WINDOWS-POHJAINEN TYÖNHALLINTAJÄRJESTELMÄ

C#- ja DOT.NET-tekniikoilla

Kimmo Silander

Opinnäytetyö
Toukokuu 2010

Ohjelmistotekniikka
Tekniikan ja liikenteen ala

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖN
KUVAILULEHTI

Tekijä(t)	Julkaisun laji	Päivämäärä
SILANDER, Kimmo	Opinnäytetyö	10.05.2010
	Sivumäärä 37	Julkaisun kieli Suomi
	Luottamuksellisuus ( ) saakka	Verkkojulkaisulupa myönnetty (X)

Työn nimi
WINDOWS-POHJAINEN TYÖNHALLINTASOVELLUS. C#- JA DOT.NET-TEKNIIKOILLA

Koulutusohjelma
Ohjelmistotekniikan koulutusohjelma

Työn ohjaaja(t)
SALMIKANGAS, Esa

Toimeksiantaja(t)
PIIPPONEN, Kari, Epitek oy

Tiivistelmä

Työn tarkoituksena oli yhtenäistää useita hajanaisia työnhallintajärjestelmiä sulautetuksi kokonaisuudeksi käyttäen Windows DOT.NET-ympäristöä. Sovelluksen primäärinen tarkoitus oli yhtenäinen tietokanta, sen päivitys ja yhtenevät raportointityökalut, korvaten siirrettävillä tietovarastoilla liikkuvan datan. Sovelluksen verkkototeutus pyrki nopeuttamaan yksinkertaistamaan viesti- ja tietoliikennettä, sekä mahdollistamaan sovelluksen käyttämisen riippumatta henkilöstön sijainnista. Raportoinnin helpottamiseksi sovellus poimii valmiiksi annettuja tietoja tietokannasta ja hyödyntää Word-automaatiota, minimoiden näppäilyvirheet ja kirjoitettavan materiaalin määrää.

Projekti toteutettiin C#-ohjelmointikielellä käyttäen Microsoft Visual Studio 2008-sovelluskehitintä. Työkalut valittiin niiden ollessa entuudestaan tutuimmat projektiin soveltuvista vaihtoehdoista. Tietokantana käytettiin Microsoft Access 2007-tietokantaa. Tiedostojen siirto palvelimelle hoidettiin WinSCP-nimisellä FTP-ohjelmistolla. Verkkopuolen testauksessa käytettiin JAMK:n student-verkkoa, johon sijoitetun Access-tietokannan nopeutta ja fyysistä kokoa seurattiin.

Projekti käynnistyi keräämällä henkilökunnan tarpeet ja suunnittelemalla heidän tarpeisiin sopiva tietokanta. Tietokannan käyttöä varten suunniteltiin ja toteutettiin käyttöliittymä, joka mahdollisti tietokannan muokkaamisen ja raporttien laatimisen sen tiedoista. Projekti täytti asiakkaan alkuperäiset vaatimukset systeemille, johon kuului toimialojen tietojen tallennus tietokantaan ja raportointityökalut sekä tietokannan varmuuskopiointi. Projektin tuotoksen laajentamiseen tehtiin suunnitelmia ja sovellus otettiin käyttöön yrityksessä.

Avainsanat (asiasanat)
C#, Access 2007, DOT.NET, Word-automation, FTP, Visual Studio 2008

Muut tiedot

Author(s)
SILANDER, Kimmo

Type of publication
Bachelor´s / Master's Thesis

Date
10052010

Pages
37

Language
Finland

Confidential

( ) Until

Permission for web publication
(X)

Title
WINDOWS-BASED WORK MANAGEMENT SOLUTION. USING C# & DOT.NET TECHNOLOGIES

Degree Programme
Software Engineering

Tutor(s)
SALMIKANGAS, Esa

Assigned by
PIIPPONEN, Kari, Epitek Oy

Abstract

This study aims to integrate multiple scattered systems to one unified solution by using Windows DOT.NET environment. The primary objective of this solution was to produce a shared database, easy updating and uniform reporting tools, replacing the data transferred by mobile data storages. The web section of this application strived to speed and simplify data and information flow, and to enable the use of the database regardless of the current location of an employee. To ease the managing and producing of reports, the application uses Word-automation, which retrieves information straight from the database, minimizing typing errors and the amount of labour.

The project was implemented with C#-language using Microsoft Visual Studio 2008 as a primary programming tool. These particular tools for this project were chosen because they were familiar to the project team beforehand. The database was developed using Microsoft Access 2007. File transfer to the server was made using an FTP program called WinSCP. Testing the speed and capacity of Access's data transfer carried out using JAMK's Student server.

The development of database was started by collecting the needs for the application from employees. A customised user interface was schemed and implemented for modifying project database and to print reports based on the data it holds. The solution, which was produced in, this project, fulfilled the requirements of the assigner. These demands include saving data from all sectors to the database, reporting tools and safe copying of the database. Plans were made concerning the extension of the projects tools and sections managing. The solution of this project was taken into use by the assigner.

Keywords
C#, Access 2007, DOT.NET, Word-automation, FTP, Visual Studio 2008

Miscellaneous

SISÄLTÖ

KUVIOT

SANASTO

Access 2007	Microsoftin tietokantaohjelmisto relaatiotietokantojen luomiseen
BCL	Base Class Library. Valmis luokkakirjasto johon sisältyy valmiita toimintoja yksinkertaisiin ohjelmointitehtäviin.
BindingSource	Toimii rajapintana tietokannan ja käyttöliittymän kontrollien kanssa. Omaa metodeja mm. tiedon muuttumisen seuraamiseen.
C#	Projektiin valittu Microsoftin kehittämä ohjelmointikieli DOT.NET ympäristöön.
CLR	Common Language Runtime
DataSet	DataSet on kokoelma tietokannan sisältämää dataa ja olioita.
DataTable	Virtuaalinen tietokannan taulu, jota voidaan muokata ja muutokset tallentaa halutessa tietokantaan.
DOT.NET	Windows-pohjainen ohjelmistoalusta, joka hyödyntää CLR-, BCL- ja IL-teknologioita.
FTP	File Transfer Protocol. Tiedonsiirtojärjestelmä verkkoyhteyden kautta palvelimelle
IL	Intermediate Language. Välikieli johon ohjelma käännetään ennen syöttämistä prosessorille.
Visual Studio 2008	Microsoftin luoma kehitysympäristö useille ohjelmistokielille kuten C#, J#, C++, Visual Basic yms.
WinSCP	Windows-pohjainen ohjelmisto FTP:n käyttöön.
XML	eXtensive Markup Language. Kevyt ja joustava kuvailukieli, jolla on tärkeä rooli Web- ja palvelinsovelluksissa.

1 TEHTÄVÄ JA TAVOITTEET

1.1 Toimeksiantaja

Projektin toimeksiantajana toimi vuonna 1997 perustettu Vaajakoskella sijaitseva Epitek Oy. Kyseinen yritys tuottaa monialaisia palveluita, joihin kuuluu mm. lämpöpumppujen myynti sekä asennus, puhdastilamittaukset sairaanhoitopiireissä ja graffitinpoisto sekä julkisivujen pesut. Yrityksen asiakaskunta koostuu pääasiallisesti yrityssektorin ja julkisen sektorin asiakkaista, mutta myös yksityiseen sektoriin on tarjolla palveluita varsinkin lämpöpumppujen osalta. Epitek Oy:ssä työskentelee vakituisesti 4 henkilöä, erinäisissä työtehtävissä, kattaen myynnin, toteutuksen, laskutuksen, yms. Toimialojen luonteiden vuoksi henkilökunta matkustaa työssään paljon.

1.2 Projektin kuvaus

Opinnäyteyön lähtökohtana oli Epitek Oy:n tarve yhteneväiselle sovellukselle joka korvaisi nykyisin käytössä olevat työkalut tiedonsiirron ja tallennuksen osalta. Ennen projektia yrityksellä oli käytössä Microsoft Excel:in perustuvat raportit ja tiedostojen tallentaminen toteutettiin kansiorakenteella. Toisin sanoen erillistä järjestelmää tiedonsiirtoon tai yleiseen jakamiseen ei ollut. Tiedonsiirto tapahtui tulostettavina työmääräiminä, jotka toimitettiin laskutuksen toteuttavalle henkilölle. Käyttöön haluttiin sovellus, joka mahdollistaisi resurssien jakamisen riippumatta henkilöstön sijainnista.

Kuten edellä mainittu, Epitek Oy:llä on eri toimialoja, jotka tuottavat paljon erilaista materiaalia. Ongelmana onkin tiedon ”työpisteytyminen”, eli eri osioiden materiaalit ja tiedot ovat saatavilla ainoastaan kyseisen toimialan työpisteeltä. Myöhemmin laskutukseen siirryttäessä tiedot pitää manuaalisesti kerätä eri toimialoilta ja toimittaa eteenpäin.

Projektin tavoitteena oli tutkia mahdollisia eri keinoja yhtenäistää ja helpottaa tiedonsiirtoa, sekä toteuttaa kyseisellä metodilla verkkopohjainen sovellus, jonka avustuksella tieto ja materiaalit, kuten Excel- ja Word-tiedostot, ovat kaikkien käytettävissä ja saatavilla missä vain, kuitenkaan unohtamatta vaivatonta käyttöliittymää. Muun

muassa asiakasrekisteri ja erinäiset raportit ovat tärkeitä kaikille toimialoille niin laskutuksessa kuin töiden järjestelemisessä. Verkkopohjaisen toteutuksen takia jokainen toimiala toimii itsenäisesti, mutta yrityksen johtokunnalla on mahdollista seurata töiden edistymistä, asiakkaiden aktiivisuutta ja toistuvuutta, toimia asianmukaisin toimenpitein ja luoda raportteja tilanteesta.

Projekti piti suunnitella yrityksen henkilöstön käyttöön ja sen piti olla pääsääntöinen väline tiedonsiirrossa ja tietojen tallentamisessa. Sovellus tulisi käyttöön jokaiselle toimialalle sisältäen niille tärkeät toiminnallisuudet omissa osioissaan. Järjestelmän alkuperäiset vaatimukset olivat vain tärkeimmät toiminnot. Sovellus suunniteltiin laajennettavaksi, koska yrityksellä oli suunnitelmia uusille työkaluille. Laajennuksen tärkeys koskisi myös uusien toimialojen liittämistä sovellukseen. Tavoitteena oli toteuttaa aluksi ainakin asiakasrekisteri, graffitinpoisto ja lämpöpumppujen tietojen tallentaminen ja näin yhdistää Excel-tiedostojen hajanaisuus yhteen hallittuun kokonaisuuteen. Mikäli projektin aikataulu antaisi myöten, seuraavaksi tärkeimmät ominaisuudet olisivat raportointi ja verkkototeutus. Näillä ominaisuuksilla pyrittiin yhtenäiseen raportointiulkoasuun ja tiedon helppoon levittämiseen. Käyttöliittymän oppimiskäyrä piti olla matala, koska yhtenä projektin tavoitteena oli helpottaa ja nopeuttaa työskentelyä. Myös henkilöstön tietotekniset vaatimukset haluttiin pitää alhaalla ongelmien välttämiseksi.

Projektin tuottaman sovelluksen piti pystyä samoihin tai vastaaviin toimenpiteisiin tiedon tallennuksen osalta kuin käytössä olleet Excel-tiedostot. Suurempia etsintätyökaluja ei henkilöstön mukaan tarvittu, paitsi puhdastilamittauksessa sen suuren listauksen vuoksi. Sovelluksen haluttiin pysyvän käytössä mahdollisimman pitkään, joten projektin määrittelyvaiheessa otettiin huomioon uudenaikaisimmat tai pitkäikäisimmällä ennusteella olevat teknologiat kuitenkaan luottamatta liikaa niiden päivitettävyyteen tai tuotetukeen. Määrittelyvaiheessa nousi esille muutamia erityyppisiä ratkaisuja. Yhtenä vaihtoehtona oli täysin verkkopohjainen ratkaisu Java-teknologioilla. Tämän vaihtoehdon projektiryhmän hylkäsi, koska selainpohjaista sovellusta ei nähty mielekkääksi. Tultiin tulokseen toteuttaa projekti Windows-sovelluksena, joka hyödyntäisi verkkoyhteyttä.

