Mittauslaitteiston tuottaman informaation jatkokäsittely

Opinnäytetyö

Harri Honkanen

Elektroniikan koulutusohjelma
Sulautetut järjestelmät

SAVONIA-AMMATTIKORKEAKOULU TEKNIIKKA KUOPIO

Koulutusohjelma

Elektroniikan koulutusohjelma

Tekijä

Harri Honkanen

Työn nimi

Mittauslaitteiston tuottaman informaation jatkokäsittely

Työn laji	Päiväys	Sivumäärä
Opinnäytetyö	20.12.2010	32+ 4

Työn valvoja	Yrityksen yhdyshenkilö
Lehtori Keijo Kuosmanen	Insinööri Jari Voutilainen

Yritys

Finero Oy

Tiivistelmä

Tämän opinnäytetyön aiheena oli tuottaa kaksi ohjelmistomoduulia (jatkossa ohjelmamoduuli A ja ohjelmamoduuli B) käytettäväksi joko yhdessä tai erikseen olemassa olevan tietokoneohjatun testausjärjestelmän yhteydessä. Ohjelmamoduuli A oli suunniteltu tehdasympäristöön tuotannon henkilöstön käyttöön ja ohjelmamoduuli B toimistoympäristöön tiedonanalysointia varten.

Työ tehtiin Finero Oy:lle, jolla on pitkä historia erilaisten automatisoitujen testausjärjestelmien toteutuksista. Tämän työn ohjelmamoduulien tarkoituksena oli laajentaa jo olemassa olevan laitteista ja ohjelmistosta koostuvan testausjärjestelmän toimintoja. Ohjelmistomoduulien tarkoitus oli toimia Windows ympäristössä. Ohjelmistomoduulit tulevat käyttämään lähiverkossa sijaitsevalla SQL Server 2005 -tietokantapalvelimella olevia tietoja.

Työssä vertailtiin mahdollisia ohjelmamoduulien toteutustapoja. Ohjelmointikielen valinta ja kehitysympäristön valinta oli ensimmäinen iso päätös. Ohjelmointikieleksi valittiin C# ja kehitysympäristöksi Microsoft Visual Studio 2010. Myös helppokäyttöisyyteen kiinnitettiin paljon huomiota. Erityisesti ohjelmamoduulin A mahdollinen käyttö kosketusnäytöllä ilman hiirtä otettiin suunnittelussa osittain huomioon. Työhön sisältyi myös uusien tietokantataulurakenteiden suunnittelu ohjelmamoduuli A:n tuottaman lisäinformaation tallennusta varten.

Tämän työn puitteissa tuotettiin ohjelmamoduuleista esittelyversiot. Moduulien viimeistely jätettiin myöhempään ajankohtaan.

Avainsanat

tietokannat, ohjelmointi, testausjärjestelmät

Luottamuksellisuus

julkinen

SAVONIA UNIVERSITY OF APPLIED SCIENCES

Degree Programme

Electronics

Author

Harri Honkanen

Title Of Project

Further Processing of Information Produced by Measurement System

Type of Project	Date	Pages
Final Project	20.December 2010	32+ 4

Academic Supervisor	Company connection
Mr Keijo Kuosmanen, Lecturer	Mr Jari Voutilainen, BEng

Orderer Company

Finero Oy

Abstract

The aim of this final project was to develop two software modules (from here on Software module A and Software module B) to be used together or separately in conjunction with the existing computer controlled measurement system. Software module A was planned to be used in factory environment by production personnel and Software module B was planned to be used in office environment for data analysation purposes.

The final project was done for Finero Oy which has a long history of implementing different kinds of measurement systems. The aim of the software modules in this project was to expand the functions of the existing system which consists of devices and software. Software modules were aimed for the Windows environment. The software modules will use data situated in a SQL Server 2005 database server through a local area network.

The final project included an evaluation of different methods for producing the software modules. The first big decision was the choosing of the programming language and development tools. C# was chosen as the programming language and Microsoft Visual Studio 2010 as the development tool. User-friendliness was also taken as a major design aspiration. Especially Software module A's possible use with a touch screen was taken into consideration in the design. The final project also entailed designing new database table structures for the storage of new information produced by Software module A.

Demo versions of both software modules were produced in the context of this thesis. The finalization of the software modules was left outside of this final project.

Keywords

Database, programming, measurement systems

Confidentiality

public

ALKUSANAT

Tämä opinnäytetyö on tehty vuoden 2010 aikana Kuopiossa ja Kausalassa. Haluan kiittää ohjaavaa opettajaa ,lehtori Keijo Kuosmasta ja Finero Oy:n edustajaa, insinööri Jari Voutilaista opastuksesta ja ohjauksesta työtä tehdessä. Haluan myös kiittää yliopettaja Väinö Maksimaista alkusysäyksen antamisesta työlle sekä koko Finero Oy:n henkilökuntaa työn mahdollistamisesta.

Kausalassa 20.12.2010

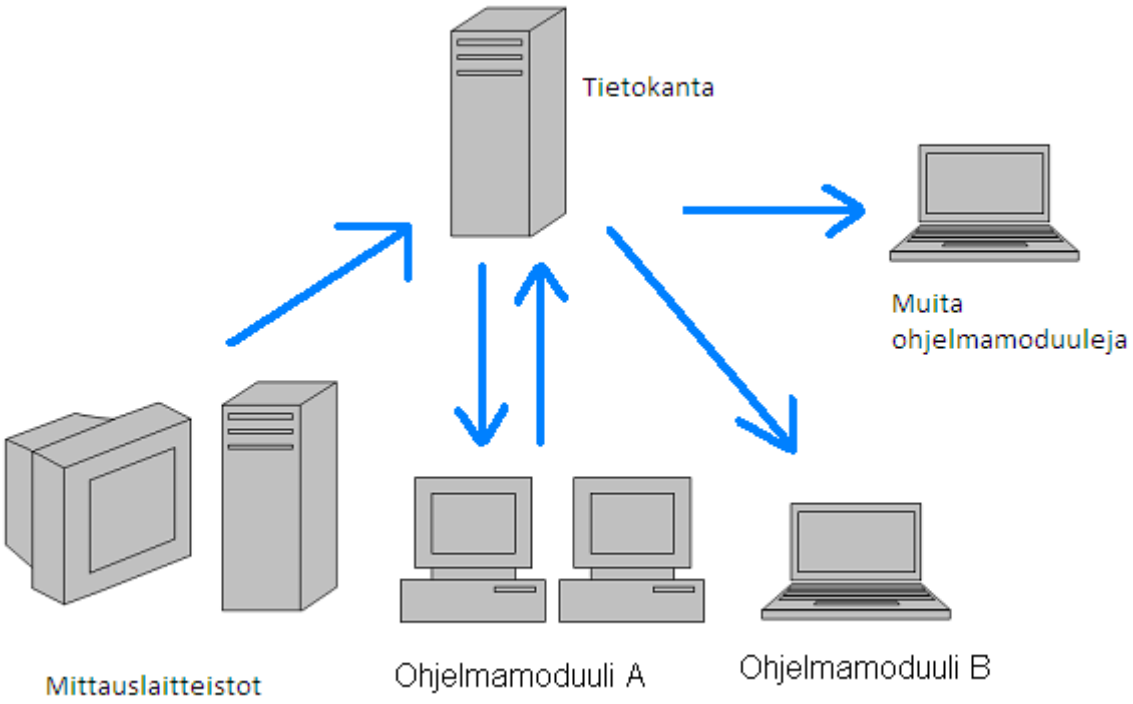
Harri Honkanen

Sisällysluettelo

1. JOHDANTO

Tämän opinnäytetyön aiheena on tuottaa kaksi ohjelmistomoduulia Finero Oy:n testausjärjestelmään. Finero Oy on vuonna 1980 perustettu teollisuuden automatisoituun sähköturvatestaamiseen keskittynyt yritys. Fineron toimitilat sijaitsevat Kausalassa. Finero tekee erilaisia perusmallisia testauslaitteita ja niiden kanssa käytettäväksi tarkoitettuja PC-ohjelmistoja. Yritys tekee ja jälleenmyy lämpökaappijärjestelmiä ja tekee lisäksi asiakkaille myös räätälöityjä testausjärjestelmiä. Fineron asiakkaina on suuria maailmanlaajuisia sähköalan yrityksiä, esimerkiksi ABB, Vacon, Ensto ja Enics. Maantieteellisestä sijainnistaan huolimatta Finero on erittäin kansainvälisesti toimiva yritys, ja jälleenmyyjiä ja asiakkaita on lähes joka mantereella.

Kuvassa 1 on esitelty tämän työn ohjelmamoduulien toiminta testausjärjestelmän kanssa. Mittauslaitteistoja ohjaavien tietokoneiden ohjelmamoduulit syöttävät mittaustuloksia tietokantaan, josta sitten muut tietoja käyttävät ohjelmamoduulit hakevat ne. Kaikki ohjelmiston ohjelmamoduulit sekä tietokantapalvelin on suunniteltu sijaitsemaan samassa lähiverkossa olevilla tietokoneilla. Suoraa keskinäistä kommunikointia ohjelmamoduulien välillä ei ole, vaan ne tallentavat, muokkaavat ja tulkitsevat tietokannassa sijaitsevaa tietoa. Tietokanta voi myös sijaita jollain mittauslaitteistojen tietokoneista riippuen systeemin koosta.



Kuva 1. Yleiskuva ohjelmiston kokonaisrakenteesta

Ohjelmamoduulin A tehtävä on hakea tietoa mittaustuloksista mittauslaitteiston välittömään läheisyyteen, jossa sitä käytetään heti testaustapahtuman jälkeen. Ohjelmamoduuli B puolestaan hakee tietoa pidemmällä aikavälillä tapahtuvaa testaustapahtumien ja mittaustulosten analysointia varten.

