Sami Halme

TUOTETIETOJEN HALLINTASOVELLUKSEN TOTEUTUS

Tietojenkäsittelyn koulutusohjelma

2013

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

TUOTETIETOJEN HALLINTASOVELLUKSEN TOTEUTUS

Halme, Sami
Satakunnan ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Huhtikuu 2013
Ohjaaja: Nieminen, Hans
Sivumäärä: 34
Liitteitä: 0

Asiasanat: tietojenkäsittely, tuotetiedot, järjestelmäsuunnittelu, projektinhallinta

---

Opinnäytetyön aiheena oli tuotetietojen hallintasovelluksen toteuttaminen. Työ toteutettiin Kankaanpääläiselle ICT-alan yritykselle, Satakunnan Verkkotekniikka Oy:lle. Opinnäytetyön alussa kuvaan asiakasyrityksen toimintaa ja tulevan järjestelmän tarpeita ja vaatimuksia, jonka jälkeen kuvaan yleisesti järjestelmäprojektien määrittelytyötä. Tämän jälkeen kuvaan projektissa käytetyitä tekniikoita ja itse projektin kulkua. Lopuksi kuvailen projektissa syntyneen järjestelmän.

IMPLEMENTATION OF PRODUCT INFORMATION MANAGEMENT APPLICATION

Halme, Sami
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in data processing
April 2013
Supervisor: Nieminen, Hans
Number of pages: 34
Appendices: 0

Keywords: data processing, product information, system design, project management

---

The purpose of this thesis was to implement a product information management application. The project was done for a customer company based in the city of Kankaanpää, Satakunnan Verkkotekniikka Oy. In the beginning of the thesis I will describe operations of the customer company and the needs of the upcoming application, and after that I will describe work in requirement specifications in general. Then I will describe the used techniques and the work process of the project. In the end I will describe the system which was born as a product of this project.

SISÄLLYS

1 JOHDANTO

Idean opinnäytetyölle antoi Erkki Vaaja, Satakunnan Verkkotekniikka Oy:n tekninen johtaja. Satakunnan Verkkotekniikka Oy, Kankaanpääläinen ICT-alan yritys ja entinen työharjoittelupaikkani, pyysi keväällä 2012 minua valmistamaan heidän käyttöönsä web-järjestelmän avustamaan toimintansa logistiikkaa. Satakunnan Verkkotekniikka Oy tarjoaa monipuolisia verkkoasennus ja –ylläpitopalveluita, työasema- ja palvelinylläpitoa, sekä ylläpitää ja myy monia erilaisia laitteita pohjois-satakunnan alueen yrityksille. Tulevaisuudessa, yritys laajentaa laitteiden myyntitoimintaa aloittamalla verkkokaupan, josta asiakkaat voivat suoraan ostaa tuotteita ja palveluita. Tämän toiminnan laajeneminen ja yrityksen nykyinenkin toiminta aiheuttaa kasvavaa tarvetta luotettavalle logistiselle tiedolle.

Näitä vanhoja ja uusia tietotarpeita varten päätettiin luoda web-järjestelmä, jonka avulla voitaisiin seurata tuotetietoja. Tämä tarkoittaisi tehtävänä siis järjestelmän määrittelyä, suunnittelua, ohjelmointityötä ja lopulta järjestelmän asennusta asiakkaan tiloihin. Työ toteutettaisiin projektina.

Tässä opinnäytetyössä pyrin kuvaamaan projektin toteutumisen kulun ja myös ongelmia joita projekti kohtasi, yleisen projektityöskentelyn näkökulmasta. Kuvaan myös valmiin järjestelmän ja siinä käytettyjä tekniikoita.

2 JÄRJESTELMÄN KUVAUS

2.1 Tilaajayritys

Satakunnan Verkkotekniikka Oy on tarjonnut Pohjois-Satakunnan alueen yrityksille monipuolisia ICT-palveluja jo vuodesta 2006. Satakunnan Verkkotekniikka Oy tarjoaa verkkoasennus ja –ylläpitopalveluita, työasema- ja palvelinylläpitoa, sekä ylläpitää ja myy monia erilaisia laitteita pohjois-satakunnan alueen yrityksille. Sen suurimpia asiakkaita ovat Kankaanpään Kaupunki ja Pohjois-Satakunnan

Peruspalvelu-Liikelaitoskuntayhtymä. Sen palveluihin kuuluu muun muassa yrityksen tietoverkkoihin liittyvät palvelut aina suunnittelusta rakentamiseen ja ylläpitoon, palvelimien kartoitus, suunnittelu ja ylläpito sekä työasemien tarvekartoitus, hankinta ja ylläpito.

2.2 Järjestelmän tarkoitus

SVT myy, ylläpitää, ja varastoi monia erilaisia koneita, laitteita ja tuotteita eri asiakasyrityksissä ja organisaatioissa Kankaanpään ja muun Pohjois-Satakunnan alueella. Ylläpidon, myynnin ja varastoinnin helpottamiseksi SVT tarvitsi järjestelmän, jonka avulla voi kerätä, seurata ja päivittää näiden tuotteiden tietoja. SVT on esimerkiksi saattanut myydä kytkinlaitteen asiakasorganisaatiolle joka sijaitsee jossain Pohjois-Satakunnassa. Tällöin täytyy tietää muun muassa:

- Laitteen omistaja (asiakas)
- Laitteen sijainti ylläpitoa varten (esim. asiakkaan toimipisteen 2. kerroksen kahvihuoneen kytkinkaappi)
- Laitteen EAN-koodi, tuotekoodi, sarjanumero
- Laitteen takuutiedot (rikkoutunut laite saadaan sujuvasti takuuhuoltoon/korjaukseen/vaihdettua uuteen)
- Onko varastossa varalaitetta rikkoutuneen tilalle
- Muita ylläpitoa helpottavia tietoja.

Järjestelmän tarkoitus on täyttää nämä ja muita tietotarpeita. Sitä tullaan käyttämään yrityksen sisäverkosta, myös viivakoodinlukijoita hyödyntäen. Tietokanta pitää sisällään muun muassa tuotetietoja, tuoterivitietoja, asiakastietoja ja toimittajatietoja. Järjestelmässä panostin erityisesti toimivaan, sujuvaan ja selkeään käyttöliittymään, jonka olen huomannut erittäin tärkeäksi toimiessani samantapaisten järjestelmien kanssa aiemmissa työharjoitteluissani.

2.3 Lähtökohdat

Ennen järjestelmän asennusta tuotantoon, yritys piti myymiensä ja varastoimiensa laitteiden tietoja yksinkertaisissa Excel-taulukoissa. Tämä toimii pienessä mittakaavassa, mutta ei ole tarpeeksi sujuvaa ja tehokasta, erityisesti toiminnan laajentuessa. Tietomäärän kasvaessa täytyy työkalu jolla pystyy hallitsemaan suuriakin määriä erityyppisiä tietoja.

2.4 Järjestelmän toiminnot ja ominaisuudet

Järjestelmän tärkeimmät ominaisuudet ovat saapuneiden tuotteiden kirjaaminen järjestelmään, lähtevien tuotteiden merkitseminen pois varastosta ja uuteen sijaintiin, ja tuotteiden tietojen seuranta sekä päivitys.

3 MÄÄRITTELY JA SUUNNITTELU

3.1 Mitä on vaatimusmäärittely ja suunnittelu?

Vaatimusmäärittely on yksi ohjelmistojärjestelmien kehitystyön perustehtävistä. Se on jossakin muodossa läsnä lähes kaikissa ei-triviaaleja ohjelmistojärjestelmiä toteuttavissa prosesseissa ja –projekteissa. Ohjelmistojen vaatimusmäärittelyä on tehty yhtä kauan kuin on tehty ohjelmistoja, mutta tieteenä sitä on tehty 1980-luvun puolivälistä ja kurinalaisesti 1990-luvun alkupuolelta lähtien. (Paakki, 2011.)

Vaatimusmäärittelyssä kartoitetaan, arvioidaan, määritellään, dokumentoidaan, analysoidaan ja muutetaan järjestelmään kohdistuvia tavoitteita ja oletuksia sekä sen toiminnallisuutta, laatuominaisuuksia ja rajoituksia. Tämä määritelmä voidaan tiivistää yksinkertaisemmin niin, että vaatimusmäärittelyssä siis selvitetään, mitä järjestelmältä vaaditaan ja miten löydetyt vaatimukset saadaan kuvatuksi jatkokehitykseen soveltuvalla tavalla. (Paakki, 2011.)

Nykyisin puuttuvat ja virheelliset vaatimukset ovat yksi suurimmista projektien epäonnistumisen syistä. Vaatimusten kartoitus, yhteensovittaminen ja tulkitseminen puhtaasti epäformaalista luonnollisesta kielestä äärimmäisen formaaliksi ohjelmakoodiksi on hyvin vaikeaa. Lisäksi modernit ohjelmistojärjestelmät ovat äärimmäisen monimutkaisia. Vaatimuksia ja sidosryhmiä on paljon, ja vaatimuksilla on huomattavasti keskinäisiä riippuvuuksia. (Paakki, 2011.)

Määrittelytyön lopputuloksena syntyy vaatimusmäärittelydokumentti, joka sisältää seuraavat asiat:

- Palvelun yleiskuvaus: tarkoitus, miksi tehdään, tavoitteet ja ominaisuudet, palvelun käyttäjät ja järjestelmän toteutustapa.
- Palvelun rakenne ja toiminnallisuus: sisältöhierarkia, sisältöosioiden kuvaus, käyttöliittymän perusajatus, näyttökuvaukset.
- Tietorakenteet: käsitemäärittely ja tietorakenteiden kuvaus, tietovarastojen sisältö, metatiedot.
- Teknisten ratkaisujen kuvaukset: laitteisto, palvelinohjelmistot, tietoliikenneratkaisut, liittymät muihin järjestelmiin, muut ominaisuudet.
- Projektin rajaus: hylätyt ratkaisuehdotukset, jatkokehitysideat, määrittelemättä jääneet asiat.

(Sininen Meteoriitti, haettu 15.4.2013.)

Vaatimusmäärittelyn työvaiheet

Vaatimusmäärittely alkaa siitä, että pyritään ymmärtämään ongelmakenttä. Tässä työvaiheessa tutustutaan nykyiseen järjestelmään ja ongelmakenttään mahdollisimman monelta suunnalta. Selvitettäviä asioita ovat muun muassa:

- Miten nykyjärjestelmä toimii osana ongelmakenttää? Millaista organisaatiota se palvelee?
- Mitkä ovat nykyisen järjestelmän tavoitteet, komponentit, säännöt ja tehtävät, mitä rajoituksia ja sääntöjä sillä on?

