



VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Tomas Lepistö

SCALA-JÄRJESTELMÄÄN INTEG-
ROITU WEB-APPLIKAATIO TIETOJEN
SYÖTTÄMISTÄ VARTEN

Tekniikka ja liikenne
2014

ALKUSANAT

Tämä opinnäytetyö on tehty Vaasan ammattikorkeakoulun tietotekniikan koulutusohjelman päättötyönä kevään 2014 aikana eCraft Oy Ab nimiselle yritykselle.

Opinnäytetyön valvojana on toiminut yliopettaja Timo Kankaanpää ja yhteyshenkilönä eCraftilla liikekehityspäällikkö Anders Tåg. Toteutuksessa minua opastivat sovellusarkkitehti Thomas Raivio sekä ohjelmistoinsinööri Mikael Palmujoki.

Haluan kiittää Timo Kankaanpäää, Anders Tågia, Thomas Raiviota sekä Mikael Palmujokea avusta opinnäytetyön tekemisen aikana.

20.4.2014 Vaasa

Tomas Lepistö

TIIVISTELMÄ

Tekijä	Tomas Lepistö
Opinnäytetyön nimi	Scala-järjestelmään integroitu web-aplikaatio tietojen syöttämistä varten
Vuosi	2014
Kieli	suomi
Sivumäärä	33
Ohjaaja	Timo Kankaanpää

Tämän opinnäytetyön tarkoituksena oli suunnitella ja toteuttaa eCraft Oy:n yritysasiakkaalle selainpohjainen ostotilauksien vastaanotto-sovellus. Sovellus tulee korvaamaan käytössä olevan iScala-pohjaisen järjestelmän, joka on käyttäjien mielestä liian monimutkainen käyttää.

Sovellus toteutettiin käyttäen Microsoftin kehittämää ASP.NET MVC-sovelluskehystä. Toimintojen toteuttamisen apuna käytettiin AJAX- sekä jQuery-kirjastoja. Taulukoiden esittämiseen käytettiin DevExpressin-kehittämiä työkaluja, joiden avulla taulukoihin saatiin paljon toiminnollisuuksia sekä visuaalista näyttävyyttä.

Työn tuloksena syntyi toimiva kokonaisuus, joka täytti kaikki vaatimusmäärittelyssä asetetut vaatimukset. Raportin kirjoitushetkellä sovellus ei ollut vielä asiakkaan käytössä, mutta sovellus tullaan ottamaan käyttöön, kun asiakkaan palvelimen testiympäristö saadaan käyttöön.

ABSTRACT

Author	Tomas Lepistö
Title	Web-application for Feeding Information, Integrated in Scala System
Year	2014
Language	Finnish
Pages	33
Name of Supervisor	Timo Kankaanpää

The purpose of this thesis was to design and implement a browser-based purchase order delivery application for eCraft's corporate customer. The new application will replace the existing iScala-based system, which is too complicated to use according to the users.

The application was implemented by using the ASP.NET MVC framework from Microsoft. AJAX and jQuery libraries were also used to create required features to the application. In order to make application more functional and visually impressive, DevExpress tools were used.

As a result of this thesis, a fully functional application was developed. The application meets all the requirements that were set in the quality function deployment. At the moment the application has not yet been put into operation but will be as soon as the client's testing environment is ready.

LYHENTEET JA MERKINNÄT

ASP.NET	<i>Active Server Pages .NET</i> , web-ohjelmistokehys
MVC	<i>Model-View-Controller</i> , ohjelmistoarkkitehtuurityyli
JavaScript	Komentosarjakieli
SQL	<i>Structured Query Language</i> , ohjelmointikieli
XML	<i>Extensible Markup Language</i> , merkintäkieli
AJAX	<i>Asynchronous JavaScript And XML</i> , nopeiden ja dynaamisten Internet-sivustojen luomiseen käytettävä tekniikka
JQuery	JavaScript-kirjasto
Tagi	Käytetään tiedon merkitsemiseen XML- ja HTML-merkintäkielissä
Mock-up	Luonnos
QFD	<i>Quality function deployment</i> , asiakkaan tarpeet kartoittava menetelmä
DevExpress	Ohjelmistokehys
JSON	<i>JavaScript Object Notation</i> , tiedonsiirtomuoto
Olio	Luokan ilmentymä
Scala	Toiminnanohjausohjelma

SISÄLLYS

ALKUSANAT

TIIVISTELMÄ

ABSTRACT

LYHENTEET JA MERKINNÄT

1	JOHDANTO	4
2	ECRAFT OY AB	5
3	KÄYTETYT OHJELMISTOT JA TEKNIIKAT	6
	3.1 Microsoft Visual Studio	6
	3.2 ASP.NET MVC	6
	3.3 Razor View Engine	8
	3.4 JavaScript	9
	3.4.1 AJAX	9
	3.5 SQL	10
	3.6 XML	10
4	VASTAANOTTO SOVELLUKSEN SUUNNITTELU JA MÄÄRITTELY	11
	4.1 Lähtökohta	11
	4.2 Sovelluksen määrittely	11
	4.2.1 Vaatimusmäärittely	13
	4.2.2 Käyttötapauskaavio	14
	4.3 Sovelluksen suunnittelu	15
	4.3.1 Sekvenssikaavio	15
	4.3.2 Luokkakaavio	18
5	VASTAANOTTO SOVELLUKSEN TOTEUTUS	20
	5.1 Toteutukseen käytetyt tekniikat	20
	5.2 Tilauksen haku	21
	5.3 Dynaamisesti päivitettävä tekstikentän taustaväri	23
	5.4 Syötettyjen arvojen kerääminen taulukosta	26
	5.5 Tietojen tallentaminen tiedostoon	28
	5.6 Tilauksen tulostaminen	29

6	TESTAUS.....	30
6.1	Testausympäristö	31
7	YHTEENVETO	32
	LÄHDELUETTELO.....	33

KUVIO – JA TAULUKKOLUETTELO

Kuvio 1. MVC-arkkitehtuurityylin toimintaperiaate. /3/.....	7
Kuvio 2. Yksinkertainen esimerkki Razor-syntaksista.....	9
Kuvio 3. Ostotilauksen vastaanottosivun mock-up.	12
Kuvio 4. Avointen ostotilausten selaussivun mock-up.	12
Kuvio 5. Vastaanotto-sovelluksen käyttötapa-kaavio.	15
Kuvio 6. Tilauksen vastaanottotapahtuman sekvenssikaavio.....	16
Kuvio 7. Tilautustietojen muutostapahtuman sekvenssikaavio.....	17
Kuvio 8. Avoimien tilauksien selaustapahtuman sekvenssikaavio.	18
Kuvio 9. Käsitetasolla tehty luokkakaavio.	19
Kuvio 10. Tilauksen hakulomake.	21
Kuvio 11. Tilauksen suodatusehtojen asetus taulukkoon.....	22
Kuvio 12. Hakutulosten näyttäminen taulukossa.	22
Kuvio 13. Dynaamisesti päivitettävän tekstikentän alustus.	24
Kuvio 14. Kahden eri solun arvojen vertailu.....	25
Kuvio 15. Tekstikentän värienvaihto.	25
Kuvio 16. Muutettujen arvojen säilyttäminen käyttäjän puolella.....	26
Kuvio 17. Muutettujen rivien vieminen palvelimelle.....	26
Kuvio 18. Vastaanotettujen rivien vahvistusnäkyminen.	27
Kuvio 19. HtmlDataCellPrepared-asetuksen käyttöönotto.....	28
Taulukko 1. Vastaanotto-sovelluksen QFD.	14

1 JOHDANTO

Nykyaikana useimmissa yrityksissä on käytössä jonkinlainen tietojärjestelmä, jolla pyritään tehostamaan yrityksen toimintaa. Varsinkin suurien yritysten toimintojen hallitseminen olisi todella vaivalloista ilman nykyaikaisia järjestelmiä.