Yrityksessä on automaatiotaustan myötä hyvin vankka Labview-osaaminen ja Labview'lla toteutettuja ohjelmamoduuleja on myös tässä ohjelmistossa käytössä. Tästä syystä yrityksen toiveena oli selvittää, voisiko Labview'ta käyttää myös tämän työn toteutukseen. Yrityksen toivomuksena oli selvittää erilaisten toteutustapojen hyötyjä.

Työssä on myös tarkoitus perehtyä SQL Server 2005 -ympäristön käyttöön ja ympäristön työlle asettamiin rajoituksiin ja mahdollisuuksiin. Tämän tietokantapalvelimen käyttö oli jo päätetty aiemmin, ja päätöksen vaikutukset tuli myös ottaa huomioon kehitysympäristöä ja ohjelmointikieltä päätettäessä.

2. KEHITYSTYÖKALUJEN VALINTA

2.1. Alkutilanne

Ensimmäinen tehtävä oli ohjelmointikielen valinta. Kohdealustan selkeän Windows-ympäristöön rajaamisen perusteella luonnollisimpiai ohjelmointikieliä olivat C#, Visual Basic ja C++. Myös Java SE soveltuu hyvin tämän tyylisten sovellusten tuottamiseen. Yrityksen toiveiden mukaisesti Myös National Instrumentsin Labview-ohjelmiston käyttöä ohjelmamoduulien tuottamiseen tutkittiin. Lisäksi eri kielien kehitysympäristöjen hinnat otettiin huomioon valintaa tehtäessä.

Alustavasti vaihtoehdoiksi valittiin C#, Java ja Labview. Visual Basic ja C++ rajattiin pois liian suurten opiskeluvaatimusten takia. Mitään lisäarvoa niiden käyttämisestä verrattuna C#:n käyttöön ei tässä yhteydessä olisi saatu. Myös ohjelmiston olemassa olevissa osissa oli jo käytetty C#-kieltä. Java oli mukana lähinnä vertailutarkoituksessa.

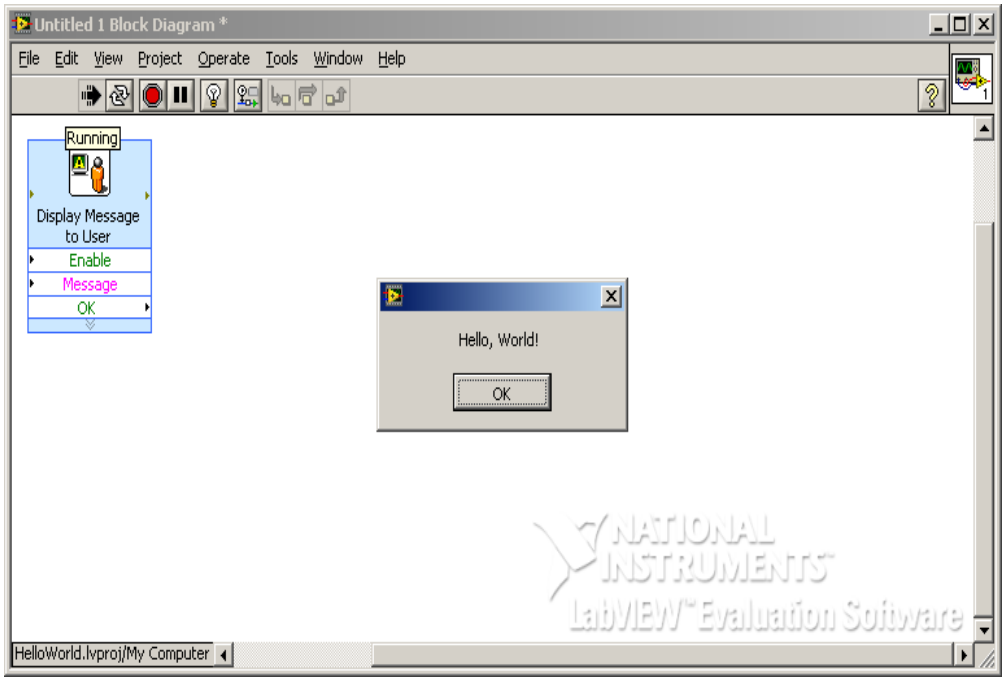
2.2. Vaihtoehdot

2.2.1. Labview

National Instrumentsin Labview-ohjelmistosta on nykyään tullut lähes standardi teollisuusautomaatiosovelluksissa. Lähes kaikki testaussovellukset joko käyttävät sitä tai ovat sen kanssa yhteensopivia. Normaali käytäntö on toimittaa laitteiden komentokirjastot tai valmiit Labview-ajurit suoraan asiakkaille. Niitä myös ladataan National Instrumentsin verkkosivustolle vapaaseen levitykseen.

Labview perustuu National Instrumentsin kehittämään graafiseen G-kieleen. Ohjelmia ei kirjoiteta, vaan ne muodostetaan graafisesti Flowchart-tyyppisellä ohjelmoinnilla, joka keskittyy datan kulun seuraamiseen ja ohjaamiseen. Tämä on hyvin erilainen lähestymistapa ohjelmointiin. [2]

Parhaiten Labview soveltuu juuri teollisuuden mittaus- ja säätöautomaation ohjaussovelluksiin, mutta nykyään se alkaa olla niin kehittynyt, että sitä voi periaatteessa käyttää erilaisten sovellusten yleisohjelmointikielenä. [3]



Kuva 2. HelloWorld-sovellus Labview-ohjelmistolla toteutettuna.[4]

Labview eroaa melko paljon perinteisistä ohjelmointikielistä. ”Koodi” on periaatteessa vain yksittäisiä valmiita komponentteja, joita yhdistellään vuokaaviomaisilla viivoilla. Kuvassa 2 esitetty HelloWorld-sovellusesimerkki onnistuu pelkästään sijoittamalla dialog-ikkunan tuottava komponentti ohjelmaan ja sijoittamalla sen tekstiksi ”Hello, World!”. Komponentti tuottaa standardin Windows-dialogin, johon voi määritellä erilaisia paluuarvoja, esimerkiksi kuvan ”Ok”.

Labviewta harkittiin tässä yhteydessä, koska se on jo käytössä yrityksessä ja ohjelmiston muita osia on toteutettu sillä, joten ohjelmiston jatkokehitys ja yhteensopivuus olisi helposti toteutettavissa. Ohjelman graafinen ilme muodostui yhdeksi kompastuskiveksi, koska Labview’lla erilaisten graafisten ilmeiden luominen on paljon työläämpää kuin Javalla tai C#:lla. Sen kirjastojen käyttöliittymäkomponentit emuloivat hyvin paljon fyysisiä nappeja ja kytkimiä ja näyttävät melko vanhanaikaisilta esimerkiksi tiedon analysointiin tarkoitetussa sovelluksessa.

Labview-kehitystyökalut ovat maksullisia, ja yksittäisen lisenssin hinnat alkavat tuhansista euroista. National Instruments tarjoaa monia erilaisia hankintavaihtoehtoja ja -versioita ohjelmistosta kunkin käyttäjän tarpeiden mukaan.

2.2.2. Java

Java on yksi suosituimmista ohjelmointikielistä ja siitä on olemassa versioita suurelle valikoimalle erilaisia alustoja. Javaa ylläpitää Sun Microsystems, joka on nykyään Oraclen omistuksessa. Java JDK on lisensoitu GPL-lisenssin alle, eli se on ilmainen kehitysresurssi.

Java on oliopohjainen ohjelmointikieli. Javalla kirjoitettuja ohjelmia ei käännetä konekieleksi käännösvaiheessa vaan niistä tehdään JRE:n (Java Runtime Engine) ymmärtämää välikieltä, joka sitten ajo-vaiheessa käännetään alustakohtaisen JRE:n mukaan. Tämä mahdollistaa ohjelmien ajon muutoksitta useilla eri alustoilla, mutta samalla tekee ajosta melko raskasta verrattuna natiiveihin eli suoraan konekieleksi käännettäviin kieliin, esimerkiksi C/C++:aan, verrattuna.

Java SE:stä löytyvä Swing-komponentti kirjasto mahdollistaa normaalien Windows-tyylisten graafisten sovellusten tekemisen pienellä vaivalla. Kirjastossa on saatavilla valtaosa perinteisistä Windows-sovelluksissa käytettävistä komponenteista. Esimerkiksi tekstinsyöttökentille ja painikkeille on olemassa valmiit muokattavat luokat. Näiden komponenttien graafinen ilme on melko hyvä ja mikäli ne eivät riitä, lisää komponentteja on saatavilla Internetistä.[5]

Javalle on saatavilla useita ilmaisia kehitystyökaluja. Esimerkiksi NetBeans on opensource-mallilla toteutettu kehitysympäristö. NetBeans’sta löytyy myös graafinen suunnittelutyökalu Swing-kirjastoa käyttävien desktop-sovellusten ulkoasun luomiseen.

```
/**
 * The HelloWorldApp class implements an application that
 * simply prints "Hello World!" to standard output.
 */
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!"); // Display the string.
    }
}
```

Kuva 3. Hello World-sovellus Java-kielellä (Java SE)[6]

Kuvassa 3 on Hello World -sovellus Java-kielellä toteutettuna. Ohjelmassa käytetään System-kirjaston println-funktiota tuottamaan teksti konsoliin. Funktion nimi ”println” on lyhennys sanoista ”print line”, joka on suomeksi ”kirjoita rivi”. Javan syntaksi onkin helposti ymmärrettävää ja lähes poikkeuksetta funktioiden ja kirjastojen nimet kertovat suoraan, mitä ne tekevät.

