VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Tuomas Lammi

# TILAUSTEN STATUS WEB-SOVELLUS

Tekniikka
2015

VAASAN AMMATTIKORKEAKOULU
Tietotekniikka

# TIIVISTELMÄ

| | |
|---|---|
| Tekijä | Tuomas Lammi |
| Opinnäytetyön nimi | Tilausten status web-sovellus |
| Vuosi | 2015 |
| Kieli | suomi |
| Sivumäärä | 35 |
| Ohjaaja | Ghodrat Moghadampour |

Tämän opinnäytetyön tarkoitus oli kehittää web-sovellus, jolla käyttäjä pystyy hakemaan erilaisia tilaustietoja asiakkaan tietokannasta. Sovellus kehitettiin työn toimeksiantaja eCraft oy:n yritysasiakkaalle. Sovellus tulee käyttöön yrityksen tilauksista vastaavalle työntekijälle.

Sovellus toteutettiin Microsoftin Visual Studio-sovelluskehitysalustalla. Ohjelmointitekniikkana käytettiin ASP.net MVC 5-sovelluskehitystä. Isona osana ohjelmaa oli DevExpress-kirjasto, joka on maksullinen ohjelmakirjasto. DevExpressin pystyy lisäämään Visual Studio-ympäristöön, ja sen avulla voi luoda tyylikkäitä ja toiminnallisia ominaisuuksia.

Opinnäytetyön lopputuloksena saatiin toimiva web-sovellus, josta löytyy kaikki asiakkaan vaatimat ominaisuudet. Vaaditut ominaisuudet olivat tilaustietojen haku asiakkaan tietokannasta tilausnumeron, tilaustyypin, päivämäärän, statuksen, asiakasnumeron ja varastokoodin perusteella. Myöhässä olevat tilaukset tuli näyttää taulukossa punaisella värillä. Sovellus menee asiakkaalle käyttöön, kun se saadaan asennettua asiakkaan palvelinympäristöön. Sovellukseen pohja sallii sen, että siihen pystyy helposti lisäämään uusia ominaisuuksia asiakkaan niin halutessa.

Avainsanat ASP.NET, MVC, Visual Studio, DevExpress

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Information technology

# ABSTRACT

| | |
|---|---|
| Author | Tuomas Lammi |
| Title | Web Application for Reading Order Status |
| Year | 2015 |
| Language | Finnish |
| Pages | 35 |
| Name of Supervisor | Ghodrat Moghadampour |

The aim of this thesis was to create a web application that can get different order datas from the customer's database. The application was developed for the principal's, Ecraft's, customer. The application will be used by the customer's employee who is responsible for handling orders.

The application was implemented using Microsoft's Visual Studio integrated development environment. The programming technique used for this application was ASP.Net MVC 5-framework. A big part of the program was DevExpress-library, which is a paid software library. It can be added to Visual Studio to create stylish and working entityes.

As a result of the thesis a working application was developed, which has all the features the customer wanted to have. The demanded features were a search by order number, order type, date, status, customer number and stock code. Orders that are late should be presented in red colour in the application. The application will be put into use after the server side is set for the customer. New features can easily be added if the customer wants to do so.

Keywords ASP.NET, MVC, Visual Studio, DevExpress

# SISÄLLYS

TIIVISTELMÄ
ABSTRACT
1 JOHDANTO ..... 6
1.1 Työn tarkoitus ..... 6
1.2 Ecraft Oy Ab ..... 6
2 SOVELLUKSESSA KÄYTETYT TEKNIIKAT ..... 7
2.1 Microsoft Visual studio ..... 7
2.2 Asp.net ..... 8
2.3 Asp.net MVC ..... 9
2.4 Razor ..... 10
2.5 SQL ..... 11
2.6 JavaScript ..... 11
2.6.1 Jquery ..... 13
2.7 AJAX ..... 14
3 SOVELLUKSEN MÄÄRITTELY ..... 16
3.1 Lähtökohdat ..... 16
3.2 Sovelluksen määrittelyt ..... 16
3.2.1 Vaatimusmäärittely ..... 17
3.2.2 Käyttötapauskaavio ..... 18
3.2.3 Sekvenssikaavio ..... 19
3.2.4 Luokkakaavio ..... 20
4 SOVELLUKSEN TOTEUTUS ..... 22
4.1 Sovelluksen toteutukseen käytetyt tekniikat ..... 22
4.2 Sovelluksen ulkoasun luonti ..... 22
4.3 Tilaustietojen hakeminen tietokannasta ..... 23
4.4 Tilaustietojen suodattaminen hakuparametrien perusteella ..... 26
4.5 Sarakkeen värin vaihto tilauspäivämäärän mukaan ..... 30
5 TESTAUS ..... 33
6 YHTEENVETO ..... 34
LÄHTEET ..... 35

## KUVIO- JA TAULUKKOLUETTELO

**Kuvio 1.** Asp.net Web Pages esimerkki s. 9
**Kuvio 2.** MVC-arkkitehtuurin toimintaperiaate s. 9
**Kuvio 3.** Yksinkertainen esimerkki Tuote-mallista s. 10
**Kuvio 4.** Yksinkertainen esimerkki jäsentäjästä s. 10
**Kuvio 5.** Yksinkertainen esimerkki näkymästä s. 10
**Kuvio 6.** Esimerkki Razor-syntaksista s. 11
**Kuvio 7.** Yksinkertainen Javascript-esimerkki s. 12
**Kuvio 8.** Yksinkertainen Javascript-esimerkki s. 12
**Kuvio 9.** Yksinkertainen jQuery-esimerkki s. 13
**Kuvio 10.** Jquery selaimessa s. 13
**Kuvio 11.** Ajax-esimerkki s. 14
**Kuvio 12.** Ajax-esimerkki selaimessa s. 15
**Kuvio 13.** Avoimien tilauksien luonnos s. 17
**Kuvio 14.** Käyttötapauskaavio s. 18
**Kuvio 15.** Sovelluksen avaamissekvenssidiagrammi s. 19
**Kuvio 16.** Sovelluksen hakutoiminnon sekvenssidiagrammi s. 19
**Kuvio 17.** Tilausten selaamisen sekvenssidiagrammi s. 20
**Kuvio 18.** Sovelluksen luokkakaavio s. 21
**Kuvio 19.** Esimerkki bootstrap gridistä s. 23
**Kuvio 20.** SQL-näkymä s. 24
**Kuvio 21.** Tietokannan entiteetti s. 24
**Kuvio 22.** OrdersGridViewPartial() s. 25
**Kuvio 23.** Jäsentäjän palauttama näkymä s. 26
**Kuvio 24.** Search ja getSearchParameters-metodit s. 27
**Kuvio 25.** GridViewPartialSearch-metodi s. 28
**Kuvio 26.** SearchModel-luokka s. 28
**Kuvio 27.** DataProvider-luokka s. 29
**Kuvio 28.** Palautettava näkymä s. 30
**Kuvio 29.** setting.HtmlDataCellPrepared-asetukset s. 31
**Kuvio 30.** Värin vaihtuminen päivämäärän mukaan s. 32

**Taulukko 1.** Vaatimusmäärittely s. 18

# 1 JOHDANTO

Nykyaikana sovelluksia halutaan siirtää yhä enemmän web-pohjaisiksi. Web-sovellukset mahdollistavat sujuvamman ohjelmien käytön selaimen kautta ja eri laitteilla. Hyvin suunnitellut web-sovellukset toimivat sekä tietokoneen että puhelimen näytöllä sujuvasti.

## 1.1 Työn tarkoitus

Tämä opinnäytetyö on tehty Vaasan ammattikorkeakoulun tietotekniikan ohjelmistotekniikan suuntautumisen lopputyönä. Työn toimeksiantajana toimi Ecraft Oy, joka on erikoistunut toiminnanohjaussovelluksiin.

Opinnäytetyö tehtiin asiakastyönä Ecraftin yritysasiakkaalle, ja työn lähtökohtana oli toteuttaa web-sovellus asp.net MVC-tekniikalla. Sovelluksella käyttäjä pystyy sujuvasti hakemaan erilaisia tilaustietoja asiakkaan tilaustietokannasta. Opinnäytetyön käytännön osuus toteutettiin eCraftin toimipisteellä Vaasassa ja kirjallinen osuus tehtiin omalla ajalla.

