19/3/2012

IFOODS

ONTWIKKELEN VAN EEN KASSA SYSTEEM

Auteur: John Renooij

Opdrachtgever : Ifoods
Opleiding : Informatica
Versie : Final

Voorwoord

Voor u ligt het afstudeerverslag van John Renooij, student aan de Haagse Hoge School ingeschreven onder studentnummer 20041267.

Middels dit voorwoord wil ik de heer Keuning van de firma Ifoods bedanken voor het vertrouwen tijdens de ontwikkeling van de project. Daarnaast wil ik ook de heer Alten bedanken voor zijn begeleiding en ondersteuning tijdens het project.

Daarnaast wil ik de mensen bedanken die tijdens het project mij hebben voorzien van wijze raad, (goed bedoelde) adviezen en commentaar.

John Renooij
16 maart 2012, Ifoods, Haarlem

Inhoud

1. Inleiding

Het project “ontwikkeling van een kassa systeem” is tot stand gekomen door de toenemende vraag vanuit restaurants. De vraag naar een simpele en doeltreffende manier om digitale bestellingen te monitoren en te verwerken. Daarnaast is er een groeide vraag naar alternatief voor de reeds dure bestaande systemen. Ifoods had de wens voor deze twee problemen één oplossing te vinden.

Het document bestaat uit 3 delen. In het eerste gedeelte van het verslag zal worden ingegaan op het probleem, de bestaande situatie en hoe de nieuwe situatie er uit zou moeten zien.

Het tweede gedeelte is een verslaglegging van de diverse activiteiten die zijn uitgevoerd tijdens het project.

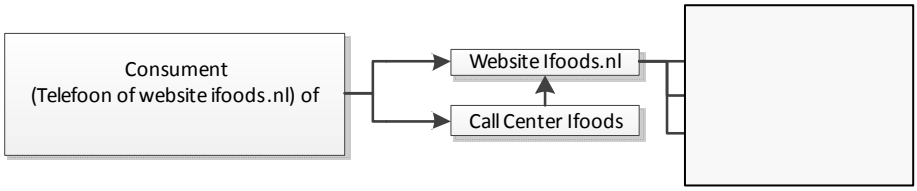
In het laatste gedeelte van het document kunt u de verantwoording ten opzichten van de gekozen beroepstaken terug vinden alsmede mijn eigen reflectie op het project.

2. Opdracht en Organisatie

2.1 Opdrachtgever

De opdracht wordt uitgevoerd binnen de organisatie Ifoods. Ifoods treedt op als intermediair tussen consumenten en diverse bezorgrestaurants. Het doel van Ifoods is het ontwikkelen en aanbieden van de beste “Eten-bestelsite” en daarmee de bestellingen zo snel en betrouwbaar mogelijk door te geven. Het streven is daarbij de site zo toegankelijk mogelijk te maken voor zowel consument als restaurant.

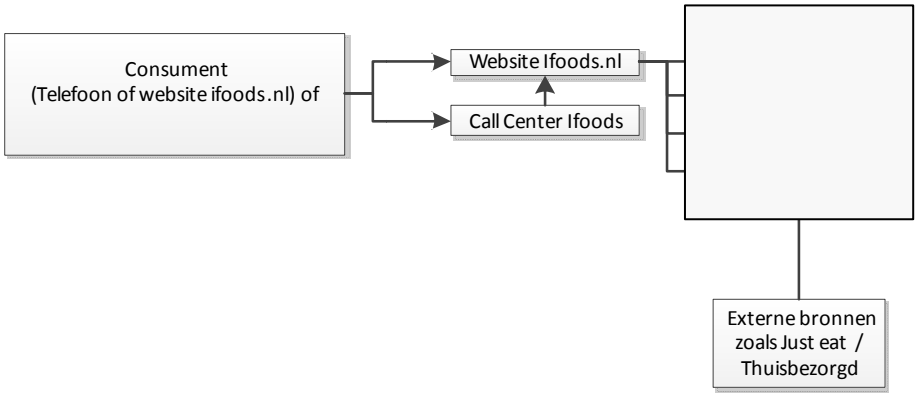
De contactpersoon binnen de organisatie is de heer R. van Aalten. De heer van Aalten is manager ICT binnen Ifoods. Daarnaast zal er ook regelmatig contact zijn met de heer Keuning, directeur van Ifoods.



Figuur 1: De intermediair rol van Ifoods tussen consument en restaurants

Zoals in figuur 1 zichtbaar is, levert Ifoods een dienst aan consumenten en restaurants. Via de website van Ifoods kan een klant een keuze maken tussen aangesloten restaurants op basis van de klant zijn postcode. Eventueel kunnen consumenten ook bellen met het callcenter van Ifoods om op deze manier een bestelling te plaatsen. Het callcenter gebruikt op dit moment dezelfde website om de bestelling te verwerken. Daarnaast levert Ifoods ook een services aan de restaurants door het maken en onderhouden van onder andere de menukaarten waardoor Ifoods een verlengstuk van de restaurants is geworden.

Bij het verwerken van een bestelling, die binnen is gekomen bij Ifoods, zijn er verschillende paden die een bestelling kan volgen. Op dit moment worden de bestelling voor de meeste restaurant (99%) doorgestuurd per fax of email. Voor een enkel bedrijf wordt de bestelling doorgegeven per telefoon. Dit is meestal het geval als bekend is dat er een tijdelijke storing is bij het restaurant met betrekking tot email of de fax. Ifoods heeft verschillende monitortechnieken die er voor zorgen dat de bestellingen en de ontvangst daarvan bij het restaurant goed zijn verlopen.



Figuur 2 De nieuwe Kassa applicatie en de plaats die het inneemt in bestaande situatie

2.2 Opdracht

2.2.1 Probleemstelling

Door de opkomst van onder andere het internet zijn er een aantal dingen veranderd voor restauranthouders. Waar vroeger enkel de bestellingen aan de balie of per telefoon werden doorgegeven, zijn er tegenwoordig een aantal kanalen bij gekomen. Bijvoorbeeld de bestellingen die binnen komen op de website van het restaurant maar ook bestellingen die worden gegenereerd via intermediairs zoals Ifoods en Just-Eat.

Deze externe bestellingen worden vaak doorgebeld of komen binnen doormiddel van fax en / of email. Door de verschillende kanalen, gebeurt het regelmatig dat een bestellingen te laat opgemerkt wordt of gewoon verdwijnt waardoor de klant langer moet wachten of zelfs helemaal vergeten wordt.

Voor veel restaurants is de aanschaf van een POS-systeem (Point of Sale systeem. Een kassa systeem veelal gebruikt in de horeca / restaurants) een (te) grote investering, die vaak niet opweegt tegen de voordelen. Daarom wordt er in deze restaurants vaak gewerkt met losse papiertjes of worden bestellingen gewoon uit het hoofd bereid en eventueel bezorgd.

Door het gebrek van een goed systeem is het lastig voor de eigenaar een goed en sluitend overzicht te maken met betrekking tot de administratieve cijfers. Ook worden er vaak fouten gemaakt tijdens het bereiden van de bestelling.

2.2.2 Doelstelling

De doelstelling van het project is het ontwikkelen van een applicatie die het mogelijk moet maken de administratieve handeling, met betrekking tot de bestellingen, te automatiseren en daarmee te verbeteren. Het streven is het aanbieden van een simpel en betaalbaar POS systeem die bestellingen van externe bronnen, zoals Ifoods, automatisch verwerkt. Tevens moet het systeem modulair uitbreidbaar zijn zodat in de toekomst het systeem ook gebruikt kan worden voor ondersteuning van andere intermediairs en/of technieken.

Zoals in het Plan van Aanpak opgenomen zijn er een aantal randvoorwaarden waaraan de applicatie dient te voldoen zoals:

- Gebaseerd op Windows platform
- eenvoudig te bedienen zijn
- intuïtief aanvoelen in gebruik
- Bedienbaar door middel van touchscreen

Om de restauranthouder een beter overzicht te geven dient de applicatie een aantal mogelijkheden bieden met betrekking tot de bedrijfsadministratie. Naast de optie om bijvoorbeeld statistieken in het programma te kunnen opvragen dient de applicatie ook de mogelijkheid te bieden om gegevens te exporteren ter behoeve van externe administratie software.

3. Plan van aanpak

Na het eerste overleg met de opdrachtgever is een plan van aanpak opgesteld. Dit plan van aanpak is bedoeld voor zowel de opdrachtgever als voor mij als opdrachtnemer.

In het plan van aanpak komen onder andere de volgende punten betreft de opdracht aan de orde:

- Definitie van betrokken personen
- Initiële opdrachtomschrijving
- Te gebruiken technieken en tools
- Afbakening van de opdracht
- Randvoorwaarden
- Risicofactoren

Het plan van aanpak zal worden gebruikt als wegwijzer gedurende dit project. Door het van te voren vastleggen van bijvoorbeeld de gebruikte technieken en tools heeft het project vanaf begin een goede structuur.

Voor een kopie van het plan van aanpak wordt u verwezen naar bijlage B.

3.1 Methode, Technieken en Tools

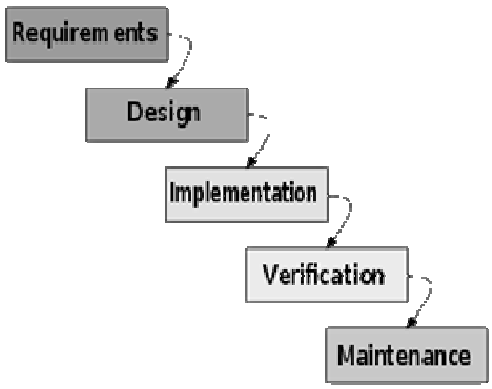
3.1.1 Methode

Tijdens dit project zal gebruik worden gemaakt van de RUP methode als ontwikkelmethode.

Bij aanvang van het project is gekeken naar welke ontwikkelmethode het beste aansloot bij het project. Er is gekeken naar het traditionele waterval methode, RUP, Spiraal methode en ASD.

waterval methode

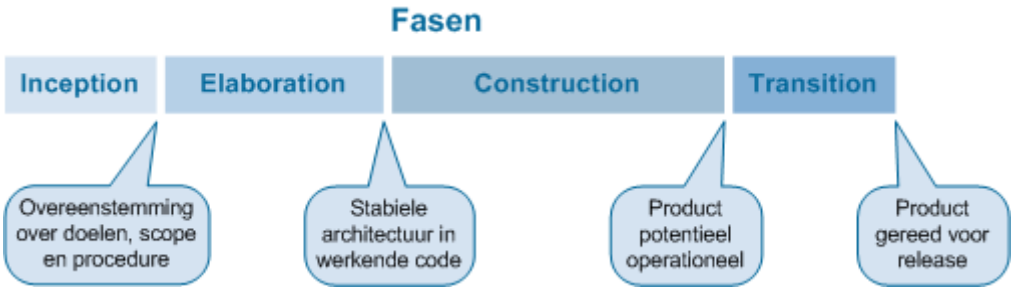
Het waterval methode is als een trein die niet omgekeerd kan worden. Er worden een aantal fases gedefinieerd en het idee is dat je van de ene fase naar de volgende fase vloeit. Net als bij een echte waterval is het niet mogelijk om terug te keren naar een eerdere fase op het moment dat er een fout wordt gevonden. Op het moment dat de requirements definitief zijn is er geen weg meer terug tenzij het project van voor af aan wordt uitgevoerd.



Figuur 3 Standaard definitie van waterval model

RUP (Rational Unified Process)

RUP is een iteratief / incrementeel software ontwikkelingsproces. Incrementeel ontwikkelen stelt je in staat om het project op te delen in een aantal deelproducten en deze afzonderlijk van elkaar op te leveren. Dit verkleint het risico dat het eindproduct niet is wat de klant wil en de klant kan nu per deelproduct zijn feedback geven wat je weer kunt meenemen tijdens de ontwikkeling van het volgende deelproduct.



Figuur 4 Standaard definitie van het RUP model

Spiraal methode

Spiraal methode is een methode gebaseerd op het uitvoeren van activiteiten die het hoogste risico met zich meebrengt. Men begint dus met het uitvoeren van het onderdeel met het hoogste risico. Op die manier wordt het mogelijk gemaakt om een bepaald deel van het systeem te implementeren, terwijl een ander deel nog ontworpen moet worden. De meest risicodragende problemen, worden in het begin van een project ingeschat en onder handen genomen. Het vertrouwen in het slagen van het project is hierdoor groot. Ook wordt eerder duidelijk dat een project onuitvoerbaar blijkt te zijn.

Risico's zijn voor een opdrachtgever kleiner, dan bij gebruik van de watervalmethode. Het nadeel is dat het maken van een planning vaak onmogelijk is. Dit komt omdat het van tevoren inschatten van de risico's, of iteraties niet voldoende kan. De opdrachtgever kan er daardoor niet zeker van zijn, wanneer het project zal zijn afgerond. Dit is overigens voor het watervalmodel ook van kracht. Daar is het onzeker wat de doorlooptijd zal zijn van de verschillende fasen.

ASD (Adaptive Software Development)

ASD is een ontwikkelmethode die zich richt op snelle resultaten en korte herhalende fases. Hierdoor moeten knopen snel worden doorgehakt maar is er tegelijkertijd veel ruimte om brede kaders te houden. Omdat niet aan het begin alle requirements worden gedefinieerd kan op basis van veranderende eisen en wensen de koers van de ontwikkeling worden bijgestuurd.

Conclusie

ASD viel als ontwikkelmethode voor ons af omdat deze zich het beste leent voor ontwikkeling van software waarbij er alleen vage omschrijvingen zijn en de koers van de ontwikkeling drastische kan veranderen. Het waterval model viel voor ons af omdat dit model te stroef is. Het bied geen mogelijkheden om gaande weg iets toe te voegen aan het project. Het spiraalmodel was ook geen goede keuze omdat dit zich het meeste richt op verschillende functionaliteiten binnen 1 applicatie.

Het RUP model past vrijwel direct op dit project. Het model bied de mogelijk om vooraf een goede definitie te maken van de te ontwikkelen software of delen daarvan maar bied daarnaast ook de ruimte om wijzigingen aan te brengen onderweg zonder dat hiervoor het hele project overnieuw moet worden gestart.

Zoals in figuur 4 al zichtbaar was, kenmerkt RUP zichzelf door het specificeren van 4 fases namelijk:

- Aanvang (Inception)
  Tijdens deze fase worden de wensen en eisen van de te ontwikkelen applicatie vastgelegd. Ook worden de randvoorwaarde en eventueel risico's vastgesteld
- Detaillering (Elaboration)
  Tijdens deze fase worden de eerder genoemde wensen en eisen omgezet in technische diagrammen bedoeld ter ondersteuning van de ontwikkelaar
- Bouw (Construction)
  Tijdens deze fase worden de diagrammen vertaald naar een werkelijk programma.
- Overgang (Transition)
  De applicatie wordt overgedragen aan de klant / gebruiker voor testen en gebruik

Als gevolg van de wensen en eisen, zijn er verschillende applicaties ontwikkeld. Voor elke applicatie is het proces doorlopen. In het hoofdstuk 4.1 wordt gedefinieerd welke applicaties er zijn ontwikkeld.

3.1.2 Technieken en Technologieën

- UML "Unified Modeling Language" zal gebruikt worden voor het maken van use cases en het ontwerpen van diverse diagrammen. Deze techniek zal worden gebruikt, om overeenstemming tussen opdrachtgever en nemer te bereiken. Tevens ondersteunen de diagrammen tijdens het ontwikkelen van de verschillende onderdelen.
- Visual C# .net 2.0 en Visual C# .net 4.0
  Het systeem zal ontwikkeld worden in Microsoft Visual C# versie 2.0 en versie 4.0.
- Windows Presentation Foundation (WPF)
  Ten behoeve van de interface tussen gebruiker en applicatie zal gebruikt worden gemaakt van het nieuwe WPF framework welke onderdeel is van Microsoft .Net 4.0.

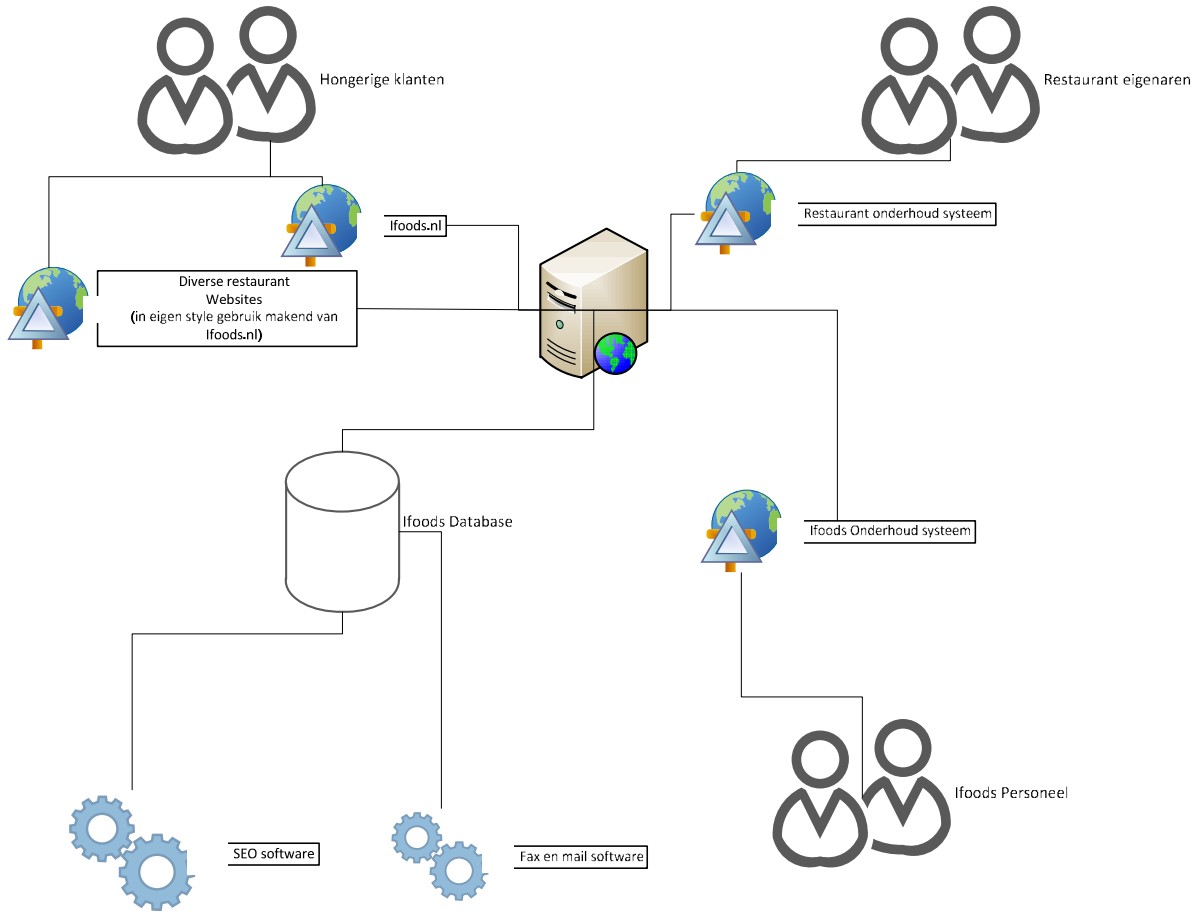
Voor de onderbouwing van de laatste twee keuzen wordt u verwezen naar de hoofdstukken WPF en Systeemarchitectuur.

4.0 Systeemarchitectuur

4.1 Globale Systeem Architectuur

4.1.1 Bestaande Architectuur

Ifoods heeft al een bestaand systeem waar dagelijks door een aantal partijen gebruik van wordt gemaakt. Zoals in de afbeelding hieronder te zien is zijn er drie groepen gebruikers die dagelijks gebruik maken van diverse verschillende onderdelen van Ifoods.



Figuur 5 Bestaande Architectuur Ifoods

Dagelijks bestellen mensen via de website van Ifoods.nl en diverse andere restaurant websites hun middag of avond eten. Door naar de website van Ifoods te navigeren en het invoeren van de postcode kunnen gebruikers hun eten bestellen bij alle aangesloten restaurants. Door een keuze te maken voor een aflevertijd, kan zelfs in de ochtend al worden besteld voor de lunch of het diner of gewoon voor de lekkere trek voor bij de voetbalwedstrijd.

Terwijl de gebruikers bijvoorbeeld hun pizza bestellen, zijn de medewerkers van Ifoods in de weer met het bijwerken van menukaarten, het toevoegen van restaurants, het monitoren van de reeds geplaatste bestellingen en het beantwoorden van vragen van klanten en restauranthouders.

Een van de tools die zij daar dagelijks voor gebruiken is het Ifoods Onderhoud systeem. Dit webbased systeem bied de medewerkers de gereedschappen om alle aspecten van een restaurant te kunnen aanpassen. Dit betekend dat de medewerken een prijswijziging kan doorvoeren maar ook bijvoorbeeld, al dan niet tijdelijk, de openingstijden kunnen aanpassen. Daarnaast bied het ook de tools om het mail en fax verkeer in de gaten te houden.

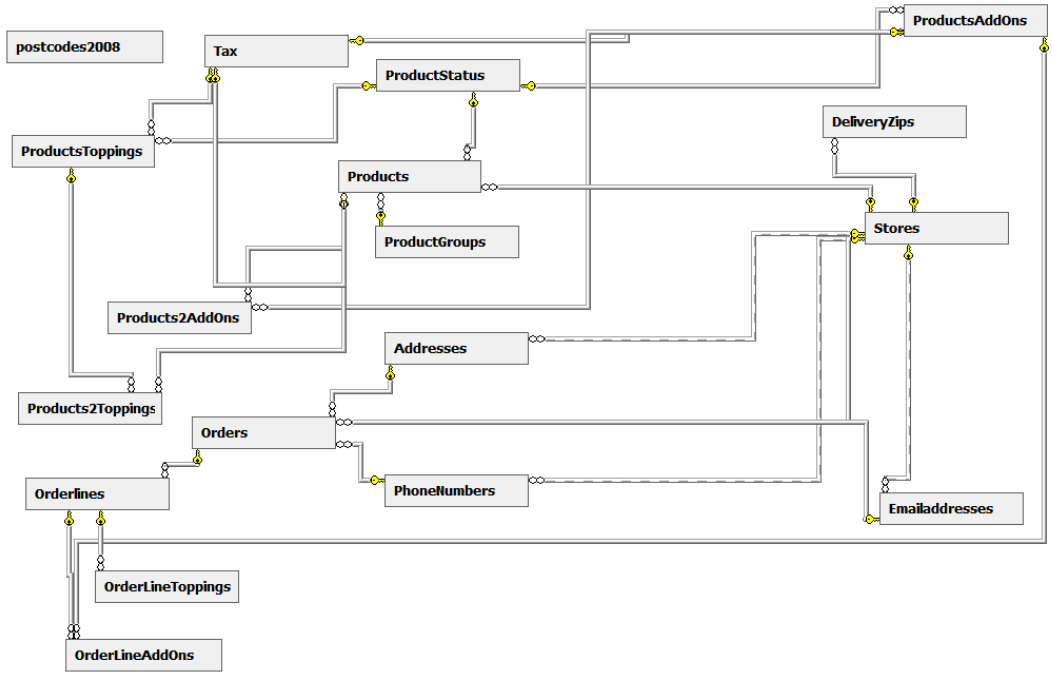
Mocht er een klant bellen met een vraag over zijn bestelling, bied het onderhoud systeem ook inzicht in alle lopende bestellingen zodat de medewerker snel de juiste gegevens bij de hand heb. Via het systeem is het ook mogelijk om een bestelling te annuleren of indien nodig opnieuw bij het restaurant aan te bieden.

Omdat het systeem van Ifoods gebaseerd is op provisiebasis, dient het systeem als primaire bron voor het versturen en genereren van facturen. Het systeem bied ook de mogelijkheid voor het exporteren van gegeven naar het boekhoud programma gebruikt door Ifoods.

Voor de restauranthouders is er een speciale website ontwikkeld waarbinnen de restauranthouder een groot gedeelte van de gegevens zelf kan bewerken. Verder kan de restauranthouder zijn facturen inzien.

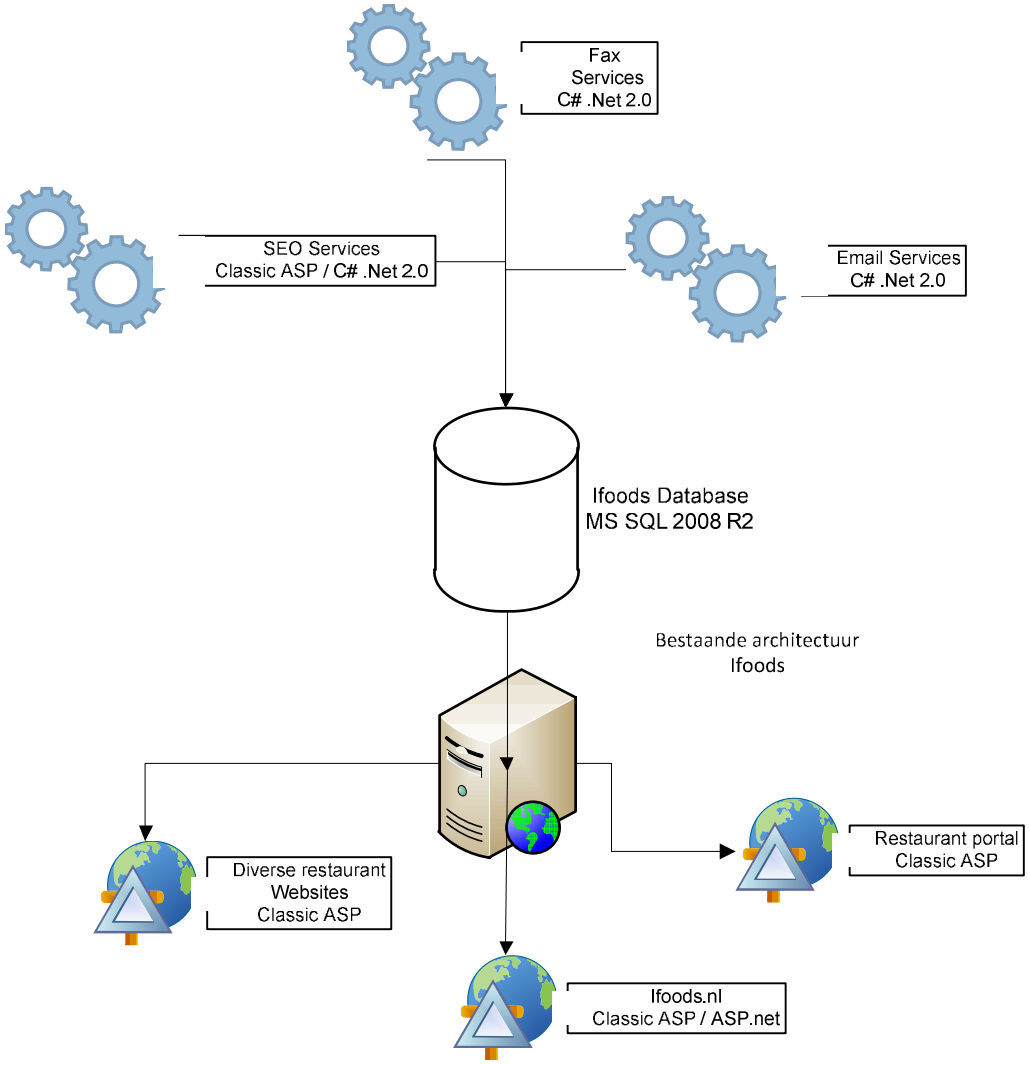
Ter behoeve van de vindbaarheid op onder andere google draaien er elke nacht een aantal losse tools. Deze tools maken bijvoorbeeld landingspagina's aan, werken de keywords bij en genereren statische pagina's ter behoeve van de laadsnelheid.

De verschillende systemen worden gevoed met data uit een enkele SQL database. Deze database is in 2004 ontwikkeld en vervolgens mee gegroeid met de databehoefte van Ifoods.nl.



