



VAASAN AMMATTIKORKEAKOULU  
VASA YRKESHÖGSKOLA  
UNIVERSITY OF APPLIED SCIENCES

Olli Asikainen

Levdoc Global  
dokumentaatio-tilausjärjestelmä

Tekniikka ja liikenne  
2013

## TIIVISTELMÄ

Tekijä	Olli Asikainen
Opinnäytetyön nimi	Levdoc Global dokumentaatiotilausjärjestelmä
Vuosi	2013
Kieli	suomi
Sivumäärä	29 + 3 liitettä
Ohjaaja	Martti Mustonen

---

Levdoc Global on Wärtsilä Services Technical Information -osastoille kehitetty selainpohjainen ohjelma, johon tallennetaan dokumentaatiotilauksia sekä niiden aikataulutuksia. Ohjelman tarkoitus on helpottaa eri maiden Technical Information -osastojen yhteistyötä tuomalla tilaukset yhteiseen järjestelmään.

Ohjelmisto on kirjoitettu C# -ohjelmointikielellä käyttäen ASP.NET MVC -ohjelmistokehystä. Tiedon tallennus tapahtuu SQL Server -tietokantaan käyttäen apuna Entity Framework -ohjelmistokehystä. Tietomallin rakentamiseen käytettiin avoimen lähdekoodin SQL Developer Data Modeler -työkalua sekä ADO.NET Entity Data Model Designeria. Osa käyttöliittymästä tuotetaan käyttäen jQuery UI sekä Knockout.js -kirjastoja.

Verkkopalvelun kehittäminen oikein valituilla sekä nykyaikaisilla työkaluilla osoittautui mielekkääksi ja tehokkaaksi.

## ABSTRACT

Author	Olli Asikainen
Title	Levdoc Global – Documentation Handling System
Year	2013
Language	Finnish
Pages	29 + 3 Appendices
Name of Supervisor	Martti Mustonen

---

The objective of the thesis was to develop the Levdoc Global web application for Wärtsilä Services Technical Information department, for dealing with documentation orders and their scheduling. The purpose of the application is to ease the work and collaboration between Technical Information departments in different countries by bringing their documentation orders to a single shared system.

The application was written in C# programming language and it uses the ASP.NET MVC framework. Data is stored to a SQL Server database with the help of Entity Framework. The data model was built using SQL Developer Data Modeler tool and ADO.NET Entity Data Model Designer. The user interface uses libraries such as jQuery UI and Knockout.js.

Developing a web service with well selected and modern tools proved to be sensible and efficient.

## SISÄLLYS

### TIIVISTELMÄ

### ABSTRACT

1	JOHDANTO.....	6
2	YRITYS.....	7
	2.1 Wärtsilä.....	7
	2.2 Wärtsilä Finland Oy.....	7
3	TYÖKALUT JA KIRJASTOT.....	8
	3.1 Microsoft Visual Studio.....	8
	3.2 .NET.....	8
	3.3 C# .....	8
	3.4 LINQ.....	9
	3.5 ASP.NET MVC .....	9
	3.6 ADO.NET Entity Framework.....	10
	3.7 jQuery, jQuery UI ja jQuery validation .....	11
	3.8 Microsoft SQL Server ja IIS .....	11
	3.9 Oracle SQL Developer Data Modeler.....	11
	3.10 Git .....	11
4	LEVDOC GLOBAL.....	12
	4.1 Vaatimukset .....	12
	4.1.1 Käyttötavat .....	12
	4.1.2 Tieto-oliot.....	12
	4.1.3 Roolit.....	14
	4.2 Suunnittelu .....	15
	4.2.1 Tietomallin luominen .....	15
	4.2.2 Käyttöliittymän suunnittelu ja toiminta .....	16
	4.3 Toteutus.....	16
	4.3.1 Web Platform Installer .....	16
	4.3.2 NuGet .....	17
	4.3.3 Tietokanta.....	18
	4.3.4 Täysnäkyä .....	20

4.3.5	Yksityiskohtanäkymä.....	22
4.3.6	SAP -tietokanta .....	24
4.4	Parempi toteutus.....	24
4.4.1	Knockout.js .....	25
4.4.2	Model first.....	25
4.4.3	Yhden sivun ohjelma.....	26
5	LOPPUPÄÄTELMÄT .....	28
	LÄHTEET.....	29
	LIITTEET	

## MERKINNÄT JA LYHENTEET

HTML	Kuvauskieli web-sisällön näyttämiseen
Javascript	Ohjelmointikieli, jota käytetään muun muassa web-sivuilla
AJAX	Tekniikka, joka mahdollistaa HTTP-kyselyt Javascriptillä
HTTP	Web-palvelimen ja –selaimen välinen keskusteluprotokolla
Installaatio	Wärtsilän tuotteiden käyttökohde, kuten laiva tai voimalaitos
DDL	Tietokannan rakenteen määrittelevä kieli
LDAP	Protokolla mm. käyttäjien hakemiseen Windows-ympäristössä
SAP	Eräs suuri tietojärjestelmä
DOM	Ohjelmointirajapinta dokumentin sisällön muokkaamiseen
JSON	Javascript-olioiden esitysmuoto tekstinä

## 1 JOHDANTO

Wärtsilä Services tarvitsee dokumentaatiotilausten hallinnointiin nykyaikaisen selainpohjaisen sovelluksen, joka otetaan käyttöön kaikissa Wärtsilä Services Technical Information –osastoissa. Tähän mennessä kukin osasto on hoitanut tilauksensa omissa sisäisissä järjestelmissään. Ohjelman tarkoituksena on tuoda kaikki dokumentaation osatilaukset saman järjestelmän alle, helpottaen osastojen välistä yhteistyötä ja mahdollistaen jokaisen osatilauksen tilan seurannan reaaliajassa.

Levdoc Global sai varsinaisesti alkunsa kesällä 2012, jolloin aloitin työt Wärtsilä Finland Oy:llä. Vuonna 2010 aloitettiin Contents Management –tiimin työkalun ”Levdocin” uudelleensuunnittelu. Työkalun kehitys keskeytyi vuoden 2012 alussa, kun sen pääkehittäjä siirtyi muihin tehtäviin.

Nimi Levdoc tulee ruotsalaisista sanoista **leverans** ja **dokumentation**, tarkoittaen dokumentaatioitoimituksia.

## **2 YRITYS**

### **2.1 Wärtsilä**

Wärtsilä on kansainvälinen yritys, joka toimittaa ratkaisuja merenkulun ja energiamarkkinoiden tarpeisiin. Yritys työllisti vuonna 2012 suunnilleen 19 000 henkeä yhteensä 70 maassa. Samana vuonna yrityksen liikevaihto oli noin 4,7 miljardia euroa. Yritys perustettiin Suomessa vuonna 1834. /1/

Yritys jakautuu kolmeen yksikköön:

- Ship power                   ➤ Meriteollisuus
- Power plants               ➤ Voimalaratkaisut
- Services                     ➤ Huolto ja palvelut

### **2.2 Wärtsilä Finland Oy**

Wärtsilä Finland Oy on Wärtsilä-konsernin tytäryhtiö Suomessa. Toimipisteitä on Helsingissä, Vaasassa sekä Turussa. Wärtsilä työllistää Suomessa yli 3600 henkilöä. /2/



## 3 TYÖKALUT JA KIRJASTOT

### 3.1 Microsoft Visual Studio

Visual Studio on –ohjelmointiympäristö, jonka avulla .NET -koodin kirjoittaminen on miellyttävää. Ennakoiva syöttö eli ”IntelliSense” vähentää näppäinpainalluksia huomattavasti ehdottamalla älykkäästi oikeat luokka-, muuttuja- ja funktionimet jo muutaman merkin syötteestä. Visual Studioon saa integroitua muun muassa versionhallinnan työkalut, kuten Visual SVN ja Visual Studio Tools for GIT. Monet Visual Studion versiot sisältävät myös tiedonmallinnustyökalun.