Lopullisena hyötynä toimeksiantajalle projektin tuottamasta sovelluksesta on tiedon ja materiaalin (valokuvat ja raportit) saaminen reaaliaikaisesti ja yhdestä selkeästä paikasta. Tämä vähentää tiedon etsimiseen kuluvaa aikaa ja nopeuttaa työprosessiin kuluvaa kokonaisaikaa. Esimerkiksi laskutuksessa tiedot saadaan suoraan tietokannasta, eikä tietoja tarvitse erikseen siirtää toiselle koneelle. Kuten edellä mainittiin, sovellus suunniteltiin kattamaan kaikki osa-alueet perustarpeineen, mutta sen täytyi olla helposti laajennettavissa uusia toimintoja varten, tämän vuoksi oli tarpeellista segmentoida kaikki toimialat omiin paketteihinsa. Tällä tavalla laajentaminen olisi mahdollisimman helppoa, eikä sovellusta päivittäessä tarvitsisi koskea muihin osa-alueisiin.

Koska sovellus toteutettiin yrityksen käyttöön, oli tietokannan ja sovelluksen oltava luotettavia, eikä tietojen katoaminen tai päätyminen vääriin käsiin ollut sallittua. Tästä syystä tietokannan varmuuskopiot suunniteltiin tallennettaviksi ulkoisiin lähteisiin, kuten CD/DVD -levyt, ulkoiset kiintolevyt tai USB -muistit, jolloin palvelimen pettäessä yrityksen tiedot olisivat fyysisesti tallessa.

2 KEHITYSYMPÄRISTÖ

2.1 Ohjelmointikielen valinta

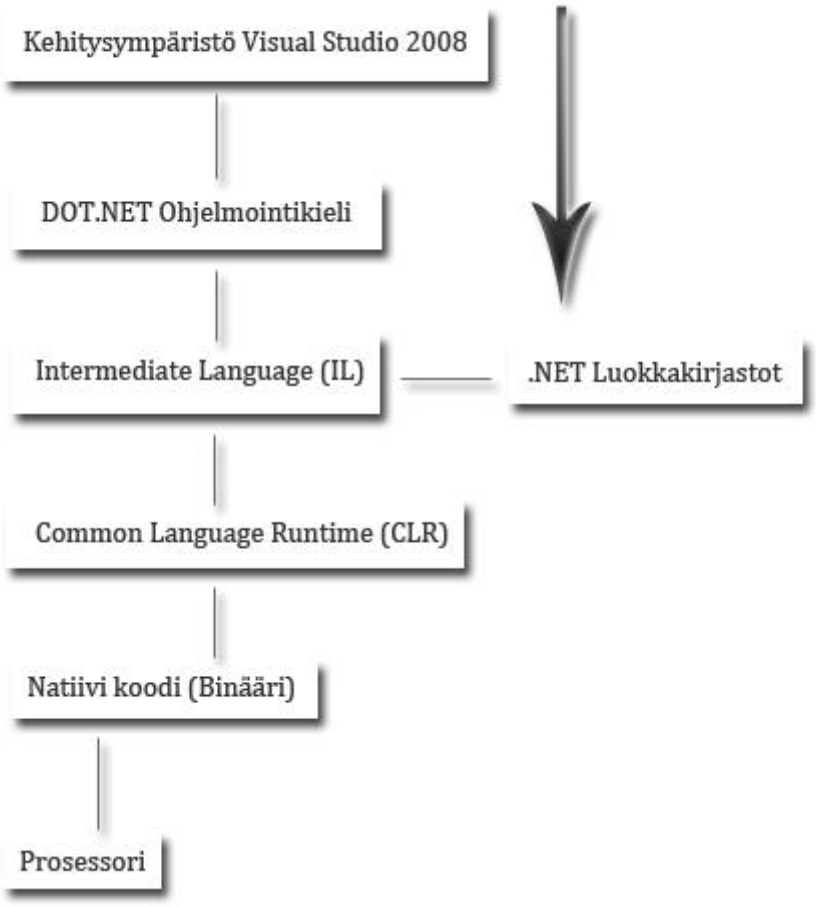
Projektin alkaessa suurimpana kysymyksenä oli millä ohjelmointikielellä sovellus toteutettaisiin. Kuten edellä mainittu, oli työnantajan vaatimuksena Windows-sovellus. Projektiin haluttiin valita kieli, jolla olisi mahdollisimman monipuoliset ominaisuudet toteuttaa tietokantayhteyksiä, sekä myös mahdollisesti raportteja. Projektiin valittu C# oli selkein vaihtoehto sisältäen mahdollisuuden Word- ja Excel-automaation sekä suhteellisen yksinkertaiset metodit verkkototeutukseen. Kielenä C# oli tuttu Jyväskylän Ammattikorkeakoulun järjestämiltä kursseilta. Koska C# pohjautuu C/C++ -kieleen, oli sen oppimiskäyrä suhteellisen matala C/C++:n ollessa projektiryhmälle ehdottomasti tutuin ohjelmointikieli.

2.2 .NET

.NET on Window-komponentti sekä ohjelmistoalusta, joka mahdollistaa ohjelmien kääntämisen ja ajamisen käyttöliittymässä. .NET koostuu osioista kuten CLR, IL , VOS, VES. Näistä tärkeimpinä osioina voidaan pitää CLR:a ja IL:a. Microsoftin sivuilla .NET:a on luonnehdittu seuraavasti: “Microsoft .NET is a set of Microsoft software technologies for connecting information, people, systems and devices.” (Microsoft DOT.NET. 2010).

DOT.NET toimii siis rajapintana sovellusten ja koneen välissä. DOT.NET Frameworkista on useampia versioita, jotka kaikki toimivat omina sovelluksinaan. Toisin sanoen yksi tietokone voi sisältää useamman Frameworkin.

Kuviossa 1 on esitetty DOT.NET –sovelluksen elinkaari. Kuvio kuvastaa sen kääntämisen jälkeiset tapahtumat prosessorille asti. Kehitysympäristön tuottama koodi käännetään Intermediate Languageksi, eli välikieleksi ja välitetään CLR:lle joka puolestaan kääntää välikielen natiiviksi binääriksi prosessorille.



Kuvio 1. Sovelluksen elinkaari kehitysympäristöstä prosessorille.

2.2.1 CLR

CLR, eli Common language Runtime on kieliriippumaton kääntöympäristö joka toimii rajapintana käyttöliittymän ja sovelluksen välissä. Sen tehtävänä on kääntää DOT.NET –kielien tuottama välikieli natiivi-kieleksi prosessorille. CLR:ssa huomattavaa on että ohjelma käännetään vain pyydettäessä, eli esimerkiksi metodeja kutsuttaessa. Käännöksestä huolehtii JIT, eli Just In Time-compilation. CLR:n huolehdittavaksi jää mm. muistinhallinta, säikeiden toteutus, roskien keruu, virheidenkäsittely, yms.

2.2.2 IL

Il on lyhenne sanoista Intermediate Language, eli käännettynä välikieli. On olemassa useampia ohjelmointikieliä, joilla voidaan tuottaa DOT.NET-pohjaista lähdekoodia. Näitä ovat muun muassa projektin ohjelmointikieleksi valittu C# sekä Visual Basic, J# yms. Riippumatta käytetystä kielestä käännetään lähdekoodi Intermediate Languageksi (IL, aiemmin MIL Microsoft Intermediate Language). IL on välikieli, jonka etuna on prosessoririippumattomuus, mutta vaatii DOT.NET Framework -tulkin suorittavan tietokoneen puolesta, samaan tapaan kuin Adobe Flash Player. Kehitysympäristön tuottama IL lähetetään CLR:lle, joka prosessoi koodin binäärimuotoiseksi nk. natiiviksi koodiksi. Binäärikoodi lähetetään edelleen tietokoneen prosessorille suoritettavaksi. Kehitysympäristön kääntäessä koodin ja lähettäessä sen CLR:lle saa se paketin mukana myös ns. ”ylimääräistä” tietoa eli metadataa. Käännettäessä projekti sovellukseksi tallennetaan IL ja metadata jatkoa varten sovelluksen käytettäväksi. (C-sharp-online 2010.)

2.2.3 JIT

Just In Time Compilation. DOT.NET Sovellusta käynnistettäessä Common Language Runtime kääntää välikielen binääriksi. Tämä tapahtuu poikkeuksellisesti sovellusta käytettäessä. Verraten normaalisti sovellus käännetään sen lähtiessä sovelluskehittimestä. (DOT.NET-guide. 2010.)

Sen sijaan että käytettäisiin aikaa ja muistia koko välikielen kääntämiseen, käännetään vain tarvittavat osiot ja tallennetaan ne jatkokäyttöä varten.

2.3 Versiot

DOT.NET Frameworkin ensimmäinen versio 1.0 ilmestyi 2002. Tästä paranneltu versio 1.1 julkaistiin 2003 ja 2.0 vuonna 2005. 1.1 oli ensimmäinen versio joka ilmestyi Windows käyttöliittymän mukana. Uusin versio projektin toteutuksen aikaan (huhtikuu 2010) oli DOT.NET Framework 4. Yrityksen laitteiston kokoonpanosta johtuen projekti toteutettiin Framework 3,5:llä. Se mitä versiota Frameworkista käytetään,

riippuu tietokoneen sisältämistä tai siihen asennettavista ohjelmista ja niiden vaatimasta versiosta. (Microsoft DOT.NET 2010.)

2.4 C#-ohjelmointikieli

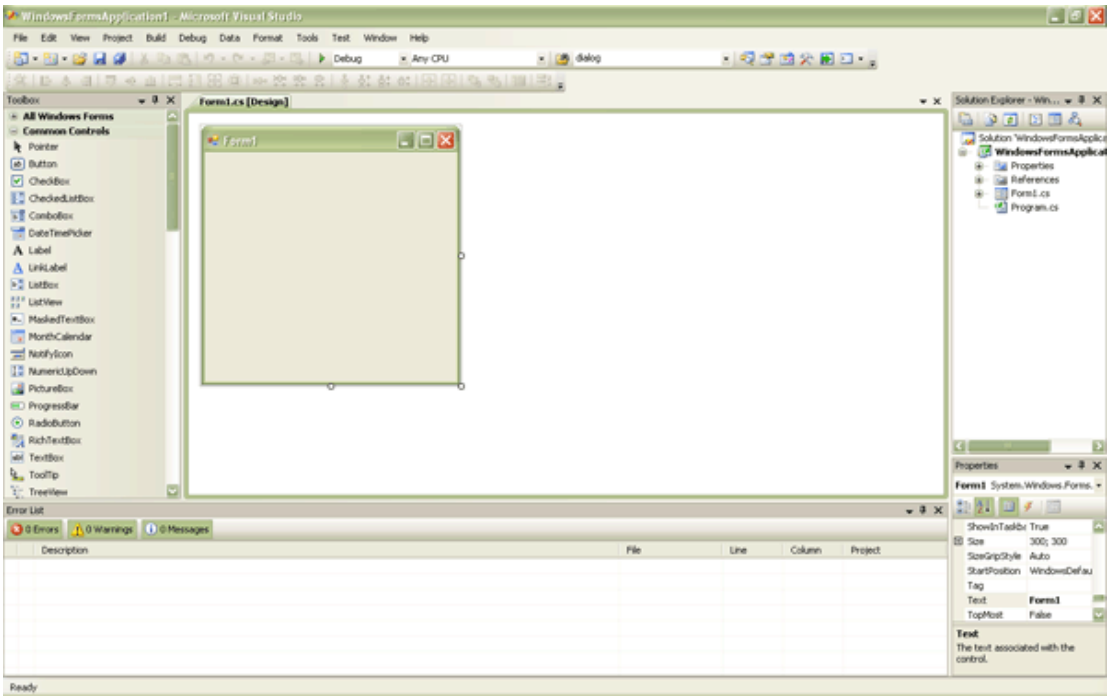
C#-kieli on Microsoftin vuonna 2000 julkaisema oliosuuntautunut ohjelmointikieli, joka suunniteltiin helposti omaksuttavaksi, mutta kuitenkin voimakkaaksi ja monipuoliseksi käyttää. Se pyrkii yhtä helppoon syntaksiin kuin Java, mutta sisältämään C/C++-kielen monipuolisuuden ja tehon. Vaikkakin C# jakaa C-kielen kanssa suurimman osan sen toiminnoista ja tavoista, eroaa se perinteisestä C-kielestä monella tapaa. Muun muassa osoittimet on karsittu pois melkein kokonaan eikä kieli tue enää tyypittömiä muuttujia. C#:in on lisätty myös Javasta tuttu ”roskien keruu”, eli muistien manuaalista poistoa ei tarvitse enää hoitaa itse esimerkiksi olioita luodessa. Tämä hoituu kyseisessä kielessä automaattisesti.