Lisäominaisuuksien lisääminen jo käytössä oleviin tietojärjestelmiin on monesti todella kovan työn takana. Tällöin on helpompaa tehdä tehtävää varten pienempi, järjestelmästä erillään oleva työkalu, ja integroida se järjestelmään.

Tämän opinnäytetyön tavoitteena on kehittää selainpohjainen sovellus ostotilauksien vastaanottamiseen. Sovelluksen on tarkoitus korvata käytössä oleva iScala-pohjainen toteutus, joka on käyttäjien mielestä liian hankala ja hidas käyttää.

Opinnäytetyön toimeksiantajana toimii eCraft Oy Ab. Opinnäytetyön suunnittelu ja kehitys tehdään eCraftin antamalla laitteilla heidän Palosaaren toimipisteessään, mutta kirjallinen osuus kirjoitetaan vapaa-ajalla.

2 ECRAFT OY AB

eCraft Oy Ab on vuonna 1999 perustettu suomalainen asiantuntijayritys, jossa työskentelee yli 100 asiantuntijaa. eCraftilla on toimipisteet Espoossa, Vaasassa sekä Malmössä. /1/

eCraftin periaatteena on tehdä yritysten toiminnanohjausjärjestelmistä innostavampia sekä helppokäyttöisempiä. eCraft tarjoaa yrityksille muokattuja asiakashallinta-, Business Intelligence- ja Smart Apps-ratkaisuja. eCraftin tunnetuimpia asiakkaita ovat mm. Altia, Ifolor, Kiinteistömaailma ja Fenno Medical. /1/

3 KÄYTETYT OHJELMISTOT JA TEKNIIKAT

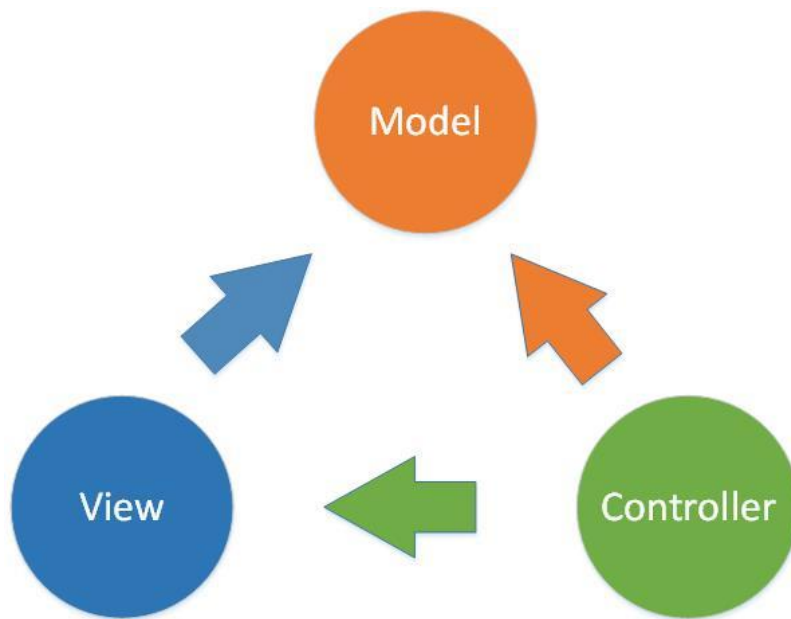
Tässä luvussa esitellään opinnäytetyön toteuttamisen aika käytettyjen ohjelmistojen ja tekniikoiden käyttötarkoitukset sekä taustat.

3.1 Microsoft Visual Studio

Microsoft Visual Studio on Microsoftin kehittämä ohjelmankehitysympäristö. Visual studion avulla voi kehittää Windows Store-, työpöytä-, mobiili- ja ASP.NET web- sovelluksia sekä XML web-palveluita. Visual studion ohjelmointikielina voi käyttää mm. Visual Basic, Visual C#, Visual C++, Visual F# sekä JavaScriptiä. Web-sovellusten ja XML web-palveluiden kehittämistä on yksinkertaistettu käyttämällä .NET Framework-ohjelmistokomponenttikirjastoa. /2/

3.2 ASP.NET MVC

ASP.NET MVC-arkkitehtuurityyli jakaa sovelluksen kolmeen eri osaan: model eli malli, view eli näkymä, ja controller eli käsittelijä (**Kuvio 1.**). ASP.NET MVC-kehys tarjoaa vaihtoehdon ASP.NET Web-Forms-mallille MVC-pohjaisten web-sovellusten luomiseen. ASP.NET MVC-kehys on kevyt ja helposti testattava arkkitehtuuri, johon on integroitu jo olemassa olevat ASP.NET-ominaisuudet, kuten pääsivut ja rooleihin perustuva tunnistautuminen. /3/



Kuvio 1. MVC-arkkitehtuurityylin toimintaperiaate. /3/

- **Mallit**
Mallit ovat sovelluksen osia, jotka toteuttavat logiikan sovelluksen datan hallintaan. Useimmiten mallit hakevat ja säilyttävät tietoa tietokantaa hyödyntäen. /3/
- **Näkymät**
Näkymät ovat sovelluksen osia, jotka näyttävät sovelluksen käyttöliittymän. Tyypillisesti käyttöliittymä luodaan malliluokan datasta. /3/
- **Käsittelijät**
Käsittelijät ovat sovelluksen osia, jotka käsittelevät käyttäjän toiminnot, hoitavat mallin kanssa kommunikoinnin, ja valitsevat näytettävän näkymän. /3/

ASP.NET MVC 1.0 versio julkaistiin vuonna 2009 ja siitä on julkaistu uusi päivitetty versio noin vuoden välein. Tällä hetkellä uusin versio on vuoden 2014 alkupuolella julkaistu 5.1. /9/

3.3 Razor View Engine

Razor on merkintäsyntaksi, joka mahdollistaa palvelin puolen koodin sisällyttämisen Internet-sivustoille HTML-koodin sekaan (**Kuvio 2.**). Razor tukee Visual Basic- ja C#-ohjelmointikieliä.

Razor perustuu ASP.NET web-ohjelmistokehykseen ja se on suunniteltu web-sovellusten tekoon. Razorissa on perinteisen ASP.NET-merkinnän vahvuudet, mutta sitä on helpompi käyttää ja sen opettelu on helpompaa.

Palvelimen puolella ajettava koodi voi luoda sivustolle dynaamista sisältöä samalla kun sivustoa ladataan selaimen. Internet-sivua kutsuttaessa palvelin suorittaa palvelinpuolen koodin sivun sisällä ennen kuin se palauttaa sivun selaimelle. Suorittamalla koodi palvelimella voidaan tehdä monimutkaisiakin tehtäviä, kuten tietokantaa yhdistäminen. /4/

```
@{  
  var message = "Hello world!";  
}  
  
<h2>@message</h2>
```

Kuvio 2. Yksinkertainen esimerkki Razor-syntaksista.