2.2.3. C#

C# on olio-orientoitunut kieli, ja se on kehitetty monikäyttöiseksi peruskieleksi Windows- ympäristössä. C#:n on kehittänyt Microsoft ja se on osa Microsoftin .NET-konseptia. C# toimii CLI:n (Common Language Infrastructure) päällä .NET-ympäristössä ja mukailee sen toimintoja. CLI:llä tarkoitetaan Microsoftin infrastruktuuria, joka mahdollistaa .NET-kielien kohdealustasta riippumattoman ajon.

C# muistuttaa hyvin paljon Java -kieltä ja sen voi näkökannan mukaan katsoa olevan suurelta osin kopio Java-kielestä. C#:ia ei käännetä suoraan konekieleksi vaan se käännetään ”välikieleksi” ja sen ajoon käytetään CLR-moottoria(Common Language Runtime).

C# muistuttaa syntaksiltaan hyvin paljon Javaa ,mutta samalla myös C/C++:aa, joten sen lähestyminen C/C++ taustalla on hyvinkin helppoa. Myös MSDN-sivuston opetusmateriaali tarjoaa hyvät lähtökohdat kielen opiskeluun.

```
using System;
class Hello
{
   static void Main() {
      Console.WriteLine("hello, world");
   }
}
```

Kuva 4. HelloWorld-sovellus C#-kielellä[7]

Kuvassa 4 on esitelty HelloWorld sovellus C#-kielellä toteutettuna. Syntaksi on hyvin lähellä Javaa ja C# onkin yksi Javan suurimmista kilpailijoista Windows-ympäristöön tarkoitettujen sovellusten kehityksessä.

C#:n luontevin kehitysympäristö on Microsoftin Visual Studio-tuoteperheen eri versiot. Valtaosa näistä on maksullisia ja hinnat alkavat tätä kirjoittaessa noin tuhannesta eurosta. Kaikille ympäristön kielille on kuitenkin saatavilla myös ilmaisversiot. C#:n tapauksessa ilmaisversion nimi on Microsoft Visual C# Express. Ilmaisversio kattaa valtaosan C# ominaisuuksista, mutta esimerkiksi Office:n lisäosien ja räätälöitävien asennusohjelmistojen teko mahdollisuus siitä puuttuu joten kaupallisten sovellusten kehittäminen sillä on melko rajoitettua.

C#:a harkitessa tuli myös ottaa huomioon eräiden Visual Studion hankintamallien mukana saatavan MSDN-tilauksen tuomat mahdollisuudet. MSDN-tilauksella saa vuosimaksua vastaan käyttöönsä valtavan määrän Microsoftin maksullisia sovelluksia ja käyttöjärjestelmiä jotka pitäisi muuten hankkia yksittäin jolloin hinta nousisi melko suureksi. MSDN-tilauksen mukana saa myös käyttöönsä kaikki aiemmat versiot Visual Studiosta. Tämä on hyödyllistä, koska uusimmat versiot eivät välttämättä tue kaikkia vanhimpia alustoja. Esimerkiksi Visual Studio 2010 versio tukee Office:n lisäosien tekoa vain Office 2007 ja Office 2010 versioihin.

2.3. Vertailun lopputulos

Kielten vertailuun ei lopulta käytetty paljon aikaa. Tärkein tavoite oli tutkia, olisiko ohjelmamoduulien toteutus ollut mahdollista Labviewlla. Tässä tapauksessa yritykseltä olisi löytynyt kehitystyökalut valmiiksi hankittuina. Java otettiin mukaan puhtaasti vertailusyistä. Ohjelmiston muita osia oli jo toteutettu C#- kielellä ja Microsoft Sql Server 2005 oli valittu tietokantapalvelimeksi. Ohjelmamoduulit oli tarkoitettu toimimaan vain Microsoft Windows ja Microsoft Office-ympäristöissä, joten C# oli hyvin luonnollinen valinta. Kehitystyö aloitettiin Visual C# Expressillä, mutta ohjelmien määrittelyjen tarkentuessa päädyttiin lopulta hankkimaan Visual Studio 2010 Professional. Myös Officen lisäosien tekemisen tarve vaikutti Professional-versioon päätymisessä.[8]

3. MICROSOFT VISUAL STUDIO 2010

3.1. Yleiset ominaisuudet

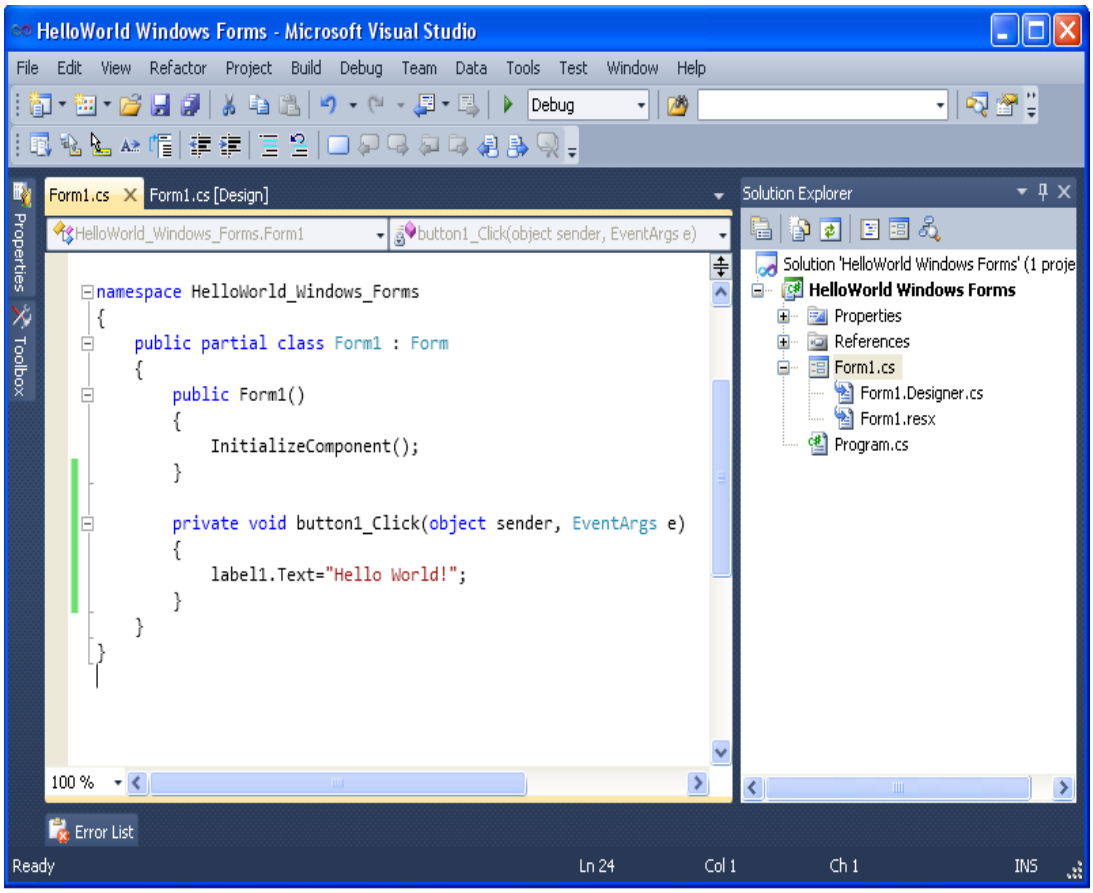
Microsoft Visual Studio on Microsoftin kehittämä kehitysympäristö, joka on tarkoitettu erityisesti Microsoftin .NET-alustan sovellusten kehitykseen. Visual Studion kielivalikoimaan kuuluvat mm. C#, C/C++, Visual Basic ja F#. Visual Studiolla voi tehdä myös asp-pohjaisia web-sivustoja. Visual Studio on hyvin kypsä tuoteperhe, ensimmäinen versio on vuodelta 97 (Visual Studio 97). Uusin versio on Microsoft Visual Studio 2010.[8]

Ohjelmasta on olemassa useita eri versioita ja lisensointimahdollisuuksia. Visual Studio on saatavilla myös MSDN (Microsoft Developer Network) –tilauksen kanssa, jolloin kaikki aiemmat versiot ja kattava kirjasto Microsoftin muita ohjelmia on ladattavissa kehitystarkoituksiin MSDN-sivustolta. Myös näistä MSDN-tilauksista on saatavilla useita erilaisia versioita eri valikoimilla ladattavia ohjelmia.

Visual Studion heikkoudeksi voisi mainita sen keskittyminen pelkästään Microsoftin alustoille, Linux- tai Mac-tukea ei ole. Tässä työssä Linux- ja Mac-tuen puute ei kuitenkaan tuota ongelmaa, koska ohjelmisto on tarkoitettu vain Windows-käyttöjärjestelmille. [8]

3.2. Windows Forms

Tässä työssä käytettyyn Windows Forms -tyyppisen sovelluspohjan sovellukseen Visual Studiossa on erillinen designer-työkalu, jolla voidaan luoda sovelluksen grafiikka hyvin pienellä vaivalla visuaalisesti. Tämä työkalu luo joko Visual Basic tai C#-pohjaista koodia designerissa luodun graafisen mallin perusteella. Samanlaiset designerit ovat olemassa myös Windows Presentation Foundation ja Web-tyypin sovelluksille. [8]



Kuva 5. Microsoft Visual Studio 2010 Professional Edition-kehitysympäristö.

Kuvassa 5 on Windows Forms -sovelluspohjalla tehty yksinkertainen sovellus. Sovellus tuottaa nappia painamalla Hello World -tekstin ikkunaan kuvan 6 mukaisesti. Tehtäessä tämäntyyppistä projektia luo Visual Studio automaattisesti ohjelman perusrungon.

