OULUN AMMATTIKORKEAKOULU

Eetu Hernesniemi

BONUSJÄRJESTELMÄ

BONUSJÄRJESTELMÄ

Eetu Hernesniemi
Opinnäytetyö
Kevät 2015
Tietotekniikan koulutusohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan koulutusohjelma, Ohjelmistokehitys

Tekijä: Eetu Hernesniemi
Opinnäytetyön nimi: Bonusjärjestelmä
Työn ohjaaja: Eino Niemi
Työn valmistumislukukausi ja -vuosi: Kevät 2015
Sivumäärä: 37 + 4 liitettä

Opinnäytetyössä kehitettiin uusi ohjelmistokomponentti, joka toimii verkkosovelluksessa ja Windows-ohjelmassa. Opinnäytetyöhön kuuluu ohjelmiston rakenteen kuvausta, ohjelmointimenetelmien teoriaa, ja uusien ominaisuuksien suunnittelua ja toteutusta jo valmiiksi olemassa olevaan ohjelmistoon Microsoft .NET 4.0 -kehikolla ja Microsoft SQL-Server 2008 -tietokannalla.

Työssä oli tavoitteena kehittää kokonaisuus, jonka avulla verkkosovelluksessa myydyistä tuotteista kerääntyisi kanta-asiakasbonuksia kanta-asiakkaille. Tämän myyntitoiminnon kehityksen lisäksi työssä kehitettiin toiminnanohjausjärjestelmään kanta-asiakasbonusten asetukset, Active Report -tuloste ja kanta-asiakas-toimintoja.

Opinnäytetyössä kuvataan verkkosovelluksen toimintaa selain- ja palvelinpuolelta. Lisäksi kuvataan, miten tietokantoja voi hyödyntää .NET 4.0 -kehikon ympäristössä Windows-ohjelman ja verkkosovelluksen ohjelmoinnissa.

Uusien ominaisuuksien toteutuksessa onnistuttiin tavoitteiden mukaisesti. Tässä opinnäytetyössä tehtiin uusia ominaisuuksia valmiiseen ohjelmistoon siten, että saatiin merkittävä uusi kokonaisuus valmiiksi laajempaa testausta ja tuotantoa varten.

Asiasanat: ASP.NET, .NET-kehikko 4.0, Microsoft SQL Server 2008, Kanta-asiakasbonus

ALKULAUSE

Kiitokset PiiMega Oy:lle ja sen henkilöstölle hyvästä työilmapiiristä, hyvistä opeista, hyvästä ohjauksesta, mahdollisuudesta edistää opintojani ja mahdollisuudesta työskennellä teidän kanssanne itseäni kiinnostavalla alalla.

Oulussa 18.3.2015

Eetu Hernesniemi

SISÄLLYS

TIIVISTELMÄ 3
ALKULAUSE 4
SISÄLLYS 5
1 JOHDANTO 6
2 TOIMINTAYMPÄRISTÖ 7
2.1 PiiMega ® Total ERP Magento Edition 7
2.2 Kassa 8
2.3 Tietokantayhteys 10
2.4 Transact-SQL 10
2.5 .NET 4.0 -kehikko 10
3 SUUNNITTELU 12
3.1 Asetukset 13
3.2 Kanta-asiakkaat 13
3.3 Kanta-asiakaskortti 15
3.4 Bonusten tilitys 15
3.5 Bonukset maksutapana 17
3.6 Bonushistoria-tuloste 18
4 TOTEUTUS JA TESTAUS 19
4.1 Kehitysympäristö 19
4.2 Bonusjärjestelmä-Winform 21
4.3 Kanta-asiakkaat 24
4.4 Kanta-asiakaskortti 26
4.5 Bonusten tilitysfunktion ajastustoiminto 27
4.6 Bonusten tilitysfunktio 29
4.7 Bonukset maksutapana 31
4.8 Bonushistoria-tuloste 33
4.9 Rakennekaavio 35
5 YHTEENVETO 37
LÄHTEET 38
LIITTEET 39

1 JOHDANTO

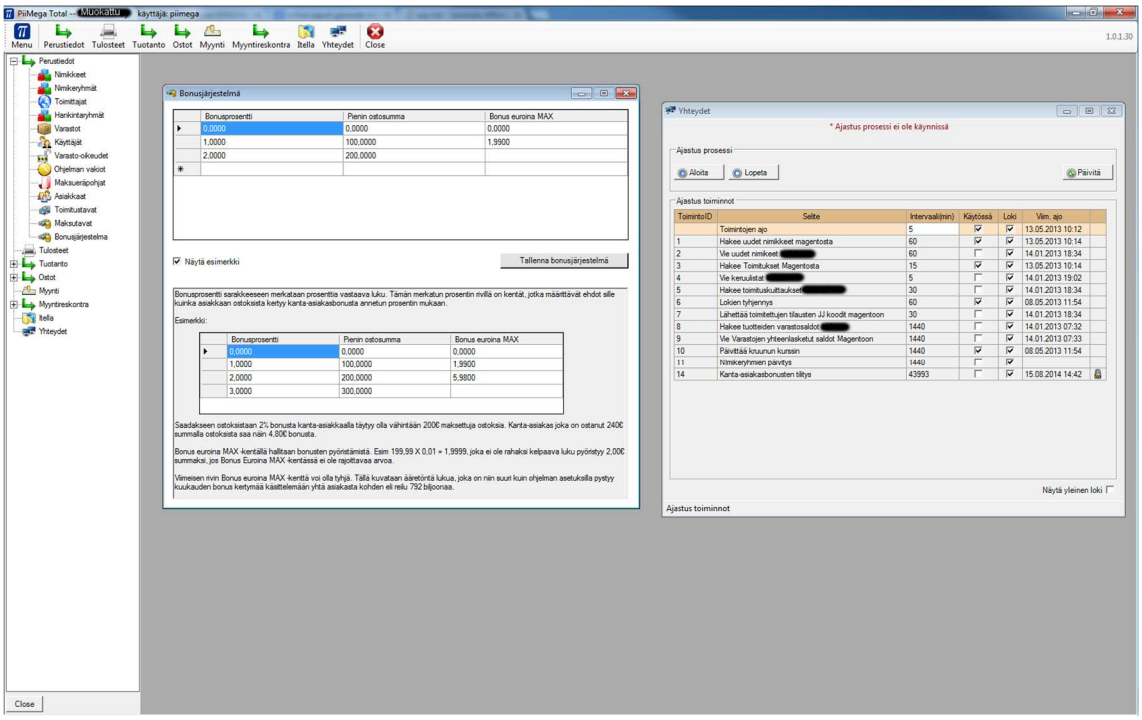
Tämä opinnäytetyö sisältää kuvauksen ohjelmistokomponentista, joka kehitettiin lisäominaisuudeksi työn tilanneen yrityksen PiiMega Oy:n toiminnanohjausjärjestelmään nimeltä PiiMega ® Total ERP Magento Edition ja verkkosovellukseen nimeltä Kassa. Työ perustui tietokantojen hallintaan, Microsoft NET 4.0 -kehikon Windows-ohjelman kehittämiseen ja ASP.NET-verkkosivun kehittämiseen.

Opinnäytetyön tavoitteena oli tehdä uusi komponentti, joka yhdistää Kassan, Total ERP:n ja Magento-verkkokaupan toiminnan siten, että lopputuloksena on Kassassa toimiva myymälä, josta kanta-asiakkaat saavat ostoksiensa arvon suuruuden perusteella rahaa bonuksina. Työssä luotiin toiminto asiakkaiden muuttamisesta kanta-asiakkaiksi, asiakaskohtainen bonusseuranta-tuloste, bonuskertymän asetukset ja kerran kuukaudessa toimiva automaattinen toiminto, joka tilittää kanta-asiakkaille bonukset käyttöön rahana. Tilityksestä kertyneistä bonuksista kehitettiin maksutapa Kassaan. Työhön ei kuulu Magenton toiminnan ohjelmointia ja Kassan muiden myyntitoimintojen kehitystä.

2 TOIMINTAYMPÄRISTÖ

2.1 PiiMega ® Total ERP Magento Edition

Total ERP Magento Edition (kuva 1) on räätälöity toiminnanohjausjärjestelmä, joka on suunniteltu otettavaksi käyttöön organisaatioilla, joilla on verkkokauppa. Se on Windows-ohjelma, joka sisältää erilaisia käyttöliittymiä ja toimintoja, joita aktivoimalla pystyy muokkaamaan käytössä olevaa tietokantaa ja käyttämään erilaisia apuohjelmia. Ohjelman painikkeet, ikkunat ja muut kohteet ovat .NET-kehikon 4.0 standardien mukaisia.



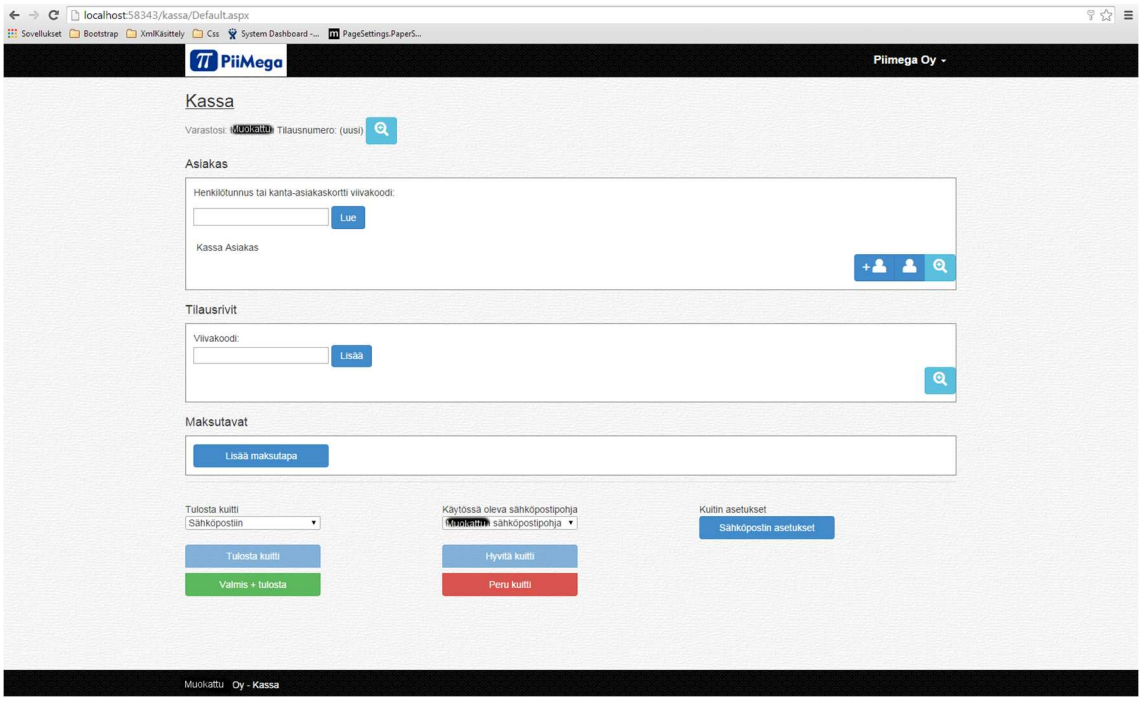
KUVA 1. Toiminnanohjausjärjestelmä Total ERP Magento Edition

Ohjelma on nimetty PiiMegan Total-toiminnanohjausjärjestelmien, Magento-verkkokaupan ja toiminnanohjausjärjestelmän englannin kielisen nimityksen ”Enterprise Resource Planning” mukaan. Sen toimintaan kuuluu mm varaston-hallintaa, kirjanpitoa, resurssien ja omaisuuden hallintaan ja yritys myy sitä käyttöön asiakkaille siten, että ainoastaan tarpeen mukaiset toiminnot ovat esillä ja käytettävissä. (1.)

Total ERP on oltava käytössä Kassaa käyttävällä järjestelmällä tai muuten Kassaa ei voi käyttää. Ohjelmalla on erityisenä tehtävänä ensimmäisellä käynnistyskerralla muokata käytössä oleva tietokanta yhteensopivaksi siihen sisäänrakennetulla kantapäivitys-moduulilla. Kantapäivityksen tarkoitus on järjestää tietokanta hallitusti toiminnanohjausta varten SQL-kyselyjä suorittamalla. Tästä johtuen toiminnanohjausjärjestelmän tietokannat ovat sisällöltään samankaltaisia eri tietokantapalvelimilla. Kantapäivitys suoritetaan ensimmäisen kerran lisäksi myös silloin, kun moduulia on päivitetty.