3.4 JavaScript

JavaScript on Netscapen kehittämä oliopohjainen komentosarjakieli, jota käytetään yleisesti Internet-sivustoilla sekä palvelinsovelluksissa. Netscapen JavaScript pohjautuu ECMA-262 versio 3 standardiin, jota kutsutaan myös nimellä ECMAScript. /5/

Monesti ihmiset yhdistävät JavaScriptin ja Java-kielen toisiinsa, mutta näillä kahdella ei ole nimiensä lisäksi mitään tekemistä toistensa kanssa. /5/

Lyhyesti sanottuna JavaScript on dynaaminen oliopohjainen komentosarjakieli, jossa oliot pohjautuvat prototyyppeihin luokkien sijaan. Merkitätapa on tarkoituksellisesti tehty muistuttamaan Javaa ja C++ oppimisen helpottamiseksi. /5/

JavaScriptin avulla Internet-sivustoille voi lisätä dynaamista toiminnollisuutta esimerkiksi lisäämällä lomakekenttiin reaaliaikaisen virheentarkistuksen.

3.4.1 AJAX

AJAX on tekniikka, jonka avulla voidaan luoda nopeita ja dynaamisia Internetsivustoja. /6/

AJAX-tekniikkaa käyttämällä Internet-sivustoja voidaan päivittää asynkronisesti liikuttamalla pieni määrä tietoa sivuston ja palvelimen välillä. Tällä tavoin voidaan

päivittää vain osa sivun näkymästä, niin ettei koko sivua tarvitse ladata uudelleen.

/6/

3.5 SQL

SQL on standardoitu ohjelmointikieli relaatiotietokantojen hallitsemiseen. SQL-kielen avulla voidaan määritellä, ylläpitää ja hakea tietokannassa säilytettävää tietoa./7/

Suurin osa relaatiotietokantojen hallintaohjelmista, kuten MySQL, MS Access, Oracle, Sybase, Informix, postgres ja Microsoft SQL Server käyttävät SQL-ohjelmointikieltä tietokannoissaan. /7/

3.6 XML

XML on merkitsemiskieli, jonka avulla voidaan kuvailla dokumentin sisältöä ja sen rakennetta. XML on yksinkertaistettu versio SGML:stä ja sitä käytetään tiedon siirtämiseen Internetissä./8/

XML-merkintäkieli käyttää tiedon merkitsemiseen alku- ja lopputageja samaan tapaan kuin HTML-kieli. Tagien välissä olevaa tietoa kutsutaan elementiksi. Jokaisella XML-dokumentilla on yksi pääelementti, jonka sisällä on kaikki muut elementit. Elementeillä voi olla määriteltynä myös attribuutteja. Tämä mahdollistaa tehokkaan tavan siirtää tietoa eri sovellusten sekä järjestelmien välillä./8/

4 VASTAANOTTOSOVELLUKSEN SUUNNITTELU JA MÄÄRITTELY

Ostotilauksien vastaanottosovellus koostuu Internet-selaimessa toimivasta sovelluksesta, jonka avulla voidaan hakea ostotilauksen tiedot ja merkitä vastaanotettujen tuotteiden määrät, jonka jälkeen sovellus päivittää ostotilauksen tiedot Scala-järjestelmään. Tämän lisäksi sovelluksella voidaan selata avoimia ostotilauksia sekä muokata tilauksen tietoja.

4.1 Lähtökohta

Lähtökohtana työlle on tehdä uusi sovellus vanhan iScala-pohjaisen sovelluksen tilalle. Vanha sovellus on asiakkaan mielestä liian monimutkainen ja hidas käyttää.

Uuden selainpohjaisen vastaanottosovelluksen avulla ostotilausten vastaanottamisen merkitseminen Scala-järjestelmään tulee olemaan huomattavasti nopeampaa ja helpompaa.

4.2 Sovelluksen määrittelyt

Tässä osiossa kerrotaan sovelluksen toimintavaatimuksista ja siitä miten vaatimukset on määritelty,

Sovelluksen määrittelyssä on otettu huomioon asiakkaan kanssa käydyt keskustelut ohjelman vaatimuksista. Asiakkaan kanssa käytyjen keskustelujen perusteella on tehty luonnokset, joilla määritellään sovelluksen ulkoasu suuntaa antavasti (**Kuvio 3 ja 4**).

Vastaanotto	Avoimet Ostotilaukset	Päivämäärä: 6.3.2014
-------------	-----------------------	----------------------

Ostotilausno:

Toimittaja: 995764 Oy Toimittaja Ab
Varastopaikka: S10

Tilousrivinro	Tuotekoodi	Nimike	Hylly	Tilattu määrä	Vastaanotettu määrä
000010	DEFR1212AX	Lamppu, 12 W Kirkas	AB1	120	120 kpl
000020	AXC-1121	LED, 40 W	AB1	50	50 kpl
000030	BB1212	LED, 20 W kirkas	AB1	45	40 kpl
000040	12000XY	Varaos, XX	CC2	400	400 kpl
000070	XQ3311-01-1	CD4	CD4	400	400 kpl
000110	004EER-X	Pakkausmateriaali, 3 x 3 m - paperi	FR1	10	10 m
000120	AIR-002	Valaisin, vihreä - muovi	TT1	30	0 kpl
000130	AIR-001	Valaisin, sininen - muovi	XP1	30	0 kpl

Kuvio 3. Ostotilauksen vastaanottosivun mock-up.

Vastaanotto	Avoimet Ostotilaukset	Päivämäärä: 6.3.2014
-------------	-----------------------	----------------------

Tilausno:

AVOIMET OSTOTILAUKSET TOIMITTAJITTAIN:

▼ 123443 - Oy Toimittaja Ab

▼ 0011234553

Tilousrivinro	Tuotekoodi	Nimike	Tilattu määrä
000010	DEFR1212AX	Lamppu, 12 W Kirkas	120 kpl
000020	AXC-1121	LED, 40 W	50 kpl
000030	BB1212	LED, 20 W kirkas	45 kpl
000040	12000XY	Varaos, XX	400 kpl
000070	XQ3311-01-1	Varaos, XQ	400 kpl
000110	004EER-X	Pakkausmateriaali, 3 x 3 m - paperi	10 m
000120	AIR-002	Valaisin, vihreä - muovi	30 kpl
000130	AIR-001	Valaisin, sininen - muovi	30 kpl

▶ 0011234555
▶ 0011298810

▶ 123444 - Supplier Ltd
▶ 123445 - Oy Ostot Ab

Kuvio 4. Avointen ostotilausten selaussivun mock-up.

Lisäksi sovellukseen halutaan kolmas näkymä, jossa käyttäjä voi hakea tilauksen ja muokata olemassa olevan tilauksen tilattua tuotemäärää tai tilauksen saapumispäivämäärää.

4.2.1 Vaatimusmäärittely

Sovellukseen haluttavien toimintojen määrittelyä varten on tehty QFD. QFD:n avulla voidaan määrittää milloin sovellus täyttää sille asetetut vaatimukset. QFD koostuu kolmesta eri vaatimustasosta. Ensimmäisenä määritellään pakolliset toiminnot, jotka asiakas haluaa. Nämä ovat yleensä asiakkaan tyytyväisyyden kannalta tärkeimpiä toimintoja ja niiden pohjalta lähdetään rakentamaan koko sovellusta. Seuraavaksi määritellään odotetut toiminnot, joita asiakas ei ehkä osaa vaatia, mutta sovelluksen toimivuuden kannalta ovat olennaisia. Nämä vaatimukset liittyvät yleensä käytettäviin laitteisiin tai ohjelmiin. Viimeisenä määritellään toiminnot, jotka eivät ole sovelluksen käyttötarkoituksen kannalta olennaisia, mutta voivat parantaa asiakkaan tyytyväisyyttä sovellusta kohtaan huomattavasti.

Asiakkaan kanssa käytyjen keskustelujen perusteella päädyttiin seuraavanlaiseen vaatimusmäärittelyyn (**Taulukko 1.**):

Taulukko 1. Vastaanotto-sovelluksen QFD.