Aluksi ohjelma luo luokan, jossa on yksi ikkuna ilman sisältöä. Ohjelma jakaa koodin kahteen tiedostoon. Toisessa tiedostossa on luokan komponenttien määrittelyt. Tähän osioon Visual Studio luo ohjelmakoodin graafisessa suunnittelutyökalussa tehtyjen muutosten mukaan. Tätäkin osuutta käyttäjä voi muokata, mutta se vaatii jo syvempää ammattitaitoa.

Kuvassa 5 näkyvä osuus on tarkoitettu käyttäjän muokattavaksi. Visual Studio luo automaattisesti kuvassa 5 olevan osuuden. Tässä osiossa myös määritellään esimerkiksi mitä tehdään, kun käyttäjä painaa hiirellä jotain painikkeista, jotka on luotu graafisessa suunnittelutyökalussa. Kuvassa 5 määritellään, että painettaessa nappia, jonka nimi on ”button1”, sijoitetaan otsikon ”label1” ominaisuudelle ”text”

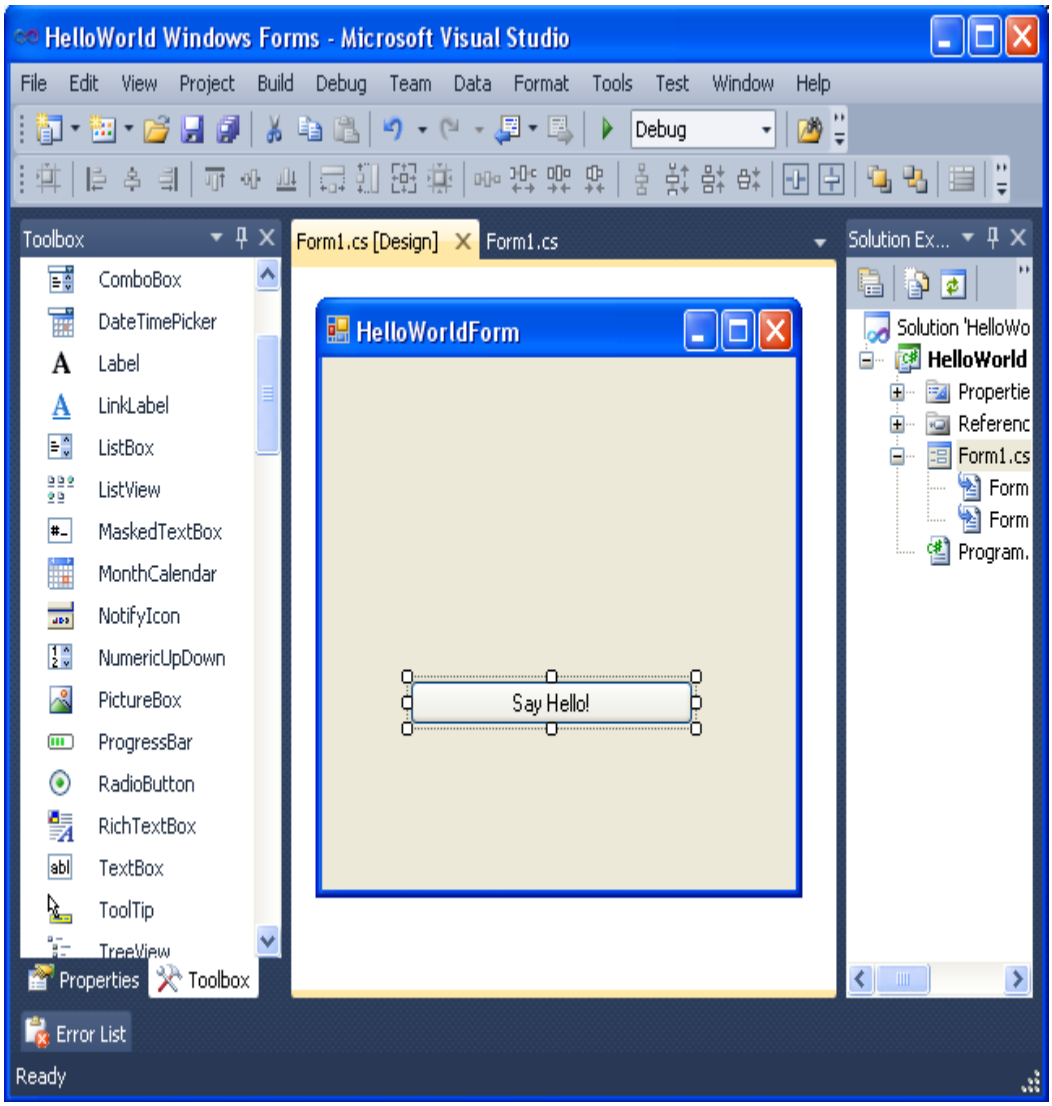
arvo ”Hello World!”. Tällöin ”label1” muuttaa käyttäjälle näkyvän tekstin ”Hello World!”:ksi.

Kuva 6. HelloWorld-sovellus

Sovelluksen grafiikkaa ei kirjoiteta käsin vaan se suunnitellaan graafisella työkalulla, joka sisältyy Visual Studioon. Visual Studio luo suunnittelun jälkeen graafisen työkalun pohjalta erillisen tiedoston, jossa tekstipohjainen komponenttien luonti tapahtuu. Tämä nopeuttaa huomattavasti graafisten käyttöliittymien luontia verrattuna jokaisen olion ja sen ominaisuuksien muokkaamiseen käsin.

Graafisen työkalun luoma määrittelytiedosto on osa luokkaa, joka määrittelee kyseisen näkymän. Luokka on jaettu partial-avainsanan avulla kahteen eri tiedostoon. Määrittelyt tapahtuvat InitializeComponent-metodin alla, ja kuvasta 5 näkyykin, että kyseistä metodia kutsutaan luokan konstruktorissa.

Peruskäyttäjän on tarkoitus muokata ohjelmakoodista vain osaa, johon tulevat esimerkiksi nappien clicked-eventit. Näitä eventtejä määritellään normaalisti suoraan graafisessa työkalussa. Esimerkiksi tuplaklikattaessa button-tyyppistä komponenttia luo Visual Studio automaattisesti metodin sen clicked-eventille.



Kuva 7. Visual Studion graafinen designer-työkalu.

Kuvassa 7 näkyvästä Toolbox-valikosta voidaan vetää lomake pohjalle erilaisia komponentteja. Properties-välilehdestä voidaan määritellä tarkemmin valittuna olevan komponentin toimintaa.

Visual Studio tarjoaa myös työkalut tietokantojen käyttöön. Yhteyden luomiseen ja tiedonkäsittelyyn on olemassa luokkakirjastot. Erityisesti Microsoft Sql Server -tietokantapalvelimen käyttöön Visual Studio on erittäin kätevä työkalu. [8]

Windows Forms on osa .NET frameworkin valmiita käyttökirjastoja. Kirjastot tarjoavat valmiit rakennuspalikat Windows:n graafisten sovellusten luontiin. Monia aiemmin esim. C/C++:lla hankalasti toteutettavia toimintoja, on tehty valmiiksi kirjastoluokiksi, joiden käyttö on hyvin yksinkertaista. Esimerkiksi tekstitiedoston

lukeminen ja kirjoittaminen C:llä ja C#:lla on hyvin erilainen toimenpide ohjelmoijan näkökannalta. System.IO-kirjaston metodeja käyttämällä tämä operaatio supistuu C#:lla toteutettuna paljon lyhyemmäksi kuin vanhempia C-kirjastoja käyttämällä. System-luokan kirjastojen käyttö on mahdollista myös C/C++ puolella.

```
int _tmain(int argc, _TCHAR* argv[])
{

    FILE *stream;
    char list[30];
    int  i, numread, numwritten;


    if( (stream = fopen( "fread.out", "w+t" )) != NULL )
    {
      for ( i = 0; i < 25; i++ )
         list[i] = (char)('z' - i);

      numwritten = fwrite( list, sizeof( char ), 25, stream );
      printf( "Wrote %d items\n", numwritten );
      fclose( stream );

    }
    else
      printf( "Problem opening the file\n" );

    if( (stream = fopen( "fread.out", "r+t" )) != NULL )
    {

      numread = fread( list, sizeof( char ), 25, stream );
      printf( "Number of items read = %d\n", numread );
      printf( "Contents of buffer = %.25s\n", list );
      fclose( stream );
    }
    else
      printf( "File could not be opened\n" );

}
```

Kuva 8. Tiedoston kirjoitus ja luku C-kielellä.[9]

Kuvassa 8. kirjoitetaan ja luetaan tekstitiedostosta merkkejä ja tulostetaan ne ruudulle käyttäen C-kieltä ja stdio-kirjastoa. Syntaksi on hankala ja funktioiden toiminta on melko vaikea hahmottaa. Yksinkertaisenkin tiedostojen lukua ja

kirjoitusta sisältävän sovelluksen ohjelmakoodista tulee helposti satojen rivien pituinen.

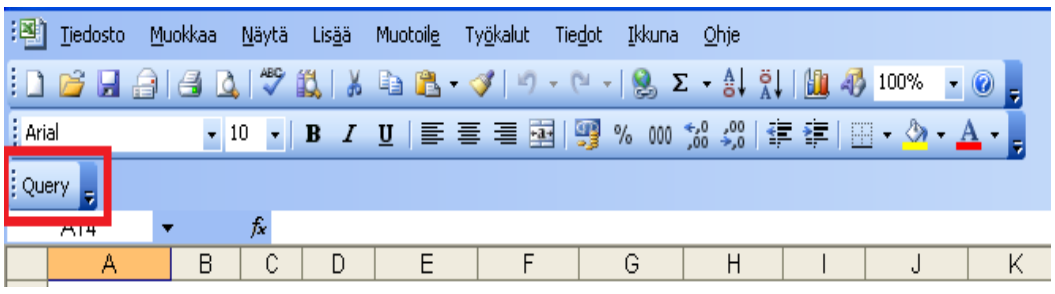
```
namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            System.IO.File.WriteAllText("example.txt", "Example text");
            label1.Text=System.IO.File.ReadAllText("example.txt");
        }
    }
}
```

Kuva 9. Tiedoston kirjoitus ja lukeminen C#:n WindowsForms-sovelluksessa.