## 1.2 Ecraft Oy Ab

Ecraft Oy Ab on vuonna 1999 perustettu ohjelmistoalan asiantuntijayritys, joka tekee yrityksille toiminnanohjausjärjestelmiä. Ecraft on ”kokonaisvaltainen ratkaisutoimittaja” ja se tarjoaa asiakkailleen ERP-, CRM-, Business Intelligence-, sovelluskehitys-, Smart Apps- sekä infrapalveluita. Esimerkiksi sovelluskehityspalvelu sisältää pilvipalveluita, tukipalveluita, konsultointia sekä mobiili- ja käytettävyyspalveluita. Järjestelmät luodaan helpottamaan asiakasyrityksen liiketoimintaa, ja yrityksen missiona onkin tehdä IT-järjestelmistä käytettäviä ja innostavia. Ecraft luo myös räätälöityjä ratkaisuja mille tahansa perusjärjestelmälle silloin kun perusjärjestelmä ei tuo asiakkaan toivomia toiminnallisuuksia. /1/

Ecraftin toimipisteet sijaitsevat Vaasassa ja Espoossa Suomessa sekä Tukholmassa, Malmössa ja Linköpingissä Ruotsissa, ja se työllistää yli 100 asiantuntijaa. Tunnettuja asiakkaita ovat mm. Kiinteistömaailma, Ifolor ja Apetit-pakaste. /1/

# 2 SOVELLUKSESSA KÄYTETYT TEKNIIKAT

Tässä luvussa esitellään opinnäytetyössä käytetyt tekniikat, sekä kerrotaan niiden käyttötarkoituksista.

## 2.1 Microsoft Visual studio

Visual studio on Microsoftin kehittämä ohjelmointityökalu, jolla pystytään kehittämään ohjelmistoja useille eri alustoille, kuten

- Windows-kauppa
- Windows-puhelin
- Windows-työpöytä
- Web-ASP.net
- Web-palvelut.

Ohjelmien kehityksessä voidaan käyttää mm. seuraavia ohjelmointikieliä:

- Visual Basic
- Visual C#
- Visual C++
- Visual F#
- JavaScript. /2/

Visual studio hyödyntää .NET sovelluskehystä. .NET on ohjelmistokehys, joka sisältää tuen erilaisille alustoille, kuten windows, web tai muihin sovelluksiin. Se sisältää esimerkiksi automaattisen muistinhallinnan, mikä helpottaa ohjelmoijan työtaakkaa paljon. Ilman muistinhallintaa ohjelmoijan pitäisi aina muistia varatessa myös miettiä, milloin vapauttaa varattu muisti. Tämän ominaisuuden ansiosta .NE-Tissä ei voi tulla niin sanottuja ’memory leakkejä’. .NET mahdollistaa myös sen, että koodin jakaminen kehyksen ympärillä on täysin mahdollista. Kehys sisältää myös common language runtimen (CLR), mikä hallinnoi ohjelmien ajamisen kaikilla .NET ohjelmointikielillä. CLR pitää myös huolen muistinhallinnasta, poikkeustiloista ja sovelluksen debuggauksesta./14/

## 2.2 Asp.net

Asp.net on web-kehitysmalli, joka sisältää kaikki tarvittavat palvelut yritystason sovelluksen tekemiseen. Asp.netin idea on minimioida koodin kirjoittaminen, ja käyttää hyväksi ympäristön tarjoamia työkaluja. Asp.net on osa .NET-ohjelmisto-kehystä. Ohjelmistot koodataan usein joko Visual Basic tai c# ohjelmointikielillä.

Asp.net tarjoaa kolme ohjelmistokehystä web-sovellusten kehittämiseen. Asp.net Webforms-kehitys on hyvin samankaltaista kuin Windows Forms-kehitys. Tämän ansiosta kehittäjän on helppo siirtyä tekniikoiden välillä. Webforms mahdollistaa komponenttien käyttämisen drag and drop-metodilla, ja sen ansiosta kehittäminen on hyvin nopeaa eikä vaadi käyttäjältään suurta tietämystä javascriptistä tai HTML-syntaksista.

Asp.net MVC on kehittäjille, jotka haluavat jakaa sovelluksensa kolmeen eri osaan; Model, View ja Controller. Tämä helpottaa sovelluksen hallintaa, varsinkin jos kyseessä on suuri ja monimutkainen sovellus. MVC-ympäristössä ei ole yhtä paljon automatisoituja ominaisuuksia kuin Web Formsseissa, mutta se antaa enemmän hallintaa kehittäjälleen.

Asp.net Web Pages on kehittäjille, jotka haluavat luoda yksinkertaisia tai pieniä web-sovelluksia. Sen avulla luodaan yksinkertaisia HTML-sivuja, ja kehittäjä voi käyttää esimerkiksi PHP:tä back-end kielenä. Web Pagesiä **(Kuvio 1.)** pidetään yksinkertaisempama vaihtoehtona Web Formsseihin. /15/

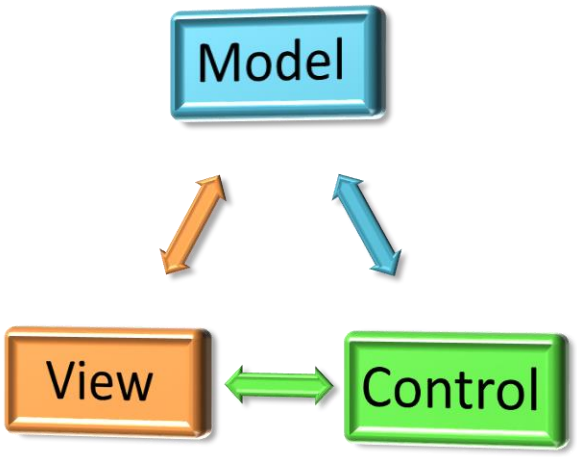
```
<!DOCTYPE html>

<html>

<body>

	<h1>Hello Web Pages</h1>


	// @ komennolla ajetaan c# koodia, ja haetaan päivämäärä + näytetään
	se käyttäjälle

	<p>The time is @DateTime.Now</p>

</body>
```

```
</html>
```

**Kuvio 1.** Asp.net Web Pages esimerkki

### 2.3 Asp.net MVC

MVC on arkkitehtuurityyli, joka jakaa sovelluksen kolmeen eri osaan: Model (malli), View (näkymä) ja Controller (jäsentäjä) (**Kuvio 2.)**.




**Kuvio 2.** MVC-arkkitehtuurin toimintaperiaate. /3/

Model (malli) hoitaa sovelluksen datan hallinnan. Model-luokalla voidaan esimerkiksi hakea tietokannan taulusta dataa, muokata sitä ja lähettää tieto takaisin tietokantaan. (**Kuvio 3.**) /3/

```
public class Tuote
{
       public string Nimi { get; set; }
        public decimal Hinta { get; set; }
        public int KplMaara { get; set; }
}
```

**Kuvio 3.** Yksinkertainen esimerkki Tuote-mallista

Controller (jäsentäjä) hoitaa sovelluksen navigoinnin, kommunikoi mallin kanssa ja ohjaa sovelluksen tarvittuun näkymään **(Kuvio 4.)**. /3/

```
public class HomeController : Controller
{
       // GET: Home
       public ActionResult Index()
       {
              Tuote tuote = new Tuote
             {
```

```
                Nimi = "Tietokone",
                Hinta = 1850,
                KplMaara = 2
            };

            return View(tuote);
        }
}
```

**Kuvio 4.** Yksinkertainen esimerkki jäsentäjästä

View (näkymä) hoitaa sovelluksen ulkoasun **(Kuvio 5.)**. Näkymä luodaan yleensä mallin mukaisesti. /3/

```
@model oppari_esimerkit.Models.tuote

@{

    ViewBag.Title = "Index"

}

<h2>Index</h2>

<p>Tuote:</p>

<p>@Model.Nimi</p>

<p>@Model.Nimi</p>

<p>@Model.Nimi</p>
```

**Kuvio 5.** Yksinkertainen esimerkki näkymästä

## 2.4 Razor

Razor on merkintäsyntaksi, jolla pystytään luomaan palvelinpuolen koodia web-sivuille HTML-koodin sekaan. Se ei ole oma ohjelmointikieli, vaan palvelinpuolen merkintäsyntaksi. Razor tukee Visual Basic- ja C#-ohjelmointikieliä.

Razorin avulla pystytään luomaan dynaamisia internetsivuja. Koska se käännetään palvelinpuolella, sillä pystytään tekemään monimutkaisia asioita, kuten hakea dataa tietokannasta. Razor perustuu ASP.NETiin, ja se on suunniteltu internetsivujen luontiin. /5/

Yleensä ohjelmissa käytetään Razoria ohjelman mallin datan esittämiseen. (**Kuvio 6.**)

```
@model oppari_esimerkit.Models.Tuote

<p>@Model.Nimi</p>
<p>@Model.Hinta</p>
<p>@Model.KplMaara</p>
```

**Kuvio 6.** Esimerkki Razor syntaksista

## 2.5 SQL

SQL (Structured Query Language) on ohjelmointikieli tietokantojen manipulointiin. SQLin avulla pystytään mm. luomaan, muokkaamaan, lisäämään tai poistamaan tietokantoja. Esimerkkejä ohjelmista, jotka käyttävät hyödyksi SQL-kieltä:

- SQL Server
- MySQL
- MS Access. /6/

## 2.6 JavaScript

Javascript on ohjelmointikieli, jota käytetään yleensä web-sivujen kehityksessä. Se ei kuitenkaan ole sidottu web-sivuihin, vaan sitä käytetään esimerkiksi pelien kehittämiseen. Javascript pyörii sivun käyttäjän omalla koneella, mikä tekee siitä kevyen ja helpon käyttää. Ainoa vaatimus Javascriptin käyttöön on, että selain tukee sitä. Javascript kuitenkin toimii kaikilla yleisimmillä internetselaimilla, kuten Firefox, Chrome, Opera ja Internet Explorer. /12/

