

Katholieke Hogeschool Sint-Lieven

Departement Industrieel Ingenieur

Afdeling Elektronica optie ICT

Gebroeders Desmetstraat 1, 9000 Gent

Ontwikkeling van een .NET toepassing voor de integratie van vlootklanten en dealers van vrachtwagenonderdelen

Eindverhandeling ingediend tot het behalen van de graad van
Master of Science in de Industriële Wetenschappen (optie Elektronica - ICT)
en aangeboden door: Joost Roelandts

Promotor: dr. ir. Annemie Vorstermans

Copromotor: ing. Joris Maervoet

Externe Promotor: ing. Frank Kint

Copyright 2005, 2006 Joost Roelandts

Hierbij geeft de auteur van dit eindwerk aan het bestuur van het departement industrieel ingenieur van KaHo Sint-Lieven de uitdrukkelijke toestemming om dit werk in bruikleen af te staan aan eender welk persoon, organisatie of firma, het ten dienste te stellen van de studenten en het geheel of gedeeltelijk te kopiëren.

Dankwoord

Ik bedank in eerste instantie ing. Frank Kint (manager AAS-ICT Solutions), ing. Ronny De Ridder (programmeur AAS-ICT Solutions) en lic. Nico Spruyt (programmeur AAS-ICT Solutions) voor hun tijd en moeite om mijn vragen op te lossen en mij te begeleiden doorheen het volledige proces. Ik ben ing. Frank Kint eveneens zeer dankbaar omdat hij mij de kans heeft gegeven om via mijn eindwerk kennis te maken met het bedrijfsleven en de job van software ingenieur. Daarnaast bedank ik mijn promotoren van KaHo Sint-Lieven dr. ir. Annemie Vorstermans en ing. Joris Maervoet voor hun begeleiding.

Verder bedank ik Peter Van Den Heurck (Algemeen Directeur Truck Trading Limburg, Houthalen), Carlo Reeckmans (Magazijnchef Truck Trading Limburg, Houthalen), Tom Coenen (Magazijnier Truck Trading Limburg, Houthalen) en Chris Verschueren (Magazijnchef Garage Van den Keybus, Sint-Niklaas) voor het vrijmaken van hun kostbare tijd. Zonder hun hulp zou het voor mij onmogelijk geweest zijn de eisen van het project op te stellen en het resultaat te evalueren.

Verder zou ik Norma Schelstraete, mevr. Kint en dr. ir. Annemie Vorstermans willen bedanken voor het lezen en verbeteren van mijn boek. Hun hulp is de kwaliteit en leesbaarheid van dit werk zeker ten goede gekomen. Daarnaast wil ik ook Norma, Dries Roelandts, Wouter Vanderbeke, Jurriaan Persyn en Lieselot De Meyer bedanken voor het testen van mijn applicatie en Norma voor het vertalen van de resource files.

Ten slotte bedank ik mijn ouders omdat ze altijd klaar stonden om op alle mogelijke vlakken te helpen en zonder wiens financiële steun dit alles niet mogelijk zou geweest zijn. Als laatste, maar zeker niet als minst belangrijke, bedank ik Lieselot voor haar morele steun en het terugvinden van mijn enthousiasme in de momenten dat ik het even kwijt was.

Abstract (Nederlands)

ROELANDTS JOOST

Brouwerijstraat 28, 9920 Lovendegem

Tel: 09/371.68.65

Ontwikkeling van een .NET toepassing voor de integratie van vlootklanten en dealers van vrachtwagenonderdelen

Dit eindwerk bestaat in eerste instantie uit de analyse van het proces waarbij vlootklanten vrachtwagenonderdelen bestellen bij hun dealers en hoe de afhandeling van deze bestellingen gebeurt. In tweede instantie zal dit proces geautomatiseerd worden aan de hand van een toepassing geschreven in .NET technologie. Ten slotte zal deze toepassing ook in de praktijk getest worden bij de klanten. De toepassing zal opgesplitst worden in twee grote delen: een deel dat zal gebruikt worden door de klanten en een deel door de dealers. Met het eerste deel zal de klant aan de hand van een ASP.NET webtoepassing (zowel op desktops als op mobiele toestellen) bij zijn dealer onderdelen kunnen bestellen, een geschiedenis van zijn bestellingen kunnen bekijken en hulp aan zijn dealer vragen. Daarnaast zal de dealer beschikken over een Windows toepassing waarmee hij onmiddellijk verwittigd wordt als de klant zijn hulp nodig heeft en/of als de klant een bestelling geplaatst heeft. Daarnaast kan de dealer met deze toepassing ook informatie over zijn klanten en promoties beheren, nagaan welke onderdelen te duur zijn volgens de klanten, . . . Deze toepassing zal gebruik maken van een SQL-server waarin alle data zal worden gecentraliseerd. Omdat er een directe link moet zijn tussen de klanten, de dealers en de databank zal er veel aandacht besteed worden aan de manier waarop alle communicatie en uitwisseling van data zal verlopen. Daarbij zal eveneens rekening moeten worden gehouden met aspecten van beveiliging en privacy en met een efficiënte identificatie van de klanten.

Trefwoorden: Visual Basic .NET - ASP.NET - SQL-server - Databeveiliging en privacy - Datacommunicatie

Abstract (English)

ROELANDTS JOOST

Brouwerijstraat 28, 9920 Lovendegem

Tel: 09/371.68.65

Development of a .NET application for the integration of fleet clients and automotive dealers

The object of this thesis is the automation of the order process between fleet clients and dealers of truck parts. The first part consists of the analysis of how the clients order their spare parts and how the dealers handle these orders. In the second part a .NET application will be developed to automate and facilitate this process. Finally I will test this application in the field. The application will be divided into two main parts: a part for the clients and a part for the dealers. The client will be able to order spare parts, check his order history and get help from his dealer using an ASP.NET web application. This application will be available for desktops as well as for mobile devices. The dealer will have a Windows application that will react immediately when the client needs his help or has placed an order. This part of the tool will also give the dealer the possibility to manage information about his clients, manage promotions, check which parts are too expensive for the clients, . . . The application will use a SQL-server which will centralise all data. Because of the direct link between the clients, the dealers and the database considerable attention will be spent to ways of communication and data exchange. Consequently, aspects concerning data security, privacy and authentication will also play an important role in this thesis.

References: Visual Basic .NET - ASP.NET - SQL-server - Data security and privacy - Data communication

Inhoudsopgave

1	Inleiding	1
1.1	Bespreking bedrijf	1
1.2	Opdrachtoomschrijving	1
1.3	Technologische eisen	2
1.4	Opbouw van dit eindwerk	3
2	Analyse	4
2.1	Eisen AAS en beperkingen	4
2.2	Eisen dealers	7
2.2.1	Zoekmogelijkheden	8
2.2.2	Bestellingen	11
2.2.3	Gebruikers	14
2.3	Eisen vlootklanten	16
3	Ontwerp	18
3.1	Ontwerpbeslissingen en verantwoording	18
3.1.1	Globalization en localization	18
3.1.2	Internet toepassing	20
3.1.3	Communicatie	22
3.1.4	Zoekmachine	28
3.1.5	Beveiliging	29
3.1.6	Drie lagen model	31
3.1.7	Mobiel en niet-mobiel	31
3.1.8	Real-time	33
3.1.9	Consistentie centrale database	33
3.1.10	Opmaak	34
3.2	UML modellering	34
3.2.1	Activiteitendiagramma's	34
3.2.2	Klassediagramma	35
3.2.3	Sequentiediagramma's	35
3.3	Ontwerp grafische user interface	35

3.4	Ontwerp database	44
4	Gebruikte technologieën	47
4.1	Servers	47
4.1.1	SQL Server 2000	47
4.1.2	Internet Information Services	47
4.1.3	Citrix Presentation Server	48
4.2	Softwarepakketten	49
4.2.1	Visual Studio .NET 2003	49
4.2.2	Microsoft Visio 2003 Professional	49
4.2.3	Macromedia RoboHelp X5	50
4.2.4	Macromedia Captivate en Wink	50
4.2.5	Openwave V7 Simulator	51
4.2.6	WinShell en MikTeX	51
4.2.7	IconCool Studio v1.6	51
4.2.8	Microsoft .NET framework 1.1	52
4.3	Gebruikte talen	52
4.3.1	Visual Basic .NET	52
4.3.2	T-SQL	53
4.3.3	ADO.NET	54
4.3.4	Microsoft ASP.NET	55
4.3.5	XML en SOAP	55
4.3.6	Cascading Style Sheets	56
4.3.7	SSL en HTTPS	56
4.3.8	UML	57
4.3.9	LaTeX	57
5	Implementatie	59
5.1	Onderdelen	59
5.1.1	ASP.NET Web Application	60
5.1.2	ASP.NET Mobile Web Application	61
5.1.3	Windows Application en Installer	62
5.1.4	Console Application en Installer	62
5.1.5	XML Webservice	64
5.1.6	Admin tool	64
5.2	Beveiliging	64
5.2.1	Paswoordgenerator	64
5.2.2	Beveiliging webservice	66
5.3	Performance	68
5.4	Link met DMS	69
5.4.1	Bestellingen afhandelen	69

5.4.2	Gebruikers toevoegen	71
5.4.3	Prijsberekening	71
5.5	Geldigheid BTW-nummer	72
5.6	Wederzijdse uitsluiting	72
5.6.1	Behandelen acties	73
5.6.2	Lezen en schrijven van foto's op de server	73
5.7	Schrijven naar de server	74
5.8	Globalization en localization	76
5.9	Mobiel en niet-mobiel	78
5.10	Loggen	79
5.11	Wettelijke bepalingen	80
5.12	Real-time	81
5.13	Configuratie	85
5.14	Testen	88
5.15	Aanpassingen na tests	88
6	Mogelijke uitbreidingen	90
6.1	Extra mogelijkheden	90
6.2	Opmerkingen gebruikers	91
6.3	Aanvullingen AAS	92
7	Besluit	93
A	Structuur DAF-bestand	95
B	Stylesheet ASP.NET toepassing	96
C	Poster eindwerk	100
D	Basispagina klanttoepassing	102
E	Inhoud bijgevoegde CD-ROM	108
	Lijst van figuren	109
	Lijst van listings	110
	Bibliografie	114

Hoofdstuk 1

Inleiding

1.1 Bespreking bedrijf

Dit eindwerk werd uitgevoerd in opdracht van AAS-ICT solutions. Dit bedrijf is een klein software bedrijf dat gespecialiseerd is in software voor dealers van vrachtwagenonderdelen en voor vrachtwagengarages. Naast het ontwikkelen van software op maat staat AAS eveneens in voor netwerkbeheer, automatiseren van magazijnbeheer aan de hand van barcoding,... [11]

Alle dealers die klant zijn van AAS gebruiken het Dealer Management System (DMS) dat door AAS oorspronkelijk uitbesteed werd aan Dolmen. Momenteel is het beheer van dit programma volledig in handen van AAS en wordt er bovendien gewerkt aan een vertaling en vernieuwing van dit systeem naar .NET (het huidige systeem werd geschreven in Visual Basic 6). Dit systeem wordt door de dealers gebruikt voor ongeveer alle geïnformatiseerde taken binnen hun bedrijf. Hierbij horen zowel de administratie, de boekhouding, het inventariseren van onderdelen, het bijhouden van klantgegevens en gegevens van werknemers,...

Aangezien deze taken reeds verwerkt zitten in deze DMS is het niet de bedoeling dat dit eindwerk deze zal overnemen. De toepassing die dit eindwerk voor ogen heeft zal daarentegen moeten aansluiten bij dit systeem en er voor een deel in verwerkt worden.

1.2 Opdrachtomschrijving

Het doel van dit eindwerk is een tool te schrijven voor de integratie van vlootklanten bij hun dealer. Vlootklanten bezitten een groot aantal DAF-

vrachtwagens en onderhouden deze in een eigen garage. Deze garages hebben frequent onderdelen nodig en deze onderdelen worden geleverd door de dealers. Elke klant heeft een eigen dealer in de buurt bij wie hij onderdelen bestelt. De dealers zelf kopen hun onderdelen aan bij de DAF-producent in Eindhoven.

Voor het bestellen, de afhaling en de levering van de onderdelen wordt momenteel volgend systeem gehanteerd: wanneer een klant een onderdeel nodig heeft, telefoneert hij naar zijn dealer die het juiste onderdeel opzoekt. Dit onderdeel of deze onderdelen worden bij de klant geleverd op de gevraagde datum of de klant haalt dit onderdeel op bij de dealer. Dit gebeurt onmiddellijk als het onderdeel in stock is en de klant het dringend nodig heeft of op een ander tijdstip. Dit eindwerk stelt zich tot doel om een tool te schrijven die dit proces automatiseert en efficiënter maakt.

Deze tool zal de klanten in staat stellen om op een eenvoudige manier onderdelen te bestellen bij hun dealer, te bepalen hoe en wanneer de onderdelen zullen geleverd of afgehaald worden en een geschiedenis bij te houden van hun bestellingen. Afhankelijk van de voorkeur van de klanten zal deze tool kunnen gebruikt worden op een pc, een PocketPC of een ander mobiel toestel.

Daarnaast zal deze tool het voor de dealer eenvoudiger maken om bestellingen op te volgen, promoties zichtbaar te maken voor de klanten, het koopgedrag van de klanten bij te houden en klanten toe te voegen en te verwijderen. De meer specifieke eisen worden gedetailleerd beschreven in hoofdstuk 2.

1.3 Technologische eisen

Vanuit AAS werd de eis gesteld om dit project uit te voeren in .NET technologie. Hiervoor zal gebruikt gemaakt worden van de Microsoft Visual Studio .NET 2003 programmeeromgeving en zal gekozen worden voor de VB.NET programmeertaal. Daarnaast zal eveneens gebruik gemaakt worden van T-SQL, ADO.NET en ASP.NET (alle gebruikte technologieën en talen worden besproken in hoofdstuk 4).

Verder wordt bij AAS en bij de dealers gebruik gemaakt van SQL Server 2000 en bij AAS wordt eveneens gebruik gemaakt van Internet Information Services (IIS). Ten slotte maakt AAS ook gebruik van Macromedia RoboHelp X5 en Macromedia Captivate voor het maken van help functies. In de planning van dit eindwerk is ook tijd vrijgemaakt om de toepassing te

voorzien van een help functie. Hierbij zal dan ook gebruik gemaakt worden van deze tools.

1.4 Opbouw van dit eindwerk

De opbouw van dit eindwerk is gebaseerd op de standaard fasen die doorlopen worden tijdens het proces van softwareontwikkeling. In eerste instantie zal een analyse uitgevoerd en besproken worden van het proces dat moet vereenvoudigd en geïnformatiseerd worden (zie hoofdstuk 2). Vanuit deze analyse zal vervolgens in hoofdstuk 3 een ontwerp van de toepassing opgesteld worden. In dit onderdeel zal zowel de opbouw van de database, de opbouw van de grafische user interface, als de opbouw van de achterliggende logica besproken worden. Daarnaast zullen in dit hoofdstuk ook de ontwerpbeslissingen besproken worden samen met hun verantwoording.

De volgende hoofdstukken zullen de implementatiefase bespreken. In hoofdstuk 4 zullen de gebruikte technologieën besproken worden en in hoofdstuk 5 zullen enkele punten behandeld worden die belangrijk waren bij de implementatie van het ontwerp. Hier worden onder andere enkele codevoorbeelden getoond, besproken hoe rekening gehouden werd met de performantie van de toepassing, . . . Hoofdstuk 6 bespreekt wat er nog verbeterd kan worden aan de toepassing. Dit zijn zowel zaken die de klanten aanhaalden bij de evaluatie als zaken die nog een meerwaarde aan de toepassing zouden kunnen geven. Tot slot volgt een algemeen besluit waarin onder andere besproken wordt wat de klanten van het resultaat vonden.

Hoofdstuk 2

Analyse

In dit hoofdstuk wordt de analyse van het huidig communicatieproces tussen de vlootklanten en dealers besproken. Deze analyse is gebaseerd op gesprekken met de dealers. Hieruit werden de belangrijkste eisen opgesteld waaraan het softwarepakket moet voldoen dat dit proces moet vereenvoudigen en automatiseren. Deze eisen werden gecategoriseerd volgens de personen die ze formuleerden en zullen uiteindelijk leiden tot het ontwerp van de toepassing. Dit ontwerp zal besproken worden in het volgende hoofdstuk.

2.1 Eisen AAS en beperkingen

Als eerste deel van de analyse van het probleem formuleren we de eisen die de opdrachtgever (AAS-ICT solutions) aan het project stelde:

- In eerste instantie moet de tool geschreven worden in en gebruik maken van de talen en technologieën die bij AAS momenteel gebruikt worden. Daarnaast moet de tool natuurlijk gebruik maken van de bestaande data-infrastructuur bij de dealers. Deze eisen werden al vermeld in de inleiding.
- Een ander belangrijk punt was dat de tool initieel bedoeld was als een volledig web gebaseerde toepassing. Dit zorgt er namelijk voor dat de implementatie bij de klant geen extra installatieprocedures vereist (de klant heeft dus enkel een toestel nodig met een internetverbinding en een standaard internetbrowser) en dat het onderhoud een stuk eenvoudiger wordt. Aanpassingen moeten namelijk slechts eenmalig op de server uitgevoerd worden. In de loop van de analyse hebben we echter moeten besluiten dat dit niet de beste oplossing was. In sectie 3.1.2 (p. 20) en sectie 3.1.3 (pp. 22-26) wordt hier verder op ingegaan.

- Voor de klant zou het handig zijn als hij zowel op een gewone pc als op een mobiel toestel gebruik kan maken van de tool. Het gebruik van een ASP.NET Mobile Web Application lijkt hier dus de beste oplossing. Dit soort toepassing kan namelijk gebruikt worden op alle toestellen met een internetbrowser (zowel gewone pc's als PocketPC's, SmartPhones, ...). In het ontwerpproces werd beslist om naast deze mobiele toepassing ook te voorzien in een versie van de toepassing voor niet-mobiele toestellen. Dat zal immers de gebruiksvriendelijkheid en het esthetische aspect van dit deel van de toepassing ten goede komen. Dit wordt verder besproken in sectie 3.1.7 (pp. 31-32).
- De tool voor de vlootklanten zal centraal beheerd worden door AAS op hun Internet Information Services (IIS) server. Aangezien de dealers zelf enkel een SQL server hebben zal de overdracht van gegevens uit deze database naar de IIS van AAS op een andere manier moeten verlopen. De dealers willen namelijk geen IIS Server aangezien dit voor hen een onnodige financiële last betekent. Hoe de communicatie zal verlopen tussen de verschillende onderdelen van de toepassing en wat dus de algemene opbouw van het systeem zal zijn wordt besproken in sectie 3.1.3 (pp. 22-26).
- Elke dealer heeft een SQL server met een database die ontworpen werd door AAS en waarin alle klanten, trucks, onderdelen, werkorders, werknemers, lonen, kortingen, ... staan. Uit deze databases zullen dus de gegevens gehaald worden van de onderdelen die de dealer verkoopt. Elke dealer verkoopt gemiddeld ongeveer 25000 verschillende onderdelen. Voor elk van deze onderdelen is er, naast algemene gegevens, een Franse en Nederlandse naam en beschrijving beschikbaar in de database van de dealers.
- De applicatie zal standaard in het Engels geschreven worden maar moet klaar zijn om eenvoudig te kunnen vertalen. De tekststrings worden dus gescheiden gehouden van de code. Om dit te verwezenlijken zijn er verschillende methodes. Welke methode gekozen zal worden en waarom wordt besproken in sectie 3.1.1 (pp. 18-20).
- AAS maakt gebruik van het toolpakket Infragistics. Dit pakket kan dus eventueel ook gebruikt worden voor dit eindwerk. We streven er echter naar om, indien mogelijk, geen gebruik te maken van Infragistics om te vermijden dat er later problemen ontstaan met compatibiliteit.
- Het moet voor de klant mogelijk zijn om een geschiedenis te bekijken van zijn bestellingen. Dit maakt de toepassing immers een stuk

interessanter voor de klanten.

- Het is noodzakelijk dat elke klant enkel onderdelen kan bestellen bij zijn dealer (en dus niet bij een andere dealer). In de database zal er dus voorzien worden in een link tussen de dealers en hun klanten.
- Om de prijs van onderdelen te weten is het nodig om de korting te berekenen. Deze korting wordt bepaald aan de hand van de klant, het artikel, het soort bestelling, . . . Omdat dit te complex is om in dit eindwerk te verwerken, zal een klasse voorzien worden met een lege functie die instaat voor deze berekening. Deze methode zal later ingevuld worden door AAS. Hiervoor zullen bepaalde voorzieningen genomen worden in de database en in het ontwerp van het programma.
- Vanuit AAS werd ook gevraagd om de implementatiefase op tijd af te ronden zodat de tool ook effectief kan ingezet en getest worden bij de klanten. De fases van het proces die na de afwerking van de tool komen nemen namelijk een groot deel van de tijd in van het software-ontwikkelingsproces. Bijgevolg is het interessant om ook deze kant te verwerken in dit eindwerk.
- De functie van de tool eindigt bij de bestelling. Eens de bestelling geplaatst en bevestigd is, wordt de facturatie en opvolging gedaan door het Dealer Management System van AAS. De tool moet dus bij de bestelling een file of trigger veroorzaken die kan gelezen worden door deze DMS.

Naast deze eisen zijn er een aantal feiten of gegevens die we ook als eisen zouden kunnen formuleren. Het gaat om zaken die niet gewijzigd kunnen worden en waarmee dus rekening moet gehouden worden bij ontwerpbeslissingen.

- Elke klant heeft een eigen dealer. Deze dealer wil niet dat de klant zonder meer zijn stock kan inkijken. Hij wil namelijk niet dat de klant onmiddellijk naar een andere dealer gaat wanneer het onderdeel dat hij zoekt niet in stock is. Als de klant een onderdeel nodig heeft dat de dealer niet in stock heeft dan wordt dit besteld bij DAF Eindhoven. Wanneer de klant dit onderdeel dringend nodig heeft (Express Order) zal dit implicaties hebben op de prijs van het onderdeel. Eventueel zou de tool er ook voor kunnen zorgen dat dealers ook onderdelen kunnen bestellen bij andere dealers (dit zal mogelijk gemaakt worden door de dealers toe te laten ook een andere dealer als klant toe te voegen). Op die manier zou de klant sneller het benodigde artikel kunnen hebben en is dit financieel voordelig voor beide dealers en eventueel ook voor

de klant. Op die manier hoeft de klant zich geen zorgen te maken over het waar en wanneer (voor hem is dan enkel de prijs en het tijdstip van de levering of afhaling van belang) en verliest de dealer geen klanten omdat hij een onderdeel niet in stock heeft. Om dit alles te realiseren moeten alle gegevens van de dealers gecentraliseerd worden bij AAS. De klanten gebruiken dan een tool die inlogt op de server van AAS waar ze zien of het item dat hij nodig heeft door de dealer verkocht wordt en wat de prijs ervan is. Het gecentraliseerd opslaan van de gegevens is ook nodig omwille van het feit dat de dealers geen IIS server willen hebben (zoals reeds hoger vermeld).

- Om de klant enkele artikels eenvoudiger en sneller te laten aankopen zal er in de tool moeten voorzien worden in een pagina met foto's van 'fast movers' (onderdelen die veel gekocht worden) en/of promoties (onderdelen die tijdelijk goedkoper aangeboden worden). Voorbeelden hiervan kunnen zijn: winteracties (zoals een product dat het aanvriezen van ruitenwisservloeistof voorkomt) en zomeracties (zoals producten om vliegjes gemakkelijker van ruiten te kunnen verwijderen). De meeste van deze zaken horen bij Truck Related Parts. TRP onderdelen zijn geschikt voor bijna alle modellen van vrachtwagens en worden ook geproduceerd door DAF Eindhoven (deze behoren ook allemaal tot een afzonderlijke categorie in DMS).

2.2 Eisen dealers

De tweede stap in het analyseproces was het opstellen van de eisen die de dealers aan het project stelden. Zij zijn namelijk de eigenlijke opdrachtgevers van het probleem. Zij moeten bovendien hun gegevens voor een stuk vrijgeven aan hun klanten. Aangezien deze tool commerciële doeleinden heeft en het openstellen van teveel gegevens financieel nadelig zou kunnen zijn voor de dealers was het belangrijk dat eerst deze groep mensen gehoord werd. Onderstaande eisen werden opgesteld op basis van gesprekken met Peter Van Den Heurck (Algemeen Directeur Truck Trading Limburg¹), Tom Coenen (Magazijnier Truck Trading Limburg), Carlo Reeckmans (Magazijnchef Truck Trading Limburg) en Chris Verschuere (Magazijnchef Garage Van

¹TTL is een dealer in Houthalen die ook een eigen vrachtwagengarage heeft. Bij deze dealer werden de meeste eisen verzameld tijdens de analysefase en zal ook de testfase van de tool uitgevoerd worden. Deze dealer beschikt namelijk over een draadloos netwerk om het mobiel gedeelte te kunnen testen.

den Keybus²).

Om een overzicht te bewaren in de eisen van de dealers zijn deze opgedeeld in een aantal categorieën. De eerste categorie beschrijft eisen die te maken hebben met de manier waarop onderdelen gezocht worden. Daarnaast zijn er een aantal eisen die te maken hebben met de manier van bestellingen plaatsen en afhandelen. Ten slotte is er ook een categorie die te maken heeft met het beheren van gebruikers.

2.2.1 Zoekmogelijkheden

- De dealers en enkele van hun grote klanten hebben een tool met de naam Parts Rapido. Dit is een elektronische catalogus van alle onderdelen die geproduceerd worden bij DAF Eindhoven (dit zijn dus zowel DAF specifieke onderdelen als onderdelen die bij de categorie TRP horen). In deze catalogus wordt het chassisnummer van een vrachtwagen ingegeven. Vervolgens wordt een groep aangeduid waartoe het onderdeel behoort. Binnen die groep wordt nog een onderverdeling aangegeven. Van deze laatste groep geeft de catalogus dan een technische tekening met alle nodige informatie van de onderdelen waaruit deze tekening is opgebouwd. Voor een voorbeeld van zo'n tekening zie figuur 2.1 op pagina 9. Op deze tekening is elk onderdeel aangegeven met een nummer. Aan de hand van dit nummer kan men in de tabel naast de figuur de nodige informatie verkrijgen van dit onderdeel. Deze informatie bestaat onder andere uit het artikelnummer, het aantal van deze stukken dat in de groep zit en een omschrijving van het onderdeel. Het artikelnummer of onderdeelnummer is het nummer dat gebruikt wordt om het onderdeel te bestellen bij de fabrikant. De dealers hebben ook een catalogus met onderdelen van Iveco maar deze werkt op een andere manier (die volgens de dealers niet zo efficiënt is). Voor dit eindwerk zijn enkel DAF-onderdelen van belang aangezien de vlootklanten zelden of nooit Iveco onderdelen nodig hebben.
- De klanten die zelf Parts Rapido gebruiken, zoeken steeds zelf het artikelnummer van de onderdelen op die ze nodig hebben en sturen dan een fax met hun bestelling of plaatsen hun bestelling telefonisch.
- De klanten die deze catalogus niet hebben regelen alles telefonisch met hun dealer. De klant telefoneert naar zijn dealer met een chassisnummer en een beschrijving van het onderdeel. De dealer zoekt dan in

²Dit is een dealer in Belsele (Sint-Niklaas) die ook een eigen vrachtwagen garage heeft.

PartsRapido - 1283507/2 * (1/1)

Bestand Bewerken Zoeken Tekening Beheer Venster ?

1283507/2 * (1/1)

59922A

Nr	Onderdeelnr Aan	Omschrijving	L	X	S	O	PB-CT
1	1246530	1 Remschoen					
2	0667965	1 Geleidingstrol					
3	0667962	1 Lagerbus					
4	0070404	1 As					
5	0593595	2 Lagerbus					
6	1243351	16 Kliknagel					
7	1246530	1 Remschoen					
8	0667965	1 Geleidingstrol					
9	0667962	1 Lagerbus					
10	0070404	1 As					
11	0593595	2 Lagerbus					
12	1243351	16 Kliknagel					
13	1301490	1 Trekveer		X			Verwijzing: 1340297;
14	0392769	1 Remmokas					L=227
15	0213965	1 Sluitring					S=2,3
16	0213966	1 Keuring, assen...					
17	1239871	1 Remankerplaat		X			Verwijzing: 0097182; 0251277; 1347414;
18	0363334	1 Steunremcilinder				X	Verwijzing: 1317432;
19	0646745	2 Lagerbus					
20	0325071	X Smeemiddel					KLEIN
20	1296622	X Snelkop, vetstm.					M8x1

Onderdeel Info

Commerc: Bel. Sales PB/Inst Assortim. Motor L-teken =>Pick L.

0E445526 1283507/2 NUM

Onderdelen

GEREED - voor Help, druk op F1

Figuur 2.1: Voorbeeld technische tekening Parts Rapido

Parts Rapido op wat het juiste artikelnummer is. De artikelnummers wijzigen frequent waardoor het voor de klant moeilijk is om dit bij te houden. Daarnaast is het zo dat de artikelnummers heel zelden op het artikel vermeld staan (soms wel het nummer van de leverancier maar zelden het DAF-artikelnummer). Vervolgens zoekt de dealer in DMS of hij dit onderdeel in stock heeft en wat de prijs is voor die klant. De klant kan dan beslissen om dit onderdeel onmiddellijk te bestellen via de telefoon. Als de klant een artikel nodig heeft dat de dealer niet in stock heeft, zal de dealer via een web applicatie van DAF Eindhoven op zoek gaan of dit onderdeel in stock is in Eindhoven. Is dat niet het geval, dan zal de dealer dit onderdeel bestellen en eventueel telefonisch contact opnemen met DAF Eindhoven om een idee te krijgen van de leveringstijd.

- Vanuit het perspectief van de dealer is de hoofdfunctie van de tool het aantrekkelijk, eenvoudig en efficiënt maken van het opzoeken en bestellen van onderdelen. Aangezien het momenteel zo is dat onderdelen steeds opgezocht worden op chassisnummer aan de hand van Parts Rapido, zou de zoekfunctie van de tool eveneens zo moeten werken om alle communicatie tussen de dealer en de klant overbodig te maken. Dit zou echter impliceren dat er een webservice van DAF Eindhoven gebruikt wordt of dat DAF Eindhoven gegevens hieromtrent vrijgeeft. Aangezien beide opties niet echt realiseerbaar zijn binnen de termijn van een jaar, zal in de tool geopteerd worden om de zoekfuncties een stuk eenvoudiger te maken. De klant kan dan onderdelen opzoeken op artikelnummer (vooral voor de klanten die zelf beschikken over Parts Rapido) of op het chassisnummer, de groep, de subgroep en een beschrijving. Dit laatste is enkel mogelijk voor de klanten die beschikken over een file van DAF Eindhoven waarin deze gegevens staan voor hun eigen vrachtwagens. Deze bestanden zullen aangemaakt worden door een tool van DAF Eindhoven en in de dealertoepassing zal er een mogelijkheid voorzien worden om zo een bestand aan een klant te koppelen. Voor de structuur van dit bestand zie appendix A (p. 95). Deze mogelijkheid zal dus enkel beschikbaar zijn voor de klanten die over zo een bestand beschikken. Daarnaast wordt er nog steeds een mogelijkheid ingebouwd om contact op te nemen met de dealer (voor het geval de klant het artikelnummer niet terugvindt). De dealer stuurt dan een antwoord terug waarop de klant dit onderdeel kan bestellen. Hoe deze zoekmachine precies werd verwezenlijkt, wordt beschreven in sectie 3.1.4 (pp. 28-29).