Kuvasta 9. näkyy, että C#:ssa monet luku- ja kirjoitusfunktiot on automatisoitu valmiisiin luokkiin. Syntaksi on myös yksinkertaista kirjoittaa. Tiedon käyttäjälle näyttäminen onnistuu suoraan sijoittamalla tekstikentän label1 Text-ominaisuuteen kyseinen teksti.

3.3. Visual Studio Tools for Office

Visual Studion mukana tulevalla VSTO:lla (Visual Studio Tools for Office) voidaan luoda lisäosia Office-ohjelmiston eri moduuleihin. Lisäosat voivat olla esimerkiksi uusia työkalurivejä, jotka esim. Excelissä hakevat tietoa tietokannasta ja täyttävät sen suoraan Excelin taulukoihin.



Kuva 9. Punaisella on rajattu lisäosa-tyyppisen ohjelman Excel 2003:en lisäämä painike.

Office lisäosa -tyyppisten ohjelmien yksi käyttötarkoitus on tuoda tietoa esim. tietokannasta Officen käyttöön ja muokata siitä suoraan Excel-taulukoita. Ohjelma on periaatteessa erillinen sovellus, joka sijaitsee Officen toimintojen sisällä. Excelin tapauksessa paras tapa on käyttää Excel’n makroja C# kielen sisältä ja sijoittaa niillä haluttua dataa suoraan taulukoihin.
Excel’n lisäosa on periaatteessa ohjelma, joka avataan suoraan Excelistä erillisen ajon sijasta.

4. TIETOKANTA

4.1. Microsoft SQL Server 2005 Express

Microsoft SQL Serverin ensimmäinen versio julkaistiin vuonna 1988 nimellä ”SQL Server”. Tämä ensimmäinen versio kehitettiin yhteistyössä Sybase:n kanssa ja sen kohdealustana oli OS/2. Ensimmäinen Windows-ympäristöön tarkoitettu SQL Serverin versio julkaistiin vuonna 1993 nimellä ”SQL Server 4.2 A desktop database”. Tämä versio oli vielä toiminnaltaan melko rajoitteinen ja soveltui lähinnä pienyritysten käyttötarkoituksiin. Se saavutti kuitenkin kohdealustansa ja helppokäyttöisyytensä takia melko suuren suosion. Varsinaisesti kaupallisella puolella SQL Server löi kunnolla läpi vuoden 2000 versiollaan ”SQL Server 2000 An enterprise database”. Vuoden 2005 versio ”SQL Server 2005” toi uutena .Net Frameworkin käytön, mikä mahdollista .Net objektien suoran integraation SQL Serveriin, jolla saavutettiin lisää joustavuutta tietokannan käytössä. Oraclella oli vastaava ominaisuus Java-kielellä toteutettuna.[12]

Tämän työn tietokantasovelluksena käytettiin Microsoft SQL Server 2005 Expressiä. Kyseessä on SQL Server 2005:n ilmaisversio. SQL Server 2005 Express käyttää samaa perusalustaa kuin kaupallinen versio, joten toimintavarmuudeltaan ja tehokkuudeltaan se on erittäin hyvä ottaen huomioon, että se on ilmainen. Suurimpina eroina kaupalliseen versioon on tietokantojen koko, joka on rajoitettu 4 gigatavuun.

Myös Express-versio on rajoitettu käyttämään vain yhtä prosessorin ydintä, jolloin sen hakunopeus suuremmissa ja useiden tahojen yhtäaikaista käyttöä vaativissa sovelluksissa laskee. Nämä rajoitteet eivät kuitenkaan tule vastaan kuin vasta melko suuressa määrässä tietoa ja tämän opinnäytetyön yhteydessä päästiin vasta muutaman megatavun tietomäärään, vaikka rivejä tietokannoissa alkoi olla useita satoja. Mikäli tila joskus loppuu kesken, on päivittäminen kaupalliseen versioon hyvin helppoa ja onnistunee joko hyvin pienillä ohjelmakoodin muutoksilla tai jopa kokonaan ilman niitä. Asiakkaalla voi myös olla jo ohjelmia hankittaessa käytössä SQL Serverin kaupallinen versio, jolloin suoraan siihen liittyminen on hyvin helppoa.[11]

4.2. Transact-SQL

T-SQL eli Transact Structured Query Language on kieli, jolla kaikki SQL Server 2005:n yhteydessä olevat sovellukset riippumatta niiden toteutustavasta antavat sille komentoja. Tietokantaa on mahdollista käyttää jopa suoraan komentokehotteesta tämän kielen avulla.[10]

SQL eli Structured Query Language:n kehitti alun perin IBM 1970-luvulla. IBM halusi tällä kielellä luoda standardoidun tavan relaatiotietokantojen käyttämiseen. Huolimatta iästään kieli on yhä kaikkien relaatiotietokanta sovellusten perusta.[11]

Transact-SQL pohjautuu hyvin pitkälle SQL-kieleen, mutta se tuo tähän useita lisäominaisuuksia. Esimerkiksi virheenhallintaa ei ole normaalissa SQL-kielessä. T-SQL tuo monista kielistä tutun try-catch-tyylisen virheenkaappaus ominaisuuden. Muita T-SQL:n ominaisuuksia ovat muun muassa paikalliset muuttujat T-SQL-kielisen ohjelman sisällä sekä ehdollisen logiikan laajennukset. Tässä työssä ei kuitenkaan käytetty T-SQL-kielen erityisominaisuuksia vaan käyttö rajoittui jo normaalista SQL-kielestä löytyviin perustoimintoihin. T-SQL-kielen on alun perin kehittänyt Sybase. [11]

```
-- Returns only two of the records in the table
SELECT ProductID, ProductName, Price, ProductDescription
    FROM dbo.Products
    WHERE ProductID < 60
GO
```

Kuva 10. Ehdollinen tiedonhaku tietokannasta T-SQL-kielellä.[13]

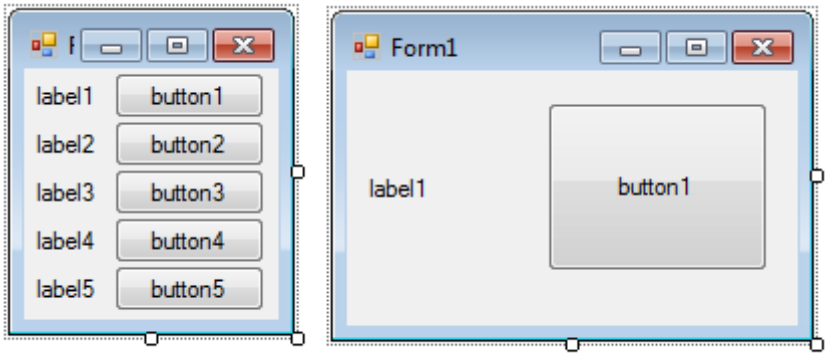
Kuvassa 10 on esitetty ehkä yleisin käytetty toiminto SQL ja T-SQL kielissä. ”SELECT”-komennolla valitaan tietoja ”FROM”-komennon osoittamasta tietokantatauluista. ”WHERE”-komennolla voidaan puolestaan määritellä jokin ehto, jolla rajataan halutut tiedot vain tietyn ehtolausekkeen täyttäviin. Tässä tapauksessa ProductID-kentän arvon pitää olla alle 60.

5. OHJELMAMODUULI A

Ohjelmamoduuli A suunniteltiin tehdasympäristössä tapahtuvaa käyttöä varten. Moduulin tarkoituksena oli tuoda tietoa testaustapahtumista ja -tuloksista tuotantohenkilöstön käyttöön. Moduulin tehtävänä oli myös tuottaa lisätietoja käsitellyistä laitteista erityisesti ohjelmamoduuli B:n käyttöön. Moduulin liitettiin myös kirjautumis-toiminto, eli sovellus kysyy aina sitä käynnistettäessä käyttäjänimen ja salasanan. Suurin syy kirjautumistietojen kyselyyn on käyttäjien valvonta. Tiedot siitä kuka ohjelmaa käyttää ja kuinka kauan, haluttiin saada tallennettua tietokantaan. Tämä on tärkeää tuntikohtaisten työvoimakustannusten analysoinnissa.

Tehtaassa käytettävän sovelluksen grafiikan suunnittelussa on otettava huomioon ympäristön tuomat rajoitukset. Esimerkiksi monissa tehdasympäristöissä käytössä voi olla vain kosketusnäyttö eikä ollenkaan hiirtä. Aina ei välttämättä myöskään ole käytettävissä näppäimistöä vaan ääritapauksessa voi olla, että käytössä on vain kosketusnäyttö ja viivakoodin lukulaite.

Tämän ohjelmistomoduulin käyttö vaatii näppäimistön mutta on mahdollista, että hiirtä ei sitä käytettäessä ole liitetty tietokoneeseen. Tämä tuli ottaa huomioon miettiessä kenttien ja nappien kokoa käyttöliittymässä. Pienten nappien käyttö kosketusnäytöllä on erittäin hankalaa ja voi pahimmillaan välillisesti johtaa näytön rikkoontumiseen, kun käyttäjät koettavat sormen sijasta painaa pientä painiketta kynällä tai ruuvimeisselillä, joita varten ei kosketusnäyttöjä ole tarkoitettu.