Javascript on syntaksiltaan hyvin samanlainen kuin useat suositut ohjelmointikielet kuten Java, c# ja c++. Myös monet ominaisuudet, kuten taulukot, toimivatkin hyvin samalla tavalla. Syntaksi on tehty samanlaiseksi sen takia, että kehittäjät, jotka tulevat eri ohjelmointitaustalta, pystyvät oppimaan javascriptiä helpommin. Javascriptin kehittäminen omatoimisesti ei vaadi kehittäjältä kuin valitsemansa tekstieditorin. /7/

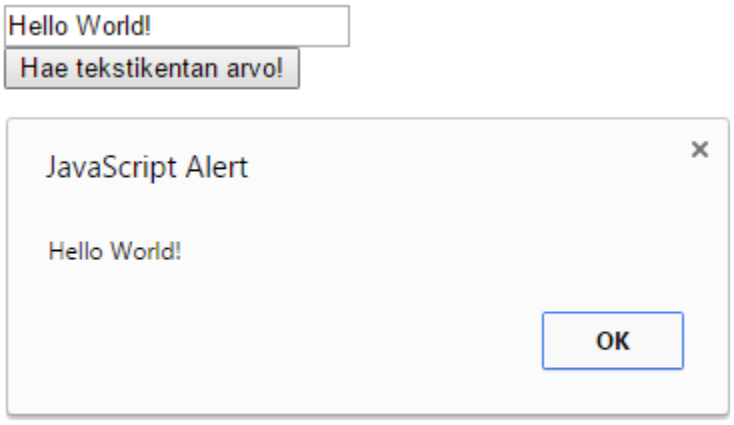
Javascriptiin on kehitetty useita eri ohjelmointikirjastoja. Kirjastojen tarkoitus on helpottaa monimutkaisten asioiden, kuten eri selainten yhteistoimivuuden ristirii-

toja. Useimmat kirjastot myös tarjoavat helpotettuja keinoja html-dokumentin manipulointiin, animaatioiden tekemiseen ja AJAX-pyyntöjen lähettämiseen. Suosituimpia kirjastoja ovat esimerkiksi jQuery, angularJS ja reactJS. /13/

Javascriptillä haetaan usein tietoa esimerkiksi HTML-dokumentin osasta. HTML-tiedostoon voidaan esimerkiksi laittaa tekstikenttä ja nappi, ja nappia painamalla haetaan tekstikentän arvo ja esitetään se käyttäjälle. (**Kuviot 7. ja 8.)**

```
<html>
<body>
    <!-- Luodaan tekstikenttä ja button-->
    <input type="text" id="id" />
    <br>
    <button type"button" onclick="haeArvo()">Hae
    tekstikentan arvo!</button>
    <script type="text/javascript">
       // Metodi, millä haetaan tieto tekstikentästä
    function haeArvo()
    {
       // Tallennetaan tekstikäntän data 'arvo' muuttujalle
        var arvo = document.getElementById("id").value;
        // Näytetään arvo muuttuja käytäjälle
        alert(arvo);
    }
    </script>
</body>
</html>
```

**Kuvio 7.** Yksinkertainen Javascript-esimerkki



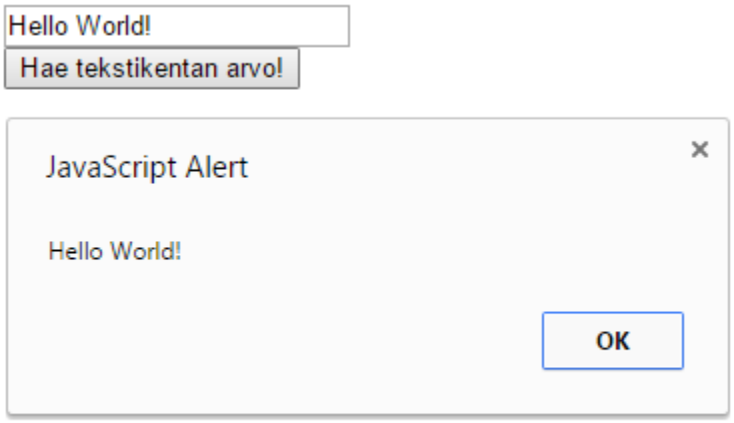

**Kuvio 8.** Yksinkertainen Javascript-esimerkki

### 2.6.1 Jquery

Jquery on kevyt ja nopea javascript-kirjasto, joka on suunniteltu helpottamaan tiettyjä javascriptin ominaisuuksia. Jquery yksinkertaistaa HTML-dokumentin läpikäyntiä, tapahtumien käsittelyä, animointia ja AJAX-pyyntöjen käsittelyä. Jquery toimii javascriptin tavoin käyttäjän omalla tietokoneella, eikä sen käyttöönotto kehittäjälle vaadi muuta kuin kirjaston liittämistä HTML-tiedostoon. Jqueryn motto on saada enemmän aikaan pienemmällä määrällä koodia. Jqueryllä voidaan esimerkiksi helposti kirjoittaa HTML-dokumenttiin kehittäjän haluama teksti. (**Kuviot 9. ja 10.)** /8/

```
<html>
<body>
    <!-- Luodaan tekstikenttä ja nappula  -->
    <input type"text" id="id" />
    <br>
    <button id="btn">Submit</button>

    <script type="text/javascript">

        // Odotetaan, että HTML-dokumentti on latautunut täysin
        $(document).ready(function(){

          // Haetaan tekstinkentän arvo, jonka id on "id"
          var value = $("#id").val();

          // Metodi, jota kutsutaan kun käyttäjä klikkaa painiketta,
          jonka id = "btn"
          $("#btn").click(function()
          {
              // Näytetään tekstinkentän arvo käyttäjälle
              alert(value);
          })

        });

    </script>
</body>
</html>
```

**Kuvio 9.** Yksinkertainen jQuery-esimerkki




**Kuvio 10.** Jquery selaimessa

## 2.7 AJAX

AJAX tarkoittaa Asynchronous JavaScript and XML. Se on tekniikka, jonka avulla luodaan nopeampia ja käyttäjäystävällisempiä internetsivuja. AJAXin käyttötarkoitus on saada intenetsivut latautumaan ilman, että koko sivusto latautuu uudelleen. Kun käyttäjä painaa esimerkiksi kuvitteellisella sivulla olevaa ”Button” nappia, niin koko sivun ei tarvitse latautua uudelleen. Javascript lähettää koodissa määriteltyyn kohteeseen datan, ja senhetkisen kohteen data päivittyy ilman, että koko sivusto latautuisi uudelleen. /11/ **(Kuviot 11. ja 12.)**

```
<head>
// Tarkistetaan, että HTML-dokumentti on täysin latautunut
$(document).ready(function(){
       // Seurataan 'button' napin painamista
        $("button").click(function(){
       // Kirjoitetaan 'div_to_target' diviin 'Hello World'
         $("#div_to_target").append("Hello World ");
    });
});
</script>
</head>
```

```
<body>

<!-- Jquery kirjoittaa tämän alapuolelle tekstin-->

<div id="div_to_target"><h1>Ajax esimerkki</h1></div>

<!-- Nappi -->

<button>Button</button>

</body>
```

**Kuvio 11.** Ajax-esimerkki


**Kuvio 12.** Ajax-esimerkki selaimessa

# 3 SOVELLUKSEN MÄÄRITTELY

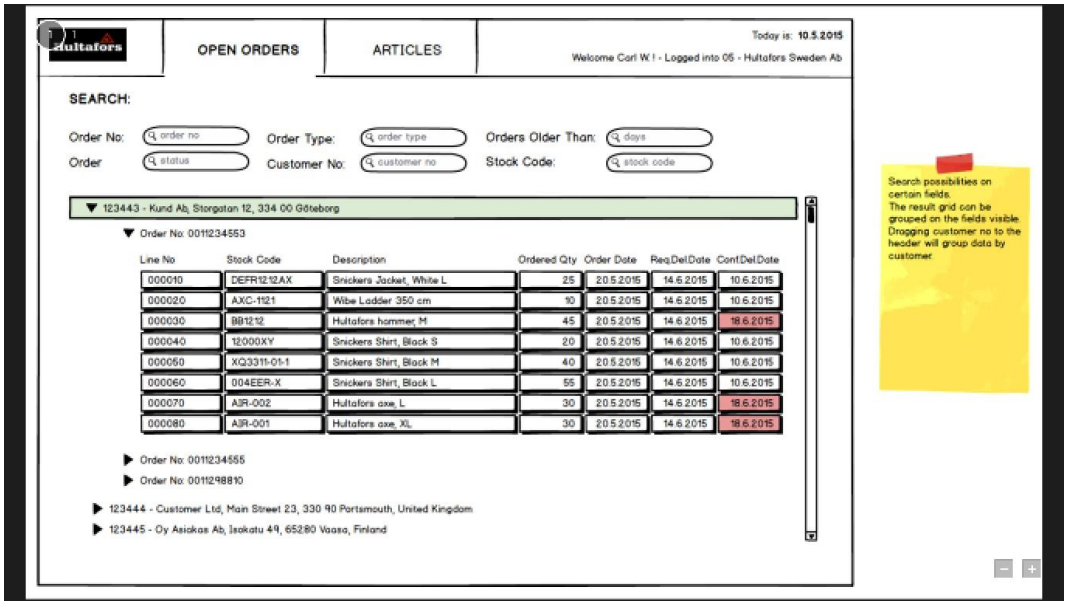
Tässä luvussa esitellään sovelluksen määrittelyjä. Sovellus koostuu internet-sivusta, jonka avulla voidaan hakea tietoa tilauksista asiakkaan tietokannasta. Sovellus esittää tilaukset DevExpress-työkalun luomalla taulukolla. Käyttäjä voi myös hakea tietoa lukuisista eri tekstikentistä, ja pystyy suodattamaan hakutulokset, esimerkiksi haluamaansa järjestykseen.

