ui->SukunimiLineEdit
                << ui->OsoiteLineEdit
                << ui->PNLineEdit
                << ui->PTLineEdit
                << ui->PNROLineEdit
                << ui->PNRO2LineEdit
                << ui->SPLineEdit
                << ui->PankkiLineEdit
                << ui->IBANLineEdit
                << ui->MaksuEhtoLineEdit
                << ui->HenkiloTunnusLineEdit;
```

KUVA 28. Koodiesimerkki tiedostosta clientwidget.cpp

Sen jälkeen loin funktion validateMandatoryFields()(kuva 29), jonka avulla vahvistetaan pakolliset kentät. Sitten loin objektin jokaisessa tiedostossa, jossa käytetään editoria.

```
bool ClientWidget::validateMandatoryFields()
{
   QLineEdit* e;
   foreach(e, m_editHints.keys())
   {
      if (e->text().length() == 0 ||
            e->text() == m_editHints.value(e))
         return false;
   }

   return true;
}
```

KUVA 29. Koodiesimerkki tiedostosta clientwidget.cpp

Ohjelman käynnistyessä kaikki editorit ovat ei-muokattavissa muodossa. Editorin tekstin voidaan muokata hiirellä kaksoispainettaessa ja myös kun lisätään uutta tietoa, esimerkiksi uusi asiakas. Aluksi en saanut muokkaus-funktioita oikein toimimaan. Aina kun uusi asiakas oli syötetty ja tiedot tallennettu, sen jälkeen editorit pysyivät aina muokattavissa muodossa. Ratkaisin sen tekemällä lippu-eli bool-funktion isNewCustomerMode() (kuva 30). Lisäsin lipun tapahtuman funktioon eventFilter(), jossa on määritetty hiiren tapahtuma double click (kuva 30).

```
//button is enabled
bool ClientWidget::isNewCustomerMode() const
{
   return !ui->NewPushButton->isEnabled();
}


bool ClientWidget::eventFilter(QObject *obj, QEvent *event)
{
   // container lookup is more expensive than event type check,
   // thus only check object if event is handable
   QLineEdit* le = qobject_cast<QLineEdit*>(obj);

   if ((event->type() == QEvent::MouseButtonDblClick ||
        event->type() == QEvent::FocusOut)
         && le
         && !isNewCustomerMode()
         && m_editFields.contains(le))
   {
      // do nothing, if control is disabled
      if (!le->isEnabled())
         return false;
```

KUVA 30. Koodiesimerkki tiedostosta clientwidget.cpp

Ohjelman comboboxit näyttävät numero ja nimet. Numero tarkoittaa järjestysnumeron tietokannassa. Ohjelman vaatimuksen mukaan aina kun käyttäjä haluaa lisätä uuden asiakkaan, ensin luodaan järjestysnumero tietokantaan, joka ilmestyy comboboxissa uusi-painiketta painettaessa. Sen jälkeen syötetään uudet tiedot tekstieditoriin. Loin funktion generateId() (kuva 31), joka luo uuden järjestysnumeron tietokannassa. Käytin QString-luokan muuttujia sekä QMap-tietorakennetta. Tiedot kerätään QStringList-luokan muuttujaan, jotka täydennetään ja sitten käytetään uuden tietojen luomisessa. Sitten kun kaikki tiedot pitäisi tallentaa tallenna-painiketta painettaessa, käytetään funktio changeData() (kuva 31). Funktiossa on kysely, joka käyttää sql-metodia UPDATE eli päivittää kaikki tiedot järjestysnumeron mukaan.

```
//generate new id
int ClientWidget::generateId(const QString &table,
                      const QMap<QString, QString>& defaultParams)
{
   QSqlQuery query;
   QString sQuery = QString("INSERT INTO %1").arg(table);

   QStringList paramList, valueList;
   QString param;
   foreach (param, defaultParams.keys())
   {
      paramList << param;
      valueList << QString(" '%1'").arg(defaultParams[param]);
   }

   if (paramList.count()) {
      sQuery.append("(%1) VALUES (%2)");
      sQuery = sQuery.arg(paramList.join(",")).arg(valueList.join(","));
   }