Suosituin C#:n käyttökohde ovat Windows sovellukset, vaikkakin se suoriutuu myös mm. konsolipohjaisista sovelluksista. Windows-sovelluksessa ikkunat, eli formit, ovat luokkia ja kaikki niihin lisätyt kontrollit ovat luokan ilmentymiä, olioita. Näille olioille on BCL:ssa valmiita metodeja, kuten klikkaus, hiiren lukeminen yms. Tämä helpottaa kontrollien tapahtumankäsittelyä nopeuttaen toiminnallisuuden ohjelmointia. C# on yksi yleisimmistä DOT.NET-ohjelmointikielistä Visual Basicin ohella.

2.5 Visual Studio 2008

Projektin pääasiallisena kehitysympäristönä käytettiin Microsoft Visual Studio 2008 Professional Editionia. Visual Studio tarjoaa mahdollisuuden useampaan ohjelmointikieleen, kuten C#, J#, C++, Visual Basic yms. Visual Studio sisältää ohjelmiston kääntömahdollisuuden ja julkaisumahdollisuuden sekä visuaalisen käyttöliittymän suunnitteluosion, joka mahdollistaa painikkeiden, tekstikenttien yms. luomisen. Lähdekoodipuolella Visual Studio hyödyntää IntelliSense-tekniikkaa, eli ohjelma ehdottaa mahdollisia vaihtoehtoja käyttäjän kirjoittaessa. Tämä helpottaa mm. metodien ja

muuttujien käyttöä, koska muuttujaa ei ehdoteta, ellei se kuulu kyseiseen lohkoon tai luokkaan. Visual Studion valintaan yksi suurimmista tekijöistä oli sen laaja-alainen käyttö, helppo saatavuus ja tukipalvelujen runsaus. Kuviossa 2 on esitetty Visual Studio 2008:n standardi käyttöliittymä, jossa esillä juuri luotu tyhjä projekti.



Kuvio 2. Microsoft Visual Studio 2008 –sovelluskehittimen käyttöliittymä.

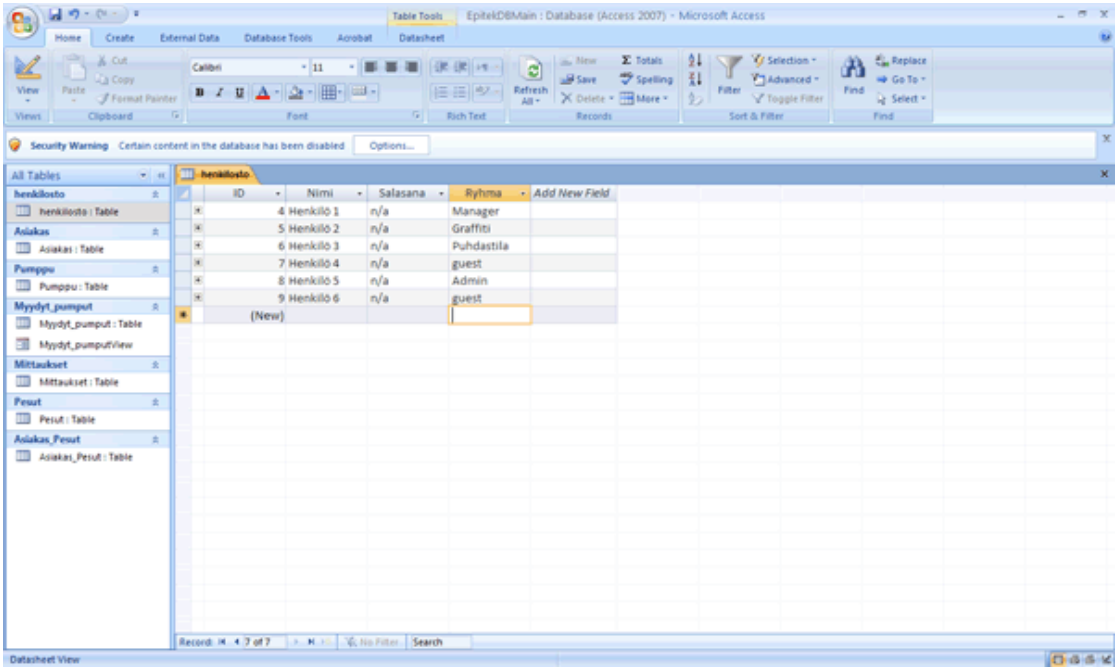
Visual studioon on mahdollista lisätä jälkikäteen kolmannen osapuolen valmistamia laajennuksia esimerkiksi graafisia kirjastoja tai nopeampia prosessorikohtaisia kääntäjiä. Visua l Studion ensimmäinen versio ilmestyi vuonna 1997 Windows 95 - ja Windows 98 –käyttöjärjestelmille. Kehitysympäristö sai nimekseen Visual Studio 97 ja se oli ensimmäinen Microsoftin julkaisema kehitysympäristö jolla oli mahdollista tuottaa ja kääntää useampaa eri ohjelmointikieltä. Ensimmäisissä versioissa (Visual Studio 97 ja 6.0) käytössä oli lähinnä C++- ja J++-kielet. Mahdollisuus DOT.NET ohjelmointiin lisättiin vuonna 2002 microsoftin julkaistaessa Visual Studio .NETin. Tällöin käyttöön tuli myös ensimmäistä kertaa CLR, eli kaikkien ohjelmointikielien toteutukset käännettiin standardisoituun kieleen. Tästä johtuen ohjelmia voitiin suorittaa alustariippumattomasti, kunhan kyseinen kone sisälsi vaaditun version DOT.NET Frameworkista. Vuonna 2oo2 julkaisiin myös ensimmäinen versio kyseisestä Frameworkista, eli versio 1.0.

2.6 Access-tietokanta

Projektin tietokantasovellukseksi valittiin Access, joka on Microsoftin kehittämä relaatiotietokanta. Accessin ensimmäinen versio julkaistiin vuonna 1992. Se eroaa muista kilpailijoistaan lähinnä graafisen ulkoasunsa puolesta. Juuri graafinen ulkoasu oli yksi tärkeimmistä ominaisuuksista suunniteltaessa helpommin lähestyttävää ja työskentelyä nopeuttavaa ratkaisua. (Microsoft Access. 2010. Homepage).

Access-tietokannan vahvuutena markkinoilla on sen soveltuminen sekä graafiseen tietokannan käsittelyyn, että SQL-pohjaiseen tekstiversioon. Toisin sanoen Access-tietokanta voidaan luoda joko käyttäen sen omaa kehitysympäristöä tai antaa sille käskyt SQL-muodossa. Tämä ominaisuus tekee siitä monipuolisemman ja on laajentanut Access:n käyttäjäkuntaa myös vannoutuneiden SQL-käyttäjien joukkoon. Tietokannan luomisen jälkeen on se käytettävissä myös SQL-komennoilla, monipuolistaen sen käytettävyyttä.

Projektissa käytettiin Access 2007 –versiota, joka oli projektin aikana uusi version kyseisestä ohjelmasta. Toimeksiantajalla oli lisenssi



Kuvio 3. Access 2007 -käyttöliittymä

2.7 Tietoperusta

Tietoperustana projektille toimi Jyväskylän Ammattikorkeakoulun tarjoamat ohjelmointi- sekä projektikurssit. Kohdistettua osaamista projektin vaatimista yksityiskohdista saatiin käyttöliittymäohjelmointikurssilta, jonka sisältöä laajennettiin muutamalla valikoidulla alan teoksella jotka kattoivat projektiin tarvittavat teknologit ja teorian. Suoritetut kurssit antoivat pohjan m. ohjelmointitehtäviin, projektinhallintaan, ohjelmistosuunnitteluun, sosiaalisiin taitoihin, raportointiin ja moniin muihin projektin kannalta tärkeisiin osa-alueisiin. Projektin tiedonhaussa käytettiin myös lukusia lähteitä internetistä. Tietojen luotettavuus varmistettiin useimmalta verkkosivulta, sekä vertailemalla sivujen luojien ammattimaista teknistä taustaa ohjelmistotekniikassa.

2.8 Lisenssit

Toimeksiantajalla oli projektin alkaessa valmiina lisenssi Access-tietokantaan. Koska projektiryhmällä ei ollut käytössään kehitysympäristöjä, päätettiin työkalut hankkia työnantajan käyttöön. Visual Studiosta on saatavilla erityyppisiä versioita ilmaisena jakeluna, mutta näillä ohjelman kääntäminen yrityksen käyttöön ei ole mahdollista.

3 KUVAUS PROJEKTIN ETENEMISESTÄ

3.1 Projektin aloittaminen

Projekti aloitettiin kokouksessa 20.10.2010, jossa määriteltiin ohjelmiston tarve ja sen sisältämät osiot. Henkilökunnan toiveiden mukaan tehtiin vaatimusmäärittely, jossa oli listattuna tärkeimmät toteutettavat ominaisuudet henkilökunnan työn helpottamiseksi. Vaatimusmäärittelyn perusteella suunniteltiin, mitä teknologioita hyödynnettäisiin projektissa ja päädyttiin toteuttamaan se C#-kielellä ja Access 2007-tietokannalla. Sovelluskehittimenä käytettiin Microsoft Visual Studio 2008 Professional Editionia, koska se oli projektiryhmälle tutuin ympäristö ja näin nopeutti projektin kulkua. Projektityökalut päätettiin asentaa kahdelle tietokoneelle, jotta siirtäminen ja esittäminen olisi mahdollisimman helppoa. Kaikki materiaalit varmuuskopioitiin myös kahteen eri sijaintiin (verkkolevy ja siirrettävä kovalevy).

Aloituskokouksessa läpikäytiin salassapidon yksityiskohdat, eli lähinnä mitä tietoja yrityksestä ja sen toiminnasta saisi julkaista. Kävimme lävitse myös yrityksen toimintatavat ja eri toimialojen yleiskatsauksen. Projektin alussa keskusteltiin jokaisen toimialan yksilöllisestä tarpeesta sovellukselle ja mahdollisista erityisistä työkaluista joita ne tarvitsisivat. Tämän perusteella alettiin suunnitella projektissa käytettäviä työkaluja ja kehitysympäristöjä.

Ennen varsinaista suunnittelua päätettiin tutustua eri toimialojen toimintaan ja niiden nykyiseen tiedonsiirtoon. Toimialojen huomattiin tuottavan materiaalia ja tallentavan ne tietokoneiden kiintolevyille kansiorakenteisiin. Kaikkien toimialojen toimintatapa oli suhteellisen lähellä toisiaan. Ainoastaan tiedostorakenteen muoto vaihteli. Tästä johtuen päädyttiin suunnittelemaan yhtenäinen tallennusmuoto ja raporttiformaatti yritykselle. Koska kaikki yrityksen tilauksen ja materiaali tallennetaan sovelluksen kautta yhteiseen tietokantaan, on yhtenäinen tietojen tallennusformaatti huomattavasti selkeämpi ratkaisu kuin hajautettu malli samasta aiheesta.

3.2 Laitteisto

Projektin alkaessa yrityksellä oli käytössään kannettavia tietokoneita. Jokaisella toimialalla oli oma tietokone, jossa säilytettiin kaikki toimialan tuottama materiaali ja data. Koska projektin tarkoituksena oli yhtenäistää yrityksen materiaalit ja data oli käytettävän palvelimen rooli keskeisessä asemassa. Keskusteltiin oman palvelimen ostamisesta ja asentamisesta yrityksen toimitiloihin, mutta lopulta päädyttiin käyttämään samaa palvelinta, jota käytetään mm. yrityksen verkkosivujen ylläpitoon. Tulokseen päädyttiin projektin kehittyvän luonteen takia. Ei haluttu kuluttaa projektin aikaa ja varoja tulevaisuudessa mahdollisesti turhaan investointiin. Palvelimen hankinta, asentaminen ja käyttöönotto olisivat syöneet todella paljon projektin ajasta ja henkilökunnan panoksesta.

Tietokoneiden käyttöliittyminä toimivat projektin alkaessa Windows XP Home Edition ja DOT.NET Framework 3.5. Projektin loppupuolella yrityksen tietokoneet uusittiin. Uudet koneet sisälsivät Windows 7 Home Premium-käyttöliittymän. Uuden käyttöliittymän tuomista muutoksista keskusteltiin, mutta koska DOT.NET Framework versio oli edelleen 3.5, ei tietokoneiden vaihto aiheuttanut toimenpiteitä.