## 3.1 Lähtökohdat

Lähtökohta on luoda selainpohjainen sovellus, jolla pystyy hakemaan tietoa tilauksista yksinkertaisessa ympäristössä. Tieto pitää pystyä hakemaan, esimerkiksi tilauspäivämäärän mukaan, tilauksen statuksen mukaan ja monella muulla hakuparametrilla.

## 3.2 Sovelluksen määrittelyt

Sovelluksen määrittely on tehty asiakkaan toiveiden mukaan. Toiveiden perusteella on luotu luonnos, joka kuvaa sovellusta. Sovellus on HTML-sivu, joka sisältää taulukon ja tekstikenttiä. Tekstikentät ovat Order Number, Order Type, Orders Older than, Order (Order status), Customer number ja Stock Code. Tektikenttien avulla tehdään erilaisia hakuja sovelluksen tietokantaan. Taulukko sisältää sarakkeet datoille, ja se ryhmittää ne asiakasnumeron perusteella. Klikkaamalla asiakasnumeroa taulukkoon, avautuu kaikki sen asiakkaan tekemät tilaukset. Taulukko näyttää tarkat tilaustiedot, ja kertoo käyttäjälle onko se myöhässä. Tämä näkyy taulukossa siten, että konfirmoitu tilauspäivämäärä on maalattu punaiseksi, jos se on myähemmin kuin vaadittu tilauspäivämäärä. Taulukko päivittyy AJAX-tekniikalla, eli sivu ei lataudu uudelleen hakujen välissä. Taulukon sarakkeita pystyy myös järjestämään klikkaamalla sarakkeen nimeä. (**Kuvio 13.)**




**Kuvio 13.** Avoimien tilauksien luonnos

### 3.2.1 Vaatimusmäärittely

Vaatimusmäärittelyssä esitetään mitä asiakas sovellukselta odottaa. Vaaditut ominaisuudet on kerätty taulukkoon, josta pystyy lukemaan mitä sovelluksen tulisi sisältää. (**Taulukko 1.**)

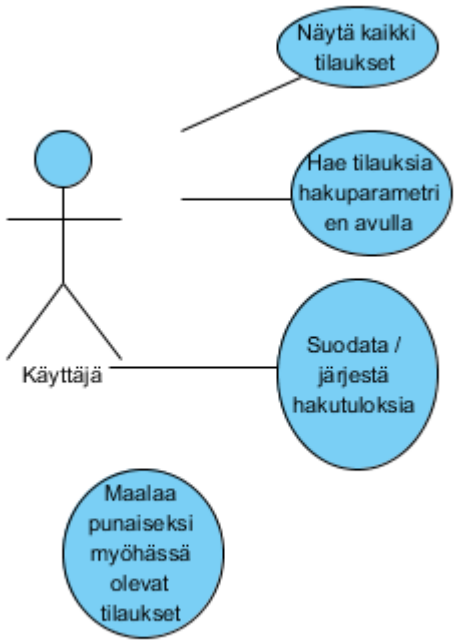
**Taulukko 1.** Vaatimusmäärittely

| **Vaaditut ominaisuudet** | **Tärkeys** |
|---|---|
| • Näkymä kaikista tilauksista. Tilauksia pitää pystyä järjestämään käyttäjän haluamalla tavalla. | **1** |
| • Näkymä kaikista backorderstilauksista, eli tilauksen status 4. | **1** |
| • Näkymä kaikista tilauksista, joiden tyyppi on 0 | **1** |
| • Tilaukset, joiden status on keskeytetty, eli 9 | **1** |
| • Tilaukset, joiden status on estetty, eli 6 | **1** |

| | |
|---|---|
| • Tilaukset pystyy suodattamaan päivämäärän mukaan. Esimerkiksi tilaukset vanhempia kuin 10, 20 ja 30 päivää.<br>• Näkymä näyttää, jos confirmed date-ja required date-sarakkeissa on eroa. | **1**<br><br>**1** |
| **1 = Must have, tulee olla** | |

### 3.2.2 Käyttötapauskaavio

Käyttötapauskaavion avulla esitetään sovelluksen eri toimintoja. Sovelluksella on yksi Windows-autentikoinnilla vahvistettu käyttäjä. Käyttäjä pystyy hakemaan ostotilauksia tilausnumeron, tilaustyypin, tilauspäivämäärän, tilauksen statuksen, asiakasnumeron ja varastonumeron perusteella. (**Kuvio 14.**)

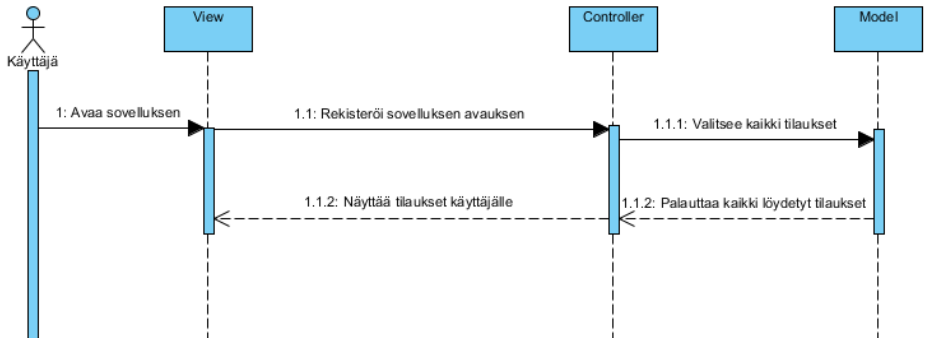



**Kuvio 14.** Käyttötapauskaavio

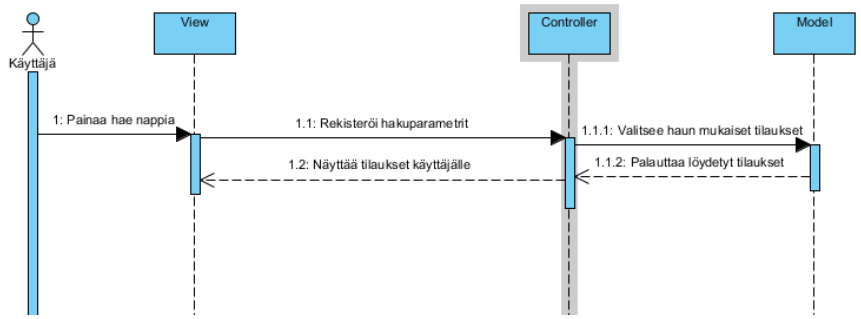
### 3.2.3 Sekvenssikaavio

Sekvenssikaavion avulla havainnollistetaan sovelluksen toimintaa. Se kuvaa sovelluksen eri osioiden yhteistyötä keskenään.