- Mitkä ovat vaatimusmäärittelyn sidosryhmät?
- Mitkä ovat sidosryhmien mielestä nykyisen järjestelmän vahvuudet ja heikkoudet?

(Paakki, 2011.)

Seuraavaksi kartoitetaan vaatimukset. Tässä työvaiheessa syvennetään tietämystä ongelmakentästä, hankitaan raakatietoa tulevasta järjestelmästä ja sen käyttöympäristöstä sekä etsitään potentiaalisia vaatimuksia esimerkiksi seuraavista asioista:

- Miten uudella teknologialla ja/tai muuttuneilla liiketoimintamalleilla voidaan korjata nykyisen järjestelmän heikkouksia kuitenkaan menettämättä sen vahvuuksia?
- Mitä nykyistä järjestelmää parantavia tavoitteita tulevalla järjestelmällä on ja mitä erilaisia vaihtoehtoja tavoitteiden täyttämiseen on?
- Mitkä ovat ympäristön ja tekniikan tulevalle järjestelmälle asettamat rajoitteet?
- Selvitetään tyypillisiä tulevan ohjelmiston ja sen toimintaympäristön välistä vuorovaikutusta kuvaavia skenaarioita.
- Selvitetään ongelmakentän ja toimintaympäristön ominaisuudet ja oletukset.
- Selvitetään edellisten kohtien kanssa yhteensopivat tulevan ohjelmiston vaatimukset.

(Paakki, 2011.)

Seuraavaksi arvioidaan vaatimukset. Tässä työvaiheessa ratkotaan kartoitusvaiheessa esille nousseita kysymyksiä ja ongelmia. Muun muassa vaatimuksiin liittyvät ristiriidat ratkotaan. Eri näkökulmat ja sidosryhmät voivat nähdä ongelmakentän ja tulevan ohjelmiston eri tavoin, mikä voi johtaa ristiriitaisiin vaatimuksiin tai erilaisiin tulkintoihin. Tehdään riskianalyysi tulevasta järjestelmästä. Kartoitusvaiheessa esille tulleet vaihtoehtoiset ratkaisut arvioidaan ja vaihtoehdoista valitaan paras.

Vaatimukset myös priorisoidaan, se helpottaa sekä vaatimusten arviointia ja toteutusaikataulun laadintaa. (Paakki, 2011.)

Näiden jälkeen tarkennetaan, strukturoidaan ja dokumentoidaan arviointivaiheessa hyväksyttyjä tulevan järjestelmän ominaisuuksia. Tuloksena saadaan vaatimusdokumentti, joka sisältää tulevan järjestelmän tavoitteet, määritelmät, ongelmakentän ominaisuudet, vastuut, järjestelmävaatimukset, ohjelmistovaatimukset ja käyttöympäristöoletukset. Dokumentti voi sisältää myös perusteluja tehdyille valinnoille, hyväksymistestitapauksia, kustannusarvioita, ja niin edelleen. (Paakki, 2011.)

Viimeisenä vahvistetaan vaatimukset. Vaatimusdokumentista varmennetaan, että tuleva järjestelmä täyttää sidosryhmien tarpeet. Samalla varmistetaan vaatimusdokumentin laatu: spesifikaatiot on kuvattu selkeästi ja yhtenevästi, ne ovat ristiriidattomat ja kattavat ongelmakentän. Vahvistettu vaatimusdokumentti toimii syötteenä kehitystyölle. Vaatimusmäärittelyprosessi on kuitenkin iteratiivinen. Kaikkea ei selvitetä kerralla, vaan ongelmakentän ymmärrys ja vaatimukset tarkentuvat vaiheittain. (Paakki, 2011.)

3.2 Määrittely LogiDa-järjestelmässä

Järjestelmän määrittely ja suunnittelu jäi pitkälti minun vastuulleni. Ensimmäisessä järjestelmää koskevassa palaverissa sovittiin suuripiirteisesti järjestelmän tärkeimmät tietotarpeet ja toiminnot. Tutkin asiakkaan vanhaa Excel-pohjaista tuotetietotaulukkoa josta sain tärkeimmät tiedot, jonka jälkeen sovittiin muista tietotarpeista järjestelmälle.

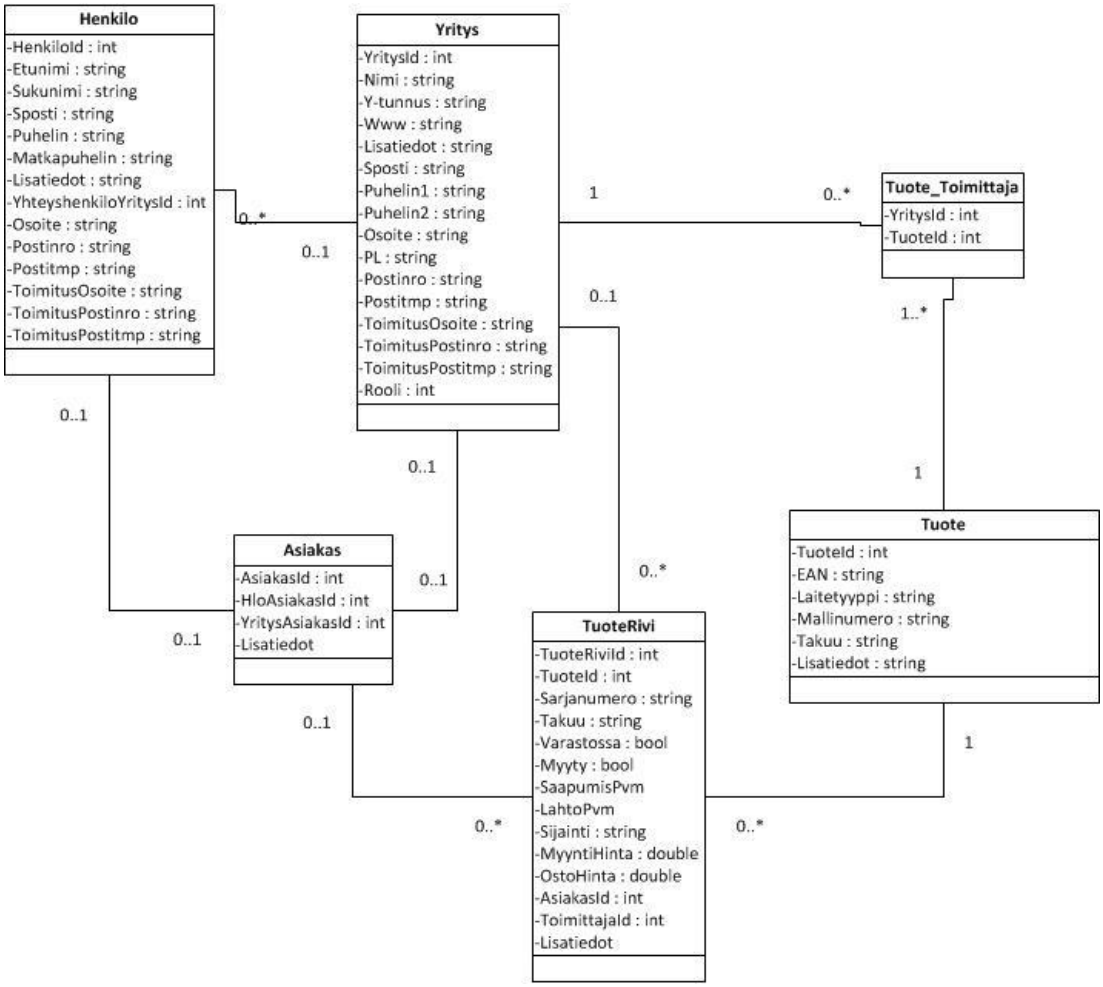
Tämän projektin määrittely ei ollut läheskään niin kattava kuin normaalien tietojärjestelmäprojektien vaatimusmäärittely. En nähnyt erityistä tarvetta tehdä laajaa ja tarkkaa selvitystä tulevan järjestelmän toiminnoista ja tietotarpeista. Asiakas luotti arvostelukykyyni järjestelmän suunnittelussa alusta loppuun, ja minulla oli jo hyvä kuva siitä mitä järjestelmän toiminnallisuudelta tarvitaan, koska olen samassa yrityksessä työskennellyt samantapaisten laitekantajärjestelmien parissa.

3.3 Luokkakaavio

Luokkakaaviot kuvaavat järjestelmään kuuluvia luokkia, niiden tietoja, toimintoja ja niiden välisiä suhteita. Niitä käytetään ohjelmistokehityksessä kuvaamaan järjestelmän staattista rakennetta. Luokkien välisiä suhteita sanotaan assosiaatioiksi. Voi olla esimerkiksi kaksi luokkaa, koulu ja opettaja. Näiden kahden luokan assosiaatio on se, että koulussa on opettajia. Luokkien välisissä yhteyksissä ilmaistaan myös assosiaation kerrannaisuus, joka voi olla yhden suhde yhteen (1:1), yhden suhde moneen(1:M) tai monen suhde moneen (M:N). Esimerkiksi koulun ja opettajan välinen assosiaatio on yhden suhde moneen, koska koulussa voi olla monia opettajia, mutta opettajalla voi olla vain yksi koulu.

Kuva 1. Esimerkki luokkakaaviosta, jossa kuvataan työnantajan ja työntekijöiden välinen assosiaatio. Luokan nimi on otsikossa, jonka alapuolella on luokkaan liittyvät tiedot (attribuutit). Yrityksen tietoja ovat esimerkiksi nimi ja y-tunnus, ja henkilön tietoja ovat etunimi, sukunimi ja puhelinnumero. Assosiaatiossa (luokat yhdistävä viiva) on selitykset, joista käy ilmi että yritys on henkilön työnantaja, ja henkilö on yrityksen työntekijä. Assosiaatiosta käy myös ilmi sen kerrannaisuus, eli yrityksellä on ei yhtään tai monta työntekijää ja henkilöllä voi olla korkeintaan yksi työnantaja.

Ensimmäisen palaverin perusteella laadin karkean luokkakaavion järjestelmän tietotarpeista. Projektin edetessä ja asiakkaan lisätietojen perusteella päivitin sitä vastaamaan paremmin asiakkaan odotuksia. Luokkakaavio toimi tärkeimpänä perustana projektissa sillä se sisälsi koko järjestelmän tärkeimmän asian, eli tietotarpeet yksityiskohtaisesti.



