VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Jukka Johannes Hahtokari

# UPCODE-MAKSUJÄRJESTELMÄ

Tekniikka ja liikenne

2010

VAASAN AMMATTIKORKEAKOULU

Tietotekniikan koulutusohjelma

# TIIVISTELMÄ

| | |
|---|---|
| Tekijä | Jukka Hahtokari |
| Opinnäytetyön nimi | UpCode Maksujärjestelmä |
| Vuosi | 2010 |
| Kieli | suomi |
| Sivumäärä | 35 + 3 liitettä |
| Ohjaaja | Ghodrat Moghadampour |

Opinnäytetyö on tehty vaasalaiselle yritykselle nimeltä UpCode Ltd. Järjestelmän tarkoitus on huolehtia yrityksen laskutuksesta, asiakasrekisteristä ja tuoterekisteristä sekä tuottaa laskuja jossa on matkapuhelinmaksamista helpottava pikamaksukoodi.

Pikamaksukoodista asiakas voi maksaa laskunsa suoraan koodia skannaamalla käyttäen matkapuhelinta tai web-kameraa. Koodin kautta maksamisessa etuna perinteiseen e-maksamiseen verrattuna on se, että laskusta ei tarvitse käsin kopioida mitään tietoja, vaan koodin kautta tiedot tulevat automaattisesti. Tämä tekee maksamisesta nopeaa ja se myös eliminoi tietojen täytön aikana tapahtuvat kirjoitusvirheet.

Laskutusohjelma on tehty .NET ympäristössä C# ohjelmointikieltä käyttäen. Valmis ohjelma on .EXE muotoinen Windows sovellus. Ohjelman ajaminen vaatii .NET Framework ympäristön sekä Windows käyttörärjestelmän.

Pikamaksukoodin kännykkäkäyttöliittymä on tehty ASP-kielellä, koska kaikki kännykkämallit ymmärtävät sitä. Pikamaksukoodi ja laskutusohjelma kommuikoivat keskenään yhteisen laskutusrekisterin kautta.

Asiasanat laskutus, 2D-datamatriisikoodi, C#,.NET

VAASAN AMMATTIKORKEAKOULU

UNIVERSITY OF APPLIED SCIENCES

Tietotekniikan koulutusohjelma

# ABSTRACT

| | |
|---|---|
| Author | Jukka Hahtokari |
| Title | UpCode Billing System |
| Year | 2010 |
| Language | Finnish |
| Pages | 35 + 3 Appendices |
| Name of Supervisor | Ghodrat Moghadampour |

This thesis was made for a company named UpCode Ltd. The purpose was to create a system to take care of invoicing functions in companies and produces invoices with an instant payment code which helps mobile payment.

From this instant payment code the client can pay an invoice by pointing the code with a mobile phone or with a normal web cam. The advantage compared to traditional e-payment is that the person, who pays the invoice, does not have to fill any information manually. This makes the paying process quick and eliminates the change of typing errors.

The billing program was made in .NET environment with the C# programming language. The program comes in .EXE format and is a windows application. The application requires .NET Framework and the Windows operating system.

The user interface to the instant payment code was done with the ASP programming language due to its ability to be understood with any mobile phone. The instant payment code and billing program communicates with each other via common database.

Keywords Billing, 2D-datamatrix code, C#, .NET

# ALKUSANAT

Opinnäytetyö on tehty vaasalaiselle yritykselle nimeltä UpCode Ltd. Työn valvojana on toiminut tohtori Ghodrat Moghadampour ja yrityksen yhdyshenkilönä insinööri Kristoffer Jansson.

Haluan kiittää Ghodrat Moghadampouria, Kristoffer Janssonia, sekä kaikkia muita, jotka ovat edesauttaneet opinnäytetyöni valmistumista.

16.5.2010 Vaasa

Jukka Hahtokari

# KÄYTETYT MERKINNÄT JA LYHENTEET

| | |
|---|---|
| 2D | *2-dimensional,* kaksiulotteinen |
| CLS | *Common Language Specification,* määrittää yhteisen ohjelmointikielirakenten |
| BCL | *Base Class Library,* yhteinen perusluokkakirjasto |
| CLR | *Common Language Rintime,* määrittää yhteisen ajonaikaisen ympäristön |
| JVM | *Java Virtual Machine,* Virtuaalinen ajoympäristö |
| Cool | Aiempi nimitys C# ohjelmointikielelle |
| C# | *C sharp,* ohjelmointikieli |
| J++ | *J Plus Plus,* ohjelmointikieli |
| QFD | *Quality function deployment,* ohjelmistosuunnittelun työkalu |

**SISÄLLYS**

# 1 JOHDANTO

Laskujen maksaminen on kaikille tuttua. Ennen verkkopankkia laskut käytiin maksamassa pankissa ja niiden läheisyydessä olevissa maksuautomaateissa. Nykyään Internet on mullistanut perinteisen laskujen maksamisen ja suurin osa ihmisistä asioi nettipankissa, jolloin maksaminen käy omalta kotikoneelta. Tämä on ollut aikoinaan suuri edistysaskel.

Maksaminen nettipankissa tapahtuu syöttämällä laskussa olevat tiedot oikeisiin kenttiin ja hyväksymällä maksusuoritus. Tämä tehdään suurimmalta osin vielä manuaalisesti näppäilemällä.

Nykyään suurimmassa osassa matkapuhelimia on Internetyhteys, joten teknisiä esteitä matkapuhelimella maksamiseen ei ole olemassa. Kuitenkin tämäntyyppinen maksaminen on vielä erittäin vähäistä. Yhtenä syynä siihen on se, että kännykällä kirjoittaminen on vaivalloista ja hidasta. Myös kirjoitusvirheiden mahdollisuus on suurempi kuin tietokoneen näppäimistöltä kirjoitettaessa, vaikka sekään tapa ei ole täysin virheetön.

Matkapuhelimella maksaminen voi olla kuitenkin seuraava edistysaskel Internet-pohjaisessa maksamisessa. Kun nettipankit aikoinaan lyhensivät huomattavasti pankkien jonoja poistamalla säännölliset vierailut pankkiin, matkapuhelimella maksaminen poistaa tarpeen olla ennalta määrätyssä paikassa, eli tietokoneen ääressä.

Päättötyössä on tehty maksujärjestelmän UpCode Ltd. nimiselle yritykselle. Järjestelmässä on otettu huomioon laskujen maksaminen matkapuhelimella ja siitä on pyritty tekemään mahdollisimman helppoa 2D-datamatriisikoodin avulla, joka toimii paperimuotoisessa laskussa pikamaksulinkkinä.

# 2 UPCODE LTD.

UpCode Ltd. on vaasalainen yritys, joka työllistää yli sata ihmistä maailmanlaajuisesti. UpCode on UPC konsultoinnin tytäryhtiö. UpCode on aloittanut alun perin toimintansa vuonna 2003. Silloin useiden muiden yritysten kanssa perustettu projekti nimeltään Print Access ei kuitenkaan tuottanut riittävää tulosta ja vuonna 2004 Sture Udd, UpCoden ja UPC konsultoinnin toimitusjohtaja, ryhtyi ajamaan Print Accessia yksin eteenpäin. Print Access-projekti vaihtoi nimeä ja siitä tuli UpCode. Samalla perustettiin UpCode Ltd:nä tunnettu tytäryhtiö. /3/

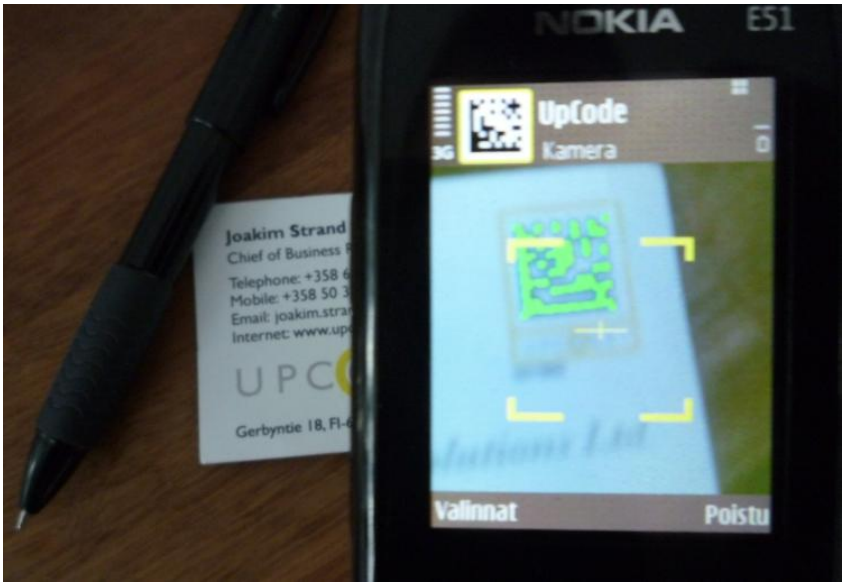
UpCoden Vaasan toimitilat sisältävät myynnin, hallinnon, tuotekehityksen, koulutuskeskuksen sekä esittelytilat. Koulutuskeskuksessa käy UpCoden edustajia joka puolelta maailmaa oppimassa lisää optisesta luvusta ja sen tuomista mahdollisuuksista kaikilla eri toimialasektoreilla. UpCoden esittelytiloissa käy päivittäin ihmisiä niin valtakunnallisen yritysmaailman kuin kansainvälisenkin kentän huipulta. Yhteistyökumpaneita ovat mm. Nokia, Stora Enso, UPM ja Wärtsilä. /3/

Yrityksen perusidea on kaksiulotteisten koodien lukeminen matkapuhelimella. Tuotevalikoimassa on tuotteita yksityisille kuluttajalle ja yrityksille luotuja ratkaisuja. Tyypillinen kuluttajaratkaisu on lehteen painettu 2D-koodi, jonka kautta voi esimerkiksi saada lisää ajan tasalla olevaa informaatiota tai ostaa lippuja erilaisiin tilaisuuksiin. Yritysratkaisu voi olla esim. työntekijöiden, tavaravirtojen tai kiinteistöjen valvontaan kehitetty valvontatyökalu, jossa työntekijä kuittaa tehdyn työn matkapuhelimellaan ja tuottaa merkinnän työnvalvojalla olevaan hallintatyökaluun. Näin ollen voidaan mm. tehokkaammin koordinoida kentällä olevien työntekijöiden rajallisia resursseja.