### 3.2 .NET

.NET on alun perin vuonna 2002 julkaistu ohjelmistokehys, jonka tarkoituksena on tarjota Windows -ohjelmille alusta, jolle kehitetyt ohjelmat toimivat kaikilla käyttöjärjestelmän versioilla, sekä niitä pystyy kirjoittamaan kaikilla kielillä, jotka noudattavat CLI-spesifikaatiota. Tämän kaiken mahdollistaa se, että .NET-ohjelmat suoritetaan CLR-käyttöympäristössä, joka hoitaa kommunikoinnin käyttöjärjestelmän kanssa. .NET tarjoaa kattavan määrän luokkia ja kutsuja, joilla voi kehittää Windowsille ohjelman kuin ohjelman. /3/

### 3.3 C#

C# on .NET-ympäristöön kehitetty ohjelmointikieli. Sen ominaisuuksia ovat C++-tyylinen syntaksi ja nopeus, vahva tyyppitys sekä Javan helppokäyttöisyys. Sisäänrakennettuna löytyy automaattinen roskienkeruu ja taulukon rajojen tarkistus muistivuotojen sekä ylivuototilanteiden välttämiseksi.

### 3.4 LINQ

LINQ on eräs .NET-ohjelmistokehyksen komponentti, jolla voi SQL-tyylisellä syntaksilla hakea tietoa erilaisista tietorakenteista (**Kuva 1**). Sen helppolukuisuuden ansiosta sitä voi hyvällä omallatunnolla käyttää esimerkiksi MVC-näkymissä.

```

List<string> array = new List<string>()
{
    "hello",
    "world",
    "this",
    "is",
    "an",
    "example"
};

string found = null;

foreach (string item in array)
{
    if (item.Equals("world"))
    {
        found = item;
        break;
    }
}

Console.WriteLine("Found: {0}", found);

```

```

List<string> array = new List<string>()
{
    "hello",
    "world",
    "this",
    "is",
    "an",
    "example"
};

string found = (from item in array
                where item.Equals("world")
                select item).First();

Console.WriteLine("Found: {0}", found);

```

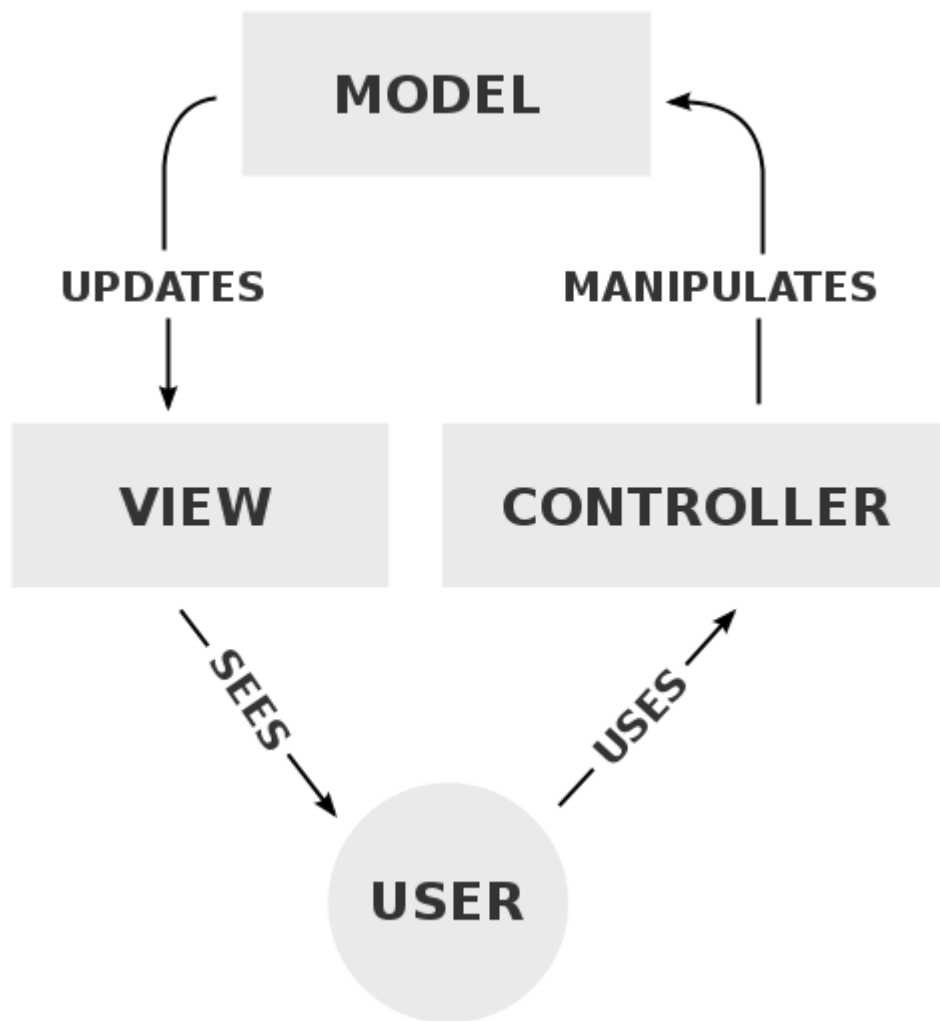
***Kuva 1:** Vasemmalla perinteinen silmukkaa käyttävä tapa etsiä alkio taulukosta, oikealla LINQ-toteutus.*

### 3.5 ASP.NET MVC

MVC (Model-View-Controller) on web-ohjelmille tarkoitettu ohjelmistokehys, joka toteuttaa malli–näkyä–käsittelijä –arkkitehtuurin (**Kuva 2**). Sen ideana on jakaa ohjelmakoodi osiin, joissa niitä on selkeä ylläpitää. /4/

Käsittelijä ottaa vastaan sivupyynnöt ja hoitaa tiedonkäsittelyn. Malliin kuuluu tieto, joka halutaan näyttää. Näkyä määrittää, kuinka tieto näytetään.

MVC 3:sta lähtien näkymien määrittämiseen on käytetty Razor-moottoria vanhan WebForms-moottorin sijaan. Molemmat ovat tapoja sisällyttää toiminnallisuutta ja viittauksia ohjelmakoodiin näkymään aiheuttamatta paljoo ”spagettikoodia”.



*Kuva 2: Model View Controller*

### 3.6 ADO.NET Entity Framework

Entity Framework on Object-Relational Mapping -tekniikan toteuttava ohjelmistokehys .NET-ympäristöön. Se tarjoaa helposti ohjelmoitavan abstraktin rajapinnan tietokantaan. Sen ominaisuuksiin kuuluu myös muutosten seuraaminen sekä transaktiot.

### **3.7 jQuery, jQuery UI ja jQuery validation**

jQuery on Javascript-kirjasto, joka tarjoaa paljon hyödyllisiä toimintoja dynaamisesti toteutetulle verkkosivulle. Mukana on muun muassa helpottavia funktioita DOM-elementtien ohjelmalliseen läpikäyntiin ja manipulointiin, tapahtumienkäsittelyyn, animaatioiden toteuttamiseen sekä AJAX-pyyntöihin.

jQuery UI on liitännäinen jQueryyn. Se tarjoaa uudenlaisia visuaalisia komponentteja verkkosivulle, kuten upotettavat välilehdet, päivämäärän poimijat sekä tekstikentät ehdottavalla syötöllä.

jQuery validation lisää kenttiin selaimen puolen tarkastukset, jolloin käyttäjää on helppo ohjeistaa tilanteissa, joissa syötetty tieto ei kelpaa.

### **3.8 Microsoft SQL Server ja IIS**

SQL Server on relaatiotietokantaohjelmisto, joka sopii hyvin yhteen Visual Studioon sekä Entity Frameworkin kanssa. Sen mukana tulee hallintatyökalu nimeltä Management Studio.

IIS (Internet Information Services) on web-palvelinympäristö.

### **3.9 Oracle SQL Developer Data Modeler**

SQL Developer Data Modeler on ilmainen tiedonmallinnustyökalu. Ollessaan Oraclen tuote, se tuottaa parasta koodia Oraclen omiin tietokantoihin. Sen avulla saa kuitenkin täysin kelpoa koodia myös SQL Server –tietokantaan.

### **3.10 Git**

Git on hajautettu versionhallintajärjestelmä, jota käytetään mm. Linuxin ytimen kehitystyössä. Sen suunnittelun aloitti Linux Torvals vuonna 2005. Sen etuna on helppo haaroittaminen ja haarojen yhdistäminen. Microsoft tarjoaa Visual Studioon Git-liitännäisen, jota käytin projektissa.