<p>Pakolliset (must have):</p> <ul style="list-style-type: none"> • Ostotilauksen haku tietokannasta ostotilausnumeron perusteella. • XML- tiedoston luonti Scalaan tallennusta varten. • Avointen tilausten selaamismahdollisuus. • Tilauksen tuotemäärän ja saapumispäivämäärän vaihtomahdollisuus
<p>Odotetut (should have):</p> <ul style="list-style-type: none"> • Sivuston toiminta selaimilla Microsoft Internet Explorer 11, Mozilla Firefox 28 & Google Chrome 34. • Syötettyjen arvojen tarkistaminen: <ul style="list-style-type: none"> ○ Käyttäjä ei voi syöttää negatiivisia arvoja. ○ Syöttökenttä muuttuu vihreäksi, kun vastaanotettu määrä on sama kuin tilattu määrä.
<p>Hienoa jos olisi (nice to have):</p> <ul style="list-style-type: none"> • Vastaanotettujen tuotemäärien tulostusmahdollisuus paperille.

Edellä mainittujen vaatimusten lisäksi sovelluksen haluttiin myös olevan mahdollisimman helppokäyttöinen.

4.2.2 Käyttötapauskaavio

Sovelluksen määrittelyä varten tehdään myös käyttötapauskaavio. Käyttötapauskaavion avulla voidaan hahmottaa paremmin mitä kaikkea käyttäjän pitää pystyä tekemään sovelluksen avulla. Vastaanotto-sovelluksella on yksi käyttäjä, joka on yrityksen työntekijä. Käyttäjän tunnistamiseen käytetään Windows-autentikointia. Käyttäjä voi hakea ostotilausnumerolla ostotilauksen ja kirjata ostotilauksen vastaanotettujen tavaroiden määrät. Käyttäjä voi myös selata sekä hakea avoimia ostotilauksia. Käyttäjä voi hakea tilauksen ja käsitellä tilauksen tilattua tuotemäärää tai

saapumispäivämäärää. Käyttäjän täytyy myös pystyä tulostamaan vastaanotettujen tai muutettujen rivien tiedot paperille (**Kuvio 5.**).



Kuvio 5. Vastaanotto-sovelluksen käyttötapauskaavio.

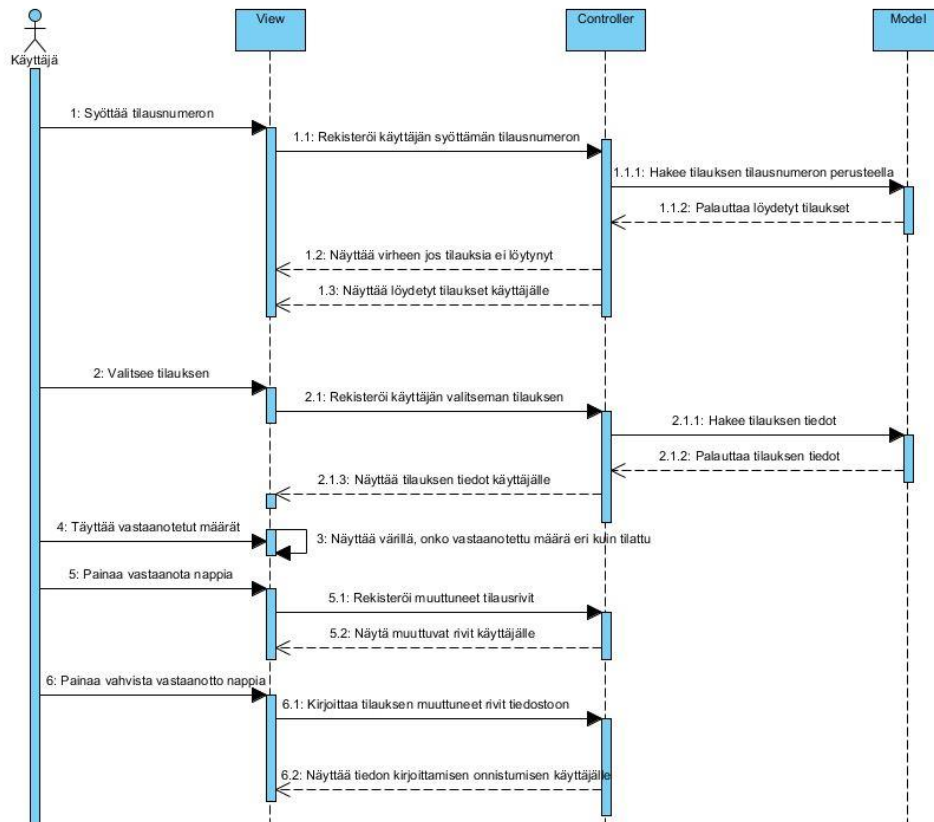
4.3 Sovelluksen suunnittelu

4.3.1 Sekvenssikaavio

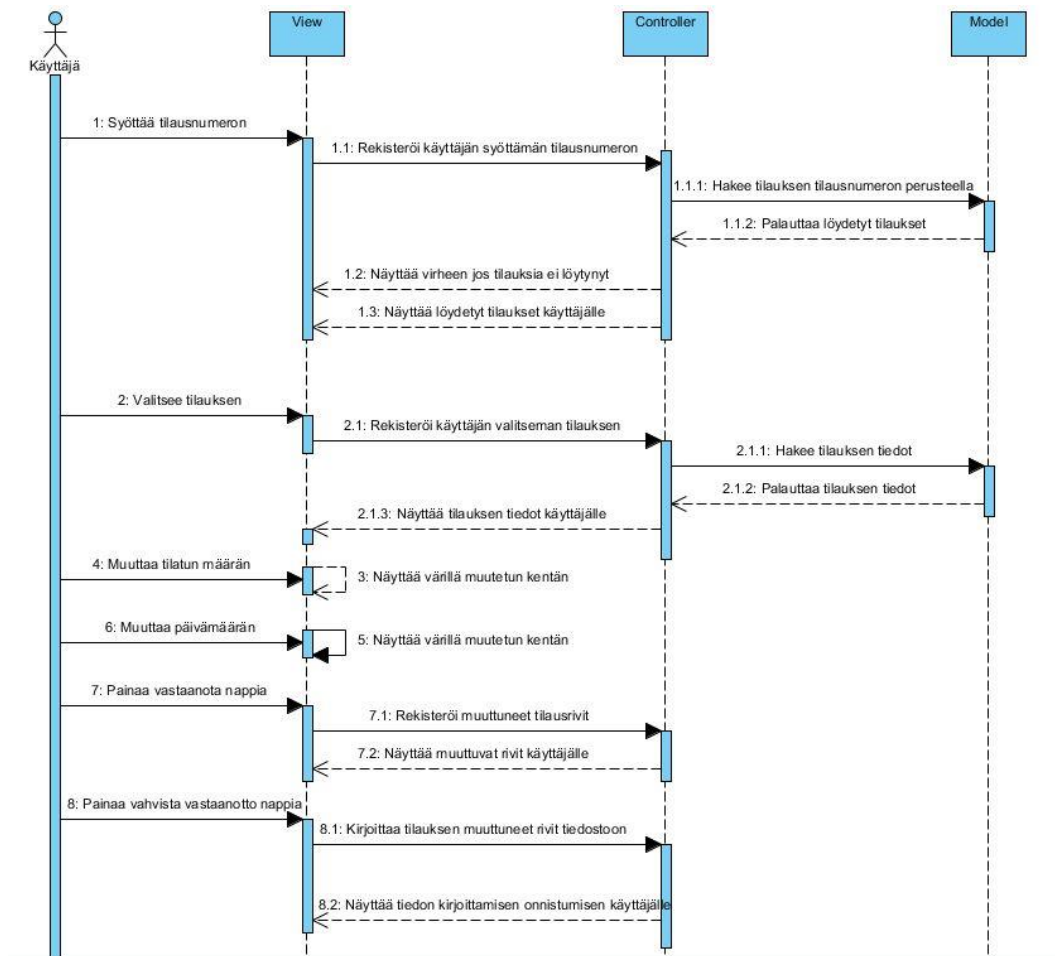
Sovelluksen toimintavaiheita kuvaamaan on luotu sekvenssikaavio. Sekvenssikaavion avulla voidaan suunnitella sovelluksen eri toimintojen tapahtumajärjestystä.