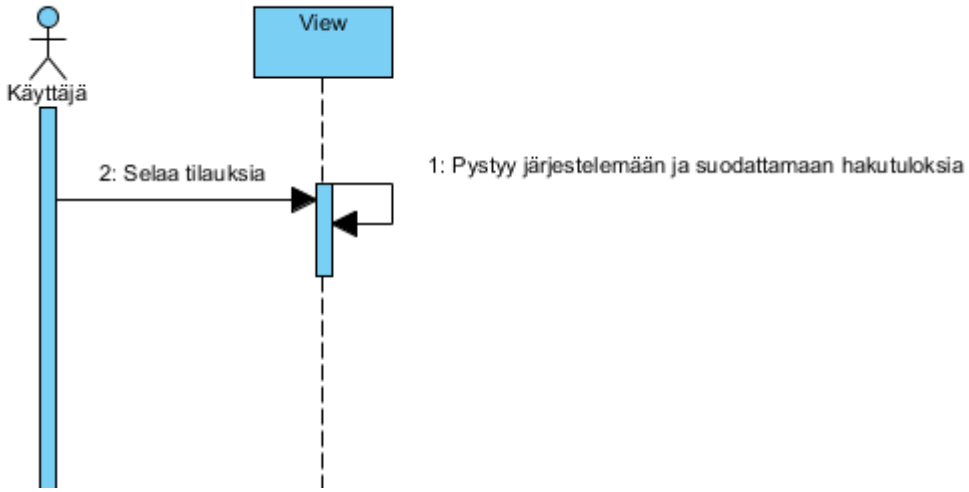
Sovelluksen avaamistapahtumasta, hakutapahtumasta ja selaamistapahtumasta on tehty sekvenssikaavio **(Kuviot 15., 16. ja 17.)**. Tapahtumat ovat hyvin samanlaisia. Kun käyttäjä avaa sovelluksen tietokannasta, haetaan kaikki sillä hetkellä olevat tilaukset ja näytetään ne käyttäjälle. Käyttäjän painaessa hae-nappia, sovellus hakee käyttäjän syöttämillä hakuparametreilla olevat tilaukset tietokannasta ja näyttää ne käyttäjälle. Käyttäjä pystyy halutessaan järjestämään ja suodattamaan hakutuloksiansa haun jälkeen




**Kuvio 15.** Sovelluksen avaamisen sekvenssidiagrammi



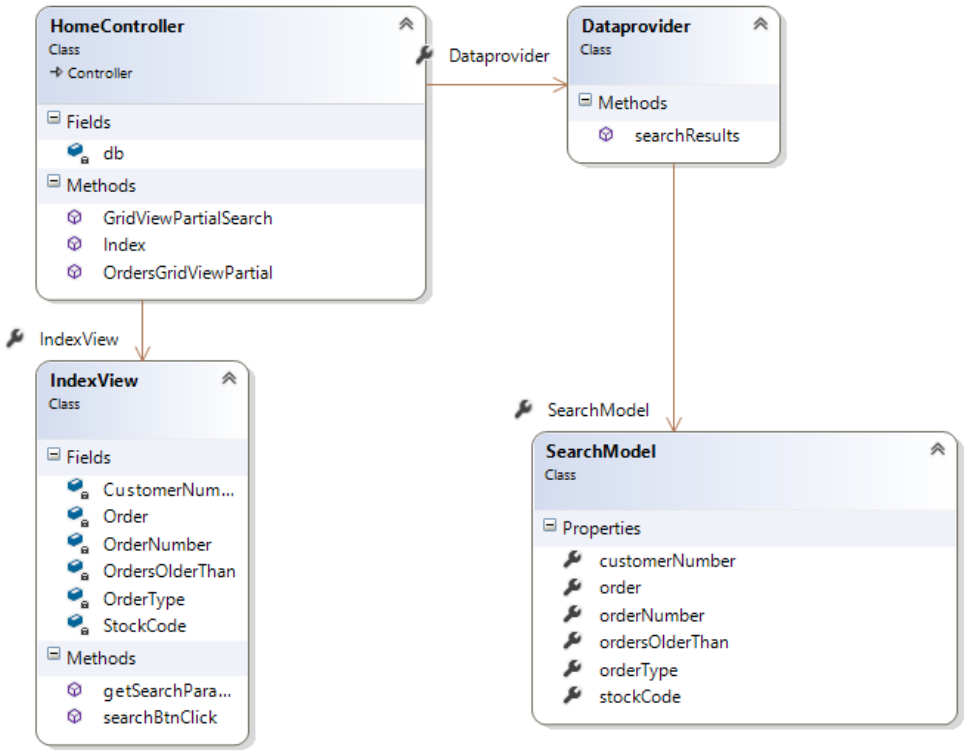

**Kuvio 16.** Sovelluksen hakutoiminnon sekvenssidiagrammi




**Kuvio 17.** Tilausten selaamisen sekvenssidiagrammi

### 3.2.4 Luokkakaavio

Luokkakaaviolla näytetään sovelluksen sisältämät luokat sekä niiden väliset suhteet. Kaaviolla pyritään avaamaan sovelluksen rakennetta. **(Kuvio 18.)** Sovellus on jaettu MVC-tyylin mukaisesti kolmeen osioon. Controller-luokka käsittelee kaikki sovelluksen näkymässä tapahtuvat ominaisuudet. Controller-luokka saa datansa DataProvider-luokalta, joka käyttää searchModel-luokkaa avukseen haussa. IndexView lähettää datansa HomeController-luokalla pyynnöstä.




**Kuvio 18.** Sovelluksen luokkakaavio

# 4 SOVELLUKSEN TOTEUTUS

Tässä luvussa käydään läpi miten sovellus toteutettiin, mitä tekniikoita toteutukseen käytettiin ja minkälaisia vaihteita toteutukseen kuului.

## 4.1 Sovelluksen toteutukseen käytetyt tekniikat

Sovellus toteutettiin Microsoftin asp.net MVC 5-tekniikalla. Koska MVC-ohjelmointityyliin kuuluu jakaa sovelluksen eri osiot omiin ryhmiinsä, sovelluksen hallinta pysyy helppona. Jäsentäjä, näkymä ja mallit ovat omissa kansioissaan. OpenOrders-sivulla on oma Jäsentäjä-luokka, jonka avulla kutsutaan sopivia metodeita, Näkymille on oma kansionsa, jossa ovat sovelluksen HTML-tiedostot ja malleille on oma kansionsa, jossa ovat sovelluksen logiikan hoitavat mallit.

Iso osa sovelluksesta on tilaustietojen näyttäminen web-sivulla. Tilaustietojen näyttäminen toteutettiin DevExpress-työkalulla, joka on .net-rajapinnalle suunniteltu kirjasto. DevExpressin avulla tilaustiedot pystytään esittämään tyylikkäästi ja vaivattomasti.

Sovelluksen luomiseen käytettiin Microsoftin Visual Studio-työkalua. Visual Studio luo projektiin automaattisesti malliprojektin, mistä löytyy perusnavigaatioon vaaditut toiminnollisuudet valmiiksi. Visual Studio antaa myös mahdollisuuden lisätä projektiin käyttäjän todennusmahdollisuuden. Tässä työssä käytetään Windows-todennusjärjestelmää. Tämä mahdollistaa sen, että käyttäjän on pitänyt kirjautua sisään määritettyyn toimialueeseen käyttääkseen sovellusta.

## 4.2 Sovelluksen ulkoasun luonti

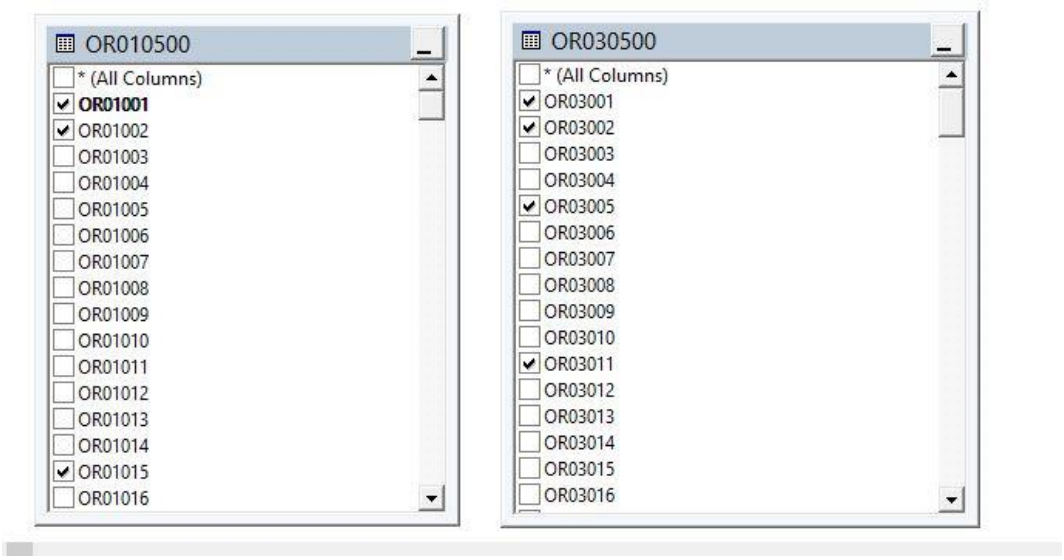
Sovelluksen ulkoasu on yleisesti sama kuin Visual Studion malli web-projekti. Se käyttää hyväkseen Twitterin luomaa bootstrap-tyylitiedostoa ja javascript-kirjastoa. Käyttämällä boostrapin luokkia hyväksi, oli yksinkertaista luoda sivulle käytännöllinen grid-ulkoasu. Grid-systeemi toimii niin, että jokaiselle riville mahtuu 12 kolumnia. Näin sivu on helppo jakaa sopiviin osiin. Sivun tekstikentät, dropbox ja button luotiin käyttämällä DevExpressin tarjoamia kirjastoja. Näin tekstikenttiin oli helppo lisätä tarvittavia parametreja. (**Kuvio 19.)**

```
<div class="row">
        <div class="col-md-4">
            @Html.DevExpress().TextBox(settings =>
            {
                settings.Name = "OrderNumberTxtBox";
                settings.Width = 150;
                settings.Properties.CaptionSettings.Position = devExpress
                 .Web.EditorCaptionPosition.Top;
                settings.Properties.Caption = "Order No";
                settings.ToolTip = "Enter the order number here.";
            }).GetHtml()
        </div>
        <div class="col-md-4">
            @Html.DevExpress().TextBox(settings =>
            {
                settings.Name = "OrderTypeTxtBox";
                settings.Width = 150;
                settings.Properties.CaptionSettings.Position =
                 DevExpress.Web.EditorCaptionPosition.Top;
                settings.Properties.Caption = "Order Type";
                settings.ToolTip = "Enter the order type here.";
            }).GetHtml()
        </div>
        <div class="col-md-4">
            @Html.DevExpress().ComboBox(settings =>
            {
                settings.Name = "OrdersOlderThanComboBox";
                settings.Width = 150;
                settings.Properties.CaptionSettings.Position =
                 DevEx    press.Web.EditorCaptionPosition.Top;
                settings.Properties.Caption = "Orders Older Than";
                settings.ToolTip =
                 "Current days - selected value from list,";
                settings.Properties.ValueType = typeof(string);
                settings.Properties.Items.Add("10");
                settings.Properties.Items.Add("20");
                settings.Properties.Items.Add("30");
                settings.Properties.Items.Add("50");
                settings.Properties.Items.Add("80");
                settings.Properties.Items.Add("120");
            }).GetHtml()
        </div>
</div>
```