Tietokoneiden suuri suorituskyky ei ollut olennainen tässä projektissa, sovelluksen sisältämän pienen graafisen sisällön johdosta. Suurimpina vaatimuksina käytettävälle tietokoneelle oli sisältää lisensoidut versiot Office 2003- tai 2007-paketteista ja verkkoyhteys, koska järjestelmä käyttää Word- automaatiota, sekä Access-tietokantaa verkkoyhteyden kautta. Verkkoyhteyden nopeuden ei todettu ratkaisevasti vaikuttavan kuvien siirtonopeuteen. Lopulta palvelimen fyysinen koko rajoittaa siirrettävien kuvien määrän niin pieneksi, että lopullinen ajansäästö nopeammalla yhteydellä on minimaalinen. Yrityksen käytössä oleva verkkoyhteys oli riittävän nopea kuvien ja datan siirtämiseen palvelimelle.

3.3 Tiedonsiirron testaus

Verkkoyhteydet ja tietokannan nopeus testattiin Jyväskylän Ammattikorkeakoulun Student palvelimella, joka on tarkoitettu koulun opiskelijoiden henkilökohtaiseen

käyttöön opintoihin liittyen. Ongelmia tiedonsiirron nopeudessa ei ollut, mutta kaupallisten palvelimien kiintolevykapasiteetti asettaa tiettyjä rajoituksia. Kuvien siirtäminen pysyvästi palvelimelle on mahdotonta niiden vaatiman tilan vuoksi. Lisäksi useampia kuvia siirettäessä samanaikaisesti on huomattavissa selkeää hitautta. Päädyttiin ratkaista ongelma pienentämällä siirrettävien kuvien kokoa, jolloin myös kuvan vaatima fyysinen koko pienenee. Kuvamateriaalissa jota yritys tuottaa maksimaalinen tarkkuus ei ole tarpeellista. Tämä mahdollistaa kuvan pienentämisen ja tarpeen tullen tarkkuuden pienentämisen nopeuden optimoimiseksi.

3.4 Arkkitehtuuri

Sovelluksen arkkitehtuuri perustuu yksinkertaiseen kokoonpanoon, joka koostuu kannettavista tietokoneista ja palvelimesta, jossa tietokanta sijaitsee. Tietokoneet muodostavat käyttöliittymän kautta verkkoyhteyden palvelimeen ja sen sisältämään Access 2007 -muotoiseen tietokantaan. Tämän tapainen ratkaisu mahdollistaa tietokannan mobiilin käsiteltävyyden. Henkilökunta pystyy tallentamaan työnsä riippumatta sijainnistaan, kunhan verkkoyhteys on saatavilla.

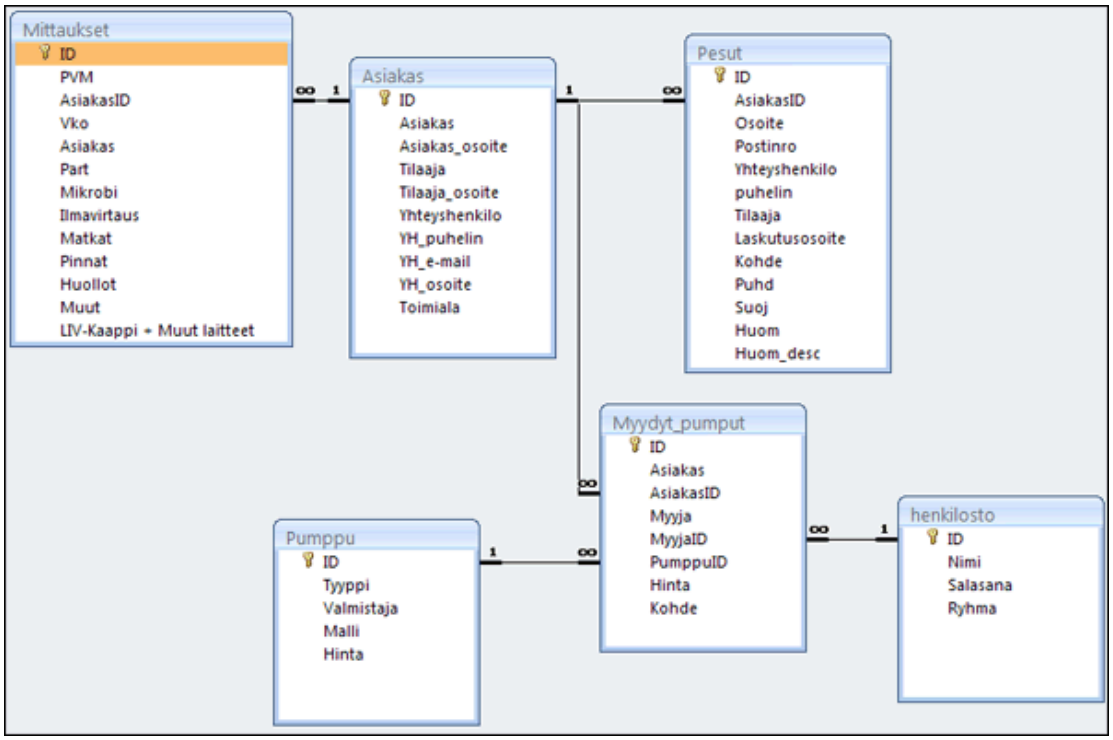
3.5 Sovelluksen graafinen ulkoasu

Sovittiin sovelluksen noudattavan yrityksen värimaailmaa, joka koostuu sinisen eri sävyistä. Pääasiallisena tunnuksena käytettiin yrityksen omaa logoa ja banneria. Ulkoasun haluttiin olevan ajattoman yksinkertainen ilman suurempia graafisia hienouksia. Muun muassa painikkeet ja tekstikentät päätettiin toteuttaa standardeilla grafiikoilla muokaten ainoastaan tekstin väriä ja fonttia.

3.6 Tietokannan suunnittelu

Tietokanta sisältää kaikki yrityksen resurssit ja tiedot yhdessä tiedostossa. Kuten luvussa 1.2 on mainittu, haluttiin tietokannalla ehkäistä materiaalien työpisteytymistä.

Tästä johtuen tietokanta päätettiin sijoittaa palvelimelle, verkkoyhteyden kautta käsiteltäväksi. Tietokannan taulujen lopullinen rakenne on esitetty Kuviossa 4.



Kuvio 4. Tietokannan taulukkorakenne

3.7 Tietokannan toteutus

Tietokanta toteutettiin relaatiomallisena Microsoft Access 2007-tietokantana. Idea oli toteuttaa tietokanta, johon olisi mahdollisimman helppo päästä käsiksi, olisi helppo päivittää ja toimivarma. Tietokannaksi valittiin Access-tietokanta, koska yrityksellä oli valmiiksi ostettu lisenssi kyseiseen ohjelmistoon. Toisena etuna on Access-tietokannan käytettävyys. Access-tietokanta on fyysisesti yksi tiedosto muiden joukossa, eikä näin ollen vaadi erityistä tukea palvelimelta, toisin kuin esim. MySQL. Selkeä tiedostomuotoisuus tekee sen siirtämisestä, käyttämisestä ja varmuuskopioinnista helpompaa. Haittapuolina valitussa tietokannassa oli sen fyysinen koko palvelimen kiintolevyllä sekä prosessoinnin raskaus tietyissä operaatioissa. Projektin kannalta merkittävä ominaisuus valitussa tietokannassa oli sen kyky käsitellä myös SQL-tyyppisiä syötteitä. Tämä helpotti huomattavasti taulujen ja niiden sisältämän tietojen käsittelyä.

Ennen projektin aloittamista kaikki tieto oli tallennettu yksittäisiin, hajanaisiin Excel-tiedostoihin. Varsinaista varmuuskopiointia ei ollut käytössä, mikä teki tiedostorakenteesta suhteellisen riskialttiin vahingoille. Projektin tietokannan varmuuskopiot ovat tallennettavissa Access 2007 -tietokantana vapaasi valittavaan kohteeseen. Järjestelmä tallentaa myös kuukauden välein varmuuskopion tietokannasta tietokoneen kiintolevylle korvaten edellisen version. Tämän kopion on tarkoitus toimia hätätilanteessa, jossa senhetkinen tietokanta on tuhoutunut tai muokkautunut laaja-alaisesti virheelliseksi.

Tietokanta siirrettiin yrityksen palvelimelle WinSCP-nimisellä sovelluksella. WinSCP-on ilmainen FTP-sovellus ja se valittiin projektiin koska se oli kustannustehokas ja työryhmälle entuudestaan tuttu Jyväskylän Ammattikorkeakoulun ohjelmointikursseilta.

3.8 Sovelluksen osioiden toteutukset

Sovellus koostuu erillisistä osioista kullekin toimialalle. Toimialat ovat lämpöpumput ja niiden myynti, graffitinpoisto sekä puhdastilamittaus. Lisäksi toteutettiin asiakasrekisteri ja käyttäjienhallinta. Kaikki osiot ovat itsenäisiä paketteja, mikä tekee sovelluksesta modulaarisen, eli laajennettavan. Osiot toteutettiin omina yksikköinään, eikä osiot jaa tietoja keskenään, paitsi asiakasrekisterin kanssa.

3.8.1 Valikkorakenne

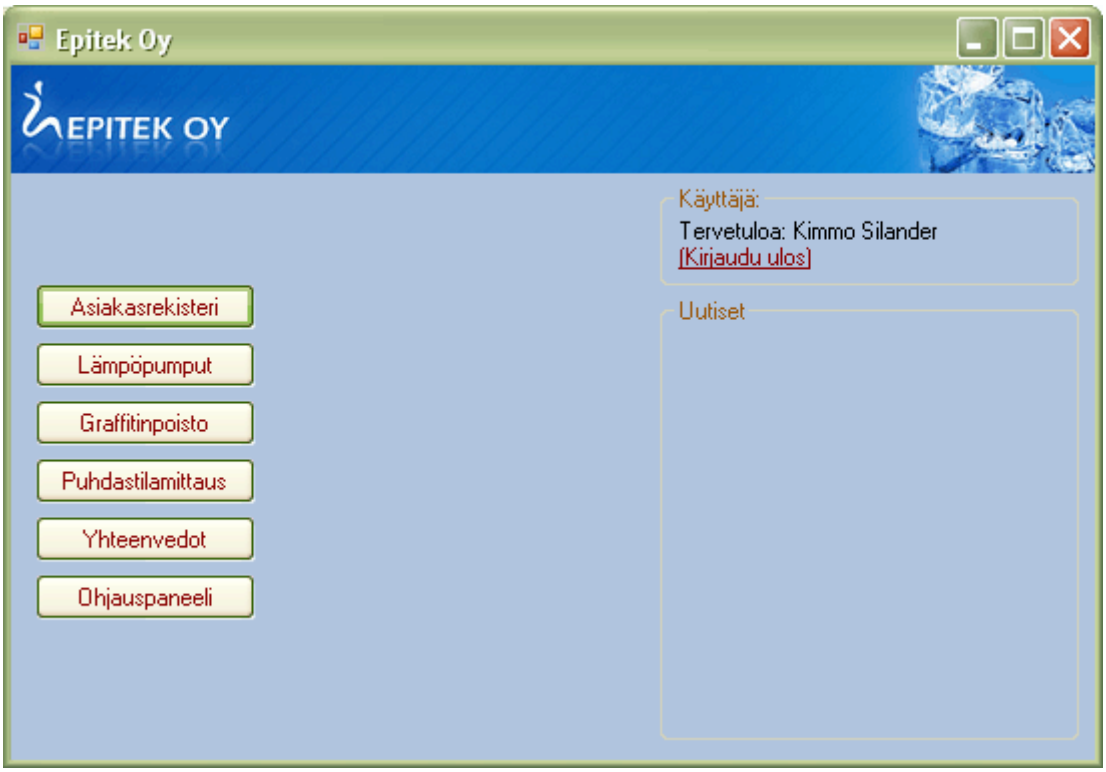
Sovelluksen käynnistyessä avautuu kirjautumisikkuna (ks. kuvio 5). Henkilön syötettyä vaaditut tiedot tarkastetaan tietojen oikeellisuus tietokannan henkilöstötaulusta. Tietojen täsmätessä kirjaudutaan sovellukseen, joka ohjautuu päävalikkoon. Kirjautumisessa poimitaan kirjautuneen henkilön ryhmä tietokannan taulukosta ”Henkilöstö”. Riippuen tästä ryhmästä päätetään, mitä valintoja työntekijälle näytetään. Näin estetään eri toimialojen kajoaminen toisten toimialojen tietoihin. Pahimmassa tapauksessa toisten toimialojen työntekijät voisivat vahingossa tuhota yrityk-

selle tärkeitä tietoja. Toimitusjohtajan on pystyttävä seuraamaan töiden etenemistä, joten hänelle luonnollisesti näytetään kaikki mahdolliset valinnat.