2.2 Kassa

Kassa-verkkosovellus (kuva 2) toimii työkaluna, jossa myydään käytössä olevan tietokannan varastossa olevia tuotteita, kuten esimerkiksi kumisaappaita tai juomakuppeja. Tietokannan varastotietojen tulee vastata käytössä jotain oikeaa varastoa, josta sitten vähennetään kappalelukumääriä, kun tuotteita on saatu myytyä. Kassa toimii kaikilla yleisesti käytössä olevilla verkkoselaimilla ja kaikki sovelluksen toiminta on sijoitettu näkymään vain yhteen verkkosivuun, jossa näkyvät kaikki olennaiset toiminnot tuotteiden myyntiä varten. Nimensä mukaisesti se muistuttaa toiminnaltaan liikkeessä käytössä olevaa myymälän kassaa, eli Kassan käyttäjänä toimii myyjä eikä ostaja. Kassan käyttöliittymä on ohjelmoitu toimimaan hyvin viivakoodinlukijalla. Kassa lataa ensimmäisellä viivakoodin käytöllä asiakkaan tiedot tietokannasta joko kanta-asiakaskortin tai henkilötunnuksen avulla. Asiakasvalinnan jälkeen viivakoodinlukija täyttää kassan myyntitietoja varastossa olevien tuotteiden viivakoodin perusteella. Kassan myyjä suorittaa lopuksi maksun käsittelyn jälkeen verkkosovelluksella myynnin omalle asiakkaalleen painamalla nappia verkkosovelluksen käyttöliittymässä. Kassan tarkoitus on myydä käyttäjän organisaation tuotteita ja kirjata ylös suoritetut kaupat tietokantaan. Sillä ei ole useampaa käyttötarkoitusta toisin kuin Total ERP:llä.



KUVA 2. Kassa-verkkosovellus

Kassa on kehitetty Microsoftin ASP.NET MVC 4 -ohjelmointimenetelmillä. ASP.NET yhdistää verkkosovelluksen kehityksessä HTML-kielen omiin asp-elementteihin, joita .NET 4.0 -kehikko tarjoaa. Esimerkiksi verkkosivun taulukot ja painikkeet ovat ulkomuodoltaan omalaatuisia verrattuna tavallisiin HTML5-elementteihin ja ne sopeutuvat paremmin sovelluksen palvelinohjelmointiin. Tämä ominaisuus helpottaa ohjelmistokehitystyötä. Lopputuloksena palvelintietokone palauttaa sovelluksen käyttäjän selaimeen vain HTML-elementtien mukaisen näkyvyyden ja lähdekoodin, joka on käännetty alkuperäisestä ASP.NET-lähdekoodista. Selaimen käyttäjä ei tämän vuoksi pääse selaimen näkyvyydessä olevilla tiedoilla käsiksi verkkosovelluksen lähdekoodiin.

MVC-lyhenne tulee englannin kielen sanoista ”Model View Controller”, jotka suomeksi käännettynä tarkoittavat mallia, näkymää ja ohjausta. Palvelintietokone, johon verkkosovellus on asennettu, sisältää lähdekoodissaan mallinnuksen, joka on jonkin olio-ohjelmointikielen luokka. Luokka sisältää rakenteen, jonka pohjalta HTML-näkymä selaimessa esitetään ja ohjauksen, jonka mukaan näkymä toimii, kun verkkoselain lähettää palvelintietokoneeseen HTTP-pyyntöjä. (2, s. 5) ASP.NET-verkkosovelluksessa palvelinohjelmointia tehtiin ohjelmointikielellä VB.NET.

2.3 Tietokantayhteys

Total ERP ja Kassa tarvitsevat molemmat käynnistyessään tietokantayhteyden Microsoft SQL Server 2008 -tietokantaan. Oletusarvoisesti molemmilla ohjelmilla on käytössä sama tietokanta. Tietokantayhteys on molempien ohjelmien elinehto. Total ERP:n ensimmäisessä näkymässä käynnistämisen jälkeen täytyykin määrittää tietokannan verkko-osoite, käyttäjä ja salasana. Kassa alkaa samanlaisella toiminnalla, mutta käyttäjä ei voi määrittää uutta verkko-osoitetta, mihin tietokantayhteys muodostetaan. Ohjelmat esiintyvät ja käyttäytyvät merkittävästi niiden tietokantavakioiden mukaisesti, mihin ohjelman verkkoyhteys muodostetaan. Tietokantavakioihin kuuluu mm. Organisaation nimi, yhteystietoja, kolmannen osapuolen verkko-osoitteita, tunnuksia ja asetuksien arvoja.

Ohjelman käyttäjä pystyy muuttamaan toiminnanohjausjärjestelmästä tietokantavakioiden arvoja vakioiden käyttöliittymästä esimerkiksi vaihtamalla jonkin tekstikentän arvon numerosta yksi nollaksi ja siitä seuraa, että jokin asetus menee pois päältä. Kassan toimintaa ohjataan esimerkiksi toiminnanohjausjärjestelmästä siten, että kassan käyttäjätunnukselle tietokannassa määritetään myytävien tuotteiden varasto, jonka sisältöä verkkosovelluksen käyttäjä tulee käyttämään. Tällä tavalla saadaan säädettyä oikealle myyjälle oikea varasto, jos Kassan käyttäjän organisaatiolla on useampia varastoja tai myymälöitä.

2.4 Transact-SQL

Transact-SQL on Microsoftin kehittämä ohjelmointikieli Microsoft SQL -palvelintietokantoja varten. Transact-SQL tarjoaa kysely- ja tiedonhallintafunktioita. Tavallisen SQL-kielen toimintojen lisäksi sillä voi määrittää ja alustaa muuttujat sekä käyttää yhdistettyjä operaattoreja lähdekoodissa, esim. += -operaattori. (3.)

2.5 .NET 4.0 -kehikko

Microsoftin .NET 4.0 -kehikko on nimitys ohjelmointikehykselle, jossa on käytettävissä erilaisia objekteja kuten painikkeita, ikkunoita ja tiputusvalikoita ohjelmistokehitystyötä varten. Kassa, Total ERP ja Bonusjärjestelmä toteutettiin .NET 4.0 -kehikon standardien ja tarjoamien objektien mukaisesti. Käytössä oli

VB.NET-ohjelmointikieli. Täysin samat kehikon objektit ja ohjelmointikehitystyökalut olisivat olleet saatavilla myös Microsoftin C#-ohjelmointikielelle Visual Studio 2010 -ohjelmassa, jolla opinnäytetyö toteutettiin. (4.)

3 SUUNNITTELU

Bonusjärjestelmän tarkoitus on kerätä etua Kassassa ostoksia tekeville kanta-asiakkaille. Tämä suunniteltiin tapahtumaan lineaarisesti suhteessa ostosumman suuruuteen kuukaudessa, kun ostosumma on tarpeeksi suuri. Lisäksi silloin, kun ostosummien suuruus kaksinkertaistuu, myös ostosta saatujen bonusten määrän pitäisi vähintään kaksinkertaistua.

Bonusjärjestelmään oli määritetty kannustava ominaisuus, joka ilmenee myös oston suuruudessa. Silloin kun ostojen suuruus saavuttaa erikseen asetuksissa määritetyn suuruuden, bonuskertymä ostosummasta saavuttaa suuremman prosenttikertoimen. Tämä tarkoittaa sitä, että suuren hinnan ostokset ovat kuukaudessa bonuskertymän takia edullisempia ostajalle kuin saman verran ostoksia jaettuna kahden kuukauden tilitykseen kahdelle tilitysjaksolle. Bonuskertymän logiikka perustuu taulukkoon 1, joka on tehty liitteen 1 tietojen perusteella.

TAULUKKO 1. Bonusjärjestelmän bonuskertymä.

Tilaussumma kuukauden aikana	**Bonus %**	**MAX euroja bonuksena**
0–99 €	0	0
100–199 €	1	1,99
200–299 €	2	5,98
300–399 €	3	11,97
400–499 €	4	19,36
500–ääretön €	5	50(1000€:sta)

Taulukko 1:n viimeistä saraketta ”MAX euroja bonuksena” tarvitaan tarkentamaan bonuskertymästä kerääntyviä summia ohjelmassa. Ohjelmallisen bonuskertymän laskemiseen tarvitaan kahden desimaaliluvun tarkkuutta, että laskun tulo olisi käytössä olevan rahasumman arvoinen. Tämän toteuttamiseen olisi käytettävä metodia, joka pyöristää luvun aina kahden desimaalin tarkkuuteen. MAX-rajoitteella pyritään estämään sellaiset pyöristykset, että esimerkiksi 199 €:n arvoisesta myynnistä kuukaudessa ei tulisi 2 € bonusta, jos ohjelman käyttäjä haluaa rajata 2 €:n bonuskertymän ainoastaan 200 €:n myyntiä varten.

Opinnäytetyön liitteinä on suunnitteluvaiheessa käytettyjä määritys- ja suunnitteludokumentteja. Bonuskertymästä talteen kirjoitettuja määrityksiä on liitteessä

1. Kanta-asiakaskortin määritetty toimintaperiaate on kirjoitettu liitteeseen 2. Tilitysfunktion ja bonushistorian toimintaperiaate on Excel-dokumentin kuvassa liitteessä 3. Bonusasetuksien kehitystä ja käyttämistä suunniteltiin liitteessä 4.

3.1 Asetukset

Bonuskertymän Asetukset -käyttöliittymä todettiin suunnitteluvaiheessa olevan parempi olla Total ERP:ssä kuin Kassassa, jottei yksinkertaisemmasta myynti-tarkoitukseen kehitetystä käyttöliittymästä tulisi monimutkaisempaa. Suunnitte-luvaiheessa päätettiin, että asetuksien toteuttamiseen tarvitaan uusi ikkuna, taulukko 1:n kaltainen muokattavissa oleva taulukko ja Tallenna-painike.

Käyttöliittymä suunniteltiin toimivan siten, että kun sen ikkuna avataan Total ERP:ssä, bonusjärjestelmän asetuksien tiedot latautuvat tietokannasta. Tieto-kannassa tulisi olla vastaavanlainen taulu kuin taulukko 1, jonka tiedot ladattai-siin Total ERP:n käyttöliittymässä olevaan tauluun. Tämä muokattavissa oleva taulu tulisi sitten voida tallentaa tietokantaan nappia painamalla. Nämä tieto-kannassa tallessa olevat asetukset tulisivat sitten määräämään bonusjärjestel-män tilitystoimintoja.

3.2 Kanta-asiakkaat

Bonuksia tulisi kertyä vain kanta-asiakkaille, joten olemassa oleviin tietokannan asiakastietoihin tulisi saada tieto siitä, että ketkä asiakkaat ovat kanta-asiakkaita. Lisäksi bonuksien kerääntyminen kuukaudessa tulisi alkaa vasta siitä ajasta lähtien, milloin kanta-asiakkaaksi on liitytty. Tämä tulisi valmiiden ohjelmien rakenteiden vuoksi toteuttaa kantapäivitys-moduulissa SQL-kyselyllä, jossa tauluun Asiakas lisättäisiin uusi sarake ”LiittynytKantaAsiakkaaksi.” Sa-rakkeen kentät sisältäisivät sitten jokaisen asiakkaan rivillä tiedon tästä kanta-asiakkaaksi liittymisen ajankohdasta päivämäärän ja kellonajan tarkkuudella. Tavallisilla asiakkailla tämä uusi kenttä sisältäisi ajan sijaan null-arvon. Bonus-järjestelmää kehittäessä asiakkaat ja kanta-asiakkaat tulisi erottaa sen toimin-nassa toisistaan tutkimalla onko tietokannassa asiakkaiden tiedoissa uuden sarakkeen kohdalla null-arvo vai aika.

Kassassa oli jo ennestään olemassa toiminto, jossa luotiin uusi asiakas ja syötettiin asiakkaalle tieto, jotka sitten tallentuivat Asiakas-tauluun tietokannassa. Molempiin ohjelmiin tarvittiin toiminto, jolla asiakas liitetään kanta-asiakkaaksi. Kassan asiakkaiden tietojen luomisen yhteyteen suunniteltiin yksinkertainen ratkaisu toiminnolle, jossa haluttiin luoda kanta-asiakas. Käyttöliittymään tulisi lisätä valintaruutu, jonka vieressä olisi teksti ”kanta-asiakas.” Valintaruutua tulisi painaa graafisessa käyttöliittymässä kerran ja sitten tieto ajasta, jolloin asiakas liittyi kanta-asiakkaaksi, tulisi siitä ajanhetkestä, milloin asiakkaan luonti vahvistetaan Kassassa. Kassassa ei ollut mahdollista muokata olemassa olevan asiakkaan tietoja, joten bonusjärjestelmän kehittämisen yhteydessä asiakkaan tietojen muokkauksen käyttöliittymä olisi kehitettävä samoilla periaatteilla kuin olemassa oleva käyttöliittymä uuden asiakkaan luontia varten. Kassan uudessa tietojen muokkaamisessa voitaisiin sitten muuttaa tavallisia asiakkaita kanta-asiakkaiksi.

Asiakkaiden tietojen muokkaus tulisi toteuttaa ASP.NET:n **DetailsView-elementillä**, joka on eräänlainen verkkosivun muotoelementti. Tämä muotoilu rajaa riveittäin tekstikenttiin asiakkaiden muokattavat tiedot ja asettaa muokkauksen vahvistamispainikkeen elementin loppuun. Verkkosovelluksen palvelinpuolen lähdekoodissa tulisi sitten hyödyntää DetailsView-elementin tapahtumien aktivointia. Verkkosovellus ohjelmoitaisiin sitten tutkimaan DetailsView-elementin tekstikenttiä ja valintaruutuja ja sitten tutkimuksen tulosten perusteella se ohjelmoitaisiin päivittämään tietokannan Asiakas-taulua uusien tietojen mukaisesti. Detailsview-elementin tulisi sijoittua painikkeesta avattavaan ponnahduslaatikkoon samalla tavalla kuin uuden asiakkaan luomisen käyttöliittymässä.