Kuva 2. Karkea luokkakaavio tulevasta järjestelmästä. Luokkiin on merkitty niiden tiedot (attribuutit) ja niiden suunnitellut tietotyypit (esimerkiksi string). Luokkien välille on merkitty assosiaatiot. Esimerkiksi assosiaatio Asiakkaan ja TuoteRivin välillä: asiakkaalla voi olla ei yhtään tai monta tuoteriviä (0..*), ja yhdellä tuoterivillä voi olla korkeintaan yksi asiakas. (0..1). Muita assosiaatioita ovat esimerkiksi se että Asiakas on joko Yritys tai Henkilö, Yrityksellä voi olla yhteyshenkilö, Tuotteella voi olla yksi tai useampi toimittajayritys tai yritys voi toimittaa ei yhtään tai useampia tuotteita. Tuotteen ja sen toimittajayrityksen väliin on merkitty molemmat yhdistävä luokka, Tuote_Toimittaja. Tämä johtuu siitä että tekninen toteutus vaatii M:N (monen suhde moneen) yhteyksistä tehtävän oman luokan, joka yhdistää molemmat alkuperäiset luokat.

4 KÄYTETYT TEKNIIKAT

4.1 Yleisistä selaintekniikoista lyhyesti

4.1.1 HTML

HTML (Hypertext Markup Language) on rakenteellinen joukko merkintäelementtejä syötettynä tiedostoon joka on tarkoitettu näytettäväksi selaimessa verkkosivulla. Merkintäkieli kertoo selaimelle kuinka näyttää sivun sanat ja kuvat käyttäjälle. Näitä merkintöjä sanotaan elementeiksi. Jotkut elementit tulevat pareina, jolloin ne ilmaisevat milloin jokin näyttövaikutus alkaa ja milloin se loppuu. (Rouse 2005.)

4.1.2 CSS

CSS (Cascading Style Sheets) on tyylikieli joka määrittelee HTML-dokumenttien rakenteellisen sisällön ulkoasua. CSS kattaa esimerkiksi fontit, värit, marginaalit, viivat, korkeuden, leveyden, taustakuvat ja monia muita asioita. Yhdellä CSS-tyylitiedostolla voi hallita monien dokumenttien ulkoasua ja se voidaan pitää täysin erillisenä itse dokumentin sisällöstä. (http://www.html.net/tutorials/css/lesson1.php)

4.1.3 JavaScript

JavaScript on ohjelmointikieli jota käytetään tehdessä web-sivuista interaktiivisia. Se ajetaan käyttäjän selaimessa ja se ei vaadi jatkuvia latauksia sivulta. Sitä käytetään muun muassa kyselyiden toteuttamiseen web-sivuilla. Javascript-koodi voidaan sisällyttää HTML-dokumentin sisään tai se voidaan pitää erillisessä tiedostossa. (Chapman, haettu 9.4.2013.)

4.2 ASP.NET Web Forms

ASP.NET Web Forms on yksi kolmesta ohjelmointimallista ASP.NET web-sivujen ja web-sovellusten luomisessa. Kaksi muuta ohjelmointimallia ovat ASP.NET Web

Pages ja ASP.NET MVC. Web Forms on vanhin ASP.NETin ohjelmointimalli, tapahtumaohjatuilla web-sivuilla jotka ovat kirjoitettu html:n, palvelinkontrollien ja palvelinkoodin yhdistelmänä. Web Formit käännetään ja suoritetaan palvelimella, joka luo HTML:n joka näyttää web-sivut. Web Formien mukana tulee satoja erilaisia web-kontrolleja ja web-komponentteja käyttäjäohjattujen web-sivujen luomiseen, joissa on pääsy dataan. (W3Schools.com, haettu 15.4.2013.)

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Pages/Site.master" AutoEventWireup="true" CodeFile="SelaaTuotteita.aspx.cs" Inherits="Pages_SelaaTuotteita" %>

<asp:Content ID="Content1" ContentPlaceHolderID="HeadContent" Runat="Server">
    <link href="../Styles/GridViewStyle.css" rel="stylesheet" type="text/css" />
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="MainContent" Runat="Server">
    <h1>Tuotteiden selaus</h1>

    <asp:ScriptManager ID="scriptmgr" runat="server"></asp:ScriptManager>
    <asp:UpdatePanel ID="updPanel" runat="server">
        <ContentTemplate>

            <fieldset>
                <legend>Haku</legend>

                <asp:ValidationSummary ID="validationSummary" runat="server" CssClass="failureNotification" HeaderText="TuoteId:n täytyy olla numero." />

                TuoteId: <asp:TextBox ID="txtHakuTuoteId" runat="server" Width="100" ></asp:TextBox>
                    <asp:CompareValidator ID="cmpHakuTuoteId" runat="server" Text="*" ControlToValidate="txtHakuTuoteId"
                                          Operator="DataTypeCheck" Type="Integer" CssClass="failureNotification"></asp:CompareValidator>
                EAN: <asp:TextBox ID="txtHakuEAN" runat="server" Width="100" ></asp:TextBox>
                Laitetyyppi: <asp:TextBox ID="txtHakuLaitetyyppi" runat="server" Width="100" ></asp:TextBox>
                Mallinumero: <asp:TextBox ID="txtHakuMallinumero" runat="server" Width="100" ></asp:TextBox>
                    <asp:Button ID="btnHaeTuotteet" runat="server" Text="Hae" onclick="btnHaeTuotteet_Click" />

                <br class="clear" />

                    <asp:LinkButton ID="lnkHaeKaikki" runat="server" onclick="lnkHaeKaikki_Click">Hae kaikki tuotenimikkeet</asp:LinkButton>
                    <div class="rightLinkki"><a href="LisaaTuote.aspx">Lisää uusi tuote</a></div>
            </fieldset>

            <asp:Label ID="lblTopIlmoitus" runat="server" Text="" Visible="false" CssClass="failureNotification"></asp:Label>

            <asp:GridView ID="gridTuotteet" runat="server"
                CssClass="grid"
                AlternatingRowStyle-CssClass="gridRivi2"
                AutoGenerateSelectButton="true"
                SelectedRowStyle-CssClass="gridValittu"
                EmptyDataText="Tuotteita ei löytynyt. &lt;a href='LisaaTuote.aspx'&gt;Lisää uusi tuote&lt;/a&gt;"
                AutoGenerateColumns="false"
                onselectedindexchanged="gridTuotteet_SelectedIndexChanged"
                AllowPaging="true"
                PageSize="20"
                onpageindexchanging="gridTuotteet_PageIndexChanging"
                onrowdatabound="gridTuotteet_RowDataBound" >
                <Columns>
                    <asp:BoundField HeaderText="Id" DataField="TuoteId" />
                    <asp:BoundField HeaderText="EAN" DataField="EAN" />
                    <asp:BoundField HeaderText="Laitetyyppi" DataField="Laitetyyppi" />
                    <asp:BoundField HeaderText="Mallinumero" DataField="Mallinumero" />
                    <asp:BoundField HeaderText="Takuu" DataField="Takuu" />
                    <asp:BoundField HeaderText="Rivejä" />
                    <asp:BoundField HeaderText="Varastossa" />
                </Columns>
            </asp:GridView>
```

Kuva 3. Esimerkki ASP.NET Forms –tekniikan käytöstä LogiDassa, tuotenimikkeiden selaus ja haku. Koodissa on tavallisia HTML-elementtejä, sekä myös ASP.NETin elementtejä (<asp:kontrollinimi />) jotka generoivat HTML-koodia joka näytetään selaimessa.

4.2.1 ASP.NET Membership

Membership on ASP.NETiin sisäänrakennettu tapa validoida ja tallentaa käyttäjätietoja. Membership auttaa siis hallitsemaan käyttäjien autentikointia web-sivuilla. Membership:iä voi käyttää helposti ASP.NETin sisäänkirjautumiskontrollien kanssa jotta voit luoda kokonaisen järjestelmän käyttäjien autentikointiin. (Microsoft Developer Network, haettu 12.4.2013.)

ASP.NET Membership:llä voi

- luoda uusia käyttäjiä ja salasanoja
- tallentaa membership-tietoja (käyttäjänimiä, salasanoja ja tukidataa) Microsoft SQL Serverissä, Active Directoryssä, tai vaihtoehtoisessa tietovarastossa.
- autentikoida käyttäjiä jotka vierailevat sivustolla. Käyttäjät voi autentikoida ohjelmallisesti, tai voit käyttää ASP.NETin kirjautumiskontrolleja luodaksesi autentikointisysteemin joka tarvitsee vähän tai ei ollenkaan ohjelmakoodia.
- hallita salasanoja. Tämä sisältää salasanojen luomisen, vaihtamisen tai nollaamisen.
- antaa yksilöllisen tunnisteen autentikoiduille käyttäjille jota voi käyttää omissa sovelluksissa ja joka integroituu ASP.NETin personointi- ja roolinhallinta-järjestelmissä.
- täsmentää mukautetun membership-tarjoajan, joka antaa sinun luoda omaa koodia membership:in hallintaan ja ylläpitoon mukautetussa tietovarastossa.

(Microsoft Developer Network, haettu 12.4.2013.)

```
<asp:Login ID="LoginUser" runat="server" EnableViewState="false" RenderOuterTable="false"
    FailureText="Sisäänkirjautuminen ei onnistunut. Yritä uudelleen." DestinationPageUrl="~/Pages/Etusivu.aspx">
    <LayoutTemplate>
        <div class="accountInfo">
            <fieldset class="login">
                <legend>Käyttäjätilin tiedot</legend>
                <p>
                    <asp:Label ID="UserNameLabel" runat="server" AssociatedControlID="UserName">Käyttäjätunnus:</asp:Label>
                    <asp:TextBox ID="UserName" runat="server" CssClass="textEntry"></asp:TextBox>
                </p>
                <p>
                    <asp:Label ID="PasswordLabel" runat="server" AssociatedControlID="Password">Salasana:</asp:Label>
                    <asp:TextBox ID="Password" runat="server" CssClass="passwordEntry" TextMode="Password"></asp:TextBox>
                </p>
            </fieldset>
            <p class="submitButton">
                <asp:Button ID="LoginButton" runat="server" CommandName="Login" Text="Kirjaudu sisään"
                    ValidationGroup="LoginUserValidationGroup" />
            </p>
        </div>
    </LayoutTemplate>
</asp:Login>
```

Kuva 4. ASP.NET Login -kontrollin käyttö. Suhteellisen pienellä määrällä koodia saadaan aikaiseksi käyttäjän kirjautuminen.

4.2.2 ASP.NET AJAX

ASP.NETin AJAX-ominaisuudet antavat luoda helposti web-sivuja jotka sallivat rikkaan käyttäjäkokemuksen responsiivisella ja tuttavallisella käyttäjäliittymällä. AJAX-ominaisuudet sisältävät client-script kirjastoja jotka sisällyttävät selainrajat ylittävän ECMAScriptin (JavaScript) ja dynaamisen HTML:n (DHTML) teknologiat, ja integroinnin ASP.NETin palvelinpohjaiseen kehitysympäristöön. Käyttämällä AJAX-ominaisuuksia, voi parantaa web-sovellusten käyttäjäkokemusta ja tehokkuutta. (Microsoft Developer Network, haettu 12.4.2013.)

AJAX-ominaisuuksilla varustetut web sovellukset tarjoavat

- parannettua tehokkuutta, koska osat web-sivun toiminnallisuudesta suoritetaan selaimessa.
- tuttavallisia käyttöliittymäelementtejä, kuten etenemisilmaisimia, työkaluvihjeitä ja ponnahdusikkunoita.
- sivun osittaiset virkistykset, jotka päivittävät sivusta vain ne osat jotka ovat muuttuneet.
- asiakas-integraation ASP.NETin sovelluspalveluille kuten forms-autentikoinnille, rooleille ja käyttäjäprofiileille.

- automaattisesti luotuja välitysluokkia jotka yksinkertaistavat web-palvelun metodien kutsumista asiakas-skripteistä.
- tuen suosituimmille ja yleisimmille selaimille.

(Microsoft Developer Network, haettu 12.4.2013.)