Suurin osa UpCoden tuotteista sisältää vähintään yhden 2D-koodin. Itse koodi ei kuitenkaan koskaan ole pääasiallinen tuote, vaan se edustaa enemmänkin helppoa tapaa päästä sisään mobiililaitteelle suunniteltuun taustajärjestelmään. Toisin sanoen UpCode Ltd. ei myy 2D-koodeja yrityksille ja yhteisöille, vaan se myy järjestelmiä, jotka on suunniteltu aina räätälöidysti ottaen huomioon asiakkaan toiveet ja helppokäyttöisyyden. Tiettyjä poikkeuksia on kuitenkin olemassa. Esimerkiksi valvontaratkaisut, joissa puhelimeen asennettava UpCode lukija lähettää 2D-

koodin luvun yhteydessä taustajärjestelmälle tietoja käyttäjän puhelimesta, jolla käyttäjä voidaan identifioida ja pääsy järjestelmään joko sallia tai evätä.

Vaikka pääpaino onkin taustajärjestelmissä, on 2D-koodin ja kännykkään asennettavan lukijan toiminta kuitenkin hyvä sisäistää ymmärtääkseen yrityksen eri toimintamallit.



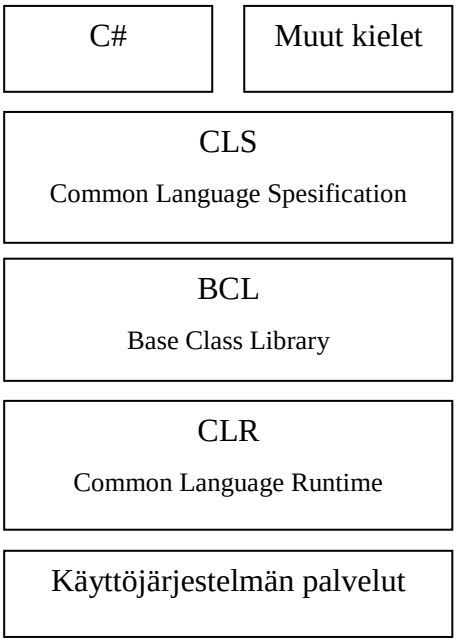

**Kuva 1. UpCode datamatriisikoodi on henkilön käyntikortissa. Puhelimessa on UpCode-Reader, joka vihreällä värillä indikoi, että käyntikortissa oleva koodi on onnistuneesti luettu.**

Kuvassa 1, UpCode-lukijaa käyttävä henkilö saa käyntikortissa olevan työntekijän yksityiskohtaiset yhteystiedot suoraan matkapuhelimeensa. Tämän jälkeen käyttäjä voi tallentaa sähköisen käyntikortin puhelimen muistiin, jolloin hän voi halutessaan palata katsomaan työntekijän tietoja. Tässä on siis kuvailtu yksi mahdollinen tilanne, jossa on käytetty UpCode teknologiaa helpottamaan jokapäiväisiä toimia. Lähtökohtana edellä mainitun esimerkin kehittämiseen on ollut käyntikorttien korkea todennäköisyys mennä hukkaan varsinkin myyntihenkilöillä, jotka voivat saada useita käyntikortteja työpäivän aikana. Suoraan kännykkään tallennetut tiedot voivat siis olla erittäin tärkeitä.

# 3 .NET FRAMEWORK OHJELMOINTIYMPÄRISTÖNÄ

Microsoftin kehittämä ohjelmistokomponenttikirjasto .NET Framework tarjoaa ohjelmistokehittäjälle erittäin hyvät puitteet kehittää ohjelmia erilaisiin tarpeisiin. Ensimmäinen versio, .NET 1.0 julkaistiin vuonna 2002. Tällä hetkellä viimeisin vakaa versio on .NET 3.5. Nykyisestä kirjastosta löytyy työkalut yksinkertaisista merkkijonon tulostuksista aina puheentunnistusfunktioihin. Tämä kertoo siitä, että kehitystä tapahtuu jakuvasti ja 4.0 versio on jo saatavana beta-versiona.

.NET Framework on ohjelmistokomponenttikirjasto, jota voi käyttää monella eri ohjelmointikielellä. .NET kielen on kuitenkin noudatettava tiettyjä perussääntöjä, kuten CLS:ää, joka määrittelee muuttujatyyppirakenteen. Myös BCL, eli perus-luokkakirjasto täytyy olla kaikille kielille sama. CLR yhdessä käyttöjärjestelmän palvelujen kanssa puolestaan huolehtii ajonaikaisesta ympäristöstä. /1/




**Kuva 2. Net Frameworkin perusrakenne. /1/**

### 3.1 Olio-ohjelmoinikieli, Java

Java ohjelmointikieli oli ensimmäinen hyvin menestynyt olio-pohjainen lähestymistapa ohjelmointiin. Ohjelmointikielen ensimmäinen versio julkaistiin 1990-luvun alussa. Pääasiallinen ero aikaisempaan on kyky jäsentää koodi eri luokkiin, olioihin, joita pystyy tarpeen mukaan luomaan pääluokasta käsin. Yksinkertainen olio voi olla esim. auto-olio, jolle luonnin yhteydessä annetaan parametrinä rekisteritunnus. auto-olioita voi olla pääluokassa lukematon määrä. Nämä auto-oliot eroavat toisistaan luonnin yhteydessä annettavalla olion nimellä, esim. Renault ja BMW, joilla kummallakin on oma rekisterinumeronsa. /1/

Kyky jäsentää koodia tällä tavalla helpottaa kehittäjien työtä tekemällä lähdekoodista helpommin luettavaa. Jäsentäminen eri kokonaisuuksiin helpottaa myös ison kehitystiimin yhteistyötä, koska jokainen saa selvästi oman työmaansa. Isoja kokonaisuuksia sisältävän ohjelman tekemisessä lähdekoodin selkeys nousee erittän suureen arvoon.

### 3.2 C# ohjelmointikieli

Anders Helsberg, kuuluisa ohjelmointikieliarkkitehti palkattiin vuonna 1996 Borlandilta Microsoftille kehittämään Microsoft Java-virtuaalikonetta (JVM) ja Visual Studion J++ ohjelmointikieltä. Sun Microsystems, Java-ohjelmointikielen kehittäjä ja Microsoft kuitenkin ajautuivat pian riitaan siitä kuinka paljon Microsoft voi laajentaa ja muunnella JVM:ää. Suurimpana kitkan aiheuttajana oli J++:aan kaavailtu mahdollisuus kutsua suoraan käyttöjärjestelmän palveluja. /1/

Samaan aikaan Microsoft kehitteli ns. ”täydellistä” ohjelmointikieltä projektinimellä Cool. Umpikuja Sunin kanssa muutti Microsoftin suunnitelmia, jolloin Microsoft yhteistyön sijaan päättivätkin kehittää täysin oman Java-kielensä. Coolista tuli C#. /1/

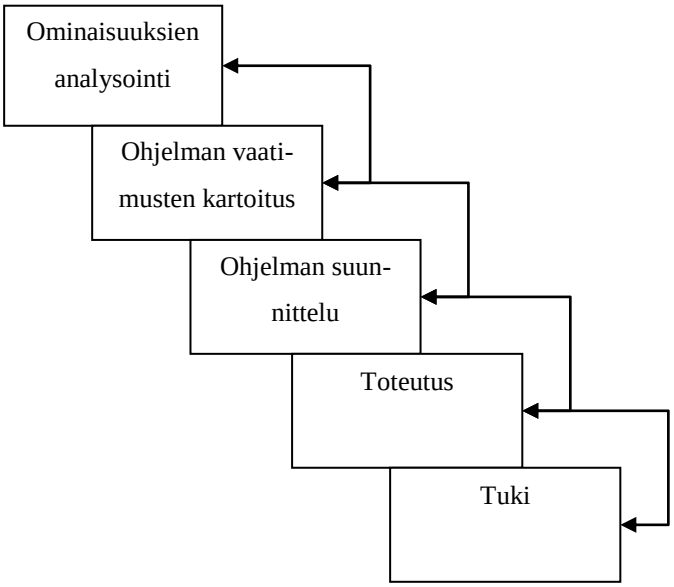
j++:n kaavailtu suora yhteys käyttöjärjestelmän palveluihin on lähes sellaisenaan C# ohjelmointikielessä. /1/

### 3.3 Visual Studio express edition

Microsoft Visul Studio on .NET ohjelmoijalle kehitetty ohjelmointityökalu, joka on .NET ohjelmoijalle käytännössä välttämättömyys. Visual Studio Expressin saa ladattua Microsoftin sivuilta täysin ilmaiseksi sekä yksityiseen että kaupalliseen käyttöön. Myös kääntäjä CSC.EXE tulee .NET Frameworkin mukana täysin ilmaiseksi.