## 4 LEVDOC GLOBAL

### 4.1 Vaatimukset

Ohjelman tarkoituksena on tarjota globaali ja helppo työkalu dokumentaation tilausten hallintaan. Hallinta käsittää tilausten lisäämisen ja muuttamisen, sekä aikataulujen luomisen. Ohjelman tulee hyödyntää Wärtsilän tietokantoja siinä määrin missä se on mahdollista. Käytön tulee olla riittävän suoraviivaista, niin ettei ohjelman käyttöönotto vaadi ylimääräistä opetusta. Virheilmoitusten tulee olla helppo ymmärtää.

#### 4.1.1 Käyttötavat

*Taulukko 1: Olennaiset käyttötavat.*

Käyttötapa	Kuvaus
Enter order	Tilauksen lisäys järjestelmään
Delivery planning	Toimituspäivämäärän lisääminen
Report	Toimituspäivämäärän varmistus
View	Tilauksen tarkastelu

#### 4.1.2 Tieto-oliot

*Taulukko 2: Määritellyt tieto-oliot, niiden tyypit ja kuvaukset.*

Olio	Tyyppi	Kuvaus
Order number	Numero	Tilauksen tunniste
Order date	Aika	Tilauspäivämäärä
Customer	Merkkijono	Tilauksen maksaja
IOS / WBS / Service notification	Merkkijono	Tunniste toisiin järjestelmiin

Project name	Merkkijono	Projektin nimi
Installation number	Merkkijono	Installaation uniikki tunniste
Category (of product)	Merkkijono	Tuotteen luokittelu (moottori, vaihdelaatikko, ...)
Product number	Merkkijono	Tuotteen uniikki tunniste
Product reference type	Merkkijono	Tuotteen perhe
Product type	Merkkijono	Tuotteen tarkka tyyppi
Info product type	Merkkijono	Infotuotteen tyyppi (manuaali, varaosaluettelo, ...)
Info system	Merkkijono	Infotuotteen järjestelmä (nidottu kansio, dvd, elektroninen, ...)
Language	Merkkijono	Infotuotteen kieli
Copies	Numero	Infotuotteen kopioiden määrä
Order contact	Merkkijono	Tilauksen yhteyshenkilö
Requested delivery date	Aika	Asiakkaan pyytämä toimituspäivämäärä tilaukselle
Planned delivery date	Aika	Tilauksen suunniteltu toimituspäivämäärä
Actual delivery date	Aika	Tilauksen varsinainen toimituspäivämäärä
Organisation	Merkkijono	Organisaatio, joka hoitaa tilauksen / osatilauksen
Order status	Merkkijono	Tilauksen tila
Start of assembly date	Aika	Päivämäärä, jolloin tuotteen (moottorin) kokoaminen on aloitettu
Responsible for order	Merkkijono	Tilauksen / osatilauksen vastuuhenkilö
Comments	Merkkijono	Tilauksen / osatilauksen kommentit
Publication	Merkkijono	Polku elektroniseen dokumenttiin

### 4.1.3 Roolit

*Taulukko 3: Roolit käyttäjille.*

Rooli	Oikeudet
User	Lukuoikeudet
(Order) Coordinator	Lukuoikeudet Tilausten lisääminen ja muokkaus
(Delivery) Planner	Lukuoikeudet Toimituspäivämäärien suunnittelu
Executer	Lukuoikeudet Toimituspäivämäärien varmistaminen
Administrator	Kaikki

## 4.2 Suunnittelu

### 4.2.1 Tietomallin luominen

Tietomallin rakentaminen alkoi hajoittamalla tieto-oliot neljään loogiseen osaan (Taulukko 4.).

*Taulukko 4: Tieto-oliot jäsenneltynä.*

<ul style="list-style-type: none"> <li>• Installation           <ul style="list-style-type: none"> <li>○ Installation number</li> <li>○ Project name</li> <li>○ Customer</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>• Product           <ul style="list-style-type: none"> <li>○ Product number</li> <li>○ Category</li> <li>○ Product reference type</li> <li>○ Product type</li> <li>○ Start of assembly date</li> </ul> </li> </ul>
<ul style="list-style-type: none"> <li>• Order           <ul style="list-style-type: none"> <li>○ Order number</li> <li>○ Order date</li> <li>○ IOS / WBS / Notification</li> <li>○ Order contact</li> <li>○ Requested delivery date</li> <li>○ Actual delivery date</li> <li>○ Order status</li> <li>○ Comments</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>• OrderLine           <ul style="list-style-type: none"> <li>○ Info product type</li> <li>○ Info system</li> <li>○ Language</li> <li>○ Copies</li> <li>○ Responsible for order</li> <li>○ Comments</li> <li>○ Publication</li> </ul> </li> </ul>

Toteuttamisen aikana tietomalliin tehtiin useita muutoksia. Kattavimman tietomallin (Liite 1.) luomiseen käytettiin Oraclen SQL Developer Data Modeler – työkalua.



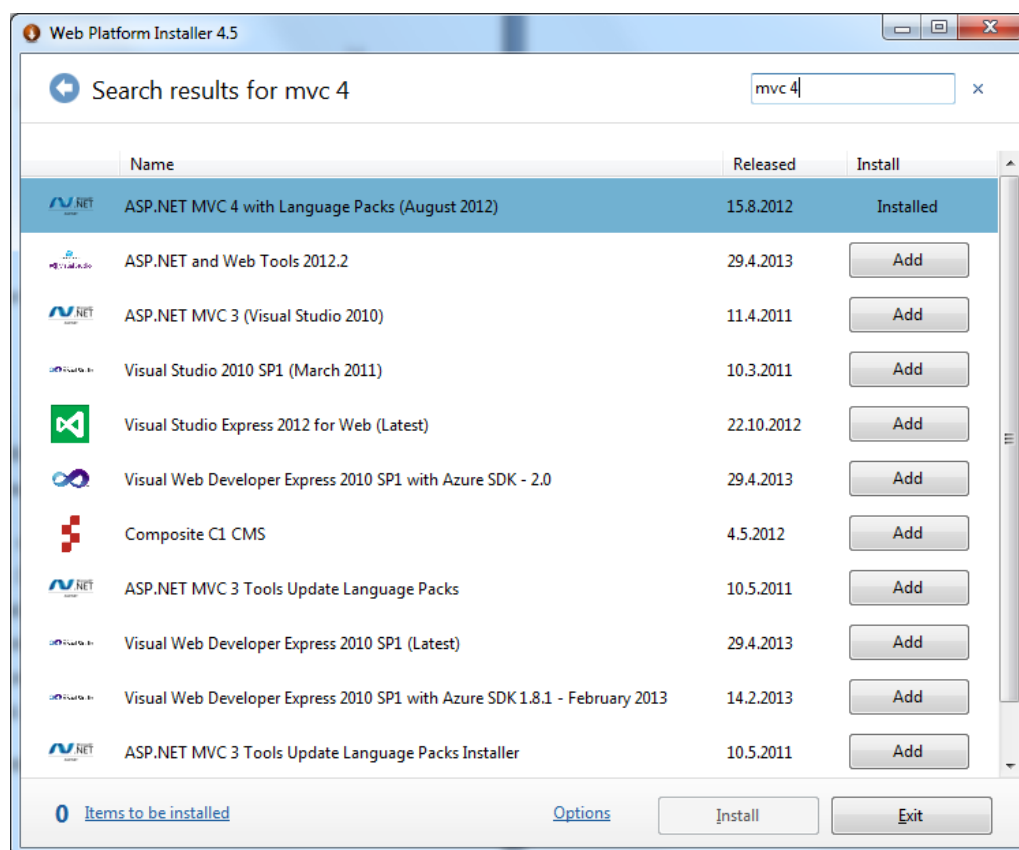
## 4.2.2 Käyttöliittymän suunnittelu ja toiminta

Käyttöliittymään tuli saada (Liite 2.) näkyviin kerralla mahdollisimman paljon niin, että yhdellä vilkaisulla selviää minkälaiset tilaukset ovat vireillään. Kaikki tieto ei kuitenkaan yksinkertaisesti mahtunut samalle sivulle, joten osa tiedosta aukeaa uusiin ponnahtusikkunoihin. Käyttöliittymän toimintaa selventää käyttötapauskaavio (Liite 3.).