```
<asp:ScriptManager ID="scriptmgrAsiakkaat" runat="server"></asp:ScriptManager>
<asp:UpdatePanel ID="updAsiakkaat" runat="server">
    <ContentTemplate>

        <fieldset>...</fieldset>

        <asp:Label ID="lblTopIlmoitus" runat="server" Text="" Visible="false" CssClass="failureNotification"></asp:Label>

        <asp:GridView ID="gridAsiakkaat" ...>...</asp:GridView>

        <br class="clear" />

        <asp:PlaceHolder ID="phAsiakasKortti" ...>...</asp:PlaceHolder>

    </ContentTemplate>
</asp:UpdatePanel>
```

Kuva 5. ASP.NET AJAX LogiDa-järjestelmässä. Sivulla päivitetään vain UpdatePanel-kontrollin sisällä oleva muuttunut sisältö (partial-page update).

4.3 ADO.NET Entity Framework

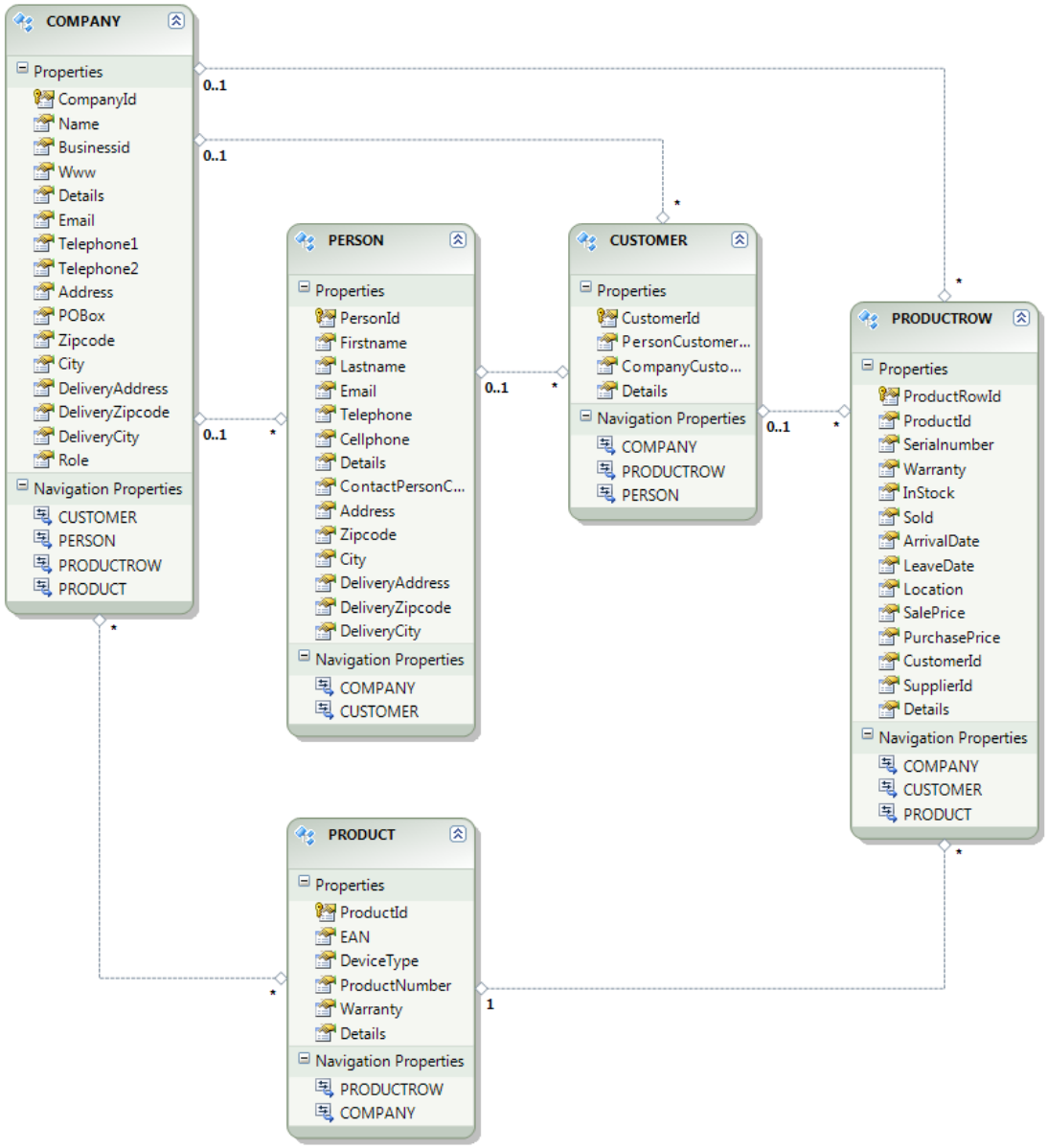
ADO.NET (ActiveX Data Objects) on .NET Frameworkin tietokantateknologia jota käytetään yhdistämään sovellusohjelma tietokantapalvelimeen. Se on .NET Frameworkin osa, joka koostuu joukosta luokkia joita käytetään datan käsittelyyn. Se käyttää XML:ää datan tallentamiseen ja siirtämiseen sovellusten kesken, joka ei ole vain alan standardi mutta tarjoaa myös nopean pääsyn dataan hajautetuille sovelluksille ja työpöytäsovelluksille. ADO.NET on hyvin skaalautuva ja yhteentoimiva. (Puran 2009.)

ADO.NET tarjoaa johdonmukaisen pääsyn datalähteisiin kuten esimerkiksi Microsoft SQL Server:iin, kuten myös OLE DB:n ja XML:n kautta esitettyihin datalähteisiin. Sovellukset voivat käyttää ADO.NETiä yhdistääkseen näihin datalähteisiin sekä datan noutamiseen, manipuloimiseen ja päivittämiseen. (Microsoft Developer Network, haettu 15.4.2013.)

```
// Luodaan uusi db_svt_logiEntities-olio
using (db_svt_logiEntities de = new db_svt_logiEntities())
{
    // luodaan uusi tuote
    PRODUCT dbRivi = de.CreateObject<PRODUCT>();
    // asetetaan uuden tuotteen ominaisuudet
    dbRivi.EAN = EAN.ToUpper();
    dbRivi.DeviceType = Laitetyyppi;
    dbRivi.ProductNumber = Mallinumero.ToUpper();
    dbRivi.Warranty = Takuu;
    dbRivi.Details = Lisatiedot;

    // lisätään uusi tuote
    de.AddToPRODUCT(dbRivi);
    // tallennetaan muutokset
    de.SaveChanges();
}
```

Kuva 6. Uuden tuotteen lisääminen LogiDa-järjestelmässä.



Kuva 7. LogiDa-järjestelmän graafinen ADO.NET entiteettidatamalli.

4.4 LinQ

LinQ (Language Integrated Query) julkaistiin vuonna 2007 osana .NET Framework 3.5:ttä. Se mahdollistaa datan kyselyn .NETin tuetuilla kielillä. Se määrittelee joukon kyselyoperaattoreita jotka mahdollistavat pääsyn dataan useammista lähteistä. (De Silva 2010.)

LinQ:ssa on monia etuja, kuten

- vahvasti kirjoitettujen kyselyiden saatavuus. Luokat luodaan automaattisesti relaatiotietokantojen relaatioiden mukaan. Kieli on erittäin helppoa ymmärtää, kuten SQL.
- viiteavaimien automaattiset liitokset. Normaalissa SQL:ssä, käyttäjän täytyy liittää taulut itse jos se on tarpeellista. LinQ tarjoaa kyvyn liittää jokaisen taulun automaattisesti mikäli niistä löytyy viiteavain.
- pienentynyt koodikoko. Monissa tilanteissa käyttäjän täytyy kirjoittaa pitkiä lauseita saadakseen SQL kyselyn aikaiseksi. LinQ tarjoaa suhteellisen lyhyitä koodeja edistyneissäkin tilanteissa. Se vähentää koodin monimutkaisuutta ja tekee koodin lukemisesta paljon helpompaa.
- koodin tasa-arvo. Yksi LinQ:n suuri etu on sen saatavuus minkä tahansa .NET alustan kielen kanssa, kuten C#.NET, VB.NET ja F#.NET.

(De Silva 2010.)

LinQ to SQL

LINQ to SQL on .NET Framework 3.5:n komponentti joka tarjoaa ajonaikaisen infastruktuurin relaatiodatan, kuten olioiden, hallitsemiseen. (Microsoft Developer Network, haettu 15.4.2013.)

LINQ to SQL:ssä, relaatiotietokannan datamalli kartoitetaan oliomalliin joka on ilmaistu sovelluksen kehittäjän ohjelmointikielellä. Kun sovellus ajetaan, LINQ to SQL käännetään SQL-kieleksi oliomallissa ja se lähetetään tietokannalle suoritettavaksi. Kun tietokanta palauttaa tulokset, LINQ to SQL kääntää ne takaisin olioiksi joiden kanssa voi työskennellä omalla ohjelmointikielellä. (Microsoft Developer Network, haettu 15.4.2013.)