Tilauksen vastaanottotapahtumasta ja muutostapahtumasta on tehty sekvenssikaaviot (**Kuvio 6 ja 7**). Tapahtumien kulut ovat samankaltaisia. Käyttäjän syöttämän tilausnumeron perusteella tietokannasta haetaan hakehtoihin sopivat tilaukset ja ilmoitetaan käyttäjälle haun tulokset. Tämän jälkeen käyttäjä valitsee haluamansa tilauksen ja tilauksen tiedot näytetään käyttäjälle. Käyttäjä syöttää vastaanotetut tuotemäärät tekstikenttiin, jolloin tekstikentän väri vaihtuu tuotemäärästä riippuen. Syötettyään haluamansa arvot käyttäjä painaa nappia, jolloin käyttäjälle avataan vahvistusnäkyvä, josta näkyy muuttuvat rivit. Tämän jälkeen käyttäjä vahvistaa

tilauksen muutokset, jonka jälkeen ohjelma kirjoittaa muuttuneet rivit XML-tiedostoon ja ilmoittaa käyttäjälle kirjoituksen onnistumisesta.

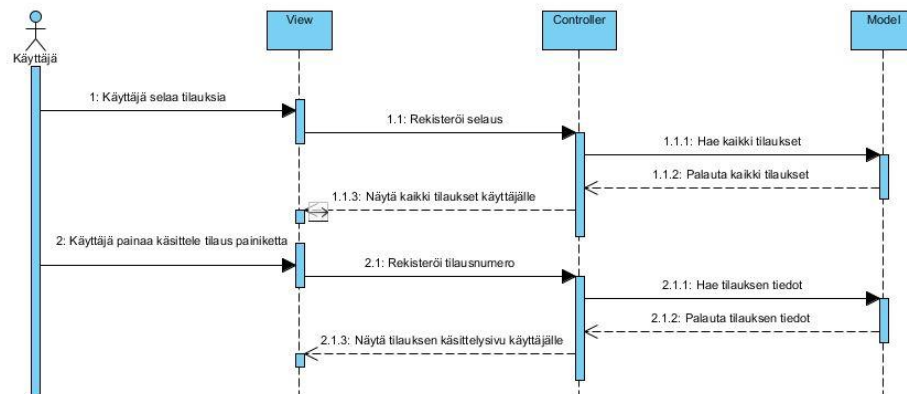


Kuvio 6. Tilauksen vastaanottotapahtuman sekvenssikaavio.



Kuvio 7. Tilaustietojen muutostapahtuman sekvenssikaavio.

Ostotilausten selaamistapahtuma kuvataan seuraavassa kuviossa (**Kuvio 8.**). Käyttäjälle näkyviin haetaan tietokannasta kaikki avoinna olevat ostotilaukset. Käyttäjän painaessa vastaanota- tai muuta-painiketta jonkin tilauksen kohdalla, rekisteröidään valittu tilausnumero ja haetaan tilauksen tiedot tietokannasta. Tämän jälkeen käyttäjälle näytetään tilauksen tiedot.



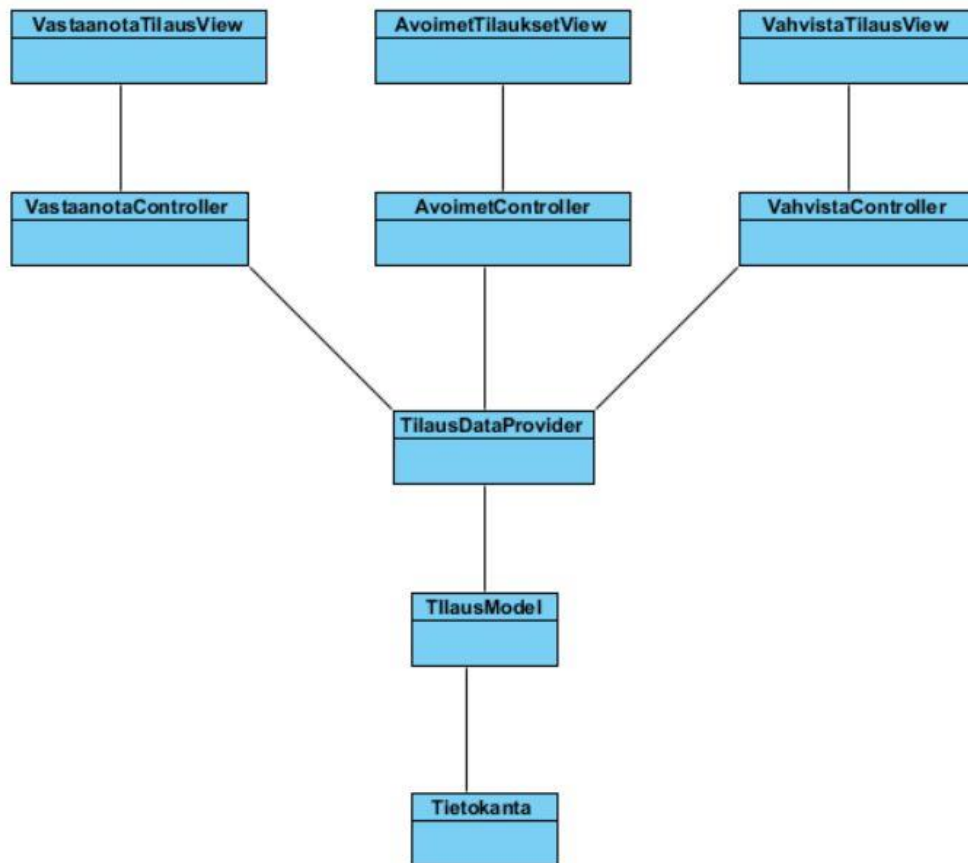
Kuvio 8. Avoimien tilauksien selaustapahtuman sekvenssikaavio.

4.3.2 Luokkakaavio

Sovelluksen luokkia kuvaamaan on luotu luokkakaavio. Luokkakaavion avulla voidaan näyttää järjestelmän sisältämät luokat sekä luokkien väliset suhteet.

Luokkakaavio on tehty käsitetasolla, koska projektin luokkien ominaisuudet tulevat muuttumaan työn edetessä. Käsitetason luokkakaavion avulla sovelluksen rakenteen ymmärtäminen helpottuu (**Kuvio 9**).

Sovelluksen näkymät on jaettu kolmeen eri ryhmään ja jokaisella ryhmällä on oma controller-luokka, joka käsittelee käyttäjän näkymissä tekemät pyynnöt. Kaikki controller-luokat hakevat tiedot TilausDataProvider-luokasta, joka hakee tiedot TilausModel-luokan avulla tietokannasta.



Kuvio 9. Käsitetasolla tehty luokkakaavio.