## 4.3 Toteutus

### 4.3.1 Web Platform Installer

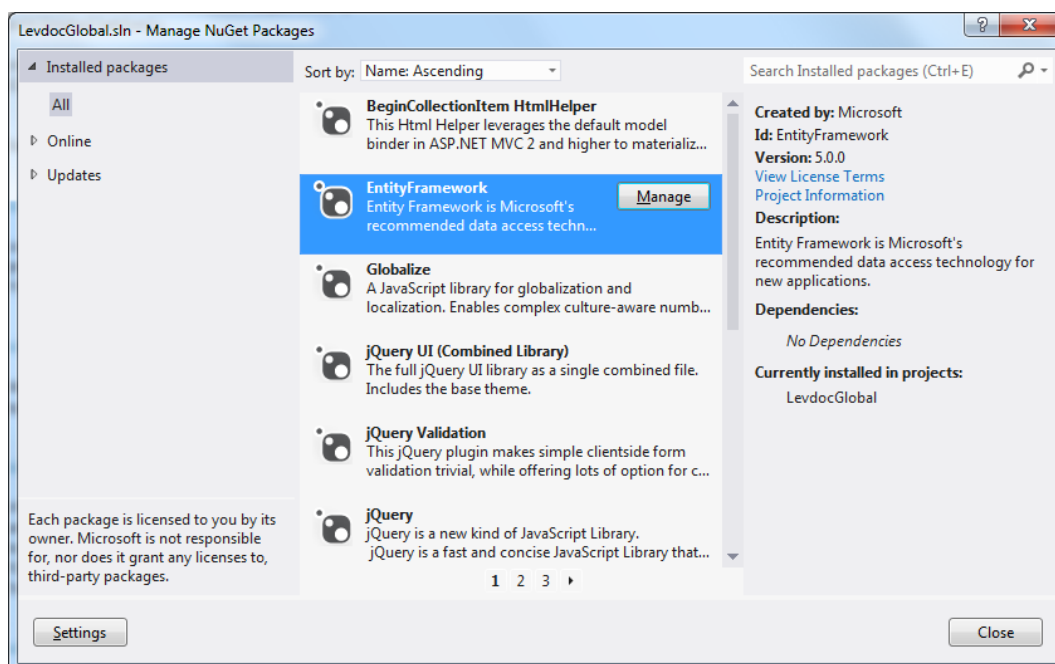
Sovelluksen kehitys lähti käyntiin MVC:n asennuksella Microsoftin Web Platform Installerin kautta (**Kuva 3.**). Sen avulla on mahdollista asentaa myös muita Microsoftin ilmaisia kehitystyökaluja, kuten .NET-ympäristö, ilmainen Visual Studio Express ja IIS Express.



**Kuva 3:** MVC 4 oli projektin toteuttamishetkellä tuorein julkaisu.

### 4.3.2 NuGet

NuGet on avoimen lähdekoodin pakettimanageri ja liitännäinen Visual Studioon, jonka avulla on helppo pitää web-sovelluksen projektikohtaiset kolmannen osapuolen liitännäiset hallinnassa ja ajan tasalla (**Kuva 4.**) /5/



**Kuva 4:** Osa projektin liitännäisistä näkyvissä NuGet -pakettimanagerissa.

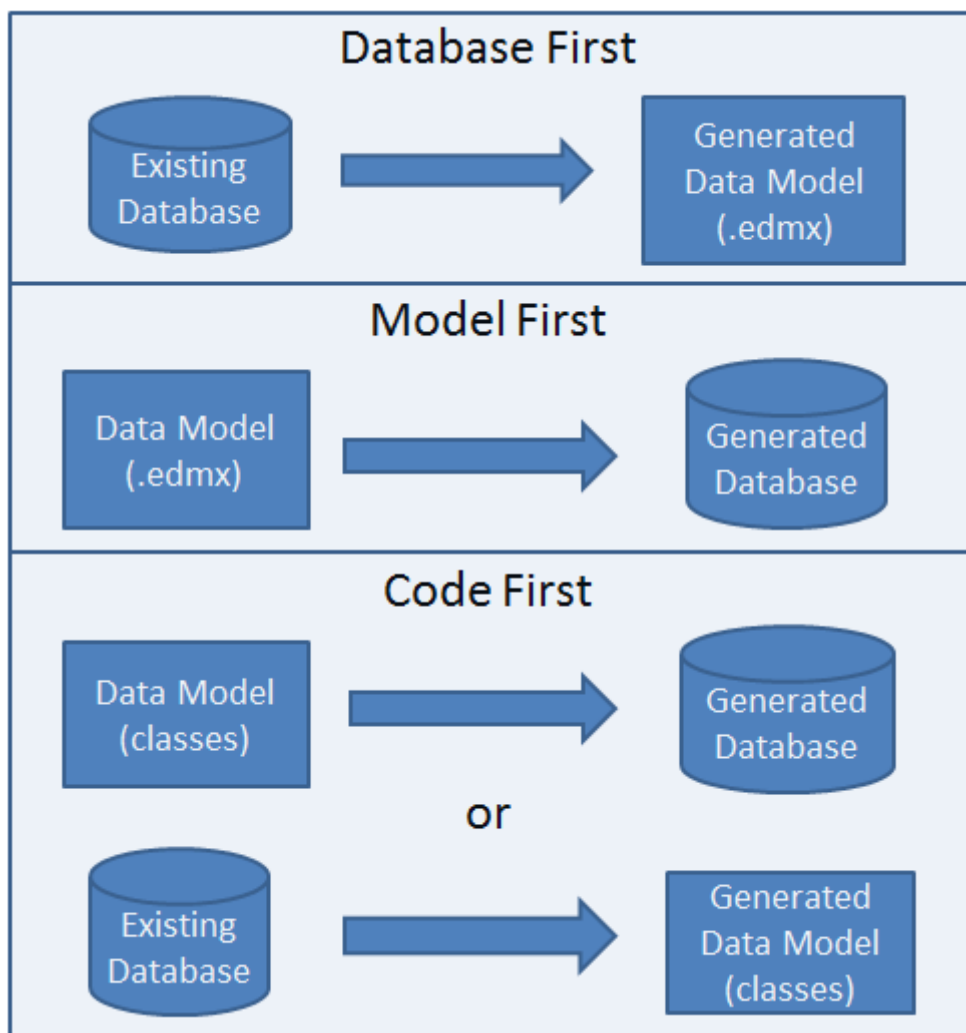
MVC:n toiminta nojaa selainpuolella vahvasti kehittäjille tuttuun ja hyväksi havaittuun jQuery-kirjastoon, joten se on aina valmiiksi asennettuna uusissa MVC-projekteissa. Myös sen osat, kuten Unobtrusive AJAX sekä Validation löytyvät joka projektista. Näiden lisäksi projektissa käytetään muun muassa Globalize-liitännäistä, jonka avulla syötetyt päivämäärät ja pilkkuluvut saadaan lokalisoitua validointia varten. Muita lisäosia ovat mm. Entity Framework, BeginCollectionItem-apuri ja Oraclen Managed Data Provider –ajuri.

Yhteensä projektissa on NuGetin hallitsemia liitännäisiä yli 20, joista karkeasti puolet on ASP.NETin pakollisia komponentteja. Loput ovat erinäisiä jQueryn osia ja edellisen kappaleen loppupuolen erikoisemmat liitännäiset.

### 4.3.3 Tietokanta

Tietokanta luotiin SQL Server 2008 -tietokantaan käyttämällä SQL Developer Data Modelerista saatua DDL -tiedostoa.

Entity Frameworkin kanssa käytettiin ns. ”database first” –lähestymistapaa (**Kuva 5.**), joka tarkoittaa käytännössä siis sitä, että Entity Framework designer ottaa yhteyden tietokantaan ja muodostaa modelin tietokannan tauluista. Vastakkainen lähestymistapa olisi ”model first”, jossa tietokanta luodaan designerin modelista. Kolmas vaihtoehto on ”code first”, jossa ei luoda modelia vaan kirjoitetaan käsin luokat, jotka vastaavat tauluja ja rivejä.