```
// Haetaan tietokannasta yritys, jonka nimi alkaa annetulla merkkijonolla
var data = (from r in de.COMPANY            // Haetaan yrityksistä
            where r.Name.StartsWith(nimi)   // Jonka nimi alkaa annetulla hakuehdolla
            select r);                      // Valitaan rivit kokonaan
```

Kuva 8. LinQ To SQL esimerkki LogiDasta.

5 TYÖN KULKU

5.1 Toteutustapa ja aikataulu

Järjestelmän toteuttaminen alkoi virallisesti 4.10.2012 sopimuksella opinnäytetyön tekemisestä, mutta työtä järjestelmän eteen on tehty jo 2012 tammikuun lopusta lähtien. Sain erittäin vapaat kädet sovelluksen toteuttamiseen itsenäisesti, kunhan se täyttäisi alunperin asiakkaan kanssa sovitut tarpeet.

Projektissa käytin pitkälti hyväksi koulussa opittuja taitoja. Itse ohjelmointityön mallin sain suurimmilta osin kurssilta jossa opiskeltiin web-järjestelmän ohjelmointia. Mallin järjestelmän määrittelyyn sain kurssilta jossa suunniteltiin ryhmäprojektina tietojärjestelmä ja mallin projektin toteuttamiseen sain paljolti samaan opintojaksoon liittyvään kurssiin, jossa toteutettiin tämä suunniteltu tietojärjestelmä. Näiden kurssien käyminen ja niissä saatu kokemus helpotti suuresti erilaisten työmäärien ja aikataulujen arviointia.

Pääasiallinen työkaluni järjestelmän toteuttamisessa oli Microsoft Visual Studio 2010. Web-palvelimena tultaisiin käyttämään IIS-palvelinohjelmistoa ja järjestelmän tietokanta tultaisiin toteuttamaan Microsoft SQL Server 2008 R2 Express palvelinohjelmistolla. Järjestelmän ohjelmointi- ja testaustyössä käytin omia työasemiani ja lopullista kohdekonetta vastaavan testausympäristön loin myös omilla laitteilla.

5.2 Toteutuksen eteneminen

Työn edetessä syksyyn tuli selväksi, että itsenäinen työskentely vaatii hyvin monia henkilökohtaisia tavoitteita, aikarajoja ja itsekuria. Kuten projektien hallintaan liittyvillä kursseilla on moneen kertaan opetettu, projekti pitäisi aloittaa aina selkeästi ja vankasti. Pitää määrittää selkeitä päivämääriä mihin mennessä tiettyjä osia täytyy olla valmiina, ja pitää niistä kiinni. Näitä rajapyykkejä saisi olla mahdollisimman paljon.

Kun järjestelmän toiminnallinen perusrunko alkoi olla kasassa, itse käyttöliittymän ja perustoiminnallisuuden luomisen myötä alkoi muodostua hyvin monia pieniä tehtäviä. Näiden pienten tehtävien määrä oli vaikeasti mitattava ja hieman hämärä. Tästä johtuen minun oli pakko luoda yksinkertainen Excel-taulukko, johon kirjasin mitä tehtäviä tarvitsee tehdä mihinkin järjestelmän osioon, missä järjestyksessä ne pitää tehdä ja muita huomioita. Täten sain selkeämmän kuvan järjestelmän työmääristä, kun oli joku pohja mihin pystyin merkitsemään vaadittuja töitä ja niitä valmiiksi. Työtä olisi todennäköisesti nopeuttanut suuresti jonkin projektinhallintaohjelmiston käyttö. Projektin alkuvaiheessa en nähnyt tätä tarpeellisena ja ajattelin että projektin etenemistietojen ylläpito veisi liikaa aikaa. Jälkeenpäin huomaan että mikäli olisin käyttänyt aikaa projektin yksityiskohtaiseen organisoimiseen, olisin säästänyt aikaa ja saanut hyvin organisoidun rungon tehtäville töille.

5.3 Haasteet

Projektin suurimman haasteen voisi sanoa olleen aikataulu, koska järjestelmä toimitettiin huomattavasti myöhemmin kuin oli alunperin suunniteltu. Todellisuudessa suurin haaste oli oman henkilökohtaisen ajan organisointi järjestelmän eteen tehtäviin töihin. Työskentelyssä ei ollut selkeätä kuvaa mitä seuraavaksi pitäisi tehdä ja miten pitäisi edetä, eikä selkeätä päätöspistettä missä vaiheessa järjestelmä tai jokin sen osakokonaisuus on valmis. Korostaisin suuresti tämän ongelman ratkaisuksi jatkossa mahdollisimman perusteellista suunnittelutyötä. Olin jo oppinut koulussa järjestelmän toteuttamiseen liittyvällä opintojaksolla, että hyvä suunnittelu on tärkeintä järjestelmän toteuttamisessa. Pitää olla paperilla konkreettisia alkupisteitä, etenemisvaiheita, ja selkeä päätöspiste. Tätä järjestelmää lähdin toteuttamaan melko hataralla suunnittelulla. Luotin pitkälti omaan mielikuvaani siitä, millainen tulevan järjestelmän pitäisi olla. Tämä ei riitä, vaan pitää olla konkreettisia suunnitelmia lopputuloksesta, ja niitä pitää jo hyvissä ajoin vertailla asiakkaan mielipiteen kanssa.