Total ERP:ssä on myös samankaltainen käyttöliittymä, jossa asiakkaan luomisen lisäksi tietoja pystyttiin muokkaamaan. Total ERP:n asiakkaiden käyttöliittymään suunniteltiin kokonaan uusi välilehti, jossa bonusjärjestelmään yksittäisiä asiakkaita koskevat toiminnot toteutettaisiin. Täällä olemassa olevia asiakkaita muutettaisiin kanta-asiakkaiksi pelkästään nappia painamalla.

3.3 Kanta-asiakaskortti

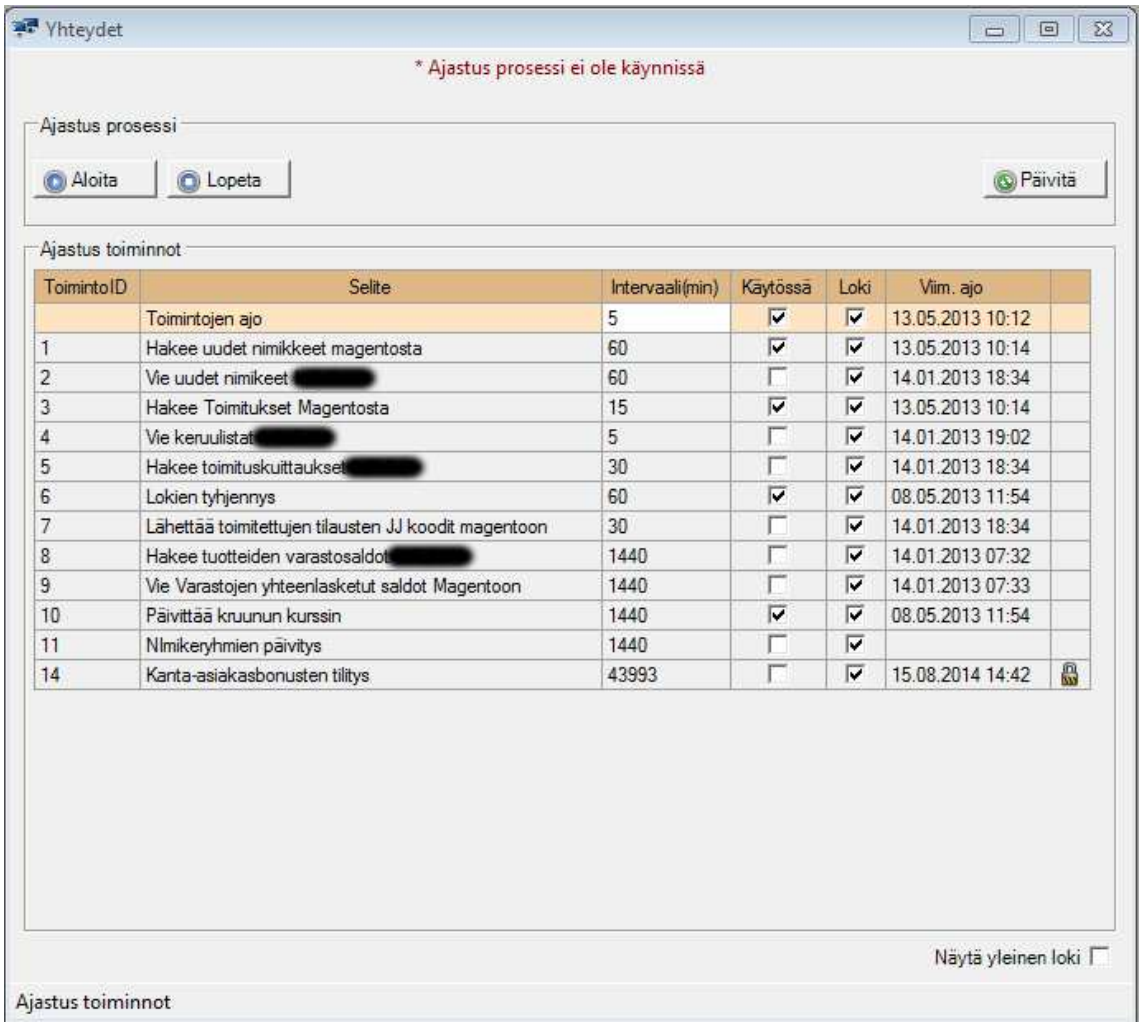
Kanta-asiakaskortti on muovinen kortti, joka sisältää nelinumeroisen luvun. Kortin käyttötarkoitus on helpottaa Kassan käyttöä viivakoodinlukijalla. Kassan käynnistyessä on sen ensimmäisenä toimintona valita asiakas tietokannasta tai luoda uusi asiakas. Kassaan tulisi asettaa vaihtoehtoinen toiminto, joka toimii nopeampana pikavalintana kanta-asiakaskortilla tai ajokortin henkilötunnuksella asiakkaan valitsemista varten.

Kanta-asiakkaaksi liittymisen ajankohdalla asiakkaalle annetaan tämä kanta-asiakaskortti ja tämä tarkoittaa sitä, että kantapäivitys-moduulin uuteen SQL-kyselyyn on saatava mukaan uusi kenttä, joka sisältää tiedon kanta-asiakkaan kortin numerosta. Muuten kortinlukija ei löydä asiakasta tietokannasta kortin numerolla.

3.4 Bonusten tilitys

Total ERP:hen on valmiiksi rakennettu ajastusprosessi. Tämän prosessin tarkoitus on olla aina päällä palvelintietokoneessa, jossa toiminnanohjausjärjestelmän tietokanta on käytössä. Toimintojen aktivoituessa ne suorittavat erilaisia funktioita, joissa esimerkiksi päivitetään tietokantoja, otetaan yhteyttä kolmannen osapuolen verkkopalvelimeen ja imuroidaan ja viedään tiedostoja. Total ERP:n toimintoihin kuuluukin lähettää kerääntyneet kanta-asiakasbonukset muiden kauppaa koskevien tietokannan tietojen kanssa ohjelman käyttäjän organisaation Magento-verkkokaupan palvelimelle. Tätä ominaisuutta ei ollut tarpeen kuitenkaan kehittää opinnäytetyössä, koska toiminto oli jo olemassa.

Bonusten tilitystä varten tarvittiin uusi ajastettu toiminto ajastettujen toimintojen käyttöliittymään (kuva 3), joka aktivoituu oletuksena joka kuukauden 15. päivä. Tätä päivää tulisi pystyä muuttamaan Total ERP:n tietokantavakioiden asetuksista. Uuden tietokantavakion luominen tulisi suorittaa kantapäivitys-moduulissa.



KUVA 3. Ajastettujen toimintojen käyttöliittymä

Ajastustoiminto kanta-asiakasbonusten tilitys suunniteltiin suorittamaan kaksi funktiota. Ensimmäinen funktio tarkistaa Total ERP:n asetuksista bonusten tilityspäivän kuukaudessa ja laskisi, paljonko aikaa toiminnon aktivointi hetkestä on seuraavaan suoritus kertaan. Ensimmäinen funktio kutsuu suorituksen aikana toista funktiota, joka suorittaa bonusten tilityksen kanta-asiakkaille järjestelmässä määritetyn bonusasetuksien mukaisesti.

Aika toiminnon seuraavaan suoritukseen näkyy käyttöliittymän taulukossa sarakkeessa Intervalli(min), joka ilmoittaa lasketun ajan seuraavaan suorituskertaan minuutteina. Ajastusprosessin toiminta on suunniteltu siten, että kun intervallin mukainen aika on kulunut, prosessi aloittaa oikean toiminnon suorittamisen.

Ajastustoimintojen suorituksen pystyy suorittamaan myös aktivoimalla halutun toiminnon käyttöliittymästä manuaalisesti. Tämä täytyi ottaa huomioon bonusten tilitys -toiminnossa siten, että toiminnon suoritus ei aktivoidu, jos ohjelman asetuksissa ei ole erikseen määritetty kuukauden tilityspäivää tai jos toimintoa yritetään suorittaa ennen määritettyä tilityspäivää.

Bonusten tilitys on bonusjärjestelmän laajin osuus ja se suunniteltiin toteutettavaksi pääasiassa erittäin laajalla SQL-kyselyllä. Tilityksen tulisi käsitellä tietokantaan tallennettujen kanta-asiakkaiden ostoksia ja tallentaa bonukset bonusjärjestelmän asetuksien mukaan tietokantaan. Tätä varten on kantapäivitysmoduulissa luotava taulu, joka kuvaa asiakkaiden bonustapahtumia. Sitä päätettiin kutsua liite 1:ssä olevien määritysten perusteella nimityksellä AsiakasBonusHistoria.

Bonusten tilittämisessä huomioidaan lisäksi kanta-asiakkaiden tuotteiden palautukset, joista asiakas saa rahallista hyvitystä. Tilityksen on toimittava siten, että ostoksilla on yhden kuukauden palautusaika, josta seuraa negatiivinen vaikutus kertymäsummaan voimassa olevan palautusajan palautuksesta, joka kanta-asiakkaalle tilitetään edellisen kuukauden ostoksista. Lisäksi tilityksessä on tarkistettava palautuksen ostohetken ajalta, onko asiakas jo liittynyt silloin kanta-asiakkaaksi. Tällaisen tarkistuksen avulla tilitystoiminto ei koskaan poista kanta-asiakkaan bonuksia enempää kuin mitä asiakkaalle on bonuksia kertynyt.

Tilitysfunktio suunniteltiin tutkimaan bonusten kertymää kaikkien kanta-asiakkaiden ostoksista edellisen kuukauden alusta lähtien kuukauden loppuun saakka. Tilitystoiminnon pitäisi tämän lisäksi tutkia onko asiakas ollut kanta-asiakas edellisen kuukauden alussa. Tutkimuksen perusteella ohjelma rajaa bonuskertymän ajankohdaksi edellisen kuukauden alun tai ajan, milloin asiakas liittyi kanta-asiakkaaksi.

3.5 Bonukset maksutapana

Kassassa oli bonusjärjestelmän suunnitteluvaiheessa mahdollista tallentaa myyntitapahtuman maksutavaksi korttimaksu, käteinen tai molemmat. Kanta-asiakkaan kerääntyneet bonukset tulisi summata yhteen valitulle asiakkaalle

kassassa, että kerääntyneitä bonuksia voitaisiin käyttää kassassa maksutapana. Maksutapa kassassa valitaan tiputusvalikosta ja siellä olisi esillä maksutapa bonus ainoastaan, jos valittuna asiakkaana olisi kanta-asiakas, jolle on kertynyt bonuksia. Kassan käyttöliittymässä tulisi olla esillä tieto vihreällä huomiovärillä siitä, että onko asiakas kanta-asiakas ja vielä erikseen siitä, että onko hänellä kerääntynyt bonuksia käytettäväksi. Bonuksia maksutapana tulisi voida käyttää muiden maksutapojen kanssa samanaikaisesti.

3.6 Bonushistoria-tuloste

Bonusjärjestelmään tarvittiin vielä lopuksi toiminto, josta asiakkaiden kerääntyneet bonukset olisivat helposti nähtävissä ja seurattavissa. Tämän ominaisuuden toteuttamiseksi olisi hyvä tehdä **ActiveReport-tuloste**, joka tarkoittaa sitä, että tietokannassa olevat tapahtumat bonusten kerääntymisestä ja käyttämisestä maksuna esitettäisiin dokumenttina pdf-tiedoston muodossa. ActiveReport on Microsoft .NET -kehitystyökalu ja tulosteen pohja kehitettäisiin työkalun tarjoamilla metodeilla, jossa määritetään esimerkiksi tiedoston luontivaiheessa tiedot tulosteeseen SQL-kyselyllä tiedoista, joita haluttaisiin esittää.

Tuloste avattaisiin Asiakas-käyttöliittymän uudesta Kanta-asiakkuus-välilehdestä ja Total ERP:n omasta tulosteet-käyttöliittymästä. Molempiin tulisi asettaa kehitystyökaluilla päivämäärän valitsemiseen tarkoitetut valikot, joista saataisiin rajattua tulosteen tietojen aikaväli. Tuloste voitaisiin avata tuloste-käyttöliittymässä siten, että se näyttää joko yksittäisen kanta-asiakkaan tai kaikkien kanta-asiakkaiden bonushistorian ja yhden asiakkaan käyttöliittymästä vain valitun asiakkaan bonushistorian.