QSqlQuery query;
// Remeber that field order here is important as it depends on the order of line edits in
// m_editFields
query.prepare( "UPDATE asiakas "
            " SET nimi = ?, sukunimi =  ?, osoite =  ?, postinumero = ?, postitoimipaikka = ?,"
            " puhelin_nro_1 = ?, puhelin_nro_2 = ?, sahkoposti = ?, "
            " pankki_nimi = ?, IBAN = ?, maksu_ehto_pv = ?, henkilotunnus = ?"
            " WHERE asiakas_nro = ? AND asiakas_status = 'yksityis'");

foreach(QLineEdit* le, m_editFields)
{
   query.addBindValue( le->text().toUtf8());
}
```

KUVA 31. Koodiesimerkki tiedostosta clientwidget.cpp

7 LOPPUTULOKSET

Opinnäytetyön päämääränä oli suunnitella ja toteuttaa työnvastaanottojärjestelmä, jotta yritys voi jatkaa sen jatkokehitystä ja käyttää omana työnvastaanottojärjestelmänä.

Olemassa olevaa tietokantaa voidaan hyödyntää yrityksen toiminnassa. Tietokannan tiedot voidaan helposti lisätä ohjelmassa. Tietokannan päivittäminen ei vaadi lisätoimenpiteitä, koska uusi tieto pystyy lisätä ohjelmassa automaattisesti tietokantapalvelimelle. Tietokannan päivitys voidaan suorittaa käyttämällä MySQL Workbench 5.2 CE- tai phpMyAdmin-työkalun sql-editoria.

Ensimmäinen virhe oli se, että alussa yliarvioin omat kyvyt ja tavoitteet olivat hieman epämääräiset. Seuraukset ilmestyivät myöhemmin opinnäytetyön suorittaessa.

Opinnäytetyön aihe oli mielenkiintoinen. Tietokannan suunnittelu ja toteuttaminen oli minulle ennestään tuttua. Työn Qt-ohjelmoinnin osuus oli hieman haastavampi. En ollut aikaisemmin toteuttanut mitään ohjelmistoa alusta pitäen. Sen takia ohjelman suunnittelun ja toteuttamisen eteneminen oli hieman hidas ja siihen olisi pitänyt kiinnittää eniten huomiota. Niin kuin olisi pitänyt tehdä kerralla yksi osa valmiiksi ja sitten vasta siirtyä seuraavaan osaan, eikä monia asioita osittain. Nyt tiedän, että on erittäin tärkeä työn alussa laatia selkeät ja tarkat työn vaatimukset. Sitten pystyy tekemään työtä askel askelta määritetyn vaatimuksien mukaan.

Raportin kirjoittaminen on vaatinut paljon aikaa ja tuntui välillä hyvin haastavalta. Opinnäytetyön aihe oli kuitenkin oma-aloitteinen eikä vaatinut paljon ulkolähteitä, siksi joskus mietin jokaista sanaa ja mi-

ten saan raportin tekstin sisältöä selkeästi kirjoitettua työn tekemisestä. Kuitenkin olen tyytyväinen raportin sisältöön.

LÄHTEET

MySQL. 2012. Saatavissa: http://en.wikipedia.org/wiki/MySQL. Hakupäivä 22.2.2012.

MySQL Cookbook. Introduction. Saatavissa: http://www.freeopenbook.com/mysqlcookbook/mysqlckbk-chp-1-sect-1.html. Hakupäivä 22.2.2012.

MySQL Workbench 5.2. Saatavissa: http://www.mysql.com/products/workbench/. Hakupäivä 22.2.2012.

Oulun Datapalvelut Oy. Etusivu. Saatavissa: http://www.oulundatapalvelut.com/. Hakupäivä 1.11.2011.

QtCore Module. Saatavissa: http://doc.qt.nokia.com/4.7-snapshot/qtcore.html. Hakupäivä 22.11.2011.

QObject Class Reference. Saatavissa: http://doc.qt.nokia.com/latest/qobject.html#details. Hakupäivä 22.11.2011.

The Meta-Object System. Saatavissa: http://doc.qt.nokia.com/stable/metaobjects.html. Hakupäivä 22.11.2011.

TIETOKANNAN EER-KAAVIO. LIITE 1