Kuva 11. Nappien kokoerot muuttavat sovelluksen luonnetta.

Kuvassa 11 oleva vasemman puoleinen sovellus on lähes mahdoton käyttää kosketus näytöllä. Riippuen kosketusnäytöstä, yhdellä sormenpainalluksella voi helposti painaa useita nappeja tai osua kokonaan väärään näppäimeen. Tällainen tilanne ajaa käyttäjän käyttämään jotain käsillä olevaa terävää esinettä nappien painamiseen. Tällöin vaste on tarkempi ja napeista helpompi osua juuri haluttuun. Tämä aiheuttaa kuitenkin rasitusta näytölle ja pahimmassa tapauksessa kallis kosketusnäyttö rikkoontuu tai kuluu huomattavasti nopeammin. Kuvan 10. oikean puoleisessa sovelluksessa näppäimen kokoa on kasvatettu huomattavasti ja sen ympärille on jätetty riittävästi tilaa. Tällöin huolettomampikaan painallus ei aiheuta virhetoimintoja.

Oletettuna käyttäjäryhmänä ovat elektroniikka- ja sähköasentajat. On myös mahdollista, että moduulia tulevat käyttämään täysin ilman teknistä taustaa olevat, suoraan tehtävään koulutetut työntekijät. Tästä johtuen siitä tuli peruskäytössään tehdä mahdollisimman yksinkertainen käyttää mutta samalla lisätä hieman kehittyneempiä ominaisuuksia enemmän asiasta perillä olevia käyttäjiä varten.

Tämän ohjelmamoduulin yhteydessä harkittiin myös salasanan käyttöä rajaamaan, mitä kukin käyttäjä voi ohjelmalla tehdä, mutta siitä luovuttiin. Käyttäjätunnuksien ja salasanojen hallinta päätettiin jättää kokonaan SQL Server 2005 varaan. SQL Server 2005:ssa voidaan määritellä salasanat ja käyttäjätunnukset joilla tietokannassa olevia tietoja voidaan käyttää. Ohjelmamoduulin käyttö on täysin mahdotonta ilman tätä yhteyttä, joten tässä tapauksella käyttäjätunnuksilla pystytään hallitsemaan kuka ohjelmaa käyttää

6. OHJELMAMODUULI B

Ohjelmamoduuli B:n tarkoituksena on tuoda käyttäjän määrittämien parametrien mukaan tietoa mittaustapahtumista ja -tuloksista käyttäjän helposti analysoitavaksi. Moduulin tarkoitus oli hakea tietoa tietokannasta ja muokata siitä käyttäjälle selkokielistä tietoa, jota tämä voisi analysoida. Mietittiin toteutustapaa, joka olisi tiedon analysointia ja esimerkiksi kustannusten laskua tekevien käyttötottumuksien mukainen. Yksi ohjelmiston osista oli toteutettu Excel'n lisäosana, joten päätettiin toteuttaa myös tämä sovellus samalla templaatilla.

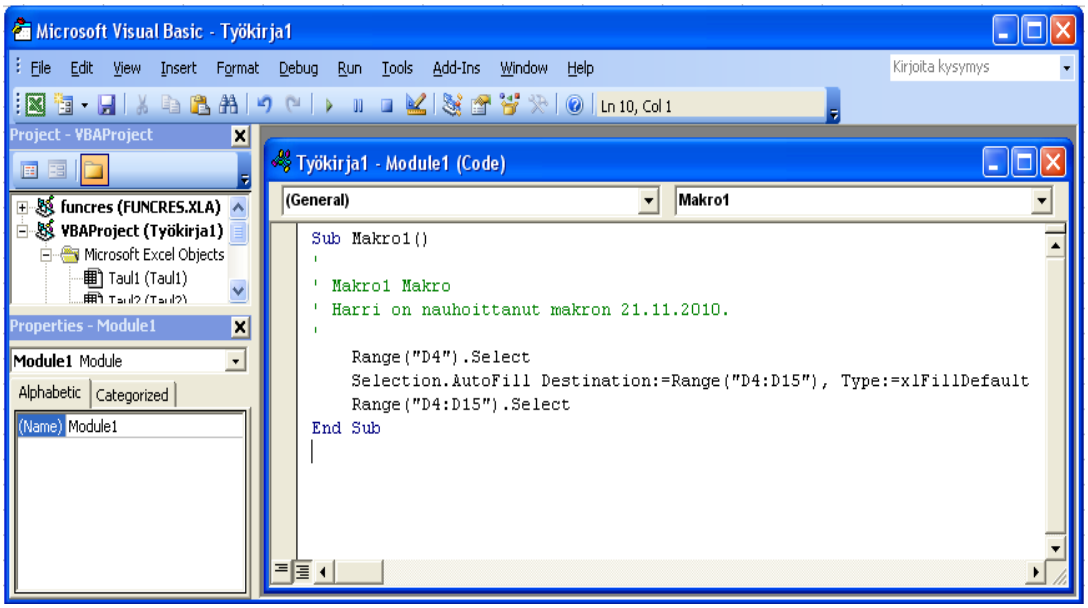
Ohjelmamoduuli B:n tapaiset analysointi ohjelmat voivat olla hyvin vaikeakäyttöisiä valtavan ominaisuuskirjon takia. Tutkittaessa markkinoilla jo olevia vastaavaan tarkoitukseen käytettäviä analysointiohjelmia ja käyttäjiltä saatuja palautteita havaittiin, että kysyntää olisi hyvin yksinkertaiselle ohjelmalle. Markkinoilla olevilla ohjelmilla on taipumusta valtavan ominaisuusmäärän ja räätälöintimahdollisuuksien takia olla hyvin vaikeakäyttöisiä. Pahimmassa tapauksessa nämä ohjelmat ovat täysin asennuskelvottomia normaalin käyttäjän taidoilla ja vaaditaankin asiantuntija asentamaan ja määrittelemään asetukset, että ohjelma voidaan edes ottaa käyttöön.

Ohjelmamoduuli B:stä pyrittiin tekemään helposti asennettava ja käytettävä, minkä vuoksi ominaisuuksia karsittiin ja asetukset rajoittuvat tietokantapalvelimen yhteyden asetuksiin. Hintana oli, että moduuli ei toimi kuin ainoastaan tämän kyseisen ohjelmiston yhteydessä eli sen myynti erillisenä tuotteena on mahdotonta. Tämän rajoittuneisuuden kuitenkin katsottiin olevan tarkoituksen mukaista.

Ohjelma tuottaa käyttäjän antamilla parametreilla Excel-taulukoita tietokannassa olevista mittaustuloksista ja ohjelmamoduuli A:n tuottamista lisätiedoista. Taulukot on pyritty tekemään hyvin selkokielisiksi mutta kuitenkin kaiken tarvittavan informaation näyttäviksi. Parametreina ohjelmalle annetaan esimerkiksi aikaväli ja yksittäinen tuotekoodi. Tämä on huomattavasti rajoittuneempaa kuin useissa muissa analysointiohjelmissa.

Excelin etuna tällaisessa sovelluksessa on sen yleiskäyttöisyys. Excel tarjoaa toiminnot tiedon muokkaamiseen ja analysointiin. Esimerkiksi erilaisten graafisten esitysten luominen onnistuu suoraan Excelissä ja kokenut käyttäjä voi muokata niistä täysin haluamiaan. Jos ohjelmamoduuli tuottaisi suoraan graafisia esityksiä tiedoista, jouduttaisiin tekemään ohjelmamoduuliin funktionaalisuus, jolla nämä graafiset esitykset voidaan tallentaa kuvana tai jopa suoraan Excel-taulukkona. Poistamalla tämän tarpeen ja tuottamalla tiedon esityksen suoraan Exceliin yksinkertaistuu hyppy tiedonhausta tiedon jatkokäyttöön huomattavasti. Käyttäjän ei tarvitse opetella uutta, monimutkaista analysointiohjelmaa vaan pelkät Excelin käyttötaidot riittävät tiedon analysointiin.

Ohjelmallisesti tiedon tuominen Exceliin tapahtuu käyttämällä hyväksi Excelin makroja. Excelin makroilla tarkoitetaan Excelissä suoraan nauhoitus ominaisuudella luotuja toimintojonoja. Makroja nauhoitettaessa, käynnistetään nauhoitus ja tehdään sarja toimintoja normaalisti hiirellä. Makroa nauhoitetaan, kunnes käyttäjä sammuttaa nauhoituksen, jolloin Excel luo ohjelmakoodin kyseisille toiminnoille ”Visual Basic for Applications” -kielellä (jatkossa VBA). Makroa voi tämän jälkeen muokata Excelistä löytyvällä VBA-editorilla. Makroja voi myös kirjoittaa suoraan ilman nauhoittamista. Tässä työssä tarvittavat makrot kirjoitettiin suoraan C#-kielisen ohjelmakoodin sekaan. Tämä oli tarpeellista, koska makrojen ohjelmakoodiin piti suoraan sijoittaa tietokannasta haetut tiedot.[14]



Kuva 12. Excel 2003 VBA -editori.

Kuvassa 12 on Excel 2003:n versio VBA-editorista. Kuvan 12 makro on nauhoitettu suoraan Excelistä löytyvällä työkalulla. Makro kopioi solussa D4 olevan sisällön soluihin D4 - D15. Kyseessä on eräs Excelin perustoiminnoista, joissa hiirellä painetaan valittuna olevan solun alakulmaa ja vedetään hiirtä alaspäin, jolloin ohjelma automaattisesti täyttää kaikki ruudut joiden yli valinta viedään. Ohjelmamoduuli B:n toteutuksen yhteydessä tämä työkalu osoittautui erittäin hyödylliseksi etsittäessä virheitä käytetyistä Excel-makroista. Suoraan käytettyjä makroja ei tällä työkalulla kuitenkaan voinut ajaa niihin käytetyjen ohjelman sisäisien muuttujien vuoksi.