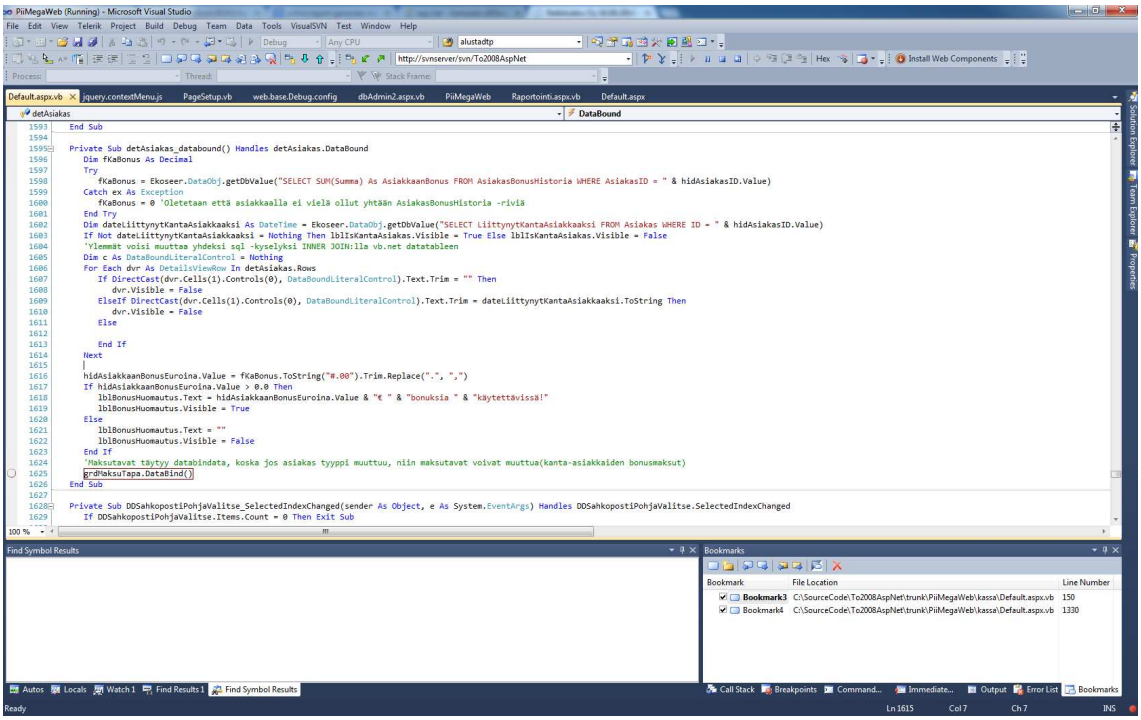
Aikavälin vaikutukset suunniteltiin siten, että tuloste näyttää valitulta aikaväliltä jokaisen kanta-asiakkaan kaikkien bonustapahtumien maksut ja kerääntymät riveittäin. Rivien jälkeen loppuun olisi laitettava yhden kanta-asiakkaan rivien yhteinen summa ja vanha bonusten käyttösumma, joka oli käytössä valitun aikavälin lopussa.

4 TOTEUTUS JA TESTAUS

Bonusjärjestelmän toteuttamisessa hyödynnettiin Total ERP:n valmiita luokkarakenteita. Luokkarakenteisiin tehtiin uusia funktiota ja muokattiin olemassa olevia funktiota. Esimerkiksi Kassa-verkkosovellus käytti Total ERP:n tarjoamaa luokkaa **VK_Yle**, joka pitää sisällään funktiota, jotka on tarkoitettu olemaan yleisessä käytössä verkkokaupan ohjelmoinnissa. Total ERP:n ohjelmaprojekti tekee lähdekoodin käännöstyön jälkeen dll-tiedoston, johon Kassan verkkosovellus-ohjelmointiprojektissa on viittaukset. Verkkosovelluksen ohjelmoinnissa sitten kutsutaan näitä dll-tiedoston tarjoamia funktiota. Tällaisilla menettelyllä saadaan kaikki erilaiset sovellukset käyttämään samaa metodia, kun on esimerkiksi tarve suorittaa verkkokaupan maksu samalla tavalla. Tämä on hyvä menettely ja sellaisen funktion toimintaan pystyttiin vaikuttamaan myös parametreilla funktion kutsun yhteydessä.

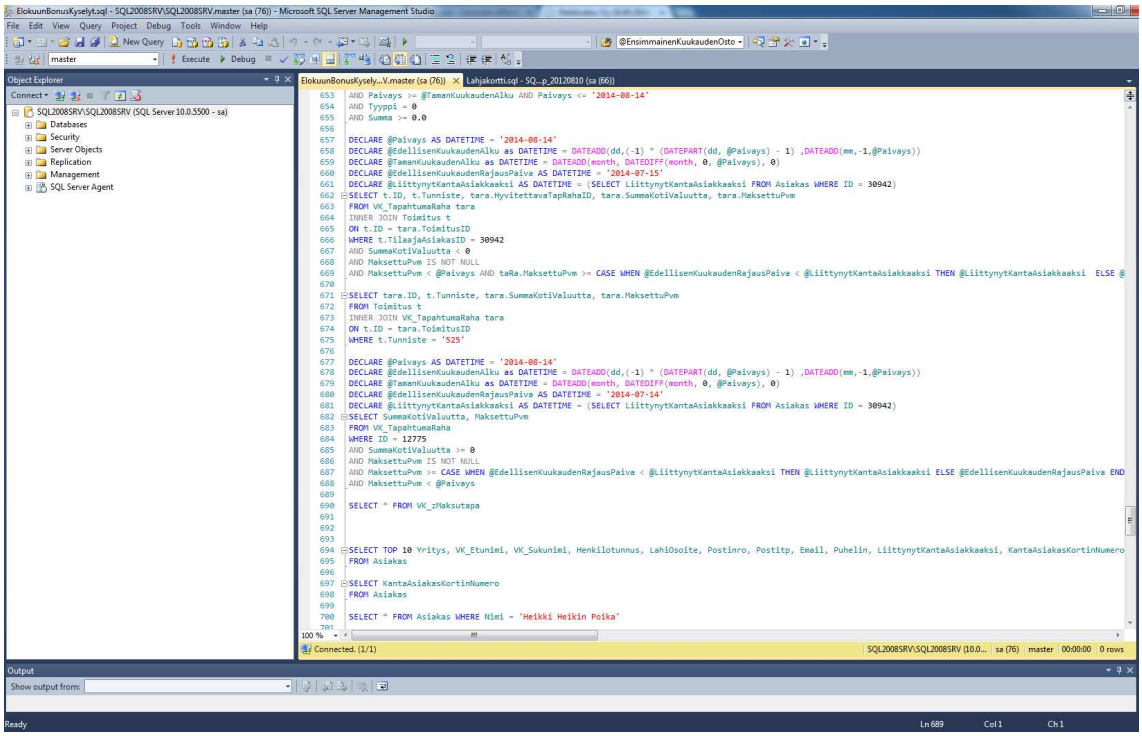
4.1 Kehitysympäristö

Bonusjärjestelmän osuuksia testattiin Microsoft Visual Studio 2010:n (kuva 4) virheiden jäljittäjä -työkalulla ja Microsoft SQL Management Studio 2010 -ohjelmalla (kuva 5). Virheiden jäljitys -työkalun avulla seurattiin lähdekoodin suoritusta riveittäin. Työkalu keskeyttää lähdekoodin suorittamisen ohjelmassa ja antaa keskeytyksen aikana tarkistella muuttujien ja objektien sen hetkisiä tietoja. Työkalun käyttöä hyödynnettiin etenkin bonusten tilityksen tarkistuksessa, jossa erilaisia tilityksien ominaisuuksia pystyttiin keskeyttämään ja tutkimaan kesken ohjelman suorituksen.



KUVA 4. Bonusjärjestelmän ohjelmistokehitystyökalu: Visual Studio 2010

SQL Management Studio -työkalulla otettiin yhteys kehitysympäristön testitietokantoihin ja suoritettiin SQL-kyselyjä. Tämän työkalun tarkoituksena oli vain testata käytännössä SQL-kyselyjen toimivuutta. Testaamiseen kuului kyselyjen oikeinkirjoituksen tarkistaminen ohjelman kääntäjällä ja kyselyn vaikutuksen analysointi tietokantapalvelimessa. Työkalu tarjoaa mahdollisuuden suorittaa SQL-kyselyjä Microsoftin omalla Transact-SQL-kielellä. Transact-SQL teki mahdolliseksi testata SQL-kyselyjä siten, että aloitettiin nk. Tran-kysely, joka varaa ohjelman muistiin tietokannan nykyisen tilan ennen SQL-kyselyn suorittamista. SQL-kyselyn jälkeen edellä mainitussa tilassa voidaan tutkia SQL-kyselyn toimivuutta ja sitten palauttaa tietokanta toimintoa edeltävään tilaan. Tämä oli varsin hyvä tapa testata laajoja ja monimutkaisia SQL-kyselyjä, sekä arvioida niiden käytettävyyttä.



KUVA 5. Bonusjärjestelmän ohjelmistokehitystyökalu: SQL Management Studio 2012

4.2 Bonusjärjestelmä-Winform

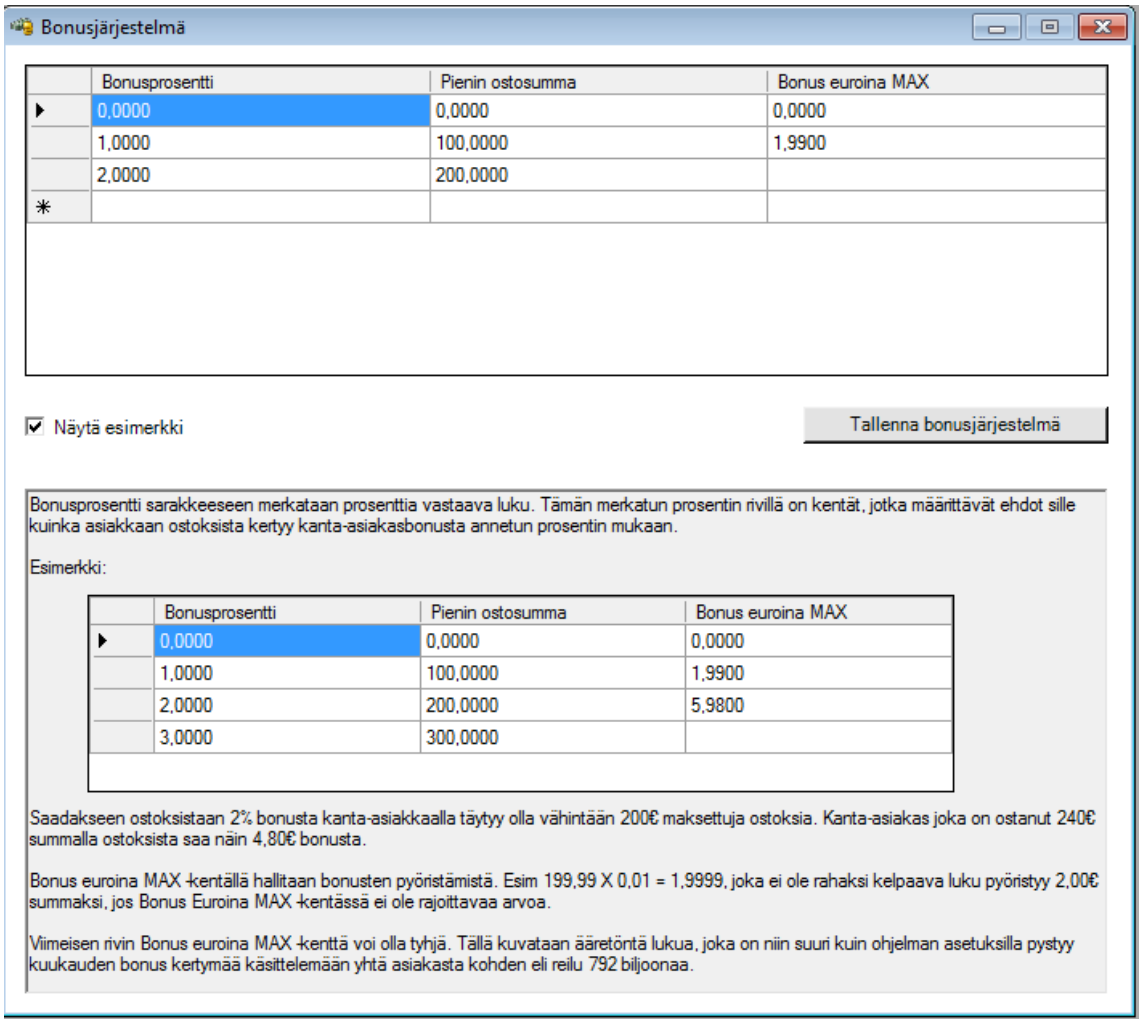
Bonusjärjestelmä-Winformiin kehitettiin bonusjärjestelmän suunnitelmien mukaiset asetukset. Bonusjärjestelmän asetuksien käyttöliittymä on aivan itsenäinen osa Total ERP:n toiminnoissa ja tähän käyttöliittymään ei ole sijoitettu mitään opinnäytetyön ulkopuolisiin asioihin liittyviä toimintoja toisin kuin muissa bonusjärjestelmän osa-alueissa.