# 4 OHJELMAN SUUNNITTELU

Ohjelmistosuunnittelussa käytetään erilaisia malleja joilla kehitystyö pyritään jäsentämään eri kokonaisuuksiin. Vesiputousmalli on pitkään käytössä ollut menetelmä, jota tässäkin työssä on käytetty. Kaksisuuntaiset nuolet tarkoittavat sitä, että aiempaan vaiheeseen voi tarvittaessa palata jos jotain uutta ja oleellista ilmenee matkan varrella. Kuvasssa 3 on vesiputousmalli havainnollistettuna.




**Kuva 3. Tyypillinen vesiputousmalli./2/**

## 4.1 Ominaisuuksien analysointi.

Tietokoneohjelmalla pyritään aina ratkaisemaan ongelma. Onnistunut ohjelma palvelee käyttäjää siirtämällä tietokoneelle osan työtaakasta. Ensimmäisenä vaiheena onkin ongelman analysointi, joka kartoittaa syitä miksi ohjelmaa aletaan ylipäätänsä kehittämään. Tämän kokonaisuuden hahmottaminen on erittäin tärkeää, jottei punainen lanka katoa missään vaiheessa ohjelman kehityskaaren aikana. /2/

### 4.2 Ohjelman vaatimusten kartoitus

Seuraavaksi kartoitetaan ohjelmiston asettamat vaatimukset eli minkälaisia työkaluja toteutukseen tarvitaan. Tämä on erittäin tärkeä vaihe esimerkiksi siinä mielessä, että joskus ohjelma voi vaatia jotain maksullisia lisäosia, tai lisenssejä, jotta valmista ohjelmaa voidaan myydä eteenpäin. Kun tämä kartoitus on tehty, tavoitteena on, että projektissa ei ole enää mitään ns. ”harmaalla aluella”, eli toisin sanoen mikään ei ole enää epäselvää suunnitteluvaiheeseen mentäessä. /2/

### 4.3 Ohjelman suunnittelu

Suunnitteluvaiheessa ohjelma jaetaan toiminnallisesti eri kokonaisuuksiin. Luodaan ohjelman runko. Eri kokonaisuuksista tehdään omat luokat ja luokkiin hahmotellaan tarvittavat funktiot. Tämän lisäksi kartoitetaan rajapinnat tietokantaan ja suunnitellaan käyttöliittymä tietokoneen ja ihmisen välillä. Suunnittelun jälkeen tavoitteena on, että seuraava vaihe eli toteutus on mahdollisimman suoraviivaista. /2/

### 4.4 Toteutus

Suunnitelma sovelluksesta täytyy muuntaa konekieliseen muotoon. Vasta tässä vaiheessa ohjelman runko, sen luokat ja funktiot saavat tarvittavan sisällön. Tietokone ei ymmärrä ohjelmoijan kirjoittamaa koodia suoraan, mutta kääntäjän jälkeen ohjelma on käännetty tietokonekielelle. /2/

### 4.5 Testaus

Testausvaiheen tärkeyttä ei voi korostaa liikaa. Vaikka ohjelman toteutus jaetaankin pienempiin osiin ja suunnitellaan hyvin, yksittäisen ohjelmoijan vastuulle tuleva kokonaisuus voi olla suuri. Tämän takia koodiin voi jäädä virheitä, jotka vaikuttavat ohjelman logiikkaan. Parhaimmassa tapauksessa virheet tulevat esiin testauksen aikana, mutta valitettavasti virheitä voi myös jäädä valmiiseen tuotteeseen. Testauksella pyritäänkin eliminoimaan mahdollisimman tehokkaasti ohjelmointivirheet, jotka voivat olla ikäviä ja noloja asioita ohjelmiston valmistajalle. /2/

### 4.6 Tuki

Viimeisenä vaiheena on ohjelmistotuki. Tuen avulla ohjelmaa voidaan kehittää edelleen, vaikka tuote on jo ehtinyt asiakkaalle asti. Monesti tässä vaiheessa vielä ilmenee seikkoja, joita ei ole ohjelman kehityskaaren aikana otettu huomioon. Tyypillisesti tällainen seikka koskee käyttäjäystävällisyyttä, mutta myös testausvaiheen seulan läpi päässeet logiikkavirheet tulevat ilmi ennemmin tai myöhemmin, jolloin uusissa versioissa ongelmat on korjattu ja otettu huomioon. /2/

# 5 UPCODE-MASKUJÄRJESTELMÄ

UpCode-maksujärjestelmä koostuu windows-ympäristöön luodusta laskutusohjelmasta, jolla voi luoda laskurekisterin, asiakasrekisterin sekä tuoterekisterin. Toinen tärkeä osa maksujärjestelmää on pikamaksukoodi, joka mahdollistaa laskun maksamisen matkapuhelimen avulla.

## 5.1 Maksujärjestelmän lähtökohta

Järjestelmän kehittämisen lähtökohtana on ollut paperimuodossa maksettavat laskut. Vaikka laskun voi jo nykyään maksaa sähköisesti ilman, että paperimuotoista laskua edes postitetaan, suurin osa kuitenkin haluaa maksaa laskunsa vielä perinteisellä tavalla. Tyypillinen maksutapahtuma tapahtuu nykyään internetpankissa. Maksaminen on kuitenkin vielä hyvin manuaalista toimintaa. Käyttäjä joutuu siirtämään käsin numerot paperilta tietokoneen ruudulle jonka jälkeen seuraa vielä numeroiden tarkistaminen. Sen lisäksi, että tapahtuma on työläs, myös virheitä sattuu. Viitenumero voi esimerkiksi mennä väärin, jolloin yrityksen laskutusosasto joutuu manuaalisesti päättelemään maksajan nimestä ja summasta sen, mistä maksutapahtumasta on kysymys.

UpCode maksujärjestelmällä pyritään tekemään helpotuksia maksamisprosessiin. Pikamaksukoodia hyväksikäyttäen numerot ja nimet, jotka lähetetään pankin järjestelmään, ovat aina täysin virheettömiä. Tällä tavoin ohjelmalla voidaan vähentää myös yritysten laskutushenkilökunnan ylimääräistä työtaakkaa. Tämän lisäksi laskun maksaja on täysin paikasta riippumaton, koska maksu suoritetaan langatonta Internet-verkkoa hyväksikäyttäen.

## 5.2 Järjestelmän määrittely

Määrittelyn perustana on ollut systemaattinen ohjelman vaatimusten kerääminen. Tässä on käytetty apuna QFD-menetelmää. QFD määrittelyn tarkoituksena on maksimoida toimeksiantajan tyytyväisyys keräämällä vaatimukset kolmeen eri kategoriaan. Ensimmäinen kategoria on selkeät toimeksiantajan vaatimukset, eli

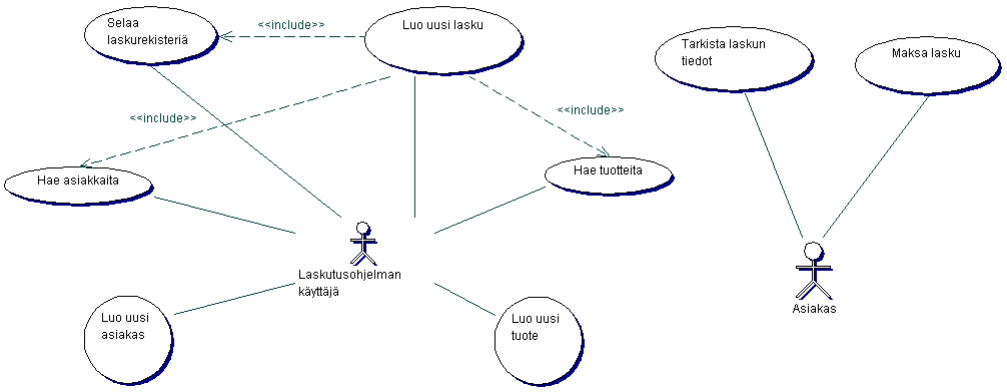
ne asiat, jotka asiakas haluaa ohjelman tekevän. Toinen, yhtä tärkeä kategoria on mainitsemattomat vaatimukset, joita toimeksiantaja ei osaa vaatia, mutta mitä järjestelmän toimivuus ehdottomasti edellyttää. Kolmas kategoria on ohjelman toimittajan näkemykset tärkeistä asioista. Nämä ominaisuudet voivat antaa ohjelmalle lisäarvoa tuottamalla asiakkaalle positiivisia yllätyksiä.

Selkeät toimeksiantajan vaatimukset ohjelmalle ovat mahdollisuus luoda maksutapahtumia laskutusrekisteriin. Uusia asiakkaita pitää voida tallentaa asiakasrekisteriin ja tuotteet on saatava tuoterekisteriin. Tämän lisäksi ohjelman on voitava luoda laskuja, joihin tulee pikamaksukoodi. Tämä mahdollistaa laskun maksamisen matkapuhelimella. Toimeksiantajan vaatimusiin kuului myös kyky tarkistaa ohjelmallisesti maksetut laskut pankin järjestelmästä. Tämä ominaisuus tullaan kuitenkin toteuttamaan vasta myöhemmässä vaiheessa, joten siihen ei kiinnitetty huomiota vielä tässä vaiheessa.