*Kuva 5: Lähestymistavat käsitellä tietoa Entity Frameworkilla.*

Loppupeleissä kaikki kolme tapaa tuottavat tauluille **kuvan 6** näköisiä luokkia.

```
//-----
// <auto-generated>
// This code was generated from a template.
//
// Manual changes to this file may cause unexpected behavior in your application.
// Manual changes to this file will be overwritten if the code is regenerated.
// </auto-generated>
//-----

namespace LevdocGlobal
{
    using System;
    using System.Collections.Generic;

    public partial class Order
    {
        public Order()
        {
            this.Miscs = new HashSet<Misc>();
            this.OrderEvents = new HashSet<OrderEvent>();
            this.OrderLines = new HashSet<OrderLine>();
        }

        public int ID { get; set; }
        public string OrderName { get; set; }
        public string InstallationNo { get; set; }
        public string Requester { get; set; }
        public string Contact { get; set; }
        public string Status { get; set; }
        public bool Claim { get; set; }
        public string Comments { get; set; }

        public virtual Installation Installation { get; set; }
        public virtual ICollection<Misc> Miscs { get; set; }
        public virtual User OrderContact { get; set; }
        public virtual User OrderRequester { get; set; }
        public virtual ICollection<OrderEvent> OrderEvents { get; set; }
        public virtual ICollection<OrderLine> OrderLines { get; set; }
    }
}
```

**Kuva 6:** Designerin luoma Order -luokka.

Eräs ärsyttävä ongelma EF designerissa on se, ettei se ymmärrä lisätä kentille attribuutteja, kuten Required-attribuuttia, vaikka esimerkiksi ”database first” – lähestymistapaa käyttäessä tämä tieto tulee tietokannasta. EF Designerin luomia luokkia täytyy siis vielä muokata lisäämällä niihin halutut attribuutit. Mikäli modelin avaa designerissa ja tallentaa, katoavat attribuutit jälleen tiedoston ylikirjoituessa, joten designeria käyttäessä tämä on tärkeä ottaa huomioon.

#### 4.3.4 Täysnäköymä

Täysnäköymä (**Kuva 7.**) toteuttaa kaavailun käyttöliittymän (**Kuva 4.**), joskin siitä jätettiin tarpeettomana pois timeline-osio. Täysnäköymässä näkyy valitun installaation tiedot, kaikki tuotteet (moottorit, vaihdelaatikat, jne.) sekä tilaukset ja niiden tilausrivit.

The screenshot shows a web application interface for 'Installation details'. The browser address bar shows 'fi01733/FullView?InstallationNo=1'. The page has a blue header with the 'Wärtsilä CONTENTS MANAGEMENT' logo. The main content area is divided into several sections:

- Installation details**: A form with fields for 'Installation' (value: 1), 'Installation name' (value: Not a real installation), 'Project name' (value: Project name), 'Customer' (value: Ship Power), and 'Special project' (value: Standard).
- Products**: A table with columns: Product number, Category, Type, Reference type. It contains two rows of test data.
 

Product number	Category	Type	Reference type
Not a real product #1	Engines	8L50	W50
Not a real product #2	Engines	8L50	W50
- Orders**: A table with columns: Shown, Order name, Claim, Requester, Contact, Date, Requested, Status. It contains one row of test data.
 

Shown	Order name	Claim	Requester	Contact	Date	Requested	Status
<input checked="" type="checkbox"/>	#1	<input type="checkbox"/>	?	OAS003	1.1.2013	19.1.2013	New
- Orderlines**: A table with columns: Order name, Info product, Language, Stage, Copies, Organisation, Responsible, Estimate, Actual, Status. It contains one row of test data.
 

Order name	Info product	Language	Stage	Copies	Organisation	Responsible	Estimate	Actual	Status
#1	Manual (Binder)	English	Final	1	Wärtsilä Finland	OAS003	1.3.2013	15.3.2013	New

At the bottom of the form, there is a 'Save changes' button. The footer of the page includes 'Build 20130311.1645', a link to 'Report bug / Request feature', and '© 2013 Wärtsilä'.

**Kuva 7:** Täysnäköymässä testidataa.

Tiedon näyttämiseen käytetään apuna MVC:n HtmlHelper -luokan EditorFor ja DisplayFor-metodeita (**Kuva 8.**), jotka soveltuvat niille annetun olion tiedot sen tyyppin mukaiseen mallipohjaan (template). Mallipohjan saa halutessaan myös määrittellä itse (**Kuva 9.**), jolloin säästyy turhalta monistamiselta ja koodi on näin ollen helpompi ylläpitää.

```

@foreach (LevdocGlobal.Product product in (from p in Model.Products orderby p.Category, p.ProdNo select p))
{
    if (ViewBag.CanEdit(product))
    {
        @Html.EditorFor(p => product)
    }
    else
    {
        @Html.DisplayFor(p => product)
    }
}
</tbody>

```

**Kuva 8:** Näkymässä käytetään *EditorFor*- ja *DisplayFor* -metodeita.

```

@model LevdocGlobal.Installation
<tr>
<td>@Html.TextBoxFor(m => m.InstallationNo, new { @Class = "text-box", @readonly = true })</td>
<td>@Html.EditorFor(m => m.InstallationName)</td>
<td>@Html.EditorFor(m => m.ProjectName)</td>
<td>@Html.DropDownListFor(m => m.Customer_ID, LevdocGlobal.Program.Utilities.Selected((SelectList
<td>@Html.DropDownListFor(m => m.SpecialProject, LevdocGlobal.Program.Utilities.Selected((Select
</tr>

```

**Kuva 9:** *EditorFor*-mallipohja *Installation* -oliolle.

Rivien oikealla puolella olevat plus- ja miinus -napit joko lisäävät tai poistavat rivejä.

Plus -napin painaminen lähettää palvelimelle AJAX -kyselyn, jonka vastauksena palautetaan MVC:llä tuotettu HTML -koodi riviä varten. Palvelimen valmiiksi muotoilema koodi lisätään jQueryn append -metodilla osaksi sivua, sekä tietenkä formia. Toteutuksen heikkous on kyselyn latenssi, palvelimesta riippuen kuitenkin alle 100 ms. Vahvuutena saavutetaan helposti ylläpidettävä ja joustava tapa tuottaa uusia rivejä.

Miinus-nappi sen sijaan poistaa sen vieressä olevan rivin. Mikäli rivi on tietokannassa, jätetään formiin Remove-kenttä, jossa on poistettavan rivin tunniste. Jos rivi on lisätty sivulle jälkikäteen eikä se sijaitse tietokannassa, tulee se poistaa vain sivulta (ja näin ollen formista), jolloin sitä ei koskaan lähetetä palvelimelle tallennusta varten.

Lisätyt rivit, rivien poistot, sekä muut muutokset tallentuvat tietokantaan ”Save changes” -napin painalluksella. Ennen kuin muutokset lähetetään palvelimelle, kaikki kentät tarkistetaan jQuery validationin avulla. Mikäli pakollisia kenttiä

jätetään tyhjiksi tai jotain tietoa ei voida hyväksyä (**Kuva 10.**), maalataan se punaiseksi ja tallentaminen estetään. Jos tiedot hyväksytään, lähetetään ne palvelimelle ja selain ohjautuu sivulle, joka kertoo tietojen tallentuneen. Palvelimella kontrolleri tarkistaa Entity Frameworkin avulla, mitkä sarakkeet riveistä muuttuivat ja tekee tarvittavat muutokset tietokantaan.

<b>Events</b>	
<b>Description</b>	<b>Date</b>
Estimated delivery	13.13.2013
Actual delivery	15.3.2013

*Kuva 10: Päivämäärä ei kelpaa.*

Osassa kenttiä käytetään ennustavaa jQuery autocomplete-widgettiä automaattiseen tekstintäyttöön (**Kuva 11.**). Esimerkiksi tuotteiden kohdalla installaatioissa on usein saman tyyppisiä tuotteita, jolloin ennustava syöttö nopeuttaa tuotteiden lisäämistä.