Kuvio 5. Sovelluksen sisäänkirjautuminen

Päävalikosta (ks. kuvio 6) valitaan haluttu osio. Itse valikko jää taustalle auki, jos halutaan esimerkiksi useampi yhtäaikainen seuranta eri toimialoilta. Päävalikossa sijaitsee myös uloskirjautumispainike halutessa vaihtaa käyttäjäprofiilia. Tämä painike vie käyttäjän takaisin kirjautumisikkunaan ja sulkee päävalikon.

Kuviossa 6 on esitetty päävalikon käyttöliittymä. Kuviosta selviää painikkeiden lukumäärä ja niiden pääasialliset funktiot, jotka ovat auki toimitusjohtajalle. Päävalikon oikeassa reunassa sijaitsevat uutiset, joita voi hallinnoida sovelluksen ohjauspaneelin kautta. Uutiset tallentuvat tietokantaa ja ovat siten näkyvillä kaikille käyttäjille, jos verkkoyhteys on mahdollinen. Muutoin käyttäjälle ilmoitetaan verkkoyhteyden puuttuvan, josta syystä uutisia ei esitetä.



Kuvio 6. Sovelluksen päävalikko

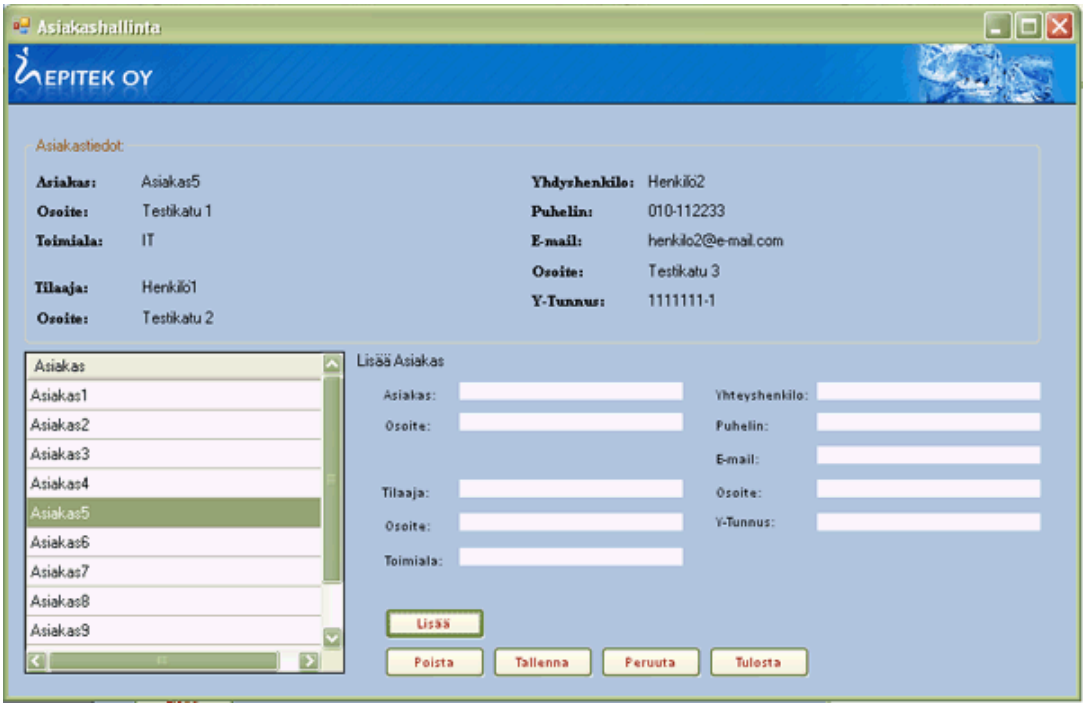
Sovellusta laajennettaessa pystytään päävalikkoon lisäämään uusia valintoja. Näin pystytään helposti hallitsemaan henkilökunnalle näytettävät mahdolliset valinnat. Myös eri toimialojen osioita päivitettäessä on helppo sulkea vain yksittäinen osio, muiden toimiessa omillaan.

3.8.2 Asiakasrekisteri

Asiakasrekisterin pääasiallinen tarkoitus on ylläpitää, lisätä, poistaa ja muokata listaa kaikkien toimialojen asiakkaista. Asiakasrekisteriä käytetään hyväksi muissakin osioissa sen sisältämien tietojen takia, mikä nopeuttaa tiedonkäsittelyä ja raportointia. Asiakaslistan rivit ovat mahdollista poimia DataSet:n kautta muihin käyttötarkoituksiin toisille toimialoille. Esimerkiksi graffitinpoistossa poimitaan valmiit asiakkaat, joiden toimiala on ”graffitinpoisto”, ja näiden asiakkaiden tietoja voidaan käyttää luotaessa uutta tilausta.

Asiakasrekisteri tallentaa kaikki yrityksen asiakkaat riippumatta heidän toimialastaan. Asiakasrekisterillä voi selata asiakkaita toimialan tai toimipaikan mukaan, sekä etsiä yhteystietoja tai laskutusosoitteita.

Projektin päättyessä asiakasrekisteri toimi lähinnä asiakaslistana helpottamassa laskutusta, poimittaessa yhteystietoja sekä nopeuttamassa uusien tilausten luomista poimimalla vanhojen asiakkaiden valmiita tietoja tietokannasta. Projektin jälkeen suunniteltiin asiakaslistan laajennusta uusilla ominaisuuksilla, kuten asiakkaan tuotto yritykselle, sähköpostin lähettäminen, laskujen luonti yms. Nämä ominaisuudet karsittiin pois projektin aikarajoituksien takia. Kuviossa 7 on esillä asiakasrekisterin käyttöliittymä, johon on luotu kokeellinen asiakaslista. Asiakaslistassa on esitetty asiakkaan nimi. Klikkaamalla halutun asiakkaan kohdalta, esitetään valitun rivin tiedot ikkunan yläreunassa sijaitsevassa tietuekentässä. Tämä selkeyttää asiakkaiden listaamista ja mahdollistaa suurien ja sekavien komponenttien pois jättämisen.



Kuvio 7. Asiakasrekisterin käyttöliittymä

Ladattaessa asiakasrekisteri listataan kaikki ”Asiakas”-taulussa sijaitsevat rivit ListBox-objektiin. Asiakkaiden tiedot suodatetaan ja ainoastaan asiakkaan nimi näytetään listattuna. ListBox:a klikattaessa näytetään ylemmässä tietotaulukossa kaikki asiakkaan tiedot. Listan tiedot poimitaan tietokannan dataset:sta luodusta Asiaks-

BindingSourcesta. Tätä Binding Sourcea käytetään myös muiden toimialojen palveluksessa poimimaan asiakastietoja, joita voidaan käyttää hyväksi prosessin nopeuttamisessa.

Asiakkaiden lisääminen tapahtuu klikkaamalla ”Lisää”-nappia, joka lukee tekstikenttien sisältämät merkkijonot ja tallentaa ne väliaikaiseen muotoon ennen tallennusta. Napin painallus kutsuu AddRow-metodia ja lähettää sille parametrina tekstikenttien sisältämät merkkijonot. AddRow-metodi luo uuden rivin (DataRow), joka perustuu ”Asiakas”-taulun kolumneihin. Uuden rivin kolumneihin asetetaan parametreina saadut arvot ja rivi lisätään DataSetiin add-komennolla. Tapahtuman lähdekoodi C#-kielellä toteutettuna on esitetty kuviossa 8.

```
private void LisaaBtn_Click(object sender, EventArgs e)
{
    addRow( AsiakasText.Text, OsoiteText.Text, TilaajaText.Text,
            TilaajaOsoiteText.Text, YhteyshenkiloText.Text,
            YH_puhelinText.Text, YH_emailText.Text, YH_OsoiteText.Text,
            ToimialaText.Text);
}

public void addRow( string attr1, string attr2, string attr3,
                    string attr4, string attr5, string attr6,
                    string attr7, string attr8, string attr9)
{
    DataRow newDataRow = _dsMain.Tables["Asiakas"].NewRow();
    newDataRow["Asiakas"] = attr1;
    newDataRow["Asiakas_osoite"] = attr2;
    newDataRow["Tilaaja"] = attr3;
    newDataRow["Tilaaja_Osoite"] = attr4;
    newDataRow["Yhteyshenkilo"] = attr5;
    newDataRow["YH_puhelin"] = attr6;
    newDataRow["YH_e-mail"] = attr7;
    newDataRow["YH_osoite"] = attr8;
    newDataRow["Toimiala"] = attr9;
    _dsMain.Tables["Asiakas"].Rows.Add(newDataRow);
}
```

Kuvio 8. Uuden rivin lisääminen DataSet’iin.

Rivien poistamisessa käydään silmukassa lävitse kaikki rivit (ks. kuvio 9), kunnes löydetään valittu rivi (SelectedRows). DataGridViewin ominaisuudeksi on määritetty mahdollisuus valita useampi rivi kerrallaan. Painettaessa taulukon yksittäistä solua valitaan koko rivi. Valitut rivit poistetaan väliaikaisesti, eikä alkuperäinen tietokanta muutu ennen tallennuspäätöstä, koska Update-metodia ei kutsuta, jolloin tehtyjä muutoksia ei päivitetä tietokantaan.

```
private void PoistaBtn_Click(object sender, EventArgs e)
{
    foreach (DataGridViewRow dr in dataGridView1.SelectedRows)
    {
        try
        {
          dataGridView1.Rows.Remove(dr);
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }
}
```

Kuvio 9. Rivin poistaminen DataGridView-kontrollista.

Lopullinen tietojen tallennus tietokantaan suoritetaan käyttäjän painaessa ”Tallenna”-painiketta. Käyttäjälle esitetään Windows-formaatin mukainen MessageBox, jossa kysytään, halutaanko muutokset tallentaa vai perua. Vastauksen tuloksen perusteella tiedot joko pyritään tallentamaan tietokantaan tai ei. Käyttäjän valitessa tallentamisen peruuttamisen jätetään väliaikainen muokattu versio näkyville. Jos käyttäjä poistuu osiosta ja palaa, on tehdyt muutokset hävinnyt ja näkyvillä olevat tiedot täsmäävät tietokannassa sijaitsevia. Samalla esitetään käyttäjälle ilmoitus jossa mainitaan tietojen tallentamisen peruttamisesta, eikä tietoja päivitetä tietokantaan. Valittaessa tietojen tallentamisen kutsutaan Validate-metodia varmentamaan tallennettava data sekä Update-metodia päivittämään muokatut tiedot tietokantaan. Kuviossa 10 on esitetty lähdekoodi muutosten tallentamiseen DataSetin ja TableAdapterin avulla. Lopullisen tallentamisen jälkeen käyttäjälle tiedotetaan, onnistuiko tallennus vai aiheutuiko tapahtumassa virhe. Mahdolliset virheet kuvaillaan käyttäjälle messagebox-objektin välityksellä. Yhtenä suurimpana mahdollisena virheenä on verkkoyhteyden katkeaminen tai puuttuminen, jolloin sovellus ei saa yhteyttä tietokantaan. Tällöin käyttäjälle ilmoitetaan verkkoyhteyden puuttuvan ja kehotetaan tarkastamaan yhteys. Muissa mahdollisissa virhetilanteissa lopullisella käyttäjällä ei ole suurempia mahdollisuuksia tilanteen korjaamiseksi. Tällaisia tilanteita ovat mm. palvelimen kaatuminen, tietokannan vahingoittuminen tai tietokoneessa ilmenevä vika.

```
private void SaveBtn_Click(object sender, EventArgs e)
{
    DialogResult dialog = MessageBox.Show(  this, "Haluatko varmasti tallentaa muutokset",
                                            "Tallennus", MessageBoxButtons.OKCancel,
                                            MessageBoxIcon.Asterisk);
    if (dialog == DialogResult.OK)
    {
        try
        {
            this.Validate();
            this.asiakasBindingSource.EndEdit();
            this.asiakasTableAdapter1.Update(this.epitekDBMainDataSet1.Asiakas);
            MessageBox.Show(    this,
                                "Muutokset tallennettu", "Tallennus",
                                MessageBoxButtons.OK, MessageBoxIcon.Information);

        }
        catch (InvalidOperationException IOE_E)
        {
            MessageBox.Show(    this, "Muutoksia ei pystytty tallentamaan",
                                "Virhe", MessageBoxButtons.OK, MessageBoxIcon.Error);
            Console.WriteLine("Error was encountered during action: " + IOE_E);
        }
    }
    else if (dialog == DialogResult.Cancel)
    {
        MessageBox.Show(    this, "Muutoksia ei tallennettu", "Peruutus",
                            MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
}
```

Kuvio 10. Muutosten tallentaminen tietokantaan

Word-automaatio

Word-automaatio tarkoittaa Microsoft Word -asiakirjan luomista koodillisesti. Tälle ominaisuudelle on oma kirjastonsa nimeltä Microsoft.Office.Core. Vaatimuksena tälle kirjastolle on tietokoneelle asennettu Microsoft Office -paketti, jonka referenssi lisätään projektiin. Kun referenssi on lisätty, pystyy sovelluskehitin kutsumaan sen kirjaston metodeja.