5.4 Testaus

Ohjelmistotestaus on prosessi jonka tarkoitus on löytää virheitä ajamalla sovellusta. Se voi olla myös toimintaa jossa arvioidaan järjestelmän kykyä suoriutua vaadituista tavoitteista. Ohjelmistovirheitä tulee melkein aina olemaan missä tahansa kohtuukokoisessa ohjelmistomoduulissa: ei sen takia että ohjelmoijat ovat huolimattomia tai vastuuttomia, vaan koska ohjelmiston monimutkaisuus on yleisesti hankalaa ja ihmisillä on vain rajoitettu kyky hallita monimutkaisuutta. On myös totta että monimutkaisissa järjestelmissä, suunnitteluvirheitä ei koskaan voida täysin sulkea pois. (Pan Jintao 1999.)

Toisin kuin useimmissa fyysisissä järjestelmissä, suurin osa ohjelmistovirheistä on suunnitteluvirheitä, ei valmistusvirheitä. Ohjelmisto ei kärsi kulumisesta tai ruostumisesta, yleisesti se ei muutu ennenkuin sitä päivitetään tai kunnes se vanhentuu. Joten kun ohjelmisto toimitetaan, suunnitteluvirheet tai bugit ovat hautautuneena ohjelmistoon ja piilevät siellä kunnes ne aktivoituvat käytössä. (Pan Jintao 1999.)

LogiDa-järjestelmän testauksen suoritin hyvin yksinkertaisesti, ilman erityisiä suunnitelmia tai työkaluja. Kyseessä oli kuitenkin kohtuullisen yksinkertainen järjestelmä jossa ei ollut erityisen monimutkaisia toimintoja. Tein testausta jatkuvasti projektin edetessä. Kun yksikin pieni komponentti oli valmis, kokeilin nopeasti erilaisia käyttötilanteita ja katsoin onko sen tulokset sitä mitä vaaditaan. Painotin tätä testausta enimmäkseen järjestelmän perustoimintalogiikan monimutkaisemmille osille, joissa oli suurempi riski kriittisten virheiden tapahtumiselle.

Kun järjestelmä oli valmis, tein vielä viimeisen testauksen miettimällä järjestelmän arkisia käyttötilanteita, kuten saapuneen tuotteen kirjaaminen järjestelmään tai lähtevän tuotteen merkintä poistuneeksi. Kirjasin ylös kaikki järjestelmän toiminnallisuutta sisältävät web-sivut ja tein niille erilaisia käyttötilannetestejä mahdollisimman monipuolisesti, jonka jälkeen merkitsin sivun testatuksi mahdollisten virheiden korjaamisen jälkeen. Saatoin esimerkiksi ottaa itseltäni löytyneen laitteen ja kirjasin sen tiedot järjestelmään sarjanumeroineen ja toimittajatietoineen. Testauksen tässä vaiheessa löysin eniten virheitä. Virheet eivät

tosin olleet kriittisiä jotka voisivat häiritä asiakasyrityksen toimintaa ja ne olivat pitkälti erittäin helppoja korjata. Testauksen perusteella olen hyvin luottavainen järjestelmän toimintaan monipuolisissakin käyttötilanteissa.

6 VALMIS JÄRJESTELMÄ

LogiDa-järjestelmä otettiin tuotantokäyttöön 20.3.2013. Järjestelmä asennettiin asiakkaan Windows Server 2008 R2 palvelimelle, joka toimii myös yrityksen Domain Controller –palvelimena. Täytyy huomauttaa, että yleensä ei ole suositeltavaa asentaa ylimääräisiä asioita DC-palvelimelle, mutta tässä tapauksessa se oli asiakkaan toive ja mahdollisten vakavien ongelmien riski oli pieni. Palvelimen Windows Serveriin asennettiin uusi IIS-palvelinrooli, sekä tietokantapalvelin MS SQL Server 2008 R2 Express.

6.1 Sovellus

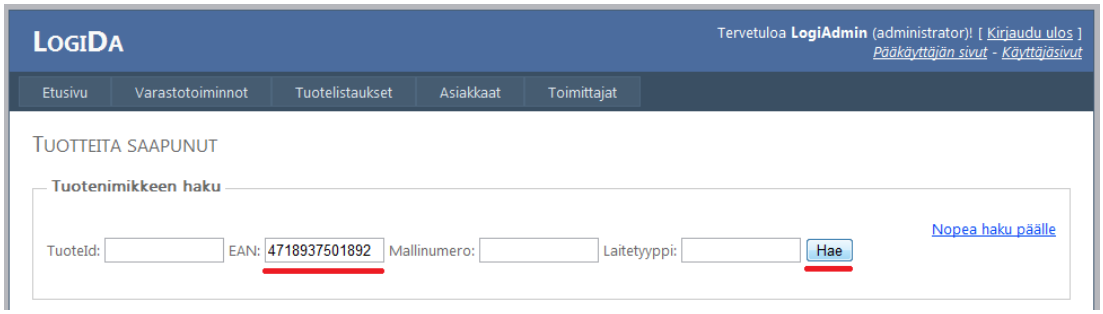
LogiDa pysyi mielikuvani mukaisesti mahdollisimman yksinkertaisena käyttää. Käyttöliittymän tein pitäen silmällä sitä että olisi aina helppoa ja nopeaa kirjata uusia tuotteita saapuneeksi ja lähteneeksi. Kuten alunperin oli suunniteltu, järjestelmää pystyy myös käyttämään viivakoodinlukijoilla. Kun esimerkiksi tuote saapuu varastoon, tuotteen tiedot voidaan hakea vaikka lukemalla EAN- tai tuotekoodi, jonka jälkeen nopeasti voidaan kirjata yksittäisen tuotteen yksilöivät tiedot, kuten sarjanumero.