Figuur 6 Belangrijkste database tabbelen met onderlingen relaties (ifoods.nl)

Zoals in onderstaande afbeelding te zien is, is er een diversiteit aan verschillende ontwikkel omgevingen die gebruikt worden binnen Ifoods. Aan de kant van de webserver zijn de meeste componenten / website ontwikkeld in “Classic ASP” omgeving. Classic ASP is de directe tegenhanger van PHP en lijkt voor een groot gedeelte op Visual Basic Script.
De overige componenten zijn de non-webbased componenten.Deze zijn ontwikkeld in de Microsoft .Net omgeving.



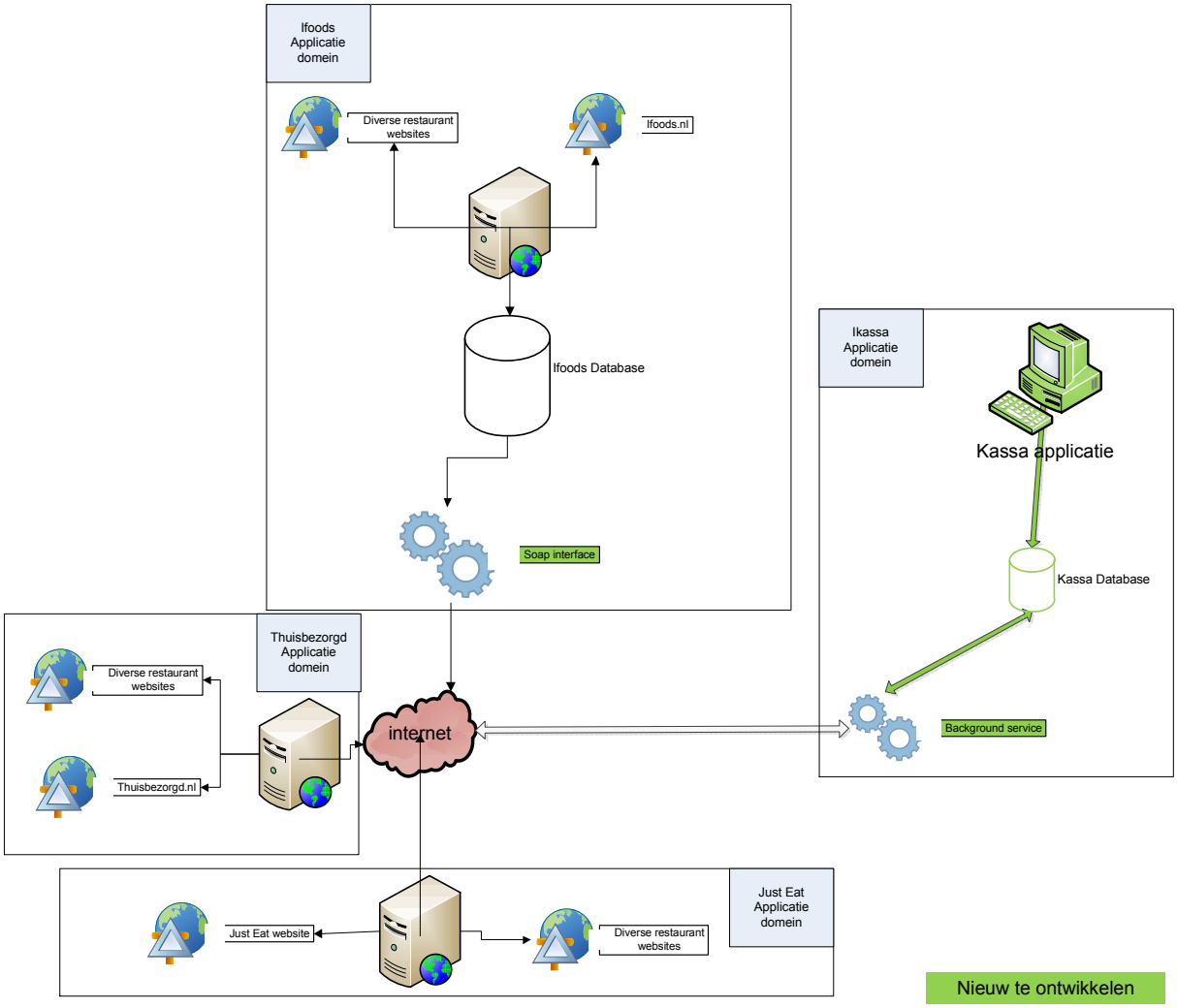
Figuur 7 Diverse componenten binnen het Ifoods Domain

4.1.2 Nieuwe Architectuur

In afbeelding 8 wordt getoond, zijn er een aantal verschillende applicaties die moeten worden ontwikkeld.

Om diverse informatie beschikbaar te stellen aan de kassa applicatie, moet er een webservice worden ontwikkeld die beschikbare informatie ontsluit uit de Ifoods Database. De informatie bevat bijvoorbeeld de bestellingen maar ook de reeds bekende menukaart. Omdat deze webservice gebruik maakt van de Ifoods database, zal deze worden geplaatst binnen het reeds bestaande Ifoods applicatie domein.

Om gegevens van externe partijen te kunnen verzamelen, moet er een Windows service worden ontwikkeld. Een Windows service is een applicatie zonder GUI die op de achtergrond binnen Windows wordt uitgevoerd. Deze Windows service zal met diensten communiceren zoals Ifoods, Just Eat en Thuisbezorgd. De uit te wisselen data en gebruikte methodes zullen in latere hoofdstukken worden uitgewerkt.

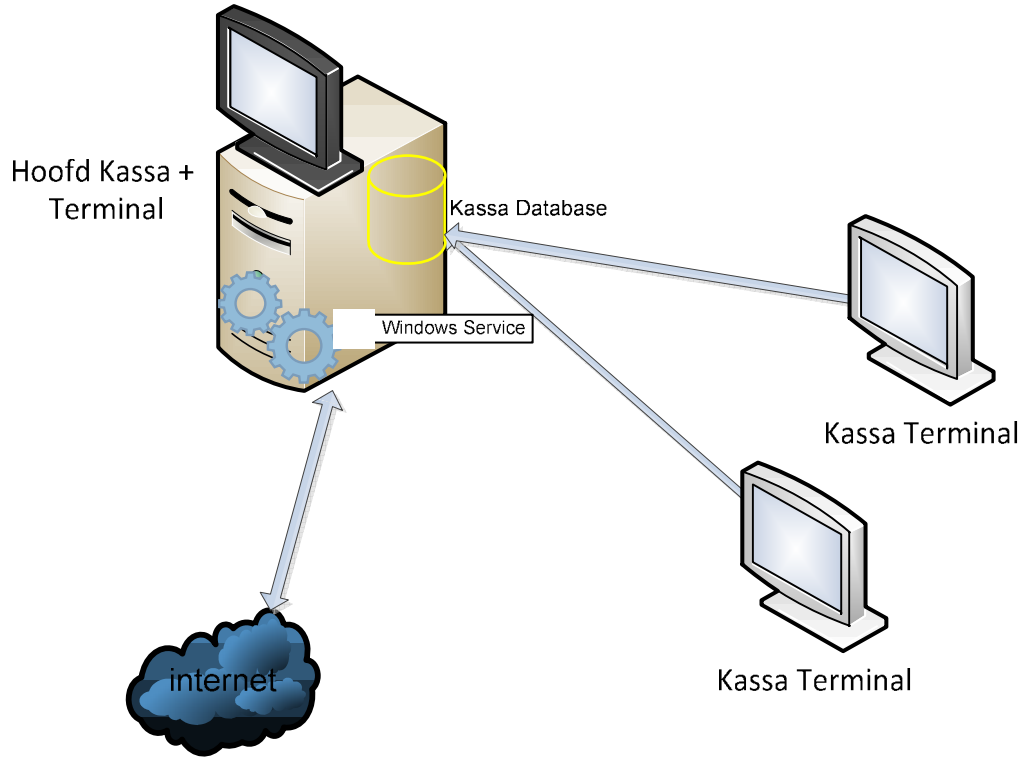


Figuur 8 Nieuwe en bestaande architectuur (Groene componenten zijn nieuwe onderdelen)

Om de data op te slaan zal er een nieuwe database worden ontwikkeld. Deze database kan worden gebruikt door meerdere kassa applicaties en Windows service. De database en Windows service zullen worden geplaatst op een van de kassa systemen in het restaurant.

De nieuw te ontwikkelen kassa-applicatie zal de interface bieden tussen de database en de restaurant medewerker. hier kan de medewerker zijn bestellingen zien, invoeren en wijzigen. Ook zal deze applicatie worden gebruikt om wijzigingen door te voeren voor bijvoorbeeld productinformatie en andere parameters ter behoeve van diverse externe partijen.

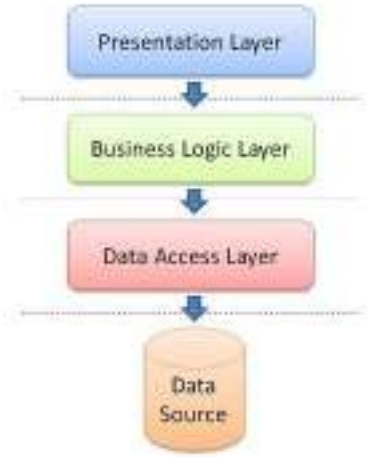
Door de opbouw van 1 centrale database is het mogelijk om meerdere kassa's met 1 database te verbinden. Onderstaande afbeelding geeft een schematische weergave hiervan. Omdat informatie behoefde kan verschillen per plaats in het restaurant, is het mogelijk om in de keuken bijvoorbeeld het keukenscherm te tonen terwijl de applicatie aan de balie in de kassa modus is geschakeld.

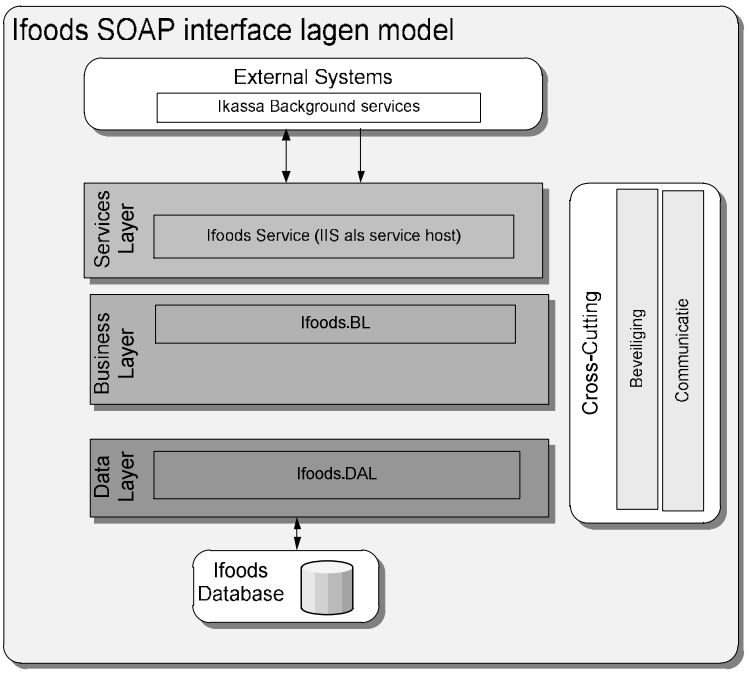


Figuur 9 Nieuwe Architectuur Ikassa. Weergave van 1 restaurant

Voor het ontwikkelen van de nieuwe onderdelen zal waar mogelijk het n-tier architectuur model worden gehanteerd. Dit model bestaat uit een aantal lagen. Deze lagen hebben allemaal hun eigen taak en kunnen los van elkaar vervangen worden. Mocht Ifoods in de toekomst een nieuwe techniek willen gebruiken voor het aanbieden van informatie, hoeft alleen de presentatie laag vervangen te worden. De business Logic en Data Acces Layer blijven gewoon het zelfde.

Een ander groot voordeel is dat de onderlinge lagen herbruikbaar zijn in andere projecten die bijvoorbeeld gebruik maken van dezelfde databronnen. Een voorbeeld hiervan is dat de DAL die nu gebruikt wordt in de nieuw te ontwikkelen web interface. Deze kunnen hergebruikt worden als de legacy systemen zoals fax en email service herschreven worden.





Figuur 10 Standaard n-tier model en implementatie zoals in webservice interface

4.1.3 Ontwikkeling volgorde nieuwe Architectuur

Zoals eerder genoemd, bestaat de nieuwe oplossing uit meerdere softwarecomponenten. Omdat er al een bestaande databron aanwezig is, zou het onlogische zijn om te beginnen aan de kant van de kassa applicatie. Dan zou namelijk voor de test van de applicatie eerst testdata gegenereerd moeten worden terwijl deze data eigenlijk al beschikbaar. Daarom is besloten de ontwikkeling volgorder van applicaties gelijk te trekken met de datastroom.

De grootste bron van informatie bevind zich in de database in van Ifoods. Daarom hebben wij gekozen om te beginnen met het ontwikkelen van de webservice die de informatie ontsluit die nodig is in de kassa applicatie.

Om de beschikbare informatie van Ifoods, Just Eat en thuisbezorgd te verwerken is het nodig om een de Windows service te ontwikkelen aan de kant van de het restaurant. Deze Windows service kan interfacen met de eerder ontwikkelde webservice voor het ophalen van product informatie, bestelingen en andere relevante gegevens. Daarnaast ook de bestellingen binnenhalen van Just Eat en Thuisbezorgd.

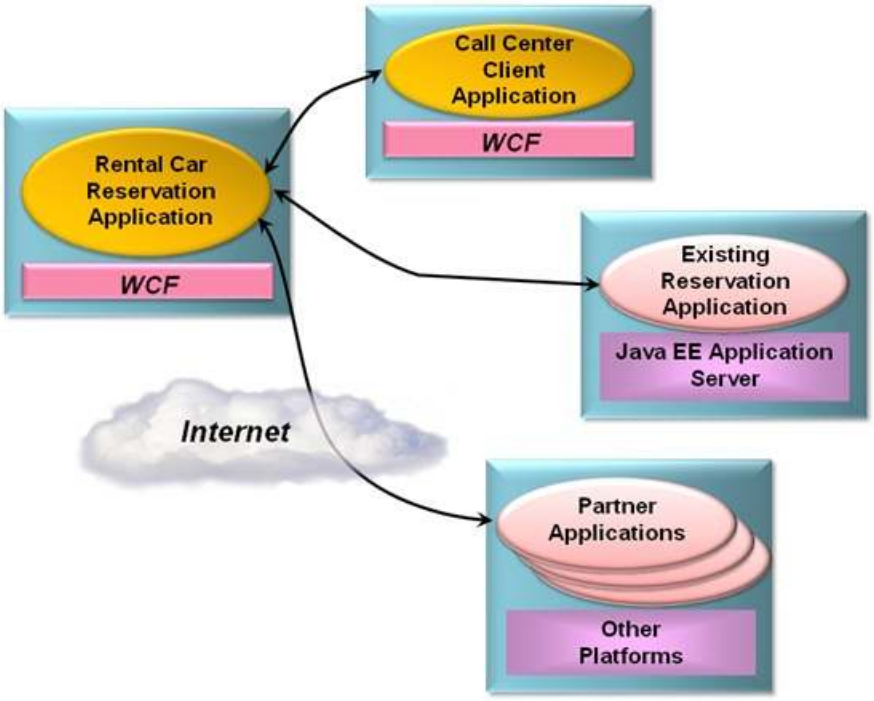
Voordat de Windows service zijn data kan opslaan, moet er eerst een nieuwe database worden ontwikkeld.

Nadat alle informatie is opgeslagen zal de kassa applicatie worden ontwikkeld. Deze zal op zijn beurt de gegevens gebruiken die reeds in de database staan opgeslagen.

Na het ontwikkelen van de kassa systeem zullen de bestaande componenten worden uitgebreid zodat waar mogelijk bijvoorbeeld status meldingen terug gestuurd kan worden naar de intermediairs.

4.2 Windows Communication Foundation (WCF) of .Net 2.0 webservice

Tijdens het ontwikkelen van het project is er gekeken naar welke technieken gebruikt kunnen worden ten behoeve van data uitwisseling. Zoals eerder beschreven is de bestaande architectuur ontwikkeld in een mix van legacy ASP code en C# .Net v1.1 en C# .Net v2.0. Daarom ligt het ook in de lijn van verwachting om een nieuw te ontwikkelen systeem te bouwen met gebruik van Microsoft technologie.



Figuur 11 Voorbeeld WCF implementatie

Met de komst van versie 4.0 van het .Net Framework is het mogelijk geworden om een generieke service te ontwikkelen ten behoeven van data uitwisseling. Dit nieuwe framewerk vervangt en bundelt een aantal communicatie technieken en bibliotheken tot een nieuw framework.

	ASMX	.NET Remoting	Enterprise Services	WSE	System. Messaging	System. Net	WCF
Interoperable Web Services	X						X
Binary .NET –.NET Communication		X					X
Distributed Transactions, etc.			X				X
Support for WS-* Specifications				X			X
Queued Messaging					X		X
RESTful Communication						X	X

WCf heeft een aantal voordelen zoals:

- More protocol options
  WCF ondersteund naast HTTP ook Nettcp, MSMQ, IPC en andere veel gebruikte communicatie protocollen.
- Write your service once, and expose it on multiple endpoints
  Doordat WCF een generiek framework biedt, is het mogelijk om de service tegelijk op meerdere protocollen in te zetten.
- self-hosting
  Het is mogelijk om een WCF services te deployen als losse applicatie in bijvoorbeeld een Windows Forms of Windows console applicatie. Daarnaast is het mogelijk om hem als webservices aan te bieden in combinatie met WAS / IIS.
- Reliable sessions, lot more security options
  Bij WCF zijn in tegenstelling tot de oude ASMX (.Net 2.0 webservice) een groot aantal verbeteringen doorgevoerd. Daardoor is het nu makkelijker om een veilige webservice te ontwikkelen

Naast bovengenoemde voordelen, levert WCF in combinatie met een .Net client applicatie tot wel 50% snelheid winst. Di t percentage is afhankelijk van de gekozen communicatie protocollen en hoeveelheid opgevraagde informatie. Op de website van Microsoft is hier uitgebreid informatie terug te vinden. (Microsoft)

Helaas bleek tijdens de eerste testen dat het niet makkelijk was om WCF te combineren met de reeds bestaande architectuur. Ondanks dat Microsoft het mogelijk heeft gemaakt om ook WCF toe te passen binnen IIS6 (de standaard webserver van Microsoft Windows Server 2003) is de combinatie van classic asp, ASP 2.0 en ASP 4.0 niet aan te raden. Verder is WCF voor een aantal van zijn voordelen afhankelijk van WAS. Windows Process Activation Service (WAS) is onderdeel van het nieuwere Windows Server 2008 platform. Daarnaast was dit voor zowel mij als Ifoods de eerste aanraking met WCF. Daarom is besloten om het op dit moment niet te gebruiken bij het ontwikkelen van de interface.

Als alternatief zijn we teruggevallen op de .Net 2.0 webservice welke communiceert doormiddel van SOAP. SOAP is gebaseerd op XML enveloppen. Deze interface optie is een reeds geaccepteerde standaard voor interfaces en kan bijna vanuit elke taal worden gebruikt om data uit te wisselen.

Een van de nadelen van SOAP is dat de interface gebaseerd is op XML. Dit betekent dat elke data die in gewone tekst zichtbaar is uitgewisseld kan worden. Dit betekent dan ook dat veel data eerst vertaald moet worden naar gewone tekst.

5. Rup Fases webservice (Send Data from Ifoods)

5.1 Inceptie fase

De inceptie fase is de eerste fase van het Rup proces. Tijdens deze fase wordt het (deel)project gedefinieerd en de wensen en eisen gedocumenteerd.

5.1.1 Opstellen van requirements

Dit deelproject staat in het teken van het ontsluiten van reeds bekende informatie uit de Ifoods database. Deze te ontsluiten informatie bestuit uit bijvoorbeeld:

- Aflevergebieden
- Openingtijden
- Product informatie
- Order informatie

Tijdens deze fase was er ook al gekeken naar de wensen en de eisen van de ikassa applicatie. Daarom was het voor een groot gedeelte ook al bekent wat de databehoefde van de applicatie was ten opzichte van Ifoods.

5.2 Elaboratie fase

De Elaboratie fase is de tweede fase van het Rup Proces. Binnen deze fase worden de schema's op papier gezet die tijdens de constructie fase worden gebruikt als ondersteuning.

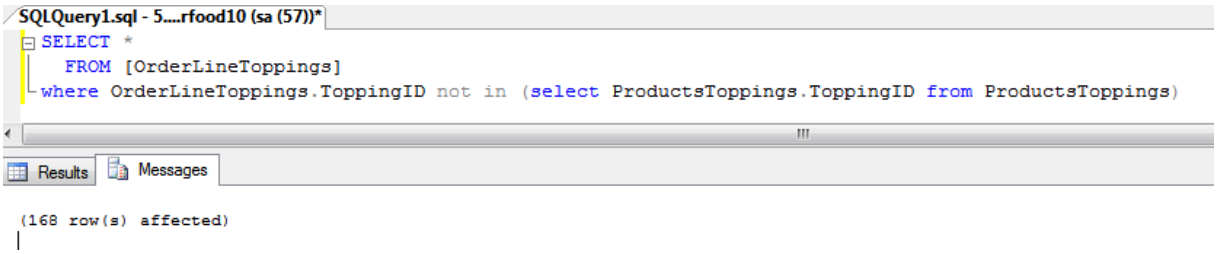
5.2.1 Klassen Diagram

Zoals bij het opstellen van de requirements al was vermeld, is het doel het ontsluiten van reeds bestaande informatie. Deze informatie is opgeslagen in de Ifoods database. De informatie wordt dagelijks gebruikt door verschillende bedrijfsprocessen binnen Ifoods.

De database is 7 jaar geleden ontworpen. Zoals gebruikelijk is er in de loop der tijd een hoop aan veranderd en bijgebouwd. Zo is op de website bijvoorbeeld de I-deal functionaliteit toegevoegd. Dit terwijl aan de achterkant een functionaliteit is bijgebouwd om bestel gegevens uit te wisselen met Sales Booster Enterprice (een ander kassasysteem).Helaas zijn een aantal van deze wijzingen nooit (goed) gedocumenteerd en is door de overdracht tussen verschillende ICT bedrijven ook de originele "blue print" van de database verloren gegaan.

Omdat sommige bedrijven de data integriteit niet belangrijk vonden, is de database op sommige punten ook niet goed ingericht. Zo ontbreekt bij sommige databasetafels bijvoorbeeld een primaire sleutel. Ook is op sommige relaties geen referentionele integriteit geïmplementeerd.

Het niet afdwingen van referentionele integriteit zorgt er voor dat het mogelijk wordt om "orphans"(wezen) te laten ontstaan. "Orphans" zijn regels uit een database waarvan de vreemde sleutel verwijst naar een primaire sleutel die niet meer bestaat.



Zoals bovenstaand voorbeeld laat zien zijn er op dit moment 168 bestelregels die een topping gebruiken die niet meer bestaat in de database. Het gevaar hiervan is dat software die gebruik maakt van deze informatie in het gunstigste geval een lege regel zal tonen maar in het ergste geval vast zou kunnen lopen.

Door referentionele integriteit te implementeren kan worden verplicht om de in cascade gekoppelde data te verwijderen voordat primaire sleutel verwijderd kan worden.

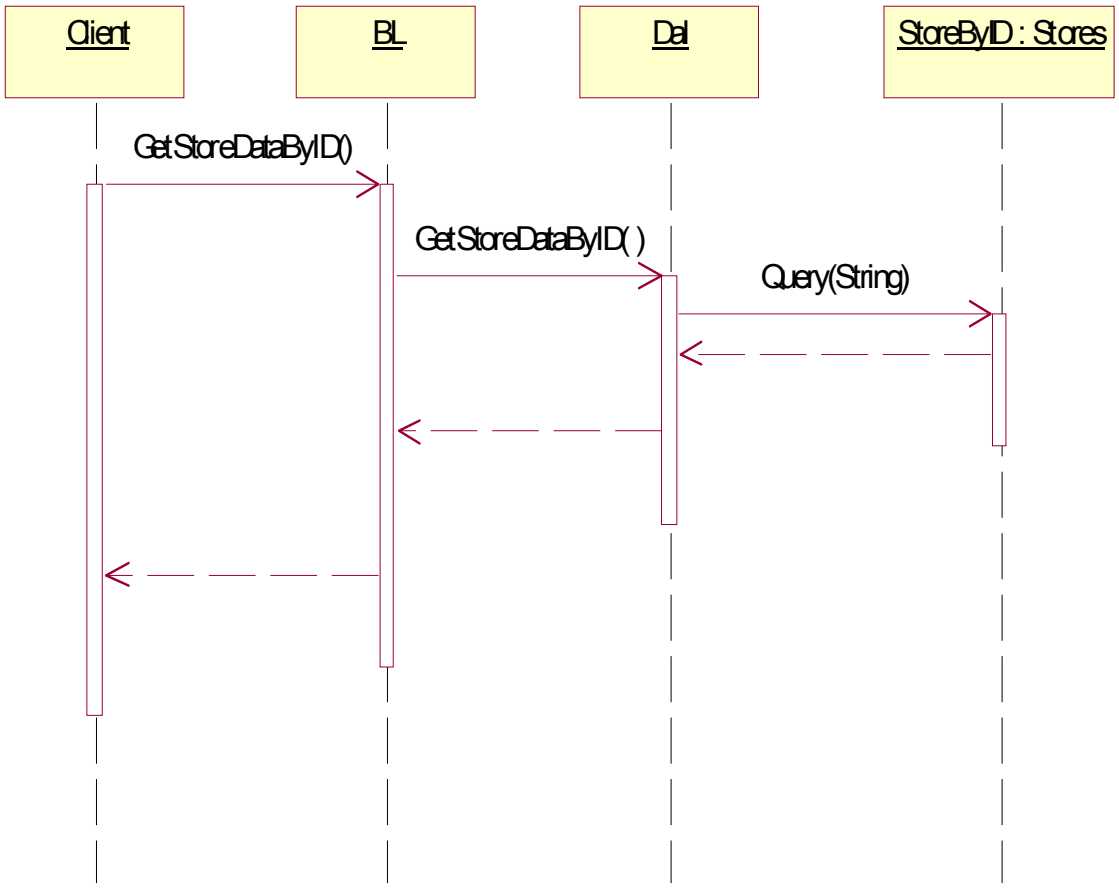
Bij het opstellen van het klasse diagram is in eerste instantie gebruik gemaakt van de ERD functie van SQL server. Omdat de database niet goed was ingericht was het vervolgens erg lastig om de missende relaties te definiëren. Uiteindelijk heb ik alle tafels nagelopen en zijn in mijn uiteindelijke klasse diagram alle relaties gedefinieerd. Ook heb ik waar mogelijk de database voorzien van “contraints” die de referentionele integriteit verder afdwingen. Voor de plekken waar dit niet mogelijk was heb ik voorstel gedaan aan de opdrachtgever deze data te schonen zodat op een later tijdstip de integriteit alsnog afgedwongen kan worden en de database 100% integer is.

5.2.2 Sequentie Diagrammen

Zoals eerder beschreven worden de applicaties ontwikkeld volgens het n-Tier model. In het geval van de webservice betekent dat binnenkomende aanroepen via de Business logic laag wordt doorgestuurd naar de Data Access Laag en resultaat komt in ongekeerde volgorde terug.

De reden om op dit moment deze ontwikkelmethode te volgen is de wetenschap dat op een later stadium de diverse lagen zullen worden uitgebreid om binnenkomende data te verwerken. Voor de validatie is dan ook een stuk Business logic nodig. Daarnaast is, zoals eerder al vermeld, het grote voordeel van deze lagen techniek dat je de lagen onderling kan vervangen waardoor het makkelijker wordt om bijvoorbeeld voor een andere database of ander communicatie platform te kiezen.

Het grootste voordeel van een lagenmodel is de hergebruikbaarheid en onderhoudsvriendelijkheid. Je hoeft code immers maar 1 maal te maken om het in meerdere applicaties te gebruiken. Daarnaast hoef je bij wijzigingen deze ook maar op 1 plek door te voeren.