Sovelluksen käyttäjä voi halutessaan tulostaa listan asiakkaista mukaan lukien näiden osoitteet, yhteyshenkilöt, s-postit ja toimialat helpottaen laskuttamista. Asiakirjaan poimitaan kaikkien asiakkaiden tiedot tietokannasta ja tulostetaan allekkain dokumenttiin. Word-asiakirja luodaan painettaessa ”Tulosta”-painiketta. Kuviossa 11 on kuvailtuna uuden Word-asiakirjan luominen C#-kielellä. Huomattavaa asiakirjan luomisessa on puuttuvien arvojen käsittely. Word-automaatio ei hyväksy NULL-arvoja, vaan sillä on oma puuttuva arvonsa ”Missing”.

```
object oMissing = System.Reflection.Missing.Value;
object oEndOfDoc = "\\endofdoc"; /* \endofdoc on valmiiksi määritelty objekti */

//Käynnistetään word ja luodaan uusi dokumentti
Word._Application oWord;
Word._Document oDoc;
oWord = new Word.Application();
oWord.Visible = true;
oDoc = oWord.Documents.Add(ref oMissing, ref oMissing,
    ref oMissing, ref oMissing);
```

Kuvio 11. Word-asiakirjan luominen C#-kielellä

Word-asiakirjan perusominaisuuksia komennetaan pääasiallisesti kolmella erityyppisellä oliolla. Kuviossa 10 esiintynyt oWord on olio, joka vastaa Word-sovelluksesta ja sen käskyistä. oDoc-olion kautta puolestaan välitetään käskyjä itse asiakirjalle. Kolmantena oliona on kappale (paragraph) eli oPara. Kappaleella jaotellaan teksti ja määrätään, mihin se sijoittuu asiakirjassa. oPara-olio hoitaa myös tekstin muokkaamisen, kuten lihavoinnin, kursivoinnin, fontin koon, yms. (MSDN Library 2010.)

Kuviossa 12 on esitetty kappaleen luonti, tekstin lisäys, lihavointi ja välistys kahdella kappaleella. Asiakirjan luonnin aloittaessa haasteellisinta Word-automaatiossa onkin kappaleiden jakaminen ja niiden välistäminen sopiviin loogisiin jakoihin. Huomioitavaa on myös se, että loogisesti virheellisestä koodista ei synny virheilmoitusta. Tästä johtuen asiakirja saattaa olla väärin tulostunut, mutta sovelluskehitin ei ilmoita virheestä. Tämä johtuu siitä että koodi itsessään toimii, mutta ohjelmoijan kannalta väärällä tavalla.

```
//Lisätään kappale dokumentin alkuun
Word.Paragraph oPara1;
oPara1 = oDoc.Content.Paragraphs.Add(ref oMissing);
oPara1.Range.Text = "Epitek Oy";
oPara1.Range.Font.Bold = 1;
oPara1.Format.SpaceAfter = 24;    //24 pt väliä kappaleen jälkeen
oPara1.Range.InsertParagraphAfter();

Word.Paragraph oPara2;
object oRng = oDoc.Bookmarks.get_Item(ref oEndOfDoc).Range;
oPara2 = oDoc.Content.Paragraphs.Add(ref oRng);
oPara2.Range.Text = "Lista laskutettavista asiakkaista";
oPara2.Format.SpaceAfter = 6;
oPara2.Range.InsertParagraphAfter();
```

Kuvio 12. Kappaleiden luonti ja tekstin lisäys Word-asiakirjaan

Viimeinen tulostettava osio dokumenttiin saatiin tulostamalla For-silmukassa kaikkien asiakkaiden tiedot. Koska Word-automaatio on toteutettu ohjelmallisesti ja pohjautuu itsekin vahvasti olio-pohjaiseen ohjelmointiin, voidaan myös C#-kielen perus metodeja ja olioita käyttää hyväksi. Esimerkiksi asiakkaiden tulostamisessa voidaan käyttää silmukkarakennetta tekstin toistamiseen ja DataGridViewin (joka siis on Asiakas-luokan olio) sisäänrakennettuja metodeja sen sisältämän tiedon poimimiseen. Kuviossa 13 on esitetty sekä silmukkarakenteen, että oliorakenteen käyttö Word-automaation yhteydessä. Tällaisella ratkaisulla voidaan nopeuttaa asiakirjojen luomista huomattavasti, koska kaikki valmiit tiedot voidaan poimia valmiiksi. Tällöin käyttäjän ei tarvitse erikseen etsiä ja kirjoittaa esimerkiksi osoitteita, puhelinnumeroita tai sähköpostiosoitteita.

```
//Lisätään kolmas kappale.
Word.Paragraph oPara3;
oRng = oDoc.Bookmarks.get_Item(ref oEndOfDoc).Range;
oPara3 = oDoc.Content.Paragraphs.Add(ref oRng);
for (int i = 0; i < dataGridView1.Rows.Count; i++)
{
    // Tulostetan kaikkien rivien [i] sisältämät tiedot
    oPara3.Range.Text =
        "Asiakas: " + dataGridView1.Rows[i].Cells[1].FormattedValue + ", " +
        "Osoite: " + dataGridView1.Rows[i].Cells[2].FormattedValue + ", " +
        "Yhteyshenkilö: " + dataGridView1.Rows[i].Cells[5].FormattedValue + ", " +
        "E-mail: " + dataGridView1.Rows[i].Cells[7].FormattedValue + ", " +
        "Toimiala: " + dataGridView1.Rows[i].Cells[9].FormattedValue;

    oPara3.Range.Font.Bold = 0;
    oPara3.Format.SpaceAfter = 18;
    oPara3.Range.InsertParagraphAfter();
}
```

Kuvio 13. Asiakaslistan lisääminen ohjelmallisesti Word-asiakirjaan

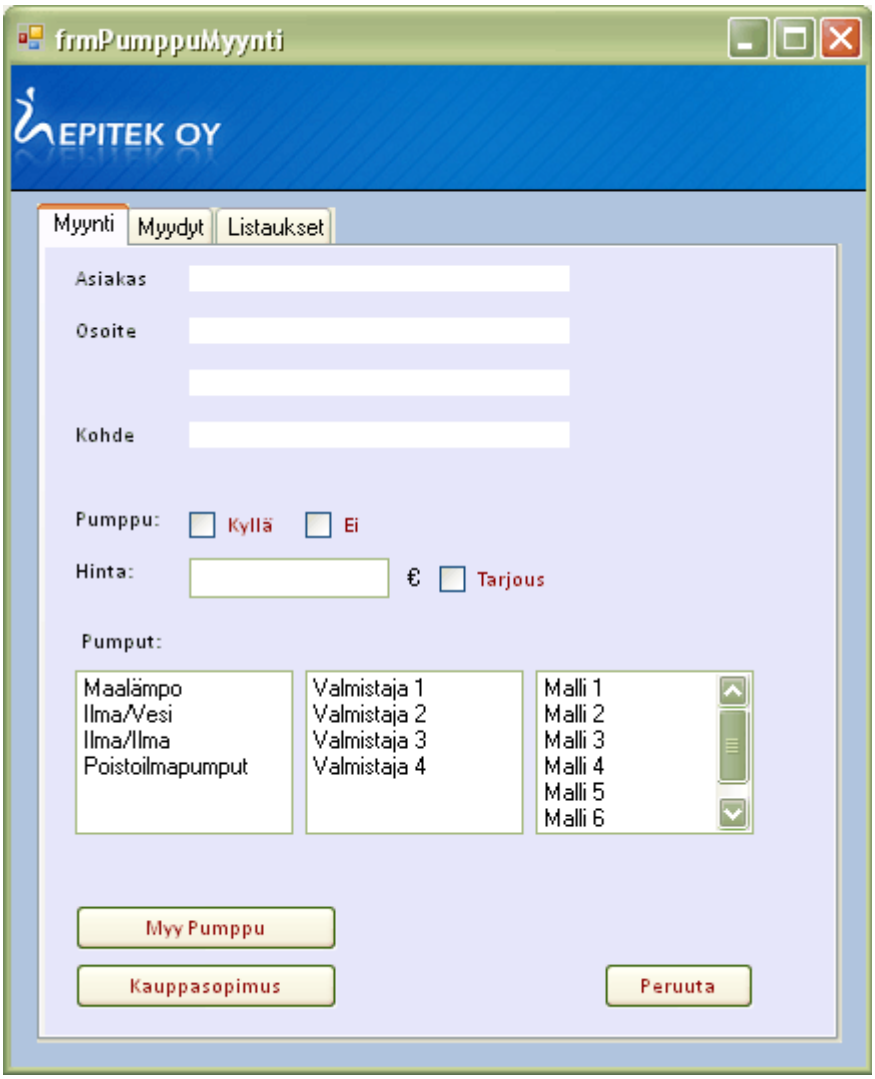
Tulostamisen jälkeen asiakirja jää normaalisti auki Microsoft Wordiin ja sitä voi muokata mielensä mukaisesti tai tallentaa haluamaansa paikkaan. Toisin sanoen Word-automaatio helpottaa käyttäjää luomaan joko valmiin asiakirjan tai osittain täytetyn pohjan viimeisteltäväksi käsin.

3.8.3 Lämpöpumput

Lämpöpumpuille luotiin kaksi taulua ”Pumput” ja ”Myydyt_pumput”, joista ensimmäinen sisältää myynnissä olevat pumput ja niiden tyypit. Tämä taulu varmistaa, että myyntihenkilöstö voi myydä ainoastaan niitä pumppuja, joita on valittu myytävien

listalle. Toisin sanoen ei pystytä myymään sellaisia pumppuja, joita yrityksellä ei ole. Jälkimmäinen taulu sisältää jo myytyjen pumppujen tiedot. Muun muassa onko pumppu jo asennettu vai onko kyseessä tarjous tarjous. Tärkeitä tietoja ovat myös luonnollisesti asiakas, asennuskohde, hinta sekä myynnin ajankohta. Yrityksen toimitusjohtaja pystyy ohjauspaneelin kautta muuttamaan myytävien pumppujen malleja kulloinkin markkinoilla oleviin.

Lämpöpuppujen myynti noudattaa samaa kaavaa kuin asiakaslistan hallinta. Syötettyjen tekstikenttien tiedot poimitaan ylös ja päivitetään tietokantaan. Koska lämpöpumppujen asiakaskunta on hyvin vaihtuvaa, ei vanhojen asiakkaiden poimimiselle katsottu olevan tarvetta nykyisellään. Kuten kuviosta 14 näkee, on käyttöliittymä lämpöpumppujen myyntiin hyvin yksinkertainen. Painikkeesta ”Tee kauppasopimus” luodaan Word-dokumentti, johon poimitaan tiedot syötetyistä arvoista, sekä valittu pumppu. Kauppasopimukseen tulostetaan lisäksi yhteyslaskelma kuluista ja paikat allekirjoituksille.