**Kuvio 19.** Esimerkki bootstrap gridistä

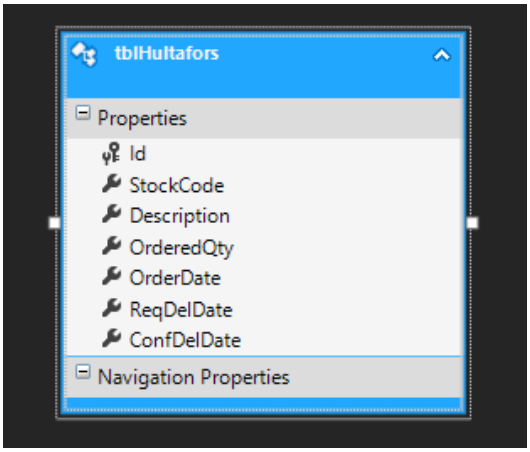
### 4.3 Tilaustietojen hakeminen tietokannasta

Sovellus käyttää paikallista SQL Server tietokantaa tietojen hakemiseen. Data jota haetaan, on asiakasyrityksen palvelimelta. Sovelluksessa ei kuitenkaan otettu suoraan yhteyttä asiakkaan tietokantapalvelimelle, vaan tarvittavat taulut kopioitiin paikalliselle koneelle. Tämä tehtiin siksi, että asiakkaan tietokanta oli todella suuri. Sen sijaan, että koko taulun tieto ladattaisiin sovellukseen, valittiin tietokannan tauluista tarvittavat sarakkeet. Näistä sarakkeista luotiin SQL-näkymä sisältäen vain ne tiedot, joita sovellus käyttää hauissaan. (**Kuvio 20.)**




**Kuvio 20.** SQL näkymä

Tietojen hakemisessa käytetään .NET Entity Framework-kirjastoa hyväkseen. Entity Framework on Microsoftin työkalu, joka poistaa tarpeen kirjoittaa koodia, jolla luodaan yhteys tietokantaan. Entity Framework luo automaattisesti kehittäjälle entiteetin eli näkymää kuvaavan mallin, jota käytetään tietojen hakemiseen tietokannasta. Visual Studio luo entiteetistä myös graafisen mallin. **(Kuvio 21.)**



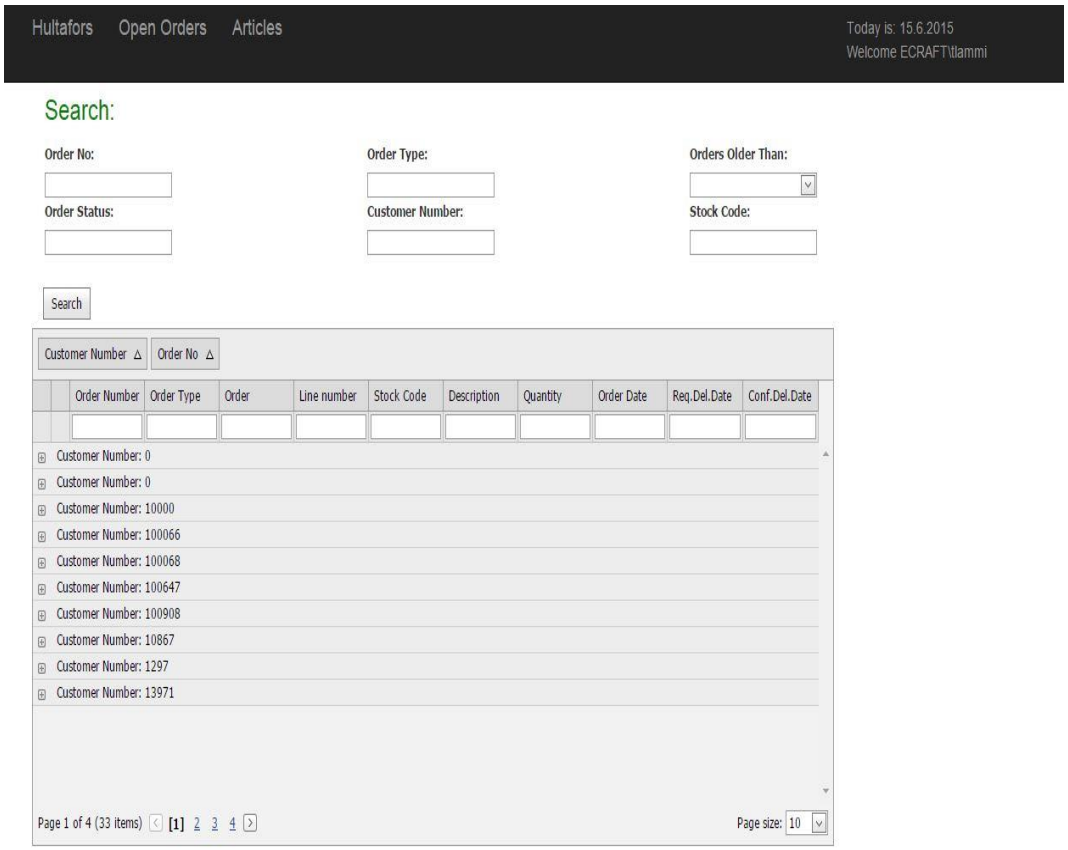

**Kuvio 21.** Tietokannan entiteetti

Tilaustietojen näyttämistä varten luotiin DevExpress grid, johon kiinnitettiin Entity Frameworkin luoma entiteetti. Sovelluksen latautuessa grid kutsuu home/OrdersGridViewPartial()-metodia. Tämä on määritelty OrdersGridViewPartial.cshtml tiedostossa, joka toimii gridin konfigurointitiedostona. Tämä metodi tarkistaa, onko selaimen istuntoon asetettu mitään hakuparametreja. Jos ei ole, metodi hakee kaikki tiedot tietokannasta ja palauttaa ne näkymään listana. PartialView tarkoittaa, että emme halua palauttaa kokonaista näkymää vaan vain osan siitä eli tässä tapauksessa gridin, jota haluamme päivittää. (**Kuvio 22.)**

```
// Luodaan db olio, millä saadaan yhteys tietokantan
HultaforsDatabaseEntities db = new HultaforsDatabaseEntities();

public ActionResult OrdersGridViewPartial()
{
       // Tarkastetaan, onko istuntoon asettu mitän haku dataa
       if (Session["searchResults"] != null)
       {
       // Jos on, palautetaan hakujen mukainen malli näkymään
              return PartialView("_OrdersGridViewPartial", Ses-
              sion["searchResults"]);
       }
       else
       {
       //Valitaan kaikki tilaukset tietokannasta
       var model = db.OrdersView;
       // Palautetaan kaikki tilaukset mallin mukana näkymään
       return PartialView("_OrdersGridViewPartial", model.ToList());
       }
}
```

**Kuvio 22.** OrdersGridViewPartial()

Home/OrdersGridViewPartial()-metodia kutsutaan aina sovelluksen avautuessa. Tällöin grid näyttää kaikki senhetkiset tilaukset. **(Kuvio 23.)**