Bonusjärjestelmä-Winform (kuva 6) on .NET 4.0 -kehikon Windows-ohjelman ikkuna. Tämä ikkuna on käytettävissä sisäänrakennettuna resurssina Visual Studio 2010 -työkalussa. Visual Studio -työkalulla luotiin uusi winform-luokka, joka nimettiin Bonusjärjestelmän mukaan nimellä Bonusjärjestelmä. Tälle luokalle on Visual Studiossa graafinen näkymä ja työkalu, jolla ikkunan muotokuvaan pystyttiin lisäämään suunnitelmien mukaisesti taulukko ja painike. Graafisen työkalun käyttö ikkunassa lisää automaattisesti lähdekoodia Bonusjärjestelmä-winformin luokkaan. Taulukko ja painike perivät .NET 4.0 -kehikon oman tarkoituksensa mukaisen luokan, kun ne lisätään osaksi Bonusjärjestelmä-

winformia. Perittyjen luokkien avulla saadaan kehitystyössä muokattavissa oleva objekti, jota muokattiin helposti sopivaksi omaan tarpeisiin luokan metodien avulla. (5.)

Taulun luokka **DataGridView** tarjoaa **init-tapahtuman**, joka antaa mahdollisuuden suorittaa ohjelmakoodia sillä hetkellä, jolloin taulu ilmestyy näkyviin ohjelmassa. Init-tapahtuman ohjelmakoodissa toteutettiin VB.NET-koodilla menetelmiä, joilla asetetaan tauluun yhtä monta riviä kuin SQL-palvelimella on rivejä taulussa **BonusAsetukset**. DataGridView-luokka sisältää mahdollisuuden asettaa SQL-kyselyn palauttamat tiedot omaksi näkymäksi taulukkoon ja tätä mahdollisuutta hyödynnettiin asetuksien kehittämisessä. Total ERP sisältää luokan **Ekoseer**, jossa on metodeja, joilla voi suorittaa SQL-kyselyjä sekä asettaa SQL-kyselyn palauttamat arvot taulun tietojen arvoksi. Taulukon kehittämisessä riitti, että käytettiin näitä valmiita metodeja ja kehitettiin sellainen SQL-kysely, joka palauttaa nykyisen tiedon tietokannan taulusta BonusAsetukset. Tätä varten kantapäivitys-moduuliin lisättiin SQL-kyselyllä taulukko 1:n kaltainen taulu, jossa oli sama määrä sarakkeita. Taulun luomista testattiin Tran-kyselyllä ennen sen liittämistä Total ERP:hen.

DataGridView-taulukon asetuksien tallentaminen tietokantaan toteutettiin ja testattiin samankaltaisella menetelmällä. Tässä tapauksessa tehtiin kuitenkin siten, että ohjelmakoodi lisättiin painikkeen painamisen **click-tapahtumaan** init-tapahtuman sijaan. SQL-kysely poikkesi siten, että se tallensi tietokantaan uuden ohjelman muokattujen asetuksien taulukon mukaisen tiedon. Tallennuspainikkeeseen ohjelmoitiin myös tarkistus, jossa tarkistetaan taulun olevan järkevästi täytetty. Napin painaminen toteutettiin siten, että painamisen jälkeen ilmestyy .NET-ponnahdusikkuna, joka ilmoittaa joko bonusjärjestelmän asetuksien tallentumisen tai korjauskehotuksen. Korjauskehotus ilmestyy, jos jokin muu kuin viimeisen rivin viimeinen ruutu asetus-taulukon kentistä on jätetty tyhjäksi. Tämä poikkeuksellinen tyhjä ruutu tarkoittaa, että käytössä on ääretön summa.



KUVA 6. Asetuksien käyttöliittymä: Bonusjärjestelmä-Winform

Bonusjärjestelmään päätettiin toteutusvaiheessa lisätä vielä taulukon ja Tallenna-painikkeen alapuolelle esimerkki täytetystä taulukosta ja selityksiä teksteillä siitä, kuinka asetuksia tulisi käyttää. Esimerkin taulukkoa muokattiin Visual Studion työkaluilla siten, että sen ominaisuuksista asetettiin **Edit** pois päältä ja tämän taulun omalla init-tapahtumalla täytettiin taulun kentät taulukon 1 arvojen mukaisesti. Näin saatiin erilainen taulukko, jonka arvoja ei pystytä muuttamaan ollenkaan. Esimerkin yläpuolelle laitettiin valintaruutu, joka ohjelmoitiin oletusarvoisesti olemaan pois päältä. Valintaruutua painamalla esimerkin tekstit saadaan käyttöliittymässä esiin ja uudestaan painamalla piiloon. Tämän toteuttamiseen käytettiin .NET:n **checkbox-luokan** tapahtumaa **checked**, jossa suoritetaan ohjelmakoodia, kun valintaruutua painetaan.

4.3 Kanta-asiakkaat

Toiminnanohjausjärjestelmä Total ERP:n kantapäivitys-moduuli kehitti jo valmiiksi tietokantaan taulun Asiakas, jossa oli riveittäin asiakkaista yhteystietoja, laskutustietoja ja yms. SQL Management Studio -työkalulla kehitettiin SQL-kysely, joka lisää Asiakas-taulun rakenteeseen sarakkeen nimeltä LiittynytKantaAsiakkaaksi. Toimivaksi todettu SQL-kysely siirrettiin kantapäivitys-moduuliin. Uuden sarakkeen kentät ovat tietotyypiltään DATE-tietotyyppejä, jotka sisältävät ajan arvon päivämäärällä ja kellonajalla mikrosekunnin tarkkuudella. Oletusarvoisesti uudet kentät olivat tavallisten asiakkaiden kohdalla tietokannassa null-arvoisia, joka tarkoittaa arvoltaan tyhjää tietotyyppiä. Null-arvo pystytään syöttämään kaikenlaisen tyyppisiin tietotyyppeihin Microsoft SQL Server 2008 -tietokannassa. Null-arvojen ilmestymistä testattiin Tran-kyselyllä. Nämä kentät täytettiin kanta-asiakkaaksi muuttamisen hetkellä Transact-SQL-kielen funktiolla **Getdate()**, joka palauttaa käyttöön sql-palvelimen järjestelmän sen hetkisen ajan. (6.)

Kassassa toteutettiin asiakkaan muokkaaminen ja kanta-asiakkaaksi muuttaminen suunnitelmien mukaisesti. Kassa-verkkosovelluksen luokkaan nimeltä **Default** lisättiin tietojen muokkauksen DetailsView-elementtiin metodi, joka suorittaa ohjelmakoodia, kun selain lähettää DetailsView-elementin painikkeista komennon. Komentoja kehitettiin asiakkaan tietojen muutosten tallentamista ja peruuttamista varten, eli kahta painiketta varten. Tallenna-komentoon ohjelmoitiin sellainen SQL-kysely, joka käyttää DetailsView-elementin tekstikenttiin syötettyjä tietoja parametreina ja sitten muokkaa asiakkaan tietoja tietokannan taulussa Asiakas. DetailsView-elementin objektit tarkistettiin ohjelmakoodissa silmukassa, jossa selvitettiin oliko kyseessä tekstikenttä, jolloin kentän arvo lisättiin suoraan parametriksi. Kanta-asiakkaaksi liittymisen toiminnoksi luotiin valintaruutu ASP-elementti, jonka tila tutkittiin silloin kuin silmukan tarkistus vahvisti kyseessä olevan tekstikentän sijaan valintaruutu. Jos valintaruutu oli valittu, niin SQL-parametriksi laitettiin Transact-SQL-kielen Getdate()-funktio. Jos sitä ei ole valittuna, niin parametrin arvoksi laitettiin tietokannan null-arvo.

Asiakkaan valitseminen kassassa aiheutti **postback-toiminnon**, joka tarkoittaa selainnäkymän päivittämistä ja uusien tietojen hakua selainnäkymään palvelimelta. (7.) Valitun asiakkaan tunnistenumero tietokannasta tallentuu verkkosovelluksen tietoihin ASP-elementin **HiddenField** arvoksi. Tämän arvon perusteella Kassan muut toiminnot suorittavat asiakaskohtaisia toimintoja verkkosivun lataamisen aikana. Asiakkan tiedot näyttävä ASP:n GridView-elementti ohjelmoitiin hakemaan lisäksi tietoa kanta-asiakkaaksi liittymisestä SQL-kyselyllä käyttäjän asiakkaan valinnan postback-arvon perusteella. GridView muuttuu selaimessa HTML-kielessä taulukoksi. Näiden tietojen perusteella selvitettiin, että oliko valittu asiakas kanta-asiakas ja sen mukaan esitettiin suunnitelmien mukaisia huomiovärejä ja tekstejä.

Uuden asiakkaan luomisen käyttöliittymä toteutettiin peilaamalla samoja periaatteita kuin asiakkaan muokkaamisessa. Olemassa olevan asiakkaan sijaan tietokannan tauluun Asiakas luotiin uusi rivi kuvaamaan uuden asiakkaan tietoja. Tämä toteutettiin omalla SQL-kyselyllään ja DetailsView-elementillä samankaltaisilla menettelyillä.

Total ERP:n asiakaskäyttöliittymään lisättiin uusi Kanta-asiakastietojen välilehti. Käyttöliittymässä oli valmiina .NET-kehikon objekti **TabControl**, jonka rakennetta ohjattiin graafisella työkalulla Visual Studiossa. Objekti oli levitettynä asiakas näkymän käyttöliittymän ikkunan kokoisesti siksi, että välilehdellä voisi muuttaa koko ikkunan näkyvyyttä. Objekti sisältää ominaisuuden **TabPageCollection**, jossa on listattuna kaikki välilehdet, joita ohjelman ikkunassa on esillä. Uusi välilehti pystyttiin lisäämään graafisessa työkalussa Visual Studio -ohjelmassa tämän ominaisuuden avulla.

Uuden välilehden sisällöksi kehitettiin käyttöliittymää, jossa on erillinen näkymä tavalliselle asiakkaalle ja kanta-asiakkaalle. Tavalliselle asiakkaalle näkymänä on ainoastaan punainen teksti, jossa ilmoitetaan, että valittu asiakas ei ole kanta-asiakas. Punaisen tekstin vieressä on painike, joka muuttaa valitun asiakkaan kanta-asiakkaaksi suorittamalla painikkeen **click-tapahtumassa** SQL-kyselyn. Lopuksi click-tapahtumassa kutsutaan metodia, joka alustaa uudelleen kanta-asiakkuus-välilehden näkymän.

Samaa metodia kutsutaan myös, kun Asiakas-käyttöliittymä aukaistaan Total ERP:ssä. Metodin tarkoituksena on alustaa käyttöliittymä asiakaskohtaisilla tiedoilla. Jos valittu asiakas on kanta-asiakas tai hän on juuri muutettu kanta-asiakkaaksi, niin tarkistuksen jälkeen käyttöliittymän punainen teksti muutetaan vihreäksi, tekstissä ilmoitetaan asiakkaan olevan kanta-asiakas ja muutoksen tekevä painike piilotetaan käyttöliittymästä.

Uuden kanta-asiakkuus -välilehden sisältö ohjelmoitiin hakemalla valitun asiakkaan tiedot SQL-kyselyllä ja syöttämällä näitä tietoja käyttöliittymän tekstikenttiin. Välilehden sisältöön kuuluu aina yhden kanta-asiakkaan tietoja kuten ilmoitus kerääntyneiden kanta-asiakasbonusten nykyisestä summasta, tieto kanta-asiakkaaksi liittämisen hetkestä ja bonushistoria-tulosteen avaamisen käyttöliittymä yhtä asiakasta varten.

4.4 Kanta-asiakaskortti

Kassa-verkkosovellukseen lisättiin ASP-paneeli ylimmäksi toiminnoksi verkkosivussa, joka sisältää tekstiä, tekstikentän ja haku-painikkeen. Haku-painikkeen click-tapahtumaan ohjelmoitiin toiminto, joka hakee asiakkaan tunnistenumeron tietokannasta tekstikenttään syötetyn kanta-asiakaskortin numeron perusteella ja asettaa sen ASP-elementin HiddenField arvoksi. Click-tapahtumaan laitettiin tapahtumaan postback, että sivu päivittyisi niillä asiakkaan tiedoilla, jossa oli syötetyn kanta-asiakaskortin numero. Painikkeen painamisen tapahtumaan ohjelmoitiin myös ponnahduslaatikko-objekti, joka ilmestyy sitten, kun asiakasta ei löydy tietokannasta.

Haku-painikkeen painamisen tapahtumassa ohjelmoitiin tarkistus, joka tutki paneelin tekstikenttään syötettyjen merkkien pituuden. Merkkien pituuden ollessa 4, SQL-kysely ohjelmoitiin hakemaan tietoa kanta-asiakaskortin numeron perusteella ja jos se on 11 merkkiä pitkä, silloin tietoa ohjelmoitiin hakemaan henkilötunnuksen perusteella. Virheistä ilmoittava ponnahduslaatikko paneelissa ohjelmoitiin huomauttamaan myös väärästä tekstin pituudesta.