Tein järjestelmään tuotteiden saapumiselle tai lähtemiselle selkeät sivut, joita käyttämällä voi nopeasti ja loogisesti merkitä tuoteriveille juuri lähtemistiedot tai saapumistiedot. En kuitenkaan halunnut että näiden tietojen kirjaaminen rajoittuu valmiiksi tehtyihin, melkein jopa ohjattuihin toimintoihin. Tästä johtuen pyrin pitämään järjestelmän sellaisena, että yksittäisten tuoterivien tietoja voisi muokata myös erittäin vapaasti ja manuaalisesti. Käyttäjä voikin ottaa käsittelyyn yksittäisen tuoterivin, lukea sen tiedot ja itse vapaasti merkitä sen myydyksi tai asiakkaalle luovutetuksi. Tämä voisi aiheuttaa ongelmia kokemattomammalla käyttäjällä.

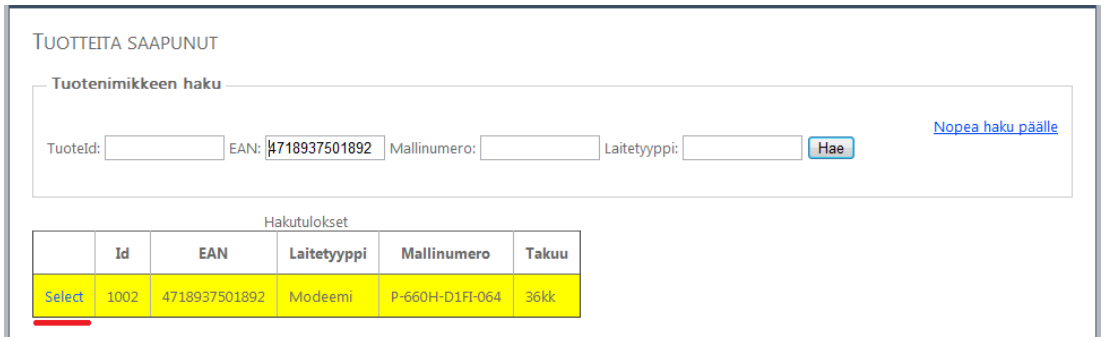
Esimerkiksi lähteneen tuotteen tiedoista voisi helposti jäädä merkitsemättä tärkeitä tietoja, kuten lähtöpäivämäärä tai tuotteen uusi sijainti. Järjestelmän tulevien käyttäjien tietotaidon huomioonottaen, en nähnyt tätä riskinä.

6.2 Esimerkkejä käyttötapauksista

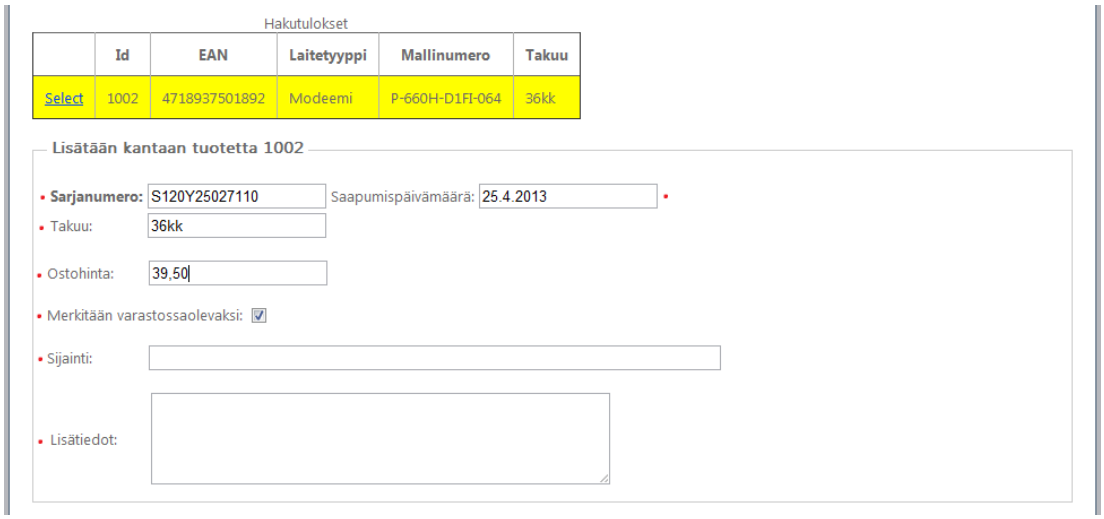
Saapuneen tuotteen kirjaaminen järjestelmään



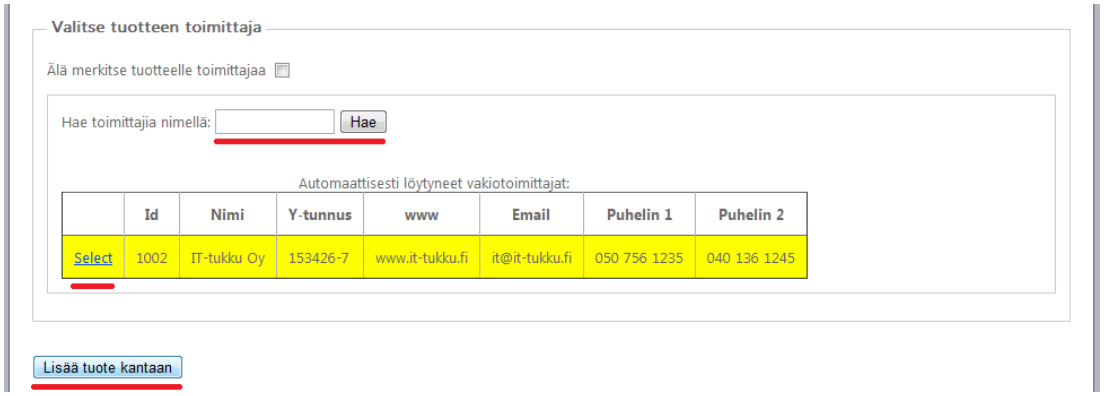
Kuva 9. Tässä esimerkissä haetaan saapuneen tuotteen EAN-koodilla tuotenimikettä kannasta. Mikäli tuotenimikettä ei ole vielä kirjattu kantaan, järjestelmä ehdottaa uuden lisäämistä.



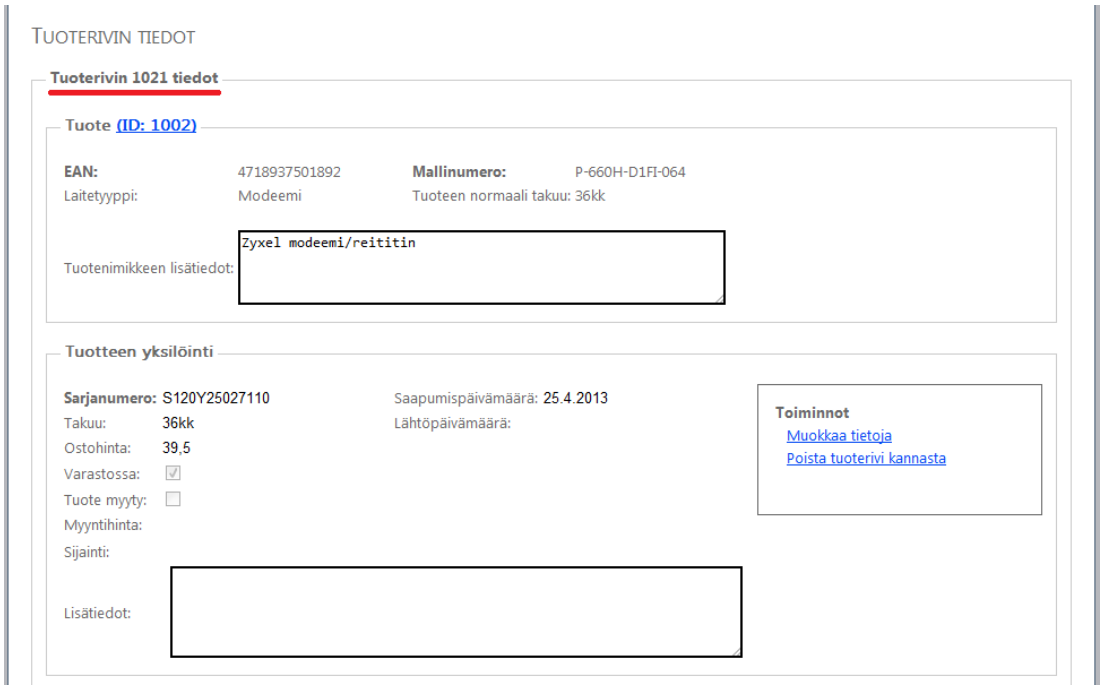
Kuva 10. Järjestelmä esittää haun ehdot täyttävät tuotenimikkeet listana, josta valitaan oikea tuotenimike. Tässä tapauksessa tuote löytyi välittömästi täydellisen EAN-koodin avulla, joten järjestelmä esittää seuraavan vaiheen tietojen kirjaamisen automaattisesti.



Kuva 11. Täydennetään tuotteen yksilöivät tiedot, kuten esimerkiksi sarjanumero. Järjestelmä ehdottaa automaattisesti saapumispäiväksi kuluvaa päivää ja takuuksi tuotenimikkeen tavallista takuuta.



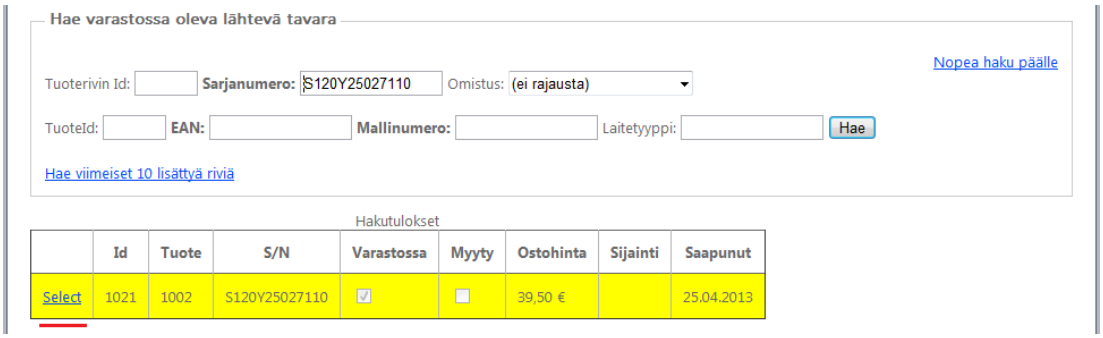
Kuva 12. Haetaan ja valitaan tuotteen toimittaja. Järjestelmä ehdottaa automaattisesti toimittajia, jotka tavallisesti toimittavat kyseistä tuotenimikettä. Toimittajia voi myös hakea nimellä tai toimittajan voi kokonaan jättää merkitsemättä. Valinnan jälkeen, lisätään tuote kantaan.



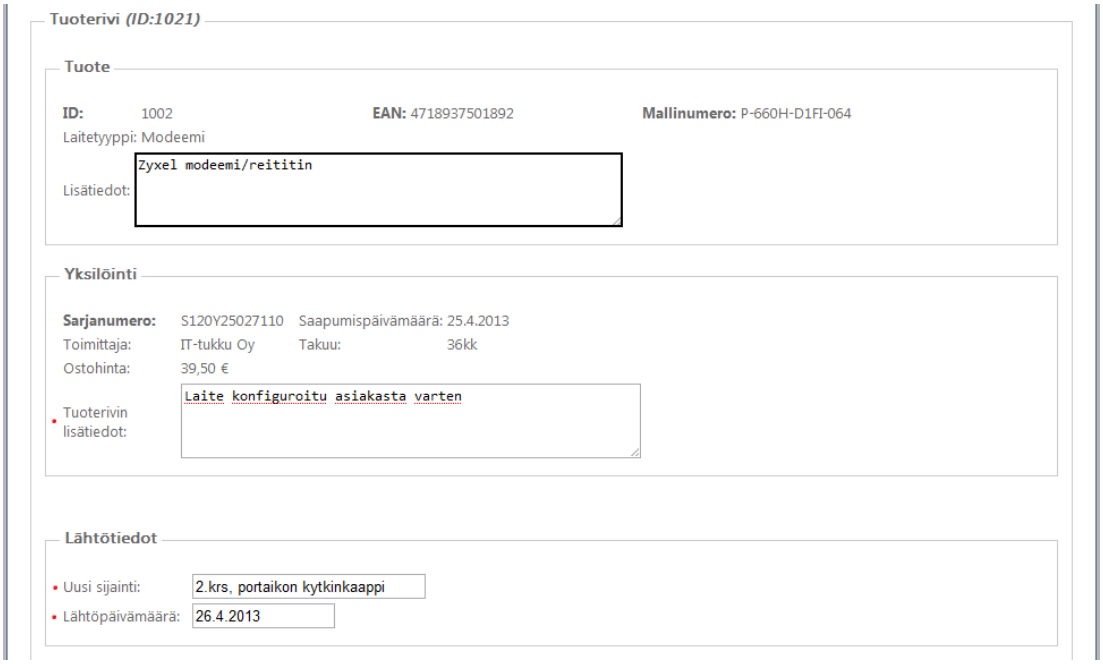
Kuva 13. Tuotteen lisäämisen jälkeen käyttäjä ohjataan sivulle jossa esitetään lisätyn tuotteen tarkat tiedot.

Lähtevän tuotteen kirjaaminen järjestelmään

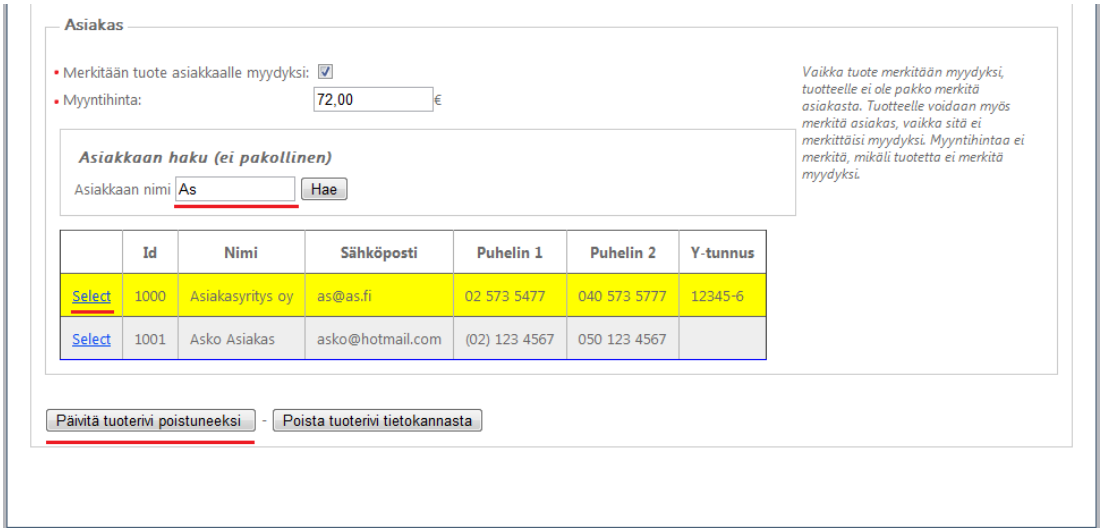
Kuva 14. Lähtevää tuotetta etsitään järjestelmästä lukemalla sarjanumero.



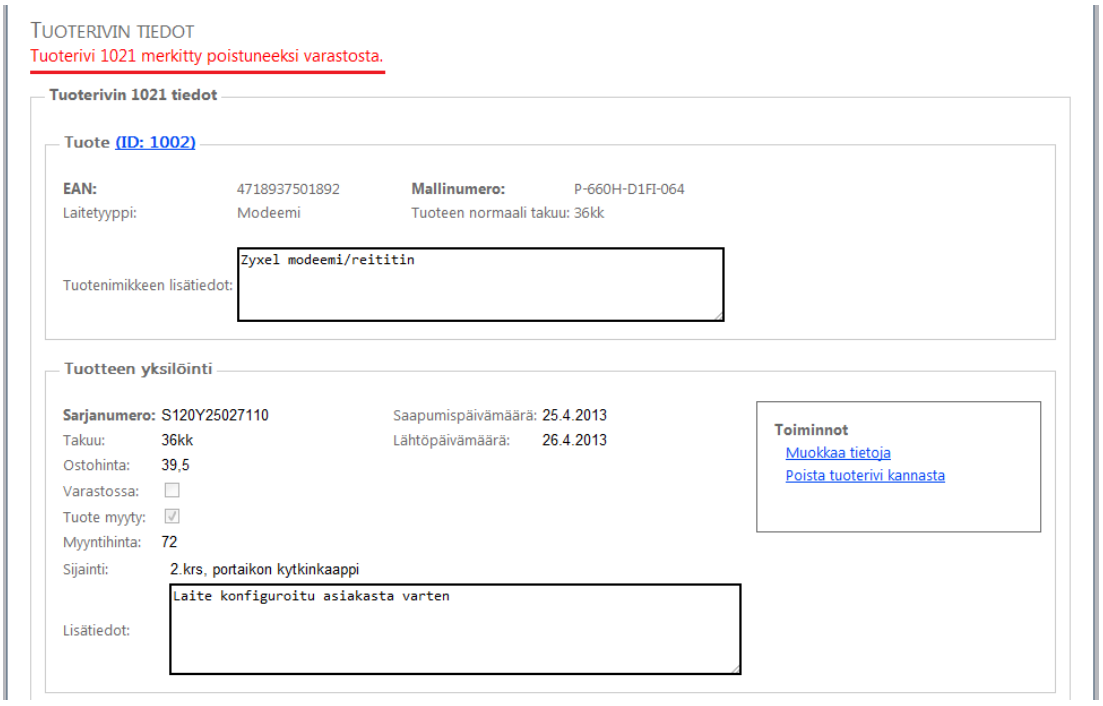
Kuva 15. Järjestelmä esittää hakutulokset listana, josta valitaan oikea tuote. Tässä tapauksessa tuote löytyi välittömästi yksilöidyn sarjanumeron perusteella, joten käyttäjälle esitetään automaattisesti seuraavan kohdan lomake tuotteen lähtötietojen kirjaamiseen.



Kuva 16. Tuotteelle merkitään lähtötiedot. Tuotteelle voi kirjata lisätietoja ja tuotteelle merkitään sen uusi sijainti. Järjestelmä ehdottaa automaattisesti kuluvaa päivää lähtöpäivämääräksi.

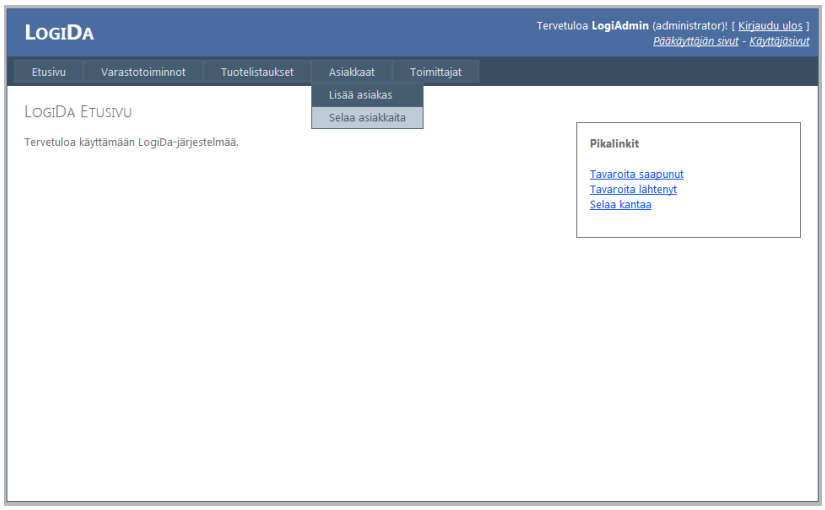


Kuva 17. Kirjataan tuotteen myyntitiedot. Tuotteelle kirjataan myyntihinta, jonka jälkeen voi etsiä sille asiakasta nimen perusteella. Järjestelmä listaa haun ehdon täyttävät tulokset ja niistä valitaan oikea asiakas joka merkitään tuotteelle. Tuotteelle voidaan myös jättää merkitsemättä erikseen asiakasta tai jättää se merkitsemättä myydyksi. Tämän jälkeen tuote klikataan poistuneeksi tai tuotteen voi suoraan poistaa täysin järjestelmästä, jos sillä ei ole enään mitään tekemistä yrityksen kanssa.

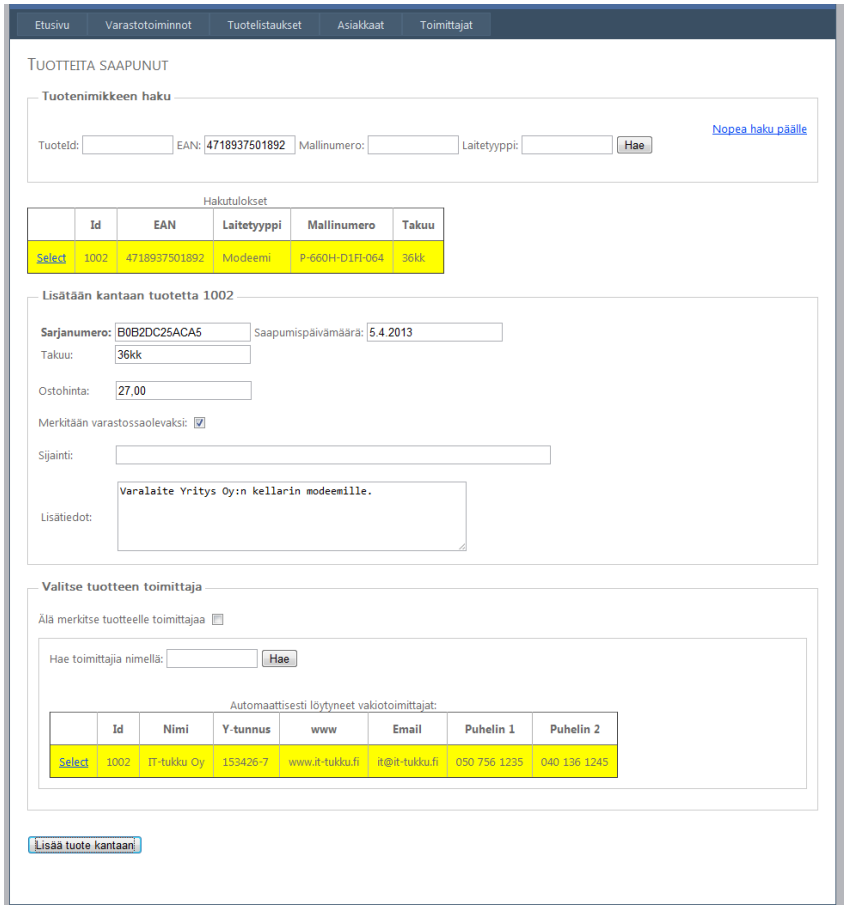


Kuva 18. Seuraavaksi käyttäjä ohjataan sivulle, jossa esitetään lähteneen tuotteen tarkat tiedot.

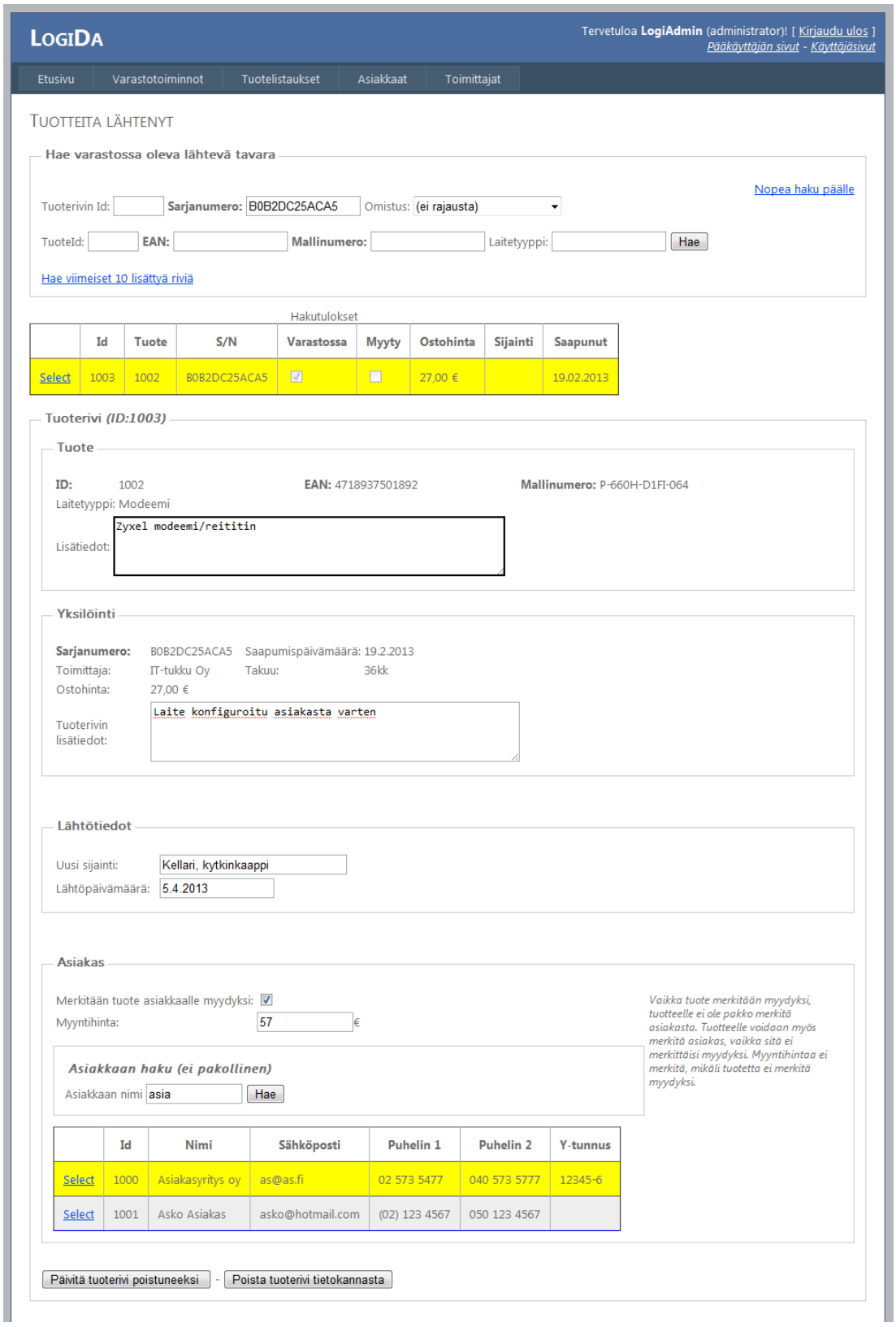
6.3 Kuvakaappauksia



Kuva 19. LogiDan etusivu.



Kuva 20. Saapuneiden tuotteiden kirjaus.



Kuva 21. Lähteneiden tuotteiden kirjaus.

7 JOHTOPÄÄTÖKSET

Olen tyytyväinen projektin lopputuloksena syntyneeseen järjestelmään. Projekti oli suuri ja aikaavievä, mutta korvaamaton oppimiskokemus. Vaikka koulussa meidät on opetettu tekemään tämänkaltaisia sovelluksia, jälleen kerran sai huomata että vain käytännön kokemuksella saa selkeimmän kuvan ja opin web-järjestelmien toteuttamisesta. Työssä oli kuitenkin joitakin asioita, joissa olisin voinut suoriutua paremmin.