Henkilöiden kohdalla (Requester, Contact, Responsible) käytetään myös ennustavaa syöttöä, johon tieto saadaan palvelinpäässä LDAP:llä käyttäen PrincipalSearcher-luokkaa. Tilausrivillä Responsible-henkilön muuttaminen vaihtaa myös Organisation-kentän vastaamaan valitun henkilön toimipaikkaa.

#### 4.3.5 Yksityiskohtanäkymä

Kaavaillussa käyttöliittymässä (**Kuva 4.**) olevat ponnahdusikkunat toteuttaa yksityiskohtanäkymä (**Kuva 11.**). Se toimii käytännössä samoin kuin täysnäköisyys, eli rivien lisääminen ja poistaminen, tiedon tarkistukset ja tallentaminen tapahtuu samalla tavoin.

Yksityiskohtanäkymään pääsee painamalla täysnäköisyys valitun rivin oikealla puolella olevaa muistion näköistä kuvaketta. Vaikka yksityiskohtanäkymä avautuu täysnäköisyydestä, on se täysin itsenäinen kokonaisuutensa ja siihen tehdyt

muutokset tallentuvat ”Save changes” –napista tietokantaan, vaikka alla olevan täysnäkömman muutokset kumottaisiin.

Yksityiskohtanäkymässä määritellään muun muassa tapahtumat (Events -kohta), joista osa näytetään täysnäkömmissä. Kun yksityiskohtanäkymässä tehdään muutoksia täysnäkömmissä näkyviin kenttiin, kuten ”Estimated Delivery”, suoritetaan tallennusruudussa Javascript-koodinpätkä, joka hakee AJAXilla päivitettyt kohdat täysnäkömään, jolloin muutokset heijastuvat suoraan täysnäkömään ilman sivun uudelleenlataamista.

The screenshot shows a web browser window with the URL <http://fi01733/FullView/OrderLineDetails?id=1>. The page title is "Orderline details (#1, Manual, Binder, English)".

**Related products**

Product	Reference type
Not a real product #1	W50
Not a real product #2	W50

**Events**

Description	Date
Estimated delivery	1.3.2013
Actual delivery	15.3.2013
es	
Estimated delivery	
Requested delivery	
Test run date	w.wartsila.com

**Comments**

No comment here.

[Save changes](#)

Build 20130311.1645  
[Report bug / Request feature](#)  
 © 2013 Wartsila

**Kuva 11:** Yksityiskohtanäkymä tilausrivistä.



### 4.3.6 SAP -tietokanta

Wärtsilä pitää tiedot installaatioista SAP-tietokannassa. Syystä tai toisesta, Vaasassa on Oraclella toimiva tietokanta, joka peilaa osan SAP-kannan tiedoista. Palvelimelta löytyy tiedot kaikista Wärtsilän installaatioista ja niihin myydyistä tuotteista. Tiedot saadaan tuotua Levdociin kätevästi käyttäen Oraclen .NET natiivia ajuria.

## 4.4 Parempi toteutus

Edellisessä käyttöliittymän toteutuksessa on omat heikkoutensa.

Valtaosa toteutuksen tiedonkäsittelyoperaatioista tapahtuu palvelimella, koska MVC on selkeästi sellaiseen tarkoitettu. Käyttäjän syöttämät tiedot taas sijaitsevat selaimessa, upotettuna formin erinäisiin input-kenttiin. Jotta tietoa voitaisiin käsitellä, täytyy formi lähettää palvelimelle, ellei tahdo sekoittaa mukaan suurta määrää Javascript -koodia. Halusin kuitenkin, että käyttäjän on mahdollista tehdä kaikki haluamansa muutokset ensin ja päättää vasta lopuksi haluaako hän tallentaa vai perua tehdyt muutokset.

Eräs ongelma oli myös yksityiskohtanäkymä, jonka modelin sisältö haetaan tietokannasta. Esimerkiksi uusien tilausten lisääminen päänäkymässä vaatii siis jälleen välitallennuksen, jotta tiedot tallentuvat tietokantaan ja ne saadaan tämän jälkeen näkyviin yksityiskohtanäkymään.

Päätin luoda prototyypin ohjelmasta, jossa MVC:n merkitys astuisi pienempään rooliin ja ohjelman toiminta tapahtuisi pääasiassa selaimen päässä käyttäen Knockout.js -kirjastoa ja Javascriptiä.

#### 4.4.1 Knockout.js

Knockout on MVVM-arkkitehtuurin (Model-View-ViewModel) toteuttava Javascript-kirjasto, jonka avulla web-sivun DOM-elementit saa sidottua observable-olioihin. Sen käyttö on suhteellisen helppoa ja koodi on helposti luettavaa ja näin ollen helposti ylläpidettävää (**Kuva 12.**) /6/

The screenshot shows the 'learn.knockoutjs.com' tutorial page. At the top, it says 'Tutorial: Introduction' and has 'help' and 'main site' links. The main content is divided into three sections:

- Step 1 of 5:** A 'Welcome!' message explaining that the tutorial covers the basics of building a web UI with the Model-View-ViewModel (MVVM) pattern using Knockout.js. It mentions that users will learn to define a UI's appearance using 'views'.
- Output:** A section with a 'Run (Ctrl+Enter)' button. Below it, there is a form with two input fields: 'First name: Olli' and 'Last name: Asikainen'. Below the form, the text reads: 'Your first name is Olli' and 'Your last name is Asikainen'.
- Code Editors:**
  - View (HTML):** Contains HTML markup for the form and text:
 

```
<!-- This is a *view* -->
<!-- HTML markup that defines the appearance of your view -->
<p>First name: <input data-bind="value: firstName"></input ></p>
<p>Last name: <input data-bind="value: lastName"></input ></p>
<p>Your first name is <span data-bind="text: firstName"></span></p>
<p>Your last name is <span data-bind="text: lastName"></span></p>
```
  - View Model (JavaScript):** Contains JavaScript code for the ViewModel:
 

```
// This is a simple *viewmodel*
// JavaScript that defines the data and behavior of your viewmodel
function AppViewModel() {
  this.firstName = ko.observable("Bert");
  this.lastName = ko.observable("Bertington");
}

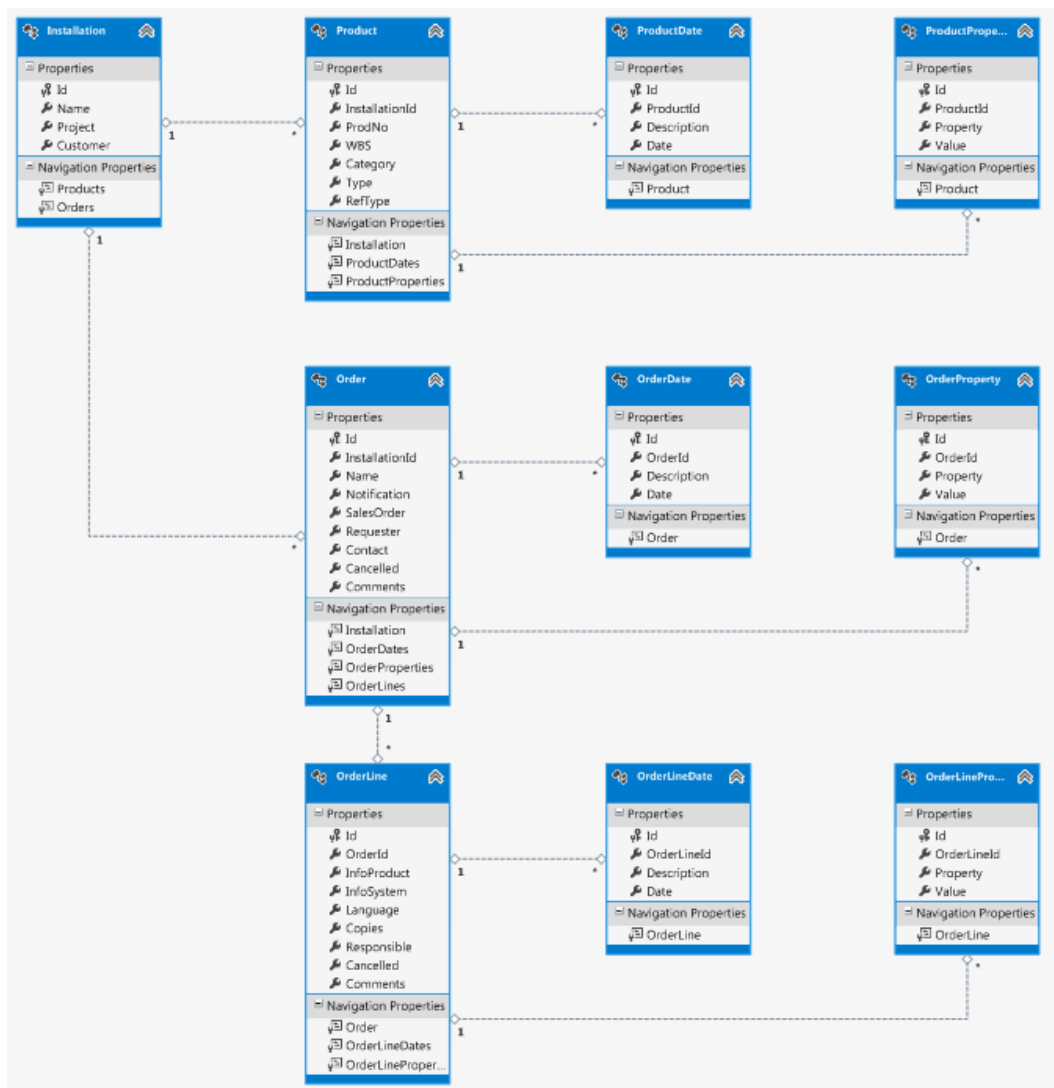
// Activates knockout.js
ko.applyBindings(new AppViewModel());
```