Kuvio 14. Lämpöpumppujen myynnin käyttöliittymä

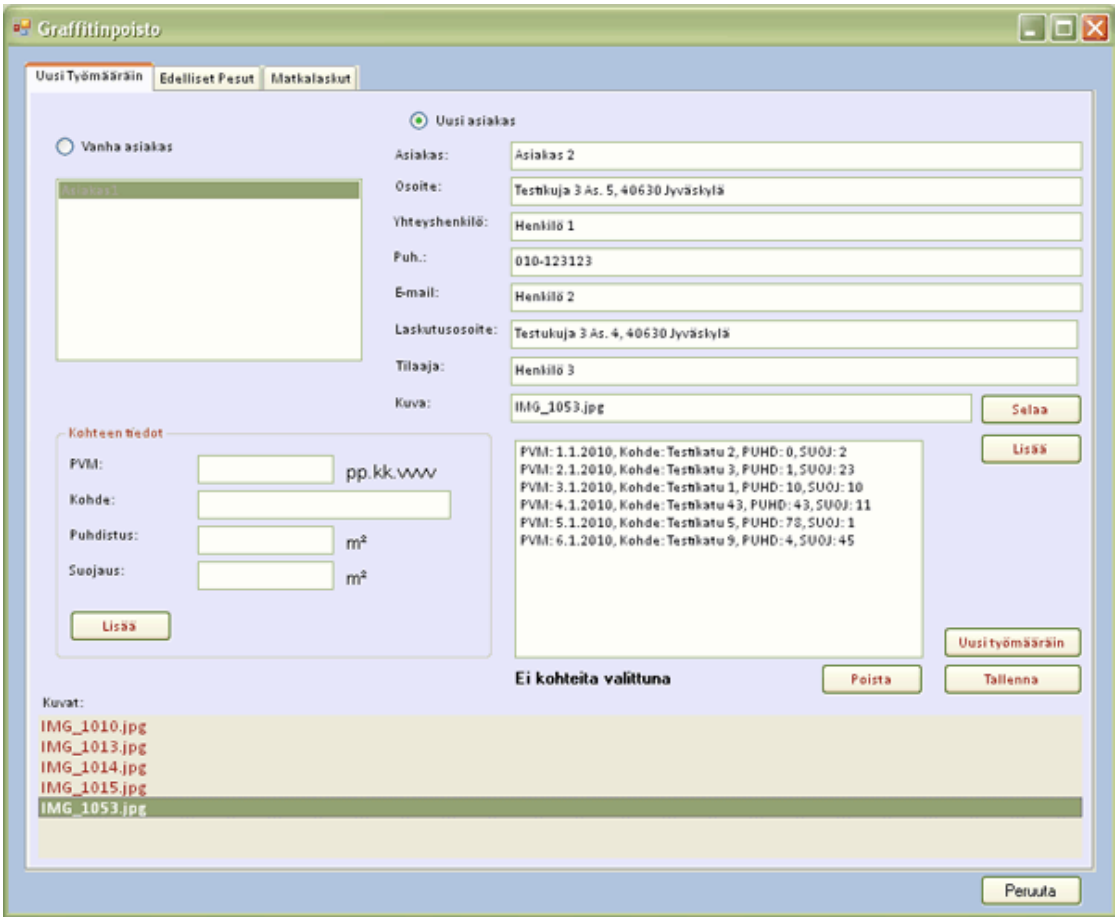
Kaikki myydyt pumput listataan toiselle välilehdelle DataGridView-kontrolliin. Listaus noudattaa perusnäkymää taulukkorakenteessa ja on järjesteltävissä kunkin kolumnin mukaan.

3.8.4 Graffitinpoisto

Yrityksen toimialoista graffitinpoisto tuottaa ehdottomasti eniten materiaalia, mm. kuvia. Kuvat tallennetaan palvelimelle, ja niiden URL osoite tallennetaan tietokantaan, jotta ne olisivat poimittavissa mistä vain. Vaikka Access 2007 tukee erinäisiä tiedostotyyppejä suoraan tietokannassa, päätettiin kuvien tallentaminen toteuttaa edellä mainitulla tavalla. Graffitinpoistossa asiakaskunta on suurimmaksi osaksi toistuvaa, eli tilaajina on yleensä samat asiakkaat kausiluonteisesti. Sovellukseen päätettiin toteuttaa asiakastietojen poiminta työskentelyn nopeuttamiseksi. Kuviossa 15 on kuvattu graffitinpoiston käyttöliittymä perustoimintoineen. Kuten kuviosta havai-

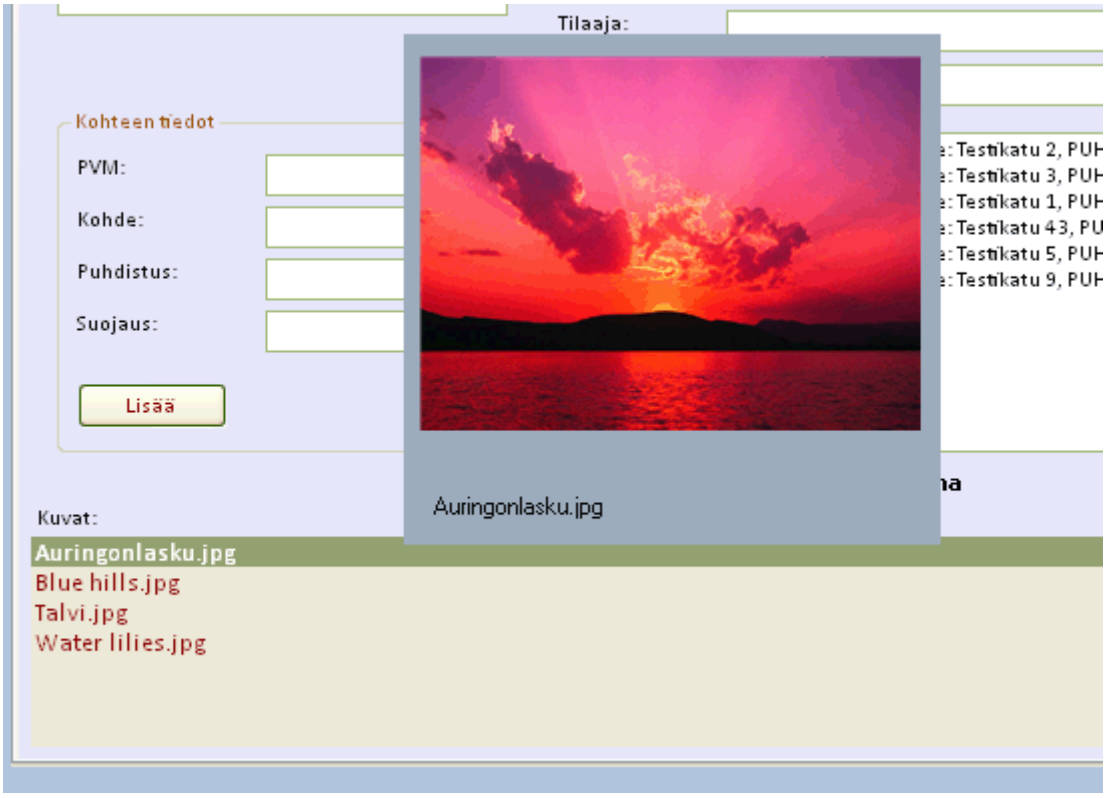
taan, käyttäjä voi valita joko vanhan tai uuden asiakkaan. Vanhat asiakkaat ovat poimittu asiakasrekisteristä ja listattu ListBox:in. Listassa on vain asiakkaat joiden toimiala on graffitinpoisto, karsien pois asiaankuulumattomat rivit ja samalla selkeyttäen käytettävyyttä. Jos käyttäjä valitsee uuden asiakkaan ja syöttää tarvittavat tiedot, tallennetaan uusi asiakas tietokantaan ja se ilmaantuu vanhoihin asiakkaihin tallennuksesta eteenpäin. Toisin sanoen asiakaslista päivittyy käytön myötä. Tallennetulle asiakkaalle määritellään automaattisesti toimialaksi graffitinpoisto. Muut tiedot poimitaan syötetyistä tekstikentistä.

työkohteen lisääminen tapahtuu täyttämällä vasemmassa alareunassa sijaitseva lomake ja painamalla lisää, jolloin tekstikenttien sisältämät tiedot tallennetaan kaksiulotteiseen taulukkoon ja tulostetaan viereiseen listaan. Näin voidaan lisäillä ja poistaa kohteita kunnes saadaan oikeat arvot listattua. Kun käyttäjä painaa lisää-painiketta, tyhjennetään tekstikentät ja aktiivinen valinta siirretään ylimpään tekstikenttään kohteen syöttökentissä.



Kuvio 15. Graffitinpoiston käyttöliittymä.

Kuvien lisääminen tapahtuu asiakastietojen alla sijaitsevasta kentästä. Käyttäjän klikatessa painiketta ”Selaa” avataan OpenFileDialog, jolla voidaan selata Windowsin tiedostorakennetta. OpenFileDialog palauttaa merkkijonona valitun tiedoston hakemistopolun, joka tallennetaan kuvalistaan painamalla ”Lisää”-painiketta. Tallennettavat kuvat listataan ikkunan alareunassa sijaitsevaan listaan. Kaksoisklikkaamalla kuvan nimeä, avataan kuva pienennettynä esikatseluun (ks. kuvio 16). Tämä mahdollistaa kuvien selaamisen ja voidaan varmistaa että oikeat kuvat ovat valittuina.



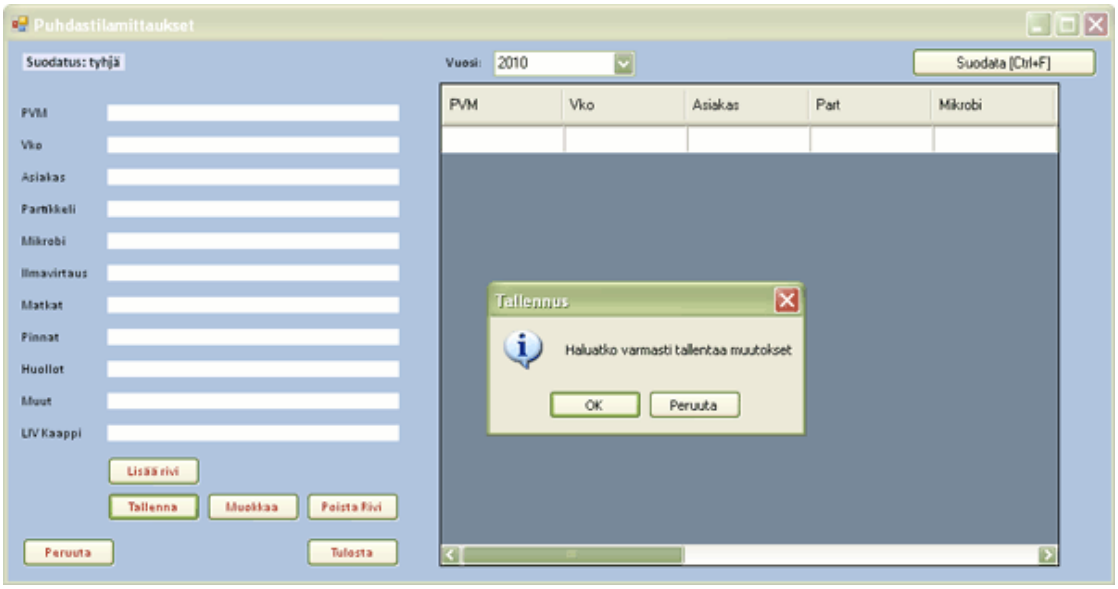
Kuvio 16. Graffitinpoiston kuvien esikatselu.

Kun kaikki tiedot ovat syötetty, käyttäjä valitsee joko uuden työmääräimen tulostamisen, tai tietojen tallentamisen. Työmääräintä tallentaessa asiakastiedot tallennetaan omaan tauluunsa, ja mittauksen tiedot omaansa.

3.8.5 Puhdastilamittaus

Puhdastilamittauksen taulu ”Mittaukset” on poimittu suoraan olemassa olevasta Excel-taulukosta Access 2007-tietokannan valmiilla käännösfunktiolla, sisältäen laajan sarjan sarakkeita.

Puhdastilamittaus muodostuu DataGridView-komponentista, joka esittää tietokannan sisältämän taulun ”Mittaukset” dataa. Tässä osiossa listauksen käsittelyyn on toteutettu perustyökalut, kuten rivien lisääminen, muokkaus ja poisto. Kuten muissakin osioissa, kysytään käyttäjältä varmistus tietojen tallentamiseen ennen varsinaista suoritusta (ks. kuvio 17). Tällä tavoin tietuetta voidaan muokata ilman vaaraa informaation tuhoamisesta. Rivit järjestellään valitun vuosiluvun mukaan, jossa kuluva vuosi on esiasetettu arvo.