Verkkosovellus ohjelmoitiin laittamaan uuden paneelin tekstikenttä automaattisesti aktiiviseksi, kun verkkosivu on latautunut. Viivakoodinlukijan pitäisi itses-

sään toimia ilman erillistä ohjelmointia siten, että aktiivinen tekstikenttä täytyy tekstillä silloin, kun sitä käytetään. Tätä ei kuitenkaan testattu.

Kanta-asiakaskortin toimintoa varten täytyi ohjelmoida SQL-kysely kantapäivitys-moduuliin, jossa lisättiin asiakas-taulun tietoihin kanta-asiakaskortin numero ja henkilötunnus omina sarakkeinaan. Uuden asiakkaan lisäämisessä ja asiakkaan muokkaamisen DetailsView-elementteihin lisättiin tekstikentät uusia tietoja varten ja vanhoja asiakkaan tietoja käsittelevät SQL-kyselyt ohjelmoitiin käsittelemään myös uusia.

4.5 Bonusten tilitysfunktion ajastustoiminto

Ensimmäisenä toteutettiin tilityksen ajankohtaa koskevat ominaisuudet kuntoon lisäämällä asetus bonusten tilityspäivästä. Total ERP:n ohjelmavakioiden käyttöliittymässä oli yli 200 tietokantavakioiden lista. Lista on ohjelmoitu siten, että listan pituus on niin suuri kuin tietokannan taulussa **yVakio** on rivejä. Kantapäivitys-moduulissa lisättiin uusi yVakio-taulun rivi SQL-kyselyllä ja tietokantavakioiden käyttöliittymä päivittyi sen myötä yhtä pitemmäksi automaattisesti.

Tietokantavakioiden käyttöliittymä on Windows-ohjelman ikkunassa oleva taulukko. Siinä on kaksi saraketta ja ensimmäisessä sarakkeessa on riveittäin vakion selite ja toisessa arvo. SQL-kyselyssä laitettiin selitteeksi teksti: ”Kanta-asiakasbonusten tilityspäivä” ja arvoksi 15. Tiekantavakioiden käyttöliittymästä pystyy vaihtamaan minkä tahansa vakion arvoa, mutta oletuksena bonusten tilitys on aina aluksi 15 päivä. Arvon muuttaminen tapahtuu tekstikentän tekstin korvaamisella ja sen tapahtuman käsittelijä funktio suorittaa SQL-kyselyn, jossa uuden tekstin perusteella arvo muutetaan oikeassa kentässä taulussa yVakio.

Tilityspäivän asetuksen jälkeen aloitettiin uuden ajastustoiminnon kehittäminen. Kaikki ajastustoiminnot olivat ohjelmoitu VB.NET-ohjelmointikielellä luokkaan **ERP**. Olemassa olevat ajastustoiminnot olivat listattuna luokan lähdekoodissa toimintojen funktioina ja siihen listaan lisättiin bonusten tilityksen funktio. Tämän funktion suoritusta ajastusprosessi sitten tulee kutsumaan oletuksena, joka kuukauden 15 päivä. Tässä vaiheessa ajastustoiminnot käyttöliittymä jo näyttää

listalla uuden ajastustoiminnon, mutta mitään varsinaista toimintoa funktiolle ei ole vielä ohjelmoitu.

Uusi ajastettu tilitystoiminto kehitettiin siten, että ensiksi se tarkistaa onko tilityspäivän arvoksi laitettu asetuksissa numeroarvo. Jos vakion asetuksen arvo on tyhjä tai viallinen, niin ajastusprosessi ei suorita toimintoa ja kirjaa tapahtuneesta virheilmoituksen toiminnanohjausjärjestelmän tietokannan lokiin. Vakion arvon tarkastaminen tapahtuu VB.NET-ohjelmointikielen valmiilla **IsNumeric-** ja **IsNothing** -funktioilla, jotka palauttavat boolean arvon tosi tai epätosi. Funktion suoritus jatkuu tai keskeytyy boolean arvon perusteella, kun sitä verrataan SQL-kyselyn palauttamaan arvoon funktion ehtolausekkeessa.

Ajastustoiminto laskee vakion tarkistuksen jälkeen, että paljonko sekunteja on jäljellä seuraavan kuukauden tilityspäivään asti. Laskutoimitus tapahtuu siten, että tallennetaan päivämäärämuuttujaan arvo seuraavan kuukauden tilityspäivämäärästä ja asetetaan päivämäärien muuttujan ja meneillään olevan suorituspäivän erotus **Timespan-muuttujan** arvoksi, joka sisältää lasketun sekuntien lukumäärän. (8.) Timespan-muuttujaan lisättiin vielä kolmen tunnin edestä ylimääräisiä sekunteja, että ajastusprosessi ei vahingossa suorittaisi sekunteja kuvaavaa kokonaislukua tilityspäivää edeltävälle päivälle. Ajastustoiminto-funktio ohjelmoitiin palauttamaan ajastusprosessiin minuutteina sekuntien lukumäärä, joka ilmenee ajastustoiminnot-käyttöliittymässä toiminnon kohdassa kentässä intervalli.

Kanta-asiakasbonusten tilitystoiminto ohjelmoitiin tekemään vielä yksi asia ennen varsinaista bonusten tilitystä. Toiminto etsii SQL-kyselyllä taulusta AsiakasBonusHistoria yksittäistä päivämäärän arvoa ajalta ennen sen hetkisen kuukauden alkua. Tämä löydetty päivämäärä sijoitetaan yhdeksi parametriksi bonusten tilitysfunktioon, jota kaiken tämän valmistelun jälkeen lopulta kutsutaan. Jos AsiakasBonusHistoria-taulusta ei löydy mitään edellistä päiväystä esimerkiksi siksi, että taulu on vielä toistaiseksi tyhjä, niin bonusten tilityksen funktion parametriksi sijoitetaan siinä tapauksessa edellisen kuukauden 15 päivä arvoksi.

4.6 Bonusten tilitysfunktio

Bonusten tilitys funktiota kutsutaan luokassa ERP ajastustoiminnosta nimeltä kanta-asiakasbonusten tilitys. Funktiota on kutsuttava kahdella päivämäärä parametrilla, joista yksi on tilityspäivä ja toinen on edellisen kuukauden tilityspäivä. Tilitysfunktion rakenne toteutettiin siten, että sitä voi kutsua uudelleen saman kuukauden aikana monta kertaa tai menneiden kuukauden tilityspäiväyksen arvolla. Ideana on antaa mahdollisuus korjata tai muokata menneitä tilityksiä manuaalisesti, jos se on syystä tai toisesta tarpeen. Muokkaus mahdollisuuden ansiosta tilitetyt bonukset korvataan toisella tilityksellä määrätyn kuukauden ajankohdalle. Näin vältytään tuplatilityksiltä yhtä kuukautta kohden, vaikka tilitysfunktiota suoritettaisiinkin useasti samalta ajankohdalta.

Tilitysfunktio tehtiin siten, että se tallentaa jokaisesta tilityksestä rivin tauluun **AsiakasBonusHistoria**. Rivissä on tunnistenumero, jolla viitataan asiakas taulun tunnisteeseen, että saadaan oikea asiakas yhdistettyä bonus historian riviin, kun tietokantoja tulkitaan. Bonus historian tapahtumariveissä on asiakkaan viitteen lisäksi tieto tapahtuma päiväyksestä ja myös viimeisimmästä muokkauspäivämäärästä. Samalle riville tallentuu myös alkuperäisen tilityksen suorittanut käyttäjä viite tietokantaan ja myös muokkauksen suorittaneen käyttäjän viite. Näin toimintojen käyttäjiä voidaan seurata ja pysytään ajan tasalla suoritetuista tilityksistä.

Tilityksen suorittaminen alkaa funktiossa siten, että ensin rajataan edellisen kuukauden voimassa olleen kanta-asiakuuden ajalta ostokset. Tämä kehitettiin testaamalla ja kehittämällä oikeanlainen SQL-kysely ensin SQL Management Studion Tran-kyselyillä. SQL-kyselyn täytyy hakea tietokannan taulusta **VK_TapahtumaRaha** kaikki maksetuksi todetut kaupat, jotka on suorittanut sellainen asiakas, joka on liittynyt kanta-asiakkaaksi. Taulussa VK_TapahtumaRaha on viite kaupan toimitukseen tietokannassa, jossa taas on lopulta viite asiakkaaseen. SQL-kysely piti saada sellaiseksi todettua, että se haki pelkästään sellaiset maksetut tapahtumarahat, joiden päivämäärät ovat määrätyltä kuukaudelta ja sen ajan jälkeen, kun asiakas on liittynyt kanta-asiakkaaksi.

Tilitysfunktio hakee SQL-kyselyllä myös palautukset, jotka ovat negatiivisia maksutapahtumia tietokannassa ja yhdistää sen jälkeen kyselyn palautuksen ostoksien taulukkoon. Myös bonuskertymät palautetaan SQL-kyselystä taulukkoon. Kyseiset taulukot ovat tilitysfunktion lähdekoodissa VB.NET-ohjelmointikielen **Datatable-objekteja**. Tilitettävien maksujen rivit yhdistettiin tauluissa DataTable-objektin tarjoamaa **Merge-metodia** käyttämällä.

Palautuksien SQL-kyselyn aikarajaus ei ole edellinen kuukauden alku vaan edellisen kuukauden tilityspäivästä tilityspäivää edeltävään päivään asti. Eri aikarajaus palautuksissa laitettiin sitä varten, että tilityspäivän kuukauden aikana tulleet palautukset pystyttiin poistamaan edellisen kuukauden bonuskertymän ostoksista, joita hyvitettiin. Lisäksi palautuksessa huomioitiin 2 muutakin asiaa. Hyvitettävän tuotteen osto ajan oli oltava korkeintaan 30 päivää sitten ja sellaiselta ajalta jolloin asiakas oli liittynyt kanta-asiakkaaksi. Jos molemmat ehdot eivät funktiossa toteudu palautuksen käsittelyssä SQL-kyselyssä, niin silloin siitä seuraavaa bonuskatoa ei tilitetä.

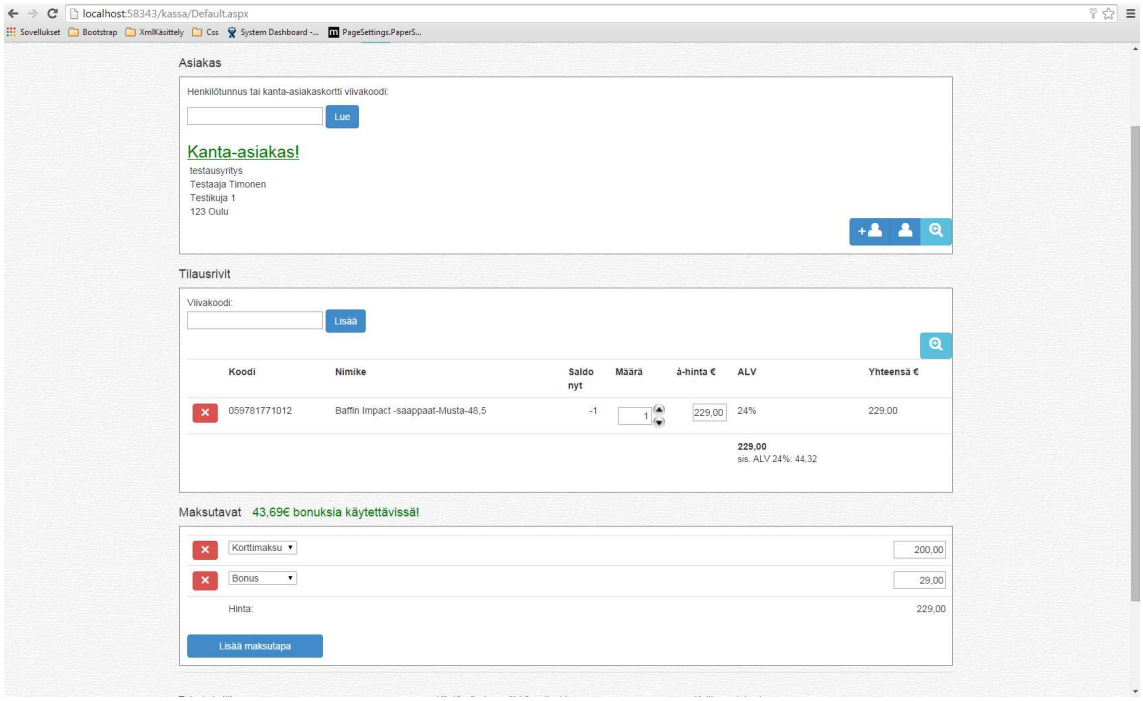
Taulussa VK_TapahtumaRaha oli olemassa sarake nimeltä **HyvitettavaTapRahaID**, johon laitetaan palautuksen ID-numero, joka on saman taulun hyvitettävän ostoksen maksetun tapahtumarahan ID-numero. Tässä vaiheessa tilitysfunktion toteutusta huomattiin, että Kassa-sovelluksessa suoritetuissa palautusmaksuissa ei tallennu hyvitettävän maksun ID-numeroa ollenkaan, kun tallennetaan palautus tietokantaan. Tämä oli Kassassa selvä vika, mutta ominaisuutta ei aloitettu korjaamaan vielä bonusjärjestelmän kehityksen aikana. Tilitys kuitenkin todettiin toimivaksi Visual Studion virheiden jäljittäjä -työkalulla suunnitelmien mukaan toimivaksi, laittamalla hyvitettävien maksujen ID-numeroita ohjelman suorituksen aikana SQL-kyselyn arvoksi.