Mainitsemattomiin vaatimuksiin kuului mahdollisuus siihen, että ohjelman eri osat on helposti muunneltavissa ja että niitä voidaan päivittää mahdollisimman vaivattomasti. Myös osien integrointi valmiina oleviin laskutusohjelmiin oli tärkeää pitää mahdollisena. Mainitsemattomiin vaatimuksiin kuului myös Internet-pohjaisen käyttöliittymän tekeminen matkapuhelimille, jonka kautta on mahdollista saada kyseisessä laskussa olevat tiedot laskutusrekisteristä. Tietojenkeruun lisäksi käyttöliittymän on kyettävä ottamaan yhteys pankkien mobiilimaskujärestelmiin ja välittää tiedot niihin käyttäjän kannalta automaattisesti.

Näiden ominaisuuksien lisäksi sekä laskutusohjelman että pikamaksukoodin helppokäyttöisyys on ollut yksi tärkeä ominaisuus, jota on pyritty noudattamaan.

Ohjelman määrittelyssä on käytetty apuna myös käyttötapakaaviota. Järjestelmässä on kaksi käyttäjää: asiakas ja laskutusohjelman käyttäjä. Asiakas voi pikamaksukoodin skannaamisen jälkeen tarkistaa laskun tiedot ja maksaa maksun. Laskutusohjelman käyttäjällä on mahdollisuus luoda uusia laskuja, asiakkaita ja tuotteita. Tämän lisäksi käyttäjä voi selata syntyneitä lasku-, asiakas-, ja tuoterekisteristä. Kuvassa 4 on maksujärjestelmän käyttötapausmalli.

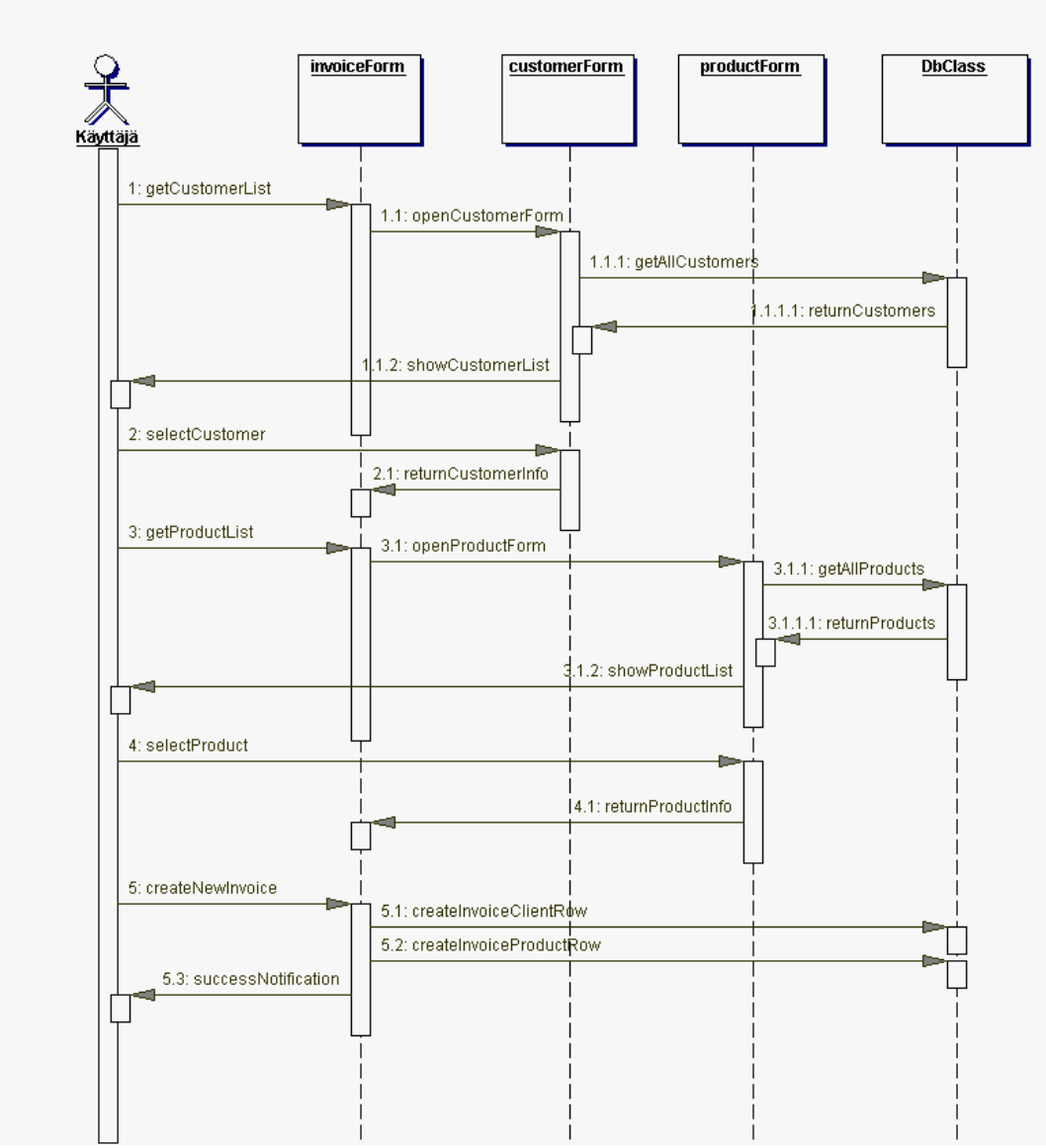



**Kuva 4. UpCode-maksujärjestelmän käyttötapausmalli.**

## 5.3 Järjestelmän suunnittelu

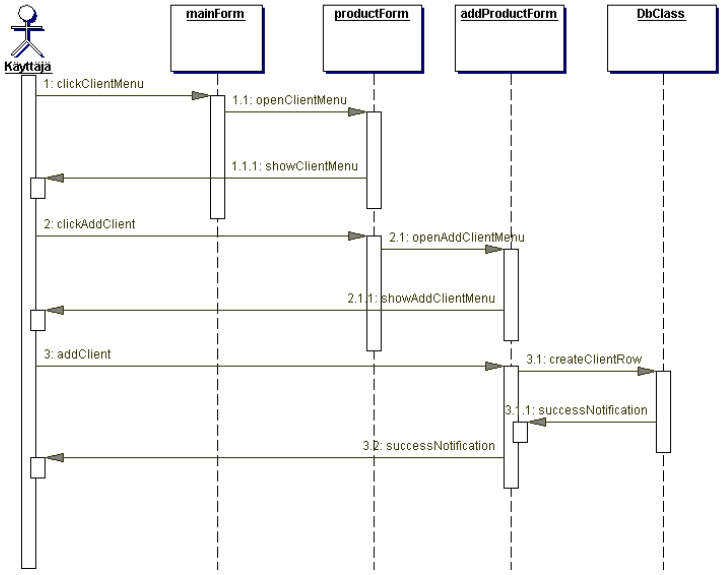
Suunnittelussa on lähdetty liikkeelle siitä periaatteesta, että se tullaan jakamaan mahdollisimman tehokkaasti eri kokonaisuuksiin. Tavoitteena on ollut tehdä mahdollisimman yksinkertaisia ja selkeitä luokkia, jonka ansiosta muutosten tekeminen ja mahdollisten uusien ominaisuuksien lisääminen tai osien integroiminen muihin ohjelmiin on mahdollisimman helppoa.

Sekvenssikaavio havainnollistaa sitä, kunka prosessi lähtee liikkeelle eri ohjelmistokomponenteissa käyttäjän toimesta. Kuva 5 havainnollistaa laskutusohjelman pääprosessi eli laskun luominen laskutusrekisteriin sekvenssikaaviolla kuvattuna.

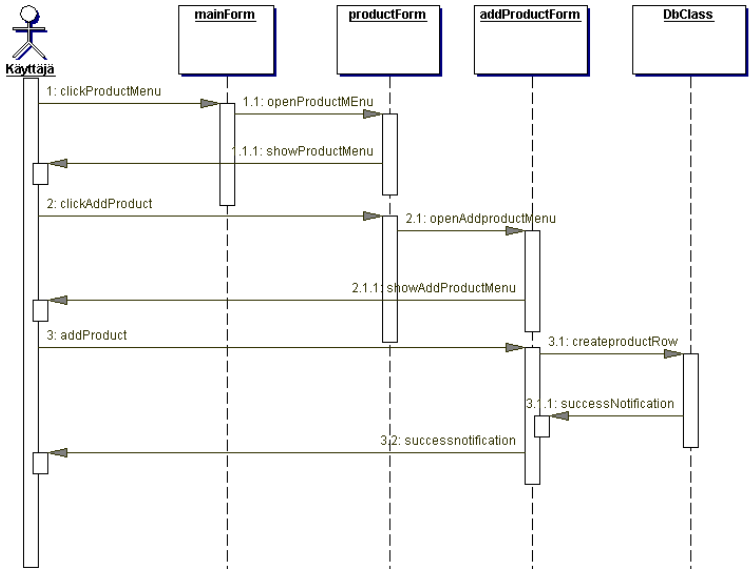



**Kuva 5. Laskun luominen sekvenssikaaviolla kuvattuna.**

Kaksi pienempää, mutta kuitenkin tärkeää funktiota laskutusohjelmassa ovat tuotteiden ja asiakkaiden lisääminen järjestelmään. Kummatkin funktiot ovat toimintaperiaatteiltaan samanlaisia. Käyttäjä avaa päämenun kautta tuote tai asiakasmenun. Tämän jälkeen käyttäjä lisää järjestelmään tuotteen tai asiakkaan, jolloin tietokantaan tulee uusi merkintä. Järjestelmä palauttaa käyttäjälle tiedon lisäyksen menestyksekkäisyydestä. Kuvassa kuusi ja seitsemän ylläkuvailtu tapahtuma on havainnollistettu sekvenssikaavion avulla.