5 VASTAANOTTOSOVELLUKSEN TOTEUTUS

Tässä osiossa käydään läpi miten sovellus päätettiin toteuttaa ja mitä eri vaiheita sen toteuttamiseen liittyi.

Toteutuksessa tehtiin paljon asioita kokeilemalla eri tekniikoita, sillä toteutusvaiheessa ei ollut aina varmuutta, pystytäänkö toiminto tekemään kyseisellä tekniikalla.

5.1 Toteutukseen käytetyt tekniikat

Sovellus päätettiin toteuttaa uutta ASP.NET MVC 5-tekniikkaa hyödyntäen. MVC:n avulla sovellusta on helppo hallita, koska toiminnot on jaettu selkeästi omiin ryhmiin. Sovelluksen näkymät pyrittiin laittamaan omiin kansioihin, jotta projektin hallinta olisi helpompaa. Jokaiselle sivulle tehtiin oma controller-luokka, jotta funktioiden kutsuminen pysyisi loogisena.

Sovelluksen tilaustietojen näyttämiseen käytettiin DevExpressin MVC-työkalua. DevExpress on .NET-rajapinnalle suunniteltu työkalu, jonka avulla sovelluksiin voidaan luoda näyttäviä toimintoja kätevästi. DXperience-työkalusta on saatavilla 30 päivän ilmainen kokeiluversio, jonka jälkeen ohjelmasta tulee maksullinen.

Sovellusta lähdettiin rakentamaan Visual Studion tarjoaman ASP.NET MVC-malliprojektin pohjalle. Malliprojekti sisälsi automaattisesti tarvittavat luokat sekä toiminnollisuudet sivuston perustoimintojen suorittamiseen.

5.2 Tilauksen haku

Tilauksen haku toteutettiin käyttämällä `Html.BeginForm`-toimintoa (**Kuvio 10**). Se sisältää tekstikentän, johon käyttäjä voi syöttää ostotilausnumeron, toimittajan nimen tai toimittajan numeron. Haku onnistuu myös hakemalla vain osaa nimestä tai numerosta, sillä hakutoiminto palauttaa kaikki tulokset, jotka sisältävät kyseisen hakusanan tai numeron.

Lomake sisältää piilotetun `source`-kentän, jonka avulla kerrotaan seuraavalle näkymälle, miltä hakusivulta pyyntö tulee. Tätä tarvitaan, koska samaa metodia kutsutaan myös toimitusvahvistusnäkyvän tilaushaussa ja sivun ulkonäkö määrittellään sen mukaan, miltä sivulta pyyntö tulee.

Painamalla Hae-näppäintä kutsutaan `OpenOrdersController`-luokan `SetFilterByID`-metodia.

```
@using (Html.BeginForm("SetFilterByID", "OpenOrders"))
{
    <input type="hidden" value="postfromreceive" name="source" />
    <input id="OrderNumber" name="OrderNumber" type="text" />
    <input type="submit" value="Hae" />
}
```

Kuvio 10. Tilauksen hakulomake.

`SetFilterByID`-metodissa lomakkeeseen syötetty arvo haetaan lomakkeesta ja asetetaan hakuehdoksi. Hakuehto, tilausnumero sekä `source` tallennetaan muistiin käyttämällä `ViewData`-ominaisuutta. `ViewData`n avulla voidaan asettaa muuttujia muistiin, ja käyttää niitä seuraavassa näkymässä. Lopuksi `SetFilterByID`-metodi ohjaa käyttäjän `OpenOrders`-näkymään.

`OpenOrders`-näkymässä sivun ulkoasu määräytyy `ViewData`ssa olevan `Source`-muuttujan perusteella. Hakutulokset näytetään `Html.Partial`-toiminnolla, jonka parametrina on näkymä `_OpenOrdersGridViewPartial`. `Html.Partial`-toiminnon avulla

sivuun voidaan liittää osittaisia näkymiä. `_OpenOrdersGridViewPartial`-näkylässä luodaan `Html.DevExpress().GridView`-taulukko. Jotta taulukossa näkyisi vain haetut tilaukset, käytetään `PreRender`-ominaisuutta (**Kuvio 11**). `PreRender`in avulla näkymää voidaan muokata ennen kuin se näytetään käyttäjälle. `PreRender`issä taulukon `FilterExpression`-ominaisuuden arvoksi haetaan aiemmin tallennettu hakuehto.

```
settings.PreRender = (sender, e) =>
{
    if (ViewData["FilterExpression"] != null)
    {
        ASPxGridView searchGrid = (ASPxGridView)sender;
        searchGrid.FilterExpression = ViewData["FilterExpression"].ToString();
    }
};
```

Kuvio 11. Tilauksen suodatusehtojen asetus taulukkoon.

Tämän jälkeen käyttäjälle näytetään haun tulokset taulukkomuodossa (**Kuvio 12**). Käyttäjä voi ottaa haluamansa tilauksen käsittelyyn vastaanota-painikkeesta.

Vastaanota ostotilaus

Haun 123 tulokset:

	Toimittajan nimi	Toimittajanro	Rivinro	Tuotekoodi	Kuvaus	Tilattu	Mittayksikkö
⊖	Tilausno: 222123						
Vastaanota	Pro Plastic	6666	000020	QWERTY	Muovilankaa	100	m
⊖	Tilausno: 333						
Vastaanota	Toimittaja3	1234	000010	WWWE-33	Lampunkanta	50	kpl
⊖	Tilausno: 444						
Vastaanota	Toimittaja3	1234	000110	FFFF-311	Johto	99	m
⊖	Tilausno: 555						
Vastaanota	Toimittaja3	1234	000010	FFGR-2323	LED 25W	4000	kpl

Kuvio 12. Hakutulosten näyttäminen taulukossa.

Vastaanota-painiketta painamalla kutsutaan `CustomButtonClick()` JavaScript-funktiota, joka poimii talteen käsiteltävän rivin avaimen. Rivin avaimen arvo on sama

kuin tilausnumero. Tilaus otetaan käsittelyyn asettamalla `window.location.href`-toiminnon arvoksi käsiteltävän rivin tilausnumero. Tämän jälkeen funktio ohjaa käyttäjän tilausnumeron perusteella osoitteeseen `/Receive/ReceiveGridViewEdit/123`.

5.3 Dynaamisesti päivitettävä tekstikentän taustaväri

Yksi projektin haastavimmista toiminnoista toteuttaa oli tekstikentän taustaväriin vaihtaminen käyttäjän syöttäessä arvoja. Vastaanotetun tuotemäärän ollessa sama kuin tilattu tuotemäärä, tuli tekstikentän taustaväriin muuttua vihreäksi. Tuotemäärän ollessa eri, täytyi taustaväriin muuttua punaiseksi, jotta käyttäjä huomaisi poikkeavuuden.