Kanta-asiakasbonusten tilittäminen kehitettiin ohjelmakoodissa silmukassa. Silmukka käyttää kertymien ja palautumisen yhdistettyä taulukko-objektia ja käy jokaisen taulukon rivin läpi ja tilittää rivin tietojen perusteella SQL-kyselyllä bonukset tietokantaan taulukkoon AsiakasBonusHistoria. .NET 4.0 -kehikon taulukko-objekti sisältää viitenumeron asiakkaasta tietokannan Asiakas-tauluun ja ostosummaan tai palautussummaan. Ostosumma tai palautussumma annettiin

tilitysfunktiossa parametriksi aliohjelmalle, joka suorittaa varsinaisen yksittäisen bonuksen laskemisen hakemalla SQL-kyselyllä tietokannasta järjestelmän bonusasetukset. Aliohjelma palauttaa lasketun bonuksen silmukassa desimaalimuuttujaan, joka laitetaan silmukassa talteen muiden tallennettavien tietojen kanssa. Lopulta silmukassa sijoitetaan kaikki bonus tilityksen tiedot muuttujista SQL-kyselyn arvoksi, jossa tapahtuma lisätään bonushistoriaan tietokantaan. Bonushistorian tiedot riittävät kaiken muun toiminnan rakentamiseen bonusjärjestelmässä.

4.7 Bonukset maksutapana

Total ERP sisältää kantapäivitys-moduulin haaran, joka on tarkoitettu ainoastaan Magento Edition toiminnanohjausjärjestelmään. Tämä haaroitus sisältää tietokantarakenteita, joita tarvitaan ainoastaan verkkokauppa toiminnassa ja siksi se on erotettu kantapäivitys-moduulin perusrungosta, joka on käytössä myös toisen mallisissa Total ERP -toiminnanohjausjärjestelmissä. Tänne haaraan lisättiin SQL-kysely, joka lisää tauluun **VKz_Maksutapa** rivin bonus maksuja varten. SQL-kyselyssä lisättiin myös oletusarvo **tosi** kenttään **”Käytä kassassa”**. Oletusarvon tarkoitus on helpottaa Kassa-verkkosovelluksen rakennetta, jonka käyttöliittymässä (kuva 7) on tiputusvalikko maksutavan valintaa varten. Tiputusvalikon sisältö täytettiin SQL-kyselyn avulla. Kyselyssä haetaan maksutapojen taulukosta rivit, joissa kenttä ”Käytä kassassa**”** on arvoltaan tosi. SQL-kyselyn palauttaman taulukon kenttä **Selite** liitetään lähdekoodissa tiputusvalikon kenttiin. Valitussa kentässä on tiputusvalikossa myös piilotettu arvo **ID**, jolla viitataan maksutavan tunnistenumeroon. Tunnistenumeroa tarvitaan parametrina Kassan funktiossa, joka suorittaa sovelluksen maksutapahtuman.



KUVA 7. Kassan maksutiedot

Kassan Maksutapahtumat-käyttöliittymässä lisättiin vihreä teksti-elementti, johon ohjelmoitiin ylälaitaan näkymään valitun kanta-asiakkaan bonushistorian summa SQL-kyselyllä Transact-SQL:n **Sum-funktiota** käyttämällä. Sum-funktio summaa yhteen kaikki asiakkaan bonuskertymät ja ostot bonusrahoilla. Tämä summa on asiakkaan käytössä olevien bonusten lukumäärä.

Kassa maksutapojen rakennetta testattiin lisäämällä teennäisiä bonuksia testikantaan kanta-asiakkaille bonushistoriaan ajamalla SQL-kyselyjä SQL Management Studiossa. Näin saatiin vahvistettua, että bonusten maksutapojen näkymä oli ulkoisesti kunnossa. Bonus-maksutavan **ID-numeroa** tarvittiin tiputusvalikosta toimintoon, jossa bonukset piilotettiin tiputusvalikosta, jos valittu asiakas ei ollut kanta-asiakas. Tiputusvalikon piilottaminen tapahtui Kassan lähdekoodissa ohjelmoimalla tarkistus, jossa kutsutaan tiputusvalikko-objektin kenttien arvoja. **Näkyvyys-arvo** asetettiin lähdekoodissa arvoon epätosi, jos käytettävien bonusten summa oli 0€.

Bonuksen käyttö maksutapana tarvitsi kuitenkin vielä lisää erityiskäsittelyä. Bonushistoriaan tarvittiin merkintä bonuksien käytöstä maksutapana, joka olisi taulukon AsiakasBonusHistoria:ssa sarakkeessa **Tyyppi** arvoltaan 1. Tämän avulla

kassassa näkyvä bonusten käytettävissä oleva summa myös vähenisi. Ilman tätä ominaisuutta bonuksilla pystyisi jo maksamaan, mutta käytettävät bonukset eivät koskaan vähenisi. Kassaan ohjelmoitiin myös erillinen funktio, joka tarkistaa käytettävissä olevan bonuksen katteen maksutapana Kassan vahvistuspainikkeen tapahtuman käsittelijän funktiossa ja keskeyttää maksutapahtuman suorittamisen, jos kate ei riitä ja antaa siitä palautetta sovelluksen käyttäjälle ponnahdusikkunan avulla.

Maksutapahtuman suorittamista kutsuttiin Kassan lähdekoodissa luokasta VK_Yle, jossa on funktio **LisaaVK_Tapahtumaraha.** Funktio otti jo valmiiksi tapahtumarahan parametreina mm. maksutavan tunnistenumeron, jonka avulla bonusmaksutapa saatiin tunnistettua. Tapahtumarahan lisäämisen funktioon ohjelmoitiin tarkistus, jossa tutkitaan onko käytetty maksutapa bonus ja vahvistamisen myötä kutsutaan funktiota, joka ennen varsinaista verkkokaupan maksutapahtuman käsittelyä laittaa SQL-kyselyllä käyttösumman ja tapahtumarivin bonushistoria-tauluun tietokannassa.

4.8 Bonushistoria-tuloste

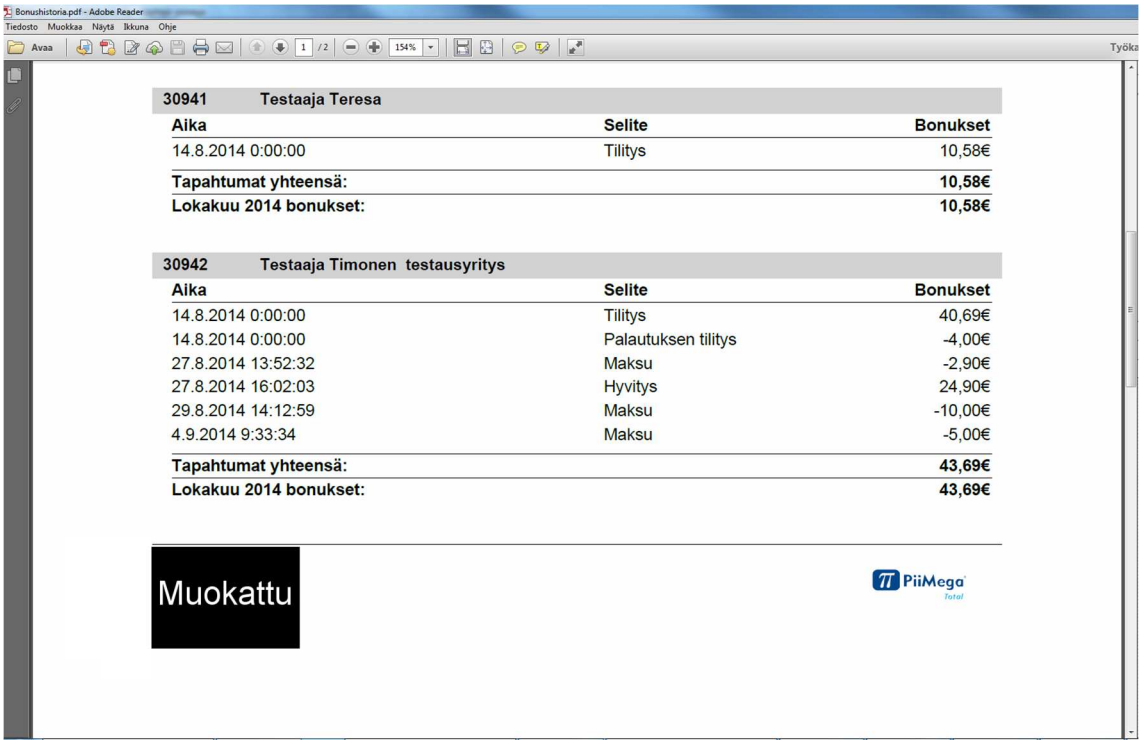
Bonushistoria-tulosteen luomiseen käytettiin Visual Studiossa suunnittelutyökalua. Ensiksi luotiin **ActiveReport-tiedosto** ohjelmointiprojektin tietoihin ja aseteltiin työkalun avulla pdf-näkymä, joka esittää yksittäisen objektin otsikon harmaalla taustalla ja sen alapuolella esittelee lisätietoja taulukon muodossa. Objektina tulosteessa toimii kanta-asiakas ja sen alapuolella on taulukon muotoisessa tekstikentissä bonushistorian asiakaskohtaiset tiedot.

Suunnittelutyökalulla lisättiin **ryhmäotsikointi-objekti**, johon tulosteen VB.NET-lähdekoodissa sidottiin SQL-kyselyssä kanta-asiakas. Tulosteen avaamisella on ohjelmassa tapahtumankäsittelijä-funktio, jossa tulosteeseen tarvittavat tiedot haettiin tulosteen objekteihin SQL-kyselyillä. Active Report tulosteiden rakenteisiin kuuluu vakiona suunnittelutyökalussa **Detail-rivi**, johon sidottiin ryhmäotsikointi-objektin asiakkaan tunnistenumeron avulla tietoja riveittäin SQL-kyselyllä. SQL-kyselyjen palauttamat rivit sidotaan Detail-rivin tapahtuman käsittelijä-funktiossa tulosteen lähdekoodissa siten, että Detail-rivit tulostuvat tulosteeseen niin useasti kuin löydettyjen tietojen esittämiseen on tarpeen.

Bonushistoria tuloste (kuva 8) esittää näiden menettelyiden avulla taulukon tekstikentissä asiakkaan tiedoista tilityksen, palautuksen tilityksen, hyvityksen tai maksun selitteenä, päiväyksen ajan päivämäärän ja sekunnin tarkkuudella, sekä bonussumman.

ActiveReport-tulosteen lähdekoodi koostuu omasta luokasta **ArBonusHistoria**, johon kehitettiin tulosteen tapahtumienkäsittelijäfunktioita suunnittelutyökalulla rakennettuihin objekteihin. Tulosteen luokan alustusfunktiota ohjelmoitiin siten, että tulostetta pystyttiin kutsumaan muualta Total ERP:stä kahdella ajan parametrilla. Parametrien ajat ovat tulosteen alku ja loppupäivämäärä, joita tarvitaan tulosteen runsaiden tietojen rajaamiseen ja menneen ajan bonustilin summan suuruuden tutkimiseen.

Tulosteessa on ryhmäotsikon ja detail-rivien jälkeen lopussa vielä details-rivien tapahtumien bonusten summa ja rajatun aikavälin bonusten summaus omana yksittäisinä riveinä. Rajatun aikavälin summalla esitetään menneen ajan bonushistorian kerääntynyt summa ilman kaikkia menneitä tapahtumia aikojen alusta asti. Tämä summa saatiin esitettyä tulosteessa SQL-kyselyn ehtolausekkeiden avulla. Bonushistorian tapahtumien summa laitettiin menneiden bonuskertymien yläpuolelle objektin loppuun. Tapahtumien summat laskettiin detail-rivin tapahtumankäsittelijäfunktiossa, jonka summa tallennetaan muuttujaan talteen ja lasketaan yhteen vanhan arvon kanssa aina, kun uusi detail-rivi muodostuu tulosteeseen. Lopulta rivien yhteen laskettu summa otettiin muuttujasta ja siirrettiin rivien jälkeen loppuun tekstikentän arvoksi omassa tapahtumankäsittelijäfunktiossaan.