Työn aikataulu epäonnistui suuresti omiin henkilökohtaisiin tavoitteisiini nähden. Työ alkoi siis epävirallisesti tammikuussa 2012, ja oletin saavani aikaiseksi asiakkaalle näytettävän selvän prototyypin tästä kuuden kuukauden kuluessa. Omat henkilökohtaiset kiireet ja koulutyöt hidastivat suuresti keväällä työn etenemistä. Suurin syy viivästymiseen on kuitenkin oman ajanhallinnan puute. Kuten jo aiemmin tekstissä painotin, pitää olla selkeitä suunnitelmia paperilla projektin etenemisestä. Kun on kirkas kuva siitä mikä pitää olla työn lopputulos, työn tekeminen helpottuu ja nopeutuu suuresti. Ilman suunnitelmia riskeeraa myös, että käyttää liikaa aikaa ylimääräisiin tai vähäpätöisiin ratkaisuihin, ja tärkeämmät ominaisuudet jäävät enemmän huomiotta.

En näe kuitenkaan työn ongelmia asiana mitä pitäisi katua, vaan enemmänkin osaamistani vahvistavana asiana jota voin peilata opintoihini.

8 LOPPUSANAT

Tein työharjoitteluni 2011 keväällä Satakunnan Verkkotekniikka Oy:ssä. Tämä työskentely antoi minulle tärkeän pohjustuksen siitä mitä mahdolliselta tuotetietojen tietojärjestelmältä odotetaan. Järjestelmäprojekteissa yksi yleinen kompastuskivi on se, että toimittajalla ei ole tarpeeksi hyvää tuntemusta asiakkaan toimintaympäristöstä ja toimintaprosesseista. Tämä ei ollut ongelma tässä projektissa, mutta ilman työskentelyäni asiakkaan yrityksessä, en olisi saanut tietoa toiminnan erityispiirteistä jotka asettavat tiettyjä yksilöllisiä vaatimuksia järjestelmälle.

Työn myötä olen saanut jälleen hieman selkeämmän kuvan siitä mitä ovat järjestelmäprojektit, mitkä ovat niiden kompastuskivet ja niiden mahdollisuudet. Työssä ei ole ollut minulle tärkeätä vain itse ohjelmointitaidon kartuttaminen, vaan myös yleisesti systeemityöskentely ja tietojärjestelmien hankintaprosessi. Onnekseni minulla on ollut tämän projektin kanssa samaan aikaan erittäin kattava kurssi liittyen yrityksen palveluiden ja järjestelmien hankintaan, jonka oppeja olen voinut peilata projektiini syvällisesti tämän työn aikana. LogiDan myötä voin sanoa nyt toimittaneeni ensimmäisen monista tietojärjestelmistä, joita tulen vielä toteuttamaan.

LÄHTEET

Chapman Steven: What Is JavaScript? Viitattu 9.4.2013. Saatavissa: http://javascript.about.com/od/reference/p/javascript.htm

De Silva Mallin 2010: What is LINQ? Viitattu 15.4.2013. Saatavissa: http://notesofgenius.com/what-linq/

HTML.net: What is CSS? Viitattu 9.4.2013. Saatavissa: http://www.html.net/tutorials/css/lesson1.php

Rouse Margaret 2005: HTML (Hypertext Markup Language). Viitattu 9.4.2013. Saatavissa: http://searchsoa.techtarget.com/definition/HTML

Sininen Meteoriitti: Vaatimusmäärittelyt. Viitattu 15.4.2013. Saatavissa: http://www.meteoriitti.com/fi-FI/palvelut/konsultointipalvelut/vaatimusmaarittelyt/

Mehra Puran 2009: What is ADO.NET? Viitattu 15.4.2013. Saatavissa: http://www.c-sharpcorner.com/uploadfile/puranindia/what-is-ado-net/

Microsoft Developer Network: ASP.NET AJAX Overview. Viitattu 12.4.2013. Saatavissa: http://msdn.microsoft.com/en-us/library/bb398874(v=vs.90).aspx

Microsoft Developer Network: Introduction to Membership. Viitattu 12.4.2013. Saatavissa: http://msdn.microsoft.com/en-us/library/yh26yfzy(v=vs.100).aspx

Microsoft Developer Network: LINQ to SQL. Viitattu 15.4.2013. Saatavissa: http://msdn.microsoft.com/en-us/library/bb386976.aspx

Microsoft Developer Network: Overview of ADO.NET. Viitattu 15.4.2013. Saatavissa: http://msdn.microsoft.com/en-us/library/h43ks021(v=vs.71).aspx

Paakki Jukka 2011: Ohjelmistojen vaatimusmäärittely. Viitattu 15.4.2013. Saatavissa: http://www.cs.helsinki.fi/u/paakki/Vaatimus-11-Luentokalvot-1.pdf

Pan Jintao 1999: Software Testing. Viitattu 19.4.2013. Saatavissa http://www.ece.cmu.edu/~koopman/des_s99/sw_testing/

W3Schools.com: ASP.NET Web Forms. Viitattu 15.4.2013. Saatavissa: http://www.w3schools.com/aspnet/aspnet_intro.asp#gsc.tab=0