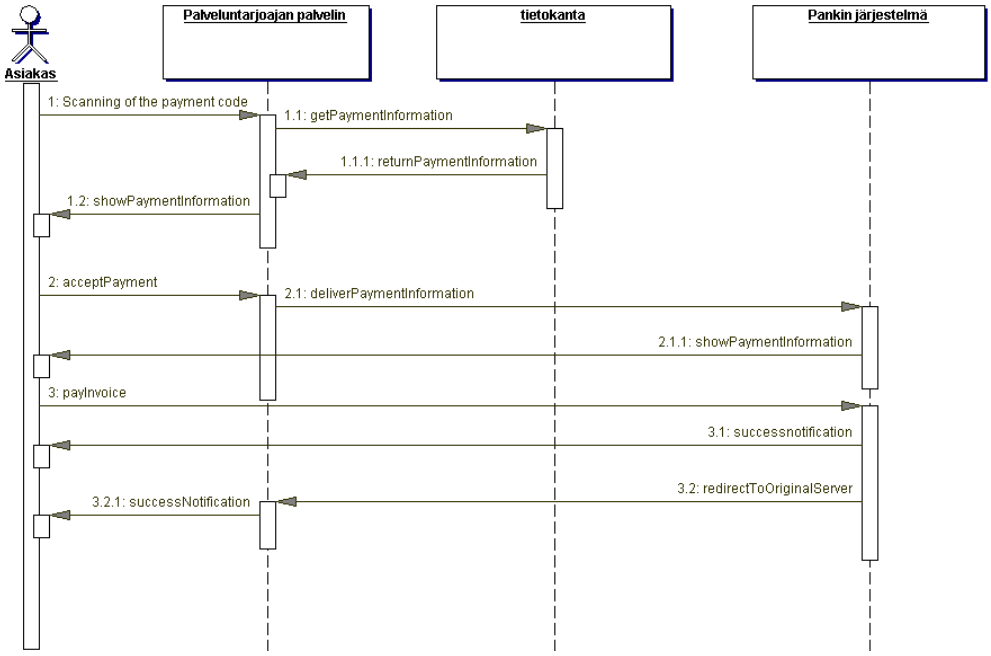

**Kuva 6. Asiakkaan lisääminen asiakasrekisteriin.**




**Kuva 7. Tuotteen lisääminen tuoterekisteriin**

Pikamaksukoodia käyttävä henkilö ottaa yhteyden palvelimeen, joka käy hakemassa koodin osoittamat tiedot tietokannasta ja näyttää ne käyttäjälle. Tämän jälkeen käyttäjä joko hyväksyy tai hylkää maksun. Hyväksyttäessä käyttäjä ohjautuu pankin järjestelmään tiedot mukanaan. Maksamisen ajan käyttäjä on vuorovaikutuksessa pelkästään pankin järjestelmän kanssa. Onnistuneen maksusuorituksen

jälkeen käyttäjä palautetaan alkuperäiselle palvelimelle, joka kertoo maksun menestyksekkäästä suorittamisesta.

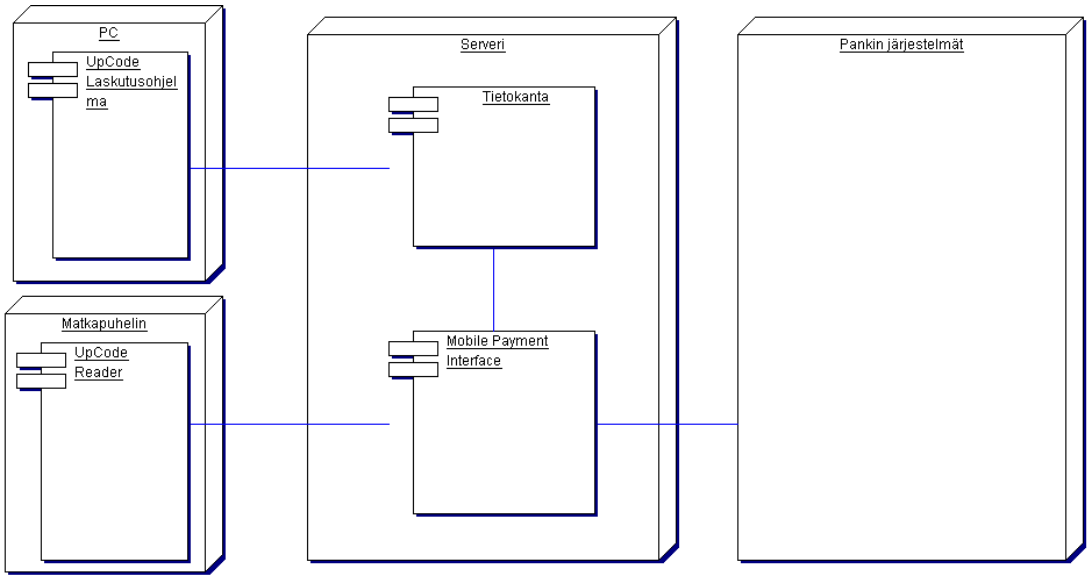



**Kuva 8. Sekvenssikaavio pikamaksukoodin käyttämisestä.**

Yhteistyökaaviolla kuvataan olioiden välisiä kytkentöjä hieman tarkemmin kuin sekvenssikaavion avulla. Yhteistyökaaviossa käytetyt funktiokutsut löytyvät samoilla nimillä myös lähdekoodista. (Liite 2)

Luokkakaaviolla pyritään kartoittamaan ohjelman sisäinen rakenne jo ennen toteutuksen alkamista. Tämä on erittäin tärkeää, varsinkin silloin kun ohjelmaa tekevä kehitystiimi on suuri. Tällöin voidaan jokaiselle antaa oma kokonaisuus toteutusvaiheessa. (Liite 3)

Laitevaatimukset on oleellista tietää jo suunnitteluvaiheessa, jolloin toteutuksessa laitteet on jo hankittuna ja keskittyminen voi olla täysipainotteisesti ohjelmoimisessa. Laitevaatimusten listaamisessa hyvänä työkaluna toimii julkistuskaavio. Siihen on hahmoteltu erikseen laitteet ja niihin asennettavat komponentit. Kuvassa 9 on UpCode maksujärjestelmän toimimiseen vaadittavat laitteet ja niihin asennettavat ohjelmistokomponentit.




**Kuva 9. UpCode maksujärjestelmän julkistuskaavio.**

## 5.4 Toteutus

Ohjelmoinnissa on käytetty C# ohjelmointikieltä ja .NET 3.5 Framework ohjelmistokomponenttikirjastoa. Windows Forms kirjasto on tarjonnut erittäin hyvät työkalut sovelluksen visuaaliseen puoleen.

Tietokannoista otettuihin tietoihin ollaan käytetty suurimmaksi osaksi DataTable luokkaa, johon on helppo tallentaa tietokantahauista syntyneet tiedot. Pyrkimyksenä on ollut tehdä mahdollisimman helppoja SQL-lauseita ja prosessoida tietoa vasta DataTablessa. Kuvassa 10 on esimerkki tyypillisestä tietokantahausta.

```
public DataTable GetAllInvoices()
    {
        ////////////////////////////////////////////////////
        //Luodaan SQL komento ja avataan yhteys tietokantaan. //
        //Tiedot tallennetaan väliaikaisesti adapteriin, josta//
        //ne siirretään DataTableen                          //
        ////////////////////////////////////////////////////
        string sql = "SELECT * FROM ExecutedInvoicesClientInfo ORDER BY
        invoiceID ASC";

        conn = new System.Data.OleDb.OleDbConnection();
        conn.ConnectionString = connstring;
        conn.Open();
        command = new System.Data.OleDb.OleDbCommand(sql, conn);
        adapter = new System.Data.OleDb.OleDbDataAdapter();
        adapter.SelectCommand = command;

        DataTable result = new DataTable();
        adapter.Fill(result);
        conn.Close();

        return result;
    }
```

**Kuva 10. Tyypillinen sovelluksen käyttämä SQL tietokantahaku.**

Datatablen visualisointiin on helppo käyttää DataGridView-luokkaa jolla DataTablessa olevat tiedot saa suoraan tulostettua tietokoneen ruudulle. DataTablen tiedot on ensin prosessoitu haluttuun muotoon, jonka jälkeen DataTable visualisoidaan DataGridViewin avulla.

```
private void ProductForm_Load(object sender, EventArgs e)
        {

        ////////////////////////////////////////////////////////////////////
        //Haetaan tietokannan tuoterekisteristä kaikki tuotteet DataTableen.//
        //Muutetaan DataTablen sarakkeiden nimet halutuksi ja visualisoidaan//
        //DataTable DataGridViewin avulla.                                  //
        ////////////////////////////////////////////////////////////////////
             db = new DbClass();
             DataTable results = new DataTable();
             results = db.GetAllProducts();

             results.Columns["ID"].ColumnName = "ID";
             results.Columns["SPRODUCTNAME"].ColumnName = "Nimike";
             results.Columns["ISTORAGEACCOUNT"].ColumnName = "Varastomäärä";
             results.Columns["SUNITNAME"].ColumnName = "Yksikkö";
             results.Columns["DPRICE"].ColumnName = "Hinta";

             dataGridView1.DataSource = results;
        }
```