**Kuvio 23**. Jäsentäjän palauttama näkymä

## 4.4 Tilaustietojen suodattaminen hakuparametrien perusteella

Tilausten haku toteutettiin käyttämällä DevExpressillä luotuja hakukenttiä, joihin käyttäjä voi syöttää tilausnumeron, tilaustyypin, valita tilauksia päivämäärän mukaan, tilauksen statuksen, asiakasnumeron tai varasto-koodin perusteella. Painamalla Search-näppäintä sovellus kutsuu Search-metodia. **(Kuvio 24.)**

```
<script type="text/javascript">
    // Metodi, mikä hakee tekstikenttien arvot
    function getSearchParameters()
    {
        // Haetaan tekstikenttien arvot
        var orderNumber = $('#OrderNumberTxtBox_I').val();
        var orderType = $('#OrderTypeTxtBox_I').val();
        var ordersOlderThan = $('#OrdersOlderThanComboBox_I').val();
        var order = $('#OrderTxtBox_I').val();
        var customerNumber = $('#CustomerNumberTxtBox_I').val();
        var stockCode = $('#StockCodeTxtBox_I').val();

        // Asetetaan arvot JS objektiin
```

```
        var criterias =
        {
            orderNumber: orderNumber,
            orderType: orderType,
            ordersOlderthan: ordersOlderThan,
            Order: order,
            CustomerNumber: customerNumber,
            StockCode: stockCode
        };
        // Palautetaan objekti funktion mukana
        return criterias;
    }

    // Metodi "hae" nappulalle
    function searchBtnClick(s, e)
    {
        // Haetaan tekstikenttien data getSearchParameters-funktiolla
        var searchParameters = getSearchParameters();
        // Muutetaan JSON muoton
        var jsonData = JSON.stringify(searchParameters);

        // Päivitetään gridi ajax-pyynnöllä
        $.ajax(
        {
            url: '@Url.Action("GridViewPartialSearch")',
            type: 'POST',
            data: jsonData,
            datatype: 'json',
            contentType: 'application/json; charset=utf-8',
            // Onnistuessa päivitetään taulukko
            success: function (data)
            {
                $('#container').html(data);
            }
        });
}
```

**Kuvio 24.** Search ja getSearchParameters-metodit

Metodi kerää kaikki tekstikenttien senhetkiset arvot getSearchParameters()-metodilla. Tieto muutetaan JSON-merkkijonoksi tiedon lähettämistä varten. Tämän jälkeen metodi tekee AJAX-pyynnön jäsentäjän GridViewPartialSearch()-metodille. **(Kuvio 25.)**

```
// Metodi, millä haetaan hakuparametrien avulla.
public ActionResult GridViewPartialSearch(SearchModel searchCriteria)
{
// Luodaan instanssi dataProvider luokasta
       Dataprovider dataProvider = new Dataprovider();

       // Tallennetaan hakutulokset muuttujaan
       var results = dataProvider.searchResults(searchCriteria);

       // Asetataan hakutulokset istuntoon
       Session["searchResults"] = results;

       // Palautetaan näkymä ja malli
       return PartialView("_OrdersGridViewPartial", results.ToList());
```

```
}
```

**Kuvio 25.** GridViewPartialSearch-metodi

GridViewPartialSearch-metodissa käytetään SearchModel-luokkaa parametrina, jotta jäsentäjälle lähetetyt hakuparametrit saadaan tallennettua. SearchModel on yksinkertainen luokka, joka kuvaa hakuja joita käyttäjä voi tehdä. **(Kuvio 26.)**

```
public class SearchModel
{
    public string orderNumber { get; set; }
    public string orderType { get; set; }
    public string ordersOlderThan { get; set; }
    public string order { get; set; }
    public string customerNumber { get; set; }
    public string stockCode { get; set; }
}
```

**Kuvio 26.** SearchModel-luokka

DataProvider-luokka hoitaa sovelluksen tietokantalogiikan. DataProvider-luokassa on searchResults-metodi, jonka parametrina käytetään SearchModel-luokkaa. **(Kuvio 27.)**

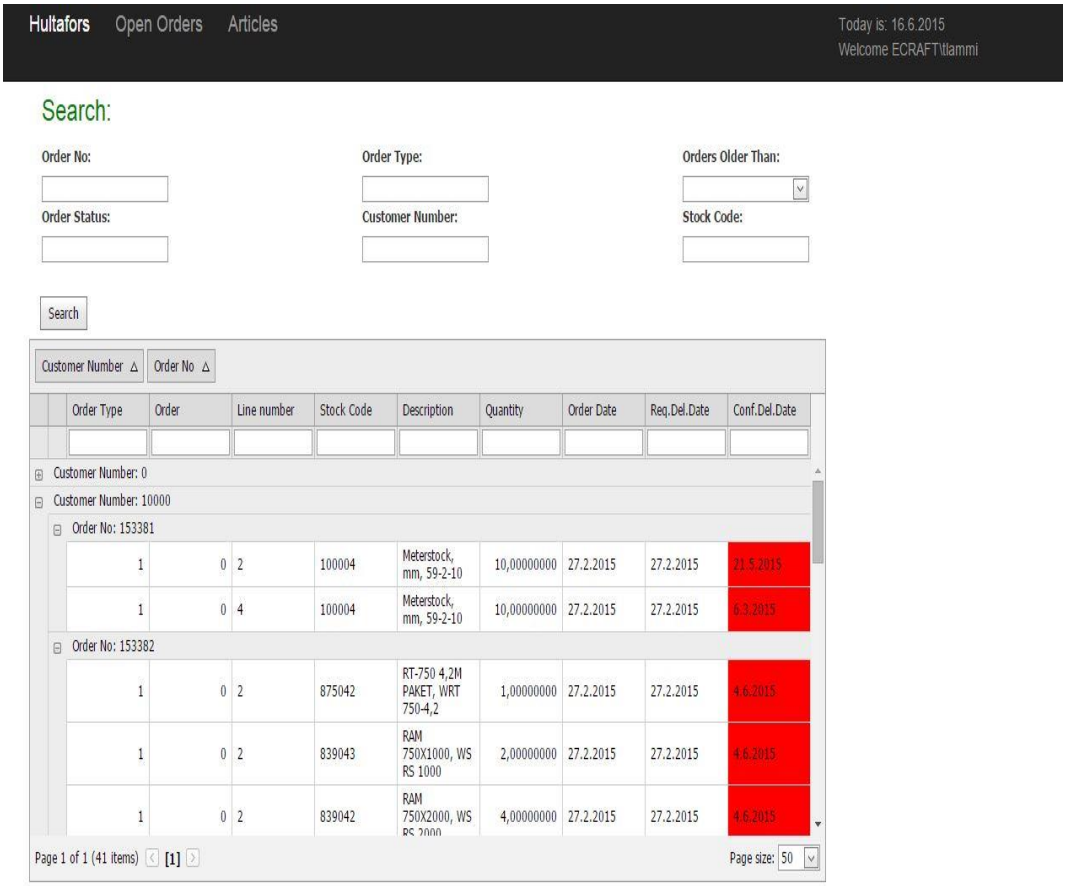
```
// Metodi, mikä palauttaa kaikki hakutulokset
public List<OrdersView> searchResults(SearchModel searchCriteria)
{
       // Luodaan instanssi Entity-Frameworking luomasta luokasta
       HultaforsDatabaseEntities db = new HultaforsDatabaseEntities();

       // Valitaan kaikki tilaukset
       var query =  (from a in db.OrdersView select a);

       // Käydään kaikki hakuehdot läpi yksitellen, ja valitaan muuttu
       jaan dataan jos sitä löytyy
       if (!String.IsNullOrEmpty(searchCriteria.orderNumber))
       {
              var orderNumber = Convert.ToString(searchCriteria.or-
              derNumber);

              query = query.Where(x => x.OrderNo == orderNumber);
       }

       // Stock Code
       if (!String.IsNullOrEmpty(searchCriteria.stockCode))
       {
              var stockCode = searchCriteria.stockCode;
              query = query.Where(x => x.StockCode.Contains(stockCode));
       }

       // Older than
       if (!String.IsNullOrEmpty(searchCriteria.ordersOlderThan))
       {
       var ordersOlderThan = Convert.ToInt32(searchCriteria.ordersOlder-
       Than);
```

```
        var dateCalc = (System.DateTime.Now.AddDays(-ordersOlderThan));

        query = query.Where(x => x.OrdersOlderThan <= dateCalc);
        }

        // Type
        if (!String.IsNullOrEmpty(searchCriteria.orderType))
        {
              var orderType = Convert.ToInt32(searchCriteria.orderType);
              query = query.Where(x => x.OrderType == orderType);
        }

        // Order
        if (!String.IsNullOrEmpty(searchCriteria.order))
        {
              var order = Convert.ToInt32(searchCriteria.order);
              query = query.Where(x => x.Order == order);
        }

        // Customer number
        if (!String.IsNullOrEmpty(searchCriteria.customerNumber))
        {
              var customerNumber = Convert.ToString(searchCriteria.custom-
              erNumber);

              query = query.Where(x => x.CustomerNumber == customer-
              Number);
        }

       // Palautetana muuttuja listana HomeControllerille.
       return query.ToList();
}
```

**Kuvio 27.** DataProvider-luokka

DataProvider-luokassa käytetään Linq-ominaisuutta hyväksi oikean tiedon palauttamiseksi. Linq on Microsoftin kehittämä ominaisuus c#-ohjelmointikieleen, millä pystyy tekemään tehokkaita hakuja, esimerkiksi listoihin. Linqun avulla muuttujaan valitaan aluksi kaikki tilaukset tietokannasta. Tämän jälkeen kaikki hakuparametrit käydään läpi. Jokaisen parametrin kohdalla tarkistetaan IsNullOrEmpty-komennon avulla, onko arvo tyhjä. Jos se ei ole, muuttujaan lisätään hakuparametrin datalinq:n query.Where-komennon avulla. Näin muuttujaan kerätään vain ne tiedot, mitä käyttäjä haluaa hakea. Data palautetaan listamuodossa. Tämän jälkeen sovellus palaa jäsentäjä-luokkaan, ja saadut hakutulokset asetetaan istuntoon. Jäsentäjä palauttaa

mallin näkymään listamuodossa ja hakutulokset päivittyvät AJAX-tyylillä grid-taulukkoon. **(Kuvio 28.)**




**Kuvio 28.** Palautettava näkymä.

Palautettavassa näkymässä käyttäjä pystyy järjestämään sarakkeita haluamallaan tavalla. Käyttäjä voi esimerkiksi vetää otsikkosarakkeita taulukon yläpalkkiin, mikä ryhmittelee taulun sinne vedetyn sarakkeen mukaan. Käyttäjä voi myös klikata taulun otsikkosarakkeita, jolloin sarake järjestyy joko pienimmästä suurimpaan tai suurimmasta pienimpään.