**Kuva 12:** Knockoutin kotisivulla on interaktiivinen tutoriaali kirjaston opetteluun.

#### 4.4.2 Model first

Tällä kertaa loin tietomallin ”model first” –lähestymistavalla käyttäen ADO.NET Entity Data Model Designeria (**Kuva 13.**). Käyttötarkoituksessa, jossa tietokanta on tyhjä, osoittautui ”model first” joustavammaksi kuin ”database first”, jossa automaattisesti luotua modelia tuli useaan otteeseen tarve muuttaa.

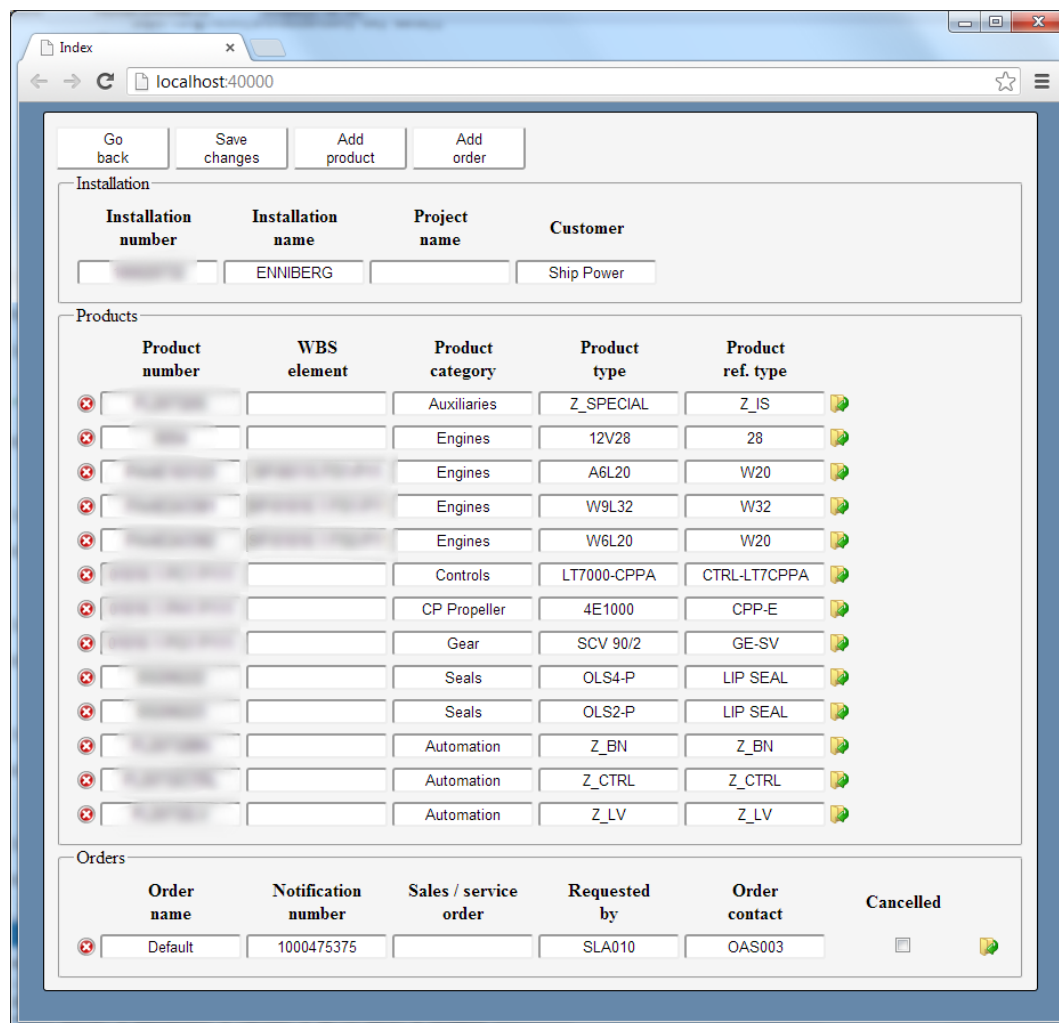
Koska kontrolleri muuntaa modeleita JSON serialiserilla, joka menee olion sisään ja käy sen läpi rekursiivisesti, täytyy kaikkiin ”takaisin päin” viittaaviin kenttiin lisätä ScriptIgnore-attribuutti, joka estää ikuisen silmukan muodostumisen serialiserissa. Tämän toteuttaa kätevästi laajentamalla modelin luokkia toisessa tiedostossa, jonka sisältöön designer ei kajoa. Samoihin modelin laajennosluokkiin lisäsin \_destroy totuusarvon, joka Knockout asettaa todeksi, kun elementti poistetaan taulukosta käyttäen observablearray.destroy-metodia.



*Kuva 13: Tietomalli ADO.NET Entity Data Model Designerissa.*

#### 4.4.3 Yhden sivun ohjelma

Knockoutin avulla ohjelman pystyy toteuttamaan ns. ”yhdeällä webbisivulla” (Kuva 14.), kun AJAXin avulla haetut tiedot voi sijoittaa ViewModeliin ja niihin sidotut DOM-elementit saavat välittömästi uudet arvot. Sidonta toimii myös toisin päin, eli DOM-elementistä observable-olioon, jolloin tiedon tallennuksen pystyy myös hoitamaan kätevästi lähettämällä tiedot JSON-muodossa AJAXilla, eikä tarvitse käyttää formeja ollenkaan ja sivun uudelleenlatausta ei tarvitse silloinkaan. Käyttöliittymä toimii äärimmäisen sulavasti.



*Kuva 14: Täysinäkymä prototyypissä.*

Näkymän vaihto tapahtuu käyttäen jQueryn `hide-/show` -metodeita, jotka suomentaen piilottavat sekä tuovat näkyviin DOM-elementtejä. Esimerkiksi tuotteen oikealla puolella olevaa kansiota painamalla ViewModeliin asetetaan rivin tuote ja näytetään div-elementti, joka Knockoutin `with-sidonnalla` renderöi ruudulle tuotteen tiedot. Latausaikoja näkymien vaihtojen välissä ei käytännössä ole lainkaan, koska tarvittava tieto löytyy ViewModelista eikä sivupyyntöjä tarvitse tehdä.