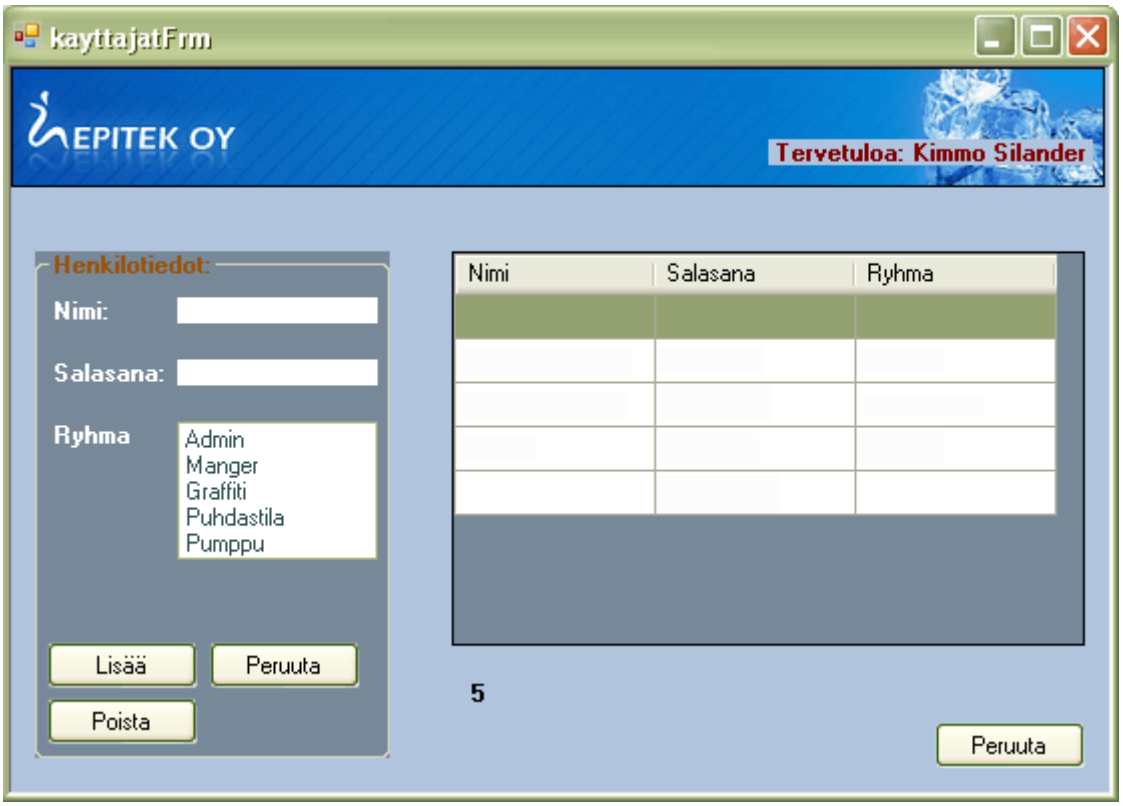


Kuvio 17. Puhdastilamittauksen käyttöliittymä.

Tulosta painikkeella tulostetaan koko lista asiakkaista ja heidän mittauksistaan valitulta vuodelta. Rivejä pystyy muokkaamaan valitsemalla haluttu rivi ja kaksoisklikkaamalla tai painamalla ”Muokkaa”-painiketta, jolloin rivin soluja voidaan vapaasti muokata.

3.8.6 Käyttäjienhallinta

Ohjauspaneelista voidaan hallita käyttäjätilejä. Sovellukseen voidaan luoda uusia käyttäjätilejä ja määrätä niille tietty käyttäjäryhmä. Tämä ryhmä määrittelee mihin sovelluksen osioihin käyttäjällä on oikeus kirjautua. Projektin päättyessä ainoastaan toimitusjohtajalla on oikeus luoda ja muokata tunnuksia. Koska käyttäjäryhmät ovat ennaltamäärättyjä on mahdolliset vaihtoehdot listattu listbox-objektiin (ks. Kuvio 18).



Kuvio 18. Henkilöstönhallinnan käyttöliittymä

Uutisten lisääminen pääsivulle tapahtuu myös ohjauspaneelin kautta. Käyttäjä luo uutiselle otsikon ja tekstin. Uutiset listataan ja no ovat näkyvillä etusivulla kunnes ne poistetaan. Uutiset tallennetaan Xml-muotoon ja ne luetaan päävalikon käynnistyessä. Uutiset toimivat muistutuksina muille käyttäjille ja ilmoituksina tärkeistä asioista. Kuten käyttäjätilien hallinnointi, myös uutisten hallinta kuului projektin loppuessa toimitusjohtajalle. Uutisten tallentamiseen valittiin XML-formaatti sen keveyden ja nopeuden johdosta. Henkilökunnan mukaan uutisia tulee noin 1 viikossa, joten ei nähty tarvetta luoda tietokantaan omaa taulua uutisille.

3.9 Toteutuneen järjestelmän testaus

Järjestelmän osa-alueita testattiin eri ryhmiin kuuluvien henkilöiden tunnuksilla. Näin varmistettiin, ettei sovelluksessa olisi osioita jotka toimisivat vain yhdellä henkilöllä. Järjestelmä toimi moitteettomasti käyttäjäroolien suhteen ja omiin ryhmiin kuuluvilla henkilöillä oli halutun kaltaisen käyttöoikeudet. Henkilökunnan puolesta esille nousi hyvissä merkeissä ohjelmiston käyttöliittymän yksinkertaisuus ja nopeus. Koska ohjelma suunniteltiin korvaamaan Excel-taulukot, niin tietovarastona kuin myös käytet-

tävyydeltään, oli toimintojen oltava nopeita ja selkeitä. Negatiivisena puolena todettiin Word-automaation hitaus luotaessa asiakirjoja. Siltikin Word-automaatio todettiin kannattavaksi tietojen poimintojen osalta ja huomattiin sen loppujen lopuksi nopeuttavan sekä selkeyttävän asiakirjojen luomista. Tietokanta toimi odotetulla tavalla, eikä suurempia hitauksia esiintynyt. Testauksessa palvelimen fyysinen tila osoittautui yhdeksi isoimmista ongelmista. Tästä johtuen päätettiin palvelimen kuvamateriaalit tallentaa ulkoisille lähteille säännöllisin väliajoin, vapauttaen huomattavan määrän fyysistä tilaa.

4 TYÖN JA TULOSTEN ARVIOINTI

4.1 Sovelluksen arviointi

Projektin alkaessa ensisijaisiksi vaatimuksiksi määriteltiin tiedon tallentamisen mahdollisuus yhteen loogiseen lähteeseen ja raporttien tuottamiseen erinäisistä tiedoista. Tutustuttaessa vanhaan järjestelmään, huomattiin näiden asioiden olevan suurimmat kehittämisen kohteet yrityksen nykyisessä tietojärjestelmässä.

Nykyinen järjestelmä tallentaa kaikki tiedot yhteen tietokantaan, korvaten lukuisat irralliset Excel-tiedostot. Lisäksi tiedot ovat käsiteltävissä mistä vain, jos on mahdollisuus verkkoyhteyteen. Voidaan näin ollen todeta sovelluksen yhtenäistävän tietojen ja materiaalin sijoittelua sekä parantavan niiden saatavuutta henkilökunnan kesken. Segmentoinnin vuoksi sovellus on suhteellisen vapaasti laajennettavissa uusille ominaisuuksille. Tästä johtuen sovellus on mahdollista pitää ajan tasalla yrityksen tarpeille, esimerkiksi uusien toimialojen lisääntyessä.

Tiettyjä ominaisuuksia ei haluttu toteuttaa sovellukseen projektin aikana johtuen niiden tarpeellisuuden kyseenalaisuudesta. Tästä johtuen sovellus toimii osittain päällekkäin vanhan systeemin kanssa, kunnes se laajennuksen jälkeen korvaa käytössä olevan järjestelmän kokonaan.

Toimintavarmuutta haluttiin lisätä toteuttamalla mahdollisuus varmuuskopioiden tallentamiseen. Varmuuskopiot ovat mahdollista tallentaa ulkoisille lähteille ja palauttaa systeemiin tarvittaessa. Sovellusta testattiin Jyväskylän Ammattikorkeakoulun student-palvelimella, eikä testeissä esiintynyt ongelmia tiedonsiirrossa. Muutaman kerran projektin aikana huomattiin hitautta kuvien siirrossa palvelimelle. Tämä saattoi johtua verkkoyhteyden nopeuden vaihtelusta, tai tietokoneen hetkellisestä hidastumisesta. Toisena ongelmana palvelimen puolelta havaittiin palvelimen tallennuskapasiteetti. Suuria määriä kuvia siirtäessä palvelimen levytila loppuu nopeasti. Kyseiselle palvelimelle sopii suurin piirtein 2 viikon kuvamateriaali. Projektin päätty-

essä tilanne ratkaistiin tallentamalla kuvat säännöllisin välein ulkoisille lähteille, kuten DVD-levylle, tai ulkoiselle kovalevylle. Ratkaisuksi suunniteltiin oman palvelimen ostoa yrityksen käyttöön, johon laajennettaisiin tarvittava määrä levykapasiteettia. Toisena vaihtoehtona olisi vaihtaa yrityksen käyttämä kaupallinen palvelin isomman levykapasiteetin omaavaan versioon.

Projektin ongelmina esiin nousivat mm. eri käyttäjien haltuunsa saamat tiedot, tietoturvallisuus, palvelimien suorituskyky tiedonsiirrossa ja toimintavarmuus. Esimerkiksi henkilökunnan piti pystyä suoriutumaan laskutuksesta ja tietojen jäsentämisestä sitä varten, muttei saada kuitenkaan kaikkea yrityksen tietoja käsiinsä. Tätä varten suunniteltiin käyttäjäprofiilit jokaiselle toimialalle, jotka ovat muokattavissa tarpeen mukaan. Palvelimien suorituskykyä testattiin projektin edetessä ja sen huomattiin alustavasti riittävän nopeudeltaan. Lähinnä tallennuskapasiteetti vaatii ratkaisun laajennuksen yhteydessä. Suurimmaksi huolenaiheeksi tulevaisuudessa muodostuivatkin palvelimen fyysinen tallennuskapasiteetti ja sen tuomat rajoitukset.

Sovelluksen todettiin täyttävän aloitusprojektissa määritellyt pääasialliset tarpeet ja muutamia ylimääräisiä ominaisuuksia. Joitain pienempiä työkaluja jäi toteuttamatta projektin aikana, mutta ohjelmistoa tullaan kehittämään jatkuvasti yrityksen muuttuviin tarpeisiin. Seuraavia toteutettavia osioita ovat mm. laajempi raportointityökalu, kuvagalleria ja laskutusosio.

4.2 Opinnäytetyöprosessin arviointi

Opinnäytetyöprosessi kehitti projektiryhmän ammatillista taitoa monipuolisesti. Lähes kaikki tekniikat joita projektissa käytettiin, olivat ryhmälle vieraita tai hyvin tuntemattomia ja vaativat syvempää perehtymistä. Isoimpana täysin vieraana osiona mainittakoon Word-automaatio, jossa riitti todella paljon tutkittavaa ja opittavaa. Projektiryhmän palaverit opettivat puolestaan asiakaspinnan monipuolisuuden ja mm. määrittelyn haasteellisuuden. Projektin tiedonhaku oli ennakoitua isommassa roolissa. Noin puolet suunnittelun ajankäytöstä kului tiedonhankintaan parhaista suoritustavoista ja tekniikoita.

Opinnäytetyö suoritettiin kokonaisuudessaan alusta loppuun projektina, kattaen määrittelyn, suunnittelun, toteutuksen, testauksen ja osittaisen käyttöönoton. Projektin aikana huomattiin monia yksittäisiä asioita projekti- ja ryhmätyöskentelystä, jotka olivat ennalta-arvaamattomia. Näitä asioita olivat muun muassa elävät aikataulut sekä oman aseman tärkeys ammatillisissa suorituksissa. Vaikka projektissa hyödynnettiin tietoa ulkoisista lähteistä, huomattiin Jyväskylän Ammattikorkeakoulun tarjoaman koulutuksen toimivan hyvin kattavana pohjana niin projektityöskentelyyn kuin itse sovelluskehitykseen.

LÄHTEET

All interview. 2010. What is JIT? Viitattu 17.5.2010. http://www.allinterview.com/showanswers/20085.html

C-Sharp-Station. 2010. C#. Viitattu 15.5.2010. http://www.csharp-station.com/

C-sharp-online. 2010. C-Sharp Overview. Viitattu 22.4.2010. http://en.csharp-online.net/CSharp_Overview

DOT.NET-guide. 2010. JIT information. Viitattu 17.5.2010. http://www.dotnet-guide.com/jit.html

Microsoft Access. 2010. Homepage. Viitattu 9.5.2010. http://office.microsoft.com/fi-fi/access/default.aspx

Microsoft DOT.NET. 2010. Overview. Viitattu 22.4.2010. http://microsoft.com/net/basics/whatis.asp

MSDN Library. 2010. Developer Library. Viitattu 2.5.2010. http://msdn.microsoft.com/en-us/library/

Price, J. 2003. C# Database Programming. Sybex

Troelsen, A. 2005. Pro C# 2005 and the .NET 2.0 Platform. Apress.

VB.NET-informations. 2010. Microsoft .NET tutorials . http://vb.net-informations.com/

Wille, C. 2001. C#. Tummavuori: Docendo Toolkit