FAK EKI
Datavetenskap

Magnus Dahlén

Windows 8-surfplatta: gränssnitt till ordersystem

Windows 8 tablet: Interface for order system

Examensarbete 15 hp
Dataingenjör

Datum/Termin: 2012-06-05
Handledare: Donald F. Ross
Examinator: Thijs J Holleboom
Ev. löpnummer: C2012:04

Karlstads universitet 651 88 Karlstad
Tfn 054-700 10 00 Fax 054-700 14 60
Information@kau.se www.kau.se

Windows 8-surfplatta: gränssnitt till ordersystem

Magnus Dahlén

This report is submitted in partial fulfillment of the requirements for the Bachelor's degree in Computer Science. All material in this report which is not my own work has been identified and no material is included for which a degree has previously been conferred.

Magnus Dahlén

Approved, 2012-06-05

Advisor: Donald F. Ross

Examiner: Thijs J Holleboom

Abstract

With the introduction of Apple's iPad, tablets became more popular among consumers. Since then a number of manufacturers have released tablets of their own, with the operating system primarily being Google's Android. Microsoft is now getting into the tablet OS market with the soon to be released Windows 8.

As the tablets are starting to make their way into the corporate world, IT consulting firm Sogeti in Karlstad want to explore the possibilities given by a Windows 8 based tablet. Sogeti is currently working with a customer to integrate a number of separate order handling systems into a central solution. The idea to develop a client for a Windows 8 tablet to manage the system surfaced.

During this project a prototype client for a tablet running Windows 8 has been developed with the purpose of realizing this idea. The resulting prototype lets the user manage orders and view customer data from a touch friendly interface running on the tablet. This essay describes the work leading up to the final prototype.

Sammanfattning

Genom introduktionen av Apples iPad skapades ett ökat intresse hos allmänheten för s.k. surfplattor. Sedan dess har ett antal olika tillverkare släppt egna plattor som framförallt baserats på Googles operativsystem Android. Microsoft tar nu steget in på surfplattemarknaden med sitt kommande operativsystem Windows 8.

Eftersom surfplattor nu börjar leta sig in i företagsvärlden önskar IT-konsultbolaget Sogeti i Karlstad utreda vilka möjligheter som finns med surfplattor. Sogeti arbetar för närvarande tillsammans med en kund på ett system som skall centralisera ett antal orderhanterings-system i en central lösning. Idén att skapa en klient för en Windows 8 surfplatta för att hantera detta system dök då upp.

Under detta projekt har en prototyp-klient för en surfplatta som kör Windows 8 utvecklats med syfte att realisera denna idé. Den slutgiltiga prototypen låter användaren hantera ordrar och titta på kunddata från ett pekskärmsvänligt gränssnitt som körs på plattan. Denna uppsats beskriver arbetet som resulterat i implementationen av prototypen.

Innehållsförteckning

Figurer

1 Inledning

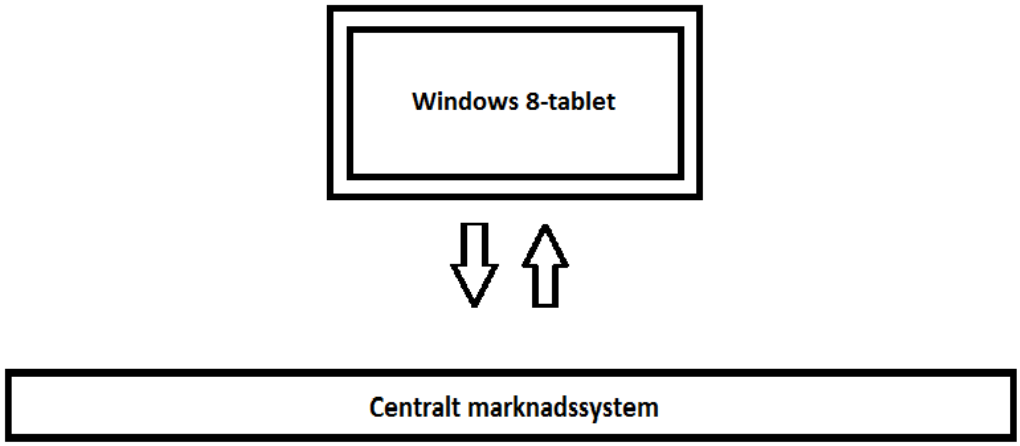
1.1 Syfte

Sogeti[1] är ett konsultföretag inom IT-branchen som för närvarande utvecklar ett marknadssystem till en stor svensk pappers-koncern. Detta system används för att registrera kunder och ordrar vid koncernens olika bruk.

För att skaffa kunskap om hur surfplattor skulle kunna användas i kundföretagets verksamhet önskar Sogeti utreda för- och nackdelarna med dessa. Eftersom Sogeti Karlstad har en stark inriktning mot Microsofts tekniker vill man att en PoC (Proof of Concept) till en Windows 8-surfplatta skall utvecklas för detta ändamål.

Windows 8 är det senaste operativsystemet från Microsoft som är planerat att släppas under 2012. Detta operativsystem kommer att kunna köras både på traditionella datorer och på så kallade surfplattor.

Den PoC som skall utvecklas kommer att kommunicera med det marknadssystem som Sogeti utvecklar åt kunden.



Figur 1-1 – Förenklad bild över systemet

Syftet med denna PoC är att ge kundföretaget en bild över hur en surfplatta skulle kunna användas i deras verksamhet.

I denna uppsats kommer projektet att ta fram denna PoC som en del av ett examensarbete i datavetenskap vid Karlstads universitet att presenteras.

1.2 Disposition

Detta dokument är uppdelat enligt denna disposition.

Kapitel 2 - Bakgrund

I detta kapitel beskrivs bakgrunden till det projekt som behandlas i denna uppsats. En introduktion till det befintliga systemet ges och bakgrund som är nödvändig för att förstå de val som gjorts senare i projektet presenteras.

Kapitel 3 - Projektbeskrivning

I detta kapitel beskrivs implementationen av projektet. En översiktsbild över systemet ges. Användargränssnittet samt den tekniska implementationen presenteras.

Kapitel 4 - Resultat och utvärdering

I detta kapitel beskrivs resultatet av projektet och en utvärdering av de olika val som gjorts under projektet ges.

Kapitel 5 - Slutsats

Detta avsnitt syftar till att utvärdera projektet som helhet. Avsnittet är tänkt att ge en bild över de erfarenheter som skaffats under projektets gång. Några idéer för framtida utveckling av produkten kommer också att ges.

2 Bakgrund

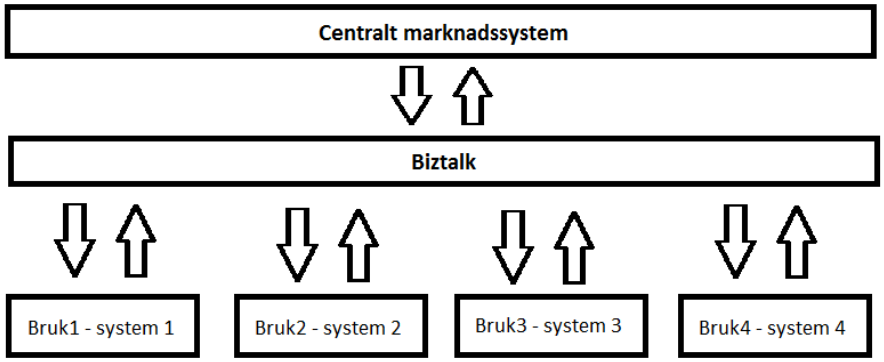
I detta kapitel beskrivs hur det marknadssystem som Sogeti utvecklar är uppbyggt samt vad dess användningsområde är. En bild [Figur 2-1] över varför systemet implementeras presenteras. Utöver detta beskrivs projektet som hanteras i denna uppsats.

2.1 Introduktion

Sogeti [1] är ett konsultföretag inom IT-branchen som för närvarande utvecklar ett marknadssystem till en stor svensk pappers-koncern. Detta system används för att registrera kunder och ordrar vid koncernens olika bruk.

Bakgrunden till detta system är att koncernen för en tid sedan köpt upp ett antal bruk i utlandet. Varje bruk har haft olika sätt att hantera kunder och ordrar på och har använt sig av olika system för detta ändamål. I samband med att de blivit en del av den större koncernen finns ett behov av att centralisera orderhanteringen. Ett centraliserat system innebär att det blir enklare att få en översikt över ordrar och kunder samt gör det lättare att distribuera ordrarna till lämpliga bruk.

Mot denna bakgrund är nu ett system under utveckling som skall göra det enkelt att hantera ordrar och kunder från ett centralt system. För att de olika systemen enkelt ska kunna kommunicera med varandra används integrationslösningen Biztalk [2]. Med hjälp av Biztalk kan de nuvarande systemen integreras i en central lösning.



Figur 2-1 – Med hjälp av Biztalk integreras olika system i en central lösning

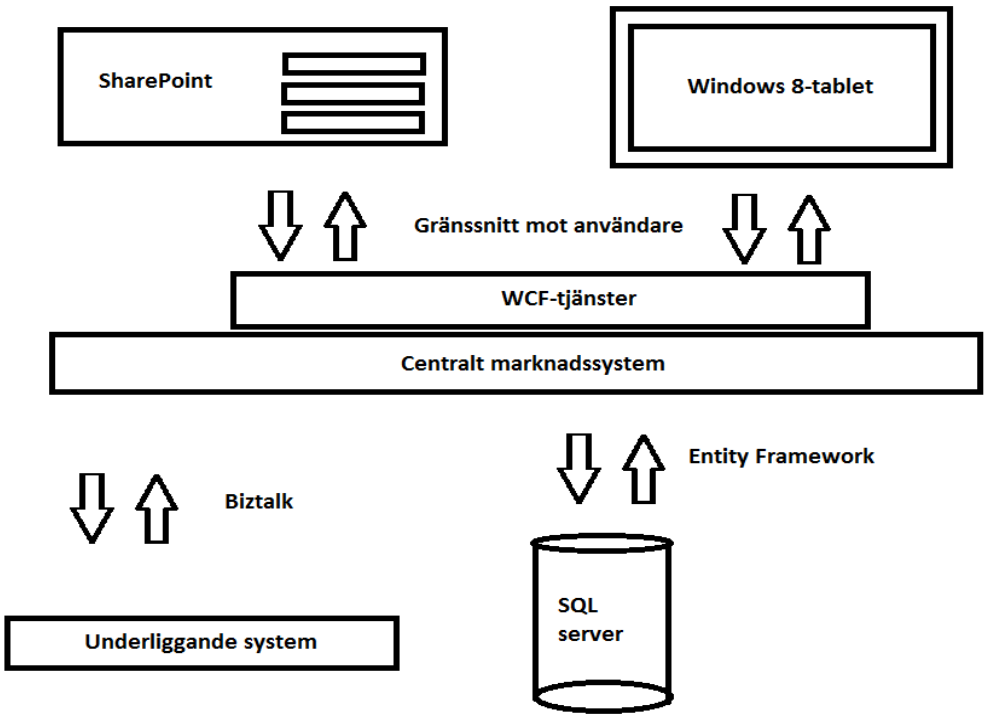
2.2 Systemet

Det centrala marknadssystem som är under utveckling kommunicerar med de underliggande systemen genom Biztalk. Detta gör att man kan byta ut några av systemen i framtiden utan att alla andra system måste byggas om.

Det centrala systemet har som uppgift att organisera alla ordrar och kunder på de olika bruken samt att ge möjlighet för användarna att lägga till eller redigera ordrar samt kunder. Det data som behöver sparas lagras i en databas som är skapad i Microsoft SQL-server [3]. Kommunikation med databasen sker med hjälp av Entity Framework [4].

Kommunikation mellan de olika systemen är viktigt men för att de skall komma till nytta krävs också ett gränssnitt ut mot användaren. Genom att det centrala systemet exponerar ett antal WCF-tjänster [5] ges möjlighet för klienter att ansluta till det.

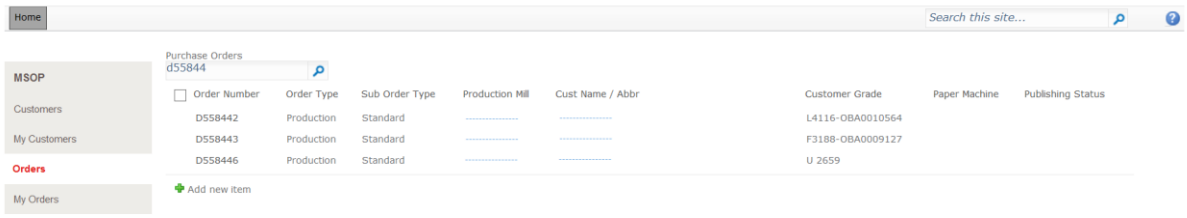
För närvarande existerar en klient som är utvecklad i SharePoint [6]. Genom denna klient kan man hantera ordrar och kunder. Detta projekt går ut på att skapa ytterligare en klient. Denna klient skall köras på en Windows 8-surfplatta [7] och använda sig av befintliga WCF-tjänster för kommunikation med det centrala systemet.



Figur 2-2 – Det centrala marknadssystemet lagrar information i en SQL-server databas och gör det möjligt att ansluta klienter med hjälp av WCF-tjänster.

2.3 Existerande klient

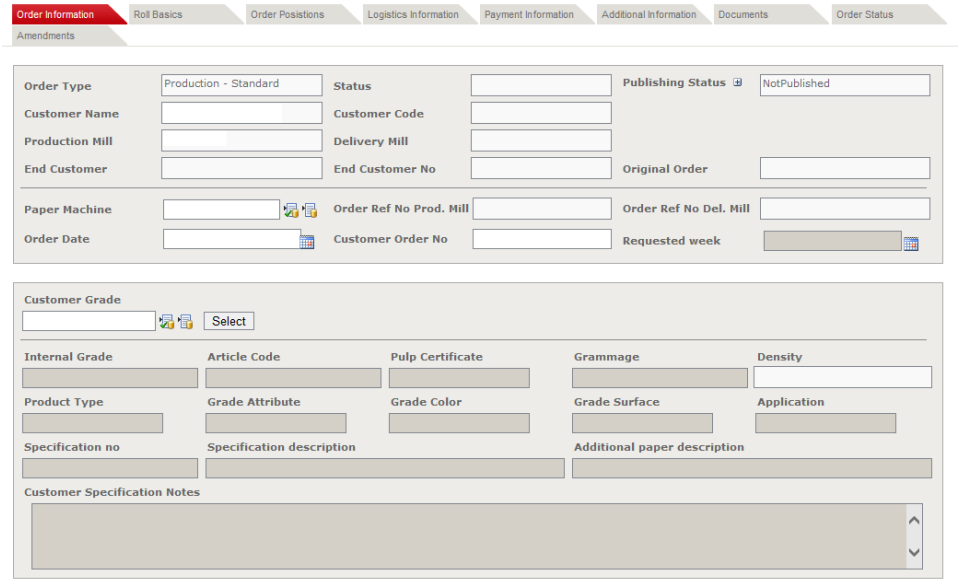
För att hantera ordrar och kunder i det centrala systemet har Sogeti [1] utvecklat en klient i SharePoint [6]. Klienten är web-baserad och man kan med hjälp av den hantera både kunder och ordrar i systemet. I Figur 2-3 visas gränssnittet för detta system, då användaren valt att lista ordrar.



Figur 2-3 – Gränssnittet för det befintliga systemet som visar listade ordrar

Genom menyn till vänster väljer användaren vilken information som skall visas. Användaren ges möjlighet att visa ordrar och kunder. Genom att välja ”My Customers” eller ”My Orders” begränsas listningen till att enbart visa användarens egna ordrar och kunder.

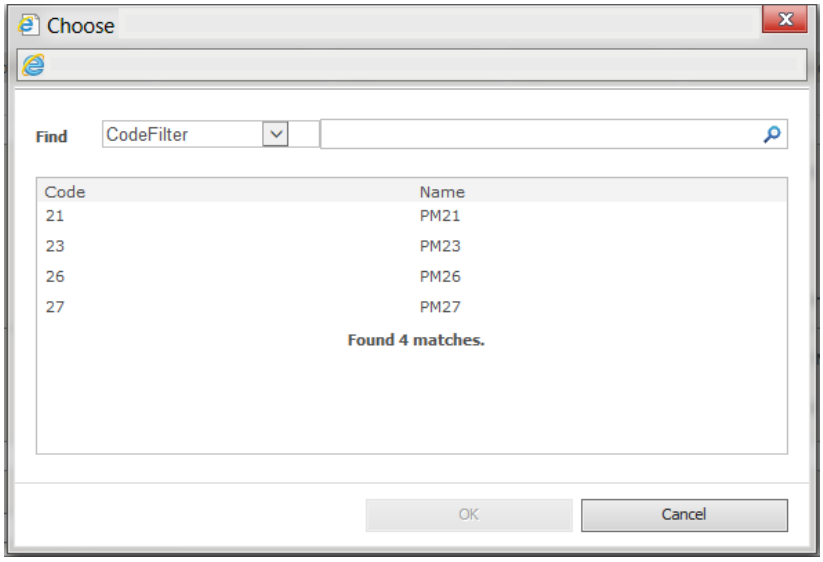
Om användaren vill visa eller redigera en specifik order kan han välja att klicka på ordernumret och sedan genom en rullista välja att antingen visa eller redigera ordern. I Figur 2-4 visas gränssnittet då användaren valt att visa en order. Motsvarande gränssnitt existerar även för redigering av kunder.



Figur 2-4 – Informationen om en order kan redigeras i gränssnittet

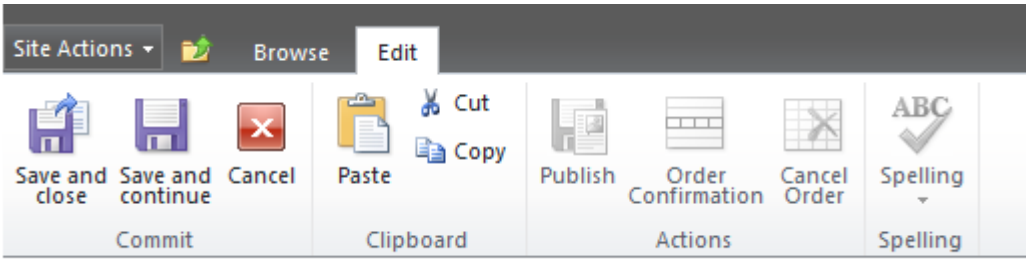
Vissa fält i Figur 2-4 är låsta och kan inte ändras av användaren. Andra fält är antingen textfält där användaren kan fylla i text eller anpassade textfält där användaren kan välja

mellan fördefinierade val eller datum. Om användaren vill välja mellan fördefinierade val dyker en ny ruta upp med en lista över valen. Användaren ges här också möjlighet att söka i listan.



Figur 2-5 – Ruta där användaren kan välja ett fördefinierat alternativ

När användaren ändrat en order eller en kund kan han välja att spara ändringarna i systemet genom en meny som visas i gränssnittets övre del. Denna meny visas i Figur 2-6.



Figur 2-6 – Användaren kan spara sina inställningar när en ändring gjorts

Utöver att spara ges användaren med hjälp av menyn i Figur 2-6 utföra några enkla redigeringsoperationer.

Det befintliga gränssnittet för att redigera ordrar och kunder kommer att ligga till grund för gränssnittet i det projekt som detta dokument behandlar. Gränssnittet kommer dock att anpassas för att vara enkelt att använda på en surfplatta och för att följa designspåret Metro som beskrivs närmare i 2.4.5.

2.4 Projektet

I denna sektion beskrivs bakgrunden till projektet.

2.4.1 Introduktion

För närvarande existerar en klient som är utvecklad i SharePoint för att administrera det centrala marknadssystemet som Sogeti Karlstad utvvecklar. För att skaffa kunskap om hur surfplattor skulle kunna effektivesera säljarnas arbete och öka deras mobilitet önskar Sogeti utreda möjligheterna att använda sig av surfplattor. Projektet går ut på att utveckla en enkel PoC (Proof of Concept), vilken syftar till att ge kunden en bild av surfplattornas möjligheter och begränsningar i deras verksamhet.

2.4.2 Applikationen

Slutprodukten från projektet skall vara en applikation för en Windows 8-platta. Syftet med denna är att visa hur man kan använda surfplattor i sitt arbete. Applikationen skall innehålla funktioner för att hantera ordrar och om tid finns skall även funktioner för att hantera kunder läggas till.

Genom att utveckla applikationen till en Windows 8-platta kan man använda redan befintlig kunskap inom C#.NET [8] och andra Microsoft-tekniker, eftersom mycket av utvecklingsmiljön och de tekniska lösningarna är lika de som används vid utveckling av vanliga Windows-applikationer. Det gör också att det blir lättare att integrera med redan existerande lösningar, samt att bygga framtida lösningar på plattformen eftersom mycket kompetens redan finns inom området. Applikationen kommer användas som ett alternativt gränssnitt till den existerande lösningen, vid sidan av SharePoint-lösningen.

2.4.3 Befintliga tjänster

I den befintliga lösningen existerar det ett antal WCF-tjänster för att komma åt och modifiera data i det centrala systemet. Applikationen som skall utvecklas kommer att använda sig av dessa tjänster för att administrera ordrar/kunder. Mer information om hur tjänsterna används ges i avsnitt 3.3.2.1. Genom att använda de befintliga tjänsterna skapas inget extra beroende till den nya klienten och påverkan på det existerande systemet blir minimal.

2.4.4 Användargränssnittet

Eftersom den nya applikationen skall utvecklas till en Windows 8-platta som använder ett touch-gränssnitt för interaktion med användaren och saknar ett fysiskt tangentbord så kommer användargränssnittet att skilja sig från hur det ser ut idag. När man utvecklar applikationer för ett system som baseras på en liten tryckkänslig skärm måste man tänka på att bildytan är begränsad och att knappar och textfält måste göras större för att användaren enkelt skall kunna nyttja dem.

2.4.5 Designspåret Metro

Metro [9] är namnet på det designspår som enligt Microsoft är det spår man ska följa när man utvecklar applikationer för Windows Phone och nu också Windows 8.

Metro bygger på att innehållet skall stå i fokus och att användargränssnittet skall presentera innehållet på ett tydligt sätt. Inspirationen kommer från tunnelbanan, där skyltarna har enkla symboler och visar tydligt vart man finner olika faciliteter. På samma sätt skall en applikation direkt visa för användaren hur man når ytterligare innehåll och navigeringen skall hållas enkel, komplexa ikoner och bakgrunder bör undvikas och typografi får en viktig roll i designen. I projektet skall användargränssnittet hållas enkelt och fokus skall ligga på innehållet. Figur 2-7 visar ett exempel på en Applikation från Windows 8 som följer designspåret Metro, användargränssnittet hålls enkelt och innehållet står i fokus.

Figur 2-7 – En exempel-applikation från Windows 8 där innehållet står i fokus

2.5 Verktyg och tekniker

Projektet kommer baseras på tekniker och verktyg från Microsoft. Vid ”vanlig” Windows-utveckling används främst Visual Studio som verktyg och användargränssnitt utvecklas vanligen i Windows Forms [10] eller WPF (Windows Presentation Foundation [11]).

För utveckling till Windows 8 har Microsoft tagit fram en ny programmeringsmodell som man valt att kalla WinRT (Windows Runtime [12]). Det är denna programmeringsmodell som kommer att användas under projektets gång.

Utveckling kommer att ske i Visual Studio 11 Developer Preview [13] som är den senaste versionen av Visual Studio som finns tillgänglig i nuläget.

2.5.1 Windows 8

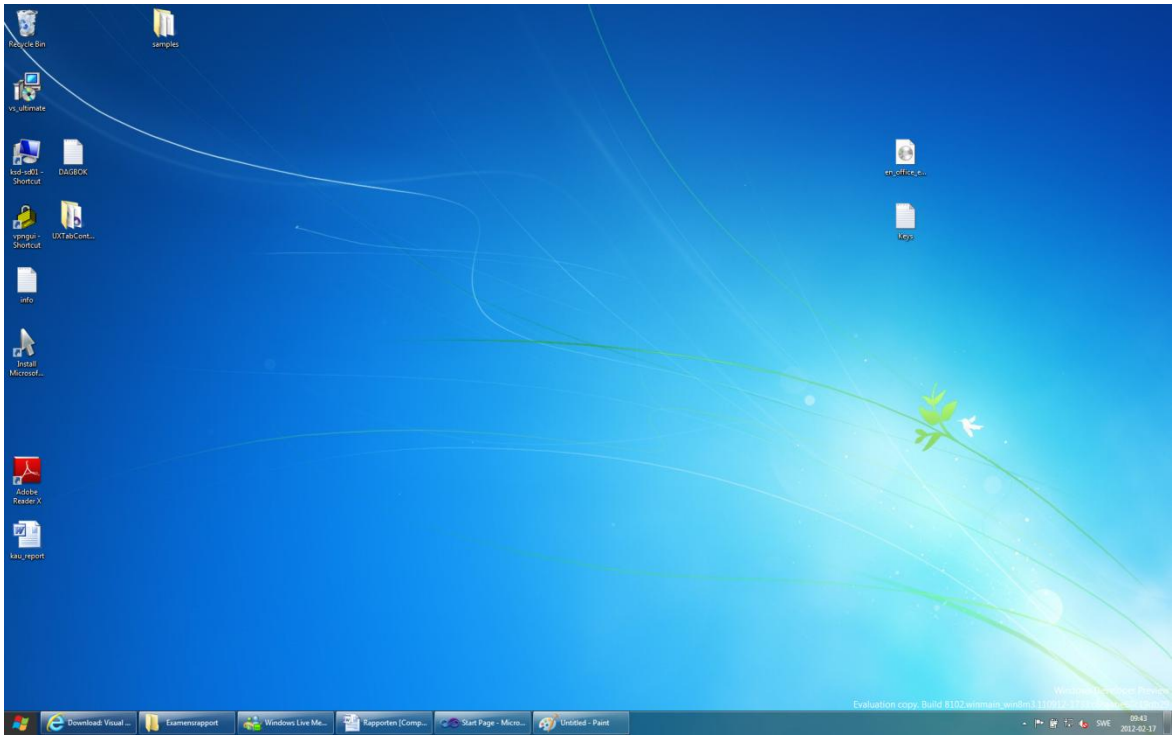
Windows 8 [7] är den nästkommande versionen av Microsoft Windows och med den utökar Microsoft Windows-plattformen till bärbara surfplattor, samtidigt som stödet för traditionella datorer bibehålls. Windows 8 kommer att kunna köras både på x86- och ARM-baserade processorer. För närvarande finns en förhandsversion för utvecklare tillgänglig och det är denna version som kommer att användas i projektet.

I Windows 8 vill man förändra det sätt man skapar och använder applikationer på. I avsnitt 2.4.5 beskrevs kort designspåret Metro. Microsoft vill att man så långt det är möjligt skall följa de riktlinjer som finns i Metro för att skapa applikationer som ger operativsystemet ett enhetligt utseende och är enkla för användaren att bruka. Med Windows 8 har Microsoft försökt fasa ut det tradtitionella skrivbordet och istället skapa ett helt nytt användargränssnitt som inspirerats av Windows Phone.

Figur 2-8 – Startmenyn har ersatts med ”tiles” i Windows 8.

Användargränssnittet bygger på s.k. Tiles, där varje applikation får en ruta som används för att starta programmet istället för de traditionella ikonerna. Dessa rutor är organiserade i ett rutnät som kan scrollas igenom i sidled. Rutorna kan sedan ändra utseende beroende på vilket tillstånd programmet befinner sig i. Om man t.ex. får e-post kan e-post-programmet anpassa sin Tile för att uppmärksamma användaren på detta.

I den version av Windows 8 som används i skrivande stund finns det möjlighet att använda sig av det traditionella utseendet från Windows 7 och många av de medföljande programmen är anpassade efter detta gränssnitt. Ytterligare information om det nya gränssnittet ges i avsnitt 2.8.



Figur 2-9 – Windows 7 gränssnitt kan användas i Windows 8.

2.5.2 Visual Studio 11 Developer Preview

Visual Studio 11 Developer Preview [13] är i skrivande stund den senaste versionen av utvecklingsverktyget Visual Studio från Microsoft. Visual studio är en så kallad IDE [25] (Integrated Development Environment). I Visual Studio finns många av de verktyg man behöver för att utveckla applikationer i Windows-miljön. Språk som stöds är bland annat Visual Basic, Visual C++, Visual F#, Javascript och Visual C#. Utöver att Visual studio innehåller en kod-editor och kompilatorer för ett antal olika språk så finns också verktyg för att analysera kod, skapa tester och inte minst möjlighet att enkelt designa och förhandsgranska ett grafiskt användargränssnitt.

Nytt i Visual Studio 11 Developer Preview är att stöd för utveckling av Metro-baserade applikationer för Windows 8 med hjälp av Windows Runtime finns integrerat i programmet. Windows Runtime kommer att beskrivas i avsnitt 2.5.3.

2.5.3 Windows Runtime

Windows Runtime [12] (WinRT) är en ny programmeringsmodell från Microsoft som är anpassad för att bygga Metro-anpassade applikationer för Windows-miljön. Utveckling av gränssnitt (och till viss del även andra delar av applikationen) i WinRT sker med hjälp av XAML[14] (Extensible Application Markup Language) eller med hjälp av HTML och Javascript. I detta avsnitt kommer jag beskriva hur man kan bygga applikationer med hjälp av XAML i WinRT.

2.5.3.1 XAML och designverktyget

XAML [14] härstammar från XML och låter utvecklaren bygga upp en applikation med taggar. Varje kontroll eller grafik i applikationen representeras av en tag. I Figur 2-10 visas XAML från en applikation. I Figur 2-11 visas resultatet i användargränssnittet.

```
<Button Background="Aqua" Foreground="Black" Margin="228,299,0,433">hejknapp</Button>
```

Figur 2-10 – XAML för att skapa en knapp i WinRT

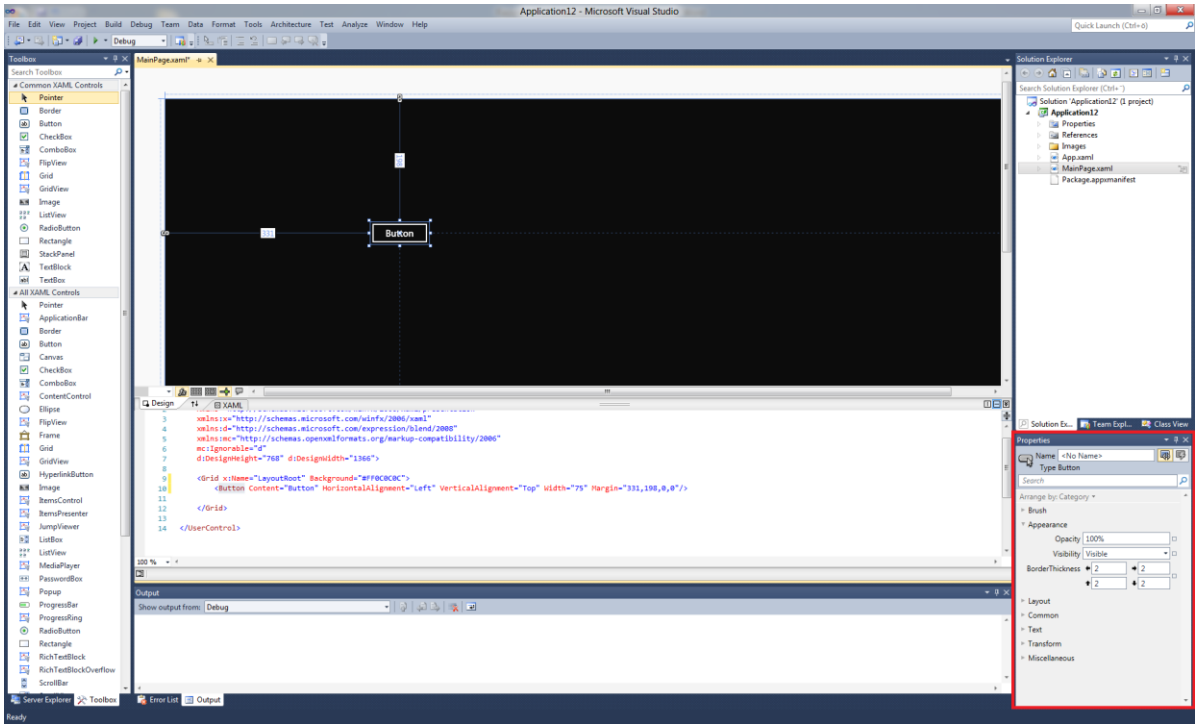
Figur 2-11 – Knappen som visas i gränssnittet när man använder XAML från Figur 2-10

Genom att skapa en tag ”Button” kan man lägga till en knapp i sin applikation. Man kan sedan ange ett antal egenskaper för knappen genom att ange olika attribut i taggen. I Figur 2-10 sattes bakgrundsfärg, förgrundsfärg och position på knappen (Background, Foreground respektive Margin) och i Figur 2-11 ser vi resultatet. Attribut anges på formen <Attributnamn> = ”<Värde>”. Mellan taggarna kan man ange knappens innehåll som i detta fall är ”hejknapp”. Man skulle även kunna använda sig av t.ex. en bild eller ett annat element som innehåll för knappen.

Genom att nästla olika taggar kan man bygga upp gränssnittet. Det finns ett antal medföljande kontroller och om dessa inte skulle räcka till kan användaren också skapa helt nya kontroller, eller kombinera de medföljande kontrollerna för att uppnå önskad funktionalitet.

Genom att använda XAML ges utvecklaren möjlighet att påverka utseendet på applikationen mer direkt än vid utveckling i t.ex. windows forms, där mer komplex kod krävs för att skapa kontroller om man inte direkt använder sig av design-verktyget.

Om man inte vill skriva all XAML själv kan man utnyttja den inbyggda designern i Visual Studio. Där ges där möjlighet att placera ut kontroller via drag and drop. Utöver detta finns också möjlighet att ändra egenskaper på de olika kontrollerna i egenskaps-fönstret. När man ändrar en egenskap eller lägger till en kontroll genereras XAML-kod automatiskt. Figur 2-12 visar designern i Visual Studio med egenskapsfönstret markerat med rött.



Figur 2-12 – Designer-verktyget i Visual Studio Developer Preview 11

2.6 Surfplattan

I avsnitt 2.4.5 gavs en introduktion till designspåret Metro som är det designspår som Microsoft vill att man skall använda sig av vid utveckling av applikationer till Windows-surfplattor. I detta avsnitt kommer några inbyggda funktioner som kan vara bra att känna till när man designar för och använder en Windows-platta behandlas. En kort introduktion till den platta som använts under projektet kommer också att ges.

2.7 Hårdvaran

Den platta som använts under utvecklingen kommer ursprungligen från Microsofts Build-konferens[15] i USA. Plattan är en version av Samsung Series 7 Slate[16]. Denna platta har en 11,6” skärm med en upplösning på 1366x768 pixlar. Processorn är en Intel Core i5 Dual-Core på 1,6GHz och den har 4GB RAM-minne samt en 64GB SSD-hårddisk. Denna specifikation gör att plattan kan anses vara likvärdig med en desktop-PC snarare än en surfplatta. Den relativt stora skärmen gör också att mer information kan rymmas jämfört med en mindre surfplatta (iPad har t.ex. en 9,7” skärm).

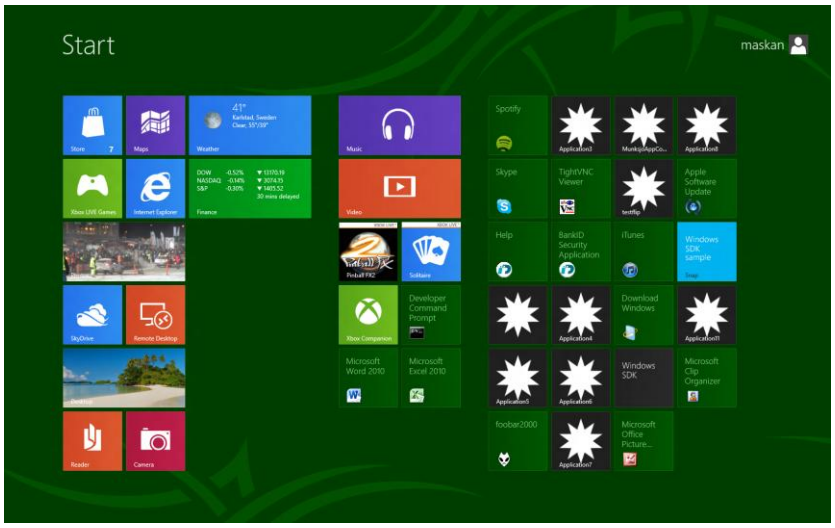
2.8 Gester

I det traditionella användargränssnittet för Windows används i huvudsak startmenyn eller Windows Explorer för att starta nya program. När ett program startats hamnar det i det s.k. aktivitetsfältet längst ner på skärmen.

Figur 2-13 – Aktivitetsfältet i ett traditionellt Windows-gränssnitt.

Användaren kan i tidigare versioner av Windows också välja att använda sig av tangentbordskombinationen Alt+Tab för att skifta mellan de öppna applikationerna. På en surfplatta kan dessa sätt att navigera på ställa till ett antal problem. Till att börja med skulle aktivitetsfältet ta skärmyta i anspråk, något som inte är önskvärt på en mindre skärm. Navigeringen måste också fungera även om inget fysiskt tangentbord finns anslutet till surfplattan vilket gör att olika tangentbordskombinationer är olämpliga att använda som navigeringsmetoder. Skrivbordet har bytts ut för att göra det enklare att starta nya applikationer. För att navigera mellan de öppna applikationerna och utföra ett antal andra

operationer har microsoft infört s.k. gester. I Figur 2-14 visas startskärmen i Windows 8. Här listas de applikationer som användaren har installerat.



Figur 2-14 – Startskärmen i Windows 8

Startskärmen kan navigeras genom att föra fingret i sidled över skärmen för att scrolla i innehållet. Användaren kan också välja att göra ett nyp med fingrarna på skärmen för att komma till en översiktsbild för att snabbare kunna navigera mellan grupper av program som ligger långt ifrån varandra.

Figur 2-15 – Användaren kan nypa med fingrarna för att snabbare navigera mellan olika grupper av program på startskärmen.

Genom att välja en grupp av program kommer användaren att navigeras direkt till den valda gruppen i den inzoomade vyn.

Microsoft har valt att använda kanterna som utgångspunkt för flera av gesterna. Genom att svepa nerifrån och upp med fingret på skärmen kommer man åt ytterligare operationer för det som visas på skärmen. Detta visas i Figur 2-16.

Figur 2-16 – Genom att svepa nerifrån och upp på skärmen kommer användaren åt ytterligare operationer.

För att skifta mellan olika applikationer kan användaren svepa med fingret från vänster kant mot mitten. Användaren kan då dra in ett nytt program på skärmen som ersätter det gamla. Användaren kan också svepa från vänster kant in mot mitten och sedan ut mot kanten igen för att visa en lista över de program som för närvarande går att navigera emellan (motsvara Alt-Tab i det traditionella gränssnittet). Figur 2-17 visar denna del av gränssnittet.

Figur 2-17 – En lista over de program användaren kan navigera emellan visas till vänster.

Ett stort problem på en surfplatta är möjligheten för användaren att använda flera program samtidigt. På grund av den ofta begränsade ytan fungerar det inte att ha många fönster öppna samtidigt då varje applikation skulle få för liten yta för att vara användbart.

Microsoft har valt att låta användaren använda två applikationer samtidigt. För att använda två applikationer samtidigt kan användaren svepa med fingret från vänster kant och sedan hålla kvar fingret till vänster på skärmen för att öppna ytterligare en applikation vid sidan av den första. En av applikationerna hamnar då i s.k. ”snapped view” medans den andra hamnar i

”filled view”[64]. Programmet som är i snapped view kommer alltid att vara 320 pixlar brett medans det andra programmet kommer ta upp resterande del av skärmen.

Figur 2-18 – Bing finance i snapped view och bing weather i fill view.

Användaren kan med hjälp av detta nu ta del av och interagera med båda programmen samtidigt. Det är upp till utvecklaren av programmet att se till att det anpassar sig korrekt för att fungera i de olika tillstånden.

Genom att använda några av standardgesterna i applikationer som utvecklas samt anpassa applikationen för de olika tillstånden kan applikationen använda Windows 8-specifika funktioner och samtidigt skapa en enhetlig upplevelse för användarna.

2.9 Användargränssnitt

2.9.1 Navigering

Som namnet Windows antyder så bygger mycket av användargränssnittet i en traditionell Windows-miljö på fönster. Användaren möts av ett huvudfönster, andra delar av applikationen visas i nya fönster, inställningar för en viss del av programmet visas i ett helt annat fönster osv. Ofta kan flera fönster vara öppna samtidigt. Detta gör att programmet kan kännas som att det består av många olika små delar och kan upplevas som rörigt. I och med Windows 8 försöker man istället fokusera på en mer Hierarkisk modell av navigering.

I en hierarkisk navigeringsmodell möts användaren först av information på en väldigt hög nivå. Genom att välja ett alternativ på den högre nivån kommer användaren till nästa nivå i navigeringen, där mer detaljerad information om den kategori man valt visas. Genom att fortsätta ner i hierarkin kommer mer och mer detaljerad information inom det tidigare valda området att göras tillgänglig för användaren. I Figur 2-19 visas en bild över konceptet

hierarkisk navigering. Varje nivå i hierarkin utmynnar i ytterligare val för användaren vilket gör det möjligt att gå djupare i hierarkin och visa mer information från sub-kategorierna.

Figur 2-19 – Hierarkisk navigering[1]

Vid hierarkisk navigering sker navigeringen ofta med hjälp av en stack. Varje gång användaren väljer att gå neråt i hierarkin hamnar den nya sidan på navigations-stacken. När användaren väljer att gå bakåt tas den senaste sidan bort från navigations-stacken och användaren befinner sig återigen på en högre nivå i hierarkin.

2.9.2 Kontroller

I traditionella gränssnitt finns ofta en uppsjö av kontroller tätt packade tillsammans i verktygsfält eller menyer. Att använda sig av kontroller på detta sätt på en tryckskärms-baserad enhet är dömt att misslyckas. Ett finger är inte ett precisionsinstrument på samma sätt som en mus. Varje kontroll behöver tillräckligt med utrymme för att enkelt kunna träffas av ett finger samtidigt som tillräckligt med utrymme måste lämnas runt ytan för att undvika att användaren kommer åt en kontroll av misstag.

Microsoft rekommenderar att områden som användaren skall trycka på är minst 7mm breda och höga samt att det är 2mm emellan dem[18].

En nackdel med att använda stora kontroller är att färre kontroller och en mindre mängd information ryms på skärmen.

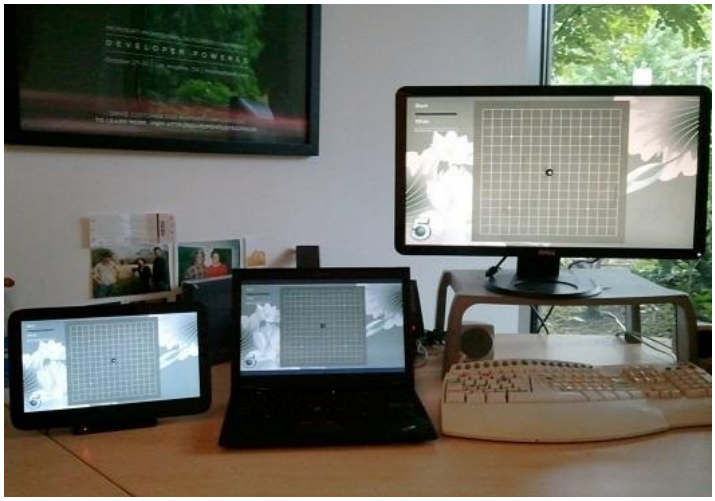
2.9.3 Upplösning och skärmstorlek

Windows 8 är konstruerat för att kunna användas på ett antal olika enheter med olika upplösningar och storlek på skärmen. En större skärm innebär ofta en högre upplösning på skärmen. Detta innebär att mer information ryms på skärmen jämfört med en mindre skärm. När man utvecklar applikationer är det därför viktigt att ta hänsyn till skärmens upplösning och storlek. Det finns ett antal olika sätt att hantera skillnader i upplösning och skärmstorlek.

[1] Bild från http://msdn.microsoft.com/en-us/library/windows/apps/hh761500.aspx

2.9.3.1 Skala efter upplösning

Ett sätt att hantera de olika upplösningarna är att skala applikationen efter upplösning. Detta innebär att alla delar av programmet blir större eller mindre beroende på vilken skärm applikationen används på. Text och bilder förstoras för att täcka hela skärmytan.



Figur 2-20 – Applikationen anpassar sin storlek efter upplösning[2]

En förhöjd upplösning innebär att om elementen behåller samma storlek i pixlar så kommer de att bli mindre rent fysiskt. De skalas därför upp för att få samma fysiska storlek på skärmen. När man använder skalning för att anpassa utseendet på applikationen till en ny upplösning måste man tänka på att bilder i bitmap-format får kraftigt försämrad kvalitet vid uppskalning. Det är därför lämpligt att använda en bitmapbild som är anpassad för högre upplösningar eftersom nerskalning ger ett bättre resultat än uppskalning för dessa bilder. Ett alternativ är att använda vektorbaserade bilder. En vektorbaserad bild kan skalas upp och ner utan att kvaliteten degraderas. Figur 2-21 visar en bitmapbild i normal skala till vänster. Till höger visas en uppskalad version av samma bild. Kvaliteten på den uppskalade bilden är märkbart sämre.

Figur 2-21 – Bilden till vänster visar en bitmapbild i normal storlek. Bilden till höger visar en uppskalad version.

[2] Bild från http://msdn.microsoft.com/en-us/library/windows/apps/hh780612.aspx

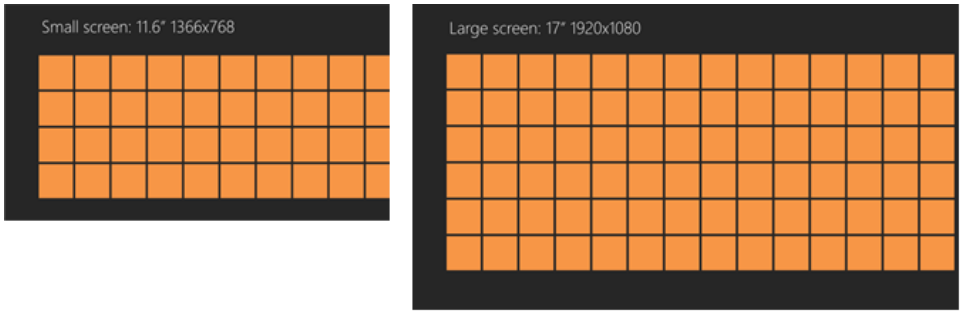
I Figur 2-22 visas en bild i normal storlek till vänster och en nerskalad version till höger. Kvaliteten på den nerskalade versionen är inte nämnvärt sämre än originalbilden.

Figur 2-22 – Bilden till vänster visar en bitmapbild i normal storlek. Bilden till höger visar en nerskalad version.

Figur 2-21 och Figur 2-22 visar att det är önskvärt att skala ner bilder snarare än upp. Bilder som anpassats för den högsta upplösningen som applikationen är tänkt att användas på är därför lämpliga att använda om bilderna är i bitmapformat. För text uppstår inte samma problem eftersom teckensnitt är anpassade för att visas korrekt i olika upplösningar.

2.9.3.2 Visa mer information

Ett alternativ till att skala applikationen är att man vid en större upplösning visar mer information. Detta kan vara lämpligt vid användade av t.ex. listor eller på sidor som innehåller mycket text. Genom att man visar mer information på skärmen undviker man att användaren behöver scrolla lika mycket i innehållet vilket gör det lättare för denne att ta del av informationen snabbt. Figur 2-23 visar ett exempel på hur man kan fylla skärmen med mer information vid högre upplösningar.



Figur 2-23 – Mer information visas vid en högre upplösning[3]

[3] Bild från http://msdn.microsoft.com/en-us/library/windows/apps/hh780612.aspx

2.9.3.3 Fast storlek

Ytterligare ett alternativ till att skala applikationen eller att visa mer information på skärmen vid högre upplösning är att helt enkelt anpassa applikationen efter enbart en upplösning.

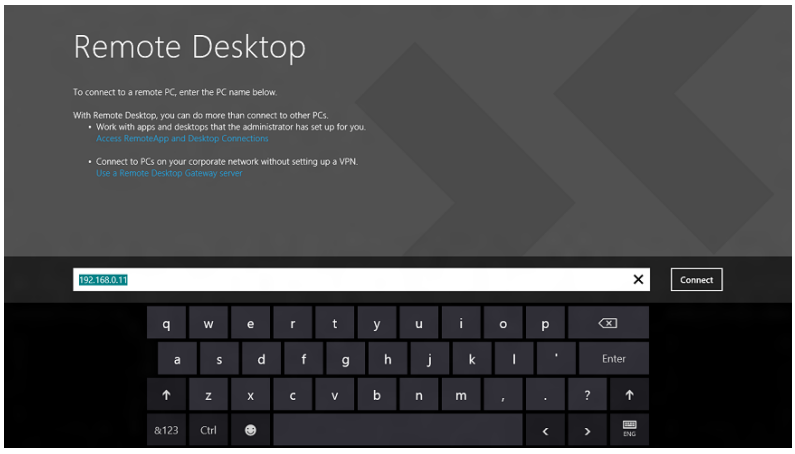
2.9.4 Animationer

När man utvecklar applikationer i Metro-stil står användarvänligheten i fokus. Men användarens upplevelser är också en viktig del av utvecklandet av en Metro-applikation. Det är viktigt att applikationen upplevs som responsiv och att det finns ett naturligt ”flyt” i applikationens olika delar. Om en användare navigerar till en ny sida och sidan omedelbart byts ut upplevs det som plötsligt och kan ge ett väldigt hårt intryck. Att istället använda animationer för att presentera den nya sidan gör att applikationen känns mer responsiv och ger användaren en helt annan upplevelse.

Animationer har en central roll i Metro-applikationer och hjälper till att skapa en bättre användarupplevelse.

2.9.5 On Screen Keyboard

Förutom att det är möjligt att ansluta ett externt tangentbord eller mus till plattan så finns också ett inbyggt skärmtangentbord i Windows 8 (On Screen Keyboard). Med hjälp av detta tangentbord kan användaren mata in värden utan att ha ett fysiskt tangentbord tillgängligt. Tangentbordet flyttar automatiskt innehållet för att inmatning skall kunna ske utan att dölja det för användaren vilket gör att designen inte behöver ändras för att tillåta användning av det. Figur 2-24 visar tangentbordet när användaren valt att mata in ett värde i ett textfält i Remote Desktop-appen. Textfältet flyttas automatiskt så att det placeras ovanför tangentbordet för att det inmatade värdet skall vara synligt.



Figur 2-24 – Remotedesktop-applikationen med skärmtangentbord

Det finns möjlighet att byta språk på tangentbordet genom att trycka på knappen längst ner till vänster i Figur 2-24. Här finns också en inställning för att använda en alternativ utformning på tangentbordet. Figur 2-25 visar en det alternativa tangentbordet.

Figur 2-25 – Alternativ utformning på tangentbordet

Om man väljer att använda denna utformning är det enklare att komma åt tangenterna när man håller plattan med en hand på vardera sida.

3 Projektbeskrivning

Projektet har innefattat att designa och konstruera ett grafiskt gränssnitt som är anpassat för en Windows 8-surfplatta, samt att implementera detta gränssnitt.

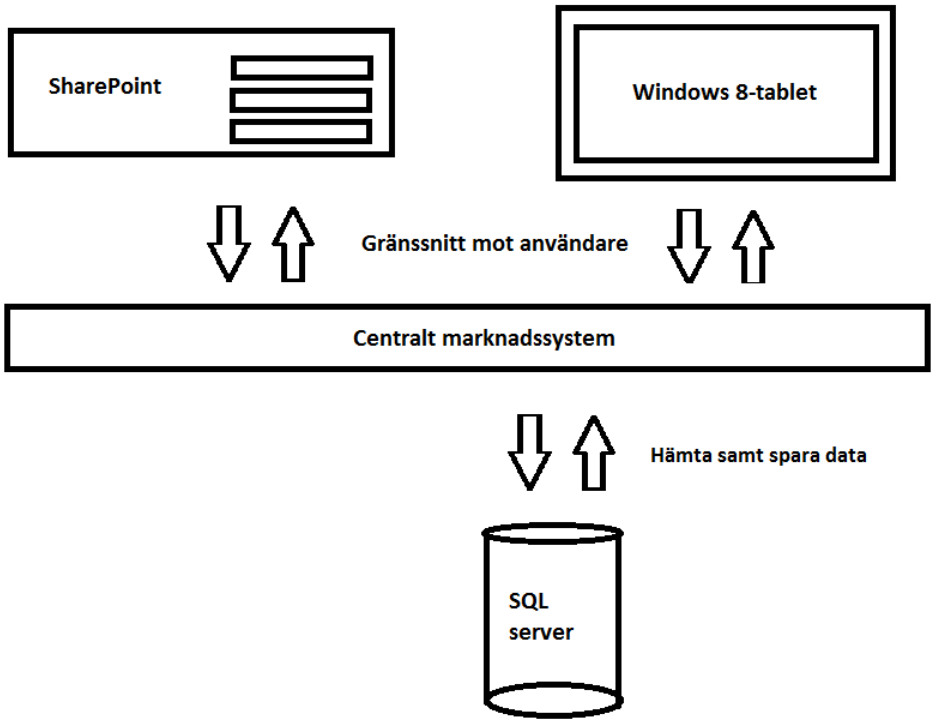
Syftet med detta projekt har varit att ta fram en prototyp-applikation för surfplattan och på så sätt tydligt visa vilka begränsningar och möjligheter som finns med denna typ av gränssnitt. Applikationen skall kommunicera med det centrala marknadssystem som Sogeti i Karlstad för närvarande utvecklar åt en stor svensk pappers-koncern. Här skall möjlighet att hantera ordrar och kunder i det centrala systemet finnas.

I detta avsnitt kommer de olika delarna av projektet att beskrivas. Inledningsvis kommer en översikt över systemet att ges. Här beskrivs systemet som helhet utan att gå in på detaljer. Därefter kommer de olika versionerna av användargränssnittet som utvecklats under projektets gång att presenteras. Detta syftar till att visa hur prototypen kan ses och användas från ett användarperspektiv samt varför det slutgiltiga användargränssnittet ser ut som det gör. Slutligen kommer de tekniska aspekterna av projektet att beskrivas. Syftet med detta är att ge en bild över hur de olika delarna i gränssnittet implementerats samt hur kommunikation med det centrala marknadssystemet sker på en teknisk nivå.

3.1 Systemöversikt

Det nya gränssnittet skall fungera som ett alternativ till det gränssnitt som redan existerar, utvecklat i SharePoint. Det existerande gränssnittet beskrevs närmare i avsnitt 2.3.

Prototypen som utvecklats ger användaren möjlighet att administrera ordrar och kunder i det centrala marknadssystemet. Figur 3-1 visar en förenklad, övergripande bild av hela systemet. Både det befintliga SharePoint-gränssnittet och det nya gränssnittet arbetar mot samma data genom att de använder sig av funktioner i det centrala marknadssystemet för att hämta samt spara data. Detta gör att det nya gränssnittet har minimal påverkan på de redan befintliga delarna och möjliggör att flera olika gränssnitt kan jobba samtidigt mot samma data. I huvudsak innehåller systemet ordrar och kunder som var för sig har ett stort antal egenskaper som kan modifieras av användaren.



Figur 3-1 – Förenklad bild av systemet

När användaren navigerar till en del av användargränssnittet kommer den information som användaren frågat efter att hämtas från databasen via det centrala systemet. Användaren kan sedan modifiera denna information på surfplattan. Om användaren väljer att spara informationen som ändrats kommer informationen i databasen att uppdateras med de nya värdena. Så länge användaren inte väljer att spara den modifierade informationen kommer ändringarna endast vara lokala.

3.2 Användargränssnitt

För att användaren enkelt skall kunna hämta samt modifiera data behövs ett användargränssnitt på surfplattan. Användargränssnittet har under projektets gång kunnat utvecklats fritt inom ramarna för dessa grundkrav från Sogeti:

- Gränssnittet skall vara estetiskt tilltalande. Detta innebär att det ska se bra ut samtidigt som det skall vara lätt och roligt att använda.
- Gränssnittet skall innehålla funktioner för att hantera ordrar och kunder i det centrala marknadssystemet
- Gränssnittet skall använda sig av befintliga tjänster i systemet

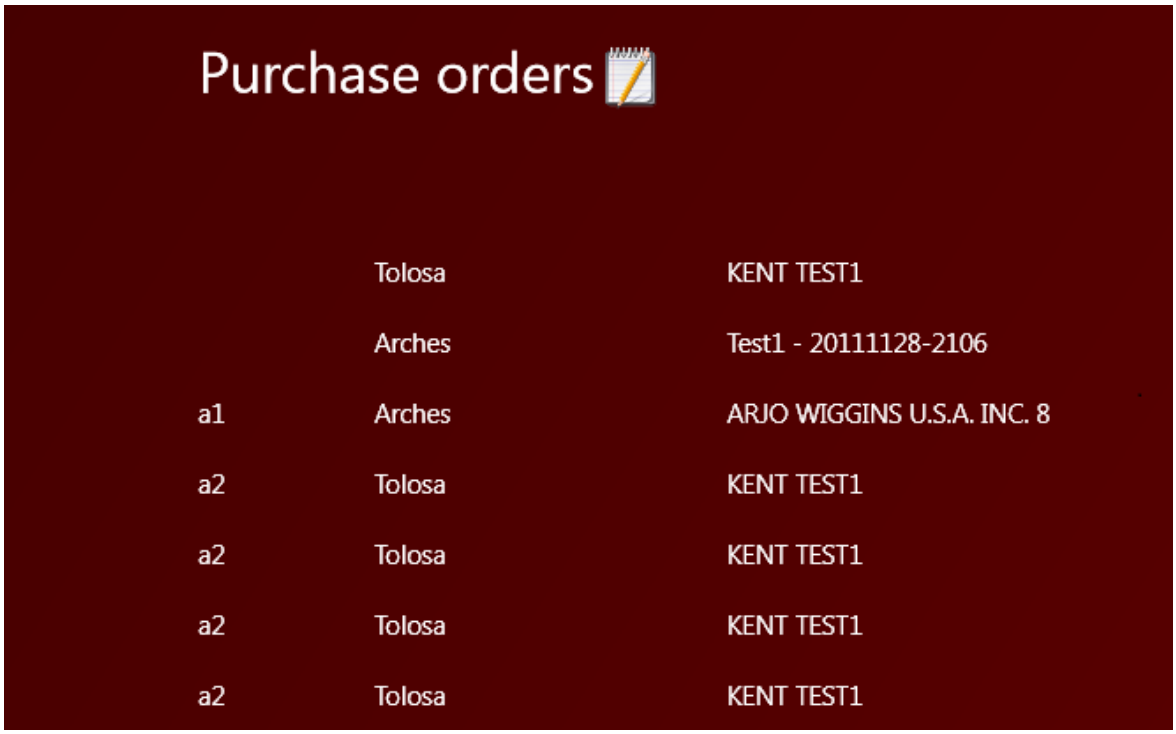
- Gränssnittet skall så långt som det är möjligt följa Microsofts riktlinjer för utvecklandet av användargränssnitt för Metro-applikationer[23]

Vad som egentligen är estetiskt tilltalande varierar från person till person. Stora delar av detta avsnitt bygger därför på personliga åsikter, tidigare erfarenheter och upplevelser vid användandet av systemet. Det finns dock riktlinjer att gå efter och ett gammalt system att använda som referens vilket gör att gränssnittet inte helt baseras på personligt tycke och smak.

Detta avsnitt inleds med att presentera de olika versionerna av gränssnittet som utvecklats. Syftet med detta är att ge en bild över varför det slutgiltiga gränssnittet ser ut som det gör. För den slutgiltiga versionen kommer de olika funktionerna i systemet att presenteras och en bild över hur systemet kan användas kommer att ges.

3.2.1 Version ett

Version ett av användargränssnittet var tänkt att fungera som en bas för fortsatt utveckling. Det konstruerades med syfte att ha någonstans att presentera den data som hämtades från databasen i ett tidigt skede. Detta gränssnitt var ej tänkt att representera den färdiga produktens gränssnitt. I Figur 3-2 visas det ursprungliga gränssnittet för listning av ordrar (Inzoomad vy).



Figur 3-2 – Listning av ordrar i den första versionen av gränssnittet (Inzoomad vy).

I listningen av ordrar i version ett visades ordernummer, produktionsbruk samt kundens namn. Listan saknar rubriker och utöver orderlistan finns det enbart en titel och en bild. När användaren väljer en order i listan presenteras detaljerna för den valda ordern. Figur 3-3 visar gränssnittet för orderdetaljerna i version ett av gränssnittet. Markerat med blått i figuren är menyn som användes för navigering. Den gröna markeringen visar huvudytan för innehållet.

Figur 3-3 – Gränssnittet för att visa orderdetaljer i version ett.

När användaren tryckte på en av rubrikerna byttes innehållet i det gröna området ut. All data presenterades i version ett i textfält som var placerade i en grid.

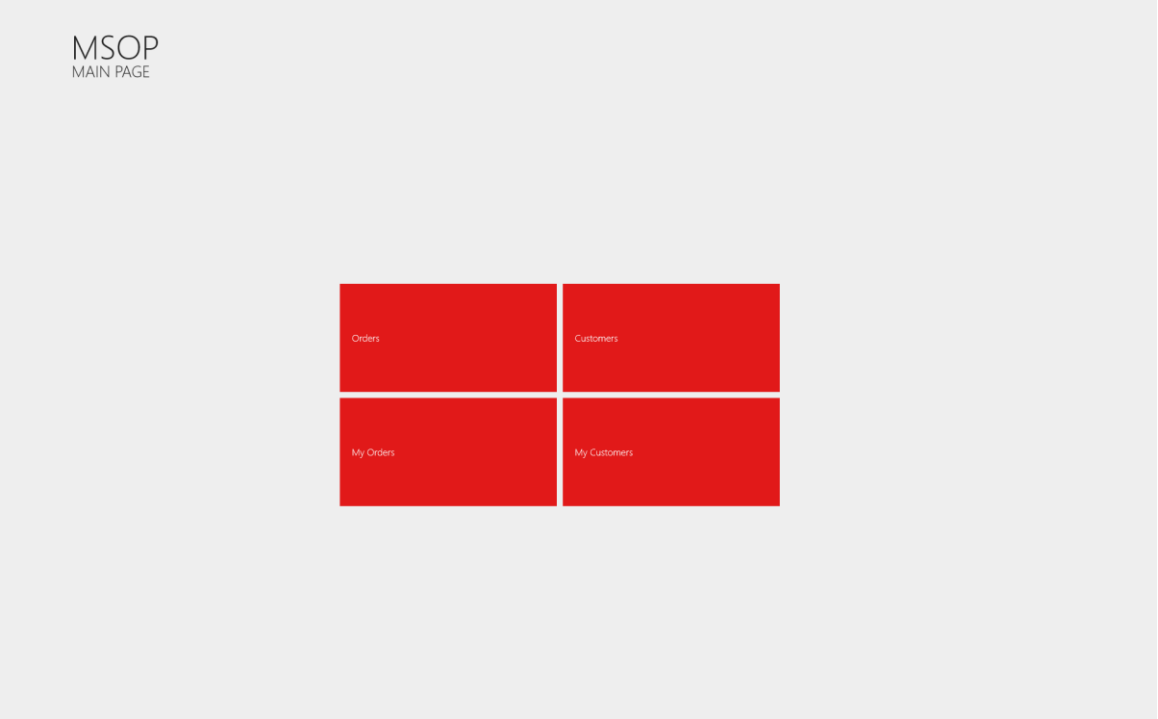
Version ett av gränssnittet kunde presentera data från databasen men hade ett antal nackdelar :

- Gränssnittet var inte speciellt estetiskt tilltalande.
- Animationer[4] saknades vilket skapade en väldigt hård upplevelse vid navigering. Det upplevdes som att applikationen ryckte till vid byte mellan vyer.
- Kontrollerna var placerade i en grid vilket gjorde XAML-koden komplex och ändringar svårare.
- Textfält och titlar var separata enheter vars XAML kopierats många gånger vilket gjorde XAML-koden komplex och ändringar svårare.
- Gränssnittet skalade inte bra till olika upplösningar och var anpassat efter upplösningen på utvecklingsdatorn då plattan inte anlänt ännu.

[4] Med animationer avses att element på skärmen ändrar antingen position eller utseende över tid.

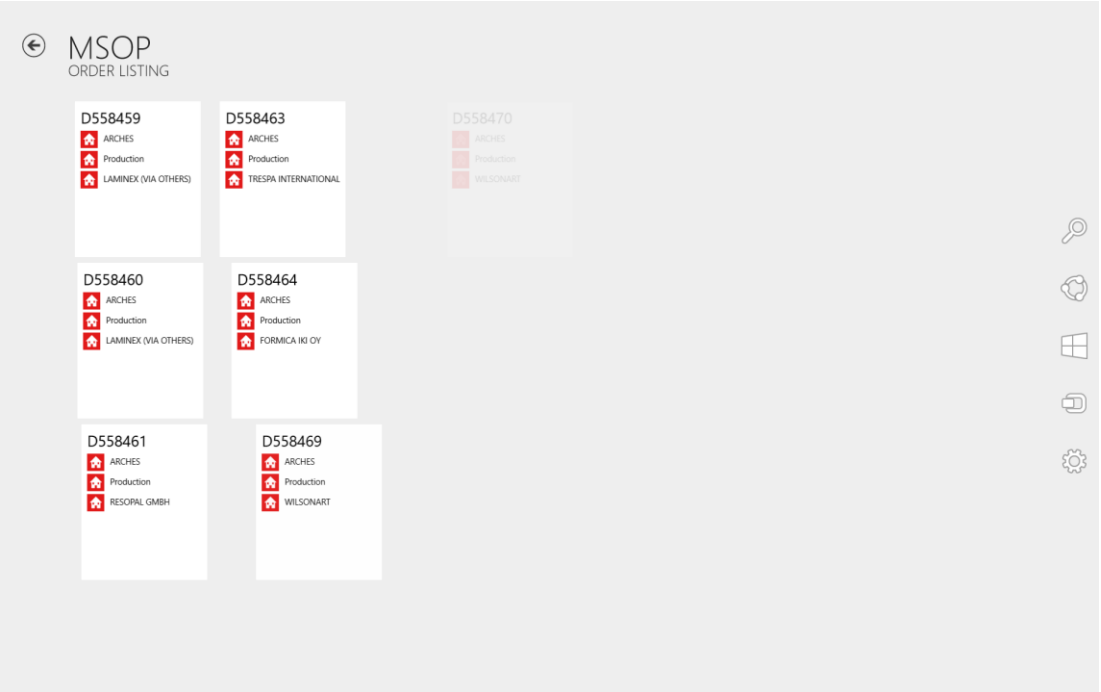
3.2.2 Version två

Version två av gränssnittet var tänkt att angripa de problem som fanns med version ett. Dessutom hade plattan anlänt vid utveckling av detta gränssnitt vilket gjorde det möjligt att testa det i den verkliga miljön. Version två av gränssnittet utvecklades med syfte att framförallt vara lättare att ändra i framtiden samtidigt som det skulle vara mer estetiskt tilltalande. Figur 3-4 visar huvudbilden i version två av gränssnittet. De olika delarna av systemet är: Orders, Customers, My Orders, My Customers.



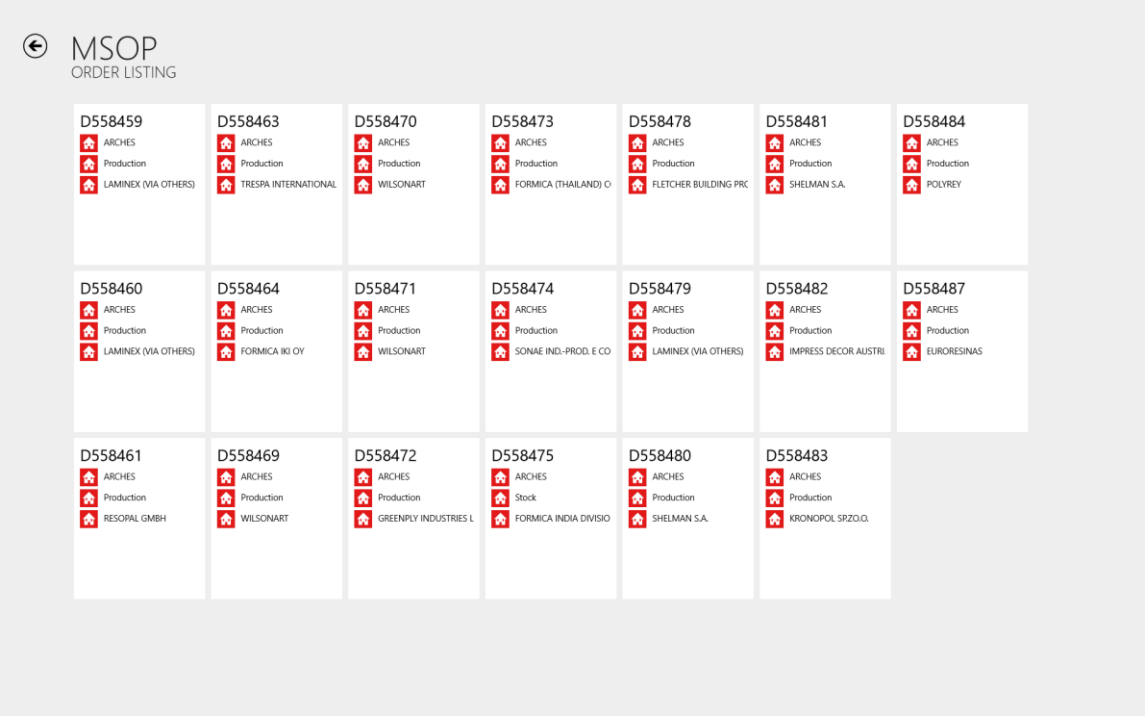
Figur 3-4 – Huvudbilden i version två av gränssnittet

Huvudbilden i version två agerar som ett nav mellan de olika delarna i applikationen. Högst upp på varje sida som användaren navigerar till presenteras titeln på applikationen tillsammans med en undertitel som visar vilken sida användaren befinner sig på. Genom att trycka på en av rutorna navigeras användaren till vald del av applikationen. I Figur 3-5 visas orderlistan strax efter att användaren navigerat till den. Varje vit ruta är ett element som är tänkt att representera en order. Elementen animeras in ett efter ett i listan. Dessutom animeras den nya titeln och undertiteln samt en bakåtknapp in. Med animeras menas i detta fall att de aktuella elementen skjuts in från höger samtidigt som opaciteten ökar så att de gradvis går från osynliga till fullt synliga. Tanken med animeringen är att skapa en mjuk övergång vid navigering. Elementen animeras in från höger till vänster / uppifrån och ner och slutligen hamnar de i en grid.



Figur 3-5 – Varje element i orderlistan animeras in från höger till vänster.

När hela orderlistan animerats in kommer så många element som får plats på den aktuella skärmen att visas i en grid. Användaren ges dessutom möjlighet att scrolla i sidled för att visa fler element om alla inte ryms på skärmens yta. I Figur 3-6 visas orderlistan strax efter att animeringen av element slutförts.



Figur 3-6 – Alla element har animerats in och fyller skärmen

Varje element representerar en order. För varje order visas följande information uppifrån och ner :

- Ordernummer
- Produktionsbruk
- Ordertyp
- Kundnamn

Genom att tycka på en order kan användaren visa en detaljerad vy över orderns innehåll. Användaren kan också använda bakåtpilen för att navigera till den föregående sidan.

Tanken med att presentera orderlistan på detta sätt är att skapa en mer estetiskt tilltalande upplevelse för användaren som bättre passar in i Windows 8-gränssnittet i stort. En nackdel med att presentera ordrarna på detta sätt är att det kan bli mindre överskådligt och svårare att hitta informationen som man letar efter. Det är därför viktigt att göra en avvägning om vilken data som är lämplig att presentera på detta sätt. När användaren tryckt på en order navigeras han eller hon till sidan för orderdetaljer. Animationer används för att skapa en mjukare och mer responsiv upplevelse för användaren vid navigering.

Figur 3-7 visar gränssnittet för orderdetaljerna i version två av grässnittet.

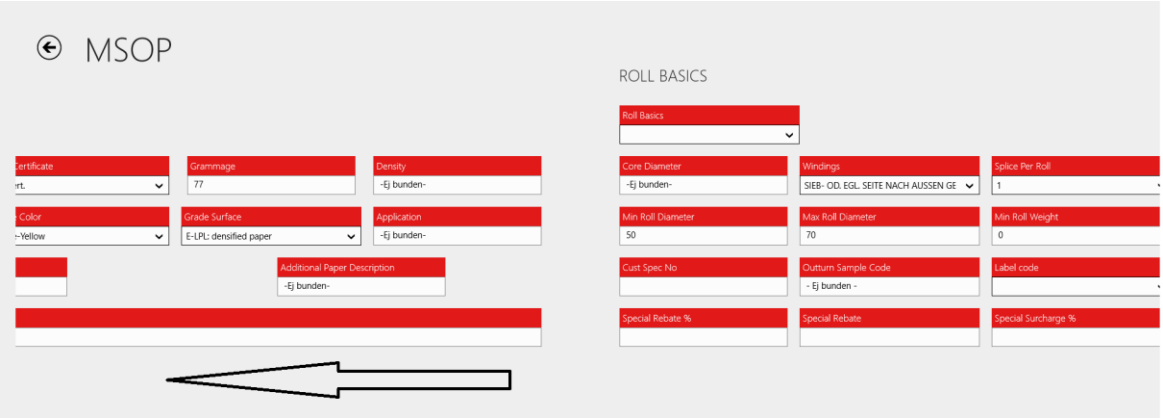
Figur 3-7 – Gränssnittet för orderdetaljer I version två av gränssnittet

Användargränssnittet för orderdetaljer har ändrats mycket från version ett. Elementen är nu implementerade som User Controls[5] för att det liknande utseendet skall kunna återanvändas. Detta gör det enklare att lägga till, ta bort eller placera om element i gränssnittet. Utöver detta kan val nu också utföras med hjälp av dropdown-listor.

Navigeringen sker inte som tidigare genom en topp-placerad meny. Användaren ges istället möjlighet att scrolla i sidled mellan de olika delarna i orderdetaljvyn. Genom att föra fingret

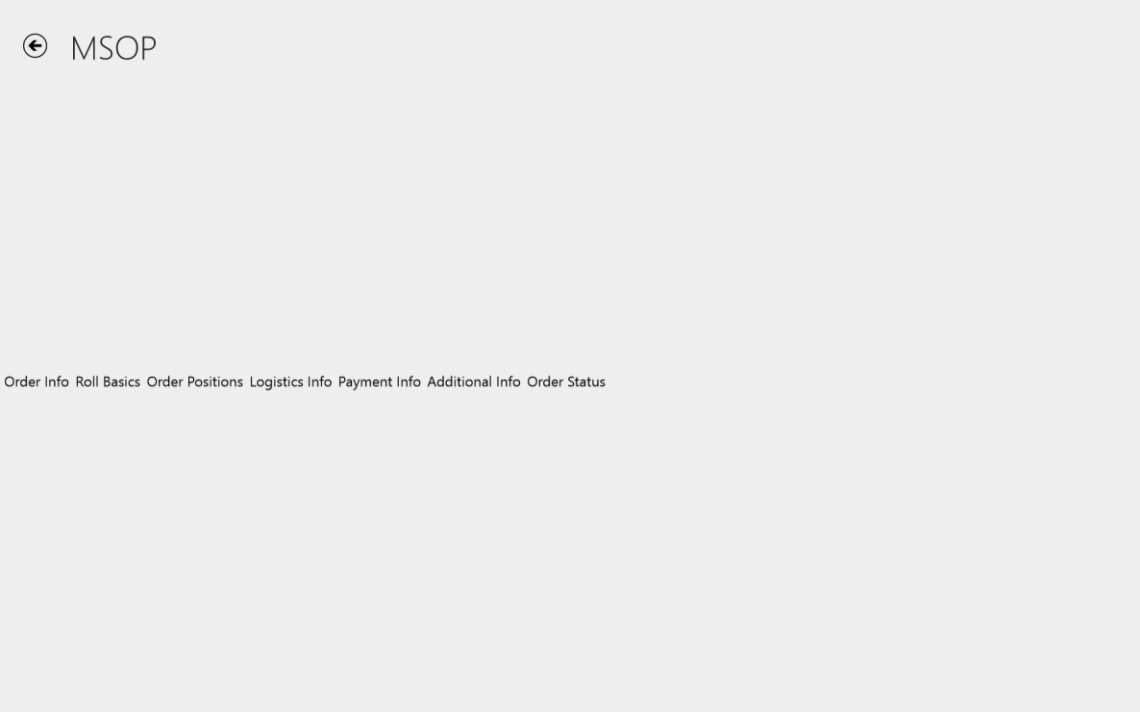
[5] En User Control är en återanvändbar kontroll som satts samman av flera andra kontroller.

över skärmen kan användaren scrolla i sidled. I Figur 3-8 visas gränssnittet under pågående scrollning av användaren. Pilen representerar den riktning som användaren för fingret i.



Figur 3-8 – Användaren kan scrolla mellan de olika delarna i orderdetaljvyn.

Tanken med att använda denna navigationsmetod var att användaren skulle få en upplevelse av att orderdetaljvyns olika delar hänger sömlöst ihop. Det kan dock vara tidsödande att behöva scrolla igenom de olika delarna av orderdetaljvyn för att komma till den sida man är intresserad av. I ett försök att avhjälpa detta infördes semantisk zoom[64]. Genom att nypa med fingrarna på skärmen zoomas vyn ut och en alternativ vy presenteras. Denna vy visas i Figur 3-9.



Figur 3-9 – Semantisk zoom, utzoomad vy

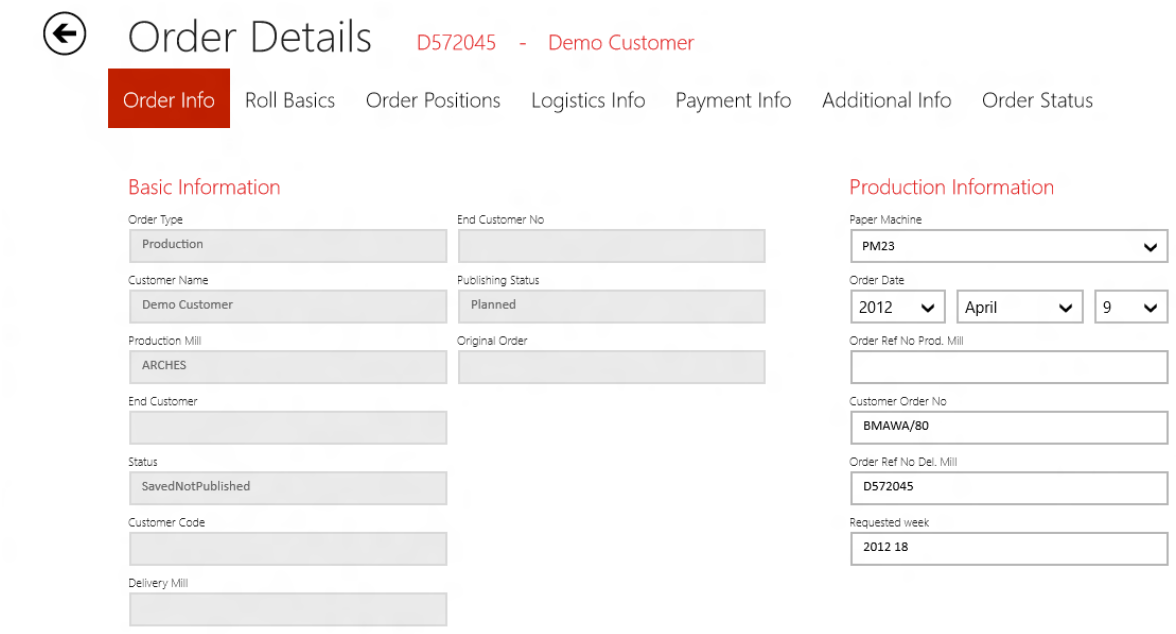
Genom att välja en av kategorierna i den utzoomade vyn kommer användaren att automatiskt navigeras till den valda delen i den inzoomade vyn. Detta gör att man enkelt kan navigera mellan de olika delarna utan att behöva scrolla förbi de delar man inte är intresserad av. Implementationen med hjälp av den inbyggda kontrollen för semantisk zoom har dock ett antal problem vilket gjorde att detta navigationssätt inte kunde användas på det sätt som det var tänkt :

- Gränssnittet blir ryckigt vid in- och utzoomning vilket skapar en dålig upplevelse när man navigerar mellan vyerna.
- När man kommer till den inzoomade vyn kommer det valda elementet att presenteras på samma ställe på skärmen som rubriken i den utzoomade vyn befann sig istället för att placeras till vänster. Detta gör att användaren måste justera placeringen genom att manuellt scrolla i vyn varje gång.

Efter att ha använt version två av gränssnittet under en tid stod det klart att gränssnittet var något osmidigt att använda (mycket scrollning krävdes och det var svårt att snabbt navigera till informationen man sökte). Eftersom den semantiska zoomen inte fungerade som väntat kunde man inte navigera mellan de olika delarna på ett bra sätt. Detta gjorde att det var osmidigt att navigera till den information man sökte. Dessutom kändes det som att för mycket information presenterades på en gång för användaren vilket gjorde orderdetaljvyn oöverskådlig. Med bakgrund av detta utvecklades en tredje version av gränssnittet.

3.2.3 Version tre

Startsidan och orderlistningen i version tre av gränssnittet har inte ändrats. Orderdetaljervyn har dock ändrats. Figur 3-10 visar en del av orderdetaljvyn i det nya gränssnittet.



Figur 3-10 – En del av orderdetaljvyn i version 3 av gränssnittet.

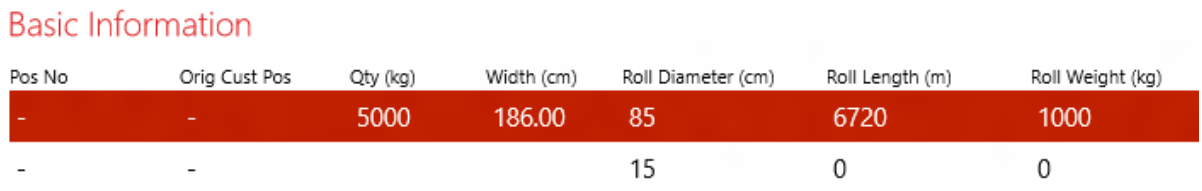
I version tre av orderdetaljvyn har version ett och version två kombinerats för att skapa en mer överskådlig och lättnavigerad vy. Den topplacerade menyn från version ett har återinförts. Samtliga fält för varje menyalternativ är placerade i en horisontellt scrollbar vy som låter användaren visa mer information genom att föra fingret över skärmen i önskad riktning (se Figur 3-8 för en bild över scrollningen i version två, version tre fungerar på samma sätt).

Utöver att grund-designen ändrats har också ett antal olika delar införts. Varje sida har en så kallad ”AppBar” som låter användaren interagera med applikationen på olika sätt beroende på vad som för närvarande visas i vyn. Figur 3-11 visar denna Appbar.

Figur 3-11 – AppBar med funktionen save order i orderdetaljvyn.

AppBar:en är från början dold. För att visa AppBare:en sveper användaren med fingret från nederkanten på skärmen upp mot mitten. AppBar:en kommer då att visas och användaren kan utföra de operationer som är möjliga för den aktuella vyn.

Save Order är tillgänglig från alla delar av orderdetaljvyn. Vissa delar av orderdetaljvyn låter användaren lägga till eller modifiera tabeller som visas i vyn. I dessa fall kommer ytterligare alternativ att visas i AppBar:en. Figur 3-12 visar en del av den tabell som återfinns under menyalternativet ”Order Positions”.



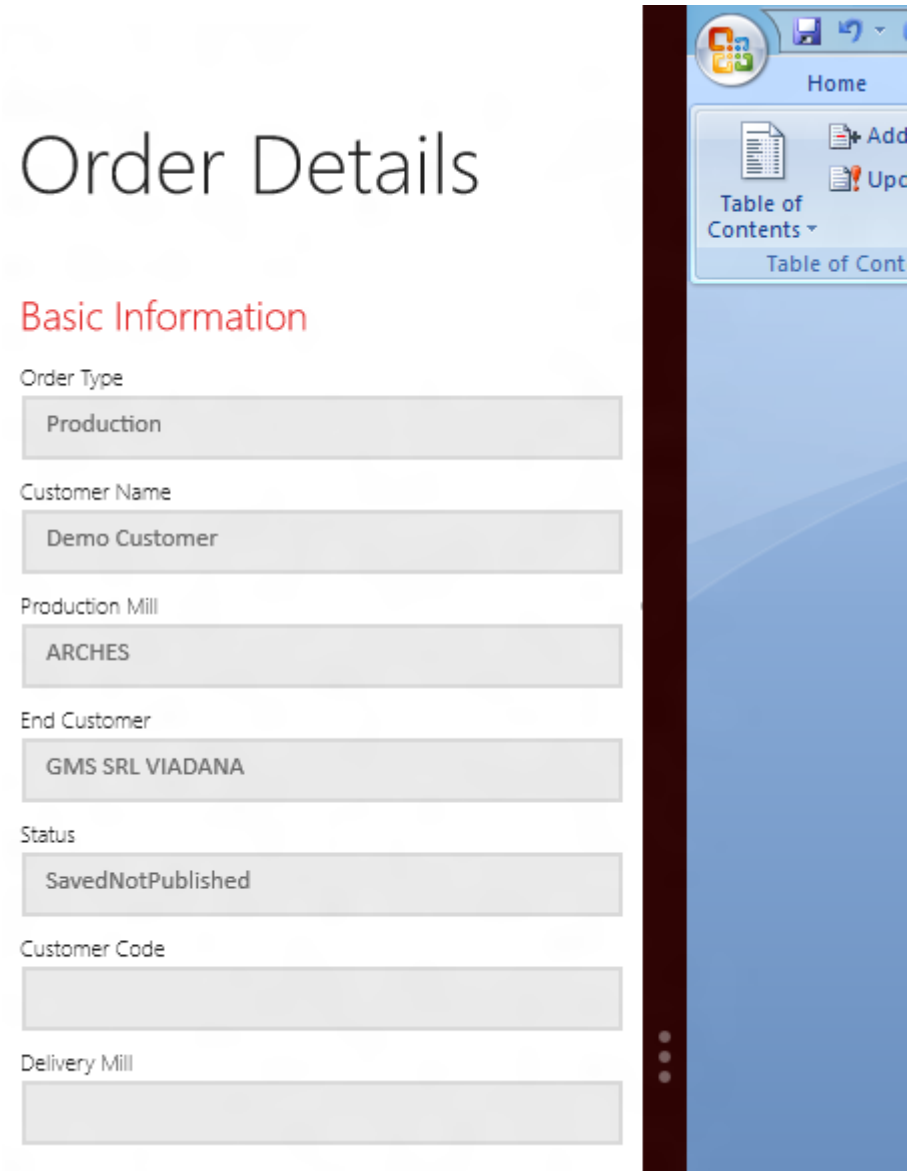
Figur 3-12 – Tabell från order positions

Denna vy visar information om varje orderrad för den aktuella ordern (en orderrad är en beställd kvantitet av en viss produkt). Användaren har här valt att markera den första orderraden. Om användaren nu väljer att öppna den AppBar som existerar i order positions-vyn kommer AppBar:en att innehålla ytterligare alternativ. Figur 3-13 visar dessa alternativ.

Figur 3-13 – Ytterligare alternative i order positions vyn

Genom att trycka på add item kan en ny orderrad läggas till. Genom att klicka på edit item kan den valda orderraden redigeras. I båda fallen kommer en ny vy att presenteras för användaren där möjlighet att redigera orderradens innehåll ges. Denna vy använder en fade-in animation, dvs. opaciteten ökar gradvis från 0 tills dessa att vyn är helt synlig för att skapa en mjuk övergång mellan vyerna. Användaren kan i den nya vyn välja att spara eller avbryta redigering/tillägg av orderrad (genom att alternativen i AppBar:en byts ut) och tabellen i order positions-vyn kommer att uppdateras. Genom att använda AppBar:en ges användaren möjlighet att enkelt utföra vy-specifika operationer från samma ställe på skärmen oavsett vilken vy han/hon befinner sig på.

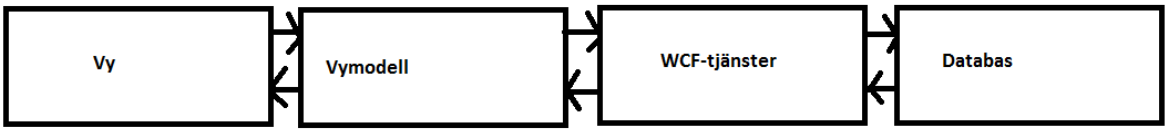
I version tre av gränssnittet har även möjlighet att använda applikationen i snapped view lagts till. Snapped view innebär att appen endast tar upp en liten del av skärmen och att ett annat program kan göra anspråk på den resterande skärmytan (se Figur 2-18 för ett helskärmsexempel). Denna funktion kan vara användbar om användaren t.ex. fått ett mail med information som rör ordern och han/hon vill kunna interagera med applikationen samtidigt som mailet är öppet i en annan applikation. Figur 3-14 visar applikationens utseende i snapped view.



Figur 3-14 – applikationen i snapped view. Till höger visas en annan applikation

I snapped view läggs alla fält i vyn vertikalt och vertikal scrollning möjliggörs. Detta gör att användaren kan interagera med applikationen även i snapped view.

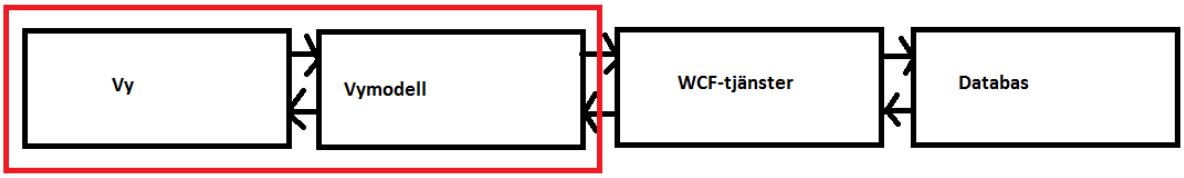
3.3 Implementationsdetaljer



Figur 3-15 – Översikt över systemet och kommunikationen mellan de olika delarna

I föregående avsnitt beskrevs de olika versionerna av användargränssnittet och en bild över hur systemet kan användas från ett användarperspektiv gavs. I detta avnsnitt kommer implementationen av användargrännsnittet samt dess kommunikation med det centrala systemet att presenteras. Figur 3-15 visar en översiktsbild över systemet och hur kommunikationen mellan de olika delarna sker. I detta avsnitt kommer varje ruta i figuren att beskrivas. Utöver detta kommer också gränssnitten mellan de olika delarna att beskrivas.

3.3.1 Vyer och Vymodeller



Figur 3-16 – Delarna som beskrivs i detta avsnitt är markerade med rött

3.3.1.1 Vy

Varje del av applikationen som användaren kan navigera till benämns vy. En vy innehåller två olika delar:

- XAML-fil – här definieras hur gränssnittet skall ritas upp med hjälp av XAML-taggar
- Code-behind – Innehåller kod som är specifik för den aktuella vyn (t.ex. kod för vad som skall hända när användaren navigerats till vyn)

Vyns uppgift är att presentera information för användaren samt låta användaren modifiera den information som visas. En vy består av ett antal olika kontroller som användaren kan interagera med. I detta avsnitt kommer inte uppbyggnaden av alla olika vyer i applikationen att presenteras, istället kommer några viktiga delar att beskrivas.

Sidor

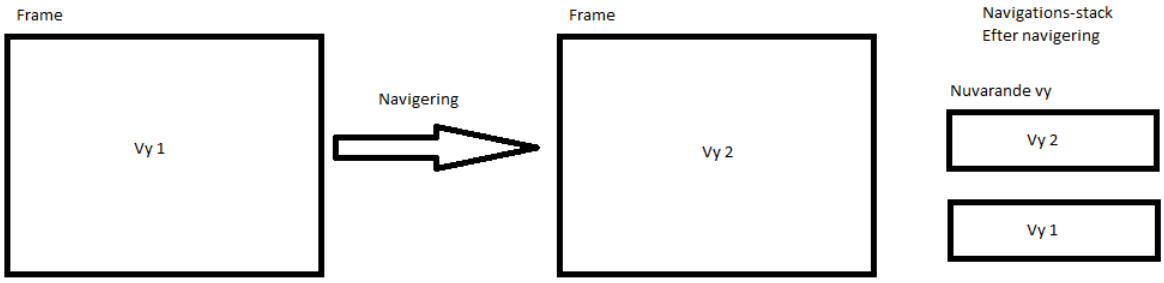
Varje vy i applikationen är implementerad som en ”Page” (sida) i XAML. Genom att använda sidor kan man enkelt dela upp de olika delarna av applikationen i separata filer samtidigt som navigering mellan dem hålls enkel. Sidor möjliggör också förändring av gränssnittet beroende på vilket läge applikationen befinner sig i (Om användaren t.ex. roterar surfplattan kan användargränssnittet anpassas).

Navigering mellan sidor

Navigering mellan sidor i applikationen sker på två olika sätt:

- Med hjälp av frames
- Genom användandet av en ContentControl

En frame är en del av sidan som kan innehålla en annan sida. Varje frame innehåller också en egen navigations-stack för att enkelt kunna navigera framåt eller bakåt mellan de sidor som användaren besökt. Navigation med frames sker från applikationens huvudsida till listning av ordrar eller kunder samt mellan listan av ordrar eller kunder till detaljer om ett specifikt objekt. Figur 3-17 visar hur navigering med hjälp av en frame sker.



Figur 3-17 – Navigering med frames, den aktuella vyn byts ut

Den senaste sidan läggs överst i navigations-stacken. Om användaren väljer att navigera bakåt kommer den tidigare besökta sidan att visas. Navigering med hjälp av frames sker genom ett anrop till den aktuella framens Navigate-metod. Navigate-metoden används på följande sätt:

Frame.Navigate(typeof(<klassen för sidan man vill navigera till>), <valfri parameter som objekt>)).

Genom att överlagra funktionen OnNavigatedTo på den sida som navigerats till kan man komma åt parametern som skickades med vid anropet till navigate-metoden. Figur 3-18 visar ett exempel på hur detta används i applikationen.

```
protected override void OnNavigatedTo(NavigationEventArgs e)
{
    ViewModel = new MainOrderInfoViewModel((MSOPService.PurchaseOrder)e.Parameter);
    this.DataContext = ViewModel;
    fadeInBackButton.Begin();
}
```

Figur 3-18 – Överlagring av metoden OnNavigatedTo i MainOrderInfo

I detta exempel har användaren klickat på en order i orderlistan och Frame.Navigate har använts för att navigera till MainOrderInfo, som är huvudvyn för att visa orderdetaljer. Genom att skicka med den valda ordern som parameter vid navigeringen kan man använda ordern på den nya sidan. För att gå till föregående sida kan Frame.GoBack() användas.

Om man istället väljer att navigera med hjälp av en ContentControl anges vilken sida som skall visas i den direkt genom att sätta dess Content-egenskap. ContentControl innehåller inget inbyggt stöd för att navigera mellan tidigare besökta sidor. Figur 3-19 visar hur en ny sida kan visas i en ContentControl. I detta fall är OrderInfo av typen Page och när koden exekverats kommer OrderInfo-vyn att visas.

```
contentControl.Content = new OrderInfo();
```

Figur 3-19 – Navigering med hjälp av en ContentControl

Anledningen till att ContentControl används istället för en frame på vissa ställen i applikationen är att man kan använda en instans av ett objekt istället för att bara ange klassen för den sida man navigerar till. Detta gör att man kan fylla den använda instansen med valfri information samt att samma instans kan återanvändas istället för att nya instanser måste skapas vid varje ny visning av sidan. Detta är speciellt fördelaktigt när användaren navigerar mellan de olika delarna i t.ex. orderdetalj-vyn där samma sida kan besökas flera gånger med korta intervall.

Inmatning

Det finns tre huvudsakliga typer av inmatning i applikationen:

- Text
- Förval
- Nummer

För att slippa kopiera samma kod flera gånger återanvänds så kallade UserControls.

UserControls

En UserControl är en kontroll som satts samman av flera andra kontroller. Genom att använda UserControls kan man bygga kontroller som kan användas på flera olika ställen i applikationen utan att behöva kopiera all XAML-kod. Man kan dessutom definiera nya egenskaper på en UserControl. I applikationen finns tre olika UserControls, InfoTile, BoxTile och DatePicker.

InfoTile

InfoTile är en kontroll som satts samman för att slippa upprepa ett vanligt mönster i applikationen: ett textfält med en titel. Eftersom detta mönster av kontroller återkommer många gånger i applikationen har InfoTile skapats. Figur 3-20 visar ett exempel på hur en instans av denna UserControl kan se ut. I detta fall har titeln satts till ”Order Date” och texten i textfältet har satts till orderns orderdatum-egenskap.

Figur 3-20 – En instans av InfoTile i vyn

På InfoTile kan användaren sätta ett antal olika egenskaper:

- Title – Titeln som visas ovanför textrutan
- Text – Texten som visas i textfältet
- TitleWidth – Bredden på titeln
- InfoWidth – Bredden på textfältet
- IsReadOnly – Anger om användaren kan skriva i textrutan eller ej

- IsEnabled – Om satt till false kommer textfältet att bli grått och användaren kan ej skriva i det. Genom att sätta IsEnabled till false istället för att använda IsReadOnly blir det tydligare att det inte går att skriva i textfältet.
- InputType – Anger vilken typ av input som textfältet är avsett för. Möjliga värden är ”Number” eller ”Decimal”. Om inget anges antas att valfri text kan skrivas i rutan. Om ”Number” eller ”Decimal” används kommer den numeriska delen av skärmtangentbordet automatiskt att visas när användaren vill göra en inmatning i textfältet. Dessutom kommer kontrollen att validera att korrekt inmatning sker och i annat fall markera detta.

BoxTile

BoxTile är en kontroll som satts samman för att slippa upprepa ett annat vanligt mönster i applikationen: en ComboBox med val och en titel. Figur 3-21 visar ett exempel på hur en instans av denna UserControl kan se ut. I detta fall har titeln satts till ”Paper Machine” och listan kommer att innehålla en lista över de pappersmaskiner som finns på det aktuella bruket.

Figur 3-21 – En instans av BoxTile i vyn

Användaren kan inte skriva i fältet, men om han/hon trycker någonstans på det visas en lista över möjliga alternativ. Figur 3-22 visar denna lista.

Figur 3-22 – Lista av alternativ i BoxTile

När användaren väljer ett alternativ i listan kommer listan att stängas och alternativet kommer att visas som det valda alternativet. BoxTile har ett antal egenskaper som kan sättas i XAML eller via kod:

- Title – Titeln som visas ovanför boxen
- BoxWidth – Bredden på boxen
- ItemsSource – Anges till en lista som innehåller de möjliga alternativen. Listan kan innehålla valfria objekt. Denna lista kan vara av flera olika typer. I applikationen

används listor av typen ObservableCollection. Observablecollection notifierar systemet om ett element läggs till eller tas bort i listan. Detta gör att vyn automatiskt kan uppdatera det som visas i listan beroende på hur listan ändras.

- SelectionChanged – Låter programmeraren ange ett event som innehåller kod som skall köras när det valda värdet ändras.
- DropDownClosed – Låter programmeraren ange ett event som innehåller kod som skall köras när listan stängs. Detta skiljer sig från SelectionChanged. SelectionChanged körs bara om valet i listan ändrats. Om användaren klickar på samma alternativ som redan var valt kommer alltså SelectionChanged inte att köras. DropDownClosed kommer att köras så fort listan stängs.
- ItemTemplate – ItemTemplate låter programmeraren specificera hur varje element i listan skall visas.
- SelectedValuePath – Anger vilken del av de objekt som listan består av som skall användas som det valda värdet.
- SelectedItem – Anger vilket objekt som är valt för tillfället. Till skillnad från det valda värdet kommer man med SelectedItem åt hela objektet och inte bara en av dess egenskaper

DatePicker

DatePicker låter användaren välja ett datum. Kontrollen består av tre ComboBoxar och låter användaren ange år månad och dag. Figur 3-23 visar ett exempel på hur denna kontroll kan se ut i vyn.

Figur 3-23 – En instans av DatePicker i vyn

Användaren har möjlighet att ange år månad och dag i kontrollen. När månaden ändras kommer antalet dagar att justeras beroende på vilken månad som valts. DatePicker har följande egenskaper:

- Title – Den text som visas ovanför ComboBoxarna
- BoxDate – Det datum som skall visas i kontrollen. Om datumet är satt till en bindning kommer det automatiskt att uppdateras när användaren väljer ett nytt datum.

Animationer

De olika vyerna innehåller ett antal animationer för att skapa mjuka övergångar mellan sidor eller för att visa ytterligare valmöjligheter för användaren. Dessa animationer startas antingen automatiskt (t.ex. när en sida byts ut) eller från kod.

3.3.1.2 Vymodell

Varje sida i applikationen har en så kallad Vymodell. Vymodellens uppgift är att tillgodogöra vyn med den data som krävs. Vymodellen innehåller de listor som visas på den aktuella sidan och andra egenskaper som vyn behöver för att kunna visa korrekt data. Tanken med detta är att vyn skall användas enbart för presentation av data i gränssnittet – vymodellen skall agera som adapter mot de olika klasser som används i applikationen (modellen).

Vymodellerna för de olika vyerna för orderdetaljer och kunddetaljer ärver från en de två basvymodellerna MainOrderViewModel och MainCustomerViewModel. Dessa vymodeller innehåller information som är övergripande för flera vyer. Exempel på detta är möjligheten att spara order- eller kund-data efter modifiering, samt referenser till nuvarande order eller kund.

3.3.1.3 Vy – Vymodell

För att vyerna och vymodellerna skall kunna kommunicera med varandra används ett designmönster kallat MVVM (Model View ViewModel). Detta mönster kommer nu att presenteras.

MVVM

I föregående avsnitt beskrevs vyerna och vymodellerna. För att användargränssnittet skall vara användbart krävs dock att dessa kan kommunicera med varandra och med modellen.

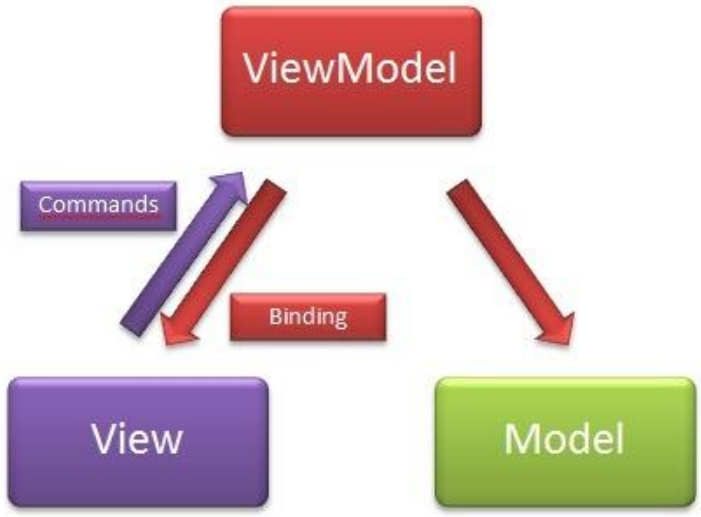
MVVM[21] är ett designmönster vars huvudsyfte är att separera implementationen av vyn från modellen men samtidigt göra det möjligt att enkelt visa och manipulera data i vyn.

I projektet har MVVM-mönstret tillämpats, även om det inte följs till fullo. Där många förändringar av vyn sker (t.ex. flertalet animationer som skall startas) implementeras istället koden för detta direkt i code-behind för vyn. Detta eftersom det kan vara svårt att påverka vissa egenskaper i vyn, då vymodellen inte har någon referens till den vy den som använder sig utav den aktuella vymodellen. Idealiskt är att så lite kod som möjligt skall finnas i vyns code-behind.

För att förstå hur applikationen är konstruerad är det bra att känna till hur MVVM fungerar på en konceptuell nivå. Detta avsnitt är tänkt att ge en grundläggande förståelse för designmönstret.

- M – Det första M:et i MVVM står för Model. Model är den bakomliggande modellen. Modellen är ansvarig för att lagra och utföra operationer på data. Exempel på klasser som tillhör modellen är en order med dess tillhörande egenskaper och operationer.
- V – Det första V:et står för View. View är själva gränssnittet, vyn. Vyn agerar gränssnitt ut mot användaren och är det som användaren ser och interagerar med när han eller hon använder applikationen.
- VM – VM står för View Model. View Model (Vymodell) är ett lager som ligger mellan vyn och modellen och har som uppgift att knyta ihop de två.

Figur 3-24 visar en principiell skiss över MVVM.



Figur 3-24 – Principiell skiss av MVVM [6]

Som Figur 3-24 visar kommunicerar vyn enbart med vymodellen som sedan sköter all kontakt med modellen. Bilden illustrerar också att denna kommunikation sker med kommandon och bindningar.

Att sätta upp all kommunikation med modellen direkt i vyns code-behind är fullt möjligt, men det bär med sig ett antal nackdelar:

[6] Bild från: http://www.silverlightblog.net/tag/mvvm/ (29/3-12)

- Alla objekt skapas direkt i vyn. Detta betyder att om man byter vy måste all kod antingen kopieras eller skrivas om.
- Om kontrollerna i vyn ändrar namn måste koden ändras
- Om man vill ge användaren möjlighet att ändra värdena i vyn måste man manuellt sätta alla värden från t.ex. listor eller textfält till modellen.
- Om ett värde i modellen ändras måste vyn uppdateras för att reflektera denna förändring.

Genom att använda sig av bindningar och en vymodell kan vyn hållas separat från modellen, vilket gör att förändringar är lättare att hantera. Nedan beskrivs hur man kan använda bindningar för kommunikation mellan vy och vymodell.

Bindningar

Bindningar är ett av de viktigaste verktygen när man implementerar MVVM. Vymodellen innehåller de listor och egenskaper som användaren behöver ha tillgång till i vyn. Genom att vyn binder till dessa egenskaper kommer systemet att ta hand om många av de problem som uppstår om man manuellt måste uppdatera vyn eller modellen vid varje ändring som sker. Det första man behöver göra för att en bindning skall fungera är att tala om för vyn vad den ska binda emot. Detta anges med egenskapen DataContext. Det är inte bara vyer som har denna egenskap, den kan sättas på flertalet kontroller. Figur 3-25 visar hur DataCotext kan sättas. Kodsnutten som visas återfinns i vyn OrderListing och ”this” refererar således till denna vy. fadeInBackButton.Begin() startar en animation som gradvis höjer opaciteten på bakåtknappen tills dess att den är fullt synlig och är inte en del av MVVM.

```
private async void Page_Loaded_1(object sender, RoutedEventArgs e)
{
    fadeInBackButton.Begin();
    ViewModel = new OrderListingViewModel();
    this.DataContext = ViewModel;
}
```

Figur 3-25 – Exempel på hur datacontext kan anges

Efter det att DataContext angetts till vymodellen OrderListingViewModel i exemplet ovan kan man binda till objekt i vymodellen. Bindningar anges med nyckelordet Binding. Som ett enkelt exempel visas i Figur 3-26 hur man kan binda texten i ett textfält till en egenskap hos vymodellen.

```
<TextBox Text="{Binding TextEgenskap, Mode=TwoWay}" />
```

Figur 3-26 – Bindning mot en egenskap i vymodellen

Vyn kommer nu att leta efter egenskapen ”TextEgenskap” i det DataContext som angetts (i detta fall OrderListingViewModel). Om denna egenskap existerar i vymodellen så kommer dess värde att visas som text i textfältet. Mode=TwoWay anger att ändringar i textfältet också skall verkställas även på den bundna egenskapen. För att vyn automatiskt skall uppdateras krävs det att egenskapen antingen är en så kallad Dependency Property eller att vymodellen implementerar INotifyPropertyChanged. Detta är två mekanismer som tillhandahålls av WinRT för att vyn skall känna av när en egenskaps värde ändrats och automatiskt uppdatera det data som presenteras.

Förutom att binda till enkla egenskaper kan man t.ex. binda till listor (ItemsSource i kontrollen BoxTile är ett exempel på detta, se avsnitt 3.3.1.1 under rubriken UserControls). Man kan också binda till egenskaper hos andra kontroller (t.ex. det valda objektet i en ComboBox). Bindningar har använts flitigt under projektets gång.

Fördelar med bindningar:

- Vyn behöver endast veta egenskapernas namn i vymodellen för att kunna binda till dem. Detta gör vyn oberoende av den bakomliggande modellen. Dessutom behöver objekten inte skapas i code-behind för vyn. Vyn kan alltså enkelt bytas mot en annan vy så länge bindningarna i den nya vyn anges korrekt.
- Om kontrollerna ändrar namn behöver ingenting ändras. Kontrollerna behöver i detta fall inte ens något namn.
- Vyn uppdateras automatiskt om ändringar i modellen / vymodellen sker.
- Modellen / vymodellen kan automatiskt uppdateras om ändringar i vyn sker.

Dessa fördelar kan spara utvecklings- och underhållstid i jämförelse med att inte använda bindningar.

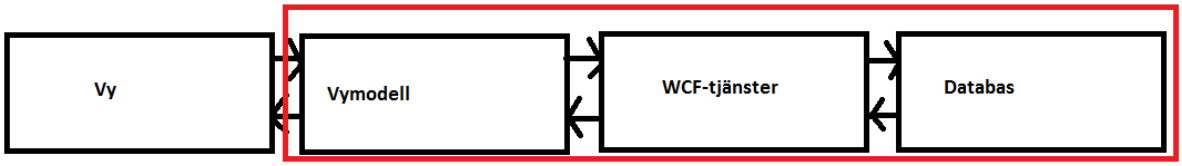
Kommandon

Kommandon är ett sätt att låta vyn exekvera kod i vymodellen. Ibland räcker det inte med att endast uppdatera egenskaper från vyn. Systemet måste kunna utföra operationer på data.

Exempel på operationer är att sortera en lista eller att spara en order till databasen. Det är här kommandon kommer in i bilden.

I traditionell event-hantering anger man event-handlers som skall köras när något sker i vyn. Detta kan vara t.ex. att användaren klickar på en knapp. Koden skrivs sedan i code-behind för vyn. Genom att skapa kommandon i vymodellen och binda till dessa i vyn kan koden istället placeras i vymodellen. Detta innebär att om vyn byts ut så behövs inte denna kod skrivas om eller kopieras, det räcker med att binda till kommandot i den nya vyn! Eftersom de egenskaper som används i vyn återfinns i vymodellen kan man också operera direkt på denna data från kommandona. Dessutom möjliggör kommandon automatisk avaktivering och aktivering av knappar baserat på villkor som kan anges vis skapandet av kommandot.

3.3.2 Kommunikation med databasen

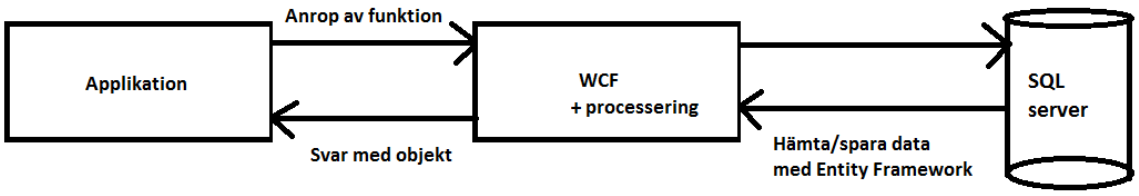


Figur 3-27 – Delarna som beskrivs detta avsnitt är markerade med rött

I avsnitt 3.3.1.2 beskrevs vymodellens syfte. I detta avsnitt kommer tjänsten som vymodellen och modellen kommunicerar med att beskrivas. Detta avsnitt kommer också att visa hur man anropar tjänsten. Eftersom kommunikationen mellan WCF och databasen inte är relevant från applikationens sida kommer denna endast att nämnas kort.

3.3.2.1 Användande och anrop av WCF

Tjänsten som applikationen anropar är en så kallad WCF-tjänst (Windows Communication Foundation[63]). WCF-tjänsten körs på serversidan och exponerar ett antal funktioner som kan exekveras från den konsumerande applikationen.



Figur 3-28 – Kommunikationen med WCF och databasen

WCF-tjänsten exponerar ett antal funktioner som kan användas från applikationen, dessa funktioner exekveras dock på servern. Efter att en funktion anropats kommer WCF-tjänsten att ansluta till databasen via Entity Framework[4]. Databasen är en SQL server databas som ligger på servern. När data hämtats från databasen processeras detta i tjänsten och relevant data skickas sedan till den anropande applikationen (Vid uppdatering sparas data i databasen via tjänsten). Det data som returneras kan direkt användas av applikationen som ett objekt av en viss typ (objektets typ beror på funktionen som anropats).

För att använda WCF-tjänsten i vymodellen måste en service-referens sättas upp. Detta kan göras direkt i Visual Studio genom att man i lösningen väljer att man vill lägga till en service-referens. Man anger sedan adressen till den WCF-tjänst man vill använda sig av och om tjänsten hittas ges man möjlighet att lägga till den. När en tjänst läggs till i projektet genereras alla de klasser och metoder som tjänsten exponerar lokalt i lösningen. Detta betyder att om tjänsten ändras måste man uppdatera referensen för att återspegla förändringarna. Med hjälp av de genererade klasserna och metoderna kan man sedan anropa tjänsten på nästan samma sätt som man anropar lokala funktioner. Figur 3-29 visar ett typiskt anrop till WCF-tjänsten i applikationen.

```
MSOPService.ClientServiceClient client = ServiceHelper.Client;
MSOPService.PurchaseOrder order = await client.GetPurchaseOrderAsync(Order.PurchaseOrderId);
```

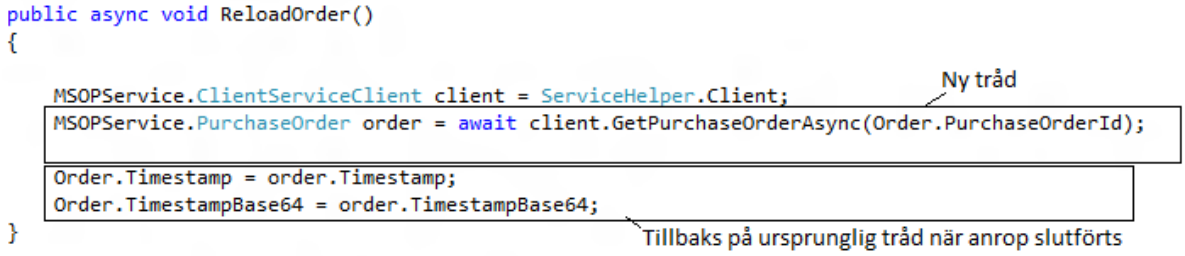
Figur 3-29 – anrop av WCF-tjänst

MSOPService.ClientServiceClient är en klass som möjliggör anrop av tjänsten. Denna klass autogenererades när tjänsten lades till i Visual Studio. MSOPService.PurchaseOrder är en klass som definierats i tjänsten och som genererats lokalt när man lade till tjänsten.

I applikationen används de genererade klasserna för att lagra och modifiera data. Man kan välja att mappa klasserna till lokala klasser, men i detta projekt har detta alternativ valts bort främst för att spara tid.

I exemplet i Figur 3-29 anropas en funktion i tjänsten som hämtar information om en viss order. Genom att skicka med en parameter med orderns id på samma sätt som parametrar kan skickas till funktioner som exekveras lokalt kan rätt order hämtas. Värt att notera i koden ovan är nyckelordet await. Ett anrop till tjänsten är asynkront, det låser inte den nuvarande tråden utan körs i bakgrunden och låter användaren fortsätta använda applikationen innan svaret från tjänsten återkommit. Problemet med detta är att resultatet inte kan användas innan tjänsten svarat. Detta gör att någon mekanism måste användas för att vänta på svar och sedan behandla svaret när det det anlänt från tjänsten.

En tidigare metod som använts för detta ändamål är att skapa en tråd och ange en så kallad callback. En callback är en funktion som körs efter det att den aktuella tråden slutfört sitt arbete. När callbacken anropas ges möjlighet att hämta resultatet från det ursprungliga anropet. Callbacken körs dock på samma tråd som anropet körs på vilket gör att problem kan uppstå med att t.ex. uppdatera användargränssnittet (uppdatering av användargränssnittet kan endast göras på en specifik tråd som har hand om detta). Dessutom kan trådhanteringen lätt bli komplicerad vid många anrop. Async och await är två nyckelord som lagts till i den senaste versionen av C#.NET och denna mekanism förenklar hanteringen av asynkrona anrop. Genom att ange att en funktion är asynkron med nyckelordet async kan await användas i funktionen. Figur 3-30 visar hur nyckelorden kan användas i en funktion. Figuren visar också vilken tråd koden exekveras på.



Figur 3-30 – en asynkron funktion

Genom att använda await kommer systemet automatiskt att skapa en ny tråd och köra det aktuella anropet på denna tråd. Raderna efter await kommer dock inte att exekveras innan ett svar från anropet inkommit. Istället kommer den ursprungliga tråden (den tråd i vilken await-anropet utförts) att kunna fortsätta användas av andra delar av applikationen. När anropet slutförts kommer den ursprungliga tråden att användas för att exekvera resterande kod i funktionen. Eftersom order skapats på den ursprungliga tråden kan detta objekt användas på denna tråd även i fortsättningen. Detta betyder att om ReloadOrder startas från t.ex. den tråd som kan ändra i användargränssnittet så kommer man kunna använda resultatet och direkt uppdatera användargränssnittet. Denna mekanism är till stor hjälp vid anrop av asynkrona funktioner.

3.4 Sammanfattning

Projektet har innefattat att designa och konstruera ett grafiskt gränssnitt som är anpassat för en Windows 8-surfplatta, samt att implementera detta gränssnitt. Gränssnittet arbetar mot ett centralt marknadssystem som Sogeti för närvarande utvecklar åt en kund.

Syftet med projektet har varit att ta fram en prototyp-applikation för surfplattan och på så sätt tydligt visa hur man kan använda denna platform i en affärs-orienterad miljö.

Ett antal olika versioner av gränssnittet har tagits fram under projektets gång och ett slutgiltigt användargränssnitt har därigenom växt fram.

Presentationen av data har skiljts från själva implementationen med hjälp av MVVM som är ett designmönster som bygger på användadet av bindningar och kommandon.

För att gränssnittet skall vara användbart måste data från det centrala marknadssystemet kunna hämtas, visas och modifieras. Data lagras i en SQL Server databas och kommunikationen med denna sker genom användande av befintliga WCF-tjänster som anropas asynkront från applikationen.

4 Resultat och utvärdering

I föregående avsnitt beskrevs implementationen av användargränssnittet samt hur kommunikationen sker från användare till databas. I detta avsnitt kommer denna implementation att utvärderas.

Att designa och kunstruera gränssnittet har varit en stor del av projektet. Detta avsnitt inleds därför med en utvärdering av det färdiga användargränssnittet. Efter att användargränssnittet utvärderats kommer också den tekniska implementationen att diskuteras.

Slutligen kommer några problem som uppkommit under projektets gång att presenteras.

4.1 Användargränssnittet

Användargränssnittet och design samt utveckling av detta har varit en stor del av projektet. I detta avsnitt diskuteras det slutgiltiga användargränssnittet. Fokus ligger i avsnittet på hur användargränssnittet ser ut och hur det upplevs vid användande.

4.1.1 Utseende och användbarhet

Ett av grundkraven för användargränssnittet var att det skulle vara estetiskt tilltalande. Vad som är estetiskt tilltalande är till stor del subjektivt men det finns vissa aspekter som krävt extra eftertanke under projektets gång. Informationen skall presenteras tydligt och det skall vara enkelt att hitta det man söker efter. Gränssnittet skall vara anpassat för olika upplösningar och det skall fungera både på en pekskärm och vid användning av tangentbord och mus. Dessutom skall tillgängliga riktlinjer för navigering och utseende[23] följas till den grad det är möjligt.

Stor vikt har lagts vid att användargränssnittet skall vara estetiskt tilltalande och användbart och det återspeglas också i fördelning av tid under projektets gång. Ungefär 40% av den tid som totalt använts för projektet har gått till att utveckla och testa olika gränssnitt för att försöka få fram ett gränssnitt som både ser bra ut och som är enkelt att bruka från användarens perspektiv.

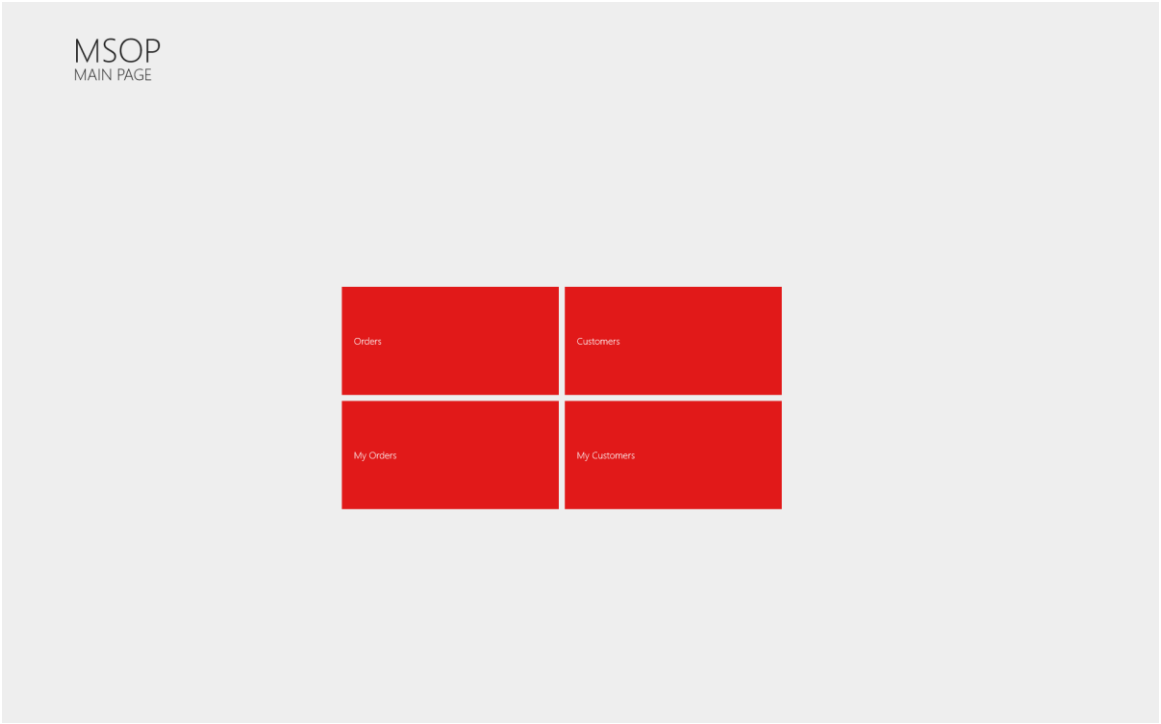
Det slutgiltiga användargränssnittet är enligt mig både estetiskt tilltalande samt lättanvänt. Det har också en högre upplevd prestanda än det redan existerande SharePoint-gränssnittet.

4.1.1.1 Färger

Färgvalet i applikationen har främst styrts av kundföretagets färger i det ursprungliga systemet. Genom att använda liknande färger som det system som redan existerar skapas en enhetlighet över olika system inom företaget. Kundföretagets färgval är också mycket likt Sogeti Sverige ABs färger vilket gör att även kopplingen till Sogeti blir tydlig.

4.1.1.2 Navigering

Navigeringen har ändrats något i de olika versionerna av gränssnittet. I slutändan användes en navigationsmetod som kombinerade de bästa delarna av de tidigare versionerna av gränssnittet. I den slutgiltiga versionen av systemet används en navigeringshierarki som är lik den som Microsoft rekommenderar[7]. Huvudsidan agerar som en hub där användaren kan välja att navigera till olika delar av applikationen. Skillnaden från rekommendationerna är att varje del endast presenteras med en röd ruta istället för dess innehåll. Detta eftersom det under utvecklingen av applikationen inte ansågs att det gick att presentera detta innehåll på ett för användaren nyttigt sätt. Figur 4-1 visar huvudsidan.



Figur 4-1 – Huvudsidan agerar som en hub till de olika delarna av applikationen

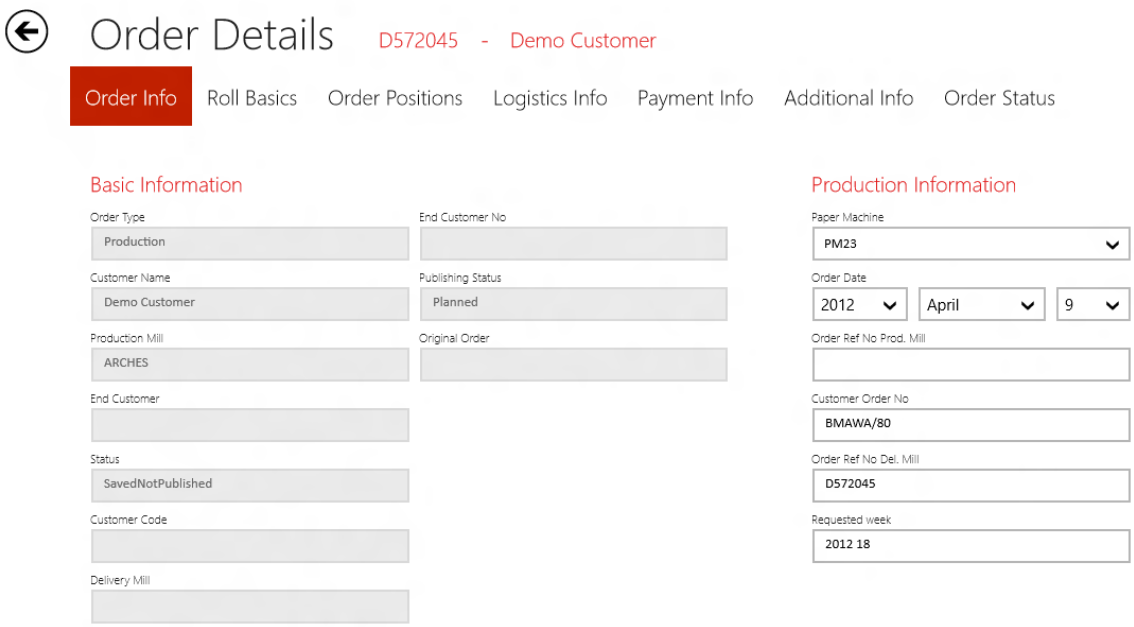
När användaren valt vilken del av applikationen han/hon vill se ytterligare information om visas en lista över denna information (se avsnitt 3.2.3 för en fullständig beskrivning av gränssnittet). Denna listning motsvarar ”Section pages” i Microsofts rekommendationer[7]. När

[7] http://msdn.microsoft.com/en-us/library/windows/apps/hh761500.aspx

användaren väljer att öppna en order visas all information om ordern, motsvarande ”Detail pages” i Microsofts rekommendationer. Genom att riktlinjerna från Microsoft till stor del följts för navigering kommer användaren att enklare kunna navigera i systemet. Applikationen känns också mer enhetlig med operativsystemet och övriga applikationer som användaren har installerade.

4.1.1.3 Uppdelning av information

Förutom att informationen delats upp med hjälp av navigering har den också delats upp inom varje vy. Särskild hänsyn har tagits för att placeringen av element och kontroller skall kännas enhetlig genom applikationen. Ett bra exempel på detta är de olika sidorna i orderdetaljvyn. Figur 4-2 visar en del av order detaljvyn.



Figur 4-2 – Uppdelning av information i orderdetaljvyn

På samtliga sidor i applikationen är titeln och eventuell undertitel placerade på samma ställen. Bakåtknappen återfinns även den på samma ställe i varje vy i applikationen. Marginalerna mellan meny och innehåll hålls konstant liksom marginalerna mellan de olika elementen. Varje sida inom orderdetaljvyn är uppdelad med ett antal underrubriker och dessa skiljs åt med hjälp av en marginal på 80 pixlar. AppBaren används för att användaren skall

kunna addera eller redigera viss information (se avsnitt 3.2.3). Genom att använda denna struktur kan användaren enkelt hitta informationen som han/hon letar efter. Applikationen blir också mer enhetlig, utseendet blir snarlikt oavsett var i applikationen användaren befinner sig.

I de tidigare versionerna av gränssnittet konstruerades gränssnittet inte efter fasta riktlinjer vilket gjorde det både mindre överskådligt samt svårare att arbeta med från ett utvecklingsperspektiv.

Uppdelningen av informationen i applikationen gör det enligt mig enkelt att hitta den information man söker efter och applikationen känns mycket enhetlig.

4.1.1.4 Olika upplösningar

I framtiden kommer olika tillverkare utveckla hårdvara för Windows 8. Denna hårdvara kommer i många fall att skilja sig i både skärmstorlek och upplösning. Det har därför varit viktigt att applikationen skall anpassa sig efter olika upplösningar. Genom att varje del av applikationen konstruerats med detta i åtanke har detta mål uppnåts. Av de val som beskrevs i avsnitt 2.9.3 har valet att visa mer information vid högre upplösning gjorts. Om användaren befinner sig på order-/kundlisningsvyerna kommer fler ordrar/kunder att visas. Om användaren befinner sig i någon av detaljvyerna kommer fler inmatningsfält att visas och användaren kommer inte behöva scrolla lika mycket i sidled. Detta gör att en högre upplösning/större skärm kan utnyttjas på ett bra sätt i applikationen, samtidigt som den hålls användbar på en mindre skärm med lägre upplösning.

I de tidigare versionerna av gränssnittet lades ingen vikt vid att förbereda applikationen för olika upplösningar. Detta visade sig under projektets gång vara ett misstag då mycket extra arbete uppkom vid ombyggnationen av gränssnittet. När man bygger denna typ av applikation är det viktigt att man redan från början planerar för hur man skall anpassa gränssnittet för olika upplösningar och skärmstorlekar.

4.1.1.5 Upplevd prestanda

Inga mätningar på prestandan i applikationen har utförts. Däremot har den upplevda prestandan kunnat jämföras med det ursprungliga systemet. Prestandan i den nya applikationen känns vid användade högre än i den ursprungliga lösningen. Detta beror troligen på att de animationer som återfinns i den nya applikationen ger ett intryck av att någonting sker så snart användaren utfört en operation. I det gamla gränssnittet kan det ibland vara svårt att se om applikationen reagerat eller ej. Den upplevda prestandaökningen beror

troligen också på att mer bearbetning görs direkt i klienten jämfört med SharePoint-lösningen som belastar servern mer.

4.1.2 Jämförelse med den existerande klienten

Under utvecklingen av applikationen har den existerande lösningen använts som utgångspunkt. Då applikationen endast skall agera PoC (Proof of Concept) har dock inte samtliga delar från ursprungssystemet implementerats. Exempel på vad som utelämnats är funktioner för att ladda upp dokument samt specialfall som gäller för olika bruk och ordertyper.

Den nya applikationen arbetar dock mot samma data som den redan existerande klienten och det är möjligt att både visa och redigera de flesta aspekterna av framförallt ordrarna. På grund av tidsbrist är kunddelen något mer begränsad då endast visning av data (ej modifiering) kan utföras i kunddelen.

Den nya klienten är till skillnad från SharePoint-klienten anpassad för att använda på Windws 8-plattan. Kontrollernas storlekar, placering samt navigeringen gör det enkelt att använda applikationen på plattan.

4.2 Den tekniska implementationen

I detta avsnitt kommer implementationen av projektet att utvärderas. Vikten läggs i detta avsnitt på de tekniska aspekterna.

4.2.1 Implementation av gränssnittet

Genom att använda de tillgängliga verktygen har själva implementationen av gränssnittet varit förhållandevis enkel (med avseende på de kontroller som funnits tillgängliga och hur man kunnat anpassa användargränssnittets utseende samt funktion).

Att använda sig utav XAML[14] vid konstruktionen av gränssnittet har minskat utvecklningstiden jämfört med om lösningen skulle ha utvecklats i t.ex. Windows Forms[10]. Detta eftersom större möjligheter för att anpassa applikationens utseende ges med hjälp av XAML samtidigt som man kan utnyttja bindningar (Se avsnitt 3.3.1.3 för en beskrivning av bindningar).

Det som har upptagit mest tid och även varit en av de stora utmaningarna i projektet har varit att konstruera gränssnittet på ett sådant sätt att det är estetiskt tilltalande. Att anpassa gränssnittet efter olika upplösning och presentera informationen på ett överskådligt sätt som är anpassat för en pekskärms-enhet har även detta varit en utmaning. Mycket tid har också

lagts på att få övergångar mellan olika delar av applikationen att se bra ut och att applikationen hela tiden skall kännas responsiv vid användande. Jag är dock mycket nöjd med slutresultatet.

4.2.2 Utvecklingsverktyg och tekniker

Ungefär 40% av tiden i projektet har gått åt till den tekniska implementationen. I detta avsnitt kommer de utvecklingsverktyg samt de tekniker som använts under projektets gång att diskuteras.

4.2.2.1 Visual Studio 11

Under utvecklandet av applikationen har Visual Studio 11 använts som främsta utvecklingsverktyg. Detta verktyg har varit mycket användbart. I Visual Studio kan man förutom att skriva C#.NET kod också enkelt bygga gränssnittet i XAML. Man kan även förhandsgranska applikationens utseende direkt i designern. Detta gör att man inte behöver bygga om applikationen varje gång gränssnittet ändrats.

Under projektets gång uppdaterades versionen av Visual Studio 11 från ”Preview” till ”Beta”. Denna uppgradering kändes mycket vital för projektet. Den version som kallades ”Preview” började efter en tid att spontant låsa sig vilket innebar att man var tvungen att starta om den. I ett skede av projektet låste den sig så ofta som varannan gång man byggde applikationen och skulle testköra den. Detta gjorde att verktyget i praktiken blev väldigt omständigt att använda. Turligt nog släpptes Visual Studio 11 ”Beta” kort därefter och denna version har under projektets gång visat sig vara mycket stabil.

4.2.2.2 Expression Blend

Microsoft Expression Blend[24] är ett verktyg som används för att designa användargränssnitt. I projektet har detta verktyg använts för att anpassa utseendet på vissa kontroller. Med hjälp av verktyget kunde man få en mall över kontrollens original-utseende och sedan anpassa detta efter behov. Expression Blend har varit till stor hjälp under projektet eftersom man annars skulle vara tvungen att själv skriva om hela utseendet på kontrollerna för att ändra endast vissa egenskaper för dess utseende.

4.2.2.3 WinRT

WinRT[12] står för Windows Runtime och är den nya programmeringsmodellen för att utveckla applikationer för Windows. Under projektet har utveckling mot WinRT skett med C#[8] och XAML[14].

Att utveckla mot WinRT har under projektets gång visat sig vara väldigt likt utveckling av WPF-applikationer[11]. Eftersom erfarenhet inom WPF existerade innan projektets start underlättades arbetet avsevärt. WinRT är dock inte färdigutvecklat vilket innebär att vissa vitala funktioner saknats under projektet:

- Ingen DatePicker som ger möjlighet att enkelt välja datum (en egen kontroll för detta har utvecklats).
- Inget bra stöd för att enkelt kunna validera data existerar i skrivande stund.
- Många funktioner tog in strängar som namn på typer istället för typen själv. Formaten på dessa strängar skiljde sig också åt. I senare versioner som släpptes under projektets gång kunde man dock använda typerna istället för strängarna. Även om detta inebar att vissa delar var tvugna att skrivas om underlättade det arbetet därefter.

I det stora hela har det varit en angenäm upplevelse att utveckla med denna nya programmeringsmodell trots dessa begränsningar. Att vissa funktioner saknas i WinRT kan dock vara värt att tänka på vid framtida utveckling.

4.2.2.4 MVVM

Användadet av MVVM har underlättat arbetet och förkortat utvecklingstiden. Det har också skapat förutsättningar för att i framtiden kunna byta ut vyerna mot andra vyer och därmed enkelt kunna förändra utseendet på programmet.

4.2.2.5 Befintliga WCF-tjänster

Att de befintliga WCF-tjänsterna existerat har varit en förutsättning för att kunna utföra projektet. Att använda dessa tjänster har varit förvånadsvärt enkelt när man väl förstått principerna bakom asynkrona anrop. Genom att använda tjänsterna har gränssnittet kunnat byggas på redan befintlig funktionalitet utan att det befintliga systemet behövt förändras.

Något som blivit tydligt under projektets gång är att det befintliga systemets uppdelning gör att det är enkelt att utveckla ytterligare klienter som arbetar mot det centrala systemet.

4.3 Tekniska Problem

Under projektet har några problem dykt upp. Dessa problem kommer att presenteras i detta avsnitt. Problemen har främst varit av teknisk karaktär.

4.3.1 Versioner av Windows 8

Några av de problem som uppstått har berott på att Windows 8 hela tiden varit under utveckling och att det därför uppdaterats under projektets gång. Vissa uppdateringar innebar att kod var tvungen att skrivas om eftersom funktionalitet hos vissa funktioner förändrades. Vid ett tillfälle började plattan bete sig underligt och registrera tryckningar som ej utförts. Det visade sig bero på att drivrutinerna till skärmen uppdaterats och en ny version av skärmens firmware var tvungen att installeras för att plattan skulle fungera som det var tänkt igen.

4.3.2 Anslutning till VPN

De existerande tjänsterna och tillgång till databasen krävde från början att man var ansluten till Sogetis nätverk eller att man använde VPN till detta (Virtual Private Network). VPN möjliggör att man säkert kan ansluta till ett nätverk och utnyttja dess domänspecifika resurser även om man egentligen är ansluten till ett annat nätverk. I Windows 8 kunde man utan problem ansluta till VPN och använda WCF-tjänsten från testklienten som följer med Visual Studio. Det visade sig dock att det inte var möjligt att ansluta till VPN från applikationer som utvecklats efter designspåret Metro.

Efter mycket efterforskning på området beslutades att en kopia av tjänsten och databasen skulle installeras på Windows 8-plattan och att applikationen istället skulle ansluta till denna. Eftersom tjänsten inte körs som en Metro-applikation är det nu möjligt att ansluta till den ursprungliga databasen genom att peka om databasens adress från plattan. Anslutning till VPN från Metro-applikationer är i dagsläget fortfarande inte möjligt, men förmodligen kommer stöd för detta i en framtida version av Windows 8.

4.3.3 Bugg i ComboBox när den används i UserControls

Något som tog tid att reda ut var en bugg som verkar existera när en ComboBox används i en UserControl (Se kontrollen BoxTile i avsnitt 3.3.1.1). Av någon anledning så kördes kontrollens ”SelectionChanged”-event två gånger varannan gång man ändrade värde. Den andra gången skickades värdet null som SelectedValue vilket gjorde att inget alternativ valdes. Detta löstes genom att i SelectionChanged-eventet lägga till en kontroll för om värdet var null. Denna lösning visas i Figur 4-3.

```
private void combo_SelectionChanged_1(object sender, SelectionChangedEventArgs e)
{

    if (combo.SelectedValue != null)
    {
        SetValue(SelectedValueProperty, combo.SelectedValue);
        SetValue(SelectedItemProperty, combo.SelectedItem);
    }
    else
    {
        combo.SelectedValue = GetValue(SelectedValueProperty);
    }

}
```

Figur 4-3 – Lösning på ComboBox-buggen i UserControls

Om värdet som inkommer inte är null sätts ComoBoxens värde till det valda värdet. Om värdet som inkommer är null sätts ComboBoxens värde till det tidigare valda värdet. På så sätt kommer null-värdet att ignoreras.

4.4 Surfplattans för- och nackdelar

Surfplattans möjligheter och begränsningar kommer att bli uppenbara först när den använts i en verklig produktionsmiljö. Några för- och nackdelar med plattformen går dock att utröna redan nu.

4.4.1 Fördelar

- Mobilitet - Formfaktorn gör att surfplattan är enkel att ta med sig vid t.ex. resor. Det är fullt möjligt att transportera plattan i en väska eller ryggsäck.
- Enkelhet - Inga tillbehör behövs för att kunna interagera med plattan eftersom all input kan göras genom touch-gränssnittet.
- Ny spännande teknik - Att använda ny teknik lockar många.
- Designspåret Metro[9] – Vackra och responsiva applikationer är roligare att använda.

4.4.2 Nackdelar

- Skärmstorlek - Mindre information får plats på skärmen.
- Batteritid - Utvecklingsplattans batteritid är något låg vid användande (2-3 timmar beroende på hur den används).
- Multitasking - Endast två applikationer kan användas samtidigt.
- Ny teknik - Det kan upplevas omständigt att lära sig hur man använder plattan.

- Skärmtangentbord - Om man ska skriva mycket är det lämpligt att koppla till ett externt tangentbord, vilket påverkar mobiliteten.
- Precision - Att använda pekskärmen istället för mus gör att sämre precision uppnås.
- Ej färdigt - Windows 8 är fortfarande i ett tidigt stadie, buggar existerar och vissa funktioner saknas.

Enligt mig är mobiliteten den viktigaste anledningen till att använda en surfplatta. Genom att man har möjlighet att enkelt ta med sig plattan kan man snabbt komma åt information utanför kontorets väggar.

4.5 Sammanfattad utvärdering av projektet

4.5.1 Gränssnittet

Den slutgiltiga versionen av gränssnittet har mött de krav som uppdragsgivaren Sogeti har ställt på det. Jag är mycket nöjd med det slutgiltiga gränssnittet och anser det vara både estetiskt tilltalande samt enkelt att använda. Systemet har presenterats för utvecklare i det ursprungliga projektet och har fått ett mycket positivt mottagande.

4.5.2 Implementation

Den slutgiltiga implementationen är enligt mig bra organiserad och överskådligheten i koden är god. Detta gör att en möjlig framtida utveckling av projektet enklare kan utföras. Implementationen av projektet har visat att nya klienter relativt enkelt kan utvecklas baserat på det befintliga systemets funktionalitet.

4.5.3 Slutprodukten

Den slutgiltiga produkten från projektet är en Windows 8-applikation som ger användaren möjlighet att hantera ordrar och kunder i det centrala marknadssystem som Sogeti för närvarande utvecklar.

Den slutgiltiga produkten har nått upp till de mål och förväntningar som jag satt på mig själv och även Sogeti är nöjda med slutresultatet. Slutprodukten ger enligt mig ett bra exempel på hur man kan använda en Windows 8-surfplatta i en affärs-orienterad applikation.

5 Slutsats

Detta avsnitt syftar till att utvärdera projektet som helhet. Avsnittet är tänkt att ge en bild över de erfarenheter som skaffats under projektets gång. Några idéer för framtida utveckling av produkten kommer också att ges.

5.1 Produkten

Målet med projektet har varit att utveckla en applikation för en Windows 8-surfplatta för att visa vilka möjligheter och begränsningar som finns med denna plattform.

Det slutgiltiga resultatet har nått upp till de mål och förväntningar som jag satt på mig själv och även Sogeti är nöjda med slutresultatet. Projektet kommer i framtiden att presenteras för Sogetis kund och det skall bli spännande att få se deras reaktion på lösningen.

Genom att utveckla produkten har jag fått utnyttja och framförallt utveckla de erfarenheter som jag skaffat mig under mina studier. Jag har också skaffat mig goda erfarenheter inom en ny och spännande plattform.

5.2 Projektvärdering

Windows 8 är en ny plattform för utveckling av applikationer för både traditionella desktop-datorer och kommande pekplattor med operativsystemet. Det har varit mycket intressant och lärorikt att utveckla en applikation för plattformen. Även om det är en ny plattform har jag haft stor nytta av att tidigare ha arbetat med framförallt C#.NET och WPF under mina studier.

Eftersom jag är en mer tekniskt lagd person snarare än en designer har det varit extra utmanande för mig att ta mig an de estetiska aspekterna av projektet, där mycket fokus legat.

Jag anser att projektet har legat på en bra nivå med avseende på mina tidigare erfarenheter och den mängd arbete som jag behövt utföra för att slutföra projektet.

5.2.1 Tidsåtgång

Tidsåtgången för projektet har visat sig vara högre än vad jag från början förväntade mig. En stor del av tiden har lagts på det estetiska.

Jag började projektet med att direkt utveckla ett användargränssnitt i den nya miljön för att utforska vilka möjligheter som fanns på plattformen. I efterhand har jag insett att det nog hade

varit bättre att utveckla gränssnittet på papper och försöka utvärdera de olika valen med hjälp av skisser istället för att direkt dyka in i implementationsvärden.

Eftersom det är en ny plattform har mycket tid också gått till att lära sig de nyheter som finns och mängden information för att direkt kunna lösa problem som uppstått har varit begränsad. Under projektets gång har dock mängden information runt plattformen ökat kraftigt.

Mina erfarenheter om tidsåtgången kommer förhoppningsvis att hjälpa mig vid framtida tidsuppskattning av projekt.

5.2.2 Kontakt med uppdragsgivaren

Eftersom större delen av projektet utförts i Sogeti Karlstads lokaler har kontinuerlig kontakt med uppdragsgivaren kunnat hållas. Jag har hela tiden stämt av mina tankar och idéer med handledaren eller annan personal som funnits tillgänglig.

Genom att ha haft tillgång till kunskapen hos Sogetis anställda under projektets gång har jag kunnat utveckla mitt tankesätt och skaffat mig ytterligare erfarenhet inom området systemutveckling. Att diskutera idéer tillsammans med erfaret folk gör att man får en annan förståelse än att läsa om ett ämne på internet eller i en bok.

5.3 Framtida utveckling

Även om jag är nöjd med slutprodukten med avseende på den tid som funnits tillgänglig för projektet finns rum för förbättring.

Det första steget i en framtida utveckling är att möjliggöra modifiering av kund-data från applikationen. Utöver detta bör en vidareutveckling av applikationen också innebära att man tar hänsyn till alla de specialfall som kan uppkomma vid vissa kombinationer av ordertyper och produktionsbruk i det existerande systemet, något som utelämnats från den nuvarande lösningen på grund av tidsbrist.

Eftersom mycket känslig data hanteras kan det också vara en bra idé att undersöka vilka extra säkerhetsåtgärder som krävs för att kunna säkra den information som skickas över nätverket om man i framtiden skulle vilja använda sig av 3G-nätet som kommunikationskanal.

5.4 Sammanfattning

Under projektet har en applikation anpassad för en Windows 8-surfplatta designats och implementerats. De mål som från början satts upp för projektet har uppnåtts. Uppdragsgivaren Sogeti liksom jag själv är nöjda med både projektets omfattning samt slutresultatet.

Under projektets gång har mycket erfarenheter inom systemutveckling i allmänhet och Windows 8 i synnerhet skaffats. Jag har också fått utnyttja mycket av det som jag lärt mig under mina tidigare studier inom området datavetenskap.

I det stora hela anser jag projektet vara mycket lyckat då det varit både roligt, stimulerande och lärorikt samtidigt som ett gott resultat uppnåtts!

6 Referenser

1. Sogeti, ”Sogeti,” [Online]. Available: http://www.sogeti.se.
2. Microsoft, ”Biztalk,” [Online]. Available: http://www.microsoft.com/biztalk/en/us/default.aspx.
3. Microsoft, ”SQL-server,” [Online]. Available: http://www.microsoft.com/sqlserver/en/us/default.aspx.
4. Microsoft, ”ADO.NET Entity Framework,” [Online]. Available: http://msdn.microsoft.com/en-us/library/aa697427(v=vs.80).aspx.
5. Microsoft, ”WCF,” [Online]. Available: http://msdn.microsoft.com/en-us/netframework/aa663324.
6. Microsoft, ”SharePoint,” [Online]. Available: http://sharepoint.microsoft.com/sv-se/Sidor/default.aspx.
7. Microsoft, ”Windows 8,” [Online]. Available: http://windows.microsoft.com/sv-SE/windows-8/preview.
8. Microsoft, ”C#.NET,” [Online]. Available: http://msdn.microsoft.com/en-us/vstudio/hh388566.
9. Microsoft, ”Metro,” [Online]. Available: http://www.microsoft.com/design/toolbox/tutorials/windows-phone-7/metro/.
10. Microsoft, ”Windows Forms,” [Online]. Available: http://msdn.microsoft.com/en-us/library/dd30h2yb.aspx.
11. Microsoft, ”WPF,” [Online]. Available: http://msdn.microsoft.com/en-us/library/ms754130.aspx.
12. Microsoft, ”Windows Runtime,” [Online]. Available: http://msdn.microsoft.com/en-us/library/windows/apps/hh464942.aspx.
13. Microsoft, ”Visual Studio 11 Developer Preview,” [Online]. Available: http://msdn.microsoft.com/en-us/vstudio/hh127353.
14. Microsoft, ”XAML,” [Online]. Available: http://msdn.microsoft.com/en-us/library/ms752059.aspx.

15. Microsoft, ”Build” [Online]. Available: http://channel9.msdn.com/Events/BUILD/BUILD2011/
16. Samsung, ”Series 7 Slate” [Online]. Available: http://www.samsung.com/us/computer/platta-pcs/XE700T1A-A03US
17. Microsoft, ”Snapped view and fill view” [Online]. Available: http://msdn.microsoft.com/en-us/library/windows/apps/hh465371.aspx
18. Microsoft, ”Touch interaction design” [Online]. Available: http://msdn.microsoft.com/en-us/library/windows/apps/hh465415.aspx
19. Microsoft, "Frame class" [Online], Available (29/3-12) : http://msdn.microsoft.com/en-us/library/windows/apps/windows.ui.xaml.controls.frame
20. Microsoft, "Guidelines for Semantic Zoom[Online], Available (29/3-12) : http://msdn.microsoft.com/en-us/library/windows/apps/hh465319.aspx
21. Josh Smith, "WPF Apps With The Model-View-ViewModel design Pattern" [Online], Available (29/3-12) : http://msdn.microsoft.com/en-us/magazine/dd419663.aspx
22. Microsoft, "Navigation design for Metro style apps "[Online], Available(30/3-12) : http://msdn.microsoft.com/en-us/library/windows/apps/hh761500.aspx
23. Microsoft, “UX Guidelines for Metro style apps” http://msdn.microsoft.com/en-us/library/windows/apps/hh465424.aspx
24. Microsoft, “Expression Blend”, http://www.microsoft.com/expression/products/Blend_Overview.aspx
25. Wikipedia, “Integrated development environment” http://en.wikipedia.org/wiki/Integrated_development_environment