**Kuva 11. Koodinäkymä DataTable:n muokkauksesta ja visualisoinnista .**

| | ID | Nimike | Varastomäärä | Yksikkö | Hinta |
|---|---|---|---|---|---|
| ▶ | 1 | Sakset | 5 | kpl | 25,9 |
| | 2 | hiiliä | 345 | kpl | 10,2 |
| | 3 | Kuusilankkua | 300 | metri | 3,5 |
| * | | | | | |

**Kuva 12. Sama asia sovelluksesta näytettynä.**

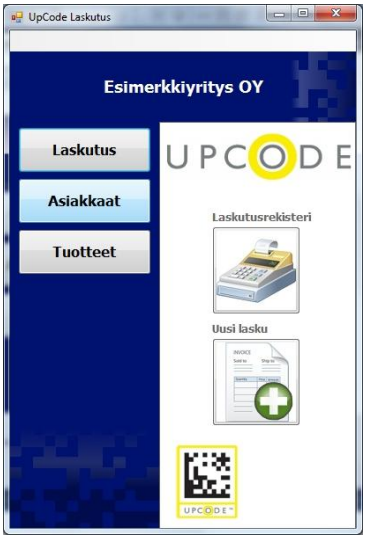
## 5.5 Näkymiä UpCode laskutusohjelmasta

Seuraavaksi muutamia näkymiä itse ohjelmasta. Läpikäytynä on asioita, jotka on olennaista näyttää hieman tarkemmin. Aloitusnäkymässä käyttäjällä on käytössään yksinkertainen menuvalikko, josta voi valita laskutuksen, asiakasrekisterin, tai tuoterekisterin. Hallinnon työntekijöiden haastatteluissa tuli jatkuvasti ilmi las-

kutusohjelmien monimutkaisuus. Yksinkertaisuus onkin ollut perustana sovelluksen toteutuksessa.

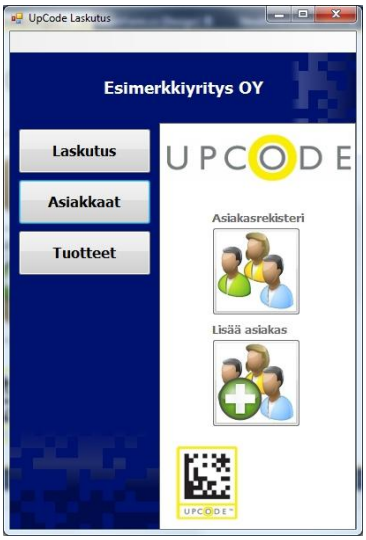

**Kuva 13. Päävalikko.**

Laskutusvalikossa käyttäjä voi valita uuden laskun tai mennä laskuturekisteriin. Yksinkertaisuutta silmälläpitäen valikon napit on tehty erittäin suuriksi ja vaihtoehtojen määrä on rajoitettu minimiin, jolloin käyttäjä osaa käyttää ohjelmaa tehokkaasti ensimmäisestä kerrasta lähtien. Käyttäjällä on kaksi vaihtoehtoa, joko mennä laskutusrekisteriin tai luoda uusi lasku.




**Kuva 14. Laskutusvalikko**

Asiakasvalikossa käyttäjällä on mahdollisuus tarkastella asiakkaita asiakasrekisterissä. Käyttäjä voi myös päivittää asiakkaiden tietoja asiakasrerekisterin kautta. Alemmasta painikkeesta luodaan uusi asiakas rekisteriin.



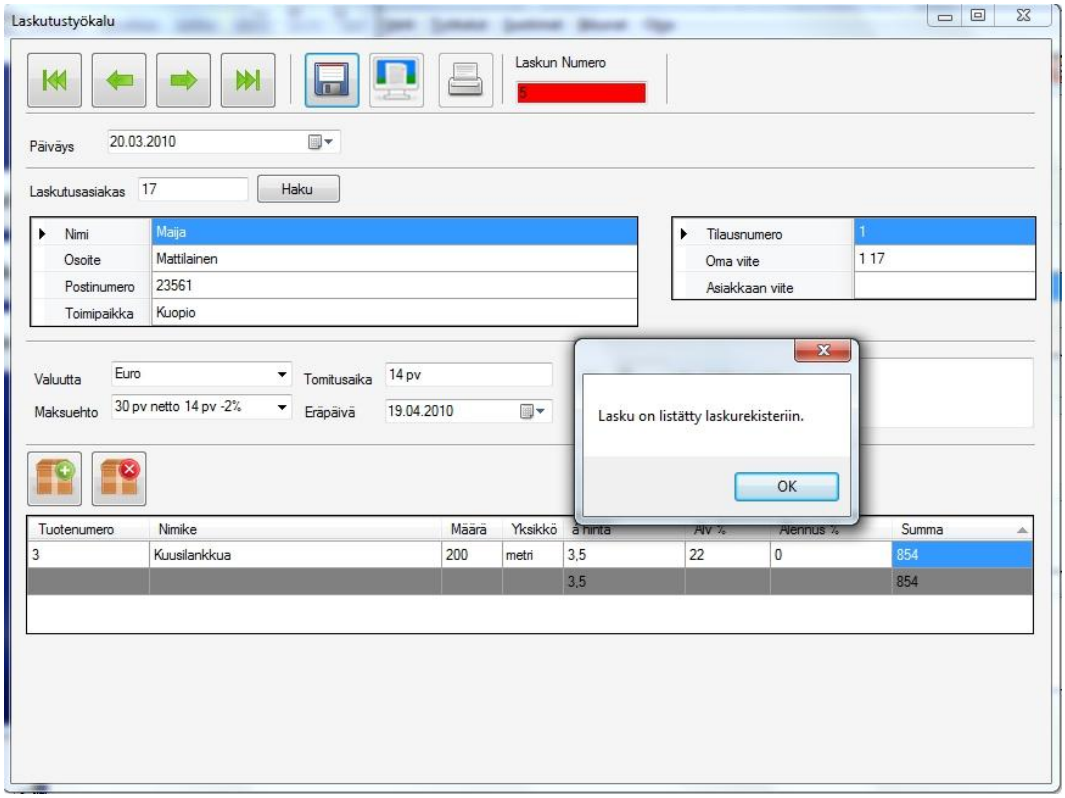

**Kuva 15. Asiakasvalikko**

Tuotevalikossa käyttäjä voi tarkastella tuotteita tuoterekisterissä ja asiakasrekisterin tavoin päivittää tuotteiden tietoja. Myös tuotteiden lisääminen tapahtuu samalla tavalla kuin asiakasvalikossa.


**Kuva 16. Tuotevalikko**

Kaikki valikot on toteutettu samalla tavalla. Form pysyy samana, mutta formin sisällä paneelissa oleva usercontrol vaihtuu sen mukaan mitä nappia käyttäjä painaa valikon vasemmasta laidasta.

Laskutusohjelman suurin kokonaisuus on uuden laskun luomiseen kehitettylle työkalulle. Tässä sovelluksen osassa käyttäjällä on mahdollisuus päästä tarkastelemaan kaikkia rekistereitä ja ottamaan sieltä tietoja uuden laskun pohjaksi ja näin ollen nopeuttaa ja helpottaa toimenpidettä.




**Kuva 17. Laskutustyökalu**

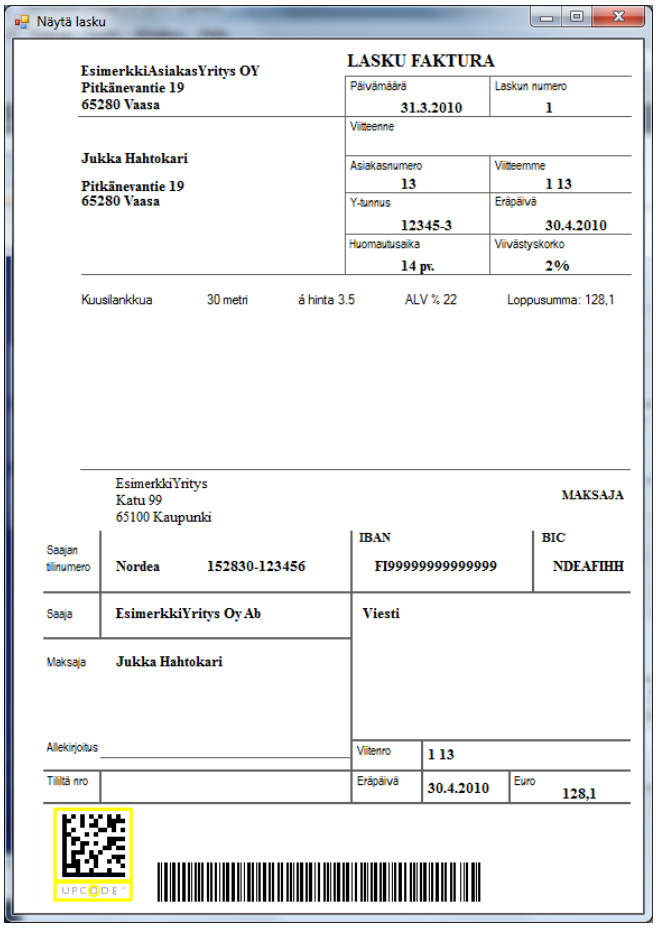
Työkalun vasemmassa yläreunassa on navigointinäppäimet, joilla voi siirtyä katsomaan laskutusrekisteriin tallennettuja laskuja. Kuvassa 17 käyttäjä on juuri painanut tallennuspainiketta ja kyseinen lasku on näin ollen siirtynyt laskutusrekisteriin. Tämän jälkeen passiivisena olevat laskun esikatselu ja laskun tulostus muuttuvat aktiiviseksi. Tallennusnäppäin puolestaan muuttuu passiiviseksi. Laskutusasiakas on haettu asiakasrekisteristä ”Haku” painikkeen kautta. Tilapäisen asiakkaan tapauksessa tekstikentät voidaan myös täyttää käsin.