Tähän ohjelmamoduuliin ei tässä opinnäytetyössä tehty minkäänlaista käyttäjähallintaa. Tietokannan yhteyden määrityksessä on kuitenkin käyttäjätunnus ja salasana, joita SQL Server 2005 hallinnoi.

7. TYÖN TOTEUTUS

Työ toteutettiin kevään ja kesän 2010 aikana. Työ alkoi tutustumalla moduulien vaatimuksiin. Moduuleista oli olemassa vaatimusspeksit joiden perusteella lähdettiin hahmottelemaan graafista käyttöliittymää. Alkuperäinen oletus oli C#-kielen käyttö, joten graafisen käyttöliittymän suunnittelua tehtiin Visual Studion designer-työkalulla. Tässä vaiheessa tutkailtiin vielä eri vaihtoehtoja potentiaaliseksi ohjelmointi kieleksi. Erityisesti Labview:n käyttöä ohjelmamoduuleissa harkittiin yrityksestä ennestään löytyvän kattavan Labview osaamisen takia. Mikäli ohjelmamoduulit olisi toteutettu Labview:lla, olisi niille löytynyt paremmin jatkokehittäjiä yrityksestä. Labview kuitenkin hylättiin sen kankean graafisen ilmeen takia. Myös Javaa harkittiin kustannussyistä mutta tämä olisi tuonut ohjelmistoon kolmannen käytettävän kielen tuomatta mitään lisäarvoa joten sitä ei vakavasti harkittu missään vaiheessa valinta-prosessia. Työ lähdettiinkin toteuttamaan C#-kielellä.

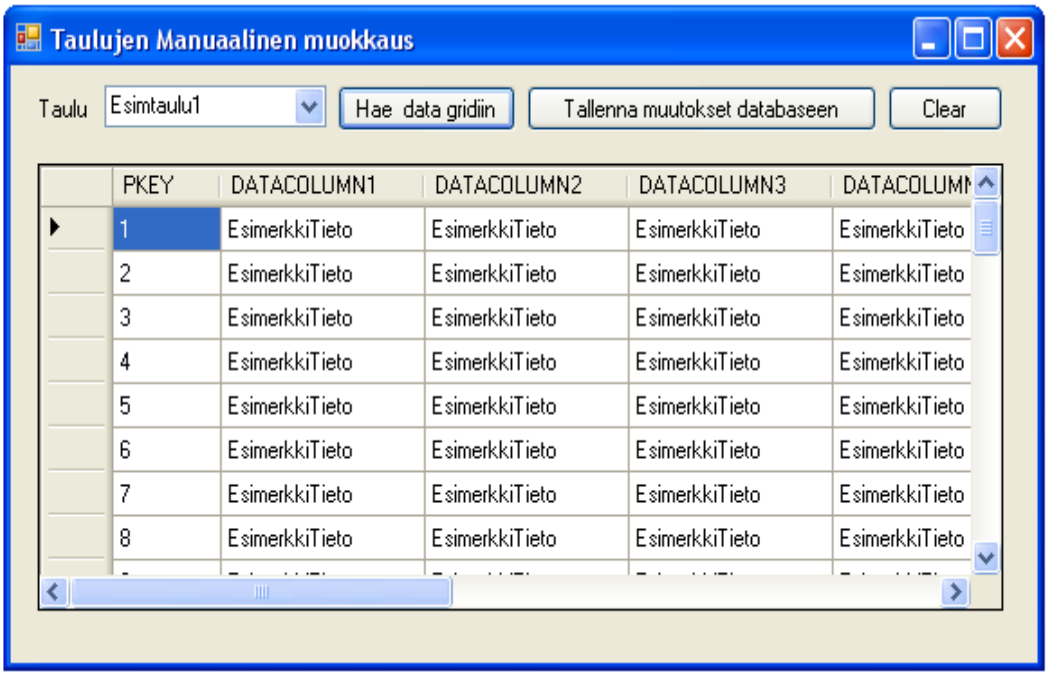
Alkuun käytössä oli Visual C# 2008 Express ohjelmisto, lähinnä sen ilmaisuuden takia. Sillä pystyi hyvin tekemään suunnitelmia käyttöliittymästä ja kirjoittamaan Myös ohjelmakoodia.

Aluksi työtä tehtiin etätyönä Kuopiossa sähköpostin ja puhelimen välityksellä. Etätyö vaiheen aikana saatiin aikaiseksi graafisten suunnitelmien peruslinjat ja hahmotelma tietokantatauluista joita tultaisiin luomaan näitä moduuleja varten. Tietokantojen hahmotteluun käytettiin MySql Workbench-ohjelmaa, joka on ilmainen. Taulujen varsinainen luonti tapahtuu ohjelmallisesti moduulien sisältä. Mikäli ohjelmamoduuli A ei löydä tauluja määritellystä tietokannasta, luo se ne sinne.

Työ alkoi perehtymisellä ohjelmiston määrittelydokumenttiin. Ohjelmasta oli olemassa hahmotelma, jossa määriteltiin yksittäiset ohjelmamoduulit ja niiden haluttu toiminta oli melko pitkälti hahmoteltu. Seuraavaksi suunniteltiin graafisen ulkoasun pääpiirteet. Tässä käytettiin suoraan Visual C# Express:n ja Visual Studio 2008:n graafista designer-työkalua. Toimintaa näihin graafisiin käyttöliittymäsuunnitelmiin ei missään vaiheessa tehty ja lopulliseen versioon nekin menivät uusiksi.

Seuraavaksi hahmoteltiin tietokantataulut, joita ohjelmamoduulit tulisivat tarvitsemaan olemassa olevien lisäksi. Ensimmäiset hahmotelmat tehtiin etätyövaiheen aikana, mutta lopulliseen muotoonsa ne muotoutuivat vasta työn alkaessa yrityksessä ja ohjelmamoduulien määrittelyjen päivittyessä.

Tässä vaiheessa alkoi työ yrityksessä, ja ensimmäinen välitavoite oli saada tietokantapalvelin toimimaan yrityksen lähiverkossa ja palvelimeen yhteys ohjelmista. Tässä vaiheessa ohjelmien tietokantayhteyksiä kehitettäessä nousi tarve työkalulle, jolla voisi tutkia tietokantatauluihin sijoitettuja tietoja samalla kun ajaa ohjelmaa. Tällaisia työkaluja ovat esimerkiksi SQL Server Management Studio Express tai Microsoft Access. Näiden kattavista ominaisuuksista huolimatta päädyttiin kuitenkin itse tekemään tällainen työkaluohjelma. Samalla harjaantui myös tekijän osaaminen C#:n tietokantayhteyksien ohjelmoinnissa.[11]



Kuva 13. Taulujen muokkaus sovellus.

Sovellus on tehty C#-kielellä. Graafinen ulkoasu on toteutettu valmiilla komponenteilla Visual Studion designer-työkalulla. Ohjelman toiminta on hyvin yksinkertainen: Se hakee koodissa määritellystä tietokannasta halutun tietokantataulun kaikki tiedot valmiiseen DataGridView-komponenttiin, joka muistuttaa paljon Excel-taulukkoa.

Kuvassa 13 on ohjelmaan syötetty tiedot eräästä tietokannan taulusta. DataGridViewissä olevia tietoja voi myös muokata ja tallentaa takaisin tietokantaan. Tästä työkalusta oli korvaamaton apu ohjelmia sunniteltaessa. Tietokannassa tapahtuneet muutokset olivat tarkistettavissa muutamalla napin painalluksella.

Ohjelmakoodi on nähtävissä liitteessä 1. Valtaosa ohjelmakoodista on suoraan designer-työkalun generoimaa. Erityisesti ”Taulujen Muokkaus.Designer.cs” tiedoston sisältö on kokonaan designer-työkalussa tehtyjen muokkausten pohjalta generoitua. Tietokantataulujen nimet ja yhteysparametrit on määritelty suoraan ohjelmakoodissa, joten tämän työkalun käyttö rajoittuu vain ja ainoastaan tähän käyttötapaukseen, ellei ohjelmaan tehdä radikaaleja muutoksia.

Työn alettua yrityksessä tuotettiin hyvin nopeasti ensimmäinen esittelyversio ohjelmamoduuli A:sta. Työn edetessä selkiytyi myös suunnitelma tehdä ohjelmamoduuli B Excel:n lisäosana. Tästä saatiin tuotettua ensimmäinen demoversio, jossa kuitenkaan ei kaikkia haluttuja toimintoja vielä ollut tehty toimiviksi. Varsinaista testaamista tai käyttäjäpalautteen perusteella tapahtunutta lopullista hiomista ohjelmista ei tässä opinnäytetyössä tehty. Työ päättyi tähän vaiheeseen ja lopullinen ohjelmamoduulien viimeistely jäi odottamaan myöhempää ajankohtaa.

8. YHTEENVETO

Tässä opinnäytetyössä luotiin ohjelmamoduulien perustoiminnallisuus. Moduuleiden lopullinen hiominen ja virheiden poisto jäi kuitenkin tämän opinnäytetyön ulkopuolelle tilaajan toiveiden mukaisesti.

Ohjelmamoduuli A:sta tuli tuotetuista ohjelmamoduuleista valmiimpi ja sen varsinaista toimintaa pystyi jo kaikilta osin kokeilemaan, mutta varsinaiseen testausvaiheeseen ei päästy. Ohjelmamoduulin toiminta vaatii vielä hiomista paremman käytettävyyden saavuttamiseksi.