Toimintoa varten vastaanottotaulukon luontivaiheessa täytyi luoda Toimitettu-sarakkeeseen `GridViewDataItemTemplateContainer`. `GridViewDataItemTemplateContainer`in avulla pystytään muokkaamaan yksittäisen solun ulkonäköä taulukossa (**Kuvio 13.**).

```

settings.Columns.Add(column =>
{
    // Sarakkeen nimi
    column.FieldName = "Toimitettu";

    // Mahdollistaa sarakkeen muokkaamisen
    column.ReadOnly = false;

    // Sarakkeessa näkyvä nimi
    column.Caption = "Vastaanotettu Määrä";

    // Sallii mallipohjan asettamisen
    column.SetDataItemTemplateContent(diTemplate =>
    {
        // Mallipohjan, jota käytetään solujen päivittämiseen
        GridViewDataItemTemplateContainer container = diTemplate as GridViewDataItemTemplateContainer;

        // Tekstikenttä
        Html.DevExpress().TextBox(txtSettings =>
        {
            // Asettaa jokaiselle tekstikentälle uniikin nimen
            txtSettings.Name = string.Format("txt_{0}", diTemplate.VisibleIndex);

            // Käyttäjän kirjoittaessa kenttään kutsutaan OnValueChanged funktiota
            // Parametreinä viedään muokattavan kentän avainarvo, sarakkeen nimi ja rivin indeksi
            txtSettings.Properties.ClientSideEvents.TextChanged =
                string.Format("function(s, e) {{OnValueChanged(s, e, '{0}', '{1}', '{2}')}}",
                    container.KeyValue.ToString(), container.Column.FieldName, diTemplate.VisibleIndex);
        }).Render();
    });
});

```

Kuvio 13. Dynaamisesti päivitettävän tekstikentän alustus.

OnValueChanged-funktiossa poimitaan syötetty määrä sekä muokattavan rivin indeksi. Tämän jälkeen kutsutaan taulukon GetRowValues-funktiota, jonka avulla haetaan muokattavan rivin Tilattu- sarakkeessa oleva arvo. Tilattu-sarakkeen arvo viedään OnGetRowValues-funktioon, jossa suoritetaan tekstikentän taustaväriin vaihto (**Kuvio 14**).

```

function OnGetRowValues(Value) {

    // Hakee tekstikentän solun sisällä olevan taulukon ja editorin
    var table = document.getElementById('txt_' + visibleIdx);
    var editor = document.getElementById('txt_' + visibleIdx + '_I');
    var color = '';

    // Jos vastaanotettu- ja toimitettumäärä on eri
    if (enteredValue > Value || enteredValue < Value)
        // Punainen
        color = '#FFAAAA';
    // Samat määrät
    else if (enteredValue == Value) {
        // Vihreä
        color = '#09EB60';
    }

    // Asettaa värit
    table.style.backgroundColor = color;
    editor.style.backgroundColor = color;
}

```

Kuvio 14. Kahden eri solun arvojen vertailu.

Tämän jälkeen tekstikentän taustaväri päivittyy saadun arvon mukaan (**Kuvio 15.**).

Tilausnumero: 1111111111

Toimittaja: Toimittaja1 9999999

Varastopaikka: S10

Rivinro	Tuotekoodi	Kuvaus	Tilattu	Mittayksikkö	Vastaanotettu Määrä	Mittayksikkö
000010	DEFR12	Lamppu kirkas	120	kpl	120	kpl
000020	BBB23	Lamppu himmeä	300	kpl	250	kpl
000030	QQ-2333	Valaisintarvike	233	kpl	233	kpl

Tallenna muutokset

Kuvio 15. Tekstikentän värienvaihto.

5.4 Syötettyjen arvojen kerääminen taulukosta

Kun käyttäjä syöttää tekstikenttään jonkin arvon, kutsutaan `OnValueTextChange`-funktiota. Käyttäjän syöttämä arvo otetaan talteen ja tallennetaan `changedValues`-kokoelmaan, jonka jälkeen tarkistetaan `changedValues.hasOwnProperty`-metodilla, löytyykö kokoelmasta ennestään muokattavaa avainarvoa (**Kuvio 16.**).

```
if (changedValues.hasOwnProperty(keyValue))
    currentRow = changedValues[keyValue];
else {
    currentRow = {};
    changedValues[keyValue] = currentRow;
}
currentRow[fieldName] = s.GetValue();
```

Kuvio 16. Muutettujen arvojen säilyttäminen käyttäjän puolella.

Käyttäjän painaessa tallenna-painiketta, kutsutaan `OnReceiveButtonClick`-funktiota, jonka parametriksi annetaan `ReceiveController`issa olevan `SaveReceiveData`-metodin osoite. Datan viemiseksi palvelimen puolelle käytetään AJAX JSON-tekniikkaa (**Kuvio 17.**).

```
function OnReceiveButtonClick(s, e, url_par) {
    id = @ViewContext.RouteData.Values["id"]
    changedValues['orderNumber'] = id;
    s.SetEnabled(false);
    $.ajax({
        type: 'POST',
        url: url_par,
        data: JSON.stringify(changedValues),
        dataType: 'json'
    }).done(CallbackComplete).fail(OnCallbackError);
}
```

Kuvio 17. Muutettujen rivien vieminen palvelimelle.

JSON lähettää datan palvelimelle merkkijonona. Palvelimella merkkijono haetaan `HttpRequest`-luokasta, jonka jälkeen merkkijonosta erotellaan avaimet ja avainten sisältämät arvot. Saatu data asetetaan `changedValues`-kokoelmaan.

Tämän jälkeen `changedValues`-kokoelma viedään `UpdateValues`-metodiin, jossa kokoelmasta poimitaan tilausnumero ja toimitetut tuotemäärät. Tietokannasta haetaan tilausnumeroa ja –riviä vastaava olio ja päivitetään olion toimitettu tuotemäärä. Toimitetulla tuotemäärällä päivitetty olio tallennetaan `OrdersToUpdate`-listaan, jonka jälkeen lista tallennetaan istuntoon nimellä `changedreceiveorders` myöhempää käyttöä varten. Tämän jälkeen käyttäjä ohjataan vahvistusnäkympään (**Kuvio 18**). Vahvistusnäkympässä istunnosta haetaan `changedreceiveorders`-lista ja asetetaan se näkympään taulukossa.

Vahvista vastaanotetut rivit

Tilausnumero: 1111111111

Toimittaja: Toimittaja1 9999999

Varastopaikka: S10

Rivinro	Tuotekoodi	Kuvaus	Tilattu	Toimitettu	Selite
000010	DEFR12	Lamppu kirkas	120	120	kpl
000020	BBB23	Lamppu himmeä	300	250	kpl
000030	QQ-2333	Valaisintarvike	233	233	kpl

Kuvio 18. Vastaanotettujen rivien vahvistusnäkympä.

Taulukossa tilatun ja toimitetun määrän eroavaisuuksien visualisointi on toteutettu `HtmlDataCellPrepared`-asetuksella (**Kuvio 19**). Jos kahden eri solun arvot ovat tiedossa taulukon luontivaiheessa, voidaan `HtmlDataCellPrepared`-toiminnon avulla vaihtaa solun taustaväriä vertailemalla näiden solujen arvoja keskenään.


```

settings.HtmlDataCellPrepared = (s, e) =>
{
    // Tarkista löytyykö Tilattu- ja Toimitettu soluista arvoja.
    if ((e.GetValue("Tilattu") != null) && (e.GetValue("Toimitettu") != null))
    {
        // Tallenna arvot muuttujiin
        string target = Convert.ToString(e.GetValue("Tilattu"));
        string actual = Convert.ToString(e.GetValue("Toimitettu"));

        if (e.DataColumn.FieldName.Equals("Toimitettu"))
        {
            // Jos tilattumäärä on sama kuin vastaanotettumäärä
            if (target.Equals(actual))
            {
                // Vaihda taustaväri vihreäksi
                e.Cell.BackColor = System.Drawing.ColorTranslator.FromHtml("#09EB60");
            }
            else
            {
                // Vaihda taustaväri punaiseksi
                e.Cell.BackColor = System.Drawing.ColorTranslator.FromHtml("#FFAAAA");
            }
        }
    }
};

```

Kuvio 19. HtmlDataCellPrepared-asetuksen käyttöönotto.