Käyttäjä on lisännyt tuotteen rekisteristä lisäyspainikkeen avulla. Tilapäisen tuotteen lisääminen taulukkoon onnistuu myös manuaalisesti. Jos tuotetaulukon numeerisia arvoja muutetaan jälkeenpäin, laskee ohjelma automaattisesti uudelleen laskun loppusumman. Tällöin esim. alennuksen antaminen tuotteen lisäämisen jälkeen on helppoa.

# 6 UPCODE PIKAMAKSUKOODI

Laskun generointivaiheessa laskutusrekisteristä käydään hakemassa oikealla laskun tunnuksella asiakkaan tiedot ja laskutukseen lähteneet tuotteet tai palvelut. Sen lisäksi laskun alareunaan generoidaan pikamaksukoodi, jolla laksu voidaan maksaa suoraan matkapuhelimella tai web-kameraa käyttäen tietokoneella. Pikamaskukoodia käytetään skannaamalla se matkapuhelimella tai web-kameralla. Molemmissa tapauksissa skannaamiseen tarvitaan erillinen ohjelma.

Koodin lukemisen jälkeen siirrytään maksupalveluun, johon siirtyy automaattisesti kaikki laskussa olevat olennaiset tiedot, joita tarvitaan sen maksamiseen.




**Kuva 18. Ohjelmalla generoitu lasku esikatseluikkunassa ja pikamaksukoodi vasemmassa alakulmassa.**

Pikamaksukoodi toimii tiedonvälittäjänä asiakkaan ja pankin välillä. Laskun luonnin aikana muiden tietojen ohessa myös koodin numero tallennetaan tietokantaan. Kun asiakas skannaa koodin, siirtyy hän palvelimelle, joka kykenee joko paikallisesti, tai web-palvelun avulla saamaan laskutusrekisterissä olevat laskun tiedot. Näillä tiedoilla esitäytetään html-lomake, joka oikeilla asiakastunnuksilla ja tarkistenumeroilla lähetetään siihen pankkiin, jonka asiakas valitsee. Itse maksun hoitaa aina pankin oma järjestelmä, jonka jälkeen pankin järjestelmä palauttaa asiakkaan takasin alkuperäiselle palvelimelle.

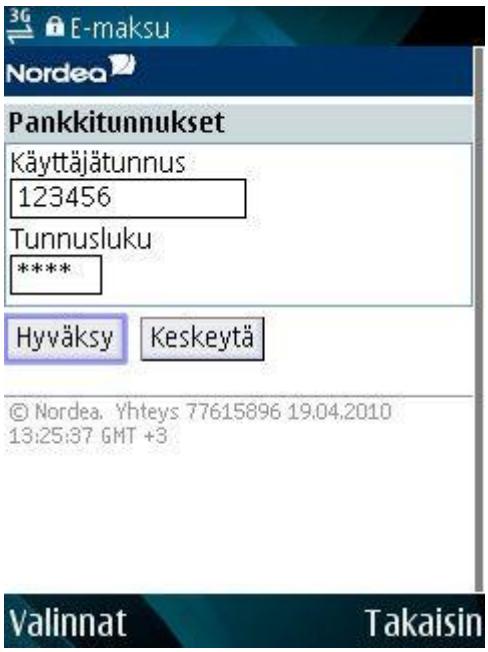
```
////////////////////////////////////////////////////////////////////////////
//Nordean järjestelmään lähetettävä html-form                               //
////////////////////////////////////////////////////////////////////////////
<form method=”post” action=”https://solo3.nordea.fi/cgi-bin/SOLOPM01”>
<input name=”SOLOPMT_VERSION type=”hidden” value=”0003”/>
<%response.write(“<SOLOPMT_STAMP type=””hidden”” value=””&stamp&””/>”)%>
<input name=”SOLOPMT_RCV_ID type=”hidden” value=”12346578”/>
<input name=”SOLOPMT_RCV_ACCOUNT type=”hidden value=”2950180000014”/>
<%response.write(“<SOLOPMT_AMOUNT type=””hidden”” value=””&amount&””/>”)%>
<%response.write(“<SOLOPMT_REF type=””hidden”” value=””&ref&””/>”)%>
<input name=”SOLOPMT_DATE type=”hidden” value=”EXPRESS”/>
<%response.write(“<SOLOPMT_RETURN type=””hidden”” value=””&successaddress&””/>”)%>
<%response.write(“<SOLOPMT_CANCEL type=””hidden”” value=””&canceladdress&””/>”)%>
<%response.write(“<SOLOPMT_REJECT type=””hidden”” value=””&rejectaddress&””/>”)%>
<%response.write(“<SOLOPMT_MAC type=””hidden”” value=””&Ucase(mac)&””/>”)%>
<input name=”SOLOPMT_KEYVERS type=”hidden” value=”0001”/>
<input name=”SOLOPMT_CUR type=”hidden” value=”EUR”/>
<input name=”SOLOPMT_PMTTYPE type=”hidden” value=”M”/>
<input type=”submit” value=”ACCEPT”/>
```

**Kuva 19. Koodinäkymä pankin järjestelmään lähetettävästä html-formista.**

Pikamaksukoodi ei siis hoida itse maskutapahtumaa, vaan siinä käytetään hyväksi pankeissa jo olevaa e-maksupalvelua. Koodin hyödyt ovatkin juuri helppokäyttöisyys ja tietojen virheettömyys.


**Kuva 20. Matkapuhelimen käyttöliittymä ja laskun tietojen tarkistus.**

Kuvassa 20 on käyttöliittymä jossa käyttäjä voi tarkastaa maksun tiedot vielä ennen hyväksymistä. Tässä vaiheessa käyttäjä on vielä palveluntarjoajan palvelimella. Hyväksyttyään maksun käyttäjä siirtyy pankin järjestelmään. Tiedot maksusta siirtyy pankin järjestelmään automaattisesti.




**Kuva 21. Esimerkissä käytetään Nordean maksupalvelua.**

Pankin järjestelmässä käyttäjä antaa omat tunnukset samalla tavalla kun perinteisessä Internetmaksutapahtumassa ja maksaa laskun. Erona perinteiseen maksamiseen on se, että mitään tietoja ei tarvitse täyttää manuaalisesti vaan kaikki maksuun liittyvät tiedot ovat esitäytettyjä.


**Kuva 22. Näkymä onnistuneen maksun jälkeen.**

Maksun jälkeen käyttäjä palautetaan palveluntarjoajan järjestelmään, jossa käyttäjällä on mahdollisuus tallentaa kuitti suoraan kännykkään tai lähettää kuitti sähköpostilla omalle tililleen. Tämän lisäksi maksu luonnollisesti tallentuu pankin omaan järjestelmään.

# 7 LOPPUSANAT

Lopputyönä tehtiin maksujärjestelmä, jossa maksuvaihtoehtoihin on lisätty mahdollisuus suorittaa laskun maksaminen matkapuhelimella. Puhelimella maksamisen helpottamiseksi laskuihin generoidaan pikamaksukoodi, jonka voi lukea matkapuhelimeen asennettavalla ohjelmalla. 2D-koodin lukemisen jälkeen asiakas saa automaattisesti oikeat tiedot, jotka hän tarvitsee laskun maksamista varten.

Yksi suuri kokonaisuus oli laskutusohjelman tekeminen. Laskutusohjelmaan haluttiin ominaisuuksiksi mahdollisuus luoda uusia laskuja, asiakkaita ja tuotteita. Myös mahdollisuus muodostaa laskutus-, asiakas- ja tuoterekistereitä kuului ohjelman tärkeisiin ominaisuuksiin. Ohjelman oli myös kyettävä generoimaan A4:n kokoisia paperimuotoisia laskuja. Generoituihin laskuihin piti saada muodostettua 2D-pikamaksukoodi matkapuhelinmaksamista varten.

Toinen kokonaisuus projektissa oli tehdä html-käyttöliittymä matkapuhelinmaksamisen mahdollistamiseksi. Kummatkin kokonaisuudet on saatu valmiiksi ja ne ovat tällä hetkellä esittely-ympäristössä UpCoden omalla palvelimella.