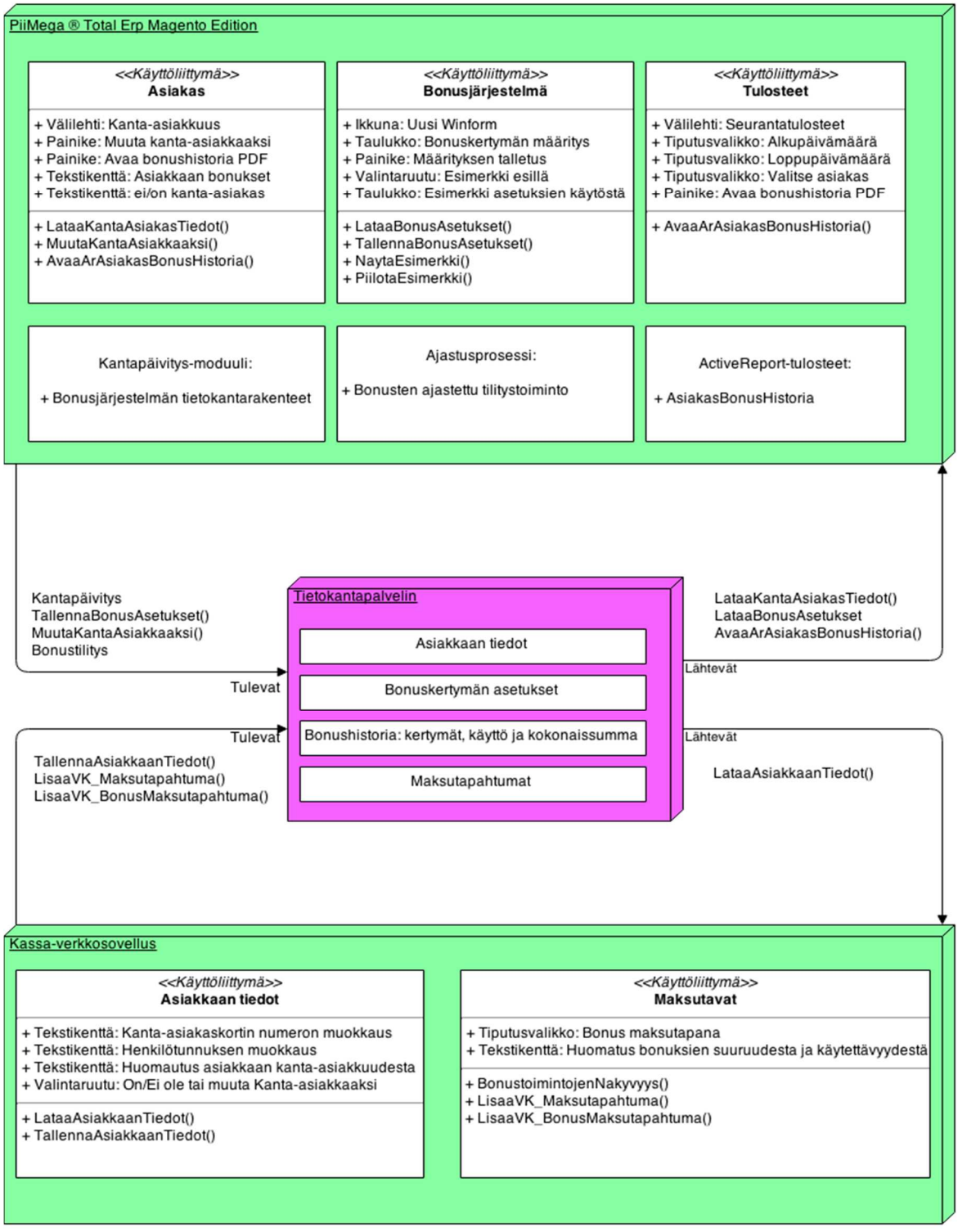


KUVA 8. Bonushistorian tuloste

4.9 Rakennekaavio

Opinnäytetyössä valmiiksi kehitetyistä ohjelmistokomponenteista tehtiin rakennekaavio (kuva 9). Kaaviota tutkimalla ilmenee bonusjärjestelmän komponenttien sijainti, vuorovaikutus, funktiot ja tiedonsiirto tietokantapalvelimen, Total ERP:n ja Kassan välillä.

Bonusjärjestelmän rakennekaavio



KUVA 9. Rakennekaavio

5 YHTEENVETO

Bonusjärjestelmän tekemisessä oli tavoitteena saada ohjelmistokomponentti, joka toteuttaa työn tilanneen yrityksen ja yrityksen asiakkaiden määrittämät vaatimukset. Työssä tehtiin ohjelmistokehitystä suunnittelu- ja tuotekehitystyönä. Työssä onnistuttiin saamaan toimiva kokonaisuus kanta-asiakasbonusten kertymistä ja käyttöä varten, sekä kehitettiin bonuksiin liittyviä toimintoja, jotka ovat tarpeellisia verkkokauppamyymälää ja toiminnanohjausjärjestelmää ylläpitävälle ohjelman käyttäjälle.

Opinnäytetyön onnistuneesta tuloksesta huolimatta ohjelmistokomponenttia ei otettu vielä tällaisena versiona laajempaan käyttöön. Bonusjärjestelmä siirrettiin testijärjestelmään käyttöön asiakkaalle. Sieltä se voi vielä palata jatkokäsittelyyn, jos puutteita tai jatkokehitettävää ilmenee. Mitään varsinaista vikaa uudessa komponentissa ei vaikuta opinnäytetyön loppuvaiheessa kuitenkaan olevan.

LÄHTEET

1. PiiMega Total ERP. Piimega. Saatavissa: http://www.piimega.fi/tuotteet/toiminnanohjausjarjestelma/piimega-total-erp. Hakupäivä 16.1.2015

2. Shepherd, George 2010. Microsoft ASP.NET 4 Step by Step. Washington. Microsoft Press. Saatavissa: https://www.microsoft.com/learning/en-us/book.aspx?id=13834. Hakupäivä 6.4.2015

3. Ben-Gan, Itzik 2008. Compund Assignment Operators. Microsoft TechNet. Saatavissa: http://technet.microsoft.com/en-us/library/cc721270(v=SQL.100).aspx. Hakupäivä 5.1.2015

4. .NET Framework 4. Visual Studio. Saatavissa: http://msdn.microsoft.com/en-us/library/vstudio/w0x726c2(v=vs.100).aspx. Hakupäivä 4.1.2015.

5. Methods. Microsoft Developer Network. Saatavissa: http://msdn.microsoft.com/en-us/library/system.data.datatable(v=vs.100).aspx. Hakupäivä 16.1.2015

6. GETDATE (Transact-SQL). Microsoft Developer Network. Saatavissa: http://msdn.microsoft.com/en-us/library/ms188383.aspx. Hakupäivä 16.1.2015

7. What is IsPostBack. Net-informations.com. Saatavissa: http://net-informations.com/faq/asp/ispostback.htm. Hakupäivä 16.1.2015

8. TimeSpan Structure. Microsoft Developer Network. Saatavissa: http://msdn.microsoft.com/en-us/library/system.timespan%28v=vs.110%29.aspx. Hakupäivä 16.1.2015

LIITTEET

Liite 1 Bonusjärjestelmän määrittelydokumentti

Liite 2 Kanta-asiakaskortti

Liite 3 Bonustilityksen Excel-suunnitteludokumentti

Liite 4 Bonusasetuksien suunnitteludokumentti

Bonusjärjestelmän määrittelydokumentti

Esimerkki bonusjärjestelmästä:

Tilaussumma kk aikana (MAX)	Bonus %	Bonus euroina
0- 99 €	0 %	0 €
100 -199 €	1 %	1,99 €
200 - 299 €	2 %	5,98 €
300 - 399 €	3 %	11,97 €
400 - 499 €	4 %	19,96 €
500 - > ääretön €	5 %	50 € (1000 € ostoista kk)

Ostokset nollautuvat kk lopussa (tai määriteltyinä päivinä). Bonukset lasketaan kuukauden viidentenätoista päivänä.

Esimerkki rahan siirtämisen asiakkaalle:

Ostokausi	Tilityspäivä (asetuksista muutettavissa oleva arvo)
1.1-30.1.2014	15.2.2014
1.2-28.2.2014	15.3.2014
1.3-30.3.2014	15.4.2014

Kassalla:

Kun kanta-asiakas kortti vedetään, kassa ilmoittaa selkeästi, että asiakkaalla on käytettävissä bonuksia, jolloin nappia painamalla voi käyttää kentässä olevan bonus summan, taikka kentän arvoa voi vaihtaa (esimerkiksi asiakkaalla kertynyt 10€ bonuksia, ja 5€ haluaa käyttää, niin kentän arvo vaihdetaan 5€ jonka jälkeen ok painikkeella hyväksytään bonus).

Kun ostos hyväksytään niin ERP käy tarkistamasta Magentosta ajantasaisin saldon ja vähentää saldon oikeaksi.

Bonushistoria tärkeä, jolla voidaan seurata bonusten kertymistä !

Asiakaspalautukset kirjataan miinus myyntinä ks. asiakkaan taakse, joka vähentää bonuksia (ei kuitenkaan vanhoja bonuksia vaan vielä laskemattomia bonuksia -> koska myynti ks. asiakkaalla laskee).

Kanta-asiakaskortti

Asiakas syöttää oman korttinsa numeron asiakastilillensä Magentosta.Tällä numerolla-myynnit kirjautuvat oikealle asiakkaalle. Asiakas voi syöttää korttinsa vain kerran ja sitä ei voi muokata. Muokkaus tapahtuu Metsolalla admin paneelista asiakastunnuksen takaa.

Myynnit kirjautuvat ks. asiakkaalle kassalla, kun hän käyttää kanta-asiakaskorttia tai verkkokaupassa kun hän tilaa omilla tunnuksilla verkkokaupan kautta.

Kanta-asiakaskorttina voidaan käyttää myös ajo-korttia (tai muuta välinettä jossa on henkilötunnus viivakoodissa), josta luetaan asiakkaan henkilötunnus. Henkilötunnus tallennetaan salattuna tietokantaan.

Bonustilityksen Excel-suunnitteludokumentti

AsiakasBonusHistoria	
* Id	
* AsiakasId	Kenen asiakkaan rivi
* Paivays	Arvopvm, milloin laskettu tai käytetty. (Milloin bonus on käytettävissä eli kesäkuu 2014 bonus, tämä pvm=15.7.2014)
* Summa	+Merkkinen on kertynyttä bonusta, - merkkinen on käytettyä bonusta
* Tyyppi	0=Bonuskertymää, 1=Bonuksen käyttöä
* LuontiPvm	
* LuontiKayttajaId	
* MuokattuPvm	
* MuokkausKayttajaId	
TapahtumaRaha-tauluun uusi kenttä	
* AsiakasBonusHistoriaId	Kun bonus lasketaan, tähän tallennetaan sen AsiakasBonusHistoria-taulun id, johon kyseinen erä laskettiin
Laskentafunktio toiminta-periaate, kutsutaan kuukaudella ja vuodella	
-Loopataan kaikki asiakkaat, joilla on yksikin TapahtumaRaha-tietue olemassa (+ merkkiset 1.6-30.6 ja - merkkinen 15.6 - 14.7)	
- Lasketaan TapahtumaRaha yhteensä edellisten asetusten mukaisesti	
- Tutkitaan onko asiakkaalle olemassa jo rivi AsiakasBonusHistoria-taulussa.	
	Jos on --> muokataan edellistä, muuttuu summa, Muokattupvm, muokkausKayttajaId
	Jos ei ole, muodostetaan uusi rivi
Kun Bonusta käytetään, TapahtumaRahan tallennuksen yhteydessä talletetaan rivi myös AsiakasBonusHistoria tauluun	
Asiakkaan bonus-kertymä lasketaan aina kassalla AsiakasBonusHistoria-taulusta. Kaikki rivit AsiakasId:llä sum(summa)	

Bonusasetuksien suunnitteludokumentti

Luodaan taulu BonusAsetukset. Taulussa on vaadittavia kenttiä ainakin OstoSummaMin ja BonusProsentti. Jokaisella rivillä taulussa on eri arvoinen bonusprosentti, joka on arvoltaan decimaali:

0.00 = 0%

0.01 = 1%

0.02 = 2%

0.03 = 3%

Ideaali tilanne on se, että varsinaista bonusprosenttia ei muutettaisi, mutta saman rivin toiset kentät olisivat muutettavissa jostain toiminnanohjaus käyttöliittymästä.

esim asetuksissa on valittavissa YHDEN PROSENTIN kohdalla, että ostoksista saa yhden prosentin bonusta, kun OstoSummaMin on 100€. (Tietokannan kenttä on tyyppiä decimal 19,4).

Asetukset on rajattava lineaariseksi. Eli täytyy varmistaa että kahden prosentin OstoSummaMin ei voi olla pienempi kuin yhden prosentin OstoSummaMin.

Ajastetussa funktiossa lasketaan bonukset siten, että käytetään olemassa olevaa ostosummaa kuukaudelta ja kysytään SQL -kyselyllä KorkeinProsentti taulusta BonusAsetukset. Seuraava kysely palauttaa 2% bonukset

```
SELECT MAX(BonusProsentti) As KorkeinProsentti
FROM BonusAsetukset
WHERE OstoSummaMIN <= 200
```

Seuraava kysely palauttaa yhden prosentin bonukset

```
SELECT MAX(BonusProsentti) As KorkeinProsentti
FROM BonusAsetukset
WHERE OstoSummaMIN <= 199
```