Figuur 12 Voorbeeld van een typische dataaanvraag

Bovenstaande afbeelding laat een typische aanvraag van een cliënt zien. De cliënt wil in dit geval restaurantdata hebben van de webinterface. De aanvraag komt binnen in de business laag. Op dit moment zit er geen logica in deze laag maar op dit niveau zou er bijvoorbeeld een controle gedaan kunnen worden om te verifiëren of deze cliënt wel die data mag benaderen. Omdat er op dit moment geen logica aanwezig is, wordt de aanvraag doorgestuurd naar de DAL laag. In dit geval is er voor gekozen om de fuctienamen het zelfde te houden. De reden hiervoor is dat de functienaam een goede beschrijving geeft over wat zowel de input is als de verwachte output. Door het gebruik van “namespaces” en classnamen kan verwarring worden voorkomen

De datalaag vraag aan het gewenste object (in het geval hierboven) doormiddel van de aanroep Query informatie op uit de database. Deze data wordt geretourneerd aan de DAL en is van het type Stores (een klasse uit het klasse diagram). Deze geeft het object weer terug aan de BL die het op zijn beurt terug geeft aan de client.

5.3 Construction Fase

De Construction fase is de derde fase binnen RUP. In deze fase, krijgt het project een fysieke vorm. De diagrammen en schema's uit inceptie en elaboratie fase worden omgezet in producten. Deze producten kunnen bestaan uit databases en 1 of meerdere programma's

Tijdens deze fase wordt de ontwikkelde code ook getest door middel van "Unit Testing". Bij "Unit Testing" wordt er gekeken of (kleine) blokken code de goede resultaten geven. Dit betekend niet alleen de gewenste resultaten bij een goede invoer maar ook dat het codeblok niet vastloopt indien er foute invoer wordt aangeboden.

Code

Voor het ontwikkelen van de webinterface hebben we gebruik gemaakt van een sjabloon uit de ontwikkelomgeving Visual Studio. Dit sjabloon wordt standaard meegeleverd. Door deze keuze wordt er een raamwerk gecreëerd waardoor je binnen enkele minuten al een webservices online kan zetten. Daarnaast hebben we gebruik gemaakt van de optie om Visual Studio een Data Acces Layer te laten generen op basis van eerdere schema's en de database.

Omdat de webinterfaces alleen gegevens kan uitwisselen op basis van platte tekst, zijn de communicatie methodes erg beperkt. Daarom is gekozen om gegevens altijd terug te sturen in de vorm van een Dataset. Het .net Framework kan een Dataset eenvoudig omzetten naar deze vorm van platte tekst. Een DataSet is een verzameling van 1 of meerdere DataTables welke weer kunnen bestaan uit 1 of meerdere DataRows. Een DataRow is vaak een representatie van 1 regel uit een database. DataSet, DataTable en DataRow zijn standaard objecten van het .Net framework. Deze objecten worden voornamelijk gebruikt in de oudere .Net 2.0 omgeving voor het opslaan van database en andere gestructureerde data.

Om te zorgen dat er geen problemen kunnen optreden aan de kant van de webinterface, heb ik er voor gekozen om in de business laag altijd eerst een DataSet te initialiseren. Op deze manier kan zelfs wanneer de database een leeg antwoord terug geeft, er nog steeds een geldig object worden terug gestuurd naar de cliënt. Onderstaand voorbeeld is hier een uitwerking van in code.

```
public DataSet GetrestaurantBySerial(String SerialCode)
{
       DataSet dsResult = new DataSet(); ← initaliseren van een leeg object
       try
       {
              IkassaSerialRestaurantsTableAdapter RestaurantBySerial = new
       IkassaSerialRestaurantsTableAdapter();
              DataTable Result = RestaurantBySerial.GetDataBySerialnummer(SerialCode);
              Result.TableName = "RestaurantBySerial";
              dsResult.Tables.Add(Result); ← Toevoegen van informatie aan het object
              return dsResult;  ← Rerug geven het object
       }
       catch (Exception ex)
       {
              logging.LogMessageToFile("Probleem met GetrestaurantBySerial
+Code:"+SerialCode+"Message:"+ex.Message);
       }
       return new DataSet();  ← Terug geven van leeg object
}
```

Testen

Aangezien er straks verschillende kassa systemengebruik gaan maken van de webservice is het van belang dat deze altijd blijft werken. Daarom hebben wij de webservice onderworpen aan testen doormiddel van unit testing. Bij Unittesting wordt een functie van een programma uitgelicht en wordt getest wat deze functie als uitvoer geeft. Deze waarde wordt vergeleken met een verwachte waarde waarna de test gelukt of mislukt is.

Ondanks dat er een goede built-in tool in Visual Studio aanwezig is voor het testen doormiddel van unit testing, hebben wij er voor gekozen om deze tool niet te gebruiken. De reden hiervoor is dat er een hoop extra mogelijkheden zitten verwerkt in deze tool. Deze mogelijkheden kunnen het instellen van een goede unittest complex maken en bieden voor een klein project als dit te veel mogelijkheden. Daarnaast bied deze losse tool ons de mogelijkheid om zonder het gebruik visual studio de webservice interface te testen. Dit kan handig zijn in de toekomst voor het testen van problemen op locatie.

```
public static void TestGetAddons2ProductsByStoreID(Guid RestaurantID, int Verwacht)
        {
            try
            {
                DataSet Result = IfoodsWeb.GetAddons2ProductsByStoreID(RestaurantID);
                Console.WriteLine("GetAddons2ProductsByStoreID parameter: " + RestaurantID.ToString());
                Console.WriteLine("Aantal Tables : " + Result.Tables.Count);
                Console.WriteLine("Aantal Rows in Table 1 : " + Result.Tables[0].Rows.Count);
                Console.WriteLine("Waarde : " + Result.Tables[0].Rows.Count + " Verwacht " + Verwacht);
                if (Result.Tables[0].Rows.Count == Verwacht)
                    Console.WriteLine("Test ok");
                else
                    Console.WriteLine("Test ERROR");
                Console.WriteLine("");
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
```

Resultaat

```
GetAddons2ProductsByStoreID parameter: 3b1f504c-7ffd-4535-9fa4-22de99be849f
Aantal Tables : 1
Aantal Rows in Table 1 : 26083
Waarde : 26083 Verwacht 26083
Test ok
```

Voorbeeld van testcode en uitvoer

Uit eerder gebruik van de unit test tool van Visual Studio hebben wij het concept overgenomen en een nieuwe programma geschreven. Zoals in bovenstaand voorbeeld is te zien gaat om een simpel programma. Bovenstaande code is herhaald voor elke mogelijke aanroep die op dit moment aanwezig is in de webservice.

De stappen die het programma doorloopt voor elke aanroep zijn:

- Ophalen informatie
- Kijken of er überhaupt resultaten geretourneerd zijn
- Tonen van aantal resultaten
- Testen of dit in de lijn van verwachting ligt
- Tonen of de test wel of niet een succes is

Bij het testen van de aanroepen worden de volgende waardes gebruikt:

- Een lege waarde om te simuleren wat er gebeurt als een paramater leeg wordt meegezonden
- een waarde die wel valide is maar niet voorkomt in de database
- een geldige waarde waar een of meerdere resultaten voor beschikbaar zijn

Omdat voor de invoer van de webinterface bewust is gekozen voor Strong Typed basis objecten, is het niet nodig om te controleren wat er gebeurt wanneer de verkeerde invoer data wordt aangeleverd. Een programma wat probeert eerder genoemde functie aan te spreken met bijvoorbeeld een integer als parameter zou simpelweg niet gecompileerd kunnen worden. Software die geen gebruik maakt van een intelligente IDE of bijvoorbeeld pas bij runtime wordt gecompileerd zouden in theorie verkeerde informatie kunnen sturen. Dit wordt echter afgevangen door het .Net framework nog voordat de webservice wordt aangesproken.

Het testrapport is als onderdeel van de source code opgenomen alsmede het programma waarmee de test is uitgevoerd. Daarbij is afgesproken dat bij wijzingen aan de webservice de test zal worden uitgebreid en voor het live zetten altijd eerst getest zal worden.

6.0 Rup Fases Windows Service (recieve data in kassa)

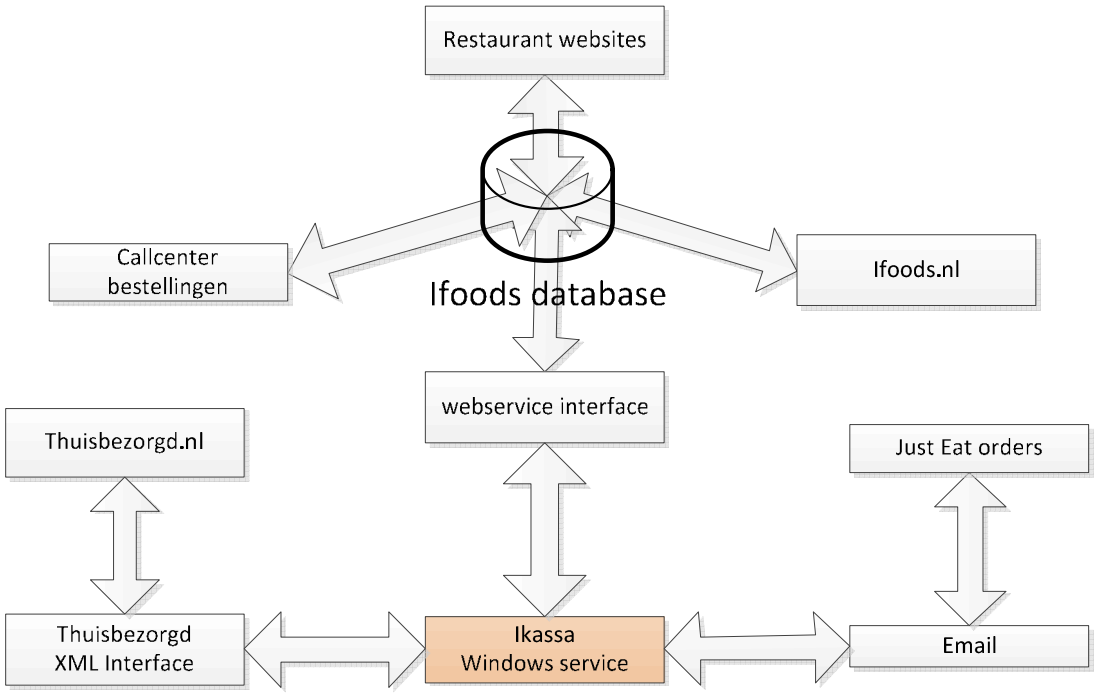
6.1 Inceptie fase

6.1.1 Opstellen van requirements

Het doel van deze iteratie is het ontwikkelen van een applicatie die het mogelijk maakt informatie te verkrijgen van verschillende bronnen en deze te verwerken in de database van het lokale systeem.

De bronnen waarmee de Windows services gaat communiceren zijn bijvoorbeeld de web interface van Ifoods, de XML brondata van Just Eat, en de email orders van thuisbezorgd.

Verder is het de bedoeling dat de applicatie geen user interface heeft en verder dan ook niet door de gebruiker benaderd kan worden.



Figuur 13 Interfacing naar diverse diensten en op verschillende methode

6.1.2 Orders ontvangen van externe partijen

Een van de belangrijkste functies van het kassa systeem is de mogelijkheid om orders te kunnen ontvangen vanaf andere bronnen.

Ifoods

Om te communiceren met Ifoods wordt gebruik gemaakt van de eerder ontwikkelde webservice. Deze interface bied de mogelijkheid om nieuwe bestellingen rechtstreeks op te halen uit de database van Ifoods. Omdat de kassa-applicatie wordt gevoed met brondata van Ifoods kan de order 1 op 1 worden overgenomen. Ook bied de interface de optie om een terugkoppeling te geven aan Ifoods. Op deze manier kan Ifoods haar klanten per email of SMS op de hoogte houden.

Deze extra optie biedt Ifoods ook de mogelijkheid om bij het uitblijven van statusverandering in te grijpen en bijvoorbeeld telefonische contact te zoeken met het restaurant. Door deze extra controle moet het aantal te late of niet geleverde bestellingen verder terugdringen.

Thuisbezorgd

Thuisbezorgd, een van de externe partijen waarvan wij de orders willen importeren, bied geen mogelijkheid om rechtstreeks te communiceren. Wel biedt Thuisbezorgd een mogelijkheid waarbij een applicatie van de firma als tussenschakel werkt. Deze software verzorgt de communicatie met de servers van Thuisbezorgd en stelt de orders beschikbaar in een lokaal opgeslagen XML bestand. Door in het juiste formaat een nieuw bestand aan te bieden aan de Thuisbezorg applicatie, kan ook de orde status terug gegeven worden. Op deze manier kan Thuisbezorgd op hun beurt de klanten verder inlichten.

Just Eat

Just Eat is een andere externe party waarvan de kassa-applicatie orders moet binnenlezen. Helaas biedt Just Eat geen enkele mogelijkheid om een interface te gebruiken. Restauranthouders hebben bij Just Eat 2 opties om hun orders te ontvangen. Zij kunnen de orders bekijken via een online klanten paneel of ze kunnen een kopie order ontvangen in een door hun opgegeven mail adres. Voor de verwerking hebben we een email parser ontwikkeld die de email vertaald naar bruikbare informatie.

6.2 Construction Fase

6.2.1 Het maken van de DAL

Voor het ontwikkelen van de DAL hebben wij gebruik gemaakt van een zo gehete mini-ORM(Object Relational Mapper). De kracht van deze tool, ten opzichten van grotere ORM's zoals NHibernate en het Entity Framework, is dat het meer focust op snelheid dan op extra geavanceerde functionaliteit. Hierdoor is de leercurve klein en het gebruikersgemak groot.

De ORM waar wij voor hebben gekozen is PetaPoco. Deze ORM kan sinds Visual Studio 2010 via de package manager worden toegevoegd aan een project. Na een paar instellingen zoals database naam en plaats en eventuele inloggegevens gaat de tool aan het werk.

Wat de tool doet is een vertaling maken van database tabelen naar POCO's (Plain Old CLR Objects). Een POCO is een klasse bestaand uit standaard objecten behoorde tot het standaard repertoire van de .Net omgeving. Omdat data onderling een relatie kunnen bevatten, kan een POCO ook andere eerdere POCO's bevatten.

Naast het genereren van .Net klasse bied de ORM ook een framework voor het bevragen van de database. Standaard operaties zoals Select,Insert, Delete en Update zijn standaard beschikbaar. Daarnaast maakt dit framework het ook mogelijk om speciale queries uit te voeren. Deze queries geven dan een verzameling van 0 of meer terug.

In sommige gevallen wil je informatie uit de database halen die versnipperd staat over meerdere tafels. Ondanks dat dit niet vaak voorkomt, bied PetaPoco hier wel ondersteuning voor. Dat maakt het mogelijk om deze ORM ook voor complexer queries in te zetten.

6.2.2 Interface methode

Zoals eerder beschreven zijn er een aantal verschillende partijen met elk hun eigen manier van communiceren naar een restaurant. Omdat elke partij op een andere manier communiceert is er voor elke partij ook een andere module ontwikkeld.

De makkelijkste implementatie was die van Ifoods. Omdat beide kanten van de applicatie worden ontwikkeld door de zelfde firma, is het makkelijk om interface en cliënt op elkaar af te stemmen. Deze makkelijk overeenstemming is te danken aan het feit dat de bron tot op de puntjes bekent is en waarnodig aangepast kan worden. Daarnaast gaat interne communicatie doorgaans makkelijker dan communicatie met externe bedrijven. Bij deze implementatie heb ik dan ook geen noemenswaardige problemen ondervonden.

De tweede implementatie was de implementatie van Thuisbezorgd. Thuisbezorgd heeft zijn interface zelf niet beschreven maar voor de verschillende XML bestanden wel een DTD schema opgesteld. Daarnaast biedt Thuisbezorgd in zijn XML een vrij veld om te koppelen. Omdat Thuisbezorgd onder andere ook samenwerkt met Sales Booster Enterprise (een ander kassasysteem) is er al een interface beschikbaar. Deze interface is niet rechtstreeks maar op basis van File IO en een aparte applicatie.

De laatste implementatie, die van Just Eat, is een koppeling op basis van email verkeer. Just Eat heeft maar 2 mogelijk communicatiemethodes naar een restaurant, namelijk per mail en per fax. Aangezien de orders automatische verwerkt dienen te worden, valt de optie voor fax al af. Ondanks dat het technische haalbaar zou zijn om een fax op een computer te ontvangen en eventueel doormiddel van OCR vertaalt zou kunnen worden is dit zo foutgevoelig dat dit op voorhand al geen optie was. Helaas bied verkeer per email ook een uitdaging. E-mailcontent hebben geen vaste structuur of een vaste opmaak. Daarom is het vrij lastig om data uit de email te extraheren.

Ifoods implementatie

Zoals al beschreven, was de integratie van Ifoods een simpele opdracht. Aan de kant van Ifoods is door mijzelf een webservice interface geschreven. Naast het beschikbaar stellen van product informatie wordt via de zelfde webservice interface ook de bestellingen die via kanalen van Ifoods binnen komen verwerkt en aangeboden.

Thuisbezorgd implementatie

Thuisbezorgd heeft er voor gekozen een applicatie aan te bieden die als service draait op de achtergrond. Deze applicatie maakt een aantal mappen aan met daarin XML bestanden. Thuisbezorgd heeft in zijn documentatie een aantal DTD beschikbaar gesteld. Een DTD is een manier van beschrijven hoe een XML bestand is opgebouwd en welke componenten verplicht en / of optioneel zijn. Daarnaast worden ook de structuur en daarmee de onderlinge relaties van de data gedefinieerd.

Om een DTD te gebruiken binnen Visual Studio of C# moet deze eerst worden omgezet naar een andere structuur. Binnen C# worden XSD bestanden gebruikt voor het definiëren van gestructureerde data. Dit kan een XML bestand zijn maar bijvoorbeeld ook een Dataset.

Omdat zowel XSD als DTD een open standaard zijn en het zelfde doel hebben, bestaan er diverse manieren en hulpmiddelen om deze bestanden onderling te vertalen naar het andere formaat. Een van deze hulpmiddelen is bijvoorbeeld de tool die standaard mee wordt geleverd bij Visual Studio

genaamd XSD. XSD is een commanline tool die het mogelijk maakt om DTD betanden om te zetten naar het formaat wat bruikbaar is voor verdere verwerking. Omdat XSD naast het transformeren van DTD naar XSD ook nog een groot aantal andere opties kent, is het geen gebruiksvriendelijke tool om mee te werken. Daarom heb ik gekozen voor de gratis versie van XMLPad. Dit is een grafische tool die utermate geschikt is voor het werken met XML data en de bijbehorende schema technieken / bestanden. Naast het visueel weergeven van hoe de structuur in elkaar zit kan deze tool de schema's van het ene naar het andere formaat vertalen. Daarnaast kan de tool ook aan de hand van de schema's voorbeeld data genereren die weer gebruikt kan worden voor het testen van de software implementatie.

Op het moment dat er een XSD is gemaakt in Visual Studio is het mogelijk om met de eerder genoemde tool XSD een vertaling te laten maken naar POCO's (Plain Old CLR Objects).

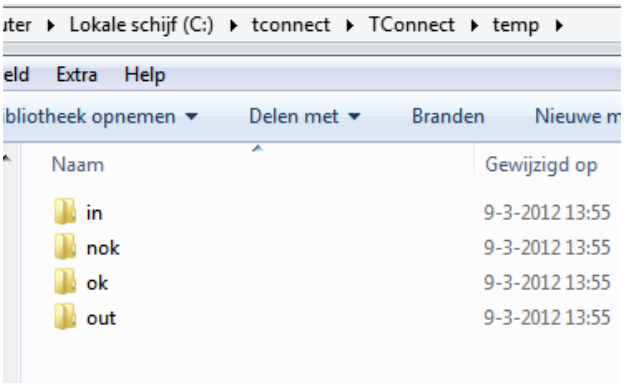
Zoals eerder al beschreven is XSD geen makkelijke tool om mee te werken. Daarom hebben we voor het genereren van code gebruik gemaakt van een zogeheten Addin voor Visual Studie genaamd XSD2Code. Deze tool werkt op basis van een eigen template en bied een groot scala aan opties. Een van deze opties is zoals de naam al doet vermoede het genereren van de POCO's. Ook bied het de opties om een aantal standaard functies toe te voegen die het inlezen en wegschrijven van de data naar bestanden mogelijk maakt. Hierdoor hoeft de ontwikkelaar niet zelf de data te laten deserializen (Deserializen is de term voor het vertalen van een platte of binair opgemaakte gestructureerde bron naar 1 of meerdere objecten)

Zoals al eerder genoemd kan Thuisbezorgd standaard orders aanleveren aan het software pakket Sales Booster Enterprise. Voor deze koppeling hebben zij in het XML bestand een extra veld opgenomen. Omdat Ifoods deze zelfde functionaliteit bied zijn daar deze codes ook bekent voor de aangesloten restaurants. Doordat de product informatie in veel gevallen door de service van Ifoods zal worden aangeleverd zijn de Sales Booster Codes ook bekend. Daarom is het redelijk makkelijk om op basis van dit unieke gegeven een match te maken tussen binnen gekomen data en relevante data.

Helaas stuurt thuisbezorgd bepaalde informatie niet mee. In de bestelde producten, ontvangen we alleen het unieke nummer het aantal en of er eventueel bijproducten bij horen. Daarom is het niet mogelijk voor ons om te bepalen of de prijs die op de website staat vermeld overeen komt met de prijs die op dat moment wordt gehanteerd in de kassa applicatie. Hiervoor hebben wij een tijdelijk oplossing gevonden om in geval van prijs verschillen in ieder geval de restauranthouder op de hoogte te stellen. Thuisbezorgd stuurt wel het verwachte totaal bedrag mee in zijn XML bestelling. Door aan het eind van de vertaling dit te vergelijken met de uitkomst die wij zelf hebben berekend, kan worden gekeken of er een afwijking is. Indien dit het geval is krijgt de kassa medewerker een melding te zien bij het accepteren van de melding.

Nadat de order is geaccepteerd door de kassa medewerker wordt er een XML bestand geplaatst in een bepaalde directory die door de Thuisbezorgd tool wordt opgepakt zodat zij weten dat de order is verwerkt en geaccepteerd. Als een order op een of andere manier niet verwerkt kan worden, wordt er een bestand geplaatst met als titel het order nummer in een speciale directory. Deze wordt ook opgepakt door de tool waardoor er een alarm wordt getrickerd bij Thuisbezorgd. Een van hun medewerkers zal dan telefonische contact zoeken om allereerst de bestelling per telefoon door te geven maar ook te kijken waar eventueel het probleem zit zodat de volgende order weer

automatische verwerkt kan worden. Er bestaat ook nog een optie waarbij een restaurant een order kan weigeren. Dit kan bijvoorbeeld het geval zijn als iemand te laat heeft besteld of net buiten het aflevergebied woont van het bezorgende restaurant. Dit laatste kan bijvoorbeeld voorkomen in plaatsen met hele lange straten met weinig nummers. Vaak hebben deze straten (zoals dijken of straten langs het water) een gezamenlijke postcode. Omdat de restaurantselectie op basis van postcode gebeurt kan het dus voorkomen dat het restaurant wel naar voren komt maar helaas niet op het betreffende huisnummer levert. In dit geval kan de applicatie ook een melding terug sturen naar Thuisbezorgd door een bestand te plaatsen in een speciale directory met daarin een XML inhoud waarnin de reden wordt toegelicht. Thuisbezorgd kan dan op zijn beurt weer de klant op de hoogte brengen van eventuele problemen.



Figuur 14 Directory structuur van Tconnect (App van Thuisbezorgd)

Just Eat implementatie

Zoals eerder beschreven is de implementatie van Just Eat het minst makkelijk. Just eat communiceert enkel op basis van tekst (fax of Email) en bied verder geen optie om de bestelling in een gestructureerd formaat te ontvangen. Zoals al eerder beschreven was de optie van het ontvangen van faxen al geen optie. Behalve de kosten die dit met zich mee zou brengen voor zowel Ifoods (aanschaf speciale componenten en extra ontwikkeltijd) als voor de klant (kosten voor telefoonlijn ed) is de foutgevoeligheid ook veel te groot.

Daardoor bleef als enige optie het uitlezen van de email en proberen de data hieruit te extraheren. Bij aanvang van het project en het horen van deze randvoorwaarde heb ik de opdrachtgever heel duidelijk gemaakt dat dit een optie is die uitgevoerd kan worden (Ifoods heeft dit al eerder gedaan binnen de eigen website ter behoeve van het kopiëren van menukaarten van de concurrenten) maar dat er ook een groot aantal nadelen en gevaren aan zitten. Het grootste gevaar dat bestaat is dat er een wijziging plaats vind in de opmaak van de email. Hierdoor zou het parsen van de mail in gevaar kunnen komen en de uitkomst er van onbruikbaar kunnen worden.

Op basis van mijn waarschuwingen is de ICT manager in gesprek getreden met de eigenaar en ontwikkelafdeling van Just Eat. Helaas bleek daaruit dat een mogelijkheid tot het aanleveren van gestructureerde data op dit moment niet ontwikkeld kon worden. Wel heeft Just Eat aangegeven de email orders niet aan te gaan passen op dit moment en indien ze die wel doen, ons ruim van te voren op de hoogte te stellen van de wijzigingen zodat wij onze software aan kunnen passen.

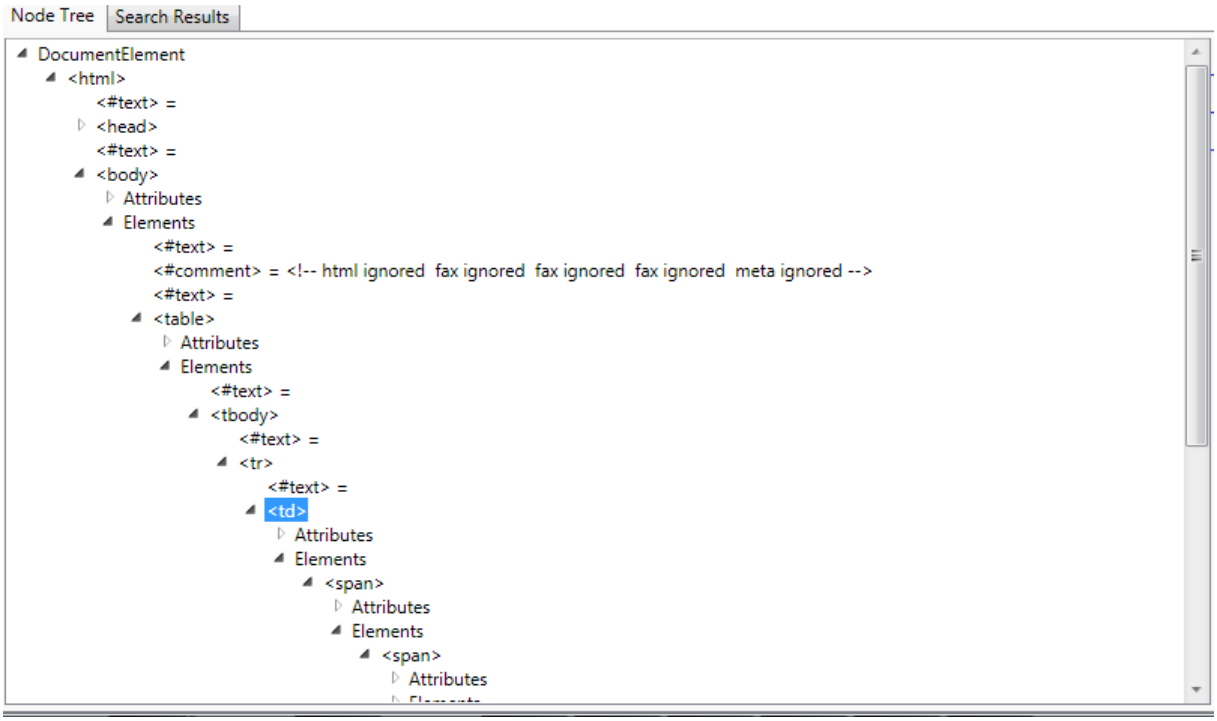
Wat betreft het technische gedeelte van de implementatie liep ik al eerder vast. Ondanks dat het .Net framework bij de laatste versies een groot aantal netwerkprotocollen is gaan ondersteunen is er nog steeds geen native support voor het POP3 of IMAP protocol. Omdat POP3 een redelijk makkelijk protocol is (plain text met ongeveer 20 commando's) zou het mogelijk zijn om op basis van de standaard TCP/IP implementatie van het .Net framework een eigen POP3 component te ontwikkelen. Dit zou echter een extra aanslag zijn op de beschikbare tijd en ook de nodige gevaren met zich meebrengen in verband met afwijkende email servers die net niet 100% het protocol volgen. Er zijn (met name in het verleden) een aantal mailservers geweest die net hun eigen implementatie hadden van diverse protocollen zodat deze beter aansloten bij de eigen cliënt software.

Omdat het risico te groot is en te veel tijd zou vergen, hebben we besloten om te zoeken naar een bestaand component. Bij het testen en implementeren van een paar componenten kwam het OpenPOP.net component als beste uit de test. Naast een goede implementatie van het protocol is het opensource. Mocht er in de toekomst een uitbreiding bij moeten of zou er toch een fout worden gevonden kan dit door ons zelf worden opgelost. Dit is natuurlijk enkel het geval als het project een stille dood zou sterven. Op dit moment is er een redelijk userbase die de software gebruikt en wordt er op het forum ook actief ondersteuning geboden. Van deze ondersteuning hebben wij geen gebruik hoeven maken aangezien er zeer goede documentatie en voorbeelden bij worden geleverd.

Een ander groot voordeel van dit component is de vertaling van de email terug naar een leesbaar formaat. De meeste mail (zeker als er plaatjes en/of meerdere versies van een bericht in de email zitten) worden verzonden in een speciaal formaat. Bijlagen worden bijvoorbeeld vertaald naar Base64 Encoding en plaatjes (direct in de mail) en tekst worden vertaald naar een Multi Part Mime formaat.

Door het terug parsen van de email naar het de oorspronkelijke email kan de email gebruikt worden. Ook bied het component opties als het alleen terug geven van bijvoorbeeld de HTML of plain tekst variant van een email (dit is natuurlijk alleen mogelijk als er ook daadwerkelijk een plain tekst versie mee wordt gestuurd).

Omdat wij niet het eerste bedrijf zijn dat informatie wil extraheren uit een HTML bron bestaan er redelijk wat betaalde en gratis componenten die dat mogelijk maken. NA enkele testen zijn wij blijven steken bij HAP (HTML Agility Pack). Dit is een HTML Parser die een gegeven invoer terug vertaald naar een zogeheten DOM object. Het mooie van dit component is dat het redelijk tolerant is. ER bestaat redelijk wat HTML bestanden en email die niet 100% conform de W3 Standaard worden opgemaakt. Hierdoor zou bij het vergeten van bijvoorbeeld een < de email onbruikbaar worden bij het parsen naar een DOM model. Een veel vaker voorkomend probleem is de misbruik van elementen binnen andere elementen. Een voorbeeld daarvan is bijvoorbeeld "<P><DIV></DIV>". Dit is een veel voorkomende fout die volgens de W3 standaard niet mag maar die webdesigners veel / programmeur regelmatig misbruiken voor diverse doeleinde.



Figuur 15 Voobeeld van DOM structuur van Just Eat Email

Een ander groot voordeel van dit component is de implementatie van Xpath. XPath is een techniek die eigenlijk bedoelt is voor het verwerken van XML bestanden. Omdat de intentie van XML en HTML het beschrijven van informatie is op een gestructureerde manier is XPath ook te gebruiken op HTML.

Door het gebruik van XPath wordt het een stuk makkelijker om bepaalde brokken informatie te ontrekken aan de HTML bron. Een voorbeeld hiervan is bijvoorbeeld de bevestigingslink die in de email zit verwerkt. Deze link dient de ontvangende restauranthouder normaal aan te klikken zodat Just Eat weet dat de order in goede orde is ontvangen. Deze link zit in de email verstopt in de eerste colom van de eerste regel van de eerste tabel. Zoals gebruikelijk zit de link verstopt in het attribuut href van het <A> object. Om de link te extraheren kan ik de volgende methode en XPath gebruiken:

```
String HTMLString = HTML.GetBodyAsText();
test.LoadHtml(HTMLString);

HtmlAttribute Bevestiglink =
test.DocumentNode.SelectSingleNode("/html[1]/body[1]/table[1]/tbody[1]/tr[1]/td[1]/spa
n[1]/span[1]/a[1]").Attributes["href"];
```

Een ander voorbeeld is bijvoorbeeld het order nummer. Dit zit in de zelfde tabel op de derde regel verwerkt in een <Strong> object. Om dit nummer te extraheren kan de volgende code worden gebruikt:

```
HtmlNode OrderNummerSpan =
test.DocumentNode.SelectSingleNode("/html[1]/body[1]/table[1]/tbody[1]/tr[3]/td[1]/str
ong[1]");
```

Zoals in bovenstaande voorbeelden zichtbaar is, maakt het gebruik van XPath het eenvoudig om door de DOM heen te navigeren. Dit is natuurlijk enkel zo wanneer de structuur eenmaal is onderzocht.

De vertaling van product informatie gebeurt op basis van een tekstuele vergelijking op productnaam. Omdat er verder geen referentie punten zijn waaraan wij de bestelinformatie kunnen koppelen is dit de enige oplossing. Tijdens het testen van deze koppeling zijn we tot de conclusie gekomen dat zolang de data consistent over wordt genomen door Just Eat, dit systeem prima werkt. Momenteel zijn is er nog overleg om te kijken of het proces verbeterd kan worden zodat type fouten en dergelijke in productnamen voorkomen kunnen worden.

Just Eat stuurt in zijn bestel email wel prijzen mee. Door een vergelijking van de kassa prijs en de prijs van de bestelling te vergelijken kan de restauranthouder snel geïnformeerd worden bij foutieve prijzen zodat de restauranthouder hier actie op kan ondernemen.

7.0 Rup Fases Kassa Systeem

7.1 Inceptie fase

7.1.1 Opstellen van requirements

Het belangrijkste onderdeel van dit project is om een duidelijk interface te ontwikkelen voor het tonen van bestellingen. Daarnaast moet de interface te mogelijkheid bieden om een nieuwe order in te voeren.

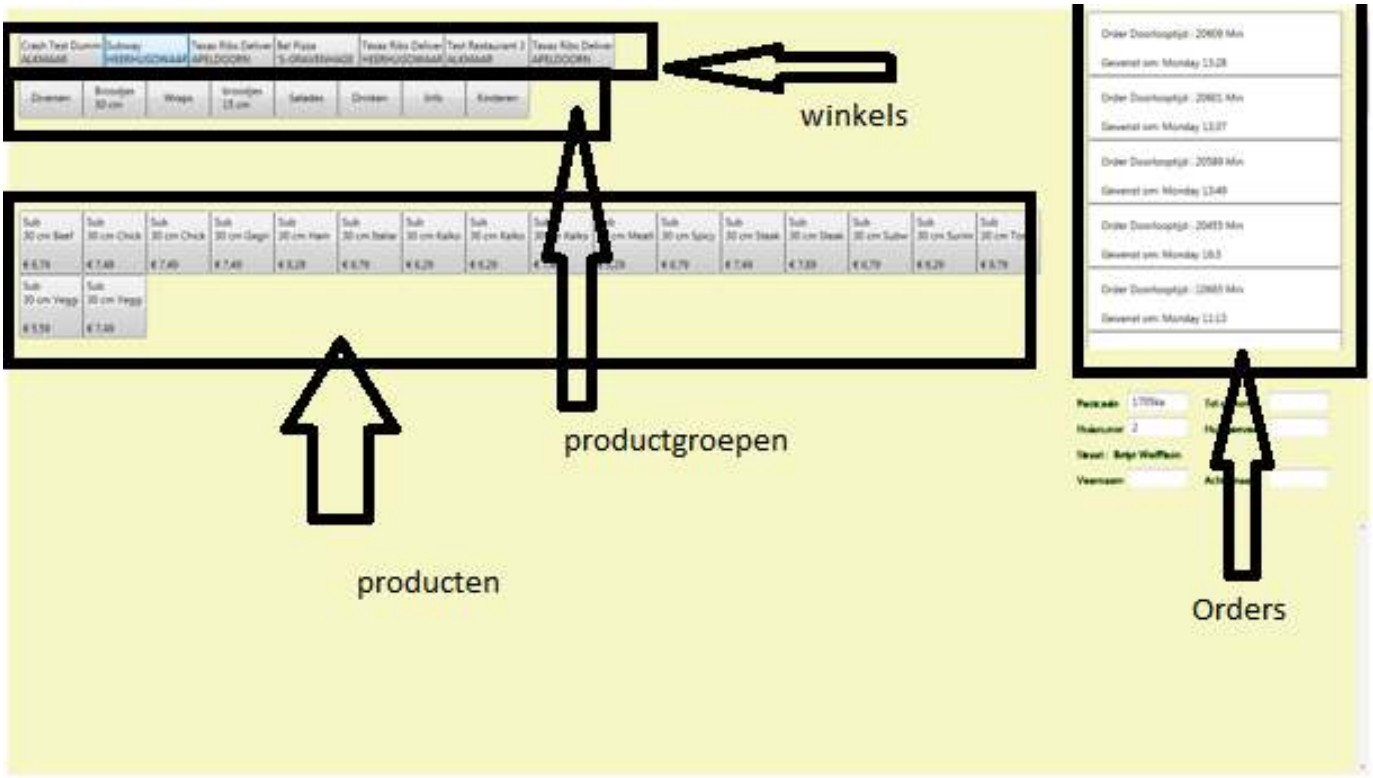
7.1.2 Proof of Concept

Aan het begin van de kassa-applicatie was geen duidelijk beeld hoe de interface van het kassa systeem er uit moest komen te zien. Om te voorkomen dat aan het eind een product zou ontstaan waar niet mee gewerkt kan worden, heb ik in overleg met de opdrachtgever afgesproken om een “proof of concept” te maken.

Het idee van een “proof of concept” is om met zo min mogelijk tijd en data een “live” situatie te creëren waarbij de belangrijkste functies een plek hebben gekregen. Daarbij hoeft niet elke functie te werken, maar gaat het meer om een globaal idee te ontwikkelen. Met dit idee als uitgangspunt, kan uiteindelijk dan 1 of meerdere applicaties worden ontwikkeld.

Tijdens de “proof of concept” is er ook gekeken naar onder andere de resolutie en hardware waarop de applicatie moet komen te draaien. Dit was een belangrijke keuze omdat schermruimte erg beperkt is en deze optimaal gebruikt dient te worden.

Verder bleek na de eerste versie van het “proof of concept” dat standaard schermcomponenten zoals we die kennen van andere applicaties niet geschikt zijn voor het gebruik bij een touchscreen. Windows heeft standaard een opties in huis die het mogelijk maken om classic applicaties te bedienen via een touchscreen interface. Desondanks komt het de snelheid en gebruiksvriendelijkheid niet ten goede. Daarom is het beter om van de classic desktop lay-out af te stappen en de interface aan te passen voor het gebruik van een touchscreen.

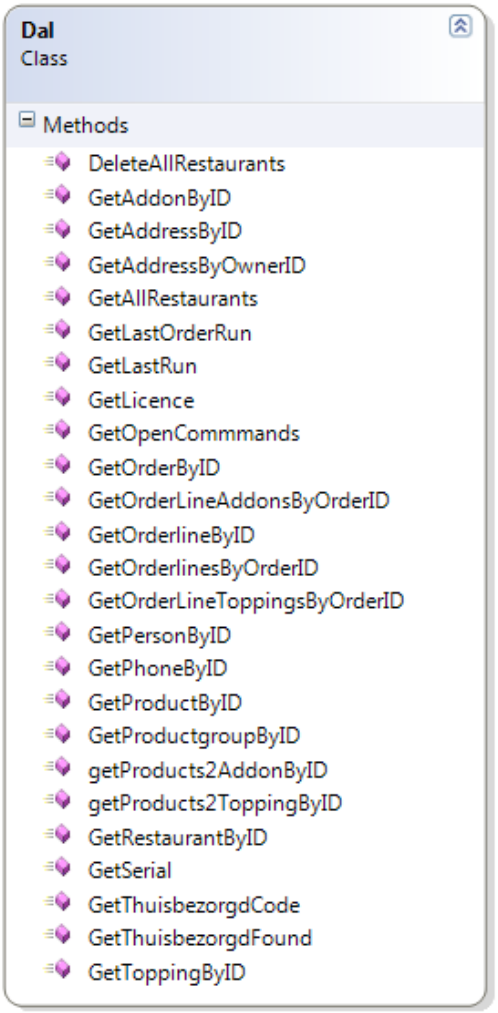


Figuur 16 Scherm opbouw eerste concept

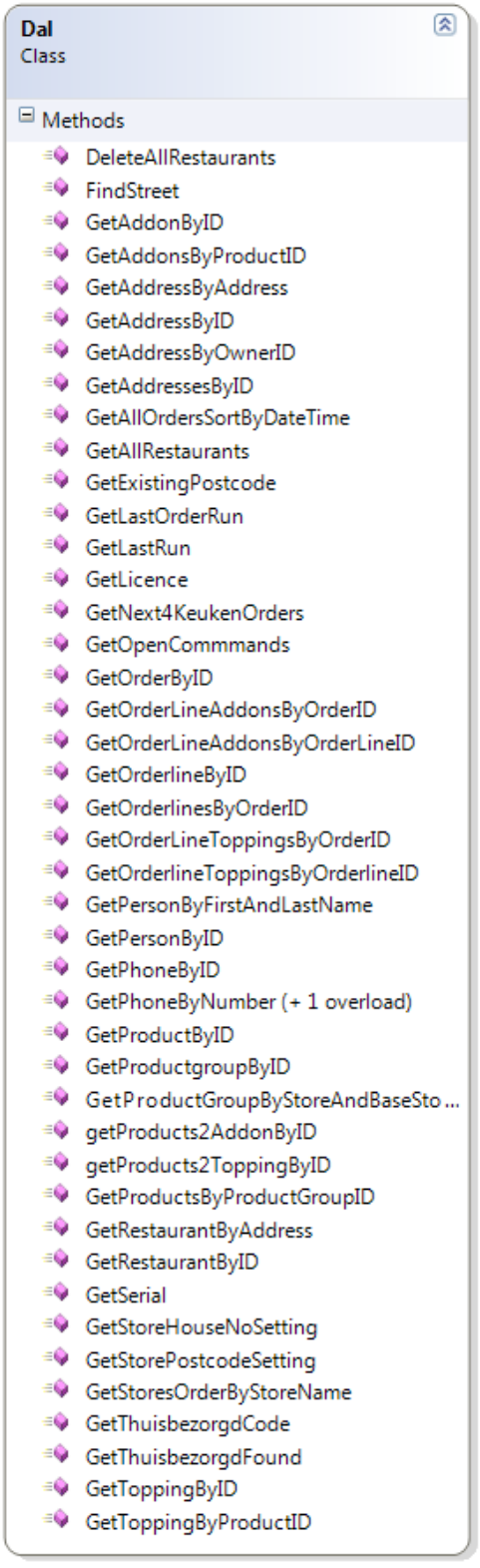
Aan het einde van de “Proof of Concept” is er een duidelijk beeld ontstaan voor zowel mij als de opdrachtgever hoe de interface er uit zou moeten komen te zien en welke functies op welke plaats dienen.

7.2 Elaboratie fase

Zoals eerder beschreven wordt er voor de ontwikkeling gebruik gemaakt van het N-Tier model waarbij data handdelingen, logica en schermweergave worden gescheiden. Omdat het klasse diagram van de Windows service en de kassa applicatie het zelfde is, is de Data Acces Laag (DAL) van beide projecten de zelfde. Ondanks dat een aantal functies gelijk zijn voor zowel de Windows service als de kassa applicatie, gebruikt de kassa applicatie een aantal functies die niet worden gebruikt in de Windows services. De figuren op de volgende pagina geven hier een helder overzicht van. Dit is tweemaal de schematische weergave van de klasse DAL. De kleinere versie met minder functies was de eerste versie van de DAL waarbij enkel de Windows service gebruik maakte van de DAL. De tweede versie is de uiteindelijke versie zoals die gemaakt is voor de Kassa applicatie.



Figuur 18 Eerste klasse Diagram DAL



Figuur 17 Definitieve versie dal

7.3 Construction Fase

7.3.1 Windows Presentation Foundation (WPF)

Zoals eerder vermeld werd tijdens ontwikkelen van het “proof of concept” al heel snel duidelijk dat de standaard controls binnen Windows Forms niet echt geschikt waren voor het inrichten van een kassa systeem. Ondanks dat het mogelijk is om de standaard controls aan te passen naar bijvoorbeeld knoppen die makkelijk zijn aan te raken op een touchscreen wilde ik eerst verder kijken naar andere mogelijkheden.

Na wat onderzoek kwam ik uit op WPF. Dit is een nieuw framework binnen Microsoft .Net versie 4.0. wat tijdens de installatie van .Net standaard wordt meegeleverd. WPF combineert een aantal componenten van de traditionele Windows Forms en Silverlight. Silverlight is een grafische omgeving in het leven geroepen als een tegenhanger van onder andere Flash. Het is een voornamelijk grafische georiënteerde framework om Rich Internet Applications te kunnen maken.

Waar bij windows Forms een groot gedeelte van de lay-out vanuit code word bepaald, is binnen WPF de opmaak voor het grootste gedeelte gescheiden van de code. Binnen WPF wordt gebruikt gemaakt van XAML (Extensible Application Markup Language). XAML is de taal die wordt gebruikt voor het definiëren van de gebruikersinterface, om elementen, gebeurtenissen en andere onderdelen daarvan te definiëren.

Een van de voordelen van het scheiden van code en lay-out is naast een beter overzicht ook een mogelijkheid om designers toegang te geven tot het project zonder dat ze achterliggende code hoeven te kennen.

WPF staat het toe om op verschillende niveaus stijlen en templates te definiëren, Hierdoor kun je bijvoorbeeld heel makkelijk je invoervelden voor je hele applicatie een bepaalde stijl te geven.

Door onderstaande stijl te definiëren, genaamd DoorZichtigeButton, op applicatie niveau kan ik op alle plekken waar ik een doorzichtige button wil hebben dit bereiken door 1 atribuut extra op te geven.

```
        <Style x:Key= "DoorZichtigeButton" TargetType="{x:Type Button}">
            <Setter Property="Background" Value="Transparent"/>
            <Setter Property="BorderThickness" Value="1"/>
            <Setter Property="Foreground" Value="White"/>
            <Setter Property="Padding" Value="1"/>
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="{x:Type Button}">
                        <Grid x:Name="Chrome" Background="{TemplateBinding
Background}" SnapsToDevicePixels="true">
                            <ContentPresenter HorizontalAlignment="{TemplateBinding
HorizontalContentAlignment}" Margin="{TemplateBinding Padding}"
RecognizesAccessKey="True" SnapsToDevicePixels="{TemplateBinding SnapsToDevicePixels}"
VerticalAlignment="{TemplateBinding VerticalContentAlignment}"/>
                        </Grid>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
        </Style>
```

Om deze stijl te kunnen gebruiken binnen een knop kan ik de volgende code gebruiken:

```
        <Button Name="btKassa" Grid.Row="1" Grid.Column="1" Style="{DynamicResource
DoorZichtigeButton}" Focusable="False" Click="btKassa_Click">
            <Image Source="/Iconnect;component/Images/btnKassa.png"
IsHitTestVisible="False"></Image>
        </Button>
```

Figuur 19 Figuur zonder stijlopmaak

Figuur 20 Figuur met doorzichtige stijlopmaak

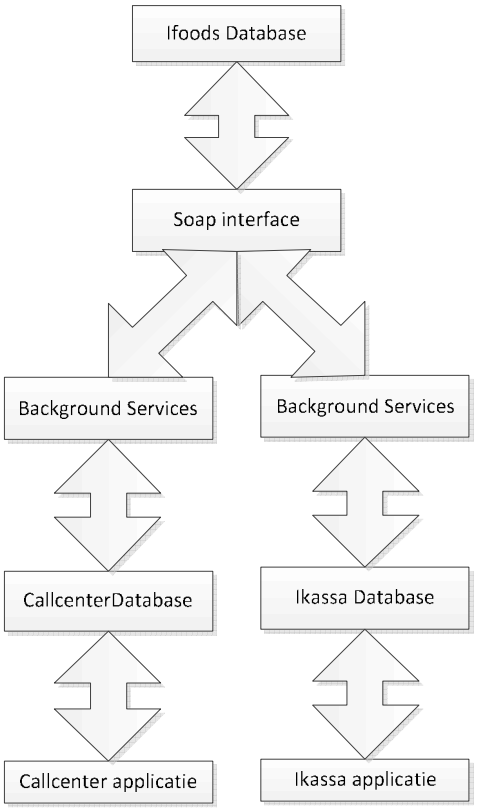
Zonder het opgeven van het stijl attribuut zou de knop er uitzien zoals elke ander knop die we kennen van Windows, een grijs kleur verloop die het gevoel van 3D moet geven (figuur 19). Door het toevoegen van het stijl is het resultaat een doorzichtig knop zoals wel zien in figuur 20.

8.0 Rup Fases Windows Service (Verder ontsluiting Ifoods)

8.1 Inceptie fase

8.1.1Change of Concept / Nieuwe functionaliteit

Halverwege het project is een wijziging doorgevoerd in de wensen en eisen van de opdrachtgever. Door het contracteren van een aantal ketens, is de vraag en het gebruik van het Ifoods callcenter sterk toegenomen. De bestaande methode die gehanteerd werd was het verwerken van de orders via de website van Ifoods.nl. Echter deze website wil een aantal zaken weten die voor een telefonische order helemaal niet relevant zijn zoals bijvoorbeeld een e-mailadres. Daarnaast werkt de website op basis van accounts met als gevolg dat er voor elke bestelling ook een account moest worden aangemaakt.



Om dit probleem te verhelpen wilde de opdrachtgever het nieuw te ontwikkelen kassasysteem ook inzetten op het callcenter voor de verwerking de telefonische bestellingen.

Ondanks dat het systeem voorbereid was op het feit dat binnen 1 kassa omgeving meerdere restaurants geplaatst kunnen worden, was er verder geen communicatie tussen Ifoods en het kassa systeem met betrekking tot lokaal ingevoerde bestellingen. Wel wordt er vanuit de kassa omgeving statusmeldingen terug gecommuniceerd naar Ifoods.

In de nieuwe situatie moet de informatie van de callcenter omgeving via de servers van Ifoods worden aangeboden aan een andere kassa omgeving. Op het moment dat zowel omgeving van het callcenter als de omgeving van kassa bestel informatie gaan uitwisselen wordt het een rommelig geheel waarbij niet duidelijk is wie op welk moment de eigenaar is van welke informatie. Hierdoor kunnen problemen ontstaan.

Op het moment dat een order van het callcenter aangekomen is bij de kassa omgeving moet deze hem accepteren. Door deze acceptatie komt de order in de status keuken terecht waardoor de kok de bestelling gaat bereiden. De kassa applicatie geeft de status door aan Ifoods die deze verwerkt. Een ogenblik later doet de callcenter een update van die zelfde status waarbij deze denkt dat deze nog steeds de status nieuwe heeft. Als vervolgens het kassa systeem de status weer ophaalt bij Ifoods ziet deze de bestelling weer als nieuw waardoor de kok de bestelling nogmaals gaat bereiden.

Om dit probleem op te lossen hebben we het volgende bedacht. Bij de installatie van een omgeving dient gekozen te worden in welke modus de software moet draaien. Op dit moment zijn er twee modi namelijk “Kassa” en “Callcenter”.

In de software hebben we een aantal beperkingen opgenomen. In de modus “Callcenter” worden ingevoerde bestellingen bij Ifoods aangeboden als nieuwe bestellingen waarna ze het normale traject doorlopen. Dit normale traject verwerkt de bestelling en bepaalt dan of de bestelling per Fax, mail of kassa aangeboden moet worden.

Verder mag de software in de modus van Callcenter geen wijzingen doorvoeren op de status van een bestelling en enkel de status opvragen bij Ifoods.

Voor de modus “Kassa” werkt dit systeem anders om. Een bestelling wordt nooit gesynchroniseerd met Ifoods maar bij een wijziging in status wordt dit wel gecommuniceerd met Ifoods.

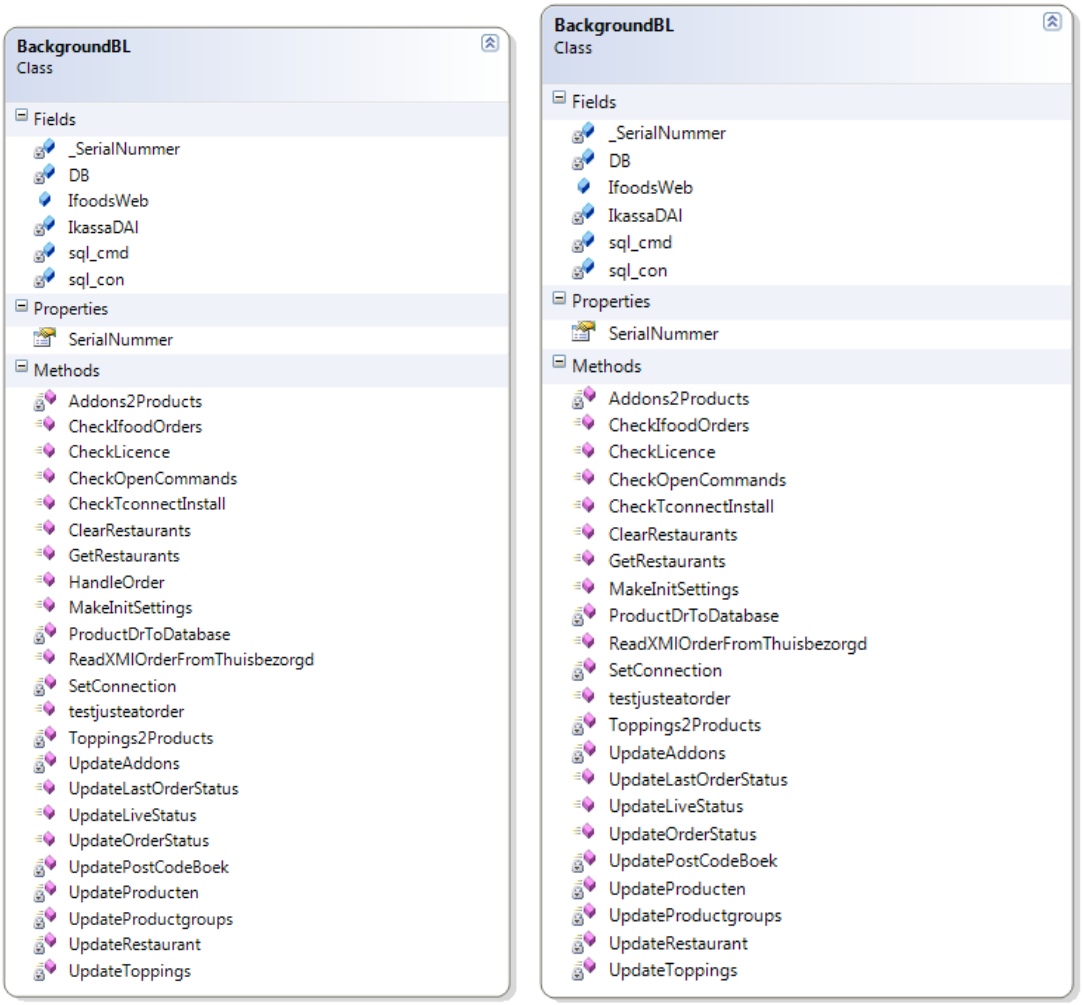
Deze strategie heeft als extra voordeel opgeleverd dat het mogelijk is om de bestellingen die in het callcenter binnen komen ook aan te bieden aan restaurants die op dit moment nog geen kassa systeem hebben. Daarnaast kan, door de interne structuur van Ifoods, nu heel makkelijk inzichtelijk worden gemaakt welke bestellingen binnen zijn gekomen via het callcenter.

8.2 Elaboratie fase

Omdat voor het tonen van de bestel gegevens in de kassa applicatie al diverse functies in de DAL zijn opgenomen was het niet nodig voor dit project een nieuwe elaboratie fase te doorlopen. Immers alle nodige klassen en hun uitwerking waren al aanwezig.

8.3 Construction Fase

Bij het uitbreiden van de bestaande functionaliteit is er een kleine functie bijgekomen die er voor zorg draagt dat de order naar de Ifoods server wordt gekopieerd genaamd HandleOrder(). In onderstaande diagrammen is dat weergegeven. Deze haalt de bestelling met bestelregels uit de database en bied deze aan aan de webservice.



Omdat de webservice door het gebruik van SOAP niet met eigen objecten overweg kan, worden de orderegels overgestuurd als losse variabelen. Zoals in onderstaand voorbeeld is te zien is dit een redelijk omslachtige methode.

```
IEnumerable<OrderLine> Orderlines = IkassaDAl.GetOrderlinesByOrderID(OrderID);

foreach (OrderLine uploadOrderline in Orderlines)
{
    IfoodsWeb.RecieveOrderLine(uploadOrderline.OrderLineID, uploadOrderline.OrderID,
        uploadOrderline.ProductID, uploadOrderline.Quantity, uploadOrderline.Prijs);
}
```

Figuur 21 Versturen van gegeven naar Ifoods webservice interface

In theorie zouden we de zelfde techniek kunnen gebruiken die we hebben toegepast bij het ontvangen van informatie. Daar is gekozen om de informatie in een DataSet te verpakken. Er zit echter een groot verschil tussen de beide systemen. Zoals eerder beschreven is de DAL aan de kant van de nieuwe applicatie ontwikkeld op basis van de nieuwe datastructuren welke zijn introduceert in .Net 3.0 en hoger. Dit model is niet 1 op 1 compatible met het oude DataSet model. In het oude systeem krijgen we vanuit de DAL een DataTable aangeleverd die zonder probleem verpakt kan worden in een DataSet.

Om de gegevens uit de nieuwe structuur te halen en deze te plaatsen in een Dataset zou betekenen dat men evengoed door de data zou moeten lopen om vervolgende de informatie te plaatsen op de juiste plaats in de DataSet. Aan de kant van de webservice zou men vervolgens weer door de data moeten heen lopen om per regel de informatie in de database weg te schrijven. Dit laatste zou automatische gebeuren door de onderliggende code van de DAL.

Qua verwerkingstijd zou het ongeveer net zoveel tijd kosten echter zou de code er een stuk minder duidelijk door worden. Daarbij door het verwerken van de order in losse stukjes te hakken kan er ook beter worden ingespeeld op eventuele fouten die kunnen optreden. Dit is echter op dit moment nog niet geïmplementeerd.

9.0 Rup Fases Webservice (Recieve Data at Ifoods)

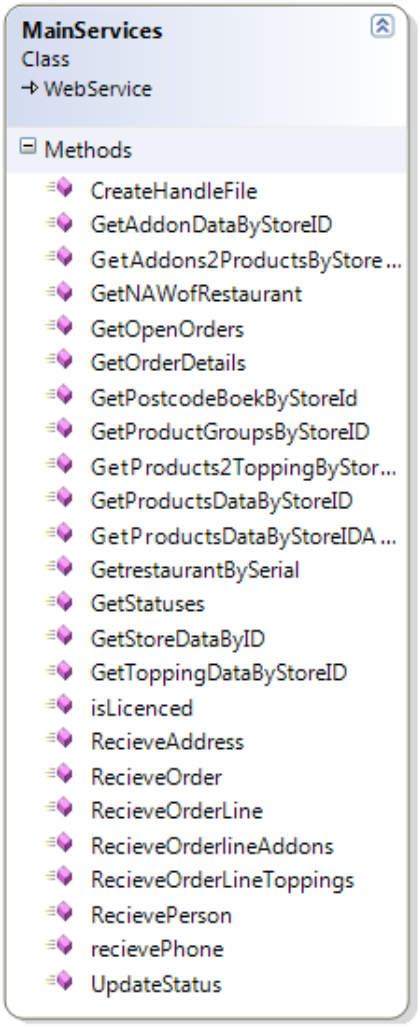
9.1 Inceptie fase

Zoals eerder beschreven, is er een noodzaak ontstaan om informatie te kunnen ontvangen van buitenaf. Deze orders moeten vervolgens door een achtergrond proces van Ifoods worden opgepakt en verder worden verwekt.

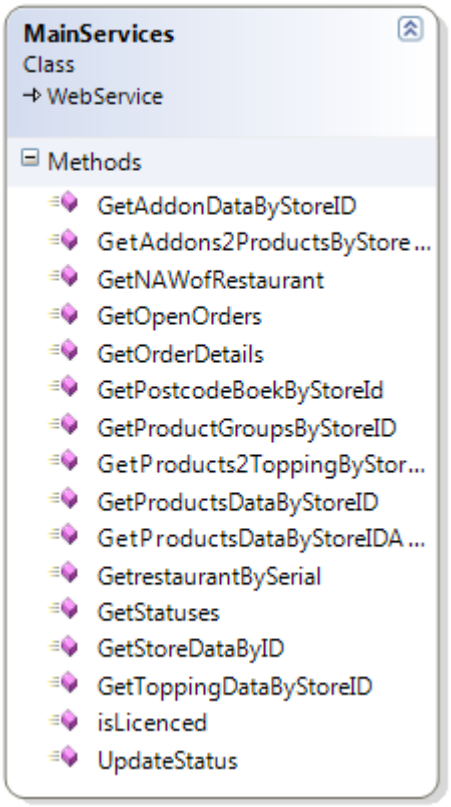
9.2 Elaboratie fase

Om het mogelijk te maken om informatie te ontvangen van buiten af zijn er een aantal nieuwe functies toegevoegd aan de webservice. Deze functies zijn te herkennen aan de prefix “Recieve” voor de functienaam.

In de DAL laag zijn een aantal extra functies gedefinieerd die het mogelijk maken om informatie toe te voegen. Omdat de data die binnen komt gelijk is aan de informatie die benodigd is voor het toevoegen van de informatie aan de database wordt de informatie vanuit de BL rechtstreeks doorgezet naar de DAL.



Figuur 19 Nieuwe BL van de Webservice interface



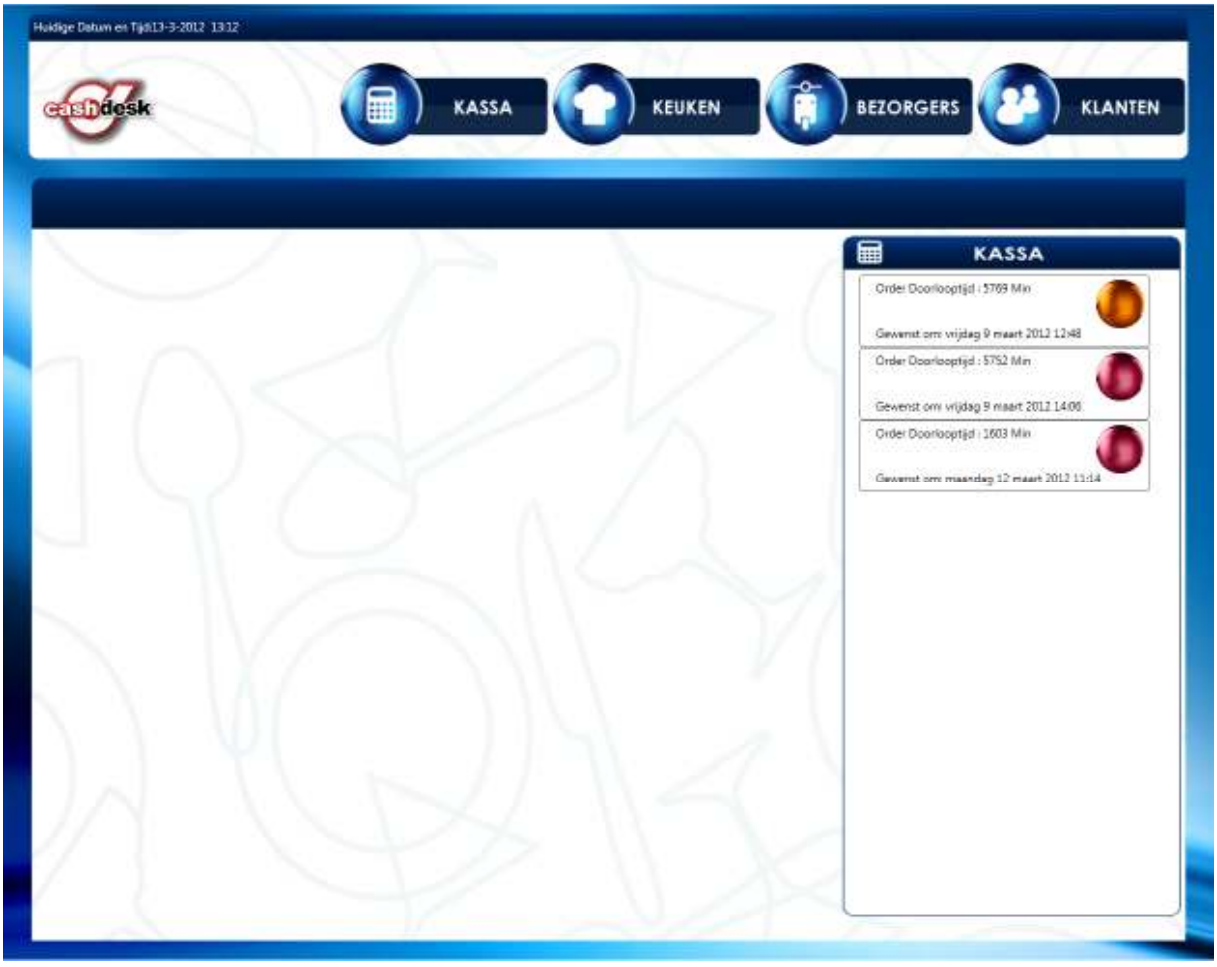
Figuur 20 Oude BL van de webservice interface

10 Project afronding

Volgens de planning uit het Plan van Aanpak zou er aan het eind van het project een werkende beta applicatie worden opgeleverd. Dit is gelukt en op het moment van schrijven draait er zelfs al een 2de beta versie in het callcenter van Ifoods. De interne beta dient tevens voor een aantal testdoeleinden.

Een van de testen die wordt uitgevoerd is de Shadowtest. Hierbij worden alle handelingen die het systeem volgens de eerder wensen en eisen zou moeten kunnen verrichten getest. Ook wordt aan de hand van usability test gekeken om het systeem werkbaar is voor een nieuwe en niet getrainde gebruiker. Daarnaast hebben de gebruikers ook de opdracht gehad om te kijken of ze het systeem kunnen ontregelen.

Om te voorkomen dat de resultaten beïnvloedbaar zouden worden door de kennis van de ontwikkelaar / betrokken personen, zijn de diverse testen opgesteld door een andere programmeur van Ifoods. Ook de personen die met het product werken / testen zijn niet betrokken geweest bij de ontwikkeling. Op deze manier is de uitkomst van de diverse testen het meest bruikbaar en reëel.



Figuur 22 Definitieve layout begin scherm

11 project evaluatie / reflectie

11.1 Webservice

Tijdens het project was ik er van overtuigd dat de .Net 2.0 webservices enkel met standaard objecten kon werken die onderdeel waren van het .Net framework. Daardoor zijn er een aantal aannames gemaakt en is er bij het ontwikkelen van de webservices gekozen voor de generieke oplossing gebruik makend van de DataSet en andere standaard CLR objecten.

Echter ben ik tijdens het project in gesprek geraakt met andere programmeurs. Deze hebben me verteld en getoond dat ook de .Net 2.0 webservice de mogelijkheid bied om zo gehete datacontracten af te sluiten daarmee het mogelijk te maken om zelf bedachte objecten uit te wisselen op een “Strongly Typed” methode.

Door het gebruik van “Strongly Typed” objecten, is de kans op fouten kleiner. De IDE weet namelijk precies welke objecten uit de services terug komen. De combinatie van een modere IDE en “Strongly Typed” zorgt er voor dat de ontwikkeltijd wordt verlaagd. Dingen als intellisense maken het meest profeit van “Strongly Typed”.

Een ander groot nadeel van de gekozen methode is dat ik aan de kant van de Windows services nu handmatig de type de objecten moet benomen. Bij het gebruik van een Datacontract zou het .Net framework dit automatisch hebben gedaan.

11.2 Aanpassing wensen en eisen

Tijdens het project zijn de wensen van de opdrachtgever veranderd. Terugkijkend op het project heeft dat tot een aantal problemen geleid.

Allereerst is het project uitgelopen qua planning. Extra functionaliteit betekend extra werk. In overleg met de opdrachtgever is besloten dat het opstellen van de inceptiefase en eleboration fase rapporten geen prioriteit hadden voor de uitbreidingen die zijn gedaan.

Omdat ik zelf vind dat een goede ontwikkeling staat en valt met ene goed ontwerk, en je een goed ontwerp niet ad-hoc kan ontwikkelen heb ik dit verzoek naast me neer gelegd en een middenweg gekozen.

De nieuwe functionaliteit zijn een uitbreiding op bestaande architectuur waarvan de meeste diagrammen tijdens het project al waren opgesteld. Daarom heb ik deze diagrammen verder uitgewerkt en gebruikt tijdens de ontwikkeling. Echter door tijdgebrek heb ik de diagrammen niet verwerkt tot een geheel rapport en als losse bestanden / tekening overgedragen aan de opdrachtgever. Verder heb ik deze diagrammen waar nodig gebruikt in dit verslag.

11.3 Beveiliging

Een punt waar in het verslag weinig tot geen aandacht aan geschonken is, is het thema beveiliging.

Ondanks dat het buiten de scope van de opdracht en het afstudeer traject valt, is tijdens de ontwikkeling ook rekening gehouden met dit thema.

Zo vind de uitwisseling van gegevens niet plaats over het standaard http protocol maar via SSL over HTTPS. Door deze versleuteling is het niet mogelijk om de gegevens tussen aanvrager en verzender te onderscheppen en te bekijken of te veranderen.

Ook is de webservice niet publiekelijk beschikbaar maar zit deze achter een beveiligd subdomein. Deze beveiliging is op basis van een gebruikersnaam en wachtwoord. Zonder deze gegevens is het niet mogelijk om de services aan te spreken.

Omdat de webservice geen publiekelijk karakter heeft, is de WDSL van de webservice in de productie omgeving uitgeschakeld. In de WDSL file staat beschreven welke publieke functies een webservice aanbied en welke parameters er worden vereist. Zonder deze gegevens is het aankoppelen van andere software op de webservice nagenoeg onmogelijk.

Verder is er op de webservice een veiligheidsmechanisme ontwikkeld, die bijhoud welke ip-adressen data aanleveren. Zo kan achteraf worden bepaald wie verantwoordelijk is voor het veranderen of toevoegen van data.

Bij het versturen van bestellingen, hebben we de aanname gemaakt dat bestellingen altijd nog betaald moeten worden. Dit is een logische aanname omdat ideal niet mogelijk is via een telefoon, het callcenter geen informatie heeft met betrekking tot de door Ifoods gebruikte besteltegoeden. De velden die normaal voor het gebruik van tegoed en / of ideal betalingen worden gebruikt zijn niet publiekelijk toegankelijk.

```
public void RecieveOrder(Guid OrderID, DateTime OrderDateTime,Guid StoreID,Guid
PersonID,Guid AddressID,Guid PhoneID,string Comment,long orderbedrag,string
orderstatus)
```

In de reeds bestaande componenten van Ifoods voor het verwerken van de bestellingen zit al beveiliging die de bestelling op bestelregel naloopt en het totaal bedrag zelf berekend. Hierdoor zou manipulatie van het totaal order bedrag ook uitgesloten zijn.

Als laatste veiligheidsmaatregel wordt de restaurant houder op de hoogte gesteld als de prijzen van een bestelling of besteld product afwijken van de prijzen die op dat moment worden gehanteerd door de restaurantkassa. Het is dan aan de kassamedewerker om te kijken of het gaat om een verschil tussen de order gegevens in de kassa en intermediair of dat het gaat om een geval van fraude. In het laatste geval zou de restauranthouder contact moeten opnemen met de intermediair om het probleem verder uit te zoeken en op te lossen.

11.4 Installer Applicatie

Bij het starten van het project was het idee van Ifoods om de software aan te bieden als een “doe het zelf” installatie voor restaurants. Restaurants zouden dan zelf de benodigde software kunnen downloaden van de website van Ifoods. Deze software zouden ze vervolgens op een of meerder computers kunnen installeren en gebruiken.

Tijdens de ontwikkeling en eerste beta test is Ifoods van dit standpunt afgeweken. Dit komt omdat de applicatie ontwikkeld is voor gebruik met een touchscreen en de meeste mensen geen touchscreen systeem zullen hebben. Ook is gebleken uit testen dat het installeren ondanks de gebruik van een installer vaak tot problemen leid.

Daarom heeft Ifoods besloten om het programma in combinatie met hardware te gaan uitleveren. De hardware wordt bij de klant afgeleverd en voor geïnstalleerd.

Omdat de installatie nu door medewerkers van Ifoods wordt uitgevoerd is de installer minder uitgebreid gemaakt. We hebben er voor gekozen om een standaard setup project te gebruiken. Dit setup project genereert een MSI bestand waarmee de basis installatie gedaan kan worden.

Omdat we bij toekomstige updates niet alle klanten een bezoek willen brengen heb ik een update mechanisme gebouwd die er voor zorgt dat zowel de Windows service als de kassa applicatie geüpdate kunnen worden. Voor de kassa applicatie maken we gebruik de ClickOnce technologie. Aangezien deze technologie niet gebruikt kan worden voor Windows service hebben we hier een kleine extra update service toegevoed. Deze controleert er een nieuwe versie beschikbaar is en download deze vanaf het internet. Als dit gelukt is wordt de Windows service gestopt, de update geïnstalleerd en daarna weer gestart.

12 Verantwoording beroepstaken

2.1 Opstellen gegevens model oor een database

Deze competentie is verwerkt in het modeleren van zowel de bestaande Ifoods database als de nieuw ontwikkelde database voor de kassa systemen en windows service.

Concepten als een op een (beselregelproduct op product), een op veel (klant met bestellingen) en veel op veel (mogelijk bijproducten bij producten)relaties zijn gebruikt voor het uitnormaliseren van de database.

Om deze database te moduleren, hebben we gebruik gemaakt oa het Klasse diagram uit UML

3.1 Ontwerpen software architectuur

Deze competentie is verwerkt in de beschrijving van de bestaande en nieuw te ontwikkelen structuur. Vanaf toplevel is er gekeken met welke systemen de nieuw te ontwikkelen oplossing moet communiceren. Van daaruit is er een opsplitsing gemaakt naar functioneel en informatie oogpunt.

Op software niveau is de software architectuur opgedeel in het n-tier model. Door gebruik te maken van sequence diagrammen is vastgesteld hoe de informatie tussen verschillende componenten verloopt

3.2 Ontwerpen systeemdeel

Deze competentie is verwerkt in de opbouw van de diverse nieuwe producten. Er is gebruik gemaakt van prototyping ter behoeve van het vaststellen van het grafische ontwerp. Daarnaast is UML ingezet voor het beschrijven van systeem functies en de interactie tussen systeem en gebruiker.

Verder zijn er Design patterns gebruikt zoals het N-tier model. Ook zijn systeem functies gescheiden op basis van fuctionaliteit en verwerkt in verschillende componenten.

3.3 Bouwen applicatie

Om de probleem stelling op te lossen zijn er meerder applicaties ontwikkeld. Door gebruik te maken van bijvoorbeeld het N-tier model zijn de componenten van de ontwikkelde applicaties makkelijk her te gebruiken en wijzigbaar. Daarnaast hebben we de ontwikkel omgeving afgestemd op beschikbare middelen zoals hardware en kennisniveau van de mensen die het systeem moeten onderhouden. Daar waar toch is gekozen om nieuwere technieken te gebruiken is dit voorzien van de nodige documentatie en uitleg aan het huidige team.

Uiteindelijk is de applicatie zoals verwacht opgeleverd aan de opdrachtgever.

13 Literatuur

Katwijk, J. v. (sd). *Spiraal_model*. Opgeroepen op 03 08, 2012, van Wikipedia: http://nl.wikipedia.org/wiki/Spiraal_model

Kleppe, J. W. (2004). *Praktisch UML 3de Editie.*

Microsoft. (sd). *A Performance Comparison of Windows Communication Foundation (WCF) with Existing Distributed Communication Technologies*. Opgeroepen op 2 27, 2012, van Microsoft: http://msdn.microsoft.com/en-us/library/bb310550.aspx#wcfperform_topic4

Ordina. (sd). *Fases*. Opgeroepen op 2 2012, 29, van Naslagsite Rup op maat: http://www.rupopmaat.nl/naslagsite2011/index.html

Rup op Wikkipedia. (sd). Opgeroepen op 11 27, 2011, van Wikipedia: http://nl.wikipedia.org/wiki/Rational_Unified_Process

Afstudeerplan

Informatie afstudeerder en gastbedrijf (*structuur niet wijzigen*)

Afstudeerblok: 2011-2.2 (start uiterlijk 14 november 2011)
Startdatum uitvoering afstudeeropdracht: 14 November 2011
Inleverdatum afstudeerdossier volgens jaarrooster: 19 maart 2012

Studentnummer: **Achternaam**:Renooij Dhr
Voorletters: J.A. (*) *weghalen niet van toepassing*
Roepnaam: John
Adres:
Postcode:
Woonplaats:
Telefoonnummer:
Mobiel nummer:
Privé emailadres:

Opleiding:
Locatie: Den Haag
Variant: voltijd (*) *weghalen niet van toepassing*

Naam studieloopbaanbegeleider: h.g.j.bechet-tjoonk
Naam begeleidend examinator: G.A.Mijnarends
Naam tweede examinator: O.Zor

Naam bedrijf: Ifoods
Afdeling bedrijf: ICT
Bezoekadres bedrijf: NVT (in overleg goedgekeurd met heer Mijnarends op 8 Nov)
Postcode bezoekadres:
Postbusnummer:
Postcode postbusnummer:
Plaats:
Telefoon bedrijf:
Telefax bedrijf:
Internetsite bedrijf: www.ifoods.nl

Achternaam opdrachtgever:van Alten dhr (*) *weghalen niet van toepassing*
Voorletters opdrachtgever*:R*
Titulatuur opdrachtgever:
Functie opdrachtgever: ICT Manager
Doorkiesnummer opdrachtgever:
Email opdrachtgever:

Achternaam bedrijfsmentor:van Alten dhr (*) *weghalen niet van toepassing*
Voorletters bedrijfsmentor: R
Titulatuur bedrijfsmentor:
Functie bedrijfsmentor:ICT Manager
Doorkiesnummer bedrijfsmentor:
Email bedrijfsmentor:

NB: bedrijfsmentor mag dezelfde zijn als de opdrachtgever

Doorkiesnummer afstudeerder*:*
Functie afstudeerder (deeltijd/duaal):

Titel afstudeeropdracht: Ontwikkellen van een POS systeem icm portal intergratie

Opdrachtomschrijving

1. Bedrijf

Ifoods treedt op als intermediair tussen consumenten en diverse bezorgrestaurants. Het doel van Ifoods is het zorgen dat online eten bestellen toegankelijk wordt voor iedereen, klant of bezorgrestaurant. Ifoods streeft er naar om de beste eten-bestelsite te ontwikkelen en te onderhouden, en om bestellingen zo snel en betrouwbaar mogelijk door te geven van de klant naar het bezorgrestaurant.

Naast dat Ifoods zich richt op de consument, heeft Ifoods ook een afdeling die zich richt op B2B oplossingen. Deze oplossingen kunnen in sommige gevallen leiden tot het vervangen van, de al dan niet ingehuurde, inhuis-catering. Deze diensten zijn bijvoorbeeld interessant voor een help/ services-desk waarbij medewerkers vaak ook in de avonduren werkzaam zijn.

Door het gerichte meedenken met de klant is het systeem zo aanpasbaar dat voor ieder restaurant of klant een oplossing geboden kan worden.

2. Probleemstelling

Voor veel restaurants is het aanschaffen van een horeca kassasysteem een investering die niet opweegt tegen de voordelen die de software doet beloven. Om deze reden wordt er in deze restaurant vaak gewerkt met losse papiertjes of worden bestellingen gewoon uit het hoofd bereid en eventueel bezorgd.

Bestellingen van buiten de winkel worden veelal doorgebeld (door de klant of diensten als Ifoods en Just-eat) of komen binnen doormiddel van fax en / of email. Doordat er verschillende kanalen zijn gebeurt het meer dan eens dat bestellingen te laat opgemerkt worden of gewoon verdwijnen waardoor de klant langer moet wachten of zelfs helemaal vergeten wordt.

Ook is het lastig voor de eigenaar een goed overzicht te hebben aan het eind van de dag met betrekking tot de omzet, kosten, gewerkte uren en andere administratieve cijfers.

Om het bestel proces te vergemakkelijken en eenduidig te laten verlopen alsmede het inzichtelijk maken van diverse financiële gegevens is Ifoods op zoek naar een hard en softwareoplossing. Deze oplossing moet de administratieve meerwaarde bieden zodat restaurant de oplossing graag willen implementeren alsmede een 2-ways interface bieden naar de systemen van ifoods.nl.

3. Doelstelling van de afstudeeropdracht

Het ontwikkelen van een applicatie waarin de volgende punten aanwezig zijn:

- De mogelijkheid tot het aannemen van orders (Balie en telefoon orders)
- Koppeling van Postcode boek en bestaande klantenbestand voor het snel invoeren van adresgegevens
- Eenvoudige te bedienen interface voor individuele verkoopacties
- Directe invoer / verwerking van internetbestellingen (via Ifoods en andere leveranciers)
- Tijdsregistratie per order vanaf moment binnenkomst tot moment van het verlaten van de winkel.
- Vastleggen NAW gegevens van nieuwe klanten
- Het tonen van ordergegevens en statistieken
- Orders koppelen aan betreffende koerier tbv afrekening.
- Diverse rapportage zoals de dagstaat, omzetten bezorging, afhaal, telefoon en internet
- Balans opmaak (Kas, op rekening, ideal betalingen)
- Optioneel dmv export gegevens geschikt maken voor koppeling naar administratiepakket zoals Informer

4. Resultaat

Het resultaat van de afstudeeropdracht zal bestaan uit meerdere modulaire applicaties. Deze applicaties zullen zijn gebaseerd op nieuwe technieken zoals C#, Soap Interfaces en T-SQL.

Het doel van deze applicaties is het verbeteren van de dienstverlening van Ifoods en het stroomlijnen van het verkoopproces van de restaurants. Daarnaast moet het e.e.a. inzichtelijk maken voor de restaurants.

Veder moet er een interface in de applicaties worden opgenomen die het mogelijk maakt om vanuit externe diensten orders te ontvangen. Een voorbeeld hiervan is de mogelijkheid van orders per email.

5. Uit te voeren werkzaamheden, inclusief een globale fasering, mijlpalen en bijbehorende activiteiten

De methode waarmee gewerkt zal worden is waarschijnlijk RUP. Dit maakt het mogelijk om de verschillende deelproducten langs elkaar te ontwikkelen en d.m.v. iteraties gaandeweg de deel producten uit te werken tot een volledig product.

Fasering:

- Inceptiefase (Aanvang)
  - ❖ Opstellen van Requirements
  - ❖ Opstellen van Use Cases & Scenario's
  - ❖ Opstellen van Business Cases
  - ❖ Opstellen van een Risk Assesment
  - ❖ Opstellen van een Project Plan

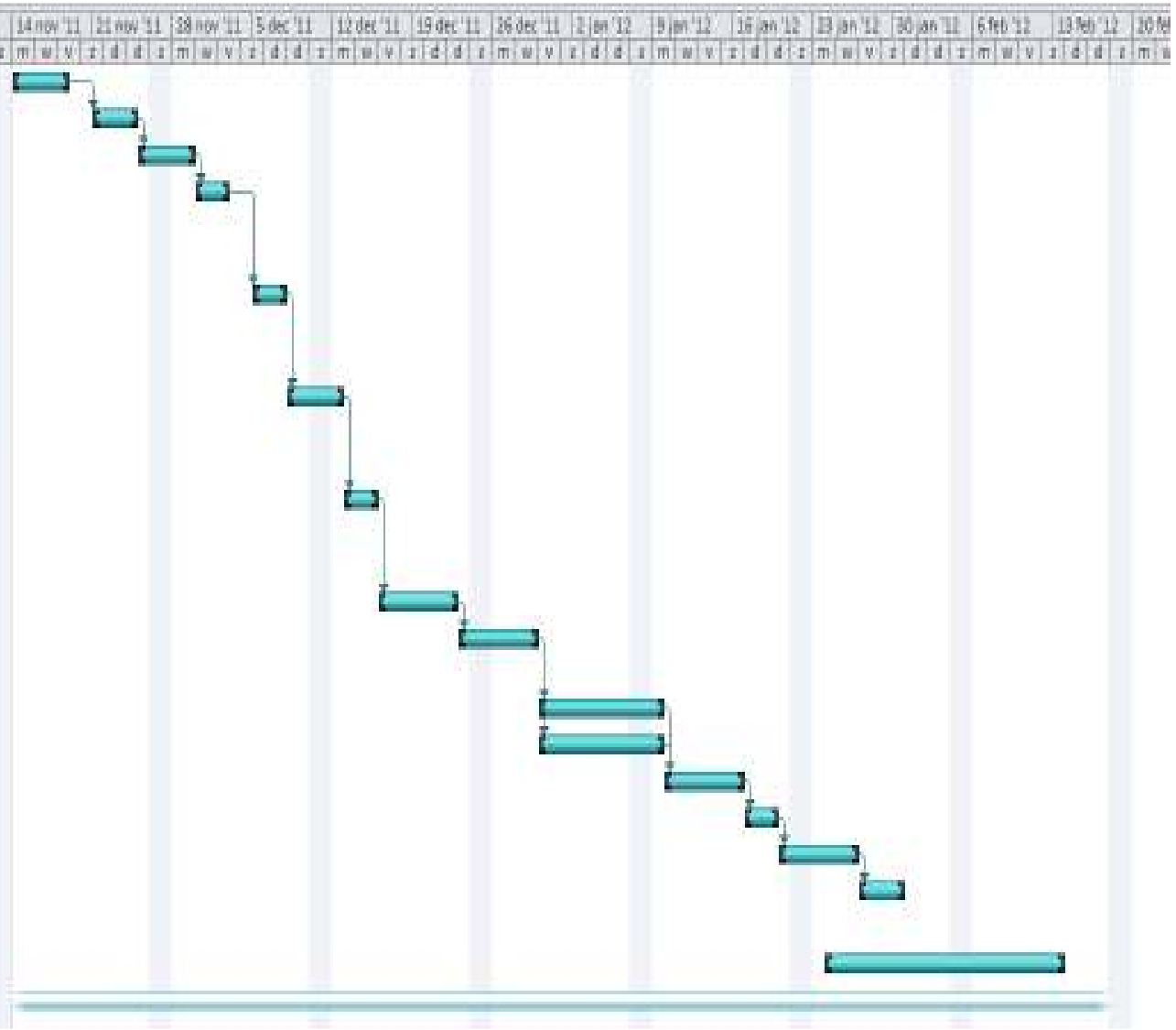
- Elaboratiefase (Detaillering)
  - ❖ Verder uitwerken van Use Cases & Scenario's
  - ❖ Opstellen Object diagrammen
  - ❖ Opstellen Sequentie diagrammen
  - ❖ Opstellen Klassen diagrammen
  - ❖ Uitweken van Database ontwerp

- Constructiefase (Bouw)
  - ❖ Testen van deelproducten dmv unit testing
  - ❖ Ontwikkelen van de applicatie

- Transitiefase (Overgang)
  - ❖ Opleveren en Overdragen

Bij aanvang van het project zal worden begonnen met een het opstellen van een plan van aanpak en het project zal worden afgesloten met het opstellen van een afstudeerverslag. Deze twee documenten vallen echter uit buiten de fasering van RUP en zijn hierin daarom niet opgenomen.

	Taaknaam	Duur	Begind
1	Opstellen Plan van Aanpak	5 dagen	maa 1
2	Opstellen van requirements	4 dagen	maa 2
3	opstellen van Use Cases	3 dagen	vri 25-
4	Opstellen van Object en Sequentie diagrammen vd product installer	3 dagen	woe 3
5	Opstellen van Object en Sequentie diagrammen vd Main applicatie	3 dagen	maa 5
6	Opstellen van Object en Sequentie diagrammen vd Background services	3 dagen	don 8-
7	Opstellen van Object en Sequentie diagrammen vd Portal interface	3 dagen	din 13
8	Uitwerken Klasse diagram	5 dagen	vri 16-
9	Uitwerken van Database ontwerpen	5 dagen	vri 23-
10	Ontwikkelen Portal Interface	7 dagen	vri 30-
11	Ontwikkelen Services	7 dagen	vri 30-
12	Ontwikkelen Main applicatie	5 dagen	din 10
13	Ontwikkelen Client Installer	3 dagen	din 17
14	Acceptatie test	5 dagen	vri 20-
15	Oplevering en overdracht	2 dagen	vri 27-
16			
17	Documentatie School	15 dagen	din 24
18	Einde stage	70 dagen	



6. Op te leveren (tussen)producten

4 Deelapplicaties bestaand uit:

- Interface portal
  De interface zorgt er voor dat de nieuwe applicatie kan communiceren met de bestaande Ifoods omgeving. Deze interface zal aan de kant van Ifoods worden geïmplementeerd en benaderbaar zijn voor de nieuwe applicatie dmv een SOAP interface.
- Backend Services
  De services zal zorg dragen van de uitwisseling van gegevens tussen de database en verschillende interfaces zoals bijvoorbeeld Ifoods SOAP interface.
- Main interface
  De main interface is waar de klant mee werkt. Hier kunnen orders worden ingevoerd en orders worden afgelezen. Afhankelijk van de modus waarin de applicatie werkt worden 1 of meerdere schermen getoond (Bijvoorbeeld Order Invoer en Keuken overzicht alsmede Product onderhoud)
- Product Installer
  Om het product op een eenvoudige wijze bij de klant te installeren en up to date te houden zal gebruik worden gemaakt van een custom installer. Deze zal eenmalig in het netwerk een SQL server instaleren die als master database zal dienen.

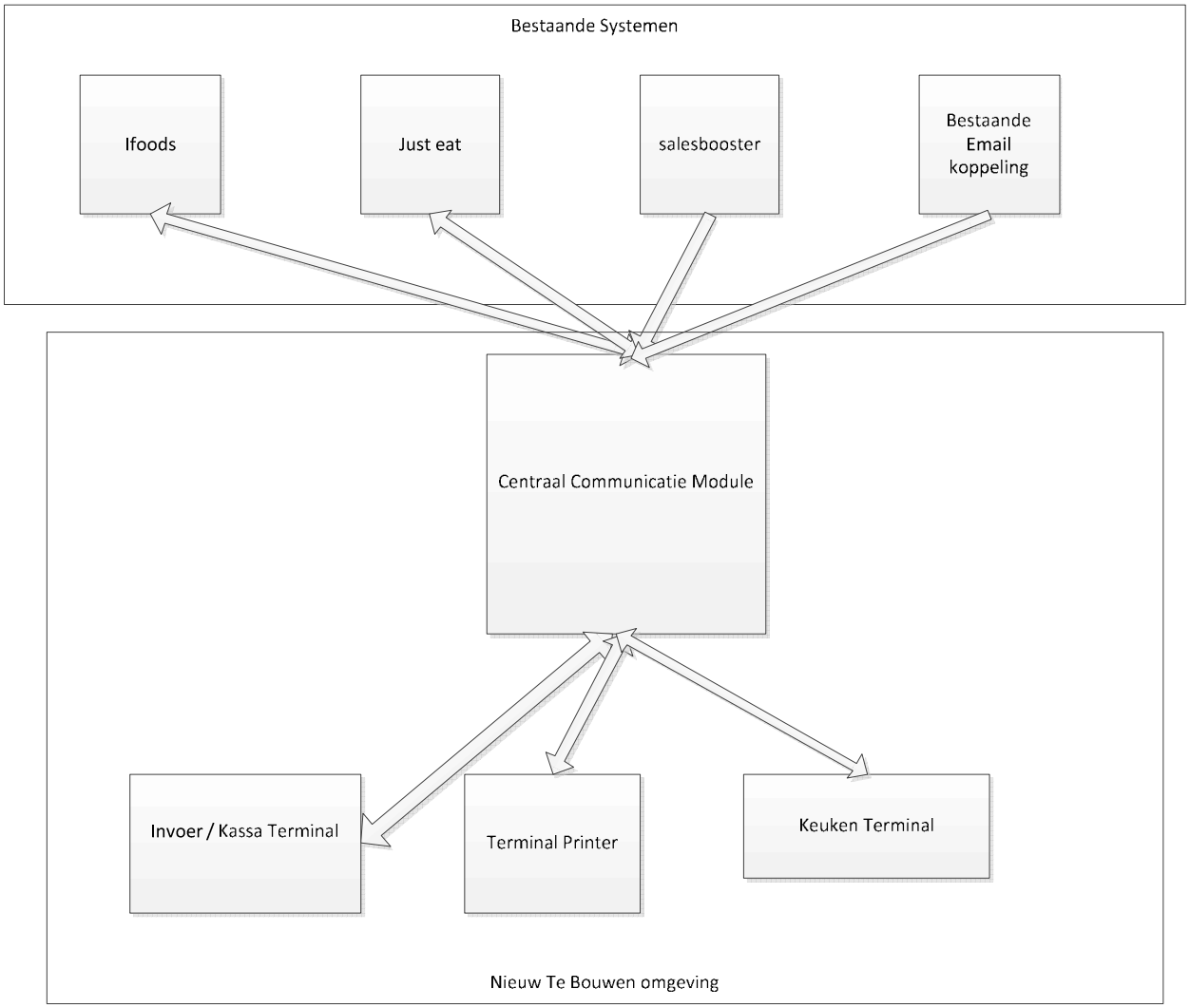
Systeem documentatie bestaande uit:

Inceptie Rapport

- Requirements
- Use Cases & Scenario's
- Business Cases
- Risk Assesment
- Project Plan

Elaboratie Rapport

- Use Cases & Scenario's
- Object diagrammen
- Sequentie diagrammen
- Klassen diagrammen
- Database ontwerp



7. Te demonstreren competenties en wijze waarop

Binnen de afstudeer opdracht wil ik de volgende competenties in het bijzonder aantonen:

2.1 Opstellen gegevens model voor een database. (niveau 3)
3.1 Ontwerpen softwarearchitectuur (niveau 4)
3.2 Ontwerpen Systeemdeel (niveau 4)
3.3 Bouwen applicatie. (niveau 3)

De keuze voor deze competenties komt voort uit de te bouwen applicatie omgeving waarbij **integratie en conversie vanuit meerdere bronnen** (2.1) plaats gaan vinden. Door de noodzaak data , data uitwisseling en gebruikers interface van elkaar te scheiden, ontstaat een **software architectuur** (3.1 en 3.2) bestaand uit meerdere lagen die met elkaar moet communiceren en gegevens moet uitwisselen.

De complexiteit binnen deze applicatie komt voort uit het feit dat data niet alleen via de software zelf binnen komt maar ook beschikbaar wordt gemaakt via interfaces van derden. Daarnaast zijn er binnen de nieuw te ontwikkelen omgeving verschillende rollen die wel of niet informatie kunnen plaatsen of kunnen displayen. Ingevoerde data moet op de betreffende systemen binnen enkele ogenblikken zichtbaar worden.

Het eindproduct (3.3) zal samen met de systeem documentatie als compleet programma worden opgeleverd aan ifoods.

B2C4U IN OPDRACHT VAN IFOODS

Elaboratie fase

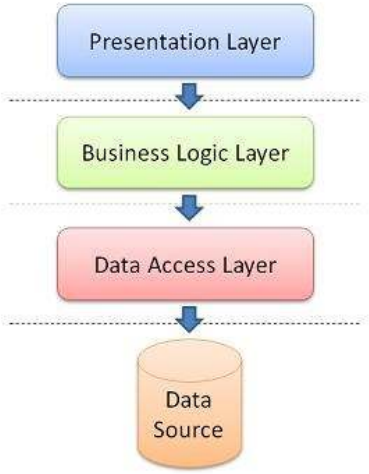
rapport Background services

John Renooij

Inhoud

Systeem architectuur

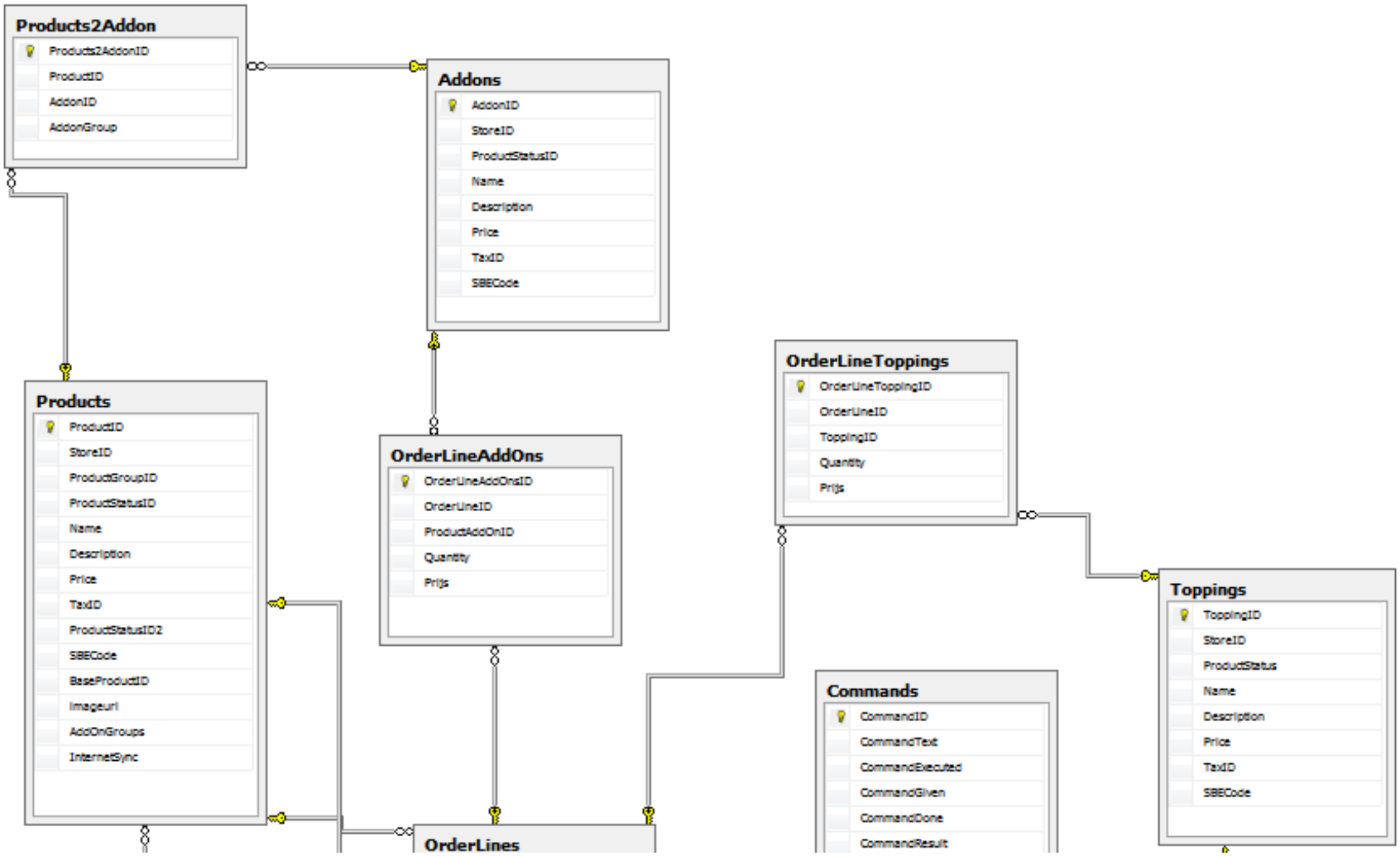
Het systeem wordt gebouwd volgens het n-tier model. Het n-tier model is een lagen model die datatoegang, business logic en interface van elkaar scheid.

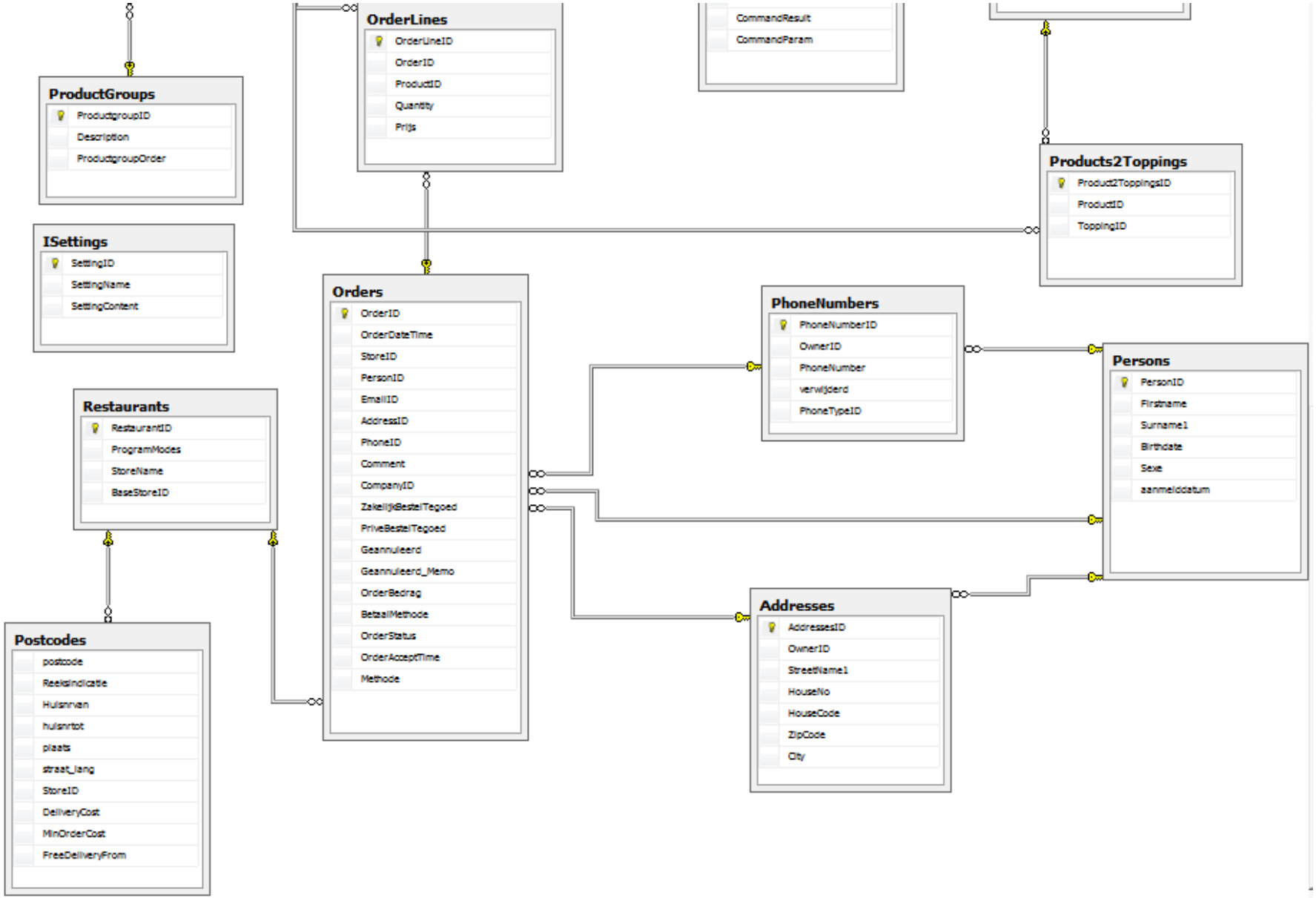


Figuur 1 Standaard definitie N-tier model

Omdat de background services geen interface heeft zijn er echter maar 2 lagen die effectief worden gebruikt. De presentatie laag wordt enkel gebruikt voor de start van de applicatie.

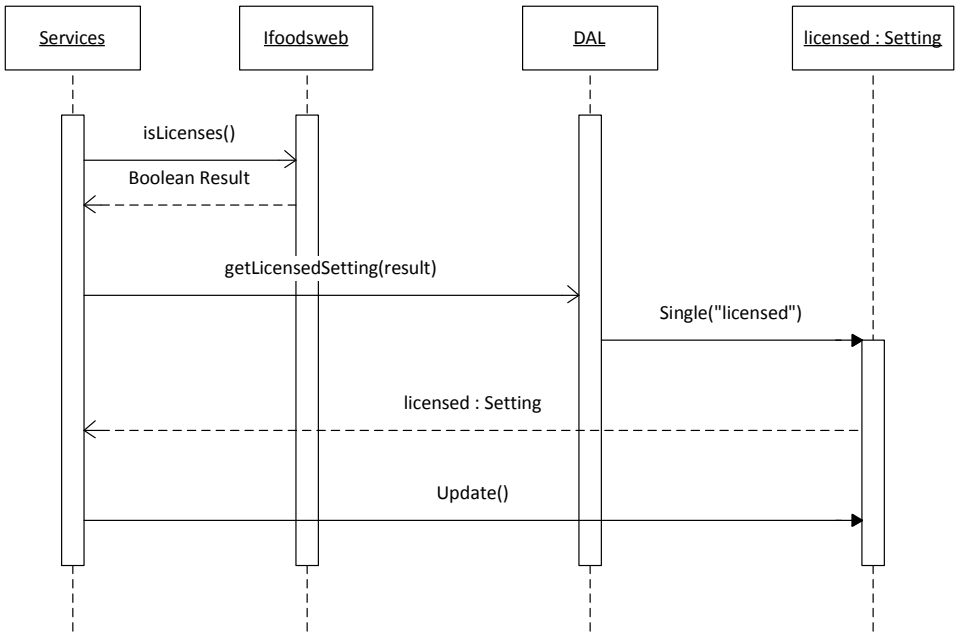
ERD


Products2Addon
Products2AddonID
ProductID
AddonID
AddonGroup
Addons
AddonID
StoreID
ProductStatusID
Name
Description
Price
TaxID
SBECode
Products
ProductID
StoreID
ProductGroupID
ProductStatusID
Name
Description
Price
TaxID
ProductStatusID2
SBECode
BaseProductID
ImageUrl
AddOnGroups
InternetSync
OrderLineAddOns
OrderLineAddOnsID
OrderLineID
ProductAddOnID
Quantity
Prijs
OrderLineToppings
OrderLineToppingID
OrderLineID
ToppingID
Quantity
Prijs
Toppings
ToppingID
StoreID
ProductStatus
Name
Description
Price
TaxID
SBECode
Commands
CommandID
CommandText
CommandExecuted
CommandGiven
CommandDone
CommandResult
OrderLines

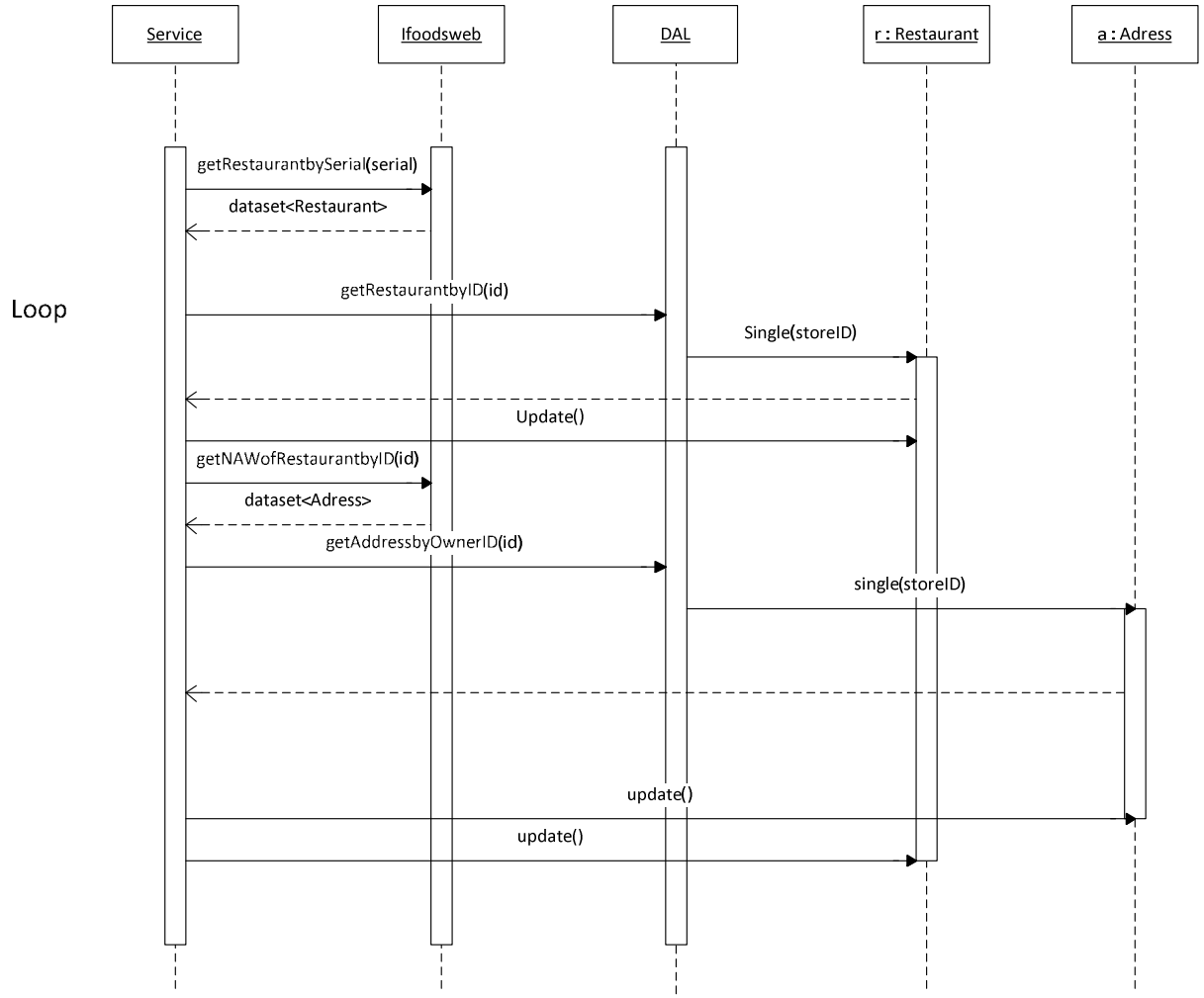

ProductGroups
ProductgroupID
Description
ProductgroupOrder
OrderLines
OrderLineID
OrderID
ProductID
Quantity
Prijs
CommandResult
CommandParam
Products2Toppings
Product2ToppingsID
ProductID
ToppingID
ISettings
SettingID
SettingName
SettingContent
Orders
OrderID
OrderDateTime
StoreID
PersonID
EmailID
AddressID
PhoneID
Comment
CompanyID
ZakelijkBesteldTegoed
PriveBesteldTegoed
Geannuleerd
Geannuleerd_Memo
OrderBedrag
BetaalMethode
OrderStatus
OrderAcceptTime
Methode
PhoneNumbers
PhoneNumberID
OwnerID
PhoneNumber
verwijderd
PhoneTypeID
Persons
PersonID
Firstname
Surname1
Birthdate
Sexe
aanmelddatum
Restaurants
RestaurantID
ProgramModes
StoreName
BaseStoreID
Addresses
AddressesID
OwnerID
StreetName1
HouseNo
HouseCode
ZipCode
City
Postcodes
postcode
ReeksIndicatie
Huisnrvan
huisnrtot
plaats
straat_lang
StoreID
DeliveryCost
MinOrderCost
FreeDeliveryFrom

Sequentie diagrammen

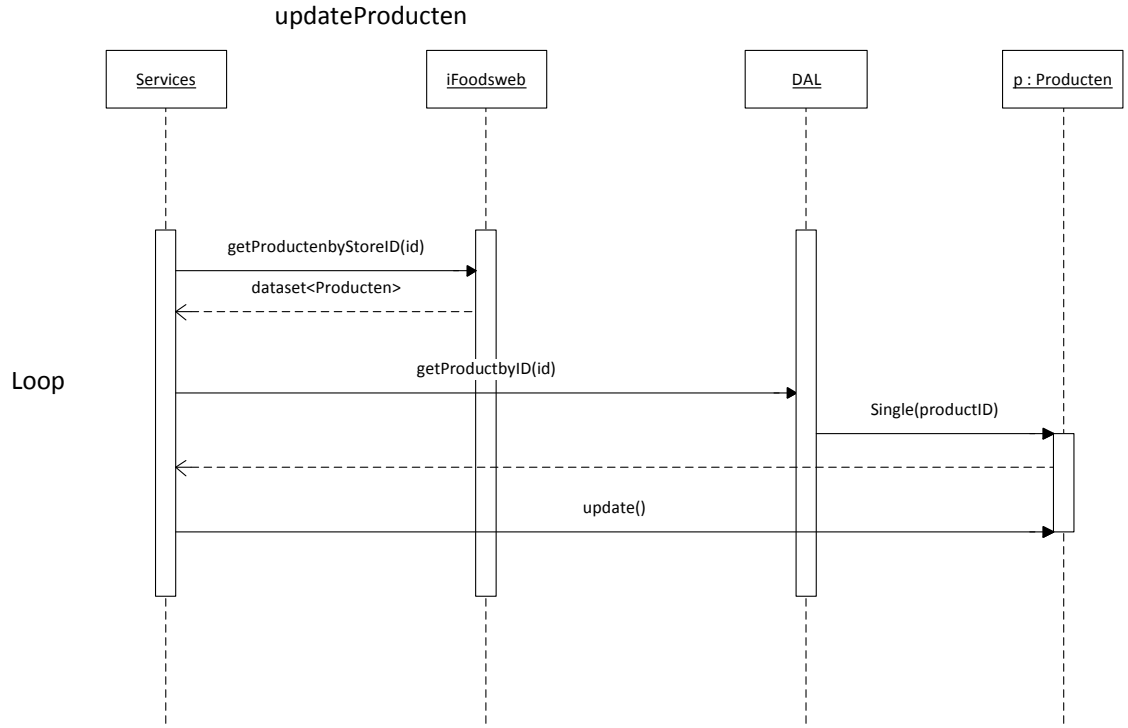
Check License



Bovenstaand diagram hoort bij de bewerking die controleert via de Soap interface of de software nog gelicenceerd is en deze instelling wegschrijft in de lokale database. Het wegschrijven gebeurt door aan de Dal het settingsobject op te vragen genaamd licensed en hier vervolgens de juiste waarde in te verwerken. Na het verwerken wordt het object opgeslagen en is de handeling klaar.

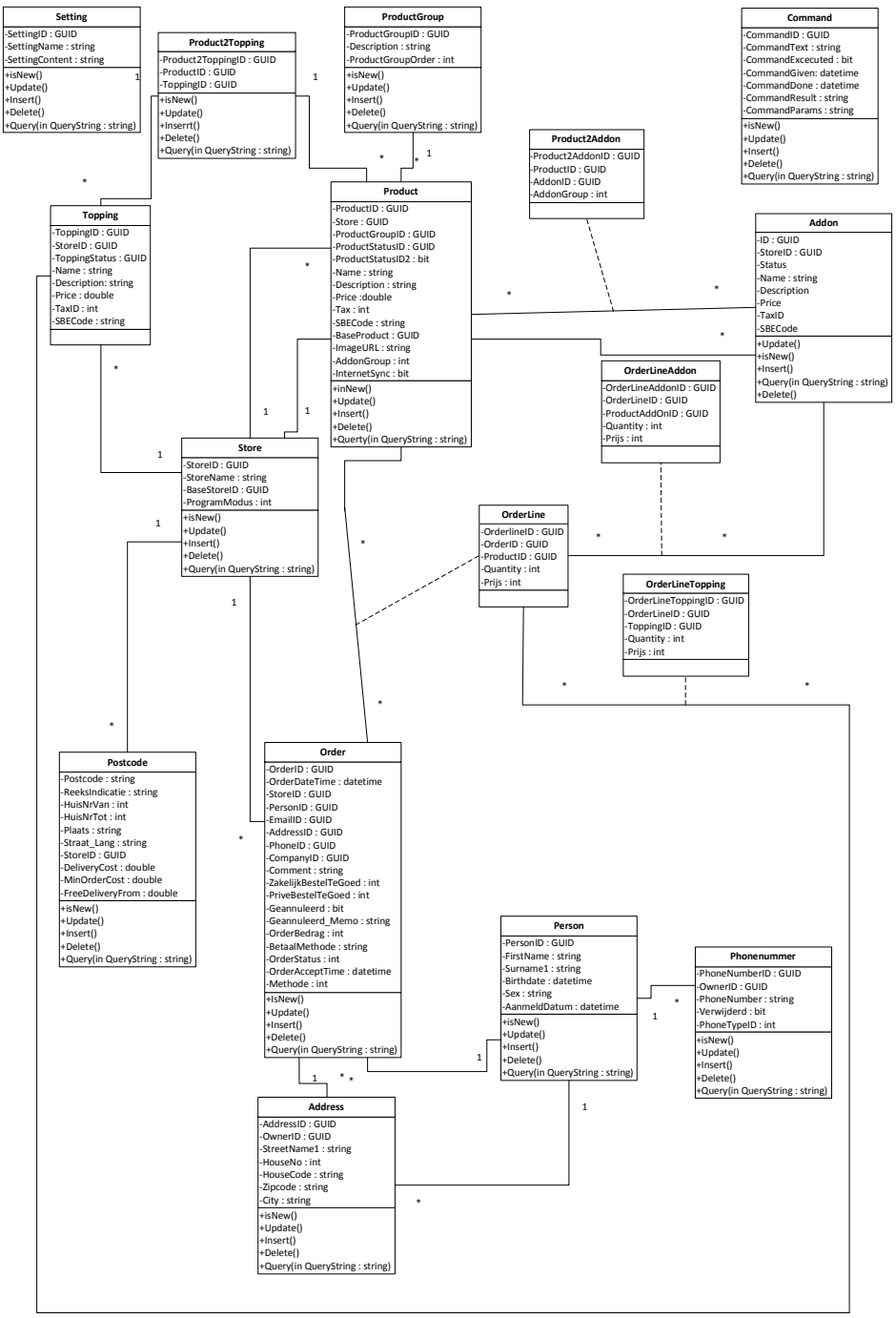


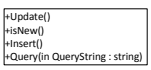
Bovenstaand diagram hoort bij het ophalen van de restaurant gegevens van de gekoppelde restaurants. Eerst worden de laatste gekoppelde restaurants opgehaald aan de hand van het serialnummer. Vervolgens worden de opgehaalde gegevens verwerkt in het lokale opgeslagen object. Daarnaast worden de adres gegevens opgehaald bij Ifoods en vervoglens in de lokale database opgeslagen.



Bovenstaand diagram definieert hoe producten kunnen worden opgehaald aan de kant van ifoods en worden verwerkt in de lokale database. Dit bovenstaande diagram kan ook worden gebruikt voor toppings, addons, product2addons, product2toppings,postcode en productgroups.

Klasse diagram





Full size diagram is bijgevoegd als bijlage 1

B2C4U IN OPDRACHT VAN IFOODS

Elaboratie fase

rapport interface stam gegevens

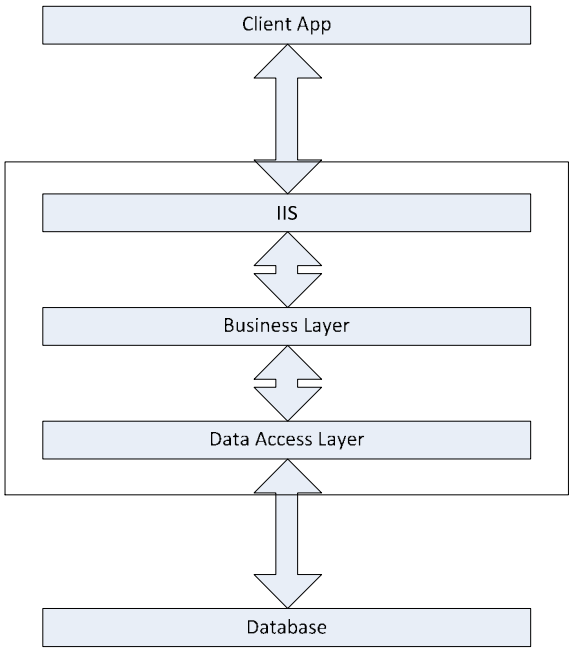
John Renooij

5-1-2012

Inhoud

Systeem architectuur

Hieronder staat de systeem architectuur van de interface

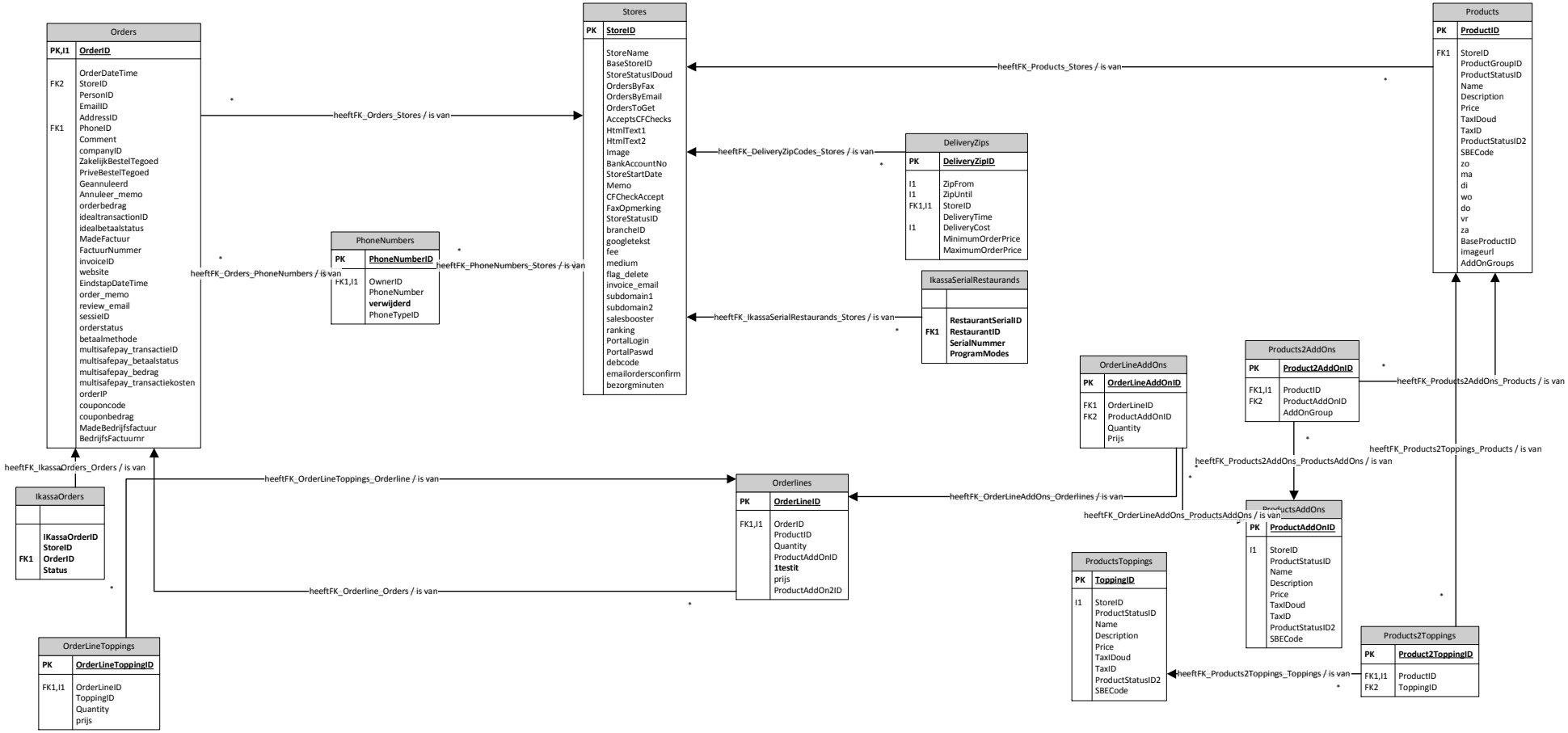


Voor de presentatie van de data naar de client toe zal gebruik worden gemaakt van Internet Information Server (IIS). Deze standaard web server van Windows wordt ingezet voor de overdracht van gegevens en zal primair ook worden gebruikt voor de autenticatie van applicaties. Op deze manier zal de interface niet publiekelijk toegankelijk worden. Ook zal IIS de afhandeling van het SSL verkeer voor zijn rekening nemen zodat data niet toegankelijk is voor mensen die het verkeer zouden onderscheppen.

Voor de database zal gebruik worden gemaakt van de reeds in gebruik zijnde SQL 2008 R2 Server. Deze server draait op het moment van schrijven nog in de gratis Express versie maar zal onder andere ten behoeve van dit (eind) product worden omgezet naar een standaard versie.

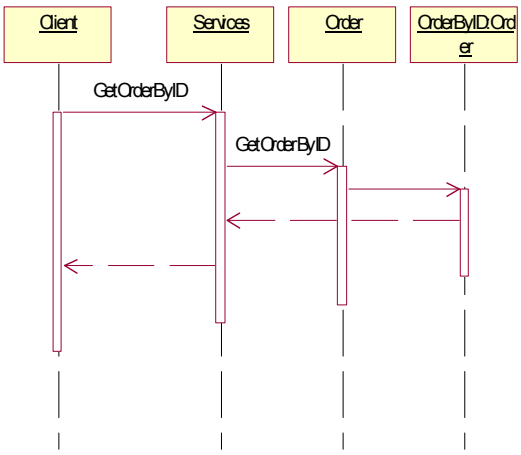
De nieuw te ontwikkelen omgeving zal worden ontwikkeld op het .net 2.0 platform.

ERD

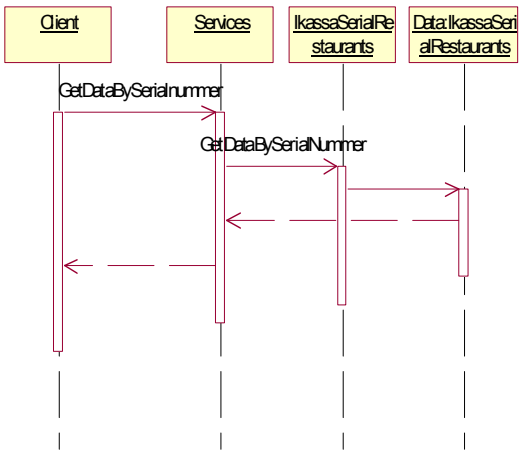


Full size versie is te vinden onder bijlage 2.

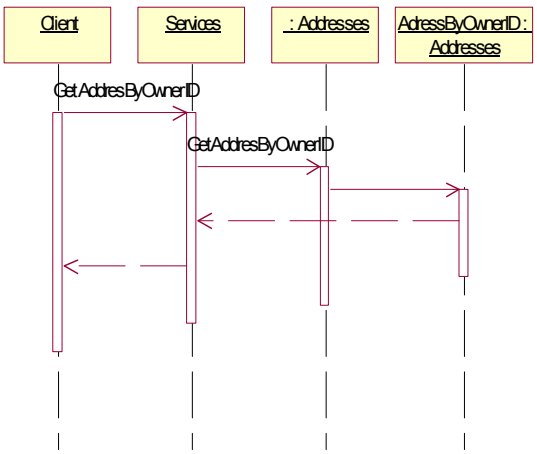
Sequentie diagrammen



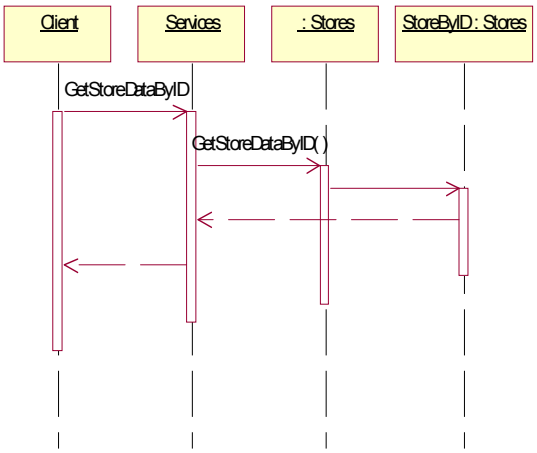
GetOrderByID.



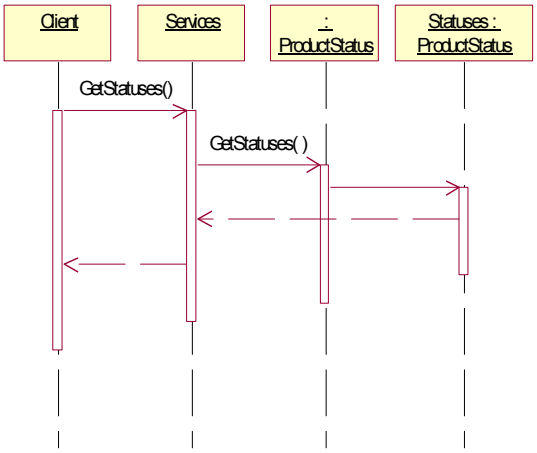
GetDataBySerialNummer



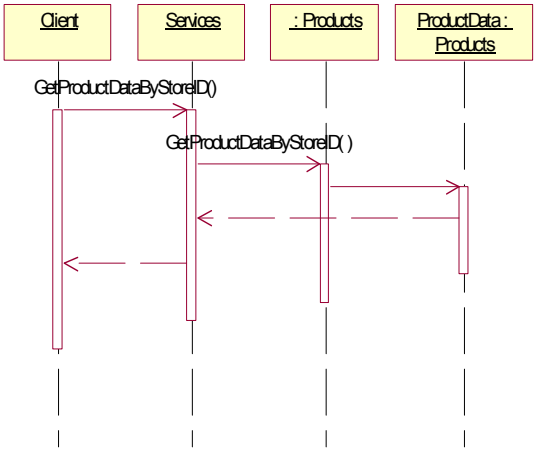
GetAddresByOwnerID



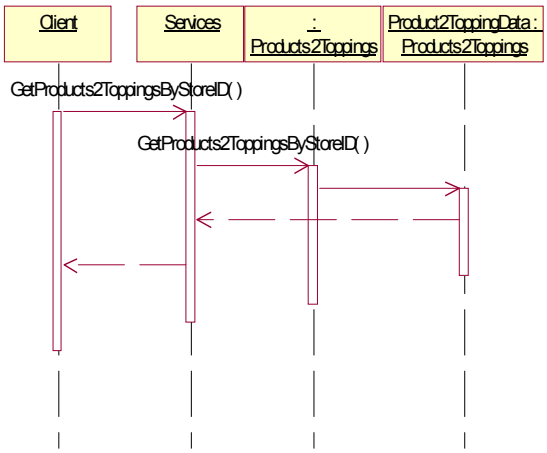
GetStoreDataByID



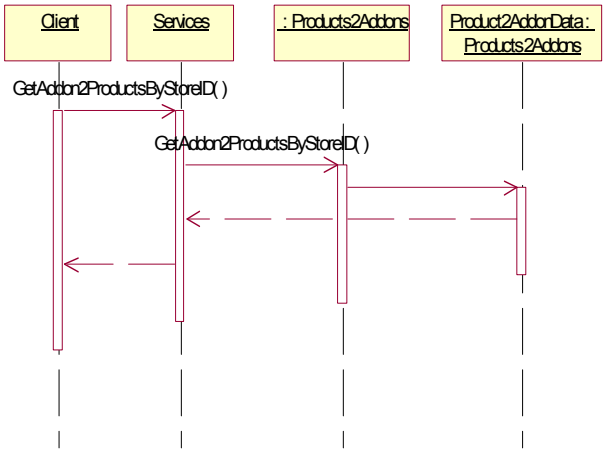
GetStatuses



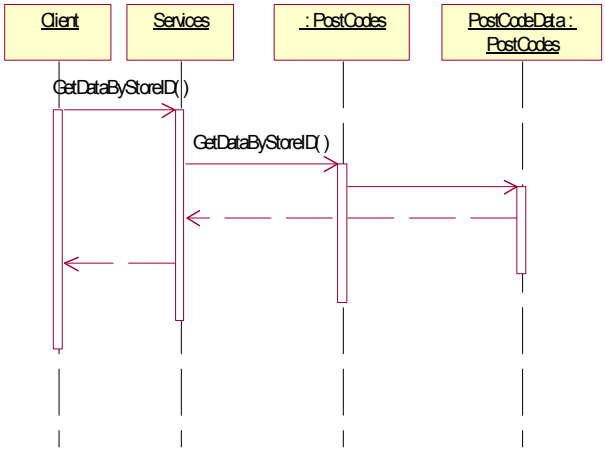
GetProductDataByStoreID



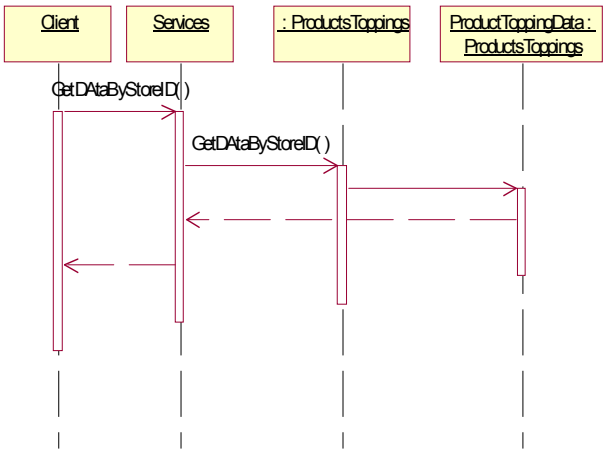
GetProducts2ToppingsByStoreID



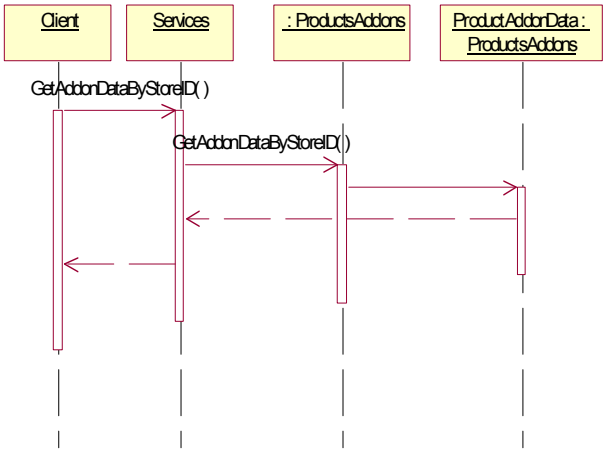
GetProducts2AddonsByStoreID



GetPostcodesByStoreID



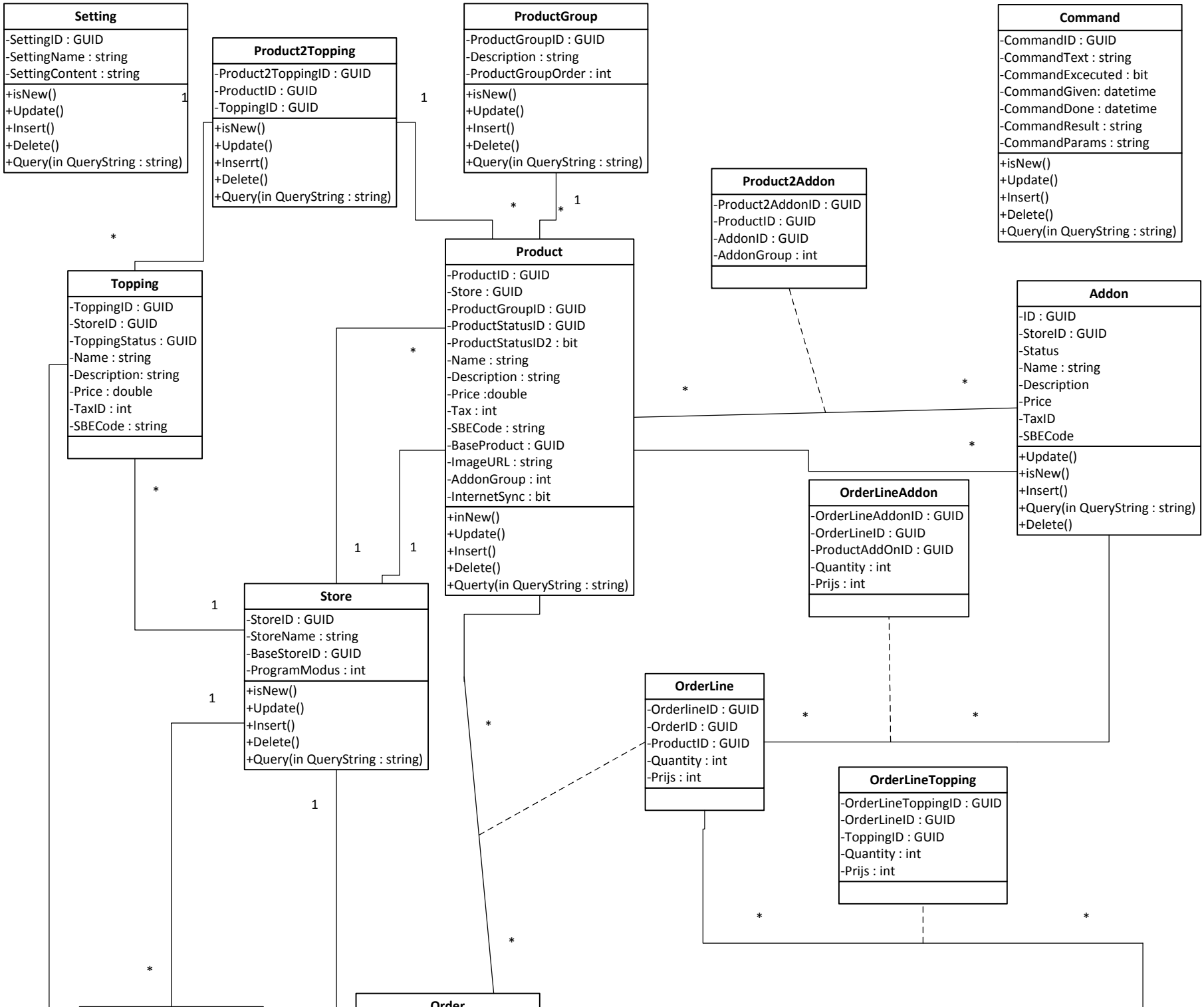
GetProductsToppingDataByStoreID

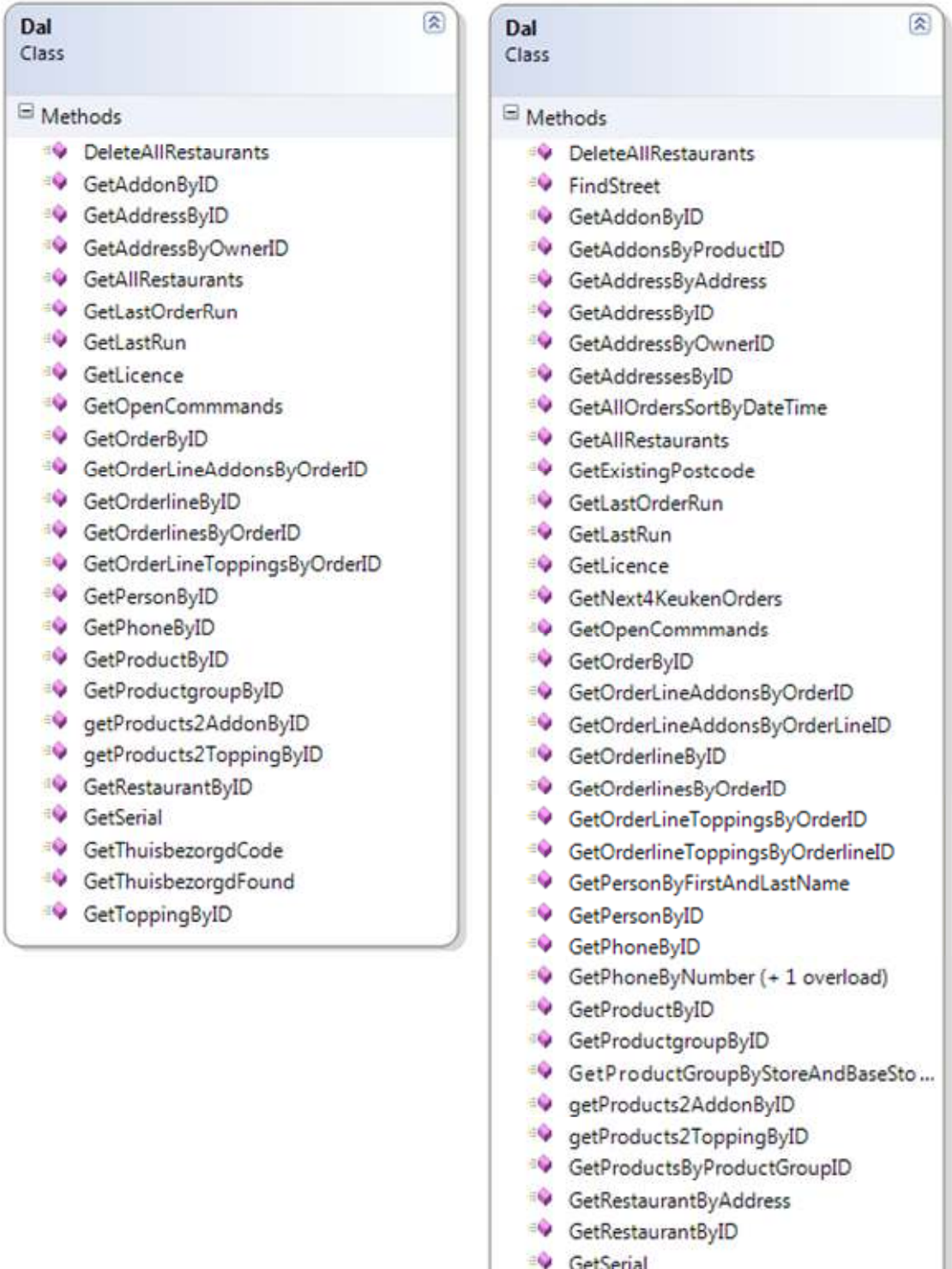


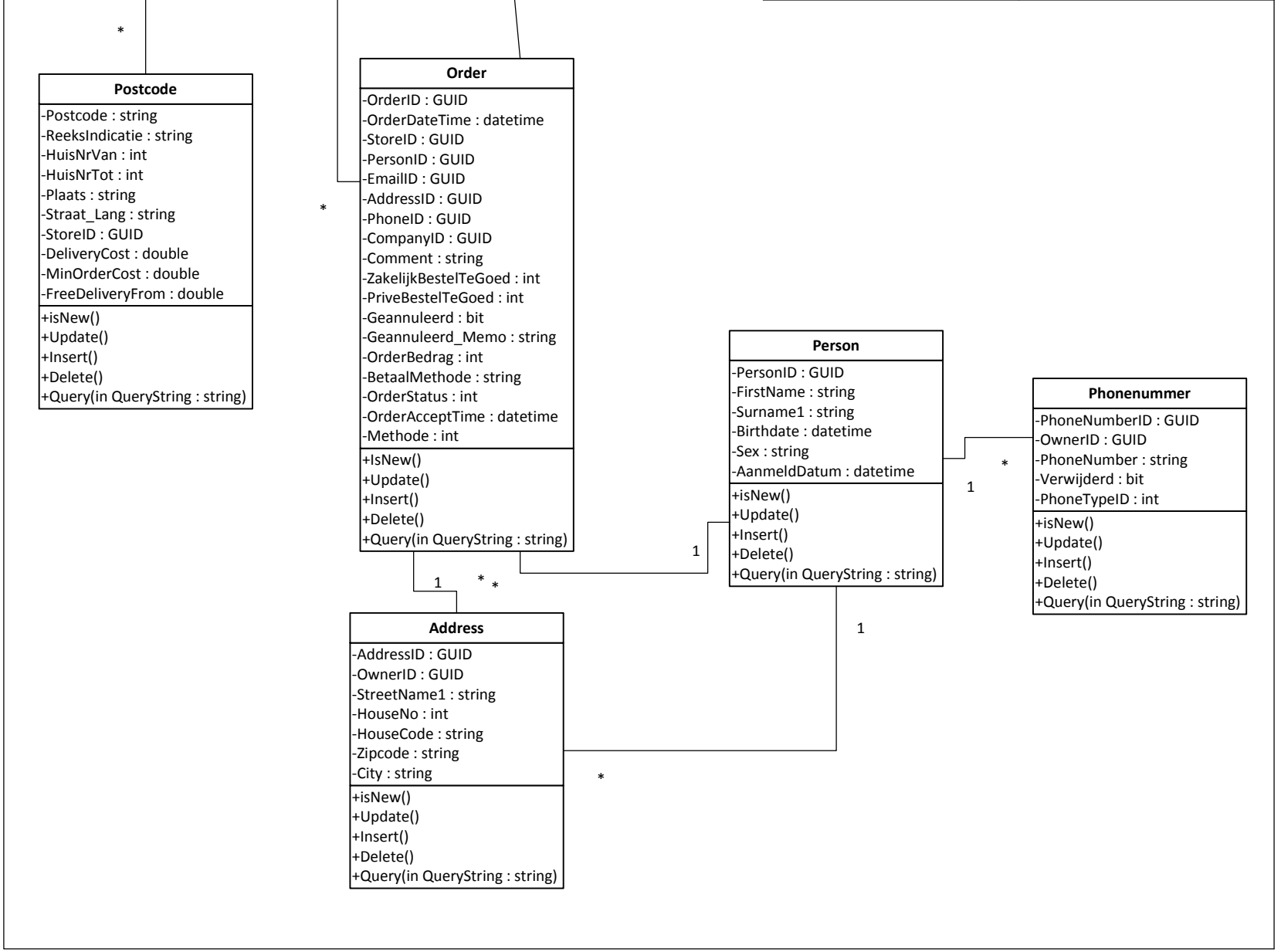
GetProductAddonsByStoreID

Klasse diagram

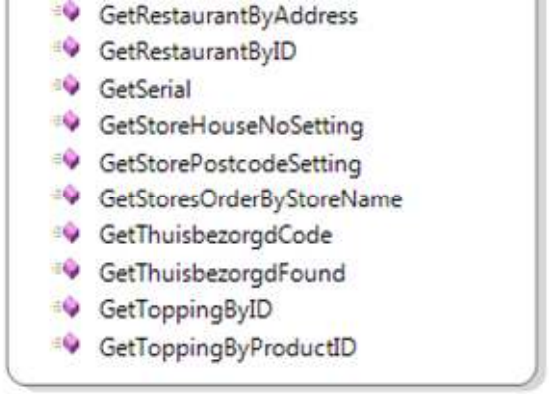
Zie Bijlage 1 voor full size model


Setting
-SettingID : GUID
-SettingName : string
-SettingContent : string
+isNew()
+Update()
+Insert()
+Delete()
+Query(in QueryString : string)
Product2Topping
-Product2ToppingID : GUID
-ProductID : GUID
-ToppingID : GUID
+isNew()
+Update()
+Inserrt()
+Delete()
+Query(in QueryString : string)
ProductGroup
-ProductGroupID : GUID
-Description : string
-ProductGroupOrder : int
+isNew()
+Update()
+Insert()
+Delete()
+Query(in QueryString : string)
Command
-CommandID : GUID
-CommandText : string
-CommandExcecuted : bit
-CommandGiven: datetime
-CommandDone : datetime
-CommandResult : string
-CommandParams : string
+isNew()
+Update()
+Insert()
+Delete()
+Query(in QueryString : string)
Product2Addon
-Product2AddonID : GUID
-ProductID : GUID
-AddonID : GUID
-AddonGroup : int
Topping
-ToppingID : GUID
-StoreID : GUID
-ToppingStatus : GUID
-Name : string
-Description: string
-Price : double
-TaxID : int
-SBECode : string
Product
-ProductID : GUID
-Store : GUID
-ProductGroupID : GUID
-ProductStatusID : GUID
-ProductStatusID2 : bit
-Name : string
-Description : string
-Price :double
-Tax : int
-SBECode : string
-BaseProduct : GUID
-ImageURL : string
-AddonGroup : int
-InternetSync : bit
+inNew()
+Update()
+Insert()
+Delete()
+Querty(in QueryString : string)
Addon
-ID : GUID
-StoreID : GUID
-Status
-Name : string
-Description
-Price
-TaxID
-SBECode
+Update()
+isNew()
+Insert()
+Query(in QueryString : string)
+Delete()
OrderLineAddon
-OrderLineAddonID : GUID
-OrderLineID : GUID
-ProductAddOnID : GUID
-Quantity : int
-Prijs : int
Store
-StoreID : GUID
-StoreName : string
-BaseStoreID : GUID
-ProgramModus : int
+isNew()
+Update()
+Insert()
+Delete()
+Query(in QueryString : string)
OrderLine
-OrderlineID : GUID
-OrderID : GUID
-ProductID : GUID
-Quantity : int
-Prijs : int
OrderLineTopping
-OrderLineToppingID : GUID
-OrderLineID : GUID
-ToppingID : GUID
-Quantity : int
-Prijs : int
1
*


Dal
Class
Methods
DeleteAllRestaurants
GetAddonByID
GetAddressByID
GetAddressByOwnerID
GetAllRestaurants
GetLastOrderRun
GetLastRun
GetLicence
GetOpenCommmands
GetOrderByID
GetOrderLineAddonsByOrderID
GetOrderlineByID
GetOrderlinesByOrderID
GetOrderLineToppingsByOrderID
GetPersonByID
GetPhoneByID
GetProductByID
GetProductgroupByID
getProducts2AddonByID
getProducts2ToppingByID
GetRestaurantByID
GetSerial
GetThuisbezorgdCode
GetThuisbezorgdFound
GetToppingByID
Dal
Class
Methods
DeleteAllRestaurants
FindStreet
GetAddonByID
GetAddonsByProductID
GetAddressByAddress
GetAddressByID
GetAddressByOwnerID
GetAddressesByID
GetAllOrdersSortByDateTime
GetAllRestaurants
GetExistingPostcode
GetLastOrderRun
GetLastRun
GetLicence
GetNext4KeukenOrders
GetOpenCommmands
GetOrderByID
GetOrderLineAddonsByOrderID
GetOrderLineAddonsByOrderLineID
GetOrderlineByID
GetOrderlinesByOrderID
GetOrderLineToppingsByOrderID
GetOrderlineToppingsByOrderlineID
GetPersonByFirstAndLastName
GetPersonByID
GetPhoneByID
GetPhoneByNumber (+ 1 overload)
GetProductByID
GetProductgroupByID
GetProductGroupByStoreAndBaseSto ...
getProducts2AddonByID
getProducts2ToppingByID
GetProductsByProductGroupID
GetRestaurantByAddress
GetRestaurantByID
GetSerial



+Update()
+isNew()
+Insert()
+Query(in QueryString : string)


GetRestaurantByAddress
GetRestaurantByID
GetSerial
GetStoreHouseNoSetting
GetStorePostcodeSetting
GetStoresOrderByStoreName
GetThuisbezorgdCode
GetThuisbezorgdFound
GetToppingByID
GetToppingByProductID

Inception fase rapport

Background service recieve data

John Renooij

1-1-2012

Inhoud

1. Vision

De Kassa-applicatie is voor een groot gedeelte afhankelijk van externe data. Onder deze data is te verstaan bijvoorbeeld de orders van verschillende partijen maar ook bijvoorbeeld of de kassa-applicatie nog wel een geldig licentie heeft.

Daarnaast is het de bedoeling dat de background services productinformatie van een externe partij gaat ophalen. Hierdoor zou de kassa-applicatie gelijk gebruikklaar zijn, mits het een reeds bekend restaurant is.

2. Requirements

2.1 Globale requirements

De services dient met 3 diensten samen te werken namelijk:

- Ifoods
- Just Eat
- Thuisbezorgd

Just Eat en Thuisbezorgd zullen enkel worden gebruikt voor het ontvangen van orders. Ifoods zal naast het aanbieden van orders ook dienst doen als primaire bron van product informatie.

2.2 Gedetailleerde beschrijving requirements

2.2.1 Ifoods

Ifoods heeft een interface beschikbaar gesteld (eerdere ontwikkeling binnen het zelfde project) die de mogelijkheid heeft om orders op te halen die via diverse andere kanalen zijn ontvangen. Daarnaast biedt dezelfde interface de optie om product informatie op te halen die reeds opgeslagen is in de database van Ifoods.

2.2.2 Just Eat

Just Eat heeft geen mogelijkheden om orders aan te bieden voor een kassa systeem. Just Eat heeft op dit moment twee manieren om een restauranthouder te informeren over een eventuele nieuwe orders. De eerste optie is via een sms waarna de restauranthouder zelf op een webportal moet kijken welke bestelling er binnen gekomen is. De tweede optie, welke tevens de optie is die wij gebruiken bij het inlezen van de orders, is een optie waarbij de bestelling binnen komt per email. Deze email moet worden ontrafeld en als bestelling worden opgeslagen.

2.2.3 Thuisbezorgd

Thuisbezorgd levert geen mogelijkheid om rechtstreeks te communiceren. Ze bieden een interface in de vorm van een losse applicatie. Deze applicatie communiceert met andere software op basis van gedocumenteerde XML bestanden. Deze bestanden, die op een vaste plaats op de lokale computer opgeslagen worden, kunnen worden gebruikt om bestelling informatie te verwerken.

3. Use Cases & Use Case Scenarios

Het systeem heeft geen user interface of een optie voor gebruikers om rechtstreeks met het systeem te communiceren.

Er zijn echter wel mogelijkheden voor gebruikers om de services commando's uit te laten voeren. Dit zal indirect gaan doormiddel van commando's die vanuit het Kassa-systeem worden doorgestuurd. Hierbij moet men denken aan bijvoorbeeld het ophalen van productinformatie van de server van Ifoods of het ophalen van een nieuw restaurant.

4 Project plan

Het project zal worden uitgevoerd zoals vermeld in het plan van aanpak. Wel zullen de verschillende diensten verschillend worden ontwikkeld. Er wordt begonnen met het implementeren van een koppeling met Ifoods. Daarna zal de koppeling met thuisbezorgd worden geïmplementeerd en daarna de koppeling met Just Eat.

5 Prototypen

Deze applicatie heeft geen interface naar de gebruiker toe. Daarom zijn er ook geen prototypes gemaakt.

B2C4U IN OPDARCHT VAN IFOODS

Inception fase rapport

Kassa applicatie

John Renooij

1-1-2012

Inhoud

1. Vision

Het doel van deze applicatie is het tonen en invoeren van bestellingen. Dit kunnen bestellingen zijn die worden aangenomen aan de balie van een restaurant maar bijvoorbeeld ook bestellingen die zijn binnen gekomen via Just Eat en Ifoods.

Daarnaast moet de kassa applicatie de mogelijkheid bieden om basisinstelling zoals openingstijden te veranderen. Ook moet de applicatie de mogelijkheid bieden om producten en dergelijk aan te passen.

Om veelzijdig inzetbaar te zijn moet de applicatie ook verschillende overzichtschermen tonen met detail informatie afgestemd op de vraag van de gebruiker. Deze overzichten zijn bijvoorbeeld voor de kassa medewerker een overzicht van openstaande bestellingen maar voor de keukenmedewerker een detail overzicht van de eerste 4 openstaande bestellingen.

2. Requirements

De applicatie moet verschillende informatie tonen met betrekking tot bestellingen en de mogelijkheid bieden om bestellingen in te voeren.

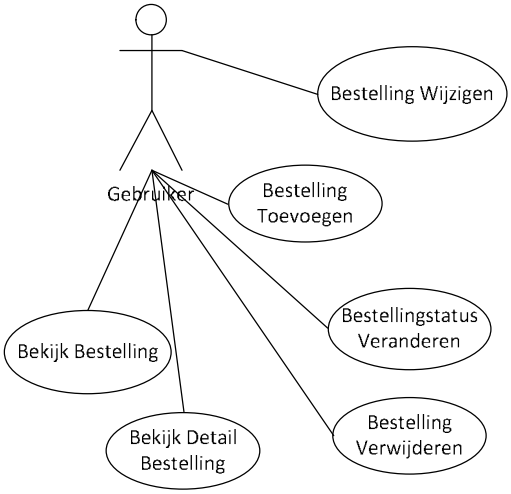
Er zijn twee verschillende rollen in het systeem aanwezig. De eerste rol, gebruiker, kan informatie van bestellingen inzien en / of een bestelling invoeren. Afhankelijk van de informatie behoefde van de gebruiker kan er gekozen worden uit de volgende overzichten:

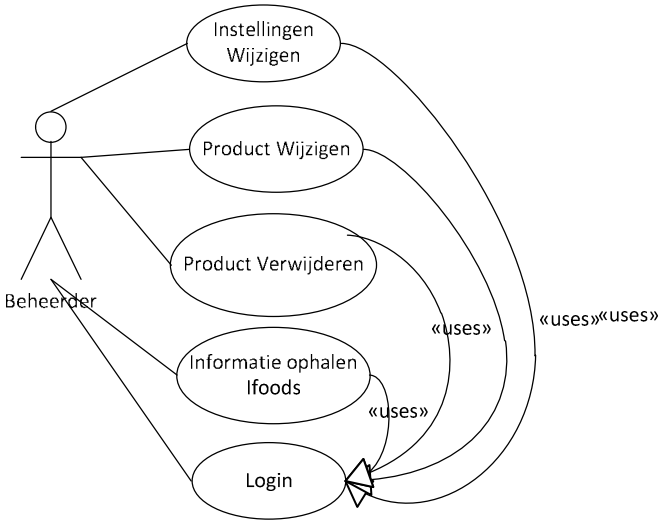
- Kassaoverzicht: Dit overzicht toont de een overzicht van alle openstaande orders en hun status. Bij het openen van een bestelling zal de bestelling in detail worden getoond.
- Keukenoverzicht: Dit overzicht toont de eerste 4 bestellingen met de status keuken in een detail overzicht.
- Keukenoverzicht 2: Dit overzicht toont alle bestellingen met de status keuken met de mogelijkheid om van 1 bestelling de details te bekijken.
- Afleveroverzicht: Dit overzicht toont alle bestellingen met de status “klaar voor transport” gegroepeerd op basis van tijd en postcode.
- Bestelling overzicht: Dit overzicht toont alle bestellingen. Dit overzicht kan gefilterd worden op diverse data zoals datum, klant, postcode en / of telefoonnummer. Van deze bestellingen kan per bestelling de details worden opgevraagd.

Verder moet de gebruiker de mogelijkheid hebben een nieuwe bestelling aan te nemen. Dit kan een bestelling zijn die wordt afgeleverd maar kan ook een bestelling zijn die wordt opgehaald aan de balie.

De tweede rol is de rol van “beheerder”. De beheerder kan wijzigingen aanbrengen in de instellingen van de applicatie. Daarnaast kan de “beheerder” wijzigingen aanbrengen in producten.

3. Use Cases & Use Case Scenarios





Naam	Bekijk Bestellingen
Samenvatting	Via deze functie kan de gebruiker een overzicht krijgen met daarin alle openstaande bestellingen.
Actoren	Gebruiker
Aanname	De gebruiker is op dit moment niet bezig met het invoeren van een bestelling of als beheerder instellingen aan het wijzingen.
Beschrijving	1. Om een overzicht te krijgen van alle order die nog niet zijn afgerond , kan deze functie gebruikt worden. Deze functie geeft een overzicht met beperkte gegevens.
Uitzonderingen	Er zijn geen uitzonderingen
Resultaat	Overzicht met niet afgeronde bestellingen

Naam	Bekijk bestelling
Samenvatting	In het overzicht van Bekijk bestellingen heeft de gebruiker een bestelling aangeklikt om hiervan de details te bekijken. Het systeem toont deze details
Actoren	Gebruiker
Aanname	De gebruiker heeft eerst een overzicht opgevraagd via bekijk bestellingen
Beschrijving	1. De gebruiker klikt op een bestelling 2. Het systeem opent een nieuw scherm en toont detail informatie omtrent de bestelling
Uitzonderingen	Er zijn geen uitzonderingen
Resultaat	Een detailscherm met informatie over de gekozen bestelling

Naam	Bestelling toevoegen
Samenvatting	Deze functie geeft de gebruiker de mogelijkheid om een bestelling in te voeren.
Actoren	Gebruiker
Aanname	De gebruiker heeft momenteel geen andere detailschermen open staan en staat in het hoofdscherm
Beschrijving	1. De gebruiker kiest voor de button Bestelling. 2. In het scherm vult de gebruiker de benodigde gegevens zoals afleveradres en naam en kiest voor bestellen of kiest voor de optie counter order 3. In het nieuwe scherm kan de gebruiker producten selecteren. 4. Na de selectie kan de gebruiker keuze maken voor toppingen en toevoegingen 5. Na de selectie van producten en topping en toevoegingen kan de bestelling worden afgerond
Uitzonderingen	Bij het opgeven van een foutief adres of een adres waar het restaurant niet op levert, toont het systeem een foutmelding
Resultaat	Een nieuw ingevoerde bestelling

Naam	Update Bestel status
Samenvatting	Om het proces te stroomlijnen kan de gebruiker de status van de bestelling bijwerken
Actoren	Gebruiker
Aanname	De gebruiker heeft via “Bekijk bestelling” een bestelling geopend
Beschrijving	1. De gebruiker openend een bestelling. 2. De gebruiker kiest de juiste statas button.
Uitzonderingen	Er zijn geen uitzonderingen
Resultaat	Het systeem update de status van de bestelling

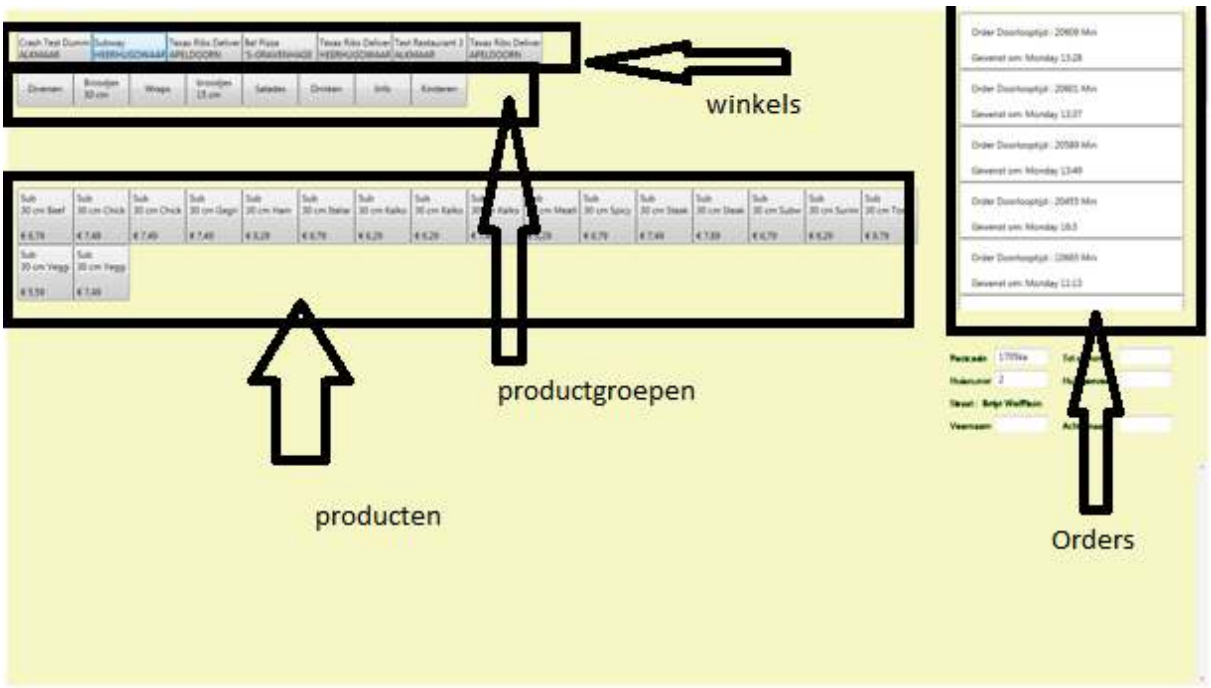
Naam	Bestelling Wijzigen
Samenvatting	De gebruiker kan een bestelling wijzigen die eerder al ingevoerd was
Actoren	Gebruiker
Aanname	De gebruiker heeft via “Bekijk bestelling” een bestelling geopend
Beschrijving	1. De gebruiker openend een bestelling. 2. De gebruiker kiest voor de optie wijzigen. 3. De gebruiker wijzigt de bestelling. 4. De gebruiker rond bestelling af.
Uitzonderingen	Er zijn geen uitzonderingen
Resultaat	Het systeem update de status van de bestelling

Naam	Login
Samenvatting	De gebruiker kan zich aanmelden als beheerder
Actoren	Gebruiker
Aanname	De gebruiker is verder geen andere handelingen aan het verrichten en staat in het hoofdscherm
Beschrijving	1. De gebruiker drukt op de knop om het beheer scherm te open 2. Het systeem vraagt om een pincode 3. Het systeem valideert de invoer van pincode 4. Bij goede code wordt het beheerscherm getoond. Bij foutieve code zal er een uitzondering optreden.
Uitzonderingen	Bij een foutieve pincode toont een systeemmelding. Na 3 foutieve pogingen zal er een waarschuwingsmail worden verzonden.
Resultaat	Het systeem opent het beheerscherm

Naam	Instellingen wijzingen
Samenvatting	De beheerder kan systeem en applicatie instellingen wijzigen
Actoren	Beheerder
Aanname	De beheerder is ingelogd dmv “Login”
Beschrijving	1. De beheerder opent het betreffende scherm. 2. De beheerder veranderd 1 of meer instellingen. 3. De beheerder klikt op de opslaan knop
Uitzonderingen	Bij foutieve invoer zal het systeem een gepaste melding tonen
Resultaat	Veranderde gegevens.

Naam	Gegevens ophalen bij Ifoods
Samenvatting	De beheerder kan product informatie laten synchroniseren met Ifoods
Actoren	Beheerder
Aanname	De beheerder is ingelogd dmv “Login”
Beschrijving	1. De beheerder opent de detail info van een restaurant 2. De beheerder klikt op de knop gegevens synchroniseren 3. Het systeem verwerkt de aanvraag in het systeem
Uitzonderingen	Er zijn geen uitzonderingen
Resultaat	Een Syncronisatie verzoek in de database (welke wordt verwerkt door de services)

4 Prototypen



Figuur 1 Eerste prototype basis schermlayout

Figuur 2 Eerste opzet van scherm componenten

Inception fase rapport

Interface Stam gegevens

John Renooij

1-1-2012

Inhoud

Vision

Om het mogelijk te maken voor programma's om op een gecontroleerde manier data uit de database van Ifoods te krijgen is het nodig om een interface te ontwikkelen. Omdat de interface generiek in gebruik dient te zijn en ook evt voor andere doeleinde gebruikt dient te kunnen worden is het van belang dat er een interface structuur gemaakt wordt, die niet alleen door Ifoods te gebruiken is.

Uit vooronderzoek is gebleken dat niet mogelijk c.q. wenselijk is om af te wijken van de huidige omgevingen omdat het omschakelen zou gevaar kan betekenen voor bestaande systemen en / of de communicatie tussen deze systemen.

Requirements

Globale requirements

De requirements binnen dit project bepalen precies wat het systeem moet kunnen. Er wordt dus niet besproken wat het niet moet kunnen. Dit zal in de afbakening van het project beschreven worden. De requirements zijn op basis van de opdrachtomschrijving ontstaan.

De requirements zijn hieronder globaal beschreven en zullen daarna in detail worden uitgediept. De globale requirements zijn als volgt:

- Het ophalen van licentie gegevens
- Het ophalen van restaurant gegevens bij een licentie
- Het ophalen van basis gegevens van een restaurant
- Het ophalen van adres gegevens van een restaurant
- Het ophalen van een orders aangeleverd via ifoods.nl e.a.
- Het ophalen van Product gegevens, Product toevoegingen en Product bijproducten
- Het ophalen van een of meerder bezorggebieden van een restaurant

Gedetailleerde beschrijving requirements

De volgende requirements zijn in globaal bespoken en zullen hieronder verder beschreven worden. Per requirements zal er in detail besproken worden wat een requirements moet inhouden.

Het ophalen van Licentie gegevens. (serienummer)

Deze aanroep maakt het mogelijk om te controleren of een systeem nog een geldige licentie heeft en dus mag opstarten.

Bij het ophalen van licentie gegevens zal het systeem “boolean” terug geven om aan te geven of een systeem wel of niet een geldige / actieve licentie heeft. De vereiste input is “string” met daarin een code bestaand uit 12 cijfers. Bij het niet opgeven van een lege “string” zal het systeem reageren met een “False”

Het ophalen van restaurant gegevens bij een licentie (serienummer)