### 4.5 Sarakkeen värin vaihto tilauspäivämäärän mukaan

Yksi sovellukseen vaadittu ominaisuus oli verrata ”Confirmed Delivery Date”-saraketta ”Required Delivery Date”-sarakkeeseen. Jos ”Confirmed Delivery Date” -

sarakkeen päivämäärä on suurempi kuin ”Required Delivery Date”-sarakkeen päivämäärä, sarakkeen taustaväri muutetaan punaiseksi. Näin käyttäjä huomaa poikkeavuuden päivämäärissä helposti ja selkeästi.

Värin vaihtaminen tapahtuu grid-taulukon konfigurointi-tiedostossa **(Kuvio 29.)**, jossa käytettiin taulukon settings.HtmlDataCellPrepared-toimintoa. Tällä toiminnolla päästiin käsiksi yksittäisiin taulukon soluihin. Koodissa otetaan ylös kahden päivämääräsarakkeen arvot, ja muutetaan ne DateTime-muotoon vertaamista varten. Jos konfirmoitu päivämäärä on suurempi kuin vaadittu, konfirmoidun päivämäärän taustaväriä vaihdetaan e.Cell.BackColor-komennolla. Väri vaihtuu punaiseksi. (**Kuvio 30.)**

```
// Värin vaihto toimituspäivien perusteella
settings.HtmlDataCellPrepared = (s, e) =>
{
    if (e.DataColumn.FieldName == "ConfDelDate")
    {
       DateTime  confDelDate  =  Convert.ToDateTime(e.GetValue(e.DataColumn.Field
       Name));
        DateTime reqReleaseDate = Convert.ToDateTime(e.GetValue("ReqDelDate"));

        if (confDelDate > reqReleaseDate)
        {
            e.Cell.BackColor = System.Drawing.ColorTranslator.FromHtml("Red");
        }
    }

};
```

**Kuvio 29.** settings.HtmlDataCellPrepared-asetukset


**Kuvio 30.** Värin vaihtuminen päivämäärän mukaan

# 5 TESTAUS

Sovelluksen testaaminen on prosessi, minkä aikana on tarkoitus selvittää täyttääkö sovellus vaaditut määritellyt, ja pystytäänkö tunnistamaan mahdollisia virheitä koodissa, käyttöliittymässä tai missä tahansa sovellukseen liittyvässä osiossa.

Sovelluksia voidaan testata monin eri tavoin, esimerkiksi black-box-testaus, white-box-testaus ja gray-box-testaus. Black-box-testauksen idea on, että sovelluksen testaaja ei tiedä sovelluksesta mitään, eikä hänellä ole pääsyä sovelluksen lähdekoodiin. Tämä on hyödyllistä, koska ulkopuolisen käyttäminen testaajana antaa näkökulmia mitä sovelluksen kehittäjällä ei välttämättä olisi. White-box-testaamisessa testaajalla on käytössään sovelluksen lähdekoodi. Tällöin testaajan on helppo katsoa ja tutkia, mikä sovelluksessa on vikana, ja sen kautta myös optimoida koodia enemmän. Gray-box-testaamisessa testaajalla on pääsy sovelluksen tietokantaan ja dokumentointiin. Siinä testaaja pystyy tarkentamaan ja suunnittelemaan paremmin mitä sovelluksessa testaa./10/

Sovelluksen testaaminen tapahtui enimmäkseen siten, että ominaisuuksia testattiin sen mukaan mitä ominaisuuksia siihen lisättiin. Koska suurin osa sovelluksesta liittyi tekstikenttiin, testaus osoitettiin enimmäkseen niihin. Tekstikenttiin yritettiin esimerkiksi syöttää erikoismerkkejä, tietoa mitä tietokannassa ei ole ja tyhjää eli välimerkkejä. Sovellusta testattiin kaikilla yleisimmillä selaimilla. Nämä selaimet olivat Internet Explorer, Firefox ja Chrome.

Kehityksen lopussa sovellus testattiin vielä, jotta se täyttää kaikki vaatimusmäärittelyssä olevat ominaisuudet. Tietokannan tauluista löytyi yksi virhe. Taulut yhdistävässä SQL-käskyssä oli pieni ajatusvirhe, joka korjattiin nopeasti. Sovelluksesta ei löytynyt suuria virheitä, koska sitä testattiin kehittämisen aikana jatkuvasti.

# 6 YHTEENVETO

Opinnäytetyön tuloksena saatiin toimiva web-sovellus, joka täytti asiakkaan haluamat vaatimukset. Tärkeimmät ominaisuudet olivat asiakkaiden tilaustietojen näyttäminen grid-taulukossa ja niiden manipulointi.

Opinnäytetyö tuli suorittaa loppuun nopealla aikataululla, sillä minulla alkoi toinen työ, joka meni ristiin tämän työn tekemisen kanssa. Myös työn aloittaminen vähän viivästyi, koska ei tiedetty vielä mitä asiakas sovellukselta haluaa.

Työ oli itselleni sopivan haastava. Minulla ei ollut aikaisempaa kokemusta asp.net MVC-ohjelmointityylistä, eikä sovelluksessa käytetyn DevExpress-työkalun käytöstä. DevExpress mahdollisti sen, ett sovelluksen runko oli nopea luoda, mutta sen muokkaaminen haluttuun lopputulokseen oli haastavaa. Kehityin omasta mielestäni paljon ohjelmoijana, ja sitä tärkeämpänä sain paljon uutta innostusta alaa kohti. Olen varma, että tämän työn tekemisen jälkeen haluan tehdä ohjelmointia työkseni.

Työ tullaan ottamaan käyttöön kun se saadaan asiakkaan palvelinympäristöön pyörimään. Työtä pystyisi kehittämään lisäämällä ”Articles”-osio työhön. Tämä saattaa olla tulevaisuudessa mahdollista, jos asiakas ja Ecraft haluavat niin tehdä. Työhön on myös yleisesti helppo lisätä ominaisuuksia, koska tietokantalogiikka on valmiina taustalla.

# LÄHTEET

/1/ eCraft. Viitattu 10.5.2015. http://www.ecraft.fi/

/2/ Getting started with Visual studio. Viitattu 10.5.2015. https://msdn.microsoft.com/en-us/library/ms165079.aspx

/3/ ASP.NET MVC Overview.Viitattu 10.5.2015. http://www.asp.net/mvc/overview/older-versions-1/overview/asp-net-mvc-overview

/4/ .NET framework Class library. Viitattu 10.5.2015. https://msdn.microsoft.com/en-us/library/zw4w595w%28v=vs.110%29.aspx

/5/ ASP.NET Razor markup. Viitattu 10.5.2015. http://www.w3schools.com/aspnet/razor_intro.asp

/6/ SQL Introduction. Viitattu 10.5.2015. http://www.w3schools.com/sql/sql_intro.asp

/7/ About JavaScript. Viitattu 11.5.2015. https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript#What_is_JavaScript.3

/8/ What is jQuery? Viitattu 14.9.2015. http://www.tutorialspoint.com/jquery/jquery-overview.htm

/9/ Asp.net 4 tutorial. Viitattu 16.9.2015. http://www.w3schools.com/aspnet/

/10/ Software testing tutorial. Viitattu 17.9.2015. http://www.tutorialspoint.com/software_testing/

/11/ What is AJAX. Viitattu 17.9.2015. http://www.tutorialspoint.com/ajax/ajax_quick_guide.htm

/12/ What Is Javascript. Viitattu 1.10.2015. http://javascript.about.com/od/reference/p/javascript.htm

/13/ Javascript Libraries. Viitattu 1.10.2015. http://www.w3schools.com/js/js_libraries.asp

/14 .NET overview. Viitattu 15.10.2015. http://www.codeproject.com/Articles/20694/Net-Framework

/15/ Introduction to visual studio. Viitattu 1.10.2015. https://msdn.microsoft.com/en-us/library/fx6bk1f4(v=vs.71).asp