## 5 LOPPUPÄÄTELMÄT

Työ tarjosi kattavan katsauksen nykyaikaisen web-sovelluksen toteuttamiseen. MVC:llä tehty ensimmäinen toteutus antoi esimakua kehittämisen helppoudesta, mutta osoittautui lopulta mielestäni liian jäykäksi, koska se kannusti vanhanaikaiseen post form + refresh –kaavaan, joka pakottaa välitalennuksiin. Knockoutin avulla onnistuin toteuttamaan skaalautuvan ohjelmiston joka tarjoaa sulavan ”Web 2.0” -kokemuksen ilman ylimääräisiä sivun latauksia.

## LÄHTEET

/1/ Wärtsilä lyhyesti. Viitattu 7.5.2013. <http://www.wartsila.com/fi/about/yhtio-johto/yhtiorakenne>

/2/ Wärtsilä Suomessa. Viitattu 7.5.2013. [http://wartsila.fi/fi\\_FI/wartsila/wartsila-suomessa](http://wartsila.fi/fi_FI/wartsila/wartsila-suomessa)

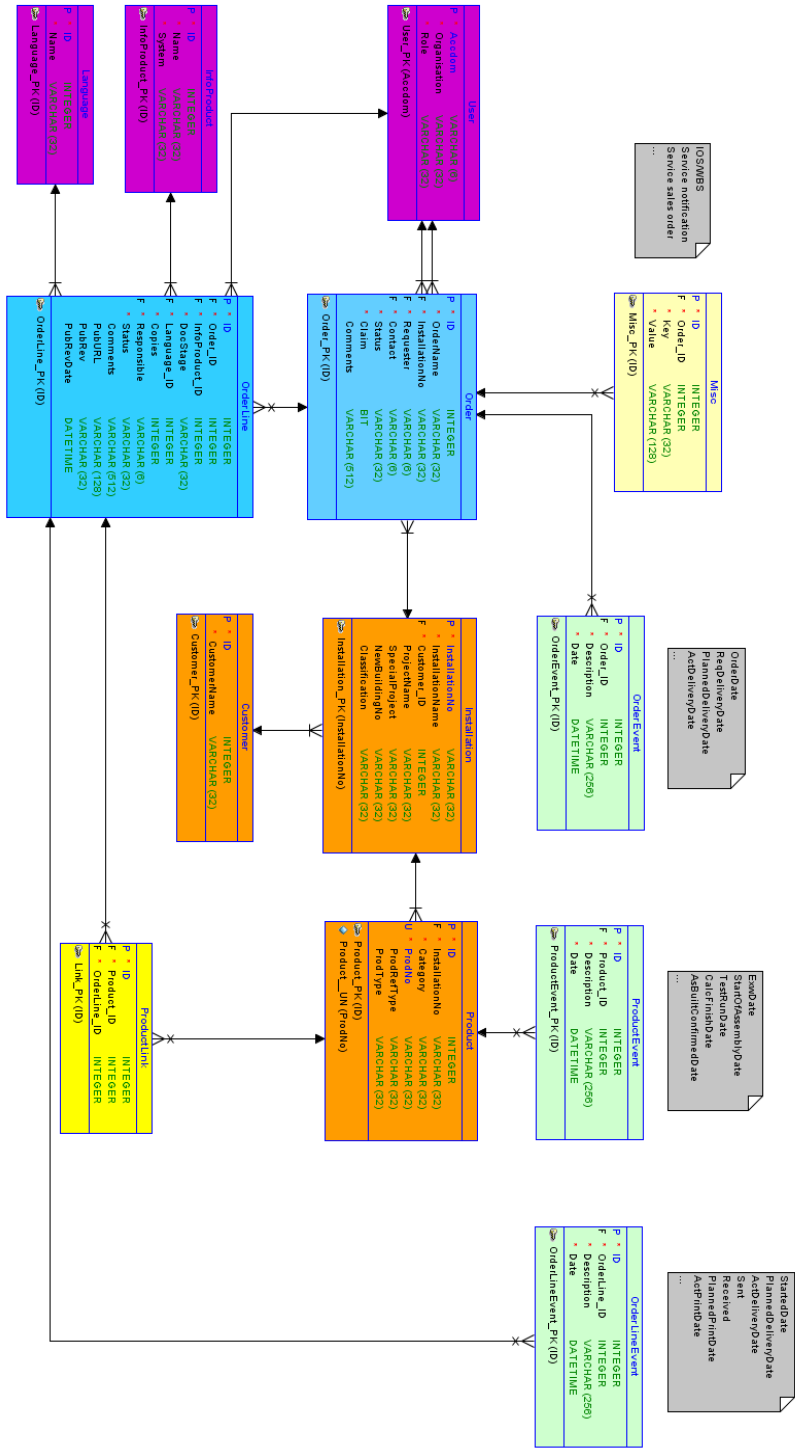
/3/ Common Language Runtime, J Katajisto, 2005, <http://www.cs.helsinki.fi/u/pohjalai/k05/okk/seminar/Katajisto-CLR.pdf>

/4/ ASP.NET MVC Overview. Viitattu 22.5.2013. <http://www.asp.net/mvc/tutorials/older-versions/overview/asp-net-mvc-overview>

/5/ What is NuGet. Viitattu 23.5.2013. <http://nuget.codeplex.com/>

/6/ Knockout Introduction. Viitattu 4.6.2013. <http://knockoutjs.com/documentation/introduction.html>

TIETOMALLI



# KAAVAILTU KÄYTTÖLIITTYMÄ

Internet Explorer

Installation Information

Installation	Project name	Customer	Special
100007132	STX 1316	Ship power	Standard

Products

Product	Category	Type	Class	Reference type
PA-ME216486	Engine	BLDOP	W50	W50
PA-ME216487	Engine	BLDOP	W50	W50
PA-ME216488	Engine	BLDOP	W50	W50
PA-AJ100029	Gear	WAS		

Order Information

Order	Claim	Requester	Contact	Date	Status
(178)	"Engine doc"	Tom Rönö	Thomas Salmén		Done
(18)	"Gear docs and safe"				Done
(18)	"Claim for engine manual"				Pending

Orderlines (only showing orderlines of selected orders)

Order	Type	Serial	Language	Stage	Copies	Organization	Responsible	Estimate	Actual	Status
Engine doc	Manual		English	Final	6	Chc	Nico Salmén			Done
Engine doc	Manual		English	Final	1	Chc	Nico Salmén			Done
Engine doc	CD		English	Final	1	WPN	Jana Salmén			Done
Engine doc	Speaker		English	Final	6	WPN	Jana Salmén			Done
Engine doc	SPC		English	Final	1	WPN	Jana Salmén			Done
Engine doc	CD		English	Final	1	WPN	Jana Salmén			Done
Engine doc	ELDOC3		English	Final	1	WPN	Jana Salmén			Done
Engine doc	Record Book		English	Final	4	WPN	Jana Salmén			Done
Engine doc	Speaker		English	Final	1	WNO				Done
Engine doc and safe	Manual		English	Final	1	WNO				Done
Engine doc and safe	Speaker		English	Final	1	WNO				Done
Engine doc and safe	SPC		English	Final	1	WNO				Done

Comments

Everything good

Timeline (only showing the events of selected orders)

Date	Engine docs	Manual	SPC	SPC	ELDOC3	Record Book	Manual	SPC
?								
?								
?								
?								
?								
?								
?								
?								
?								

Internet Explorer

PA-ME216487

Date	Description
?	Est. date
?	Assembly started
?	Test run
?	As built confirmed

Internet Explorer

"Engine docs"

Extra information

Key	Value
ICD/VMS	SPN0267 / RSJ 4215
Notification	100007056
Sales order	10087622

Events

Date	Description
?	Other date
?	Requested delivery date
?	Delivery to customer

Internet Explorer

"Engine doc", "SPC", Binder, English, & WPN

Related products

Product	Reference type	Label
PA-ME216486	W50	(178)
PA-ME216487	W50	(178)
PA-ME216488	W50	(178)

Events

Date	Description
?	Printed final date
?	Actual finish date
?	Sent to printing
?	Received from printing
?	Delivered to customer

Comments

Had some problems

Report production date, if you wish. Current Lander has these

Add reference link to existing system

Report data from these events are provided, but deliver events with custom description

Note the order number of the "10007132" value

Not fully designed yet, have to see what is technically feasible.



# KÄYTTÖTAPAAUSKAAVIO