5.5 Tietojen tallentaminen tiedostoon

Tietojen päivittäminen Scala-järjestelmään voidaan tehdä kirjoittamalla vastaanotetun tilauksen tiedot sekä muuttuneet rivit xml-tiedostoon. Kun xml-tiedosto on luotu, käy Scala-lukemassa muuttuneet rivit ja päivittää tiedot omaan järjestelmäänsä.

Käyttäjän painaessa tallenna muutokset-näppäintä haetaan tilauksen tiedot istunnosta ja ohjataan lista CreateReceiveXML-metodiin. CreateReceiveXML-metodi kirjoittaa tilauksen tiedot xml-tiedostoon käyttämällä XDocument-luokkaa. Käyttäjälle palautetaan tieto tallennuksen onnistuvuudesta TempData:n avulla. TempData säilyttää tietoa vain yhden http-pyynnön ajan, joten se sopii hyvin ilmoittamaan käyttäjälle tehtävien onnistumisesta.

5.6 Tilauksen tulostaminen

Tilauksen tulostamisen toteuttaminen oli todella yksinkertaista. Käyttäjän painaessa tulosta-näppäintä kutsutaan javascript-funktiota PrintDiv. Funktio suorittaa komennon `window.print()`, joka avaa näkyvissä olevan näkymän tulostusdialogiin.

Oletuksena tulostetaan kaikki näkyvissä olevat sivun elementit, mutta elementti voidaan piilottaa tulostusnäkyvästä asettamalla luokaksi `class="hidden-print"`.

6 TESTAUS

Sovellustestauksen tarkoituksena on testata sovellukselle määriteltyjen toiminta-vaatimusten toimivuus sekä yrittää löytää sovelluksesta mahdolliset virheet ennen kuin tuote luovutetaan asiakkaalle. Hyvin suoritettu testaus tekee sovelluksesta laadukkaamman. Sovellusta tulisikin testata sekä kehittämisen aikana että sen loppuvaiheessa./10/

Sovelluksen testaus tapahtui pääosin toteutuksen yhteydessä samalla, kun jokin uusi toiminto otettiin käyttöön. Testausta suoritettiin esimerkiksi syöttämällä tekstikenttään odotettujen arvojen sijaan odottamattomia arvoja, kuten erikoismerkkejä, tai jättämällä tekstikenttiä tyhjäksi.

Sovelluskehityksen loppuvaiheessa testattiin vielä vaatimusmäärittelyssä luetellut toiminnot, joista suurin osa toimi odotetulla tavalla, mutta muutama toimintavirhekin löytyi.

Yksi virheistä oli Internet Explorer-selaimessa tapahtuva epämääräinen sivun zoomaus. Oletuksena zoomaus on 125 %, mutta sivua käyttäessä zoomaus vaihteli 100 % ja 150 % välillä riippuen näkyvissä olleesta sivusta. Myöhemmin huomasimme kuitenkin sivun zoomauksen johtuvan Internet Explorer-selaimesta, sillä samaa tapahtui myös muilla Internet-sivustoilla.

Toinen löydetty virhe oli päivämääräkenttien taustaväriin hidas päivittyminen päivitettäessä yhtä aikaa monta kenttää. Tämä johtui siitä, että värien päivittämiseen käytettiin Callback-toimintoa, joka haki palvelimelta vertailtavan kentän arvo jokaiselle syötetylle arvolle erikseen. Tämä saatiin korjattua tallentamalla vertailuarvot välimuistiin taulukon luontivaiheessa, jolloin vertailuun ei enää tarvittu yhteydenottoa palvelimelle.

Kolmas virhe löytyi tilausten hakutoiminnosta. Hakutoiminto antoi virheilmoituksen, jos hakukentän hakuehdossa oli heittomerkki, kuten esimerkiksi McDonald's. Virhe johtui siitä, että palvelimella hakusana erotetaan muusta koodista asettamalla

heittomerkki hakusanan molemmille puolille jolloin ohjelma luuli, että hakusana päättyi heittomerkkiin.

Tietojen viemistä Scala-järjestelmään ei pystytty vielä testaamaan, sillä asiakkaan testiympäristö ei ollut raportin kirjoitushetkellä valmis. Sovelluksen luomasta XML-tiedostosta voitiin kuitenkin nähdä, että luodun XML-tiedoston rakenne oli oikeanlainen, ja sisälsi tarvittut arvot.

6.1 Testausympäristö

Sovellusta kehitettiin ja testattiin Windows 8-ympäristössä. Kehittämisen aikana käytettiin uusimpia Google Chrome-, Internet Explorer- ja Mozilla Firefox-se-laimia sovelluksen testaamiseen.

7 YHTEENVETO

Työn lopputuloksena saatiin toimiva sovellus, joka tekee kaikki vaatimusmäärittelyssä vaaditut asiat sekä joitain asioita paremminkin kuin mitä oli vaadittu.

Raportin kirjoitusvaiheessa sovellusta ei ole vielä otettu käyttöön asiakkaan käyttöympäristössä. Sovellus tullaan ottamaan käyttöön asiakkaan käyttöympäristössä, kunhan vaaditut palvelinpuolen asiat saadaan kuntoon.

Sovellusta voitaisiin kehittää tietoturvallisemmaksi vaatimalla käyttäjää kirjautumaan sisälle ennen kuin hän pääsisi käsiksi tietokannasta haettuihin tietoihin.

Opinnäytetyö suoritettiin nopealla tahdilla, sillä aikaa oli vähän. Opinnäytetyön aihe saatiin maaliskuun alussa, mutta itse toteuttaminen päästiin aloittamaan vasta maaliskuun lopussa, joten työ piti suorittaa loppuun reilussa kuukaudessa.

Opinnäytetyö tuotti minulle paljon haasteita, sillä minulla ei ennestään ollut kokemusta ASP.NET MVC-arkkitehtuurityylistä. Myös DevExpress-työkalu aiheutti omia haasteita, sillä vaikka työkalulla sai luotua hienoja taulukoita helposti, oli sen muokkaaminen halutunlaiseksi vaikeaa.

Mielestäni työ oli erittäin mielenkiintoinen ja sopivan haastava ottaen huomioon käytettävissä olleen ajan. Projektin aikana opin paljon asioita ja kehityin ohjelmoijana.

LÄHDELUETTELO

- /1/ eCraft. Viitattu 24.3.2014.4 <http://www.ecraft.fi>
- /2/ Getting started with Visual Studio. Viitattu 29.3.2014. <http://msdn.microsoft.com/en-us/library/ms165079.aspx>
- /3/ ASP.NET MVC Overview. Viitattu 29.3.2014. <http://www.asp.net/mvc/tutorials/older-versions/overview/asp-net-mvc-overview>
- /4/ w3schools, ASP.NET Razor. Viitattu 29.3.2014. http://www.w3schools.com/aspnet/razor_syntax.asp
- /5/ Mozilla, About JavaScript. Viitattu 30.3.2014. https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript
- /6/ w3schools, Ajax introduction. Viitattu 30.3.2014. http://www.w3schools.com/ajax/ajax_intro.asp
- /7/ Tutorialspoint, SQL overview. Viitattu 2.4.2014. <http://www.tutorialspoint.com/sql/sql-overview.htm>
- /8/ Oracle, What is XML? Viitattu 2.4.2014. http://docs.oracle.com/cd/E13222_01/wls/docs92/xml/intro.html#wp187575
- /9/ Asp, mvc releases Viitattu 6.4.2014 <http://www.asp.net/mvc/overview/releases>
- /10/ Codeproject, What is software testing? Viitattu 28.4.2014 <http://www.codeproject.com/Tips/351122/What-is-software-testing-What-are-the-different-ty>