Ohjelmamoduuli B saatiin tuottamaan tietoa Excel-taulukon muotoon ainoastaan yhdellä valittavissa olevista hakuparametreista, joskaan muiden käyttöönotto ei vaadi suuria ohjelmakoodin lisäyksiä. Ohjelmamoduuli B:n asentamisessa havaittiin ongelmia, joiden korjaaminen jää opinnäytetyön ulkopuolelle.

Henkilökohtaisena tavoitteena olleet ohjelmointikielien opettelu ja yleisen ohjelmointiosaamisen syventyminen onnistuivat hyvin. Työn aikana opin valtavasti uutta tietoa ohjelmoinnista. Opinnäytetyön tekeminen päättyi ohjelmien tilan raportointiin, minkä jälkeen siirryin yrityksen sisällä muihin tehtäviin.

LÄHTEET

1. MSDN Training and Certification Resources [WWW-sivusto]. [viitattu 30.10.2010].

2. Labview User Manual, 1998 National Instruments Corporation.

3. Product Information: What is NI LabVIEW? [WWW-sivusto]. [viitattu 14.10.2010]. www.ni.com/labview/whatis

4. Labview: Hello World picture [WWW-sivusto]. [viitattu 14.10.2010]. http://progopedia.com/static/upload_img/2010/08/26/hello-world.png

5. Java SE at a Glance [WWW-sivusto]. [viitattu 14.10.2010]. www.oracle.com/technetwork/java/javase/overview/index.html?ssSourceSiteId=ocomen

6. The Java Tutorials:Lesson A Closer Look at the "Hello World!" Application[WWW-sivusto][viitattu 30.10.2010] download.oracle.com/javase/tutorial/getStarted/application/index.html

7. ECMA-334 C# Language Specification, 4th edition (June 2006)[verkkodokumentti, PDF] . [viitattu 30.10.2010] saatavissa: www.ecma-international.org/publications/standards/Standard.htm

8. MSDN: Visual Studio [WWW-sivusto]. [viitattu 1.11.2010]. saatavissa: msdn.microsoft.com/en-us/library/52f3sw5c.aspx

9. MSDN: fread [WWW-sivusto]. [viitattu 1.11.2010] saatavissa: http://msdn.microsoft.com/en-us/library/kt0etdcs%28v=VS.71%29.aspx

10. MSDN: Transact-SQL Reference(Transact-SQL)[WWW-sivusto]. [viitattu 3.11.2010]. saatavissa: http://msdn.microsoft.com/en-us/library/ms189826.aspx

11. Microsoft SQL Server 2005 Express Edition FOR DUMMIES. Robert Schneider 2006[Kirja].

12. Beginning SQL Server 2005 Express for Developers: From novice to Professional. Robin Dewson 2007.

13. MSDN: Reading the Data in a Table (Tutorial)[WWW-sivusto]. [viitattu 7.11.2010] saatavissa: http://msdn.microsoft.com/en-us/library/ms365310.aspx

14. An Excel/Visual Basic for Applications (VBA) Programming Primer, R. J. Ribando 18.6.2010[verkkodokumentti, PDF]. [viitattu 21.11.2010] saatavissa: http://faculty.virginia.edu/ribando/modules/xls/VBAPrimer.pdf

Taulujen Muokkaus Sovelluksen ohjelmakoodi Liite 1

```
//TAULUJEN MUOKKAUS.CS-------------------------------------------------------------
--
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.Sql;
using System.Data.SqlClient;


namespace Taulujen_Muokkaus
{
    public partial class Form1 : Form
    {
        //Luodaan SqlDataAdapter Datasetin ja Sql Serverin välistä yhteyttä varten
        SqlDataAdapter dataAdapter;

        //Luodaan dataset, käytetään sijoittamaan tiedot DataGridViewiin
        DataSet dataSet;

        //etukäteen tiedossa oleva tietokanta ja taulun nimi.
        #region string strConnection=Connection String käytössä olleelle
tietokannalle
             ***
        #endregion

        public Form1()
        {
            InitializeComponent();
        }

        /// <summary>
        /// button1_click luo yhteyden tietokantaan käyttäjän määrittelemän
        /// connection stringin perusteella ja täyttää comboBox1:llä valitun
tietokantataulun tiedot
        /// dataGridView1:en.
        /// </summary>
        private void button1_Click(object sender, EventArgs e)
        {

            if (comboBox1.SelectedItem ==null )
            {
                MessageBox.Show("Unohdit valita tietokantataulun!");
                return;
            }
            dataSet = new DataSet();
            dataGridView1.DataSource = null;
            dataAdapter =
            new SqlDataAdapter("SELECT * FROM " + comboBox1.SelectedItem,
strConnection);


            try
            {
                dataAdapter.Fill(dataSet);
            }
            catch (Exception asd)
```

```
            {
                //Näytetään virheen tiedot käyttäjälle
                MessageBox.Show(asd.Message);
                return;
            }
            //dataSet:n tiedot näytölle mikäli ei virheitä
            dataGridView1.DataSource = dataSet.Tables[0];
        }

        /// <summary>
        /// Tallentaa käyttäjän tekemät muutokset dataGridView1:stä tietokantaan.
        /// </summary>
        private void button2_Click(object sender, EventArgs e)
        {
            SqlCommandBuilder commandBuilder = new SqlCommandBuilder(dataAdapter);
            dataAdapter.Update(dataSet);
            dataGridView1.DataSource = null;
        }

        /// <summary>
        /// Tyhjentää dataSet:n ja dataGridView1:n.
        /// </summary>
        private void button3_Click(object sender, EventArgs e)
        {
            dataSet.Clear();
            dataGridView1.DataSource = null;
        }

    }
}
//TAULUJEN MUOKKAUS.CS LOPPU-------------------------------------------------------------
--

//TAULUJEN MUOKKAUS.Designer.CS-----------------------------------------------------
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.Sql;
using System.Data.SqlClient;


namespace Taulujen_Muokkaus
{
    partial class Form1
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed;
otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
```

```
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.button1 = new System.Windows.Forms.Button();
            this.dataGridView1 = new System.Windows.Forms.DataGridView();
            this.button2 = new System.Windows.Forms.Button();
            this.label1 = new System.Windows.Forms.Label();
            this.button3 = new System.Windows.Forms.Button();
            this.comboBox1 = new System.Windows.Forms.ComboBox();

((System.ComponentModel.ISupportInitialize)(this.dataGridView1)).BeginInit();
            this.SuspendLayout();
            //
            // button1
            //
            this.button1.Location = new System.Drawing.Point(181, 10);
            this.button1.Name = "button1";
            this.button1.Size = new System.Drawing.Size(99, 23);
            this.button1.TabIndex = 0;
            this.button1.Text = "Hae  data gridiin";
            this.button1.UseVisualStyleBackColor = true;
            this.button1.Click += new System.EventHandler(this.button1_Click);
            //
            // dataGridView1
            //
            this.dataGridView1.AutoSizeColumnsMode =
System.Windows.Forms.DataGridViewAutoSizeColumnsMode.AllCells;
            this.dataGridView1.ColumnHeadersHeightSizeMode =
System.Windows.Forms.DataGridViewColumnHeadersHeightSizeMode.AutoSize;
            this.dataGridView1.EditMode =
System.Windows.Forms.DataGridViewEditMode.EditOnEnter;
            this.dataGridView1.Location = new System.Drawing.Point(15, 49);
            this.dataGridView1.Name = "dataGridView1";
            this.dataGridView1.Size = new System.Drawing.Size(535, 487);
            this.dataGridView1.TabIndex = 1;
            //
            // button2
            //
            this.button2.Location = new System.Drawing.Point(286, 10);
            this.button2.Name = "button2";
            this.button2.Size = new System.Drawing.Size(194, 23);
            this.button2.TabIndex = 2;
            this.button2.Text = "Tallenna muutokset databaseen";
            this.button2.UseVisualStyleBackColor = true;
            this.button2.Click += new System.EventHandler(this.button2_Click);
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Location = new System.Drawing.Point(12, 15);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(34, 13);
            this.label1.TabIndex = 4;
            this.label1.Text = "Taulu";
```

```
            //
            // button3
            //
            this.button3.Location = new System.Drawing.Point(486, 10);
            this.button3.Name = "button3";
            this.button3.Size = new System.Drawing.Size(64, 23);
            this.button3.TabIndex = 5;
            this.button3.Text = "Clear";
            this.button3.UseVisualStyleBackColor = true;
            this.button3.Click += new System.EventHandler(this.button3_Click);
            //
            // comboBox1
            //
            this.comboBox1.FormattingEnabled = true;
            this.comboBox1.Items.AddRange(new object[] {
            "esimTaulu1",
            "esimTaulu2",
            "esimTaulu3"});
            this.comboBox1.Location = new System.Drawing.Point(52, 10);
            this.comboBox1.Name = "comboBox1";
            this.comboBox1.Size = new System.Drawing.Size(123, 21);
            this.comboBox1.TabIndex = 6;
            this.comboBox1.Text = "Choose Table";
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(569, 572);
            this.Controls.Add(this.comboBox1);
            this.Controls.Add(this.button3);
            this.Controls.Add(this.label1);
            this.Controls.Add(this.button2);
            this.Controls.Add(this.dataGridView1);
            this.Controls.Add(this.button1);
            this.Name = "Form1";
            this.Text = "Taulujen Manuaalinen muokkaus";

((System.ComponentModel.ISupportInitialize)(this.dataGridView1)).EndInit();
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.Button button1;
        private System.Windows.Forms.DataGridView dataGridView1;
        private System.Windows.Forms.Button button2;

        private Label label1;
        private Button button3;
        private ComboBox comboBox1;

    }
}
//TAULUJEN MUOKKAUS.Designer.CS LOPPU----------------------------------------------
```