Tulevaisuudessa laskutusohjelman on määrä ottaa suora yhteys pankin palvelimeen ja hakea sieltä viitepohjaiset maksut maksusuorituksien tarkistamista varten. Tämä saadaan aikaan SOAP-kutsulla, joka allekirjoitetaan pankista saatavalla asiakassertifikaatilla. Tämä ominaisuus on erittäin tärkeä ja se tullaan yhdistämään projektiin. Lisäominaisuus toteutetaan kuitenkin vasta sitten kun sen tekemisen katsotaan olevan ajankohtaista ja kannattavaa.

# 8 LÄHDELUETTELO

/1/ Haukilehto Antti 2002. Visual C#.NET. 1 p. Helsinki. Edita Prima Oy.

/2/ Roger S. Pressman 2000. Software Engineering. A practitioner’s Approach. Berkshire, England. McGraw-Hill Publishing Company.

/3/ UpCode:n Intranet. [viitattu 15.4.2010]

# LIITELUETTELO

**EsimerkkiAsiakasYritys OY**
**Pitkänevantie 19**
**65280 Vaasa**

# LASKU FAKTURA

| Päivämäärä | Laskun numero |
|---|---|
| **31.3.2010** | **1** |

Viitteenne

**Jukka Hahtokari**

**Pitkänevantie 19**
**65280 Vaasa**

| Asiakasnumero | Viitteemme |
|---|---|
| **13** | **1 13** |
| Y-tunnus | Eräpäivä |
| **12345-3** | **30.4.2010** |
| Huomautusaika | Viivästyskorko |
| **14 pv.** | **2%** |

| | | | | |
|---|---|---|---|---|
| Kuusilankkua | 30 metri | á hinta 3.5 | ALV % 22 | Loppusumma: 128,1 |

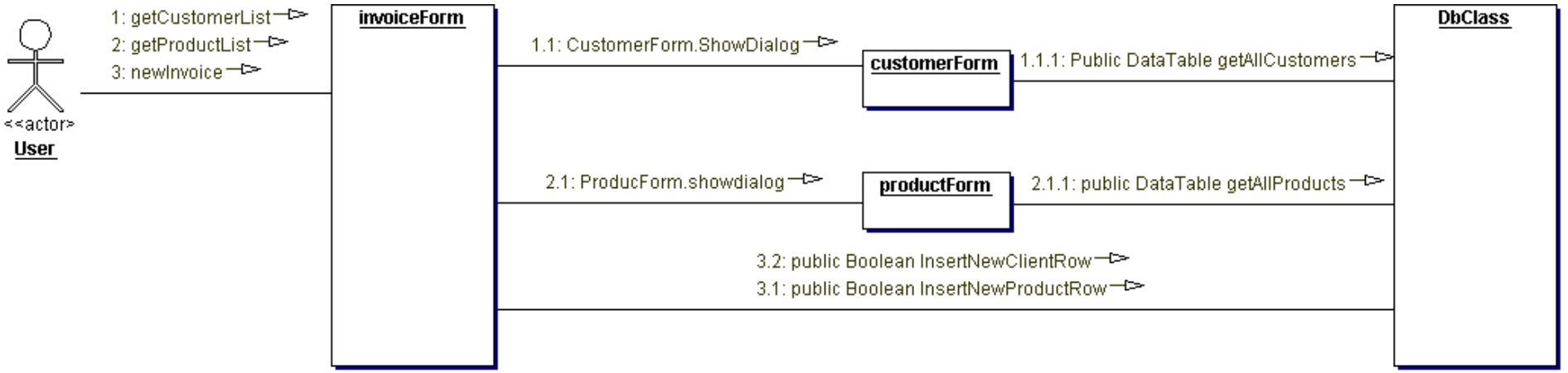
EsimerkkiYritys
Katu 99
65100 Kaupunki

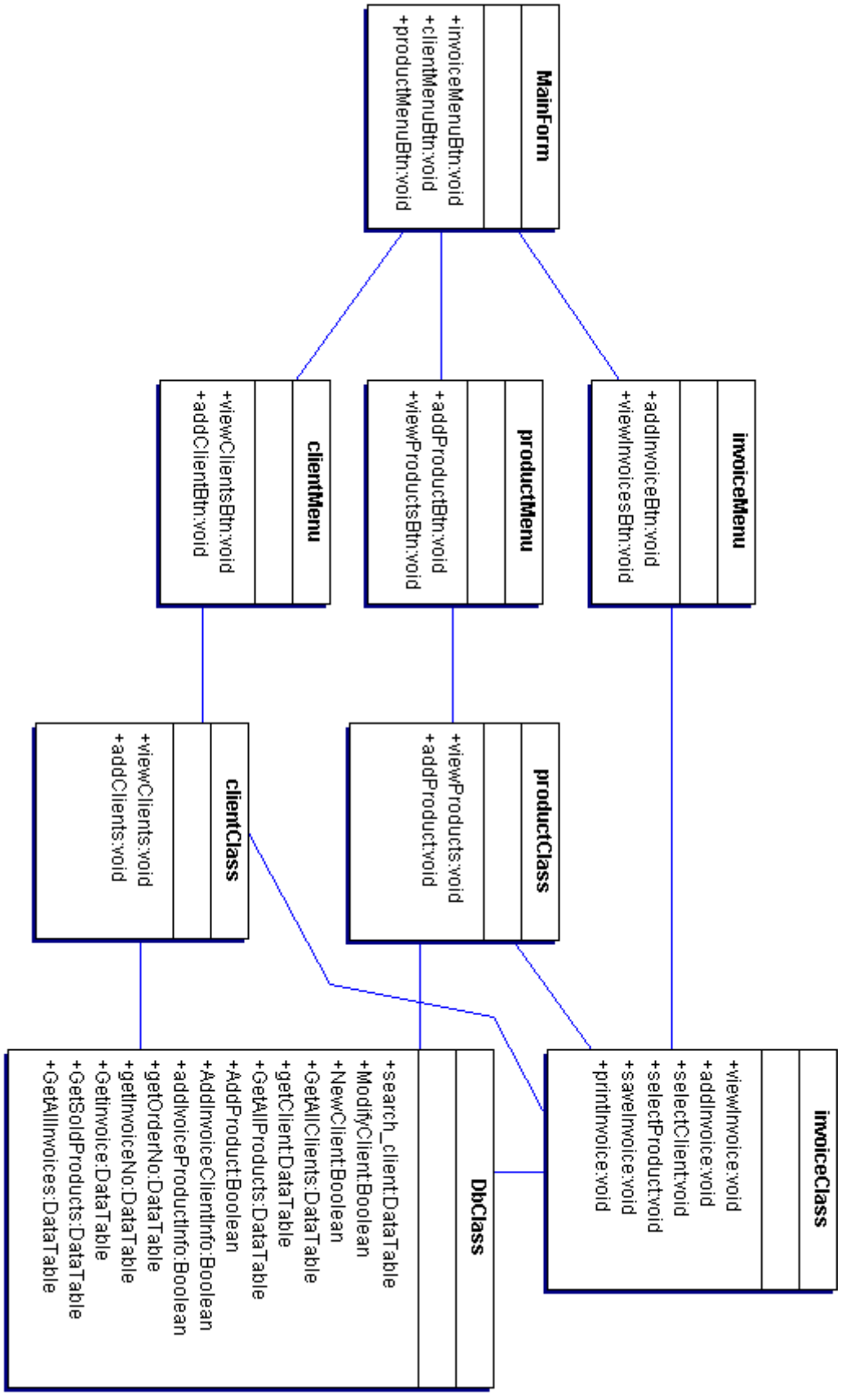
**MAKSAJA**

| Saajan tilinumero | | IBAN | BIC |
|---|---|---|---|
| | **Nordea 152830-123456** | **FI99999999999999** | **NDEAFIHH** |
| Saaja | **EsimerkkiYritys Oy Ab** | **Viesti** | |
| Maksaja | **Jukka Hahtokari** | | |
| Allekirjoitus | | Viitenro | **1 13** |
| Tililtä nro | | Eräpäivä | **30.4.2010** |
| | | Euro | **128,1** |

UPCODE

Liite 2


1: getCustomerList
2: getProductList
3: newInvoice
<<actor>
User
invoiceForm
1.1: CustomerForm.ShowDialog
customerForm
1.1.1: Public DataTable getAllCustomers
DbClass
2.1: ProducForm.showdialog
productForm
2.1.1: public DataTable getAllProducts
3.2: public Boolean InsertNewClientRow
3.1: public Boolean InsertNewProductRow


MainForm
+invoiceMenuBtn:void
+clientMenuBtn:void
+productMenuBtn:void
invoiceMenu
+addInvoiceBtn:void
+viewInvoicesBtn:void
productMenu
+addProductBtn:void
+viewProductsBtn:void
clientMenu
+viewClientsBtn:void
+addClientBtn:void
invoiceClass
+viewInvoice:void
+addInvoice:void
+selectClient:void
+selectProduct:void
+saveInvoice:void
+printInvoice:void
productClass
+viewProducts:void
+addProduct:void
clientClass
+viewClients:void
+addClients:void
DbClass
+search_client:DataTable
+ModifyClient:Boolean
+NewClient:Boolean
+GetAllClients:DataTable
+getClient:DataTable
+GetAllProducts:DataTable
+AddProduct:Boolean
+AddInvoiceClientInfo:Boolean
+addInvoiceProductInfo:Boolean
+getOrderNo:DataTable
+getInvoiceNo:DataTable
+GetInvoice:DataTable
+GetSoldProducts:DataTable
+GetAllInvoices:DataTable