- De interface van de zoekmachine is heel belangrijk en moet dus eenvoudig en gebruiksvriendelijk zijn. Daarnaast moeten de zoekfuncties efficiënt zijn. Ten minste 80 % van de zoekacties moet succesvol zijn.

2.2.2 Bestellingen

- Als het juiste onderdeel gevonden is, wordt dit toegevoegd aan een winkelmand. Hierin staan alle geselecteerde onderdelen, de hoeveelheid, het type bestelling en de prijs. Voor de dealers is het belangrijk dat in de winkelmand geen nettoprijs wordt meegegeven. Daarnaast wordt aan de klant, indien mogelijk, het artikelnummer niet getoond. De nettoprijs en het artikelnummer maken het voor de klant namelijk eenvoudiger om prijzen te vergelijken en dus eventueel de aankoop ergens anders te doen. Voor de gewone onderdelen zal dus in de zoekresultatenlijst enkel de brutoprijs getoond worden. De nettoprijs (of de brutoprijs en de korting) worden pas in de laatste stap getoond.
- De levering gebeurt vanaf een bepaald bedrag. Dit bedrag is afhankelijk van een aantal afspraken met de klant (en de afstand tussen de dealer en de klant). In de winkelmand krijgt de klant een waarschuwing als het bedrag van zijn bestelling te laag is voor levering. Sommige klanten komen ook onderdelen halen in plaats van ze te laten leveren. Bijgevolg moet het mogelijk zijn voor de klant om zijn keuze te vermelden.
- De bestelling moet wettelijk bindend zijn. Dit wil zeggen dat er geen mogelijkheid mag zijn dat een klant na een bestelling kan beweren dat hij deze bestelling niet gedaan heeft. Hiervoor zullen de nodige wettelijke bepalingen moeten getoond worden wanneer de klant een bestelling plaatst. Deze wettelijke aspecten worden besproken in sectie 5.11 (pp. 80-81).
- Wanneer de klant zijn bestelling door de dealer goedgekeurd is, wordt een email gestuurd waarin de prijs, de leveringstermijn en een lijst met alle bestelde onderdelen staat. De leveringstermijn hangt hierbij af van de bestelling. Is het onderdeel in stock bij de dealer dan kan dit onmiddellijk geleverd worden. Is het onderdeel niet in stock bij de dealer dan moet de dealer dit onderdeel bestellen in Eindhoven. Als dit onderdeel ook in Eindhoven niet in stock is dan hangt de leveringstermijn af van verschillende zaken. Voor deze laatste gevallen moet de dealer wachten op een bevestiging van Eindhoven in verband met de leveringstermijn. Deze termijn wordt pas aan de klant meegedeeld wanneer de klant om

de rest van zijn bestelling komt. De klant wordt hiervan dus niet afzonderlijk verwittigd. Dit zorgt ervoor dat er dus geen communicatie in de tool meer nodig is van de dealer naar de klant eens de bestelling goedgekeurd is en de klant een bevestiging gekregen heeft.

- In sommige gevallen zal dealer A een onderdeel aankopen bij dealer B wanneer een klant van dealer A het onderdeel dringend nodig heeft. Soms wordt dit onderdeel gratis doorgegeven en later terug vervangen door hetzelfde onderdeel. Als het gaat over iets wat dealer B zelf frequent nodig heeft kan hij beslissen om het onderdeel te verkopen aan een meerprijs (meestal 5% bovenop de inkoopprijs) of helemaal niet verkopen. De meerprijs wordt in dit geval doorgerekend aan de klant. Dit geval komt zelden voor, ongeveer één keer om de drie à vier maanden.
- Kleine klanten komen meestal zelf hun onderdelen afhalen. Grotere klanten laten ook onderdelen leveren (dit is het geval bij de meeste vlootklanten). Wanneer de onderdelen moeten geleverd worden, vermeldt de klant de datum waarop hij elk onderdeel wil hebben. Leveringen gebeuren enkel wanneer een bepaald bedrag overschreden is maar er wordt geen rekening gehouden met de grootte van de bestelling. Als de klant niet opgeeft wanneer hij de onderdelen wil, worden alle bestellingen van een week geleverd op een dag die afgesproken werd met de klant. Soms worden onderdelen vroeger geleverd als iemand tijd heeft om dit te doen of als het over grote bedragen gaat.
- In het uitzonderlijke geval dat een klant een onderdeel nodig heeft dat niet verkocht wordt door zijn dealer, gaat deze klant dit onderdeel zelf afhalen bij DAF Eindhoven. De bestelling gebeurt echter nog steeds via de dealer.
- Speciale bestellingen (bijvoorbeeld als de klant zijn onderdeel onmiddellijk wil maar hier geen meerprijs voor wil betalen) worden steeds besproken door de magazijnchef en eventueel de algemeen directeur van de dealer.
- Uitzonderlijk grote bestellingen worden gesignaleerd door de dealer en worden eventueel verkocht aan een speciale korting. Bij het berekenen van de nettoprijs (of korting) wordt namelijk steeds rekening gehouden met het aantal bestelde onderdelen.
- Momenteel gebeuren alle bestellingen via fax of telefoon. Bestellingen worden niet via email gedaan (dit is dus geen optie in de tool) om-

dat bij de dealers meestal verschillende mensen verantwoordelijk zijn voor de communicatie met de vlootklanten. Een email zou dus naar verschillende personen moeten gestuurd worden en zou dus kunnen afgehandeld worden door verschillende personen. Een goede oplossing voor deze tool is om de bestelling te integreren in het DMS systeem van AAS waarmee de dealers nu werken. Op die manier moet de dealer de bestellingen niet meer manueel ingeven. De dealers leggen er wel de nadruk op dat een training voor het gebruik van de tool (in combinatie met DMS) zeker geen overbodige luxe is.

- De prijs voor levering wordt bepaald door het aantal kilometer dat de klant van de dealer verwijderd is, hoeveel onderdelen de klant bij de dealer bestelt,...
- Bij promoties mogen nettoprijzen weergegeven worden. Daarnaast wordt er, ter verduidelijking, een foto toegevoegd. Met zijn onderdeel van de tool moet de dealer dus promoties kunnen toevoegen en verwijderen. Hierbij wordt eveneens een einddatum van de promotie ingevuld. Is de einddatum verstreken dan wordt de promotie automatisch verwijderd. Bovendien kunnen bepaalde promoties enkel voor bepaalde klanten gelden, ook dit kan meegegeven worden in de tool. De promoties worden direct na het inloggen aan de klant getoond zodat hij hier steeds voorbij moet.
- De klant kan kiezen voor een leveringstermijn van 24 uur (Express Order) of van 48 uur (Stock Order). Wanneer de klant een Express Order kiest moet hij bereid zijn om hiervoor een meerprijs te betalen (meestal is dit 5 % extra). Stock orders moeten binnen zijn voor 16 uur en Express orders voor 18 uur. Aangezien er door de tool sommige zaken sneller en andere misschien trager zullen verwerkt worden, zullen deze uren eventueel aangepast moeten worden.
- Als een klant meer besteld heeft dan de dealer in stock heeft, moet er een mogelijkheid zijn om een deel te kunnen afleveren en de rest in bestelling bij Eindhoven te plaatsen. Dit wordt geregeld aan de hand van het Dealer Management System.
- Aangezien bij de dealers verschillende personen verantwoordelijk kunnen zijn voor de communicatie met de vlootklanten zal het onderdeel voor de dealer beschikbaar moeten zijn op de pc van alle verantwoordelijken. Er moet dan ook voor gezorgd worden dat slechts één persoon tegelijk een bepaalde vraag of bestelling kan behandelen. Daarnaast is

het zo dat alle onbehandelde vragen en bestellingen steeds op alle pc's beschikbaar moeten zijn en blijven zolang niemand ze volledig afgehandeld heeft. Hoe dit opgelost zal worden, wordt besproken in sectie 5.6.1 (p. 73).

- Om ervoor te zorgen dat een bestelling of vraag niet te lang onbehandeld blijft, zou het interessant zijn dat na een bepaalde tijd een venster verschijnt om de dealer hierop attent te maken. Dit venster wordt periodiek herhaald zolang de bestelling of vraag niet afgehandeld is. Zoals verder zal besproken worden zal er bij elke gebeurtenis een kleurverandering optreden van het icoontje van het dealerprogramma. Om er voor te zorgen dat de kleurverandering van het icoontje niet over het hoofd zou gezien worden, kan eventueel ook een waarschuwingsvenster getoond worden als er een verandering van status is.

2.2.3 Gebruikers

- Op dit moment is het zo dat de dealer bijhoudt over welke onderdelen de klanten informatie ingewonnen hebben. Heeft een klant over een bepaald artikel informatie opgevraagd en het enkele dagen later niet aangekocht dan wordt de klant gecontacteerd en naar de reden gevraagd. Dit wil immers meestal zeggen dat de klant ergens anders dit onderdeel gekocht heeft voor een betere prijs. Als dit het geval is laat de dealer dit weten aan DAF Eindhoven (via telefoon, fax of email) zodat deze hun prijzen eventueel kunnen aanpassen en zo deze gevallen in de toekomst vermijden. Aangezien dit regelmatig voorkomt moet hier rekening mee gehouden worden in de tool. Dit zal opgevangen worden door aan de klant te vragen waarom hij onderdelen uit zijn winkelmandje verwijdert. Hoe dit precies zal gerealiseerd worden, wordt beschreven in sectie 3.3 (pp. 35-44).
- In sommige gevallen komt een verkoper bij een klant en verkoopt hem rechtstreeks onderdelen. Om dit mogelijk te maken moeten de verkopers kunnen inloggen als klant. De verkoper kan dan aanduiden voor welke klant hij die bestelling gedaan heeft. Aangezien verkopers dikwijls gebruik maken van PocketPC's zal deze mogelijkheid ook in de mobiele versie moeten zitten. Deze mogelijkheid zal enkel ingebouwd worden als er tijd over is en wordt dus niet als een noodzakelijke eis beschouwd.
- Voor de dealers is het essentieel dat de klant nooit kan zien wat er precies in de stock van de dealer zit. Elke dealer heeft immers een eigen

garage en verkoopt soms een onderdeel niet (hoewel hij het in stock heeft) omdat hij het zelf nodig zou kunnen hebben. Voor de dealer zelf is het wel interessant om te weten of hij de onderdelen in stock heeft en waar in zijn magazijn hij de onderdelen moet vinden. Sommige dealers hebben ook verschillende vestigingen. Voor deze is het ook interessant om te weten in welke vestiging het onderdeel zich bevindt. Hoewel de klanten deze informatie niet te zien krijgen is het dus de bedoeling dat de dealer deze informatie wel krijgt bij een bestelling. Deze gegevens zijn allemaal beschikbaar in de SQL server van de dealer.

- De meeste vlootklanten bestellen bijna enkel onderdelen in Stock orders aangezien ze zelf een kleine voorraad onderdelen hebben. Express orders komen dus minder voor. Enkel bij deze laatste is de leverings-termijn van groot belang.
- De personen die de bestelling plaatsen (bij de klant) zijn voor 90 % mensen met technische kennis. Dikwijls is dit het hoofd van de mechaniciens van de garage. Gezien de opbouw van Parts Rapido hebben ook de mensen die de artikelnummers opzoeken bij de dealers best technische kennis.
- Er is een gevaar dat een klant zijn aanmeldnaam en -paswoord doorgeeft aan een concurrent. Op die manier zou een concurrent van de dealer immers alle prijzen te weten kunnen komen. Hoewel dit probleem niet helemaal te vermijden is, kan de dealer dit geval opmerken doordat deze 'klant' meer onderdelen zal verwijderen dan aankopen.
- Het is voor de dealer de bedoeling dat enkel de goede klanten (waarmee reeds een vertrouwensrelatie is opgebouwd) over een login en paswoord voor de internettoepassing beschikken. De vragen aan de dealer over artikelnummers zullen dus behoorlijk beperkt zijn aangezien deze klanten goed weten waar ze mee bezig zijn (en sommige van deze klanten zelf Parts Rapido gebruiken). Klanten die vragen stellen over bepaalde onderdelen kunnen het soms zelfs moeilijk uitleggen aan de telefoon over welk onderdeel het gaat. Deze vraag stellen via de tool zal dus misschien voor problemen kunnen zorgen maar zou normaal niet frequent mogen voorkomen.
- Van de klanten moeten naast de aanmeldnaam en -paswoord ook algemene gegevens bijgehouden worden. Hiertoe behoren in eerste instantie het BTW nummer (dit is immers uniek voor elke klant) en het email-adres (dit is nodig om de bevestiging van de bestellingen te sturen). Daarnaast zal uiteraard de naam en het adres opgeslagen worden.

2.3 Eisen vlootklanten

Als laatste stap in het analyseproces werden een aantal eisen opgesteld van de vlootklanten. Zij maken immers ook een groot deel van de gebruikers uit. Tot onze grote spijt konden we, omwille van twee redenen, met geen enkele vlootklant een gesprek regelen. De eerste reden is dat vele dealers weigerachtig stonden om gegevens over hun klanten vrij te geven. Daardoor konden we ook geen contact opnemen met deze klanten. De tweede reden was dat de vlootklanten waarvan we wel de contactgegevens gekregen hadden een gebrek aan tijd hadden en ook niet enthousiast waren om hun huidige werkwijze te veranderen. Hoewel dit misschien te wijten is aan *Change Reluctancy* waren de meeste vlootklanten ook tevreden over de service van hun dealers. Daarom zagen ze niet onmiddellijk de relevantie van deze tool in. De eisen werden dan ook opgesteld aan de hand van de gesprekken met de dealers.

- Voor de klanten is het belangrijk dat deze toepassing een meerwaarde biedt aan het bestellen van onderdelen. Momenteel is het namelijk zo dat alles nagenoeg onmiddellijk gebeurt door het gebruik van de telefoon en fax. Een eerste belangrijke eis is dus dat het proces van bestellingen plaatsen met deze toepassing ook in real-time moet gebeuren.
- Een tweede belangrijke eis is dat het opzoeken van onderdelen eenvoudig kan gebeuren. Momenteel is hier namelijk zo goed als altijd hulp van de dealer voor nodig. Als dit aspect vervangen wordt dan kan de klant een stuk sneller zijn onderdelen bestellen en zorgt dit ook voor de dealer voor een pak minder werk.
- Voor de klanten zou deze tool een stuk hun werk vereenvoudigen als bijvoorbeeld ook automatisch hun boekhouding wordt gedaan als ze bestellingen plaatsen. Dit kan eventueel gedaan worden door AAS als dit buiten het bestek van dit eindwerk ligt.
- Momenteel is het zo dat sommige klanten prijsvraag (nettoprijs) doen van een artikel zonder dit te bestellen. Daarnaast stellen sommige klanten de vraag of een artikel op dat moment in stock is. Deze beide vragen naar de dealer toe moeten dus eveneens mogelijk zijn met de tool voor de klanten.
- Het is een vereiste dat de winkelmand blijft bestaan als de toepassing afgesloten wordt. Op deze manier kan de klant zijn mand opbouwen over verschillende dagen of zelfs langer. Dit zal opgelost worden door dit mandje op te slaan in de database. Op deze manier kan ook een

geschiedenis bijgehouden worden. Dit laatste zorgt ook voor een meerwaarde ten opzichte van de huidige werkwijze.

- Klanten die momenteel hun bestellingen doen via fax, vermelden nu steeds een bonnummer per bestelling (dus niet per onderdeel maar per afgewerkte winkelmand). De tool moet de klant ook deze mogelijkheid bieden aangezien de klant dit nodig heeft voor zijn administratie.
- Om zeker te zijn dat klanten het correcte onderdeel bestellen zou het handig zijn om te voorzien in een mogelijkheid om foto's aan de onderdelen toe te voegen. Op PocketPC kan die foto eventueel vervangen worden door een uitgebreidere beschrijving om de performance aanvaardbaar te houden.
- De tool moet enigzins de intelligentie van de magazijnchef integreren. Er moet dus voor gezorgd worden dat de klant zo weinig mogelijk fouten kan maken (bijvoorbeeld geen onderdelen bestellen die hij onmogelijk nodig kan hebben). Dit zal gerealiseerd worden door de file met de chassisnummers van de klant en door het feit dat de klant steeds hulp aan zijn dealer kan vragen.

Hoofdstuk 3

Ontwerp

In dit hoofdstuk zal de ontwerpfase van dit eindwerk besproken worden. In eerste instantie worden een aantal ontwerpbeslissingen genomen (op basis van enkele eisen die geformuleerd werden in de analyse) en wordt verklaard waarom deze beslissingen genomen worden. Vervolgens worden deze beslissingen, samen met de rest van de eisen, in een algemeen ontwerp gegoten aan de hand van enkele UML-diagramma's. Deze geven ten slotte aanleiding tot meer specifieke ontwerpen: de algemene opbouw van de user interface, het databasemodel en de achterliggende logica.

3.1 Ontwerpbeslissingen en verantwoording

3.1.1 Globalization en localization

Zoals reeds vermeld in de analyse moet er voor deze toepassing rekening gehouden worden met het feit dat er op een eenvoudige manier van taal kan veranderd worden. Bij de gebruikers zijn er immers zowel Nederlandstalige als Franstalige dealers en klanten. Standaard zal de toepassing in het Engels geschreven worden maar er zal voor gezorgd worden dat de vertaling op een eenvoudige manier kan verlopen. Om hiervoor te zorgen zijn er verschillende aanpakken mogelijk. We sommen hier de belangrijkste op en verantwoorden de keuze van twee ervan. De drie methodes zijn ook duidelijk gemaakt aan de hand van figuur 3.1 op pagina 21.

- De eerste methode maakt gebruik van de *System.Globalization* namespace. Bij deze methode zullen we voor elke taal één XML resourcefile (.resx) aanmaken. Deze methode zorgt ervoor dat het bijvoorbeeld mogelijk wordt om de cultuurinformatie van een browser of van het besturingssysteem op te vragen en op basis daarvan de gewenste taal

automatisch te laten selecteren. Als we van dit aspect willen gebruik maken, zullen we ook de *System.Threading* en de *System.Resources* namespaces gebruiken. De eerste zal het mogelijk maken om de cultuur op te vragen en de tweede hebben we nodig om de resourcefiles met de talen te kunnen oproepen in de toepassing. Het voordeel van deze methode is dat het toevoegen van een nieuwe taal eenvoudig is. Daarvoor moet er slechts een nieuwe file aangemaakt worden met de juiste naam. Deze file en dus de nieuwe taal kan toegevoegd worden zonder het programma te hercompileren. Om dit alles mogelijk te maken voorzien we eerst en vooral een file met de standaard taal. We geven deze file bijvoorbeeld de naam *ResourceFile.resx*. Voegen we bijvoorbeeld de taal Nederlands toe, dan maken we eenvoudigweg een kopie van *ResourceFile.resx* en vertalen de waarden van de verschillende tekststrings. Deze nieuwe file geven we de naam *ResourceFile.nl.resx*. Als we nu in de toepassing de cultuur instellen op *NL* zal de resource manager automatisch de tekststrings uit de file *ResourceFile.nl.resx* halen. Wordt er geen file met de juiste taal gevonden wordt gebruik gemaakt van de standaard taal uit *ResourceFile.resx*.

- Bij de tweede methode wordt er gebruik gemaakt van een zelfgeschreven XML-file die gelezen wordt aan de hand van een filereader. Deze methode wordt gebruikt in de Pocket DMS van AAS (dit is een mobiele versie van het Dealer Management System). Het voordeel van deze methode is dat alle taal informatie kan opgeslagen worden in één enkele file. Alle talen worden dus in dezelfde file opgeslagen (bij de eerste methode wordt er per taal een nieuwe file aangemaakt). Het nadeel van deze aanpak is dat het toevoegen van een nieuwe taal minder eenvoudig is dan bij de eerste methode en dat we de toepassing ook expliciet moeten opgeven waar een bepaalde taal uit de resourcefile moet gehaald worden. Het principe van de eerste en de tweede methode zijn echter gelijkaardig.
- Bij de derde methode tenslotte worden alle tekststrings in de verschillende talen opgeslagen in de database. Bij het laden van de pagina worden deze uit de database opgehaald aan de hand van een SQL-connectie. Deze methode wordt gebruikt in het Dealer Management System van AAS. Het voordeel van deze aanpak is dat alle informatie gecentraliseerd wordt opgeslagen in de database. Nadelen van deze aanpak zijn dat we op die manier het verkeer met de database verhogen, dat er meer informatie zal moeten opgeslagen worden in de database (als er in de database bijvoorbeeld informatie over 25000 onderdelen

moet opgeslagen worden in 3 verschillende talen dan zullen er 75000 records moeten opgeslagen worden in plaats van 25000) en dat het toevoegen van een nieuwe taal moet gebeuren in de database.

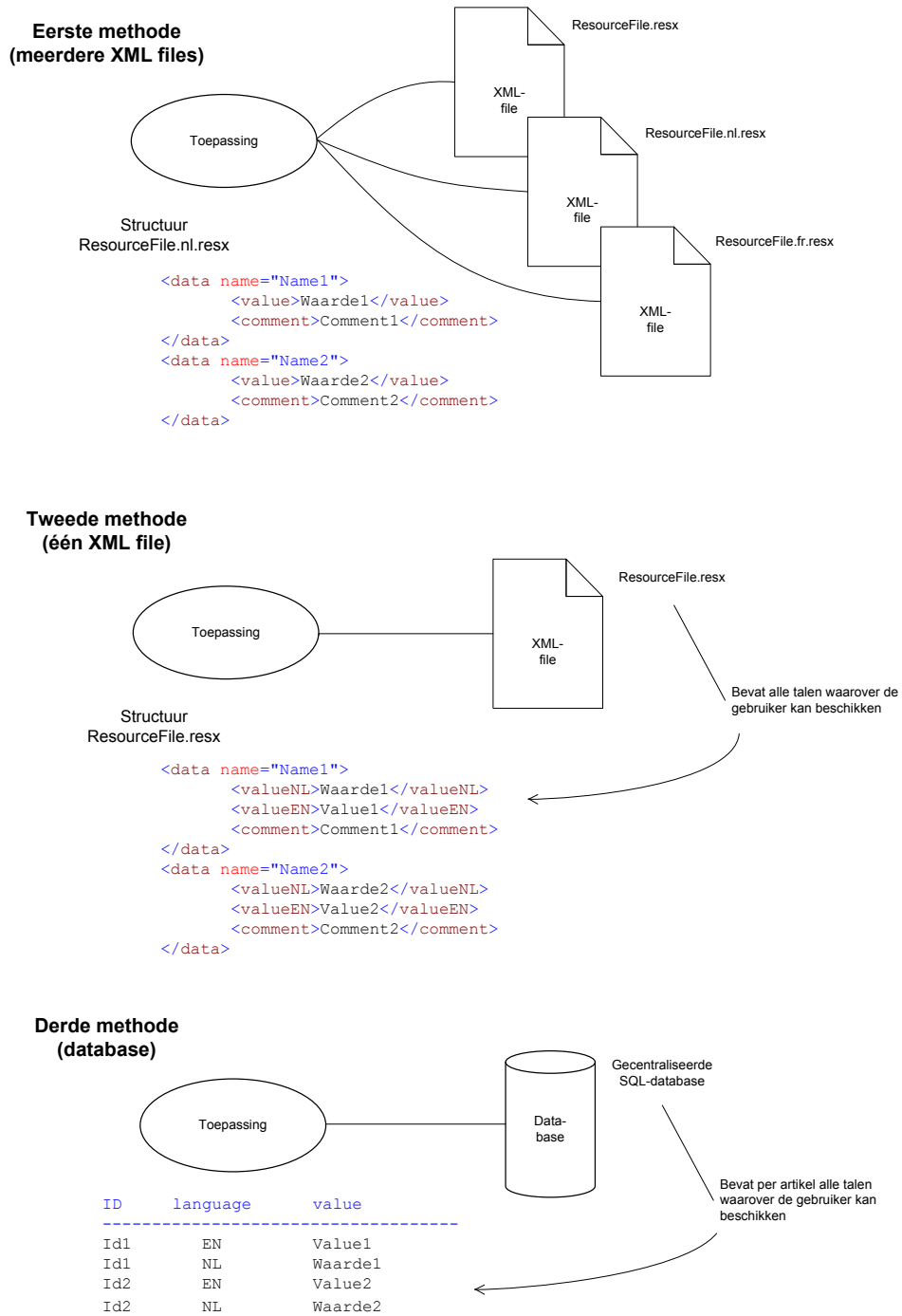
Voor de namen en beschrijvingen van de onderdelen zullen we de derde methode gebruiken. De gegevens van de onderdelen bevinden zich namelijk in de database van de dealers in het Frans en het Nederlands. Bij het opvragen van deze informatie zal dus enkel die taal opgevraagd worden die de gebruiker nodig heeft.

Voor de andere taalafhankelijke zaken in de toepassing werd geopteerd voor het gebruik van de eerste methode. Deze methode leent zich, ons inziens, namelijk het best voor het toevoegen van een nieuwe taal. Daarnaast is het voordeel van deze methode dat het ophalen van de strings op een eenvoudige manier gebeurt door de klasse *ResourceManager* uit de *System.Resources* namespace. Er moet dus geen gebruik gemaakt worden van een filereader of databaseconnectie. Een concreet voorbeeld van het gebruik van deze methodes zal geïllustreerd worden in sectie 5.8 (pp. 76-78).

3.1.2 Internet toepassing

Zoals reeds vermeld in de eisen werd oorspronkelijk gedacht aan een volledig webgebaseerde toepassing. Dit zou er immers voor zorgen dat het in werking stellen van het programma (en latere updates) zeer eenvoudig is. Een volledig webgebaseerde toepassing kan immers volledig beheerd worden op een centrale plaats (in het geval van dit eindwerk de servers van AAS in Temse). Het grootste nadeel van dit type toepassing is vanzelfsprekend dat ze niet bruikbaar is in het geval dat het netwerk niet beschikbaar is of de servers van AAS niet functioneren. Dit nadeel is echter te verwaarlozen als we weten dat in België de typische netwerkbeschikbaarheden liggen tussen 99.95% en 99.99% en ook de servers van AAS een hoge beschikbaarheid hebben.

Om het real-time aspect van de communicatie te kunnen verwezenlijken werd echter geopteerd om voor het deel voor de dealers een Windows toepassing te gebruiken. Het is namelijk zo dat een Windows toepassing voortdurend open kan blijven op de achtergrond en zo de dealer waarschuwen als er een actie moet ondernomen worden. Op die manier hoeft de dealer niet steeds zelf op een webpagina te controleren of er acties opgetreden zijn. Hoe dit deel voor de dealer dan zal communiceren met de andere delen wordt besproken in de volgende sectie.



Figuur 3.1: Verduidelijking methodes resources

3.1.3 Communicatie

Omdat er in de toepassing een directe link moet zijn tussen de dealer en de klant zal communicatie belangrijk zijn. Daarnaast is het zo dat de dealertoe-passing onmiddellijk (of schijnbaar onmiddellijk) moet reageren op bepaalde acties van de klant. Voor de klanttoepassing zullen we gebruik maken van een ASP.NET toepassing. Op die manier hoeft de klant enkel te beschikken over een toestel met een internet browser. Dit kan zowel een mobiel toestel als een niet-mobiel toestel zijn. Omdat de informatiestromen tussen de verschillende onderdelen belangrijk zijn, bespreken we hier de mogelijke implementaties van deze communicatie en verantwoorden we de beslissingen die hieromtrent genomen werden. Deze keuzes zullen immers een grote impact hebben op de performantie en de opbouw van het systeem.

We bespreken eerst welke communicatie nodig is tussen de verschillende onderdelen van het systeem. Het systeem bestaat omwille van de verscheidenheid van mogelijke acties van klanten en dealers uit twee grote delen: het klantprogramma en het dealerprogramma. Beide hebben een aantal acties die communicatie vereisen. Voor het klantprogramma zijn dit de volgende acties: aanmelden bij de applicatie, een onderdeel zoeken, promoties bekijken, de geschiedenis van de bestellingen bekijken, het toevoegen van een onderdeel aan de winkelmand, het verwijderen van een onderdeel uit de winkelmand, een vraag stellen aan de dealer, het paswoord wijzigen en een bestelling plaatsen bij de dealer. Voor het dealerprogramma zijn het de volgende acties die gebruik maken van communicatiemechanismen: kijken of er acties (vragen, bestellingen, . . .) staan te wachten, promoties beheren (toevoegen, aanpassen en verwijderen), gebruikers beheren (toevoegen, aanpassen en verwijderen), antwoorden op een vraag van een klant, een bestelling afhandelen, reageren op het feit dat een klant de prijs van een artikel te duur vond en informatie uit de SQL server van de dealer naar de SQL server van AAS sturen.

Deze acties vereisen allen een communicatiestroom tussen het klantprogramma en het dealerprogramma, een uitwisseling van informatie met een database (SQL server) of het opvragen of ontvangen van informatie van een IIS server. Het grootste deel van de communicatie zal dus over Internet verlopen. Aangezien het klantprogramma een ASP.NET toepassing is, gebeurt de communicatie tussen de klant en alle andere componenten via HTTP. Hoe de communicatie tussen de IIS server (die de ASP.NET toepassing van de klanten host) en het dealerprogramma verloopt is een ander verhaal. We overlopen nu enkele mogelijke oplossingen en houden hierbij ook rekening met de beveiliging van de informatie.

- De eerste mogelijke oplossing is deze waarbij alles gebeurt via rechtstreekse communicatie met de **database**. Net zoals bij alle andere mogelijke oplossingen zullen alle gegevens van de klanten en de dealers worden opgeslagen op een SQL server bij AAS. Acties van de klant die een reactie verwachten van de dealer zetten rechtstreeks een vlag in de database. De dealertoepassing controleert periodiek (bijvoorbeeld elke minuut) deze vlag via een rechtstreekse SQL-connectie en reageert indien nodig door de benodigde gegevens uit de database in te lezen. Deze methode is één van de meest eenvoudige om te implementeren maar veroorzaakt heel wat verkeer van en naar de database bij AAS. Daarnaast is het zo dat de dealertoepassing niet onmiddellijk reageert op een actie van de klant maar slechts na controle van de database. Om deze tijd te verkleinen zouden we de dealertoepassing ook continu de database kunnen laten aanspreken en controleren. Dit zorgt echter voor een overbelasting van het netwerk en de databaseserver. Deze beide oplossingen werden tijdens de implementatiefase uitgetest en daaruit bleek dat de continue oplossing onaanvaardbaar is. Als we bij de periodieke oplossing een interval van bijvoorbeeld 10 seconden nemen is de hoeveelheid verkeer aanvaardbaar en is de reactie van het dealerprogramma ook schijnbaar in real-time. Een ander nadeel van deze oplossing is dat de beveiliging van de verbinding niet eenvoudig is.
- Een gelijkaardige oplossing is die met **webservices**. Webservices maken gebruik van Simple Object Access Protocol (SOAP). Dit is een protocol dat ontwikkeld werd om applicaties met elkaar te laten communiceren over HTTP en maakt gebruik van XML. De communicatie van de client naar de service gebeurt met SOAP objecten en de service antwoordt de client aan de hand van XML. Webservices draaien op een IIS server en maken het verkeer tussen de verschillende toestellen minder omvangrijk maar niet minder frequent. De dealertoepassing moet nog steeds periodiek aan een webservice gaan vragen of er acties opgetreden zijn die een reactie vereisen. De webservice zelf zal dit controleren door terug de database aan te spreken. Hier gelden dus analoge voor- en nadelen als in het vorige geval. Deze methode is relatief eenvoudig te implementeren en is zeker aanvaardbaar op het gebied van veiligheid als de webservices goed beveiligd worden. Deze methode werd ook uitgetest tijdens de implementatiefase en zorgt voor een analoge performantie als de vorige oplossing. Op het vlak van veiligheid en hoeveelheid informatie die wordt doorgestuurd geniet deze oplossing

echter de voorkeur op de vorige. Een mogelijkheid om de hoeveelheid verkeer te beperken is om webservices te installeren bij de dealer. De klant kan dan deze services aanspreken wanneer er een reactie nodig is. Het nadeel van deze oplossing is dat de dealers hiervoor een IIS server moeten hebben wat een meerkost kan betekenen voor de dealers (zoals vermeld in sectie 2.1 op pp. 4-7).

- Een volgende mogelijkheid is het gebruik van **sockets** (TCP). In dit geval functioneert het dealerprogramma als een soort server. Er is namelijk voorzien in een oneindige lus (in een afzonderlijke thread die afgesloten wordt als het programma afgesloten wordt) die luistert op een bepaalde poort. Het klantprogramma functioneert dan als client en stuurt een bericht naar het dealerprogramma via de openstaande, luisterende poort. Op die manier kan het dealerprogramma onmiddellijk reageren op acties van de klant. Het grootste nadeel aan deze methode is het feit dat er bij de dealer steeds een poort moet open staan. Dit kan een bron zijn van mogelijke inbreuken en is moeilijk hiertegen te beveiligen. Bovendien vereist deze methode het instellen van routers en firewalls.
- Een andere oplossing is om te werken met **emails**. Deze methode is echter niet de meeste elegante oplossing. Dit vereist bovendien het gebruik van externe mailprogramma's wat zeker niet de gebruiksvriendelijkheid van de tool zal bevorderen. Bovendien is het zo dat bij de dealers dikwijls meerdere mensen actief bezig zijn met de communicatie met de vlootklanten. Dit zou impliceren dat elk van die personen diezelfde email zou moeten krijgen. Hoewel dit geen probleem mag zijn, is het op die manier wel moeilijker om ervoor te zorgen dat er nooit twee personen dezelfde actie kunnen behandelen.
- Een volgende technologie die in aanmerking kwam voor de communicatie binnen dit systeem is Real Time Communication [4],[17]. In de .NET technologie is immers voorzien in een API die gebaseerd is op de functionaliteiten van Windows Messenger. Deze **RTC Client API** voorziet in de volgende functionaliteiten:

The RTC Client API enables you to build applications that can make PC-PC, PC-phone, or phone-phone calls or create Instant Messaging (IM) sessions over the Internet. Both voice and video calls can be established on PC-PC calls. Presence information on a list of contacts is also supported. Application sharing and whiteboard can be added to enhance the

communication capabilities of any type of session. [17]

Om deze oplossing te kunnen implementeren moeten we gebruik maken van een emailadres of een IP-adres om de client op afstand te kunnen aanspreken. Aan de hand van dit adres kan een applicatie (die gebruik maakt van RTC) een bericht sturen naar een andere pc. Deze oproep kan opgevangen worden door een applicatie op deze pc en hierop reageren met de gepaste actie. Dit kan gebeuren via een Session Initiation Protocol (SIP) server. Deze server houdt namelijk bij welke gebruikers online zijn en stuurt berichten van de ene client naar de andere. Dit impliceert dus de installatie van een extra server. Het voordeel van deze aanpak is dat de communicatie onmiddellijk gebeurt en dat de overhead dus tot een minimum beperkt wordt. Om RTC te kunnen gebruiken op PocketPC moeten we er rekening mee houden dat dit enkel kan met versie 1.2 van de RTC Client API. Latere versies zijn namelijk nog niet beschikbaar voor mobiele toestellen. Deze vorm van communicatie maakt gebruik van enkele poorten in de range van 1024 tot 65535. Bovendien is het zo dat RTC enkel kan doorgegeven worden door routers (NAT's) die Universal Plug 'n' Play (UPnP) ondersteunen.

- Een laatste oplossing is gebruik te maken van **File Transfer Protocol** (of verwante protocols zoals FTPS, Secure FTP, SFTP, TFTP). Dit is een methode om bestanden van een client naar een server te sturen en vice versa. Aan de hand van een FTP server zou het mogelijk zijn om bestanden met de nodige informatie door te sturen van zowel het klant programma als het dealerprogramma. Deze bestanden kunnen dan geïnterpreteerd worden door de server en daaruit kunnen dan de nodige acties ondernomen worden. Deze oplossing impliceert het gebruik van een FTP server. Het grootste nadeel van deze aanpak is dat het doorsturen van informatie vooral gebeurt aan de hand van files. Deze aanpak is dus vergelijkbaar met het gebruik van emails en is dus zeker niet de meest elegante oplossing. Daarnaast is ook hier de beveiliging een moeilijkheid.

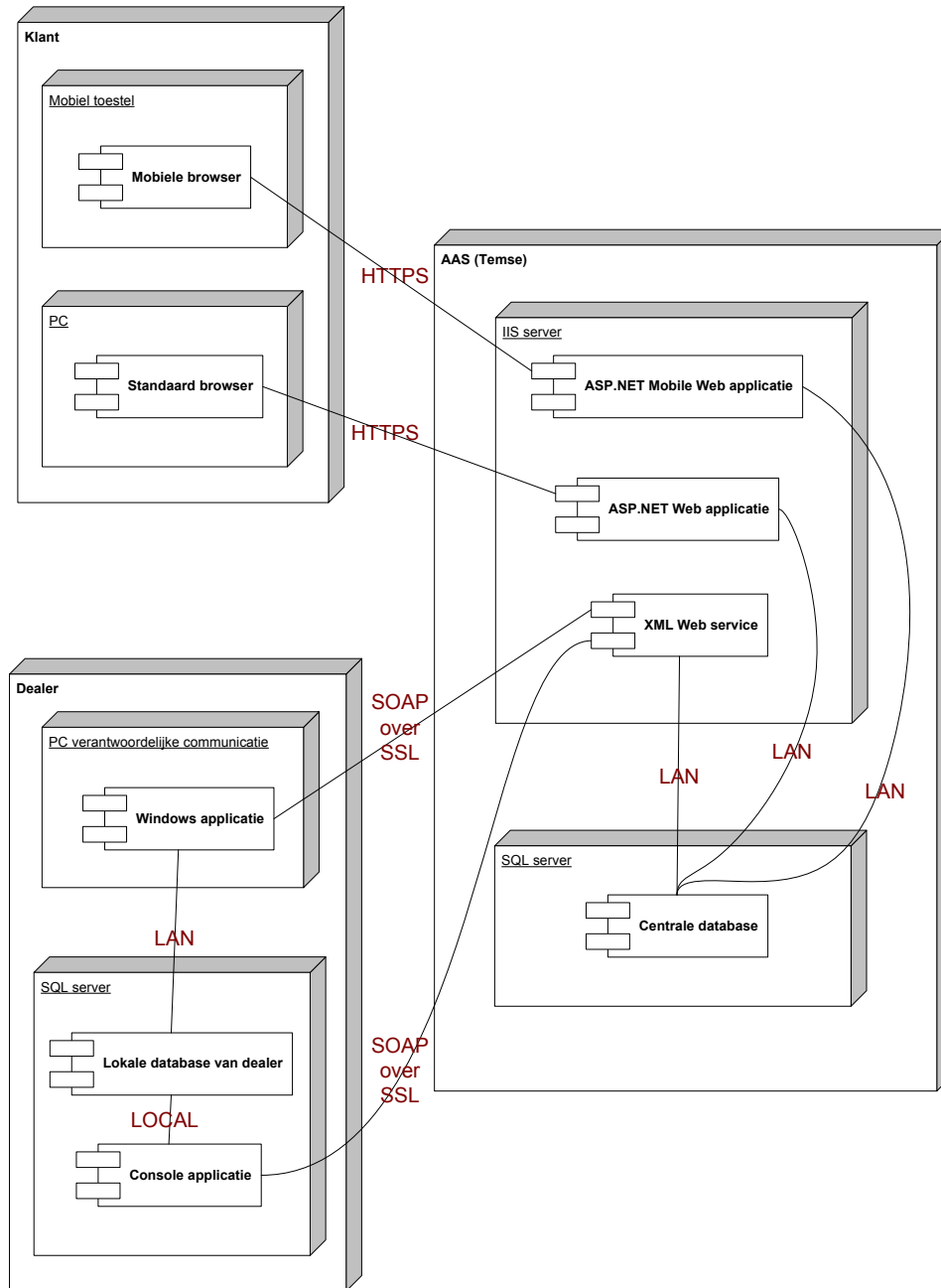
Een eerste criterium waarop deze oplossingen bekeken werden is het gebruik van bepaalde poorten. Aangezien de meeste van de dealers en klanten gebruik maken van routers en firewalls die in sommige gevallen beheerd worden door externe firma's is het aangewezen om enkel gebruik te maken van TCP poort 80 of poort 443. De eerste poort staat immers standaard open voor internet-toepassingen en de tweede poort staat open bij de dealers om het gebruik van HTTPS webpagina's mogelijk te maken. Op basis van dit criterium moesten we de oplossing met de **RTC Client API** en de oplossing met

sockets uitsluiten. Daarnaast waren de oplossingen met **emails** of **FTP** zeker niet de eenvoudigste en eveneens niet de meest efficiënte oplossingen aangezien hier ofwel gebruik zou moeten gemaakt worden van een extern mailprogramma of van een FTP-server.

De overblijvende oplossingen (database en webservices) zijn beide gebaseerd op het zelfde principe en zijn daarom ook vergelijkbaar op het vlak van performantie en gebruikte technologie. Van deze twee kozen we uiteindelijk de oplossing met webservices aangezien dit de oplossing is waarbij de communicatie tot een minimum beperkt wordt en de beveiliging het eenvoudigst is.

Dit impliceert dat bij de dealer een Windows toepassing zal geïnstalleerd worden die periodiek een webservice bevraagt die op zijn beurt in de database gaat controleren of er bepaalde acties opgetreden zijn. Sommige dealers beschikken over een Citrix-server. Bij deze zal het programma op deze server geïnstalleerd worden en gebruikt iedereen dus hetzelfde programma. Op die manier gebeurt het periodiek bevroegen van de webservice slechts één keer per dealer. Bij de dealers zonder Citrix-server waarbij er wel verschillende personen verantwoordelijk zijn voor de communicatie met de vlootklanten zal het programma op de pc van elk van deze personen geïnstalleerd worden. Bij deze dealers zal de webservice periodiek bevraagd worden één keer per verantwoordelijke. Hoewel dit niet de meest ideale oplossing is, is deze nog steeds aanvaardbaar. Later kan eventueel gezocht worden naar een gedistribueerde versie van dit programma.

Kort samengevat zal de structuur van de toepassing er uitzien als volgt (zie figuur 3.2 op pagina 27): het klantprogramma is een web-gebaseerde toepassing die draait op een IIS server. De klant heeft dus enkel een (mobiel of vast) toestel nodig dat beschikt over een internet browser. De IIS server die deze ASP.NET toepassing draait bevindt zich bij AAS in Temse en communiceert via een lokaal netwerk met de gecentraliseerde SQL server (eveneens bij AAS in Temse). De dealertoepassing is een stand-alone Windows toepassing die periodiek een webservice controleert die draait op diezelfde IIS server bij AAS. Deze webservice controleert via het lokaal netwerk de SQL server op veranderingen die wijzen op een actie van de klant. Naast deze webservice zijn er nog enkele services beschikbaar voor het dealerprogramma om gegevens in en uit de database te kunnen schrijven en lezen. Ten slotte haalt de dealertoepassing ook periodiek (bijvoorbeeld éénmaal per dag) gegevens uit de SQL server van de dealer via het lokaal netwerk om deze informatie te centraliseren in de database bij AAS. Dit gebeurt eveneens aan de hand van een webservice.



Figuur 3.2: Deployment diagramma volledige toepassing

3.1.4 Zoekmachine

Voor elke vrachtwagen die geproduceerd wordt bij DAF Eindhoven wordt (op basis van het chassisnummer) in een database bijgehouden welke onderdelen in deze vrachtwagen zitten. Deze onderdelen worden eerst ingedeeld in enkele groepen en binnen deze groepen nog eens in verschillende categorieën. Van elk van deze groepen van het laagste niveau bestaat een technische tekening. Op deze tekening staan de gegevens van elk van de onderdelen in deze groep. Deze gegevens zijn voor de dealers aanspreekbaar aan de hand van Parts Rapido. Dit pakket wordt door DAF aangeboden sinds 1994.

Dit systeem, dat draait op DVD of online via het Internet, bevat alle onderdeleninformatie van elke op dit moment rijdende DAF, ... Op chassisniveau is exact en onmiddellijk te zien uit welke onderdelen een bepaalde truck bestaat en welke softwareversies er in de aanwezige elektronische systemen gebruikt worden. [1]

De zoekfunctionaliteit van Parts Rapido kan enkel overgenomen worden aan de hand van gegevens van DAF Eindhoven of via een webservice aangeboden door DAF Eindhoven. Hoewel voor dit laatste plannen zijn, is dit niet mogelijk binnen het tijdsbestek van dit eindwerk. Daarom werd beslist om de zoekfunctie voor de klanten te vereenvoudigen.

- In eerste instantie zal er voorzien worden in een functie om te zoeken op artikelnummer. Dit is hoofdzakelijk voor de klanten die deze artikelnummers kennen uit ervaring of de klanten die zelf over Parts Rapido beschikken.
- Een tweede functionaliteit wordt voorzien aan de hand van een bestand dat door de dealers beschikbaar gesteld wordt aan bepaalde klanten. In dit bestand zijn alle artikels die bij een aantal chassisnummers horen vermeld. Dit bestand zal worden gegenereerd aan de hand van Parts Rapido. Voor de structuur van dit bestand zie appendix A (p. 95). Dit bestand zal geïntegreerd worden in de klanttoepassing zodat de klant op zoek kan naar een onderdeel aan de hand van het chassisnummer, de groep, de subgroep en een beschrijving van het artikel.
- Ten slotte wordt ook nog voorzien in een mogelijkheid om hulp te vragen aan de dealer. Deze optie is voor de klanten die geen gebruik kunnen maken van beide vorige opties of indien het onderdeel niet gevonden werd met de andere mogelijkheden. Deze oplossing vraagt aan de klant het chassisnummer en een beschrijving van het onderdeel (en de groep en subgroep als hij deze kent). Deze gegevens worden naar

het dealerprogramma gestuurd. Eens de dealer een antwoord op deze vraag heeft geformuleerd, kan de klant dit antwoord lezen en op die manier het artikel toevoegen aan zijn winkelmand.

Uiteindelijk zal de klant dus aan de hand van deze zoekfuncties zijn winkelmand kunnen samenstellen en onmiddellijk of later beslissen om de gevonden onderdelen te bestellen. Naast de gezochte onderdelen kan de klant eventueel ook nog onderdelen toevoegen die in promotie staan. Deze promoties worden getoond wanneer de klant inlogt.

3.1.5 Beveiliging

Secure Sockets Layer

Omdat in de toepassing veel gevoelige data zal verstuurd worden, werd geopteerd om voor zowel de ASP.NET Web applicatie, de ASP.NET Mobile Web applicatie als de webservice, gebruik te maken van Secure Sockets Layer (SSL). AAS zal, voor het in werking treden van de toepassing, een SSL-certificaat aanvragen. Het klantprogramma zal dus beschikbaar zijn via HTTPS en het dealerprogramma zal de webservice eveneens aanspreken via een HTTPS link. Dit verkeer maakt gebruik van TCP poort 443. Deze poort staat echter standaard open op de firewalls en routers van de dealers aangezien ook bepaalde diensten van DAF Eindhoven aangeboden worden via HTTPS.

Het gebruik van SSL zorgt ervoor dat alle data die verstuurd wordt van client (klant en dealer) naar server (AAS) en van server naar client geëncrypteerd wordt. Op die manier kan iemand die onderweg de data onderschept, nog steeds de gevoelige data niet lezen.

Authentication

Bij de ASP.NET toepassing zal gewerkt worden zonder cookies omdat het gebruik van cookies niet mogelijk is bij sommige mobiele toestellen. Ook voor de niet-mobiele versie gebruiken we geen cookies omdat deze kunnen geblokkeerd worden door bepaalde instellingen en firewalls. Op die manier is de kans dat sommige klanten problemen hebben met de toepassing het kleinst.

Aangezien we te maken hebben met externe gebruikers kunnen we geen Windows authentication gebruiken. We zullen ook geen gebruik maken van Passport authentication aangezien hiervoor de Passport SDK op de server

moet geïnstalleerd worden. We opteren hier dus voor Forms authentication waarbij de gebruikers en hun paswoord in de database opgeslagen worden. Alle gebruikte paswoorden worden gehashed opgeslagen. Op die manier kan het paswoord ook niet gelezen worden vanuit de database. Voor de dealer toepassing wordt de hashfunctie lokaal uitgevoerd. Bij de ASP.NET klant-toepassing wordt er gehashed op de server omdat mobiele toestellen geen clientside scripts toelaten. Dit is echter geen probleem aangezien alle communicatie geëncrypteerd wordt dankzij het gebruik van SSL.

Om er voor te zorgen dat de dealer het paswoord van de klant niet weet zal geopteerd worden om volgende methode te gebruiken: wanneer de dealer een nieuwe klant toevoegt wordt automatisch een paswoord gegenereerd (hoe dit concreet aangepakt zal worden, wordt besproken in sectie 5.2.1 op pp. 64-66). Dit paswoord wordt op de server van AAS gehashed en opgeslagen in de database. Vervolgens wordt dit paswoord, samen met de login van de klant die gekozen wordt door de dealer, naar de klant gestuurd via email. De klant kan zich met zijn nieuwe login en paswoord aanmelden bij de ASP.NET toepassing. In deze toepassing kan de klant desgewenst zijn paswoord wijzigen. Is de klant zijn paswoord vergeten, neemt hij contact op met de dealer die in zijn toepassing kan kiezen om de klant een nieuw paswoord te sturen (dit gebeurt uiteraard op dezelfde manier als bij de creatie van een nieuwe gebruiker).

Webservice

Een webservice is in principe een publiek gegeven. Als iemand de juiste URL heeft, kan hij de webservice aanspreken. Dit is niet de bedoeling in dit eindwerk. Vandaar dat hier geopteerd werd om de webservice te beveiligen met een paswoord. Wanneer de dealer zijn programma opstart, wordt naar zijn login en paswoord gevraagd. Op dit moment worden deze gegevens gecontroleerd met de database. Als deze gegevens correct zijn wordt de gebruiker geauthenticeerd bij de webservice. Dit gebeurt aan de hand van zijn login en paswoord die meegegeven worden in de SOAP-header van de webservice. Hoe dit concreet geïmplementeerd zal worden, wordt getoond in sectie 5.2.2 (pp. 66-68).

Op deze manier kan dus niemand zonder correct paswoord gegevens opvragen van de webservice. Daarnaast kan, dankzij de SSL-encryptie, een onbevoegd persoon onderschepte gegevens niet lezen zonder deze eerst te decrypteren.

Consistentie data

Om consistentie van gegevens te garanderen zal gebruik gemaakt worden van T-SQL. Alle acties op de database die atomair moeten zijn, worden uitgevoerd aan de hand van transacties. Deze transacties kunnen enkel als één geheel uitgevoerd worden. Mislukt een deel van de actie dan wordt de hele transactie ongedaan gemaakt. Dit zal bijvoorbeeld gebruikt worden om er voor te zorgen dat wanneer een klant de artikelen uit zijn winkelmand bestelt het nooit kan gebeuren dat een bepaald artikel besteld wordt maar toch nog in zijn winkelmand aanwezig blijft. Op die manier wordt er vermeden dat een klant een artikel foutief twee maal kan bestellen.

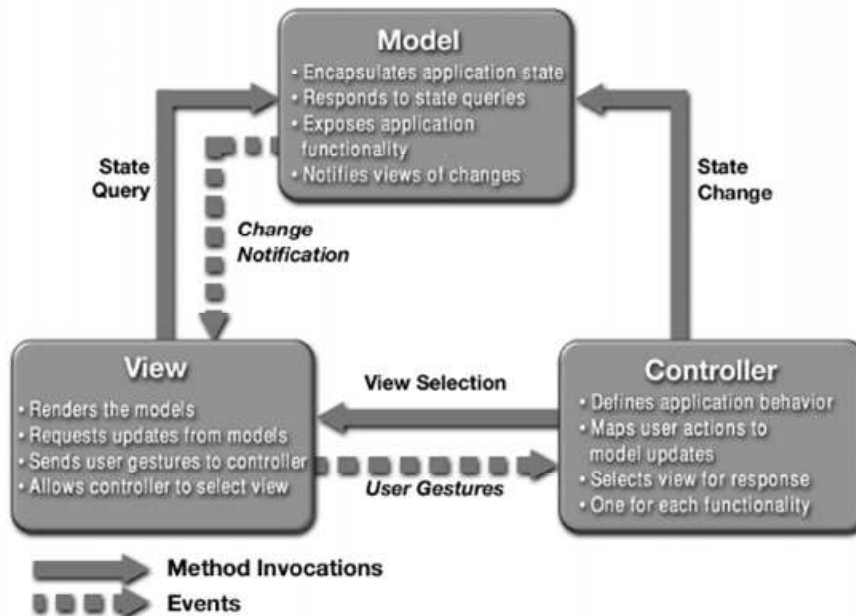
3.1.6 Drie lagen model

Er werd voor deze toepassing geopteerd om gebruik te maken van het Model-View-Controller model (zie figuur 3.3 op pagina 32) of drie lagen model [38],[10]. Dit zal ervoor zorgen dat de implementatie van de verschillende grafische user interfaces eenvoudig zal verlopen. Dit model zorgt er immers voor dat de logica van de toepassing volledig gescheiden is van de grafische interface.

Het *model* is een representatie van de gegevens waarop de toepassing steunt. Dit model bestaat uit de basisklassen die alle business logic bevatten. De *view* zet het model om in een interface die geschikt is om mee te interageren. In een webapplicatie zijn dit bijvoorbeeld de HTML-pagina's. De *controller* is ten slotte het onderdeel dat reageert op gebeurtenissen zoals gebruikersacties. De controller veroorzaakt veranderingen in het model en daarmee eventueel in de view. Een centrale database is dikwijls het kernelement van de controller.

3.1.7 Mobiel en niet-mobiel

Vanuit de eisen van het project werd gekozen om de klanttoepassing beschikbaar te maken op zowel vaste toestellen als mobiele toestellen. Om dit mogelijk te maken werd gekozen om een ASP.NET Mobile Web applicatie te implementeren. Dit soort toepassing kan zowel op gewone browsers bekeken worden als op browsers voor mobiele toestellen (zoals Pocket Internet Explorer op PocketPC). Om rekening te houden met de beperkte bronnen, zoals de omvang van het scherm, zijn de mogelijkheden van dit soort toepassingen beperkt. Daarom werd er gekozen om twee versies te voorzien van de grafische user interface van de klanttoepassing. Op die manier kan voor zowel desktops als mobiele toestellen een optimale versie voorzien worden.



Figuur 3.3: Model-View-Controller [10]

Voor de niet-mobiele versie werd ten volle gebruik gemaakt van de grootte van een standaard computerscherm. Elke pagina bevat op die manier alle relevante informatie waardoor het aantal paginawissels tot een minimum beperkt wordt. Dit zorgt ervoor dat de gebruiker minder frequent van pagina moet wisselen en dus dat er minder frequent gegevens moeten opgehaald worden (wat de performantie ten goede komt). Bij de mobiele versie werd vooral rekening gehouden met het kleine scherm en de moeilijkheid om tekst in te geven. Er werd vooral voor gezorgd dat er zo weinig mogelijk nood is aan scrollen. De inhoud werd dus beperkt tot de grootte van een scherm van een mobiel toestel. Daarnaast werd input van tekst zo veel mogelijk vermeden.

Voor de implementatie van dit onderdeel verwijzen we naar sectie 5.9 (p. 78). Daar zal eveneens beschreven worden aan de hand van welk criterium beslist wordt om van de mobiele versie over te schakelen op de niet-mobiele.

3.1.8 Real-time

Het doel van die eindwerk is om het huidig bestelproces te verbeteren. Momenteel gebeurt dit proces hoofdzakelijk via de telefoon en dus nagenoeg onmiddellijk. Het programma dat dit proces vervangt moet dus in real-time werken: als de klant een bestelling plaatst moet de dealer hier onmiddellijk (of bijna onmiddellijk) op kunnen reageren.

Dit real-time aspect zal in deze tool opgevangen worden door een zekere vorm van determinisme. Dit wil zeggen dat het systeem bepaalde bewerkingen binnen vooraf vastgestelde tijdsintervallen zal uitvoeren. De belangrijkste taak die regelmatig moet herhaald worden is het controleren op acties. Deze acties kunnen drie zaken zijn: een klant heeft een bestelling geplaatst, een klant heeft een artikel uit zijn winkelmand verwijderd (omdat hij het bijvoorbeeld te duur vond) en de klant heeft een vraag gesteld.

In de dealertoepassing werd gekozen om gebruik te maken van twee timers (de concrete implementatie hiervan wordt beschreven in sectie 5.12 op pp. 81-85). De eerste timer staat in voor de controle op acties. Deze timer zal op een vast interval (bijvoorbeeld 10 seconden) een webservice aanspreken. Deze webservice controleert de centrale database en haalt daaruit de nodige informatie over acties. De tweede timer zal er voor zorgen dat bepaalde acties niet te lang op een reactie wachten. Als de eerste timer afgelopen is en er is gebleken dat een actie staat te wachten dan wordt dit aan de dealer duidelijk gemaakt door een bepaald signaal. De tweede timer wordt op dit moment in werking gezet. Heeft de dealer 10 minuten niet naar de wachtende actie(s) gekeken dan zal deze timer de dealer een signaal geven. Negeert de dealer dit signaal, dan wordt dit proces herhaald tot alle acties behandeld zijn.

3.1.9 Consistentie centrale database

Zoals eerder vermeld zal alle nodige data gecentraliseerd opgeslagen worden in de database van AAS in Temse. Nu is het zo dat lokaal bij de dealer soms gegevens gewijzigd worden die eveneens in Temse zullen moeten gewijzigd worden. Dit is het geval als de dealer bijvoorbeeld de brutoprijs van een artikel wijzigt, een artikel verwijdert uit zijn aanbod of een artikel toevoegt. Op dit moment kan de dealer beslissen om deze data door te sturen aan de hand van een functie in zijn deel van de toepassing (zoals verder zal uitgelegd worden). Omdat dit echter een tijdje kan duren en om het werk van de dealers te verlichten zal er echter ook een automatische update voorzien worden.

Voor het updaten van deze gegevens zal een console applicatie geschreven worden. Deze toepassing zal dan geïnstalleerd worden op de SQL server van de dealer. Daar zal de updater gescheduled worden om bijvoorbeeld wekelijks (of dagelijks) 's nachts uitgevoerd te worden. Op die manier is de data op de centrale database in Temse steeds up to date en beschikt de klant steeds over de juiste gegevens.

Er werd voor deze updater gekozen voor een console applicatie omdat deze eenvoudig kan gescheduled worden aan de hand van de Microsoft Windows Task Scheduler (die standaard meegeleverd wordt met de Microsoft Windows besturingssystemen). Een alternatief hiervoor zou kunnen een Windows Service zijn waarin een functie opgenomen is die zelf voor de scheduling zorgt. Dit werd echter niet gedaan omdat dit meer arbeidsintensief is om te ontwikkelen dan de oplossing met een console applicatie en toch hetzelfde resultaat heeft.

3.1.10 Opmaak

Voor zowel de dealertoepassing als de klanttoepassing werd gekozen voor een intuïtieve, zo eenvoudig mogelijke opmaak. Op die manier werd geprobeerd om de toepassing zo eenvoudig mogelijk in gebruik te nemen. Een lange leerperiode voor de gebruikers zou dus moeten overbodig zijn.

Daarnaast zal er voor gezorgd worden dat voor de ASP.NET toepassing alle opmaak is gecentraliseerd in een Cascading Style Sheets (CSS) bestand en dat de basisopbouw van de toepassing zich bevindt in één centrale klasse (PageBase.vb). In deze klasse zal bijvoorbeeld de titelbalk, de menu's, . . . verwerkt worden. De specifieke pagina's worden dan van deze klasse afgeleid. Deze klasse is bijgevoegd in bijlage D (pp. 102-107). Beide zaken zorgen ervoor dat de *look and feel* eenvoudig aangepast kan worden.

3.2 UML modellering

3.2.1 Activiteitendiagramma's

In dit deel zal ik aan de hand van enkele toestandsdiagramma's de algemene werking van het programma proberen duidelijk te maken. In figuur 3.4 en figuur 3.5 (pagina 36 en 37) worden de mogelijke acties van het dealerprogramma weergegeven.

In figuur 3.6 en figuur 3.7 (pagina 38 en 39) worden de mogelijke acties van het klantprogramma weergegeven. Op basis van deze diagramma's zal verder de grafische user interface ontworpen worden.

3.2.2 Klassediagramma

De logica van het programma zal geconcentreerd worden in een aantal basisklassen. Deze klassen werden voorgesteld in het klassediagramma van figuur 3.8 op pagina 40 en kunnen gezien worden als de business logic van de toepassing. Naast deze centrale klassen zullen uiteraard nog extra klassen gemaakt worden die instaan voor de communicatie met de database, de communicatie met de webservices, de grafische user interface,...

3.2.3 Sequentiediagramma's

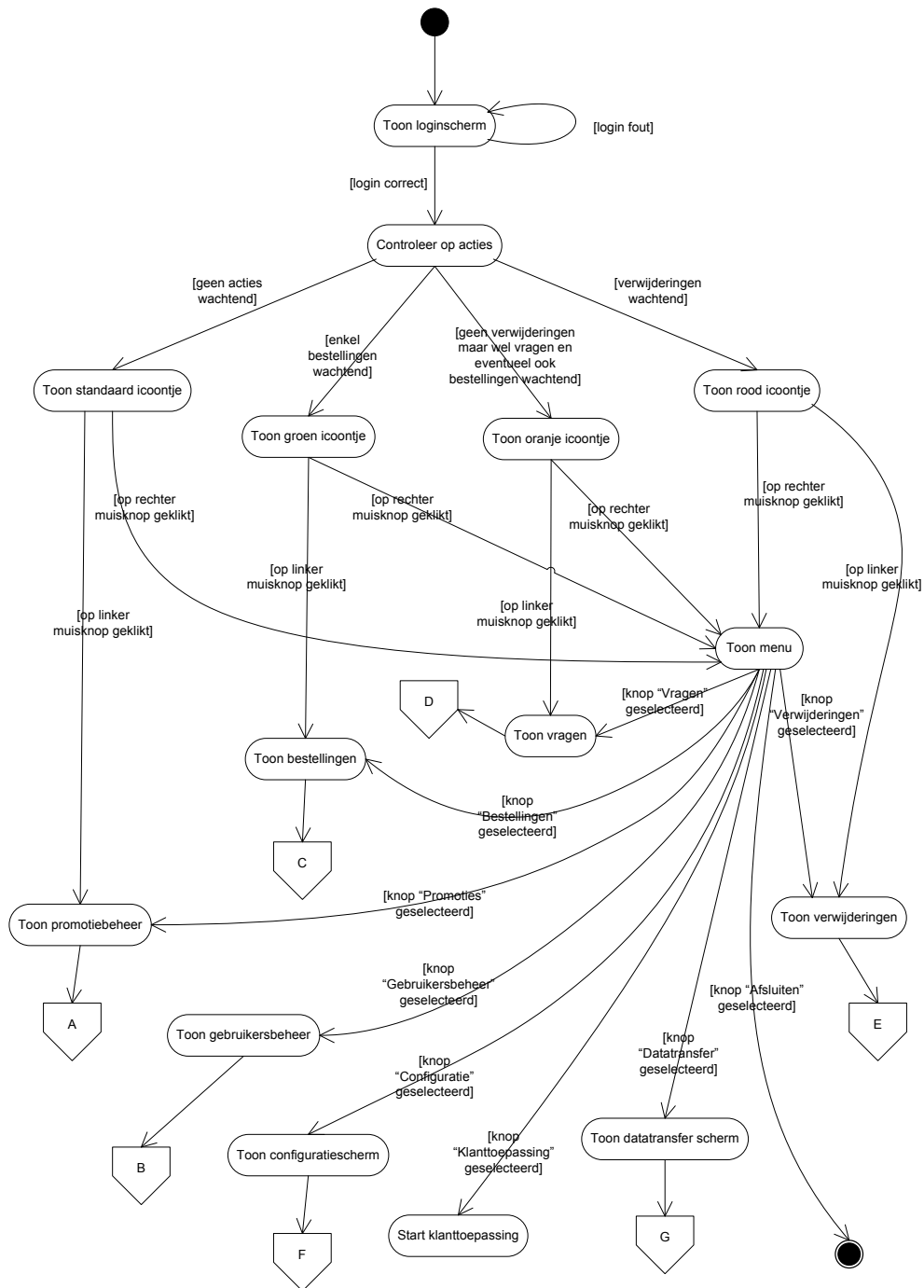
Om enkele zaken in verband met de communicatie tussen de verschillende objecten duidelijk te maken, werden enkele sequentiediagramma's opgesteld (zie figuur 3.9 op pagina 41). Deze maakten het mogelijk om een overzicht te behouden van welke gebeurtenissen optreden bij het plaatsen van een bestelling (figuur bovenaan) en bij het stellen van een vraag aan de dealer (figuur onderaan). Het verloop van het melden van een te dure prijs verloopt op een analoge wijze als bij een vraag.

3.3 Ontwerp grafische user interface

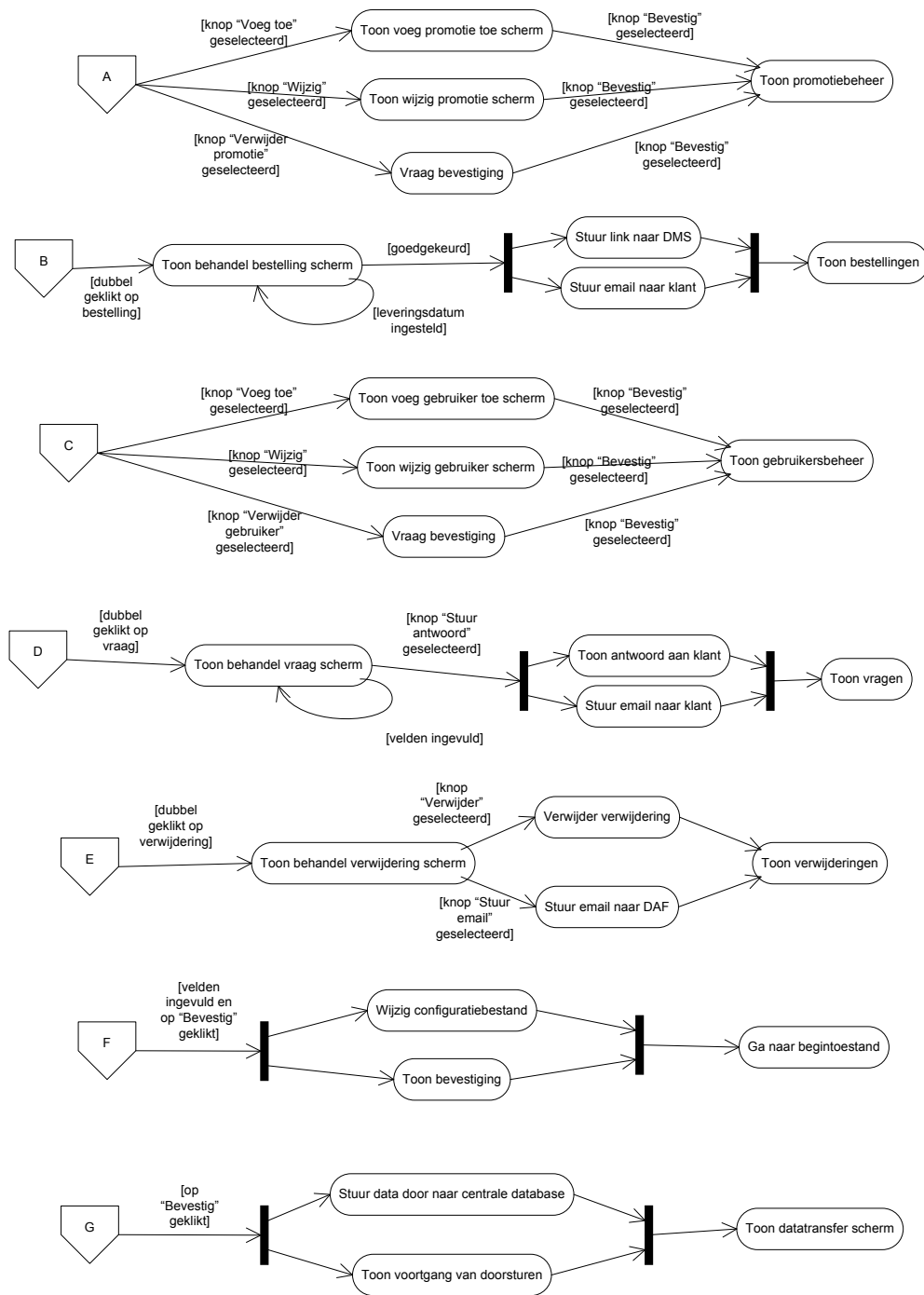
De volledige applicatie wordt in twee grote stukken opgesplitst: een deel voor de klanten (dit is webgebaseerd en zowel te gebruiken op mobiele toestellen als op vaste toestellen met een standaard webbrowser) en een afzonderlijk deel voor de dealer (dit is een Windows toepassing die geïnstalleerd wordt op de pc's van de dealer).

Het deel voor de dealer wordt bij de dealers geïnstalleerd op de pc's van de personen die verantwoordelijk zijn voor de communicatie met de vlootklanten en werkt als volgt:

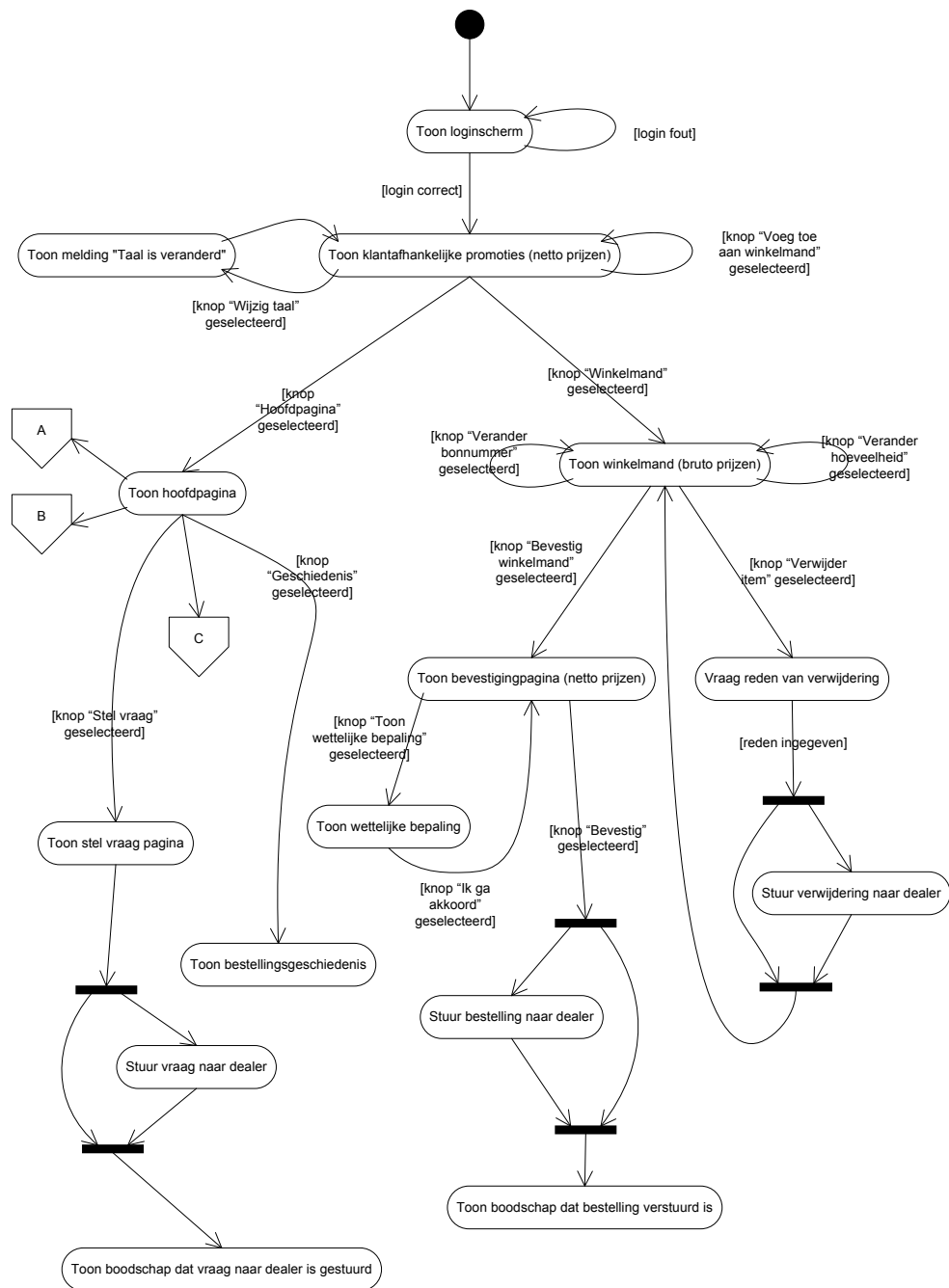
- Naast de klok (in de system tray) ziet de dealer een icoontje. Dit icoontje verandert naargelang de status van de acties. Deze acties (*requests*) treden op wanneer een klant een bepaalde handeling ondernomen heeft. Er zijn drie kleuren:



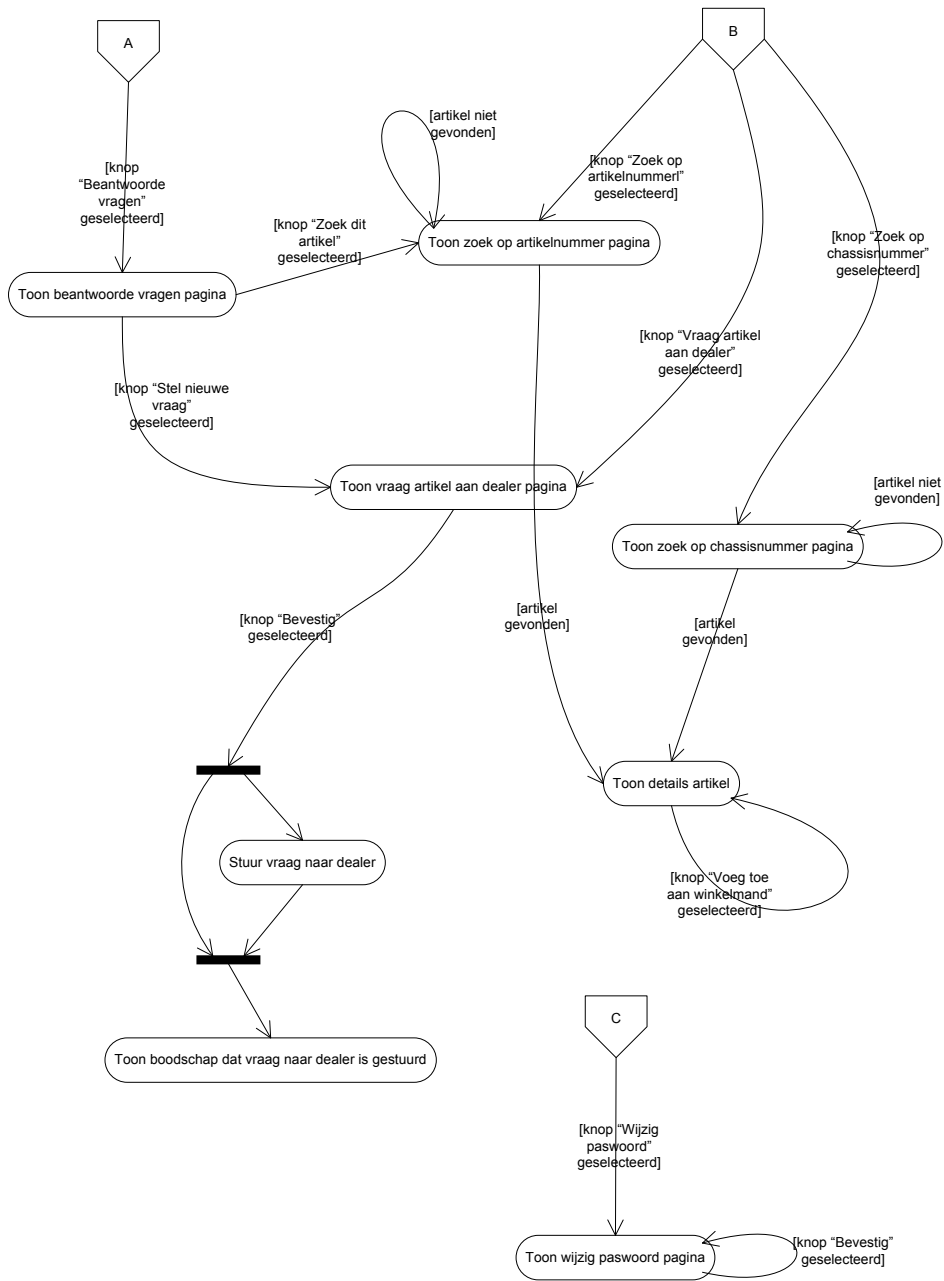
Figuur 3.4: Activiteitendiagramma dealertoepassing (deel 1)



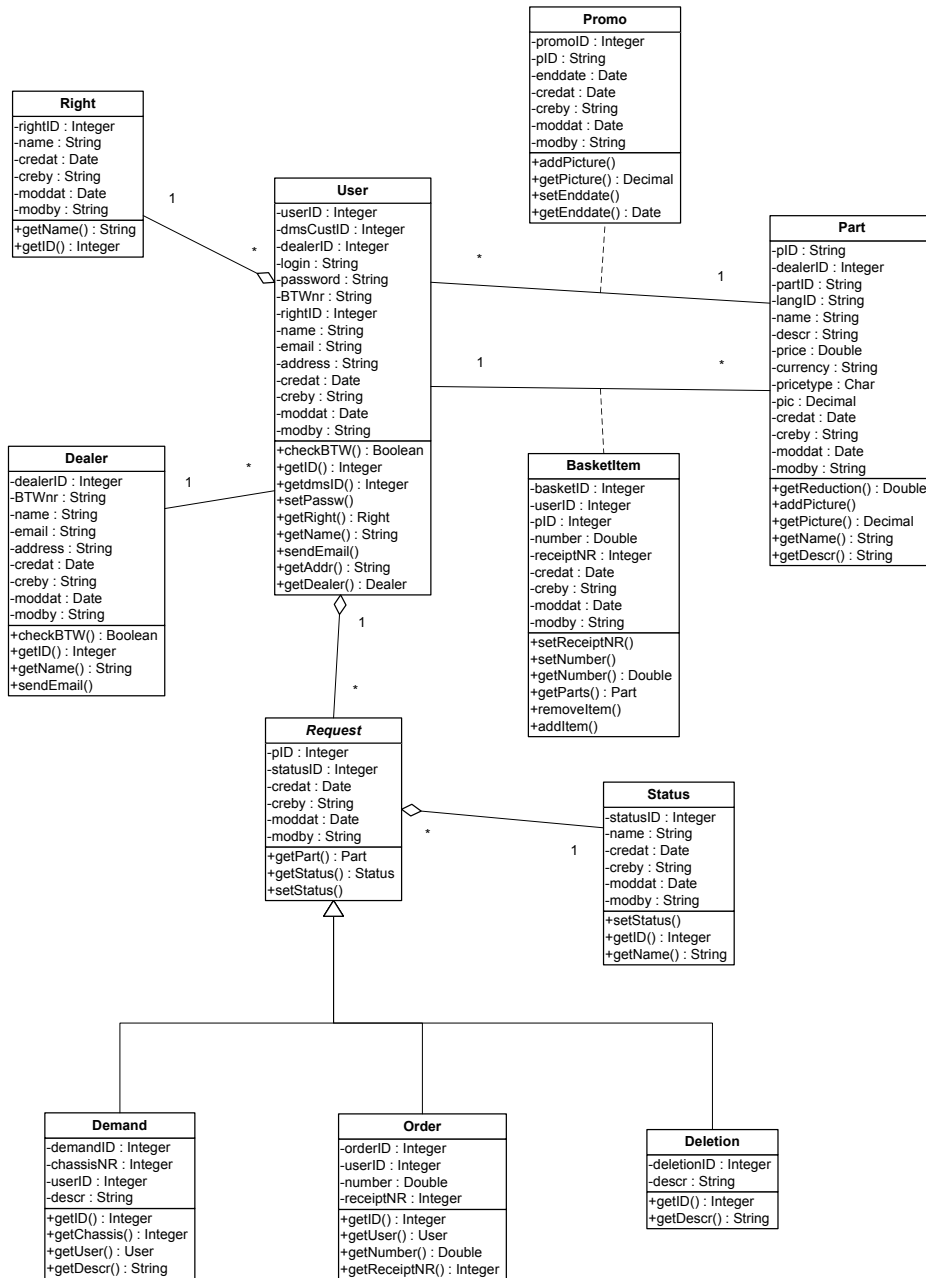
Figuur 3.5: Activiteitendiagramma dealertoepassing (deel 2)



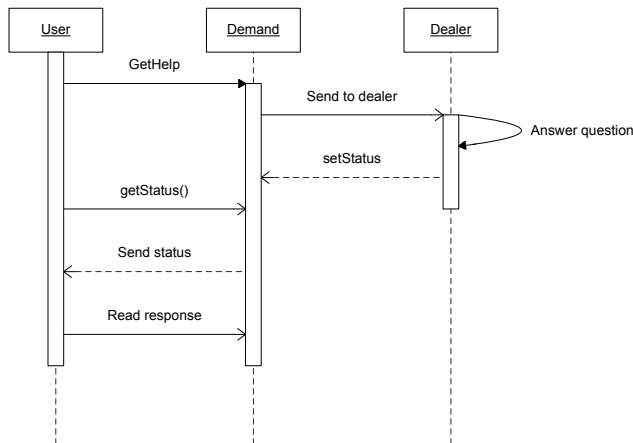
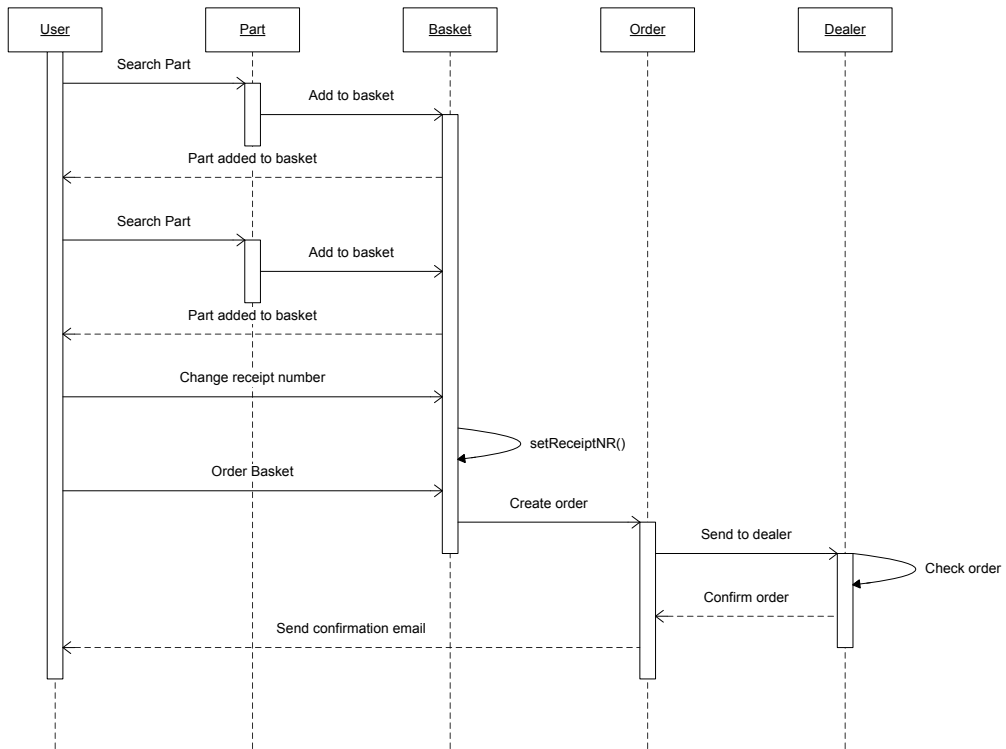
Figuur 3.6: Activiteitendiagramma klanttoepassing (deel 1)



Figuur 3.7: Activiteitendiagramma klanttoepassing (deel 2)



Figuur 3.8: Klassediagramma



Figuur 3.9: Sequentiediagramma's

1. Groen geeft aan dat de klant een bestelling geplaatst heeft die moet afgehandeld worden (hier geeft het programma aan de dealer onmiddellijk het artikelnummer, de hoeveelheid, het type order, . . . mee). Bovendien wordt hier voorzien in een link naar het DMS programma.
2. Oranje geeft aan dat de klant een vraag heeft voor de dealer (dit wil zeggen dat de klant op zoek is naar een artikelnummer of een andere vraag heeft). Hier kan de klant verschillende parameters meegeven zoals chassisnummer, de groep waartoe een artikel behoort, . . .
3. Rood geeft aan dat een klant een artikel in zijn winkelmand geplaatst heeft maar later terug verwijderd heeft (hier geeft het programma mee over welk artikel het gaat en wat de reden is waarom de klant dit artikel niet aangekocht heeft).

Als er geen acties zijn (of alle acties afgehandeld zijn) wordt het standaard icoontje van het programma getoond.

- Wanneer de dealer met de rechter muisknop op het icoontje klikt, krijgt hij een venster te zien met de mogelijkheden van het programma:
 1. Start het klantprogramma op. Hiermee kan de dealer eventueel bestellingen doen in de plaats van de klant.
 2. Start het gebruikersbeheer op. Hiermee kan de dealer gebruikers toevoegen en verwijderen.
 3. Start het promotiebeheer op. Hiermee kan de dealer promoties toevoegen en verwijderen. Bij het toevoegen wordt meegegeven voor welke klant(en) deze promoties gelden en tot welke datum deze promoties lopen. Op de einddatum worden de promoties automatisch uit het programma verwijderd.
 4. Toon de huidige acties. Deze acties staan geordend per klant, per type en in volgorde van prioriteit. Wanneer een actie behandeld wordt, verandert de status, verdwijnt het uit de lijst en wordt de gepaste actie ondernomen naar het klantprogramma.
 5. Toon de programma-instellingen. Hier kan de dealer aan aantal parameters instellen zoals de aanmeldnaam voor zijn lokale SQL server, het paswoord voor de server, . . .
 6. Stuur gegevens door. Hier kan de dealer kiezen om alle nodige gegevens uit zijn lokale database te halen en door te sturen naar de centrale server in Temse.

- Als de gebruiker met de linker muisknop op het icoontje van het programma klikt wordt het meest relevante venster geopend. Als er geen acties staan te wachten is dit het venster om promoties te beheren. Als er bestellingen staan te wachten (maar geen vragen of verwijderingen) is dit het venster waarin deze bestellingen kunnen afgehandeld worden. Als er vragen van klanten staan te wachten (maar geen verwijderingen) is dit het venster dat de behandeling van deze vragen verzorgt. Als er artikels te duur zijn voor klanten is dit ten slotte het venster waarin deze verwijderingen getoond worden. Rood is dus prioritair op oranje en oranje is op zijn beurt prioritair op groen.
- In dit deel wordt er ook voor gezorgd dat er steeds maar één persoon tegelijk een actie kan behandelen. Acties die reeds behandeld worden, zijn aangegeven met een vinkje en kunnen dus niet behandeld worden.

Het deel voor de klanten is gebaseerd op de eerder vermeldde activiteitendiagramma's. Dit ontwerp geldt enkel voor de niet-mobiele versie. De mobiele versie is gelijkaardig maar is aangepast voor de beperkte schermgrootte. Het ontwerp is samen te vatten als volgt:

- Vanuit elk venster kan de klant naar het hoofdvenster, naar de pagina met promoties, uitloggen of naar zijn winkelmandje gaan. In pagina's waar producten staan, is er steeds een mogelijkheid om deze producten aan het winkelmandje toe te voegen.
- De klant ziet (na het ingeven van zijn login en paswoord) onmiddellijk de promoties die voor hem op dat moment gelden met hun nettoprijs. Daarna komt de klant op het hoofdvenster.
- In het hoofdvenster kan de klant een onderdeel zoeken, terug naar de promoties, een vraag stellen aan de dealer, zijn paswoord wijzigen, het antwoord op gestelde vragen bekijken of de geschiedenis van zijn bestellingen bekijken.
- De klant kan op drie manieren een artikel zoeken:
 1. op artikelnummer
 2. op chassisnummer, groep, subgroep en beschrijving
 3. via hulp van de dealer
- Als de klant hulp vraagt aan de dealer (3) wordt de nodige informatie als actie naar het dealerprogramma gestuurd. De kleur van zijn icoontje wordt dan onmiddellijk aangepast.

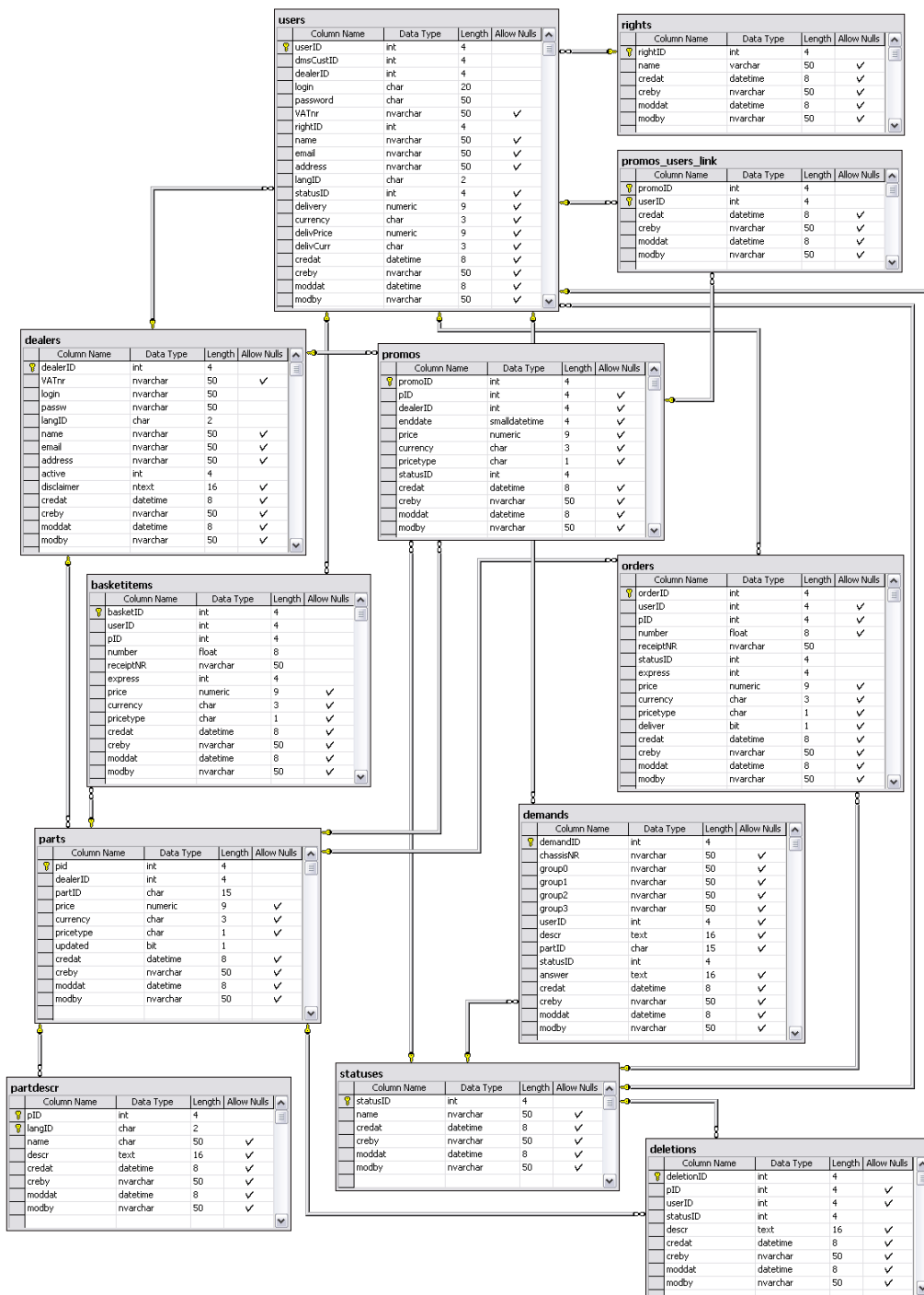
- Naast hulp vragen in verband met een artikelnummer kan de klant van op de hoofdpagina kiezen om een vraag te stellen aan de dealer. Dit kan bijvoorbeeld een vraag naar de nettoprijs van een artikel zijn of de vraag of een bepaald artikel in stock is. Deze vraag wordt eveneens onmiddellijk naar het dealerprogramma gestuurd en de status van zijn programma wordt aangepast. De antwoorden op deze vragen worden zichtbaar gemaakt op een afzonderlijke pagina van het klantprogramma.
- Als de klant een product gevonden heeft, krijgt hij de details van het product te zien (met de brutoprijs) en kan hij dit product in zijn winkelmand plaatsen.
- Als de klant naar zijn winkelmand gaat, kan hij nog producten toevoegen, het geselecteerde aantal wijzigen of het artikel verwijderen. Hij ziet hier de brutoprijs van de producten (of de nettoprijs als het over promoties gaat). In de winkelmand kan hij ook aangeven wanneer hij het product nodig heeft (binnen 24 uur voor Express Order of 48 uur voor Stock Order) en kan hij een bonnummer toevoegen voor zijn bestelling. Wanneer de klant iets uit zijn winkelmand verwijdert moet hij hier een reden voor geven. Als deze reden niet is dat hij het artikel fout gekozen heeft dan wordt de reden als actie naar de dealer gestuurd. De kleur van zijn icoontje wordt dan onmiddellijk aangepast.
- Als de klant zijn winkelmand bevestigt, krijgt hij de nodige wettelijke bepalingen te zien en een lijst met alle producten (met de nettoprijs). Hier ziet de klant de totale nettoprijs van zijn bestelling en ziet hij ook een waarschuwing als de minimumprijs voor levering niet bereikt is. Als hij dan toch kiest voor levering zal hij hiervoor extra moeten betalen.
- Verder kan de klant de bestelling definitief maken. Er wordt dan een bestelling als actie naar de dealer gestuurd. De kleur van zijn icoontje wordt dan onmiddellijk aangepast. Als de bestelling afgehandeld is door de dealer ontvangt de klant een bevestiging van zijn bestelling via email met de nettoprijs en de tijd waarop de artikels zullen geleverd of afgehaald kunnen worden.

3.4 Ontwerp database

Om aan alle eisen te voldoen werd gekozen om te voorzien in één enkele database die zich op de SQL server van AAS in Temse zal bevinden. In deze

database wordt alle data die nodig is voor de toepassing verzameld. Dit zijn onder andere de gegevens van de onderdelen van de dealers, de klantgegevens, de opgetreden acties,...

De gegevens die nodig zijn uit de databases van de dealers worden op regelmatige tijdstippen ook in de database bij AAS verzameld. Deze database zal een structuur hebben zoals voorgesteld in figuur 3.10 op pagina 46.



Figuur 3.10: Model database

Hoofdstuk 4

Gebruikte technologieën

4.1 Servers

4.1.1 SQL Server 2000

Microsoft SQL Server is een relationele database server waarvan Transact-SQL (T-SQL) de primaire querytaal is. T-SQL is een implementatie van de ANSI/ISO standaard Structured Query Language (SQL) die zowel gebruikt wordt door Microsoft als door Sybase [37]. T-SQL voegt een aantal mogelijkheden toe aan de standaard (SQL92, goedgekeurd in 1992) die vooral gebruikt kunnen worden in stored procedures. Microsoft SQL Server maakt het eveneens mogelijk om vanuit het ontworpen databasemodel een diagramma op te stellen dat dan automatisch vertaald wordt in de nodige tabellen en relaties tussen tabellen. Op die manier werd de implementatie van de database een stuk versneld.

Er werd gekozen voor versie 2000 omdat in september de nieuwste versie (2005) nog niet beschikbaar was. Er werd geopteerd om niet over te schakelen op een nieuwe versie tijdens het proces. Bovendien vereist vlot gebruik van de nieuwste versie meer dan 512Mb fysisch geheugen wat niet beschikbaar was op de computer waarop dit eindwerk ontwikkeld werd. Daarnaast is het zo dat zowel de dealers als AAS momenteel gebruik maken van SQL Server 2000 waardoor de keuze hier voor een stuk vast lag.

4.1.2 Internet Information Services

Microsoft Internet Information Services (IIS) is een verzameling internetgebaseerde diensten voor servers die Microsoft Windows gebruiken [36]. IIS is één van de meest gebruikte servers en biedt momenteel diensten voor FTP,

SMTP, NNTP en HTTP/HTTPS. Deze server wordt gebruikt door AAS en is de standaard server voor het gebruik van ASP.NET toepassingen.

Tijdens de ontwikkelingsfase van dit eindwerk werd gebruik gemaakt van IIS versie 5.1 omdat deze versie inbegrepen is in het besturingssysteem Microsoft Windows XP Professional. Bij AAS wordt gebruik gemaakt van Windows Server 2003. Dit besturingssysteem bevat IIS versie 6.0 en deze versie zal dus gebruikt worden bij het in werking stellen van dit eindwerk.

4.1.3 Citrix Presentation Server

Citrix Systems is een bedrijf dat ontstaan is in Fort Lauderdale, Florida. Dit bedrijf is gesticht in 1989 door ex-IBM ontwikkelaar Ed Iacobucci [34]. Dit bedrijf staat bekend omwille van een aantal softwareproducten die vooral te maken hebben met beveiliging en content management. Eén van hun meeste gekende producten is Citrix Presentation Server (vroeger Citrix MetaFrame). Dit pakket wordt door een aantal dealers gebruikt waardoor het dealerprogramma op deze servers zal moeten geïnstalleerd worden.

Citrix Presentation Server zorgt ervoor dat gewone pc's kunnen gebruikt worden als terminals [33],[22]. Dit systeem wordt gebruikt omwille van het feit dat toepassingen enkel op de server moeten geïnstalleerd worden en dan gebruikt kunnen worden op alle clients. Hierdoor moeten er minder licenties aangekocht worden en is het beheer centraal. Daarnaast is het zo dat het verkeer tussen clients en server tot een minimum beperkt wordt waardoor de lijnen tussen de server en clients geen hoge capaciteit moeten hebben (en dus in prijs kunnen beperkt worden en zich eventueel op grote afstand van elkaar mogen bevinden). Dit alles wordt mogelijk gemaakt door het gebruik van Independent Computing Architecture (ICA, dit is een protocol voor thin clients ontworpen door Citrix Systems). Dit protocol stuurt scherm informatie op een hoog niveau door en dus niet enkel grafische informatie (dit is vergelijkbaar met het X11 protocol).

Kort samengevat kunnen we dus stellen dat hierdoor een mainframe-terminal architectuur gesimuleerd wordt. Hierbij wordt het meeste rekenwerk en verwerking gedaan op de server (mainframe) en wordt de grafische user interface verzorgd door de clients (terminals) die dus geen krachtige machines moeten zijn. Citrix Presentation Server clients zijn beschikbaar voor meerdere besturingssystemen zoals Microsoft Windows (zowel 16 als 32 bit), Mac OS, Linux en andere Unix gebaseerde systemen. Daarnaast bestaat er ook een webgebaseerde client die beschikbaar is over een beveiligde ICA proxy (wanneer dit

gecombineerd wordt met Citrix Secure Gateway gebeurt dit via HTTPS). De server zelf wordt gehost op een Microsoft Windows platform. Dit kan één enkele server zijn of uit een cluster bestaan.

4.2 Softwarepakketten

4.2.1 Visual Studio .NET 2003

Voor dit eindwerk werd gebruik gemaakt van de Microsoft Visual Studio programmeeromgeving. Deze omgeving laat immers toe om zowel ASP.NET toepassingen, als Windows toepassingen, XML Webservices, console applicaties als setup projecten (installers) te ontwikkelen. Er werd gebruik gemaakt van de .NET 2003 versie omdat de nieuwste versie (2005) pas beschikbaar was vanaf januari 2005 en er geopteerd werd om tijdens het proces niet om te schakelen.

Visual Studio beschikt over een aantal mogelijkheden die tijdens de ontwikkeling van dit eindwerk werden gebruikt. In eerste instantie biedt deze omgeving een teksteditor met syntax highlighting, Intellisense, . . . Deze editor (en de volledige omgeving) biedt ondersteuning voor Visual Basic .NET, C++, C#, J#, CSS, XML, ASP.NET, WML, XHTML en HTML. Daarnaast biedt de omgeving eveneens een wysiwyg (*What you see is what you get*) editor voor gebruikersinterfaces. Verder is ook de debugger een niet te ontberen middel voor de ontwikkeling van software. Ten slotte werd ook gebruik gemaakt van de ondersteuning voor mobiele toestellen en de bijhorende emulatoren voor mobiele toestellen (zoals PocketPC's).

Naast de bovenvermelde programmeertalen ondersteunt Visual Studio het .NET Framework 1.1. Omwille van al deze zaken is Microsoft Visual Studio .NET de standaard voor het ontwikkelen van .NET gebaseerde Windowstoepassingen en webapplicaties. De keuze voor deze omgeving lag dan ook voor de hand.

4.2.2 Microsoft Visio 2003 Professional

Microsoft Visio is een softwarepakket dat behoort tot de Microsoft Office suite. Visio maakt het mogelijk om allerlei soorten diagramma's op te stellen. Dit pakket werd gebruikt om een planning voor te stellen aan de hand van een Gantt Chart en om de modellering van dit eindwerk op te stellen. Voor dit laatste werd vooral gebruik gemaakt van Unified Modeling Language (UML).

Dit zal verder besproken worden in sectie 4.3.8 (p. 57).

4.2.3 Macromedia RoboHelp X5

Macromedia RoboHelp is een Help Authoring Tool (HAT) die het mogelijk maakt om op een snelle manier helpsystemen en documentatie te maken bij zowel desktop applicaties als internet applicaties [3]. RoboHelp maakt het mogelijk om een systeem te ontwikkelen dat onder andere kan bestaan uit help onderwerpen, een inhoudstabel, een index, een glossarium, contextafhankelijke help, . . . Dit pakket kan een helpfunctie uitvoeren in een aantal online helpformaten en biedt ook de mogelijkheid om printklare documentatie te genereren.

Een testversie van deze commerciële tool (die momenteel in handen is van Adobe Systems na hun overname van Macromedia) werd gebruikt om een helpfunctie te creëren voor zowel de klanttoepassing als de dealertoepassing. Deze helpfuncties bieden een overzicht van de mogelijke functies en de manier waarop deze gebruikt kunnen worden.

4.2.4 Macromedia Captivate en Wink

Macromedia Captivate is een softwarepakket voor *screencasting* [2]. Een screencast is een digitale opname van de output van een computerscherm. In essentie is dit dus eigenlijk een opeenvolging van screenshots. Dit soort filmpjes wordt vooral gebruikt voor het maken van tutorials en helpfuncties. Macromedia Captivate maakt het mogelijk om deze filmpjes onder andere uit te voeren als Compiled Macromedia Flash bestanden (.swf) die compact zijn en dus heel geschikt om ook via internet te verspreiden. Bij deze filmpjes kan bovendien op een eenvoudige manier audio toegevoegd worden, bepaalde zaken duidelijk gemaakt worden aan de hand van tekstballonnen, . . . Daarnaast biedt Macromedia Captivate een eenvoudige integratie met Macromedia RoboHelp voor het maken van helpfuncties.

DebugMode Wink is een gelijkaardig softwarepakket als Macromedia Captivate maar is gratis te downloaden voor zowel Windows als Linux platformen [45],[12]. Beide programma's (een trial versie van Macromedia Captivate en Wink 2.0 voor Windows) zullen gebruikt worden om een aantal filmpjes op te nemen die verwerkt werden in de helpfunctie. Een aantal handelingen worden hiermee verduidelijkt voor de gebruikers. Ze tonen stap voor stap hoe bepaalde zaken kunnen verwezenlijkt worden en kunnen dikwijls veel sneller duidelijk maken hoe bepaalde handelingen werken. Deze filmpjes kunnen

eveneens als tutorial gebruikt worden.

4.2.5 Openwave V7 Simulator

De Openwave Phone Simulator is een simulator van smartphones met browser [23]. Hoewel er heel wat gratis simulators van dit type beschikbaar zijn werd gekozen om de Openwave simulator te gebruiken omwille van de gebruiksvriendelijkheid en de eenvoudige integratie van dit product in Microsoft Visual Studio.

Hoewel de bruikbaarheid van de ASP.NET Mobile Web applicatie ook getest werd op deze simulator werd voor dit deel van de applicatie vooral gebruik gemaakt van een PocketPC Emulator. Deze emulator maakt immers deel uit van de Visual Studio .NET 2003 programmeeromgeving en is zeer eenvoudig in gebruik.

4.2.6 WinShell en MikTeX

Deze beide open source tools maakten het mogelijk om \LaTeX te gebruiken op het Windows platform [21]. MikTeX is een \TeX/\LaTeX distributie voor Microsoft Windows die de nodige compilers en tools bevat om het omzetten van .tex bestanden naar .ps, .dvi of .pdf bestanden mogelijk te maken. Dit pakket bevat onder andere ook *BibTex* en *MakeIndex*, beide werden gebruikt voor dit eindwerk en zullen verder besproken worden.

WinShell is een teksteditor met syntax highlighting voor .tex files. Daarnaast bevat deze editor enkele eenvoudige hulpmiddelen om het creëren en compileren van deze files eenvoudiger te maken. WinShell maakt gebruik van de MikTeX omgeving om dit alles mogelijk te maken.

4.2.7 IconCool Studio v1.6

IconCool Studio van Newera Software Technology is een softwarepakket dat het mogelijk maakt om op een eenvoudige manier icoontjes te ontwerpen [24]. Dit programma biedt onder andere de mogelijkheid om een bestaande figuur in te voegen en om te vormen tot een icoontje met hoge resolutie.

Een testversie van dit product werd gebruikt om de icoontjes van het dealerprogramma op te stellen. Hoewel Visual Studio eveneens de mogelijkheid biedt om icoontjes te creëren, biedt dit programma een uitgebreider gamma

aan hulpmiddelen om icoontjes te maken die compatibel zijn met de ondersteuning van Windows XP voor hoge resolutie icoontjes.

4.2.8 Microsoft .NET framework 1.1

Het Microsoft .NET framework is een component van het Microsoft Windows besturingssysteem die voor het eerst beschikbaar werd gemaakt in 2002 [39]. Dit framework voorziet in een aantal voorgeprogrammeerde functies die veel gebruikt worden. Deze functies vormen een bibliotheek en bieden functies in zaken zoals de user interface, dataconnecties, encryptie, algoritmen,...

Daarnaast beheert het framework het uitvoeren van programma's die specifiek geschreven zijn voor dit framework. Programma's geschreven voor het .NET framework worden uitgevoerd in een software-omgeving die de Common Language Runtime (CLR) genoemd wordt. De CLR voorziet een soort virtuele machine (analoog aan de Java Virtual Machine) die er voor zorgt dat bij het programmeren geen rekening moet gehouden worden met de specifieke kenmerken van de machines waarop de software zal uitgevoerd worden. Bovendien voorziet de CLR eveneens in diensten zoals beveiliging, geheugenbeheer en foutafhandeling.

Samen met de bibliotheek vormt de CLR dus het .NET framework. Dit alles maakt het eenvoudiger om toepassingen te creëren en zorgt ervoor dat kwetsbaarheid en beveiligingsrisico's tot een minimum beperkt worden. Hoewel in november 2005 versie 2.0 van het framework beschikbaar gemaakt is werd voor dit eindwerk gebruik gemaakt van versie 1.1 om problemen bij het overschakelen te vermijden. Delen van dit eindwerk waren immers al gemaakt voor de release van de nieuwe versie.

4.3 Gebruikte talen

4.3.1 Visual Basic .NET

Visual Basic .NET (VB.NET) is een object-georiënteerde programmeertaal [43]. Deze taal is de adaptatie van Microsofts Visual Basic (VB) voor het .NET framework. Deze taal werd oorspronkelijk uitgebracht in 2002, tegelijk met ASP.NET en de nieuwe programmeertaal C# (het tegenwoord van Microsoft op Java). Visual Basic .NET staat net als zijn voorganger bekend om de eenvoud van programmeren, die te wijten is aan de intuïtieve syntax.

Deze taal werd gebruikt voor zowel de Windows applicatie, de console applicatie, de setup projecten (installers) voor beide voorgaande als voor de code-behind van de ASP.NET applicatie. Er werd voor deze programmeertaal gekozen omdat bij AAS vooral gebruik gemaakt wordt van VB6 en VB.NET en het dus eenvoudiger is voor hen om later nog aanpassingen te doen in deze taal.

Omdat deze programmeertaal voor mij ongekend was op het moment dat dit eindwerk aangevangen werd, doorliep ik in september een leerperiode om mij deze taal eigen te maken. Hierbij maakte ik vooral gebruik van [6], [20], [9], [27] en [15].

4.3.2 T-SQL

Zoals eerder vermeld is Transact-Structured Query Language (T-SQL) een uitbreiding op SQL [40]. De uitbreidingen bevinden zich op een aantal verschillende vlakken. Een eerste uitbreiding heeft te maken met *control-of-flow*. Hieronder verstaan we bijvoorbeeld mogelijkheden om conditionele acties toe te staan. Voorbeelden hiervan zijn onder andere IF...ELSE statements en WHILE lussen.

Een volgende belangrijke uitbreiding is de mogelijkheid om lokale variabelen te definiëren. Het sleutelwoord DECLARE maakt het mogelijk om in T-SQL procedures gebruik te maken van tijdelijke hulpvariabelen. Dit maakt het opstellen van ingewikkelde queries een stuk eenvoudiger.

Naast deze twee belangrijke uitbreidingen zijn er ook nog uitbreidingen die te maken hebben met Microsoft Windows geïntegreerde gebruikersauthenticatie en enkele functies die ondersteuning bieden voor het verwerken van strings, datums en wiskundige formules.

Het gebruik van T-SQL is verbonden met de keuze van het gebruik van Microsoft SQL Server. Aangezien communicatie met de database een belangrijke rol speelt in alle onderdelen van dit eindwerk werd veelvuldig gebruik gemaakt van T-SQL. Dit gaat van eenvoudige SELECT statements tot meer gecompliceerde transacties. Hoewel deze taal reeds voor een stuk gekend was, werd veel geleerd uit [19].

Voor veel van het dataverkeer werd ook gebruik gemaakt van stored procedures. Dit zijn functies die op de database server zelf gedefinieerd zijn en dan ook op de server uitgevoerd worden. Dit zorgt ervoor dat een stuk

van de verwerking op de meer krachtige server gebeurt in plaats van op de clientcomputer. Het gebruik van stored procedures zorgt eveneens voor een tijds winst aangezien SQL server deze procedures vooraf compileert en op die manier de verwerking bij gebruik beperkt tot een minimum.

4.3.3 ADO.NET

ADO.NET is de versie van ActiveX Data Objects (ADO) ontwikkeld voor het .NET framework en is het primaire model voor datatoegang voor .NET toepassingen [30]. ADO.NET bestaat uit twee grote delen: de *data provider* en de *dataset*.

De data provider bestaat uit een aantal klassen die het mogelijk maken om te communiceren met een database. Deze zaken zijn de verbinding (die instaat voor een connectie met de databron), de opdracht (een statement dat een bepaalde actie oproept, zoals lezen of schrijven), parameter (een waarde die kan meegegeven worden aan de opdracht), een data-adapter (een soort brug om data te verplaatsen tussen de databron en de dataset) en een data-reader (een object dat gebruikt wordt om grote lijsten data record voor record op te halen uit de databron zonder deze te moeten opslaan).

De dataset objecten is een groep klassen die een eenvoudige relationele database in het geheugen beschrijven. Een dataset representeert dus een volledige database. De set kan verschillende tabellen bevatten en relaties tussen deze tabellen. Elk van deze tabellen kan op zich meerdere kolommen en rijen bevatten. Relaties tussen tabellen kunnen bijvoorbeeld primaire sleutel - vreemde sleutel relaties zijn. Daarnaast kan een dataset ook bepaalde regels (*constraints*) bevatten. Deze zorgen er bijvoorbeeld voor dat een primaire sleutel steeds uniek is of dat er geen verwijder- of toevoegfouten optreden.

Voor de communicatie met de database werd in dit eindwerk gebruik gemaakt van ADO.NET onder de vorm van de *System.Data.SqlClient* namespace. Naast deze connectievorm bestaan er ook meer algemene verbindingsvormen zoals OLE DB en ODBC. Hier werd echter gekozen voor een SQL-connectie omdat deze functies specifiek geoptimaliseerd zijn voor het gebruik met een Microsoft SQL database. Het nadeel van deze keuze is dat het achteraf moeilijker is om de huidige database te vervangen door een andere database (zoals een Oracle of MySQL database). Omdat hier echter niet direct plannen voor zijn, werd met dit nadeel geen rekening gehouden.

4.3.4 Microsoft ASP.NET

ASP.NET is de opvolger van Microsofts Active Server Pages (ASP) en is een deel van het .NET framework [31]. ASP.NET is een set van technologieën voor de ontwikkeling van webgebaseerde toepassingen, dynamische websites en XML web services. ASP.NET code kan geschreven worden in gelijk welke .NET taal (VB.NET, C# of JScript .NET) maar ook in andere talen zoals Perl en Python. Het voordeel van ASP.NET is dat de server-side code vooraf gecompileerd wordt in een aantal DLL-bestanden op de web server. Dit zorgt dus voor een tijdswinst in vergelijking met script-gebaseerde technologieën.

ASP.NET controls zijn vergelijkbaar met controls voor Windows toepassingen in die zin dat eigenschappen en events aan de control kunnen toegewezen worden in de code. In tegenstelling tot Windows controls die weten hoe ze zichzelf moeten tekenen, genereren ASP.NET controls stukken HTML code en JavaScript die dan vertaald worden door de browser van de gebruiker.

De vele verschillende beschikbare .NET controls, klassen en hulpmiddelen zorgen ervoor dat het programmeren van webtoepassingen een stuk sneller en eenvoudiger kan. In combinatie met ADO.NET is ook de communicatie met een relationele database een stuk eenvoudiger dan met traditionele scriptgebaseerde technologieën zoals ASP en PHP.

Omdat ik ook deze taal niet kende bij de aanvang van dit eindwerk, werd een stuk van mijn beschikbare tijd besteed om kennis op te doen van deze taal. Dit werd onder andere gedaan aan de hand van [29], [16] en [13]. Er werd steeds gebruikt gemaakt van de code-behind methode bij het schrijven de ASP.NET toepassing. Dit houdt in dat de ASP.NET code (die de controls definieert en een plaats geeft ten opzichte van elkaar) in een .aspx bestand opgeslagen werd en de VB.NET code (die instaat voor de acties die bij de controls horen) in een .aspx.vb bestand. Dit laatste bestand noemt men de code-behind. Deze methode zorgt ervoor dat er op een overzichtelijke manier rekening kan gehouden worden met het Model-View-Controller model.

4.3.5 XML en SOAP

Zoals reeds eerder vermeld zal voor de communicatie tussen de Windows toepassing voor de dealers en de ASP.NET toepassing voor de vlootklanten gebruik gemaakt worden van XML Webservices. Voor het aanleren van deze technologie werd vooral gebruikt gemaakt van [18], [16], [29] en [14]. Een web service is een softwaresysteem dat ontwikkeld is om eenvoudige commu-

nicatie tussen twee machines over een network mogelijk te maken [44]. Dit systeem heeft een interface die geschreven is in een formaat dat door machines kan verwerkt worden. Een voorbeeld van zo'n interface is Web Service Description Language (WSDL).

XML Webservices worden gehost op een IIS en zijn aanspreekbaar door heel wat verschillende machines en vanuit heel wat verschillende programmeertalen. Boodschappen die naar een webservice gestuurd worden en van een webservice komen worden in XML gestuurd over HTTP of HTTPS en zijn verpakt in SOAP enveloppes. Praktisch gezien bestaat een webservice dus uit een aantal methodes die van op afstand aanspreekbaar zijn.

4.3.6 Cascading Style Sheets

Cascading Style Sheets (CSS) is een taal voor *stylesheets* die gebruikt wordt voor de opmaak van een document dat geschreven is in een markup taal [32]. De meest gebruikte toepassing is voor webpagina's die geschreven zijn in HTML of XHTML. De CSS specificaties worden beheerd door het World Wide Web Consortium (W3C).

Om de opmaak en de inhoud van de ASP.NET pagina's van het klantprogramma gescheiden te houden werd gebruikt gemaakt van CSS bij de niet-mobiele versie. Op die manier is het eenvoudiger om later de opmaak van de hele toepassing op één plaats te wijzigen. Dit zorgt er ook voor dat de opmaak door een ander persoon kan aangepast worden zonder dat hij daarvoor de ASP.NET code of VB.NET code-behind moet aanpassen. De CSS-file die gebruikt werd voor dit deel van de toepassing is bijgevoegd in bijlage B (pp. 96-99).

In het mobiele gedeelte van de klanttoepassing werd geen gebruik gemaakt van CSS omdat dit niet door alle mobiele toestellen ondersteund wordt. Enkel toestellen die Extensible HTML Mobile Profile (XHTML-MP) ondersteunen, laten toe om CSS te gebruiken (zie [29] pp. 260-261). XHTML-MP werd goedgekeurd door het W3C en is de markup taal van WAP 2.0 toestellen. Omdat veel huidige toestellen dit echter niet ondersteunen werd gekozen om dit niet te implementeren.

4.3.7 SSL en HTTPS

Zoals eerder vermeld zal voor alle connecties met de server in Temse gebruik gemaakt worden van Secure Sockets Layer (SSL) encryptie. SSL is een

cryptografisch protocol dat voorziet in beveiligde communicatie over Internet [41]. Kort samengevat zorgt SSL ervoor dat de server geauthenticeerd wordt zodat de client weet dat hij een verbinding heeft met de juiste en betrouwbare server. Daarnaast gebeurt de communicatie niet in plain text maar geëncrypteerd. Op die manier wordt het risico beperkt dat data in de verkeerde handen terecht komt.

HyperText Transfer Protocol Secure (HTTPS) verwijst naar de combinatie van een standaard HTTP verbinding en het gebruik van een beveiligingsprotocol zoals SSL [35]. Het gebruik van deze HTTP over SSL zorgt ervoor dat gevoelige data over Internet kan verstuurd worden met een gereduceerd gevaar voor af luisteren onderweg.

4.3.8 UML

Unified Modeling Language (UML) is een gestandaardiseerde taal voor het modelleren van softwareprojecten [42]. De belangrijkste component van UML is de grafische notatie die gebruikt wordt om een abstract model van een softwarepakket op te stellen. Hoewel UML ontwikkeld werd specifiek om software te modelleren kan deze taal ook gebruikt worden om bijvoorbeeld hardware voor te stellen, bedrijfsprocessen te modelleren,...

Door standaard vormen te bieden voor klassen, overerving, toestanden, activiteiten,... laat UML toe om modellen op te stellen van allerlei aspecten die voorkomen in het software ontwikkelingsproces. Door het gebruik van deze standaard wordt het ook mogelijk om deze modellen leesbaar te maken voor iedereen die deze taal kent.

Tijdens de ontwerpfase van dit eindwerk werd gebruikt gemaakt van UML om een aantal diagramma's op te stellen die samen een grondplan van de toepassing vormen. Hier werd gebruik gemaakt van een klassediagramma voor de centrale opbouw van de business logic van de toepassing, enkele activiteitendiagramma's voor de opbouw van de user interface, enkele sequentiediagramma's om de communicatie tussen de verschillende onderdelen duidelijk te maken en ten slotte een deployment diagramma om de verschillende onderdelen van de toepassing duidelijk voor te stellen.

4.3.9 LaTeX

LaTeX is een open source zetsysteem dat werd gebruikt om dit boek op te stellen. Het gebruik van dit systeem heeft als voordeel dat de opmaak ver-

zorgd wordt door de $\text{T}_{\text{E}}\text{X}$ compiler en dat de auteur zich dus kan concentreren op de inhoud. Deze compiler houdt rekening met de belangrijkste typografische regels. Andere voordelen van het gebruik van $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ is de eenvoud om crossreferenties bij te houden, inhoudstabellen en lijsten van figuren te laten genereren, figuren in te voegen,...

In combinatie met $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ werd eveneens gebruik gemaakt van *BibTex*. Deze toepassing zorgt ervoor dat de gebruikte werken opgeslagen kunnen worden in een eenvoudige database (in tekstvorm) en de nodige verwijzingen bijgehouden worden door de compiler. Ten slotte werd ook gebruik gemaakt van *MakeIndex* om een index aan dit boek toe te voegen.

Hoofdstuk 5

Implementatie

In dit hoofdstuk zullen een aantal zaken aangehaald worden die belangrijk waren bij de implementatie van het reeds besproken ontwerp. Er zal niet tot in de details uitgelegd worden hoe alles verwezenlijkt is, daarvoor verwijs ik graag naar de code op de bij dit boek gevoegde CD-ROM.

Het doel van dit hoofdstuk is in eerste instantie om duidelijk te maken hoe de opbouw van het geheel er concreet uit ziet. Verder zal voor een aantal minder alledaagse zaken duidelijk gemaakt worden hoe deze concreet aangepakt werden.

5.1 Onderdelen

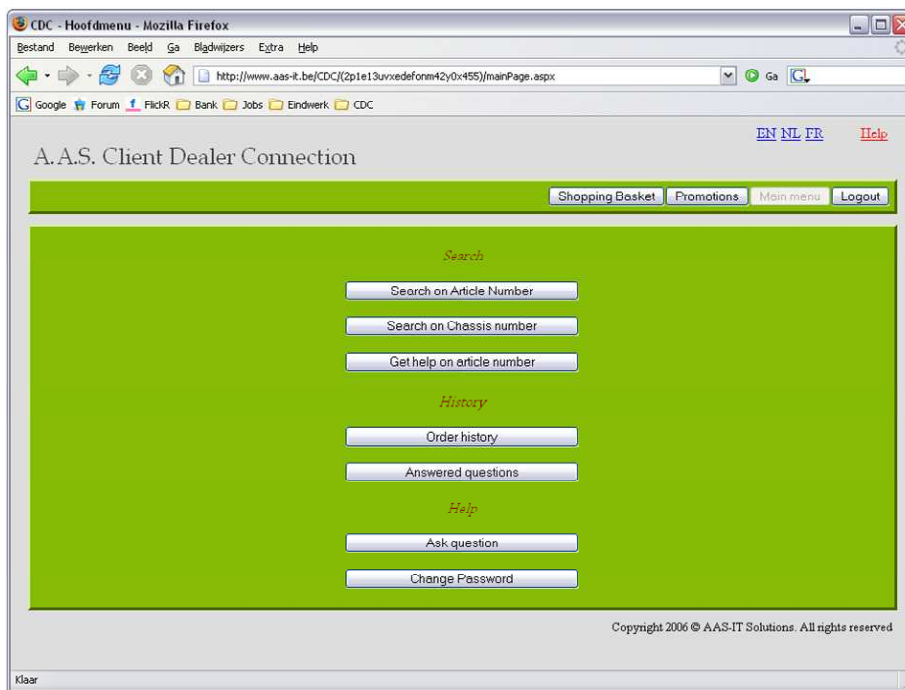
Zoals reeds in het hoofdstuk Ontwerp werd besproken, bestaat de toepassing van dit eindwerk uit een aantal componenten. Hieronder wordt elk van deze onderdelen kort besproken. Op figuur 5.4 op pagina 65 is nog eens een overzicht gegeven van de onderlinge relaties tussen de onderdelen en de naam van elk van de componenten. In figuur 5.1 (pagina 60), figuur 5.2 (pagina 61) en figuur 5.3 (pagina 63) worden bovendien een aantal screenshots getoond van de componenten van de toepassing.

Voor de volledige toepassing werd als naam het letterwoord CDC bedacht. Voluit staat dit voor Client Dealer Connection. Elk van de afzonderlijke onderdelen zal een naam hebben die verwant is aan deze naam en die meer zegt over de specifieke kenmerken van dat onderdeel.

5.1.1 ASP.NET Web Application

Een eerste onderdeel is de ASP.NET Web applicatie voor de vlootklanten die geoptimaliseerd is voor gebruik op een vaste computer. Voor dit deel hoeft de klant enkel te beschikken over een standaard browser met ondersteuning voor JavaScript. ASP.NET maakt immers voor heel wat controls gebruik van JavaScript. Dit onderdeel zal verder aangeduid worden als *Client CDC*.

Deze component wordt gehost op een IIS server bij AAS in Temse en maakt gebruik van SSL-encryptie. Voor de klanten is deze website dus beschikbaar als HTTPS en hiervoor moet de klant TCP poort 443 open zetten op eventuele routers en firewalls. De IIS server in Temse maakt voor deze component een rechtstreekse verbinding (over het lokaal netwerk) met de centrale database die op de SQL server van AAS gehost wordt. Deze database werd specifiek ontworpen voor het gebruik met alle componenten van CDC en kreeg ook de naam CDC.



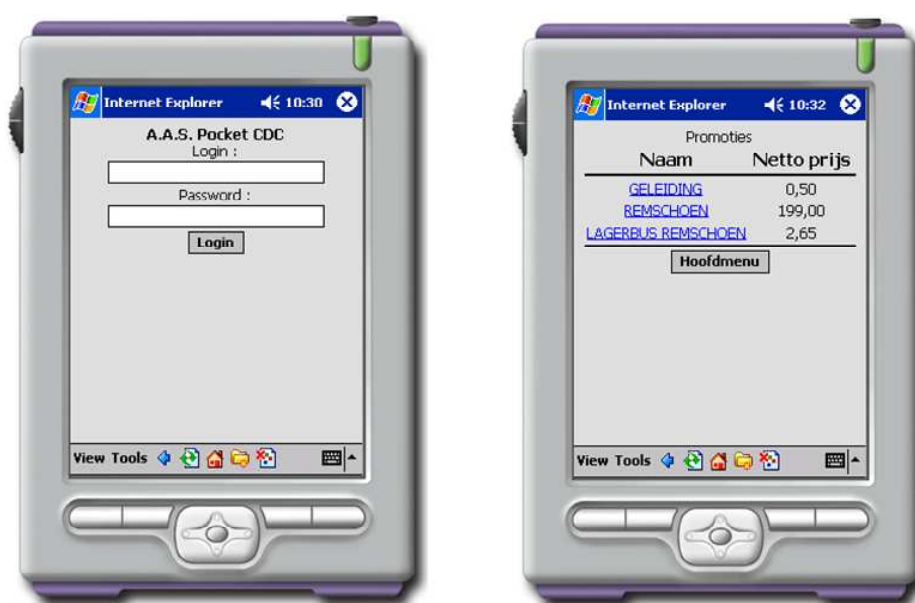
Figuur 5.1: Screenshot klanttoepassing

5.1.2 ASP.NET Mobile Web Application

Een tweede onderdeel is de ASP.NET Mobile Web applicatie voor de vlootklanten. Deze component is gelijkaardig aan Client CDC maar is geoptimaliseerd voor gebruik op een mobiel toestel. Hier hoeft de klant enkel te beschikken over een draadloos netwerk en een mobiel toestel met een browser. Dit toestel maakt dan verbinding met het Internet via een access point in het draadloos netwerk. Dit onderdeel kreeg de naam *Pocket CDC*.

Deze component wordt net zoals Client CDC gehost op de IIS server van AAS (met SSL beveiliging) en maakt op dezelfde manier verbinding met de centrale database. Hoe de toepassing weet of de mobiele versie moet gebruikt worden of de versie voor desktops wordt verder besproken.

Om gebruik te kunnen maken van zowel Client CDC als Pocket CDC beschikt de klant over een aanmeldnaam en paswoord die hij gekregen heeft van zijn dealer. Op die manier is de toepassing enkel beschikbaar voor de klanten die de dealer deze toepassing wil aanbieden.



Figuur 5.2: Screenshot mobiele versie klanttoepassing

5.1.3 Windows Application en Installer

Een volgend onderdeel is de Windows applicatie voor de dealers. Deze toepassing maakt gebruik van Windows forms en wordt geïnstalleerd op de pc van elke persoon die bij de dealer verantwoordelijk is voor de communicatie met de vlootklanten. Elke dealer die dit systeem zal gebruiken, wordt aan de centrale database toegevoegd en krijgt van AAS een aanmeldnaam en paswoord. Deze kan hij gebruiken om zich aan te melden bij zijn onderdeel van de toepassing, namelijk *Dealer CDC*.

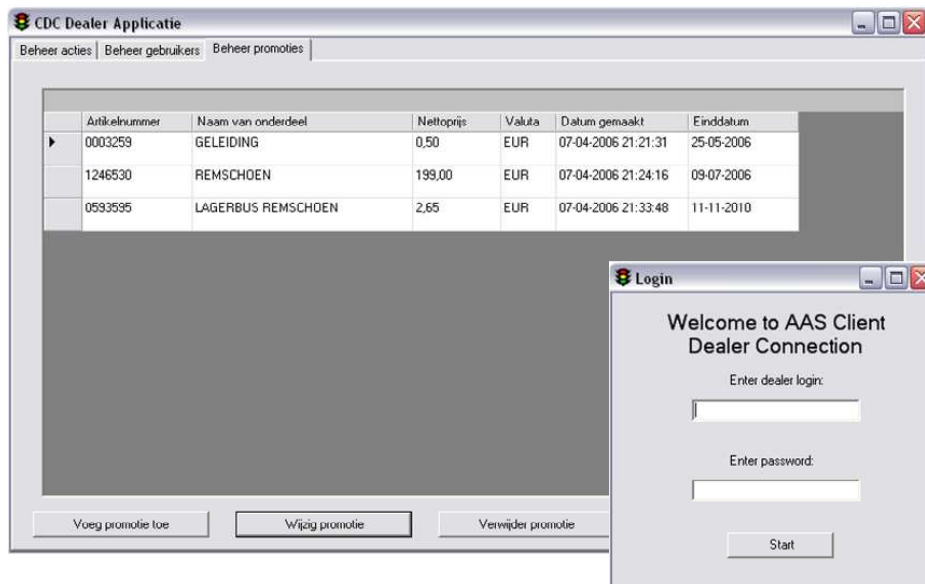
Deze toepassing maakt via het lokaal netwerk van de dealer verbinding met de lokale database van de dealer en wisselt informatie uit met de ASP.NET toepassing aan de hand van een XML Webservice. Op deze manier wordt zowel informatie uit de centrale database als dealerspecifieke informatie uit de lokale database gebruikt voor deze toepassing. In deze toepassing is er rekening gehouden met het feit dat verschillende personen tegelijk deze component kunnen gebruiken. Meer hierover wordt besproken in sectie 5.6.1 (p. 73).

Om de installatie van deze component te vereenvoudigen werd ook een installer van Dealer CDC gemaakt. Tijdens het installatieproces kan de dealer de map kiezen waar de toepassing geïnstalleerd wordt. Andere instellingen (zoals de locatie van zijn SQL server, de aanmeldnaam van de server,...) moet de dealer opgeven wanneer hij de eerste maal de toepassing opstart. Later kan hij deze ook aanpassen vanuit de toepassing zelf.

5.1.4 Console Application en Installer

Een kleiner onderdeel is een Console applicatie voor het up to date houden van de data in de centrale database. Deze component, die de naam *CDCUpdate* kreeg, wordt geïnstalleerd op de SQL server van de dealer en wordt daar aan de hand van de Windows Task Scheduler periodiek gepland. Afhankelijk van de noodzaak kan dit elke nacht gebeuren of één nacht per week of met een andere frequentie.

De CDCUpdate zorgt ervoor dat de dealer zich geen zorgen hoeft te maken over de recentheid van de data in de centrale database. De updater zorgt er immers voor dat nieuwe data rechtstreeks uit de database van de dealer gehaald wordt en aan de hand van een XML Webservice in de centrale database in Temse gestoken wordt (en dat data die niet meer gebruikt wordt uit de centrale database verwijderd wordt). Bovendien zorgt deze component



Figuur 5.3: Screenshot dealertoepassing

ervoor dat het doorsturen van mogelijks grote hoeveelheden data gebeurt op momenten dat er minst gebruik gemaakt wordt van de server en het netwerk ('s nachts).

Naast deze automatische vorm van updaten, is in Dealer CDC ook een mogelijkheid voorzien om dezelfde functie uit te voeren. Als de dealer bijvoorbeeld een aantal artikels verwijdert uit zijn aanbod of toevoegt, kan hij beslissen om deze wijzigingen onmiddellijk up te daten op de centrale database (door gebruik te maken van de datatransfer functie in Dealer CDC).

Voor deze component werd, net als voor Dealer CDC, een installer geschreven. Dit maakt de installatie van dit programma heel eenvoudig en gemakkelijk eveneens de verspreiding van het product.

5.1.5 XML Webservice

Een component die onzichtbaar is voor zowel de vlootklanten als de dealers is de XML Webservice. Deze belangrijke component zorgt voor de communicatie tussen de klanten en de dealers. Dit onderdeel, met de naam *CDCService*, wordt gehost op de IIS server van AAS en maakt eveneens een rechtstreekse connectie met de centrale database. Deze service speelt, zoals vermeld in het hoofdstuk Ontwerp, een belangrijke rol bij het real-time aspect van de toepassing.

De *CDCService* wordt aangesproken door de Windows applicatie van de dealers (Dealer CDC) en tevens door de console applicatie op de SQL server van de dealer (*CDCUpdate*). Het gebruik van deze webservice zorgt ervoor dat de communicatie tussen de verschillende componenten op een snelle en betrouwbare manier gebeurt. Het verkeer dat van en naar deze webservice gaat gebeurt aan de hand van XML, verpakt in SOAP enveloppes. Hier wordt ook gebruik gemaakt van SSL om gevoelige data te beschermen tegen ongewenste inbreuken.

5.1.6 Admin tool

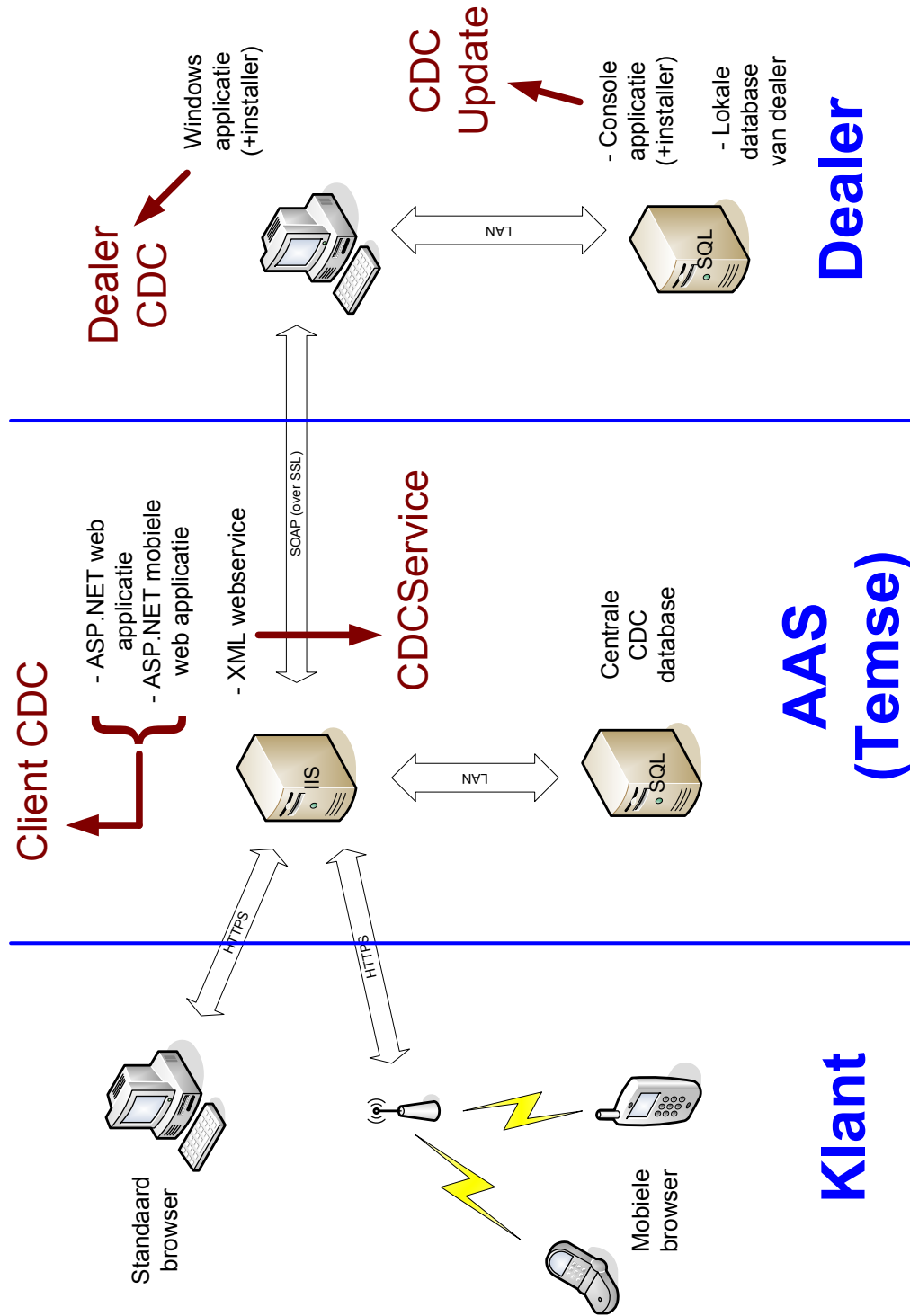
In eerste instantie werd het plan opgevat om ook een administratietoepassing te schrijven voor AAS. Deze toepassing zou het toevoegen van een dealer aan het systeem vereenvoudigen. Vanuit AAS werd echter gesteld dat dit niet noodzakelijk was, aangezien dit ook rechtstreeks in de database mogelijk is. Daarom werd besloten om dit niet meer te implementeren.

5.2 Beveiliging

Zoals hoger vermeld zijn een aantal zaken ondernomen om te voorzien in een beveiligde communicatie en bescherming van gevoelige data. We bespreken hier twee van deze maatregelen meer in detail.

5.2.1 Paswoordgenerator

Een eerste manier van beveiligen werd bekomen door het gebruik van paswoorden. Voor alle componenten in de toepassing wordt er gebruik gemaakt van een aanmeldnaam en bijhorend paswoord. Specifiek voor de klanttoepassing werd ook een paswoordgenerator geschreven. In combinatie met het gehashed opslaan van paswoorden zorgt dit er namelijk voor dat enkel de klant zijn paswoord weet. Zijn initieel paswoord wordt hem immers via email



Figuur 5.4: Onderdelen

opgestuurd en is dus nooit gekend door de dealer of door AAS. Hieronder is de methode gegeven die dit paswoord genereert.

```

1  'codefragment uit CDCService.asmx.vb
2  Function makePassword(ByVal length As Integer) As String
3      Dim passw As String
4      Dim nextOne, upper, lower, intCounter As Integer
5      Randomize()
6      For intCounter = 1 To length
7          nextOne = Int(4 * Rnd())
8          Select Case nextOne
9              Case 0
10                 'capital letters
11                 upper = 90
12                 lower = 65
13             Case 1
14                 'numbers
15                 upper = 57
16                 lower = 48
17             Case 2
18                 'letters
19                 upper = 122
20                 lower = 97
21             Case 3
22                 'symbols
23                 upper = 47
24                 lower = 33
25             End Select
26             passw += Chr(Int((upper - lower + 1) * Rnd() + lower
27                 ))
28         Next
29     Return passw
End Function

```

Listing 5.1: Paswoordgenerator gebaseerd op [5]

Deze methode genereert een willekeurig paswoord met een meegegeven lengte en kan zowel kleine letters, hoofdletters, leestekens als cijfers bevatten. Dit zorgt ervoor dat het paswoord moeilijk kan achterhaald worden met veel gebruikte methodes voor het kraken van paswoorden (zoals het gebruik van woordenboeken).

5.2.2 Beveiliging webservice

Een ander belangrijk element in de beveiliging is de authentication van de webservice. Zonder maatregelen is het namelijk zo dat een webservice vrij beschikbaar is voor iedereen die over de correcte URL beschikt. Aangezien

het hier gaat over gegevens die we enkel willen vrijgeven aan de juiste personen moeten we hier maatregelen treffen.

Om dit probleem op te lossen maken we opnieuw gebruik van een aanmeldnaam en een paswoord. Elke dealer krijgt deze wanneer hij in het systeem wordt toegevoegd. Deze gegevens moeten meegegeven worden in de soapheader als we de webservice aanspreken. Als dit niet het geval is, worden er geen gegevens vrijgegeven. De dealer wordt dus eerst geauthenticeerd bij de webservice voor hij deze kan oproepen.

Om dit te realiseren hebben we in eerste instantie een klasse nodig die de header definieert en een methode voorziet om na te gaan of de header geauthenticeerd is.

```

1 Public Class AuthHeader
2     Inherits SoapHeader
3
4     Public UserName As String
5     Public Passw As String
6
7     Public Function isAuthenticated() As Boolean
8         Dim c As New AAS.Connection
9         Try
10             'check if user password matches with stored password
11             c.storedProcedure("GetDealerPassw")
12             c.addPar("@login", SqlDbType.Char, 50, UserName)
13             c.addOutPar("@pw", SqlDbType.Char, 50)
14             c.execute()
15             If Trim(c.getParValue("@pw")).Equals(Passw) Then
16                 Return True
17             Else
18                 Return False
19             End If
20         Catch ex As Exception
21             Throw (New Exception(ex.ToString))
22         Finally
23             c.dispose()
24         End Try
25     End Function
26
27 End Class

```

Listing 5.2: Authentication Header klasse

Vervolgens geven we in alle functies van de webservice mee dat deze soapheader meegegeven moet worden en dat de functie niet mag uitgevoerd wor-

den als de naam en het paswoord niet correct zijn.

```

1 <WebMethod(Description:="Function_description"), SoapHeader("
  header")> -
2 Public Sub FunctionName(ByVal par As Parameter)
3     If header.isAuthenticated() Then
4         'execute function
5     Else
6         Throw New HttpException(401, "Not_authorized")
7     End If
8 End Sub

```

Listing 5.3: Gebruik van de authentication header

Ten slotte tonen we nog een voorbeeld hoe deze webservice nu kan aangesproken worden. Hiervoor moeten we eerst een instantie aanmaken van de authentication header. Vervolgens authenticeren we de gebruiker bij deze header. Op dit moment kunnen we binnen de toepassing gebruik maken van de, nu geauthenticerde, webservice.

```

1 'create object of webservice and of SOAP header
2 Private serv As New CDCService
3 Public auth As New AuthHeader
4
5 'authenticate user with CDCService
6 auth.UserName = Trim(TextBoxLogin.Text)
7 auth.Passw = t.hashPassword(Trim(TextBoxPassw.Text))
8 serv.AuthHeaderValue = auth
9
10 'call function of webservice
11 serv.FunctionName(par)

```

Listing 5.4: Aanspreken webservice

5.3 Performance

Omdat de snelheid van een toepassing een grote invloed heeft op de mate waarin gebruikers graag met een bepaalde toepassing werken, werden een aantal maatregelen getroffen om de snelheid van de toepassing van dit eindwerk te optimaliseren.

- Debug ondersteuning werd uitgeschakeld door de optie *compilation debug* op false te zetten in de *Web.config* bestanden.

- Er werd gebruik gemaakt van de `IsPostBack` eigenschap om de hoeveelheid data die naar de server gestuurd wordt te beperken tot een minimum.
- Er werd gebruik gemaakt van stored procedures voor bepaalde communicatie met de database. Deze zijn immers geprecompileerd en de verwerking gebeurt op server (die krachtiger is dan client).
- Er werd gebruik gemaakt van `SqlDataReaders` in plaats van `DataSets` op de plaatsen waar enkel leestoeegang vereist is.
- Datatypes werden steeds expliciet gedeclareerd.
- Om de snelheid van de mobiele toepassing te optimaliseren wordt in deze versie geen gebruik gemaakt van figuren terwijl dit op de niet-mobiele versie wel het geval is.

Deze maatregelen hebben vooral betrekking op de ASP.NET componenten (zie [29], pp. 397-412) maar hebben ook betrekking op de webservice. Omdat bij de Windows en console toepassing van de dealer de snelheid vooral bepaald wordt door de webservice werden er hier geen extra maatregelen voor getroffen.

5.4 Link met DMS

Om het voor de dealers interessant te maken om deze toepassing te gebruiken is een link met hun Dealer Management System onontbeerlijk. Deze link is er op een aantal verschillende gebieden en wordt besproken in onderstaande secties.

5.4.1 Bestellingen afhandelen

Als een bestelling momenteel binnenkomt via telefoon of fax dan moet deze ingegeven worden in DMS. Dit programma genereert dan de nodige administratieve hulpmiddelen, een pickinglist (een lijst met de plaatsen van de onderdelen in het magazijn), een factuur voor de klant,...

Opdat de dealer dit niet meer zou moeten doen, is er een link voorzien in de toepassing van dit eindwerk die er voor zorgt dat wanneer een bestelling goedgekeurd wordt dit automatisch ook gebeurt in DMS. Hoewel deze link nog niet geïmplementeerd is in DMS is er in de toepassing van dit eindwerk een webservice voorzien die deze implementatie later mogelijk maakt. Deze

webservice is hieronder gegeven.

```

1  'codefragment uit CDCService.asmx.vb
2  <WebMethod(Description:="Get_confirmed_orders_(link_to_DMS)" ),
   SoapHeader("header")> -
3  Public Function GetConfirmedOrders(ByVal dealer As String) As
   DataSet
4     If header.isAuthenticated() Then
5         Try
6             conn = New AAS.Connection
7             conn.storedProcedure("GetConfirmedOrders")
8             conn.addPar("@dealerID", SqlDbType.Int, dealer)
9             Dim ds As DataSet = conn.getDataSetFromProcedure()
10            ds.Tables(0).TableName = "ConfirmedOrders"
11            Return ds
12        Catch ex As Exception
13            Throw (New Exception(ex.ToString))
14        Finally
15            conn.dispose()
16        End Try
17    Else
18        Throw New HttpException(401, "Not_authorized")
19    End If
20 End Function

```

Listing 5.5: Link naar DMS om goedgekeurde orders te verwerken

Deze webservice maakt gebruik van een stored procedure op de centrale database in Temse. Deze stored procedure is hieronder gegeven en doet het volgende: als een bestelling goedgekeurd wordt in het dealerprogramma wordt de status van deze bestelling op 2 gezet. Het Dealer Management System kan dan de bestellingen met status 2 opvragen en verwerken. Wanneer deze opgevraagd zijn, wordt met dezelfde stored procedure de status op 3 gezet.

```

1  CREATE PROCEDURE GetConfirmedOrders
2  @dealerID as int
3
4  AS
5
6  begin transaction
7
8  SELECT *
9     FROM orders
10    WHERE statusID=2
11    and userID in (select userID
12                   from users
13                   where dealerID=@dealerID)

```

```
14  
15 update orders  
16   set statusID=3  
17   where statusID=2  
18   and userID in (select userID  
19                   from users  
20                   where dealerID=@dealerID)  
21  
22 commit  
23 GO
```

Listing 5.6: Stored procedure die link naar DMS verwerkt

5.4.2 Gebruikers toevoegen

Hoewel momenteel nog geen integratie is van de klanttabel bij DMS en de klanttabel voor de toepassing van dit eindwerk, zal AAS zorgen voor deze link. Op die manier is het overbodig om de klanten die reeds in DMS aanwezig zijn toe te voegen aan deze toepassing.

Uiteindelijk zal er gestreefd worden om tot één gebruikerstabel te komen voor alle toepassingen die bij een dealer gebruikt worden. Dit is mogelijk door gebruik te maken van één fysieke tabel en één of meerdere virtuele klanttabellen (views) per toepassing. Op die manier is er geen redundantie van de gegevens en kan toch elke toepassing over specifieke gegevens beschikken.

5.4.3 Prijsberekening

De beschikbare onderdelen (van elke dealer) worden opgeslagen in de centrale database samen met hun brutoprijs. In het ontwerp wordt een functie voorzien om de kortingen voor de klanten te berekenen. Deze korting wordt van de brutoprijs afgetrokken om de nettoprijs te bekomen. Omdat het berekenen van deze kortingen behoorlijk complex is en buiten het bestek van dit eindwerk ligt, zal deze functie later ingevuld worden door AAS. Deze functie zal reeds voorzien worden en zal voorlopig steeds 30% weergeven. Aan deze functie worden volgende parameters meegegeven: het DMS-customer nummer van de klant, het artikelnummer, het aantal stuks en het type van de bestelling (express order of stock order). Met deze gegevens kan de korting berekend worden.

5.5 Geldigheid BTW-nummer

In het dealerprogramma wordt de geldigheid van een BTW-nummer van een klant gecontroleerd wanneer deze klant toegevoegd wordt (in de AddUser-form) of gewijzigd wordt (in de EditUser-form). Deze controle gebeurt aan de hand van een gratis webservice die aangeboden wordt door de Europese Unie.

Voor meer informatie over deze service zie de WSDL-file [25]. In onderstaande code wordt een instantie van de webservice gemaakt. Aan de hand van deze instantie wordt nagegaan of het BTW nummer geldig is en van wie dit nummer is. Wanneer de service niet beschikbaar is, wordt gevraagd of de klant mag toegevoegd worden zonder controle van het BTW nummer.

```
1  'Codefragment uit AddUser.vb
2  Imports DealerAppl.europa
3
4  'define helpvariables
5  Dim vatIsValid As Boolean = False
6  Dim vatName As String
7  Dim vatAddr As String
8
9  'access European Union Webservice and check if VAT number is
   valid
10 Try
11     Dim vatserv As europa.checkVatService = New checkVatService
12     vatserv.checkVat(VATcountryCode, VATnumber, vatIsValid,
   vatName, vatAddr)
13     If vatIsValid Then
14         'show vatName and vatAddr and ask confirmation
15     Else 'ask if user can be added without checking VAT number
16     End if
17 Catch ex As Exception
18     t.log(ex.ToString, lofi)
19 End try
```

Listing 5.7: Gebruik Webservice EU

5.6 Wederzijdse uitsluiting

Er zijn twee situaties in de implementatie van dit eindwerk waarbij wederzijdse uitsluiting een belangrijke rol spelen. We bespreken hier kort beide gevallen.

5.6.1 Behandelen acties

Aangezien verschillende personen bij een dealer dezelfde toepassing tegelijk kunnen gebruiken, is het nodig dat in elk van deze toepassingen dezelfde informatie beschikbaar is. Dit is eenvoudig te bereiken door elk van deze toepassingen de informatie te laten ophalen uit de centrale database.

Daarnaast is het noodzakelijk dat twee personen nooit tegelijk dezelfde gegevens mogen aanpassen. Om hiervoor te zorgen werd in vijf tabellen in de centrale database een extra veld toegevoegd met de naam *statusID*. Deze tabellen zijn diegene waarin de bestellingen (*ORDERS*), de vragen (*DEMANDS*), de verwijderingen (*DELETIONS*), de gebruikers (*USERS*) en de promoties (*PROMOS*) staan. Wanneer deze items niet behandeld worden staat hun status op 0. Wanneer een gebruiker of promotie gewijzigd wordt, zal de status van dit item op 1 gezet worden. Als een andere persoon dan dit item probeert te wijzigen zal hij een melding krijgen dat iemand dit item reeds aan het wijzigen is. Als de promotie of de gebruiker niet meer gewijzigd wordt zal zijn status terug op 0 gezet worden.

Voor de bestellingen, vragen en verwijderingen geldt een gelijkaardig principe. Hier wordt de status echter voor nog meer gebruikt. In zijn toepassing ziet de dealer een lijst met deze drie acties. Is een actie onbehandeld dan is de status ervan 0 en is de statuscheckbox voor het item leeg. Klikkt een persoon op een actie om ze te behandelen verandert de status naar 1 en wordt de statuscheckbox gevuld. Op die manier zien de andere personen dat iemand reeds de actie aan het behandelen is. Als hij toch op deze actie klikt (om ze te behandelen) krijgt hij een boodschap dat dit niet mogelijk is.

Wanneer de persoon die een actie aan het behandelen is op Annuleren heeft geklikt, zal de status terug op 0 gezet worden en verdwijnt het vinkje terug uit de statuscheckbox. Heeft de persoon echter de actie behandeld dan zal de status op 2 gezet worden en verdwijnt de actie uit de lijst.

5.6.2 Lezen en schrijven van foto's op de server

Het tweede geval waarbij wederzijdse uitsluiting belangrijk is heeft te maken met het schrijven op de IIS server. Aan de hand van Dealer CDC kan de dealer foto's toevoegen aan een promotie. Als de dealer dit doet, wordt de foto eerst verkleind om het verkeer over het netwerk te beperken. Daarna wordt de foto, aan de hand van CDCService, naar de IIS server gestuurd en daar in een map die hiervoor voorzien is geschreven. Hoe dit gebeurt wordt

in de volgende sectie beschreven.

Door bovenvermelde status kan het niet voorkomen dat twee personen van dezelfde dealer tegelijk een foto op de server proberen schrijven. Om echter te vermijden dat twee verschillende dealers tegelijk naar een zelfde bestand proberen te schrijven is de naamgeving van de foto's opgebouwd uit het ID van de dealer en het ID van het onderdeel.

5.7 Schrijven naar de server

Wanneer een dealer een foto toevoegt aan een promotie wordt deze foto op een map in de server geschreven. Dit gebeurt als volgt. In eerste instantie wordt een open file dialoogvenster geopend. Hiermee kan de gebruiker een foto selecteren op zijn lokaal systeem. Als deze file bestaat en een correcte indeling heeft wordt deze verkleind en getoond in het venster van het dealerprogramma:

```
1 Dim img As Bitmap
2 Dim imgAdded As Boolean = False
3 Try
4     AddPic.ShowDialog()
5     Dim name As String = AddPic.FileName
6     If AddPic.CheckFileExists Then
7         'resize image to 160x120
8         img = New Bitmap(AddPic.OpenFile())
9         img = New Bitmap(img, New Size(160, 120))
10        'show image in picturebox
11        PictureB.Image = img
12        imgAdded = True
13    End If
14 Catch nopic As ArgumentException
15     MsgBox(myGParent.rm.GetString("InvalidPic"), MsgBoxStyle.
16         Exclamation, myGParent.rm.GetString("Error"))
17 Catch ex As Exception
18     MsgBox(myGParent.rm.GetString("ErrorOccurred"),
19         MsgBoxStyle.Exclamation, myGParent.rm.GetString("Error
20         "))
21     t.log(ex.ToString, lofi)
22 End Try
```

Listing 5.8: Verkleinen van foto

Wanneer de gebruiker nu op de knop klikt om het toevoegen of wijzigen van de promotie goed te keuren, wordt deze verkleinde foto eerst omgezet naar

binair formaat. Deze array van bytes wordt dan naar de webservice in Temse gestuurd:

```

1  If imgAdded Then
2      Try
3          'convert image file to byte array
4          Dim ms As New MemoryStream
5          img.Save(ms, Imaging.ImageFormat.Jpeg)
6          Dim picByte(ms.Length - 1) As Byte
7          ms.Seek(0, SeekOrigin.Begin)
8          ms.Read(picByte, 0, ms.Length)
9          'send byte array to webservice
10         serv.setPicture(picByte, myGParent.dealerID, pid)
11     Catch ex As Exception
12         MsgBox(myGParent.rm.GetString("ErrorOccurred"),
13             MsgBoxStyle.Exclamation, myGParent.rm.GetString("
14             Error"))
15         t.log(ex.ToString, lofi)
16     End Try
17 End If

```

Listing 5.9: Doorsturen van foto

Ten slotte wordt deze array van bytes in de webservice terug omgezet naar een figuur en opgeslagen als .jpg bestand in een map op de server. Deze map is een onderdeel van de Client CDC map op de IIS server en moet uiteraard schrijfrechten toestaan.

```

1  <WebMethod(Description:="Saves_picture_to_server"), SoapHeader("
2      header")> -
3  Public Sub setPicture(ByVal pic As Byte(), ByVal dealer As
4      Integer, ByVal part As String)
5      If header.isAuthenticated() Then
6          Dim objFstream As FileStream
7          Try
8              objFstream = File.Open(Server.MapPath("
9              ..\CDC\images\" + dealer.ToString +
10             "_ " + part + ".jpg"), FileMode.Create
11             , FileAccess.Write)
12             Dim lngLen As Long = pic.Length
13             objFstream.Write(pic, 0, CInt(lngLen))
14             objFstream.Flush()
15         Catch ex As Exception
16             Throw (New Exception(ex.ToString))
17         Finally
18             objFstream.Close()
19         End Try

```

```

15         Else
16             Throw New HttpException(401, "Not_authorized")
17         End If
18 End Sub

```

Listing 5.10: Opslaan van foto op de server

5.8 Globalization en localization

Zoals besproken in het hoofdstuk Ontwerp zijn er twee methodes van ondersteuning voor verschillende talen gebruikt. De eerste methode is het gebruik van de beschrijvingen van de onderdelen die in verschillende talen aanwezig zijn in de centrale database.

De tweede methode is die waarbij gebruik gemaakt wordt van XML resource files. Dit is gebruikt voor alle tekststrings die niet zijn opgeslagen in de database in alle componenten van de toepassing. De keuze van de taal is afhankelijk van de component en het moment. Zo wordt de taal van het loginscherm van Client CDC bepaald door de taal van de browser op te vragen (als dit niet mogelijk is, zoals bij de huidige versie van Pocket IE, wordt de taal ingesteld op Engels):

```

1  'initialise resource manager
2  Dim rm As ResourceManager
3  rm = New ResourceManager("CDC.TextRes", GetType(loginPage).
   Module.Assembly)
4  'set culture to first browser preference or set to default if
   browser doesn't support culture
5  Try
6      t.SetCulture(Request.UserLanguages(0))
7  Catch ex As Exception
8      t.SetCulture("en")
9  End Try

```

Listing 5.11: Localization loginscherm

Voor het instellen van de taal werd gebruik gemaakt van een zelf geschreven methode:

```

1  'method that sets the culture to a specified locale
2  Public Sub setCulture(ByVal lang As String)
3      'strip off any unneeded info
4      Dim separators() As Char = (" ;")

```

```

5   Dim langOnly As String = lang.Split(separators)(0)
6   'set culture (date-time, currency formats, ...)
7   Dim ci As CultureInfo = CultureInfo.CreateSpecificCulture(
      langOnly)
8   System.Threading.Thread.CurrentThread.CurrentCulture = ci
9   'set the UI Culture (main language: nl, en, fr, ...)
10  ci = CultureInfo.CreateSpecificCulture(langOnly.Substring
      (0, 2))
11  System.Threading.Thread.CurrentThread.CurrentUICulture =
      ci
12 End Sub

```

Listing 5.12: Methode om taal in te stellen

Eens de taal ingesteld is zal de resource manager op zoek gaan naar een resource file van de gekozen taal. Is de taal bijvoorbeeld ingesteld op Nederlands zal een file met extensie .nl.resx gezocht worden. Wordt deze niet gevonden zal de standaard taal genomen worden (met extensie .resx).

Eens de gebruiker aangemeld is wordt voor de rest van de applicatie de taal bepaald aan de hand van de voorkeurstaal van de gebruiker. Deze taal is opgeslagen in de database en kan door de gebruiker gewijzigd worden in de toepassing zelf. De taal instellen gebeurt op het moment dat de gebruiker geauthenticeerd wordt:

```

1  Private Function IsAuthenticated(ByVal user As String, ByVal
      password As String) As Boolean
2      Try
3          Dim usr As New AAS.User(user)
4          If usr.isAuthenticated(password) Then
5              usr.init()
6              'set user for this session
7              Session("usr") = usr
8              'set culture to user language (defined in database)
9              t.setCulture(Session("usr").getLangId)
10             Return True
11         Else
12             Return False
13         End If
14     Catch ex As Exception
15         Throw New Exception(ex.ToString)
16     Finally
17         c.dispose()
18     End Try
19 End Function

```

Listing 5.13: Taal van de gebruiker instellen

In deze methode wordt eveneens een object van de klasse `User` als gebruiker ingesteld voor de duur van de sessie. Deze methode biedt een aantal getters aan die de eigenschappen van de gebruiker kunnen opvragen (zoals gebeurt met de taal in bovenstaande listing).

Voor de component van de dealer is een gelijkaardige methode gebruikt. Daar is de taal van het loginscherm steeds Engels. Eens de klant ingelogd is, wordt de taal ingesteld op de taal van de dealer (eveneens te vinden in de database).

5.9 Mobiel en niet-mobiel

Aangezien er uiteindelijk voorzien werd in twee verschillende versies van het klantprogramma moet er een manier voorzien worden om een onderscheid te maken wanneer welke versie moet gebruikt worden. De loginpagina van beide versies is dezelfde. Op deze pagina wordt dan ook het onderscheid gemaakt. Hier wordt namelijk opgevraagd welke browser gebruikt wordt. Is deze browser een mobiele browser dan zal een parameter die persistent is voor de hele sessie op *true* gezet worden.

Na het inloggen wordt de gebruiker naar de startpagina gebracht. Dit is de startpagina van de mobiele versie. Hier wordt nagegaan wat de waarde is van de hoger vermelde parameter. Is deze parameter *false* dan wordt de gebruiker omgeleid naar de niet-mobiele versie, anders wordt de mobiele versie verder geladen.

```
1  'Codefragment uit Login.aspx.vb
2  If Request.Browser.Browser.Equals("Pocket_IE") Then Session("
    isMobile") = True
3
4  'Codefragment uit default.aspx.vb
5  'Load non-mobile version if device is not mobile
6  If Session("isMobile") = False Then Response.Redirect("
    promoPage.aspx")
```

Listing 5.14: Redirectie naar mobiel of niet-mobiel

Zoals in de code te zien is wordt momenteel enkel gecontroleerd of de browser *Pocket Internet Explorer* is. Dit kan echter uitgebreid worden naar meerdere types van mobiele browsers. Op die manier kunnen ook bijvoorbeeld smartphones ondersteund worden.

5.10 Loggen

Het loggen van de activiteiten gebeurt voornamelijk in de database. Elke rij in de database heeft vier extra kolommen om bij te houden wie en wanneer de rij aangemaakt heeft en door wie en wanneer de rij aangepast werd. Deze kolommen hebben de naam *CREDAT* (datum waarop de rij is aangemaakt), *CREBY* (persoon die de rij heeft aangemaakt), *MODDAT* (datum waarop de rij laatst is gewijzigd) en *MODBY* (persoon die de rij laatst heeft gewijzigd). Deze gegevens bieden de mogelijkheid om een aantal statistieken op te stellen of om bij een fout na te gaan wanneer de fout opgetreden is en door welke gebruiker de fout veroorzaakt is.

Daarnaast worden fouten die optreden steeds opgevangen en gelogd. Bij de ASP.NET toepassingen gebeurt dit in een map op de IIS server. Op die manier kan AAS snel controleren wat er fout gelopen is als dit nodig is. Bij de Windows en de console toepassing van de dealer gebeurt dit in een map op de lokale pc, namelijk een submap van de installatiemap. Voor het loggen werd volgende methode gebruikt:

```
1  'method that logs a message to the specified file
2  Public Sub log(ByVal msg As String , ByVal name As String)
3      Dim logfile As New System.IO.StreamWriter(name, True)
4      logfile.WriteLine(Date.Now.ToString("yyyy-MM-dd_HH:mm:ss")
5          + " _-->_" + msg)
6      logfile.WriteLine()
7      logfile.Close()
End Sub
```

Listing 5.15: Methode voor logbestanden te schrijven

Momenteel wordt er (behalve in de hoger vernoemde velden in de database) niet bijgehouden wat het gedrag is van de gebruikers van het systeem. Voor de dealer kan het misschien interessant zijn om het koopgedrag van zijn klanten bij te houden. Op die manier kan hij bijvoorbeeld te weten komen welke de meest verkochte onderdelen zijn.

Deze mogelijkheid kan ingebouwd worden door bijvoorbeeld gebruik te maken van Google Analytics [8]. Deze gratis tool wordt aangeboden door Google. Om over deze uitgebreide set van statistische tools te beschikken volstaat het om een gratis account aan te vragen en op elke pagina van de ASP.NET toepassing de conversiecode van Google Analytics in te voegen.

5.11 Wettelijke bepalingen

Omdat dit buiten het bestek van dit eindwerk ligt, werden de wettelijke bepalingen die de klant moet te zien krijgen in zijn deel van de toepassing niet opgesteld. Wel werd er voorzien in een pagina die deze bepaling kan tonen. De bepaling zelf kan ingevoegd worden in de database bij elke dealer. Het programma roept deze tekst dan op vanuit de database.

Volgende zaken zijn binnen de Europese Unie wettelijk verplicht om op te nemen in de wettelijke bepaling die bij een systeem van elektronische handel gevoegd moeten worden [26],[28],[7]:

- Naam en adres van de dealer
- Emailadres en telefoonnummer van de dealer (email is verplicht bij elektronische handel omdat een snel communicatiemiddel moet beschikbaar zijn)
- Inschrijvingsnummer bij het handelsregister en BTW-nummer van de dealer
- Voorwaarden voor eventuele kortingen
- De stappen om tot een contract te komen
- Welke gegevens van de klanten bijgehouden worden
- De mogelijke talen waarin communicatie mogelijk is en waarin de elektronische middelen beschikbaar zijn
- De gedragscodes van de dienstverlener
- Als een klant iets heeft besteld moet dit bevestigd worden via email
- De melding dat de klant 7 dagen bedenktijd heeft (dit wil zeggen dat de klant binnen 7 dagen kan afzien van zijn bestelling, tenzij de bestelling ondertussen geleverd is)
- Een melding dat de toepassing voorziet in een methode voor het vermijden van invoerfouten

Door de getroffen voorzieningen is het dus eenvoudig om elke dealer zelf zijn specifieke bepalingen te laten opstellen en deze op te slaan in de database. Op die manier kan elke dealer naast de verplichte zaken ook extra vermeldingen doen die specifiek voor hem gelden.

Daarnaast werd er ook voor gezorgd dat de klant een bevestigingsemail krijgt als hij een bestelling geplaatst heeft. Ten slotte zal in de wettelijke bepaling zeker moeten vermeld worden dat bijgehouden wordt dat het koop- en verwijdergedrag van de klanten bijgehouden wordt. De klant heeft immers het recht om hiervan op de hoogte te zijn en de dealer is verplicht om dit duidelijk te maken.

5.12 Real-time

Zoals hoger vermeld zijn er een aantal maatregelen getroffen in de toepassing om ervoor te zorgen dat de dealers in real-time kunnen zien wanneer hun klanten een bestelling geplaatst hebben, een vraag gesteld hebben of een item verwijderd hebben omdat het te duur is (of om een andere reden). Hoe dit werd verwezenlijkt wordt hier besproken.

De kern van het real-time aspect zit in het gebruik van een timer. Deze timer wordt gebruikt om periodiek (elke 30 seconden) een webservice te bevragen. Dit wordt bovendien gedaan in een thread die op de achtergrond uitgevoerd wordt. Op die manier wordt de normale werking van het programma hier niet door gestoord. Eerst declareren we een nieuwe timer en een nieuwe thread:

```

1 Dim thr As Thread
2   'create intervals for timer
3 Dim timer1ms As Integer = 30000   'interval= 30 sec
4   'create timer
5 Dim t1 As New System.Timers.Timer(timer1ms)

```

Listing 5.16: Declaratie timer

Nadat de gebruiker geauthenticeerd is, starten we vervolgens de thread die voor de controle zal zorgen:

```

1   'start background thread that periodically checks for actions
2   thr = New Thread(AddressOf Me.checkAction)
3   thr.IsBackground() = True
4   thr.Start()

```

Listing 5.17: Starten van thread

Deze thread zelf is geformuleerd in de *checkAction* functie. Deze functie zal twee timers starten. De tweede timer zorgt ervoor dat de dealer elke 10 mi-

nuten herinnerd wordt als hij een actie onbehandeld laat. Elke keer de eerste timer afloopt (dus na elke 30 seconden) zal dan de methode *action* opgeroepen worden voor deze dealer:

```

1  'thread that fires timers
2  Private Sub checkAction()
3      AddHandler t1.Elapsed, AddressOf timerFired
4      AddHandler t2.Elapsed, AddressOf timer2Fired
5      t1.Enabled = True
6  End Sub
7
8  'calls action every time t1 fires
9  Private Sub timerFired(ByVal sender As Object, ByVal e As System
10     .Timers.ElapsedEventArgs)
11     action(dealerID)
12 End Sub

```

Listing 5.18: Starten van de timer

De methode *action* zal tenslotte controleren welke acties er voor deze dealer staan te wachten. Er zijn drie soorten acties en dus vier verschillende toestanden. Als er geen actie staat te wachten wordt de variable *pendingAct* op -1 gezet, als er bestellingen wachten op 0, als er vragen wachten op 1 en als er verwijderingen zijn op 2. De prioriteit wordt bepaald door deze variabele. Hoe hoger het cijfer, hoe hoger de prioriteit.

```

1  'function that checks webservice for actions
2  Private Sub action(ByVal dealer As Integer)
3      Try
4          Select Case serv.Action(dealer)
5              Case -1 'no actions
6                  If pendingAct <> -1 Then
7                      t2.Enabled = False
8                      NotifyIcon.Icon = New Icon([Assembly].
9                          GetExecutingAssembly.
10                             GetManifestResourceStream("DealerAppl.cdc
11                                 .ico"))
12                     pendingAct = -1
13                 End If
14             Case 0 'orders
15                 If pendingAct <> 0 Then
16                     NotifyIcon.Icon = New Icon([Assembly].
17                         GetExecutingAssembly.
18                             GetManifestResourceStream("DealerAppl.
19                                 green.ico"))
19                     If pendingAct < 0 Then NotifyIcon.
20                         ShowBalloon(rm.GetString("Orders"), rm.

```

```

15         GetString("NewOrder") + vbLf + rm.
16         GetString("ClickToHandle"), BalloonTip.
17         NotifyInfoFlags.Warning, 10000)
18     pendingAct = 0
19     If Not t2.Enabled Then t2.Enabled = True
20     End If
21 Case 1 'demands
22     If pendingAct < 1 Then
23         NotifyIcon.Icon = New Icon ([Assembly].
24         GetExecutingAssembly.
25         GetManifestResourceStream("DealerAppl.
26         orange.ico"))
27         If pendingAct < 1 Then NotifyIcon.
28         ShowBalloon(rm.GetString("Demands"), rm.
29         GetString("NewDemand") + vbLf + rm.
30         GetString("ClickToHandle"), BalloonTip.
31         NotifyInfoFlags.Warning, 10000)
32         pendingAct = 1
33         If Not t2.Enabled Then t2.Enabled = True
34         End If
35     Case 2 'deletions
36     If pendingAct < 2 Then
37         NotifyIcon.Icon = New Icon ([Assembly].
38         GetExecutingAssembly.
39         GetManifestResourceStream("DealerAppl.red
40         .ico"))
41         NotifyIcon.ShowBalloon(rm.GetString("
42         Deletions"), rm.GetString("NewDeletion")
43         + vbLf + rm.GetString("ClickToHandle"),
44         BalloonTip.NotifyInfoFlags.Warning,
45         10000)
46         pendingAct = 2
47         If Not t2.Enabled Then t2.Enabled = True
48         End If
49     End Select
50     err = 0
51 Catch ex As Exception
52     If err = 0 Then
53         err = 1
54         MsgBox(rm.GetString("ServiceNotAvailable") + vbLf +
55         rm.GetString("CheckLog"), MsgBoxStyle.Exclamation
56         , rm.GetString("Error"))
57         t.log(ex.ToString, 10fi)
58     End If
59 End Try
60 End Sub

```

Listing 5.19: Controleren op acties

In deze methode wordt ervoor gezorgd dat het icoontje van het dealerprogramma aangepast wordt naargelang de wachtende acties. Daarnaast wordt ook een signaal gegeven aan de dealer wanneer een nieuwe actie met hogere prioriteit dan de huidige wachtende aangekomen is. Dit wordt gedaan door een tooltip te tonen bij het programma-icoontje. Wanneer op deze tooltip geklikt wordt, wordt onmiddellijk het venster getoond waarop de actie met de hoogste prioriteit kan afgehandeld worden. In figuur 5.5 wordt een voorbeeld getoond van de mogelijke toestanden.

De methode in CDCService die aangesproken wordt door bovenstaande code maakt gebruik van drie stored procedures om na te gaan of er acties staan te wachten. Deze methode bepaalt ook de prioriteit van de acties.

```

1 <WebMethod(Description:= "Returns an integer that defines the
  client action"), SoapHeader("header")>
2 Public Function Action(ByVal dealerID As Integer) As Integer
3   Dim i As Integer = -1
4   Try
5     conn = New AAS.Connection
6
7     'check for orders
8     conn.storedProcedure("CheckForOrders")
9     conn.addPar("@dealerID", SqlDbType.Int, dealerID)
10    conn.addOutPar("@result", SqlDbType.Int)
11    conn.execute()
12    If conn.getParValue("@result") > 0 Then i = 0
13
14    'check for demands
15    conn.storedProcedure("CheckForDemands")
16    conn.addPar("@dealerID", SqlDbType.Int, dealerID)
17    conn.addOutPar("@result", SqlDbType.Int)
18    conn.execute()
19    If conn.getParValue("@result") > 0 Then i = 1
20
21    'check for deletions
22    conn.storedProcedure("CheckForDeletions")
23    conn.addPar("@dealerID", SqlDbType.Int, dealerID)
24    conn.addOutPar("@result", SqlDbType.Int)
25    conn.execute()
26    If conn.getParValue("@result") > 0 Then i = 2
27
28  Catch ex As Exception
29    i = -1
30    Throw (New Exception(ex.ToString))
31  Finally
32    conn.dispose()

```

```

33     End Try
34     Return i
35 End Function

```

Listing 5.20: Webservice die acties controleert



Figuur 5.5: Screenshot mogelijke toestanden

5.13 Configuratie

Om de installatie en configuratie van de toepassing te vereenvoudigen werden de belangrijkste instellingen verzameld in een aantal centrale configuratiebestanden. Voor de ASP.NET toepassingen en de XML webservice zijn de instellingen opgeslagen in het XML bestand met de naam *Web.config*. Hier zijn onder andere de authentication methode, de methode voor weergave van fouten en de connectionstring voor de communicatie met de centrale database opgeslagen. In het configuratiebestand van de webservice staan ook de instellingen voor het versturen van emails (het emailadres, de naam en de uitgaande mailserver).

Voor de Windows toepassing en de console toepassing werd een eigen XML bestand aangemaakt met de naam *App.config*. Hierin staan onder andere de gegevens voor de verbinding met de lokale database van de dealer, het emailadres van DAF Eindhoven (voor het versturen van emails over te hoge prijzen), de locatie van de klanttoepassing (zodat de dealer vanuit zijn programma het klantprogramma kan opstarten) en het aantal filialen waarvan de dealer de stockinhoud wil zien.

In Dealer CDC is het bovendien mogelijk om de instellingen te wijzigen. De methode die hiervoor instaat, leest een aantal velden in en schrijft die dan terug naar het XML configuratiebestand. In deze methode wordt eveneens gecontroleerd of de invoer van deze velden het correct formaat heeft. Dit gebeurt door het opvangen van fouten die veroorzaakt worden door een ongeldige typecast.

```

1      Try
2          'configuration file strings
3          connServer = TextBoxServer.Text
4          connDatabase = TextBoxDatabase.Text
5          connUser = TextBoxUser.Text
6          connPassword = TextBoxPassword.Text
7
8          emailDaf = TextBoxDaf.Text
9          branches = TextBoxBranches.Text
10
11         'define configuration file path
12         Dim assemblypath As String = "DealerAppl.exe"
13         Dim appConfigPath As String = assemblypath + ".config"
14
15         'generate xml document
16         Dim doc As XmlDocument = New XmlDocument
17         doc.Load(appConfigPath)
18
19         Dim configuration As XmlNode = Nothing
20         For Each node As XmlNode In doc.ChildNodes
21             If node.Name = "configuration" Then configuration =
                node
22         Next
23
24         If Not configuration Is Nothing Then
25
26             Dim settingNode As XmlNode = Nothing
27             For Each node As XmlNode In configuration.ChildNodes
28                 If node.Name = "appSettings" Then settingNode =

```

```

node
29     Next
30
31     If Not settingNode Is Nothing Then
32         For Each node As XmlNode In settingNode.
            ChildNodes
33             Dim att As XmlAttribute = node.Attributes("
                key")
34             If Not att Is Nothing Then
35                 Select Case att.Value
36                     Case "ConnServer"
37                         att = node.Attributes("value")
38                         att.Value = connServer
39                     Case "ConnDatabase"
40                         att = node.Attributes("value")
41                         att.Value = connDatabase
42                     Case "ConnUser"
43                         att = node.Attributes("value")
44                         att.Value = connUser
45                     Case "ConnPassword"
46                         att = node.Attributes("value")
47                         att.Value = connPassword
48                     Case "EmailDaf"
49                         att = node.Attributes("value")
50                         att.Value = emailDaf
51                     Case "Branches"
52                         att = node.Attributes("value")
53                         att.Value = branches
54                 End Select
55             End If
56         Next
57         doc.Save(appConfigPath)
58         If MsgBox(myParent.rm.GetString("ConfigIsChanged"
            ") + vbLf + myParent.rm.GetString("Shutdown?"
            )), -
59             MsgBoxStyle.Question + MsgBoxStyle.YesNo,
                myParent.rm.GetString("ConfigChanged")) =
60             MsgBoxResult.Yes Then myParent.Dispose()
61         End If
62     End If
63     Me.Dispose()
64 Catch iex As InvalidCastException
65     MsgBox(myParent.rm.GetString("InvalidNumber"),
        MsgBoxStyle.Exclamation, myParent.rm.GetString("
        Invalid"))
66     TextBoxBranches.Text = 1
67 Catch ex As Exception
68     MsgBox(myParent.rm.GetString("ErrorOccurred"),
        MsgBoxStyle.Exclamation, myParent.rm.GetString("Error

```

```
69         ))
70         t.log(ex.ToString, l of i)
71     End Try
End Sub
```

Listing 5.21: Configuratiebestand aanpassen

Op het moment dat de windows toepassing gestart wordt, zal het .NET framework de *App.config* file in een alleen lezen bestand laden. Daarom schrijft bovenstaande methode de instellingen naar de *App.config* file en niet naar het actieve configuratiebestand.

Het gevolg hiervan is dat de wijzigingen pas actief worden als de toepassing de volgende keer gestart wordt. De gebruiker wordt na het instellen van de parameters dan ook gevraagd om de toepassing af te sluiten en terug op te starten. In het .NET framework versie 2.0 zijn er voorzieningen getroffen om dit te vereenvoudigen en waardoor het herstarten van de toepassing overbodig wordt.

5.14 Testen

In eerste instantie werd de toepassing getest door een aantal collega studenten en familieleden. Dit leverde al een aantal opmerkingen op die verwerkt zijn in het eindresultaat. Verder werd de toepassing ook getest door AAS. Ten slotte was het ook de bedoeling om de toepassing te laten testen door de eindgebruikers. Omwille van het feit dat de kortingsmethode nog niet ingevuld is was deze optie echter niet mogelijk. Er werd wel een demonstratie gegeven aan de gebruikers om hun visie van het eindresultaat te kunnen verwerken in het besluit van dit boek.

5.15 Aanpassingen na tests

Een eerste opmerking tijdens de tests had te maken met de menubalk in de klanttoepassing. Oorspronkelijk was het zo dat de knop van de pagina die op dat moment zichtbaar was verdween uit de lijst. Dit zorgde ervoor dat de positie van de menuknoppen niet steeds dezelfde was. Daarom werd dit aangepast. Nu is het zo dat de knop van de huidige pagina gewoon niet beschikbaar is en een ander kleur heeft dan de beschikbare knoppen. Op die manier weet de gebruiker ook steeds op welke pagina hij zich bevindt.

In de oorspronkelijke versie was de pagina waar de klant zijn beantwoorde

vragen kon bekijken behoorlijk statisch. Op suggestie van een testgebruiker werd er voorzien in een mogelijkheid om vanuit een beantwoorde vraag opnieuw een vraag te stellen (met de oorspronkelijke vraag en antwoord in de nieuwe vraag) of om onmiddellijk het artikel waarover de vraag gaat op te zoeken.

Een andere aanpassing was dat in de tabellen met onderdelen (zoals de promotiepagina) minder lege ruimte getoond wordt. Dit zorgt ervoor dat er meer informatie op een kleiner oppervlak kan. Daarnaast werd er voorzien in een scrollbar in de tabellen voor als het oppervlak van het scherm toch te klein zou zijn om alle gegevens tegelijk weer te geven.

De laatste twee aanpassing werden gedaan op aanvraag van de gecontacteerde dealers. De eerste aanpassing was het versturen van een email naar de klant wanneer de status van zijn vragen aangepast is. Op die manier hoeft de klant niet periodiek de pagina met beantwoorde vragen te controleren. Elke keer de dealer een vraag beantwoordt, kan hij kiezen om de gebruiker al dan niet een email te sturen met de melding dat zijn vraag beantwoord is. In deze automatisch gegenereerde email is ook een rechtstreekse link voorzien naar de pagina met de beantwoorde vragen.

De laatste aanpassing die werd gedaan heeft te maken met het toevoegen van promoties. Momenteel sturen de dealers bij het aanmaken van een nieuwe promotie een email naar de klanten voor wie deze promotie geldt. In de dealertoepassing wordt nu een optie hiervoor meegegeven bij het toevoegen van een promotie. De dealer kan dus kiezen om automatisch een email te laten versturen naar alle klanten voor wie de promotie aangemaakt wordt.

Hoofdstuk 6

Mogelijke uitbreidingen

Na het afronden van dit eindwerk blijven nog een aantal open punten over. In eerste instantie werden een aantal punten vastgesteld die een meerwaarde zouden kunnen bieden maar die niet noodzakelijk zijn voor de werking. Daarnaast werden door de dealers een aantal zaken aangeduid die voor een beter eindproduct zouden kunnen zorgen. Ten slotte zijn er nog een aantal zaken die moeten afgewerkt worden door AAS omdat deze buiten het bestek van dit eindwerk lagen.

6.1 Extra mogelijkheden

In eerste instantie was het de bedoeling om de zoekfunctionaliteit voor de klanten vergelijkbaar te maken met die van het Parts Rapido programma dat door de dealers wordt gebruikt. Doordat een webservice van DAF Eindhoven binnen het tijdsbestek van dit eindwerk niet te verwachten was, is deze mogelijkheid niet geïntegreerd. Als dit later wel zou kunnen, zou dit zeker een meerwaarde betekenen voor de zoekmogelijkheden voor de klanten.

Momenteel is het zo dat bij de dealers die meerdere personen hebben die Dealer CDC gebruiken er per pc elke 30 seconden een webservice aangesproken wordt voor de controle op wachtende acties. Een gedistribueerde versie van het dealerprogramma die de controle slechts één maal per dealer zou uitvoeren zou een iets minder zware belasting van het netwerk betekenen.

Tijdens de analysefase werden slechts twee dealers aangesproken voor het opstellen van de eisen. Daarnaast was het niet mogelijk om een gesprek te regelen met vlootklanten. Een analyse bij meerdere dealers en gesprekken met vlootklanten zouden misschien nog enkele punten kunnen opgeleverd

hebben waar rekening mee gehouden kon worden.

In de huidige versie is het zo dat de klant bij het plaatsen van zijn bestelling geen idee heeft over de onmiddellijke beschikbaarheid van de onderdelen. Rekening houdend met de vraag van de dealers om hun stockinhoud niet vrij te geven zou kunnen geopteerd worden om de klant een indicatie te geven van de leverbaarheid. Een voorbeeld hiervan zou kunnen zijn: een rode kleur voor niet in voorraad, oranje voor misschien in voorraad en groen voor zeker in voorraad. Zo hoeft de dealer zijn stock niet vrij te geven en heeft de klant toch een idee van de levertermijn.

Ten slotte zou het foutief bestellen van onderdelen door klanten voor een stuk kunnen vermeden worden door bij het zoeken op artikelnummer in het bestand van DAF Eindhoven te controleren of dit onderdeel wel behoort tot een vrachtwagen van die klant. Dit is echter enkel mogelijk voor de klanten waarvoor dat bestand beschikbaar is. Bovendien zou dit een stuk eenvoudiger zijn mocht dit bestand bijvoorbeeld in XML opgesteld zijn in plaats van in platte tekst.

6.2 Opmerkingen gebruikers

Het belangrijkste punt waar een aanpassing gevraagd werd door de dealers heeft te maken met het zoeken op chassisnummer. Momenteel is het zo dat de gebruiker naast het chassisnummer de primaire en de secundaire groep moet invullen. Nu blijkt echter dat weinig vlootklanten deze gegevens kennen (dit kwam pas aan het licht na het afronden van de toepassing). Daarom is het aangewezen om de twee textboxes te vervangen door dropdownboxes. Op die manier kan de klant een primaire groep kiezen uit een lijst met beschikbare groepen. De lijst van de secundaire groep wordt dan aangepast aan deze keuze. In de lijst van primaire groepen kan ook een optie voor alle groepen meegegeven worden.

De volgende punten die aangehaald werden zijn eerder extra's maar zijn toch zeker de moeite om te vermelden en toe te voegen voor het effectief in werking zetten van het programma. Een eerste punt is de mogelijkheid voor de dealers om een geschiedenis bij te houden van de vragen die klanten gesteld hebben en eventueel ook van de verwijderingen en de bestellingen. Ten slotte zou het voor de klanten interessant zijn om onmiddellijk te kunnen kiezen uit een aantal veelgebruikte onderdelen. Zo zijn er een aantal onderdelen die voor een bepaald chassisnummer veel verkocht worden. Als de klant deze

kan zien moet hij niet steeds opnieuw op zoek via de primaire en secundaire groep.

6.3 Aanvullingen AAS

Voor de klanten zou hun stuk van de tool interessanter zijn mocht ook hun administratie hierdoor voor een stuk opgevangen worden. Dit zou bijvoorbeeld kunnen door hen de mogelijkheid te bieden om rapporten en/of facturen uit te voeren (bijvoorbeeld aan de hand van Crystal Reports). Dit werd echter bij de aanvang van dit project als niet noodzakelijk beschouwd en is dan ook niet essentieel voor de werking.

Om dit project echt nuttig te maken voor de dealers is een goede link met het Dealer Management System zeker nodig. In eerste instantie moeten de nu afzonderlijke klanttabellen van DMS en CDC geïntegreerd worden. Anders moet de dealer manueel zijn klanten invoeren wat zeker niet de bedoeling is. Verder zou het ook zo moeten zijn dat wanneer een klant toegevoegd of gewijzigd wordt in het ene programma dit ook gebeurt in het andere. Het tweede punt waarop integratie nodig is heeft te maken met het afhandelen van de bestellingen. Nu moeten bestellingen manueel ingegeven worden in DMS omwille van administratieve redenen en om een pickinglist te kunnen opstellen. Het lijkt ons evident dat wanneer de dealer een bestelling goedkeurt in CDC dit automatisch moet ingevoerd worden in DMS. Hier werd, zoals reeds besproken, dan ook een webservice voor voorzien in CDC.

Ten slotte is het noodzakelijk dat de (nu lege) methode om kortingen te berekenen ingevuld wordt. Anders is het onmogelijk om dit project in gebruik te nemen. Zonder deze berekening klopt de nettoprijs voor de klanten immers niet.

Hoofdstuk 7

Besluit

Het doel van dit eindwerk was om de communicatie tussen verdelers van vrachtwagenonderdelen en hun grote klanten te automatiseren en te vereenvoudigen. Voor de klanten is de hoofdfunctie van deze communicatie het bestellen van onderdelen. Voor de dealer is de hoofdfunctie het afhandelen van deze bestellingen. Een minimum vereiste van een tool die de communicatie automatiseert is dan ook het vereenvoudigen van dit proces.

Om tot een duidelijk beeld van de huidige werkwijze te komen werd eerst een grondige analyse uitgevoerd aan de hand van gesprekken met dealers. Uit deze gesprekken werden de eisen opgesteld van de link tussen de vlootklanten en dealers. Uit deze eisen werd een ontwerp opgesteld. Hierbij werden heel wat technologieën bestudeerd op hun bruikbaarheid voor deze toepassing. Na de selectie van meest relevante technologieën werd het ontwerp ten slotte omgezet in een werkbaar programma.

Tijdens de implementatiefase werd, met het oog op verdere uitbreiding of herbruikbaarheid van bepaalde componenten, tevens rekening gehouden met object-geöriënteerde programmeertechnieken en het Model-View-Controller model. Bovendien werd regelmatig contact opgenomen met de dealers om rekening te kunnen houden met hun feedback.

Voor de evaluatie van het eindproduct werd terug contact opgenomen met de dealers die betrokken werden tijdens de analysefase. Aan deze personen werd een demonstratie gegeven en hun commentaar gevraagd. De algemene indruk was dat de twee belangrijkste componenten (het klantdeel en het dealerdeel) zeer eenvoudig en gebruiksvriendelijk opgesteld zijn. Daarnaast werd vastgesteld dat het eindproduct voldoet aan de eisen die de dealers stelden tijdens de analysefase.

Er kan dus besloten worden dat het eindresultaat, met de hierboven vermelde aanpassingen, een bruikbaar product is waarover de vragende partij tevreden is. Verder kan ik besluiten dat er tijdens het hele proces heel wat kennis opgedaan is in verband met het hele software-ontwikkelingsproces. Er werd geleerd om vanuit een hoeveelheid aan informatie een aantal essentiële eisen op te stellen en hieruit een ontwerp te creëren. Daarnaast werden heel wat technologieën aangeleerd tijdens de implementatiefase. Het uitvoeren van dit eindwerk was dan ook een heel leerrijk proces.

Bijlage A

Structuur DAF-bestand

Momenteel is DAF Eindhoven aan het werken aan een toepassing die gegevens kan exporteren uit het Parts Rapido pakket. Deze exportbestanden zijn platte tekst en zullen de volgende structuur hebben:

```
1 chassisnr: 17 text
2 artikelnummer: 20 text
3 artikelnummer bij leverancier: 20 text
4 groepering1: 10 text
5 groepering2: 10 text
6 groepering3: 10 text
7 groepering4: 10 text
8 omschrijving artikelnr: 40 text
9 filler: 100 text
```

Listing A.1: Structuur Parts Rapido export bestand

Hierbij is het zo dat het eerste artikelnummer het DAF-nummer is en het tweede het nummer dat het artikel heeft bij de leverancier (indien dit artikel niet door DAF zelf geproduceerd wordt). Enkel het eerste nummer is belangrijk om het onderdeel te kunnen bestellen bij DAF Eindhoven. Momenteel worden de onderdelen enkel gegroepeerd in twee niveaus. Niveau 3 en 4 zijn er dus enkel om toekomstige uitbreidingen mogelijk te maken.

Bijlage B

Stylesheet ASP.NET toepassing

```
1 body
2 {
3     background-color:#DDDDDD;
4 }
5
6 #LabelText
7 {
8     margin-left:15px;
9     text-align:left;
10    font-size:x-large;
11    font-family:Book Antiqua;
12    color:#3C3C3C;
13 }
14
15 #PanelHead
16 {
17     margin:10px 10px 10px 10px;
18     padding:5px 5px 5px 5px;
19     border-style:outset;
20     border-color:#85BB04;
21     background-color:#85BB04;
22     text-align:right;
23 }
24
25 #PanelLang
26 {
27     margin:0px 20px 0px 0px;
28     text-align:right;
29 }
30
31 #PanelMain
32 {
33     margin:10px 10px 10px 10px;
```

```
34         padding:20px 20px 20px 20px;
35         border-color:#85BB04;
36         border-style:outset;
37         background-color:#85BB04;
38         text-align:center;
39         overflow:auto;
40     }
41
42     #PanelL
43     {
44         text-align:center;
45     }
46
47     #PanelR
48     {
49         text-align:center;
50     }
51
52     /* Look and feel of grids */
53     .Grid
54     {
55         border-color:#404040;
56         text-align:center;
57         margin:auto;
58     }
59
60     .Grid td
61     {
62         border-color:#404040;
63         padding:10px 10px 10px 10px;
64     }
65
66     .GridHeader
67     {
68         font-style:italic;
69     }
70
71     .Picture
72     {
73         width:160px;
74         height:120px;
75     }
76
77     .ShortBox
78     {
79         width:50px;
80     }
81
82     .LongBox
```

```
83 {
84     width:100px;
85 }
86
87 .MultilineBox
88 {
89     height:200px;
90     width:400px;
91 }
92
93 .DisclaimerBox
94 {
95     height:400px;
96     width:600px;
97 }
98
99 .List
100 {
101     margin:auto;
102 }
103
104 .Warning
105 {
106     color:#660000;
107 }
108
109 .Button
110 {
111     width:250px;
112 }
113
114 .LanguageButton
115 {
116     padding:0px 0px 5px 5px;
117 }
118
119 .HelpButton
120 {
121     padding:0px 0px 40px 40px;
122     color:Red;
123 }
124
125 .Label
126 {
127     font-style:italic;
128     font-family:Sylfaen;
129     color:Maroon;
130 }
131
```

```
132 #Footer
133 {
134     margin-right:15px;
135     text-align:right;
136     font-size:smaller;
137 }
```

Listing B.1: StyleSheet.css

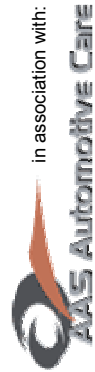
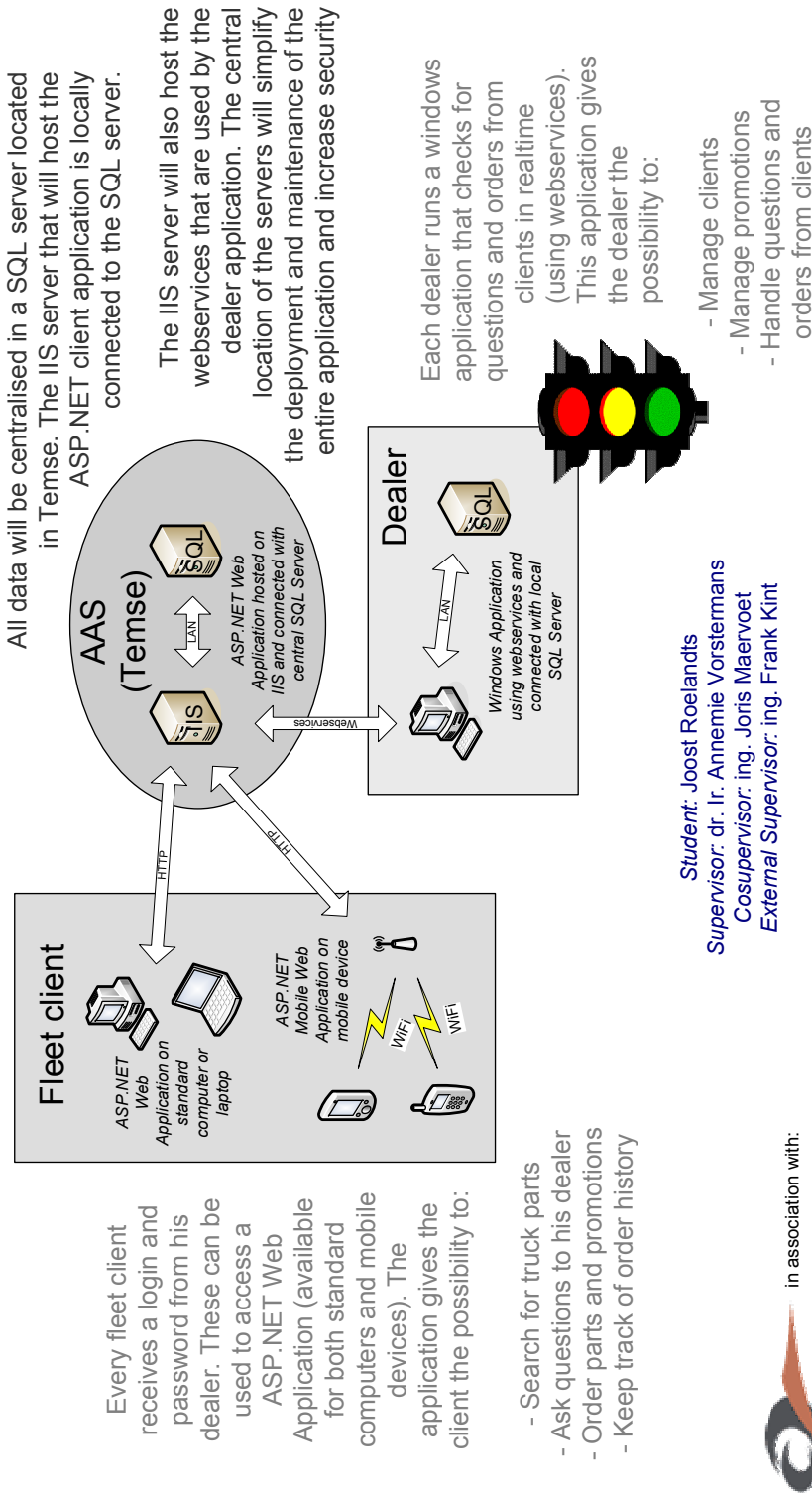
Bijlage C

Poster eindwerk

Vanuit KaHo Sint-Lieven werd dit academiejaar de vraag gesteld aan de vierdejaarsstudenten Industrieel Ingenieur van de optie Elektronica om een poster over hun eindwerk te maken.

Deze poster toont de probleemstelling en de oplossing van dit eindwerk op een korte en overzichtelijke manier. Omdat dit het eindwerk op een andere manier toont en de nadruk legt op de kern van de zaak, vond ik het dan ook relevant om deze hierbij te voegen.

Development of a .NET application for the integration of fleet clients and automotive dealers



Figuur C.1: Poster eindwerk

Bijlage D

Basispagina klanttoepassing

```
1 Imports System
2 Imports System.Web.UI
3 Imports System.Web.UI.HtmlControls
4 Imports System.Web.UI.WebControls
5 Imports System.Web.Security
6 Imports System.Resources
7 Imports System.Globalization
8
9 Namespace AAS
10
11     Public Class PageBase
12         Inherits System.Web.UI.Page
13
14         Public rm As New ResourceManager("CDC.TextRes", GetType(
15             loginPage).Module.Assembly)
16         Public t As New AAS.tools
17         Public _pageTitle As String
18
19         Protected WithEvents LabelTitle As System.Web.UI.
20             WebControls.Label
21         Protected WithEvents LabelLang As System.Web.UI.
22             WebControls.Label
23         Protected WithEvents LabelFoot As System.Web.UI.
24             WebControls.Label
25         Protected WithEvents ButtonBasket As System.Web.UI.
26             WebControls.Button
27         Protected WithEvents ButtonMain As System.Web.UI.
28             WebControls.Button
29         Protected WithEvents ButtonLogout As System.Web.UI.
30             WebControls.Button
31         Protected WithEvents ButtonPromos As System.Web.UI.
32             WebControls.Button
```

```

25     Protected WithEvents ButtonHtmlHelp As System.Web.UI.
        WebControls.LinkButton
26     Protected WithEvents ButtonEN As System.Web.UI.
        WebControls.LinkButton
27     Protected WithEvents ButtonNL As System.Web.UI.
        WebControls.LinkButton
28     Protected WithEvents ButtonFR As System.Web.UI.
        WebControls.LinkButton
29
30     Public Property pageTitle() As String
31         Get
32             Return _pageTitle
33         End Get
34         Set(ByVal Value As String)
35             _pageTitle = Value
36         End Set
37     End Property
38
39     Protected Overrides Sub OnInit(ByVal e As System.
        EventArgs)
40         'set culture to browser first preference
41         t.SetCulture(Session("usr").getLangId)
42         'make page
43         BuildPage(GenerateHtmlForm())
44         MyBase.OnInit(e)
45     End Sub
46
47     Protected Sub BuildPage(ByVal Form As HtmlForm)
48         Me.Controls.AddAt(0, New LiteralControl(MakeHeader()
        ))
49         Me.Controls.Add(Form)
50     End Sub
51
52     Protected Function MakeHeader() As String
53         Dim text As String = _
54         "<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.0 _
        Transitional//EN'>" _
55         & "<html>" _
56         & "<head>" _
57         & "<title>CDC_-" & pageTitle & "</title>" _
58         & "<meta_name='GENERATOR' _content='Microsoft_Visual_
        Studio_.NET_7.1'>" _
59         & "<meta_name='CODELANGUAGE' _content='Visual_Basic_
        .NET_7.1'>" _
60         & "<meta_name='vs_defaultClientScript' _content='
        JavaScript'>" _
61         & "<link_rel='stylesheet' _type='text/css' _href='
        StyleSheet.css'>" _
62         & "</head>" _

```

```

63         & "<body>"
64         Return text
65     End Function
66
67     Protected Function GenerateHtmlForm() As HtmlForm
68         Dim form As HtmlForm = New HtmlForm
69         addHead(form)
70         addMainPanel(form)
71         addFooter(form)
72         Return form
73     End Function
74
75     Protected Sub addcontrolsFromDerivedPage(ByVal form As
76         HtmlForm)
77         Dim count As Integer = Me.Controls.Count
78         Dim i
79         For i = 0 To count - 1
80             Dim ctrl As System.Web.UI.Control = Me.Controls
81                 (0)
82             form.Controls.Add(ctrl)
83             Me.Controls.Remove(ctrl)
84         Next i
85     End Sub
86
87     Protected Sub addMainPanel(ByVal form As HtmlForm)
88         LabelLang = New Label
89         LabelLang.Visible = False
90
91         form.Controls.Add(New LiteralControl("<div_id='
92             PanelMain'>"))
93
94         form.Controls.Add(New LiteralControl("<div_id='
95             LabelLang'>"))
96         form.Controls.Add(LabelLang)
97         form.Controls.Add(New LiteralControl("</div>"))
98
99         addcontrolsFromDerivedPage(form)
100
101         form.Controls.Add(New LiteralControl("</div>"))
102     End Sub
103
104     Protected Sub addFooter(ByVal form As HtmlForm)
105         LabelFoot = New Label
106         LabelFoot.Text = rm.GetString("Foot")
107
108         form.Controls.Add(New LiteralControl("<div_id='
109             Footer'>"))
110         form.Controls.Add(LabelFoot)
111         form.Controls.Add(New LiteralControl("</div>"))

```

```
107         form.Controls.Add(New LiteralControl("</body></html>
108         "))
109     End Sub
110     Protected Sub addHead(ByVal form As HtmlForm)
111         LabelTitle = New Label
112         LabelTitle.Text = rm.GetString("Title")
113
114         ButtonBasket = New Button
115         ButtonBasket.Text = rm.GetString("Basket")
116         AddHandler ButtonBasket.Click, AddressOf
117             onButtonBasketClicked
118
119         ButtonPromos = New Button
120         ButtonPromos.Text = rm.GetString("Promos")
121         AddHandler ButtonPromos.Click, AddressOf
122             onButtonPromosClicked
123
124         ButtonMain = New Button
125         ButtonMain.Text = rm.GetString("Main")
126         AddHandler ButtonMain.Click, AddressOf
127             onButtonMainClicked
128
129         ButtonLogout = New Button
130         ButtonLogout.Text = rm.GetString("Logout")
131         AddHandler ButtonLogout.Click, AddressOf
132             onButtonLogoutClicked
133
134         ButtonHtmlHelp = New LinkButton
135         ButtonHtmlHelp.Text = rm.GetString("Help")
136         ButtonHtmlHelp.CssClass = "HelpButton"
137         AddHandler ButtonHtmlHelp.Click, AddressOf
138             onButtonHelpClicked
139
140         ButtonEN = New LinkButton
141         ButtonEN.Text = "EN"
142         ButtonEN.CssClass = "LanguageButton"
143         AddHandler ButtonEN.Click, AddressOf
144             onButtonENClicked
145
146         ButtonNL = New LinkButton
147         ButtonNL.Text = "NL"
148         ButtonNL.CssClass = "LanguageButton"
149         AddHandler ButtonNL.Click, AddressOf
150             onButtonNLClicked
151
152         ButtonFR = New LinkButton
153         ButtonFR.Text = "FR"
154         ButtonFR.CssClass = "LanguageButton"
```

```
148         AddHandler ButtonFR.Click , AddressOf
           onButtonFRClicked
149
150         form.Controls.Add(New LiteralControl("<div_id='
           PanelLang'>"))
151         form.Controls.Add(ButtonEN)
152         form.Controls.Add(ButtonNL)
153         form.Controls.Add(ButtonFR)
154         form.Controls.Add(ButtonHtmlHelp)
155         form.Controls.Add(New LiteralControl("</div>"))
156
157         form.Controls.Add(New LiteralControl("<div_id='
           LabelTitle'>"))
158         form.Controls.Add(LabelTitle)
159         form.Controls.Add(New LiteralControl("</div>"))
160
161         form.Controls.Add(New LiteralControl("<div_id='
           PanelHead'>"))
162         form.Controls.Add(ButtonBasket)
163         form.Controls.Add(ButtonPromos)
164         form.Controls.Add(ButtonMain)
165         form.Controls.Add(ButtonLogout)
166         form.Controls.Add(New LiteralControl("</div>"))
167     End Sub
168
169     Protected Sub onButtonMainClicked(ByVal sender As Object
           , ByVal e As EventArgs)
           Response.Redirect("mainPage.aspx")
170
171     End Sub
172
173     Protected Sub onButtonLogoutClicked(ByVal sender As
           Object , ByVal e As EventArgs)
           FormsAuthentication.SignOut()
174         Response.Redirect("loginPage.aspx")
175
176     End Sub
177
178     Protected Sub onButtonPromosClicked(ByVal sender As
           Object , ByVal e As EventArgs)
           Response.Redirect("promoPage.aspx")
179
180     End Sub
181
182     Protected Sub onButtonBasketClicked(ByVal sender As
           Object , ByVal e As EventArgs)
           Response.Redirect("basketPage.aspx")
183
184     End Sub
185
186     Protected Sub onButtonHelpClicked(ByVal sender As Object
           , ByVal e As EventArgs)
           Response.Redirect("WebHelp_Pro\Client_CDC.htm")
187
```

```
188     End Sub
189
190     Protected Sub onButtonENClicked(ByVal sender As Object ,
191         ByVal e As EventArgs)
192         Session("usr").setLangId("EN")
193         LabelLang.Text = rm.GetString("LangChanged")
194         LabelLang.Visible = True
195     End Sub
196
197     Protected Sub onButtonFRClicked(ByVal sender As Object ,
198         ByVal e As EventArgs)
199         Session("usr").setLangId("FR")
200         LabelLang.Text = rm.GetString("LangChanged")
201         LabelLang.Visible = True
202     End Sub
203
204     Protected Sub onButtonNLClicked(ByVal sender As Object ,
205         ByVal e As EventArgs)
206         Session("usr").setLangId("NL")
207         LabelLang.Text = rm.GetString("LangChanged")
208         LabelLang.Visible = True
209     End Sub
End Class
End Namespace 'AAS
```

Listing D.1: PageBase.vb

Bijlage E

Inhoud bijgevoegde CD-ROM

De bijgevoegde cd-rom bevat:

- De abstracts van dit eindwerk in DOC-formaat
- Dit boek in PDF-formaat
- Een backup van de centrale database (Microsoft SQL Server 2000)
- De programmacode en de bijhorende Visual Studio .NET projecten
- De Macromedia RoboHelpX5 projecten van de help functies
- De Macromedia Captivate projecten van de tutorials
- De poster van dit eindwerk in PDF-formaat

Lijst van figuren

2.1	Voorbeeld technische tekening Parts Rapido	9
3.1	Verduidelijking methodes resources	21
3.2	Deployment diagramma volledige toepassing	27
3.3	Model-View-Controller [10]	32
3.4	Activiteitendiagramma dealertoepassing (deel 1)	36
3.5	Activiteitendiagramma dealertoepassing (deel 2)	37
3.6	Activiteitendiagramma klanttoepassing (deel 1)	38
3.7	Activiteitendiagramma klanttoepassing (deel 2)	39
3.8	Klassediagramma	40
3.9	Sequentiediagramma's	41
3.10	Model database	46
5.1	Screenshot klanttoepassing	60
5.2	Screenshot mobiele versie klanttoepassing	61
5.3	Screenshot dealertoepassing	63
5.4	Onderdelen	65
5.5	Screenshot mogelijke toestanden	85
C.1	Poster eindwerk	101

Listings

5.1	Paswoordgenerator gebaseerd op [5]	66
5.2	Authentication Header klasse	67
5.3	Gebruik van de authentication header	68
5.4	Aanspreken webservice	68
5.5	Link naar DMS om goedgekeurde orders te verwerken	70
5.6	Stored procedure die link naar DMS verwerkt	70
5.7	Gebruik Webservice EU	72
5.8	Verkleinen van foto	74
5.9	Doorsturen van foto	75
5.10	Opslaan van foto op de server	75
5.11	Localization loginscherm	76
5.12	Methode om taal in te stellen	76
5.13	Taal van de gebruiker instellen	77
5.14	Redirectie naar mobiel of niet-mobiel	78
5.15	Methode voor logbestanden te schrijven	79
5.16	Declaratie timer	81
5.17	Starten van thread	81
5.18	Starten van de timer	82
5.19	Controleren op acties	82
5.20	Webservice die acties controleert	84
5.21	Configuratiebestand aanpassen	86
A.1	Structuur Parts Rapido export bestand	95
B.1	StyleSheet.css	96
D.1	PageBase.vb	102

Bibliografie

- [1] Service rapido 2.0 de volgende stap in optimale service. *DAF in action*, nummer 2:pp. 20–21, 2005.
- [2] Adobe. *Macromedia Captivate*. <http://www.adobe.com/products/captivate/>.
- [3] Adobe. *Macromedia RoboHelp X5*. <http://www.adobe.com/products/robohelp/>.
- [4] Ross CARTER. *Microsoft Real-Time Communications: Protocols and Technologies*. <http://www.microsoft.com/technet/prodtechnol/winxpro/plan/rtcprot.mspx>.
- [5] Robert CHARTIER. *Random password generator function*. <http://www.aspfree.com/c/a/ASP-Code/Random-password-generator-function-by-Robert-C/>.
- [6] Gary CORNELL and Jonathan MORRISON. *Programming VB.NET: A Guide for Experienced Programmers*. Berkeley, California: Apress, 2002.
- [7] A.J. DE JONG. *Regelgeving verlamt online-ondernemer*. <http://www.wisemen.nl/cgi-bin/media/mediadoc.cgi?PR22>.
- [8] Google. *Google Analytics*. <http://www.google.com/analytics/nl-NL/>.
- [9] Dave GRUNDGEIGER. *Programming Visual Basic .NET*. Sebastopol, California: O'Reilly, 2002.
- [10] Sun Microsystems Inc. *Java Blueprints Model-View-Controller*. <http://java.sun.com/blueprints/patterns/MVC-detailed.html>.
- [11] Frank KINT. *Homepage AAS-ICT solutions*. <http://www.aas-it.be/homepage>.

- [12] Satish KUMAR. *DebugMode Wink*. <http://www.debugmode.com/wink/>.
- [13] MICROSOFT. *Building Secure ASP.NET Applications*. Microsoft Corporation, 2002.
- [14] Course 2524B MICROSOFT. *Developing XML WebServices Using Microsoft ASP.NET*. Microsoft Corporation, 2002.
- [15] Course 2373B MICROSOFT Class Pack. *Programming with Microsoft Visual Basic .NET*. Microsoft Corporation, 2002.
- [16] Course 2524C MICROSOFT Class Pack. *Developing XML Web Services Using Microsoft ASP.NET*. Microsoft Corporation, 2002.
- [17] MSDN. *RTC Client API*. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/rtccclnt/rtc/real_time_communications_rtc_client_start_page.asp.
- [18] Alexander NAKHIMOVSKY and Tom MYERS. *Google, Amazon, and Beyond: Creating and Consuming Web Services*. Berkeley, California: Apress, 2004.
- [19] Evangelos PETROUTSOS and Asli BILGIN. *Mastering Visual Basic .NET Database Programming*. Alameda, California: Sybex Inc., 2002.
- [20] Steven ROMAN and Paul LOMAX. *Visual Basic .NET Language in a nutshell*. Sebastopol, California: O'Reilly, 2001.
- [21] Christian SCHENK. *MiKTeX*. <http://www.miktex.org/>.
- [22] Citrix Systems. *Citrix Presentation Server*. http://www.citrix.com/English/ps2/products/product.asp?contentID=186&ntref=hp_nav_US.
- [23] Openwave Systems. *Openwave Phone Simulator*. http://developer.openwave.com/dvl/tools_and_sdk/phone_simulator/.
- [24] Newera Software Technology. *IconCool Studio*. <http://www.iconcool.com/iconcoolstudio.htm>.
- [25] Europese Unie. *Webservice voor de controle van BTW-nummers*. http://europa.eu.int/comm/taxation_customs/vies/api/checkVatPort?wsdl.
- [26] Europese Unie. *Wettelijke bepalingen van elektronische handel*. <http://europa.eu.int/scadplus/leg/nl/lvb/l24204.htm>.

- [27] Craig UTLEY. *A Programmer's Introduction to Visual Basic .NET*. Indianapolis, Indiana: Sams Publishing, 2001.
- [28] Annemie VORSTERMANS. *Cursus InternetTechnologie*. <http://ingenieur.kahosl.be/personeel/Annemie.Vorstermans/Internet/index.htm>.
- [29] Andy WIGLY and Peter ROXBURGH. *Building Microsoft ASP.NET Applications for mobile devices*. Redmond, Washington: Microsoft Press, second edition, 2003.
- [30] Wikipedia. *ADO.NET*. <http://en.wikipedia.org/wiki/ADO.NET>.
- [31] Wikipedia. *ASP.NET*. <http://en.wikipedia.org/wiki/ASP.NET>.
- [32] Wikipedia. *Cascading Style Sheets*. http://en.wikipedia.org/wiki/Cascading_Style_Sheets.
- [33] Wikipedia. *Citrix Presentation Server*. http://en.wikipedia.org/wiki/Citrix_Presentation_Server.
- [34] Wikipedia. *Citrix Systems*. <http://en.wikipedia.org/wiki/Citrix>.
- [35] Wikipedia. *Https*. <http://en.wikipedia.org/wiki/Https>.
- [36] Wikipedia. *Internet Information Services*. <http://en.wikipedia.org/wiki/IIS>.
- [37] Wikipedia. *Microsoft SQL Server*. http://en.wikipedia.org/wiki/Microsoft_SQL_Server_2000.
- [38] Wikipedia. *Model-view-controller*. <http://en.wikipedia.org/wiki/Model-view-controller>.
- [39] Wikipedia. *.NET Framework*. http://en.wikipedia.org/wiki/.NET_framework.
- [40] Wikipedia. *Transact-SQL*. <http://en.wikipedia.org/wiki/T-sql>.
- [41] Wikipedia. *Transport Layer Security*. http://en.wikipedia.org/wiki/Secure_Sockets_Layer.
- [42] Wikipedia. *Unified Modeling Language*. http://en.wikipedia.org/wiki/Unified_Modeling_Language.
- [43] Wikipedia. *Visual Basic .NET*. <http://en.wikipedia.org/wiki/VB.NET>.

- [44] Wikipedia. *Web service*. http://en.wikipedia.org/wiki/XML_Web_Services.
- [45] Wikipedia. *Wink (tutorial software)*. http://en.wikipedia.org/wiki/Wink_%28tutorial_software%29.

Index

- ADO.NET, 2, 54, 55
- analyse, 3, 4, 18, 90, 93
 - analysefase, 7, 90, 93
 - analyseproces, 7, 16
- API, 24, 25
- ASP.NET, 2, 49, 52, 55, 60
- authenticatie, 53
- authentication, 29, 30, 66, 68, 85

- BTW-nummer, 72, 80

- C#, 49, 52, 55
- Citrix, 26, 48, 49
- code-behind, 53, 55, 56

- Dealer Management System, 1, 6, 13, 19, 69, 70, 92
- DMS, 1, 6, 7, 10, 13, 19, 42, 69, 71, 92

- firewall, 24, 25, 29, 60
- framework, 52, 54, 55
- FTP, 25, 26, 47

- hardware, 57
- HTML, 31, 49, 55, 56
- HTTP, 22, 23, 48, 56, 57
- HTTPS, 25, 29, 48, 49, 56, 57, 60

- icoon, 14, 35, 42–44, 51, 52, 84
- IIS, 2, 5, 7, 22–24, 26, 47, 48, 56, 60, 61, 64, 73, 75, 79
- Internet Information Services, 2, 47

- JavaScript, 55, 60

- Linux, 48, 50

- Microsoft, 2, 34, 47–49, 51–55
- Model-View-Controller, 31, 55, 93

- namespace, 18–20, 54
- NAT, 25

- object-geörienteerd, 52, 93
- open source, 51, 57

- Parts Rapido, 8, 10, 15, 28, 90, 95
- PocketPC, 2, 5, 14, 17, 25, 31, 49, 51

- router, 24, 25, 29, 60
- RTC, 24, 25

- SOAP, 23, 30, 55, 56, 64, 67
- socket, 24, 26
- software, 1, 3, 4, 6, 28, 48–52, 55, 57, 94

- SQL, 47
 - MySQL, 54
 - SQL database, 54
 - SQL server, 5, 15, 22, 23, 26, 34, 44, 47, 53, 54, 60, 62, 64
 - SQL Server 2000, 2, 47
 - SQL server 2000, 47
 - SQL-connectie, 19, 23, 54
 - SQL92, 47
- SQL Server, 42
- SSL, 29, 30, 56, 57, 61, 64
 - SSL-certificaat, 29
 - SSL-encryptie, 30, 60
- stored procedure, 47, 53, 54, 69, 70, 84

- system tray, 35
- T-SQL, 2, 31, 47, 53
- Task Scheduler, 34, 62
- terminal, 48
- UML, 18, 34, 49, 57
- UPnP, 25
- Visual Basic, 52
 - VB.NET, 2, 52, 53, 55, 56
 - VB6, 1, 53
 - Visual Basic .NET, 49, 52
- Visual Studio, 2, 49, 51
- W3C, 56
- WAP, 56
- wederzijdse uitsluiting, 72, 73
- Windows, 20, 24, 26, 29, 34, 35, 47–
53, 55, 62, 64, 69, 79, 86
- winkelmand, 11, 14, 16, 17, 22, 29,
31, 33, 42–44
- XHTML, 49, 56
- XHTML-MP, 56
- XML, 18, 19, 23, 49, 55, 56, 62, 64,
76, 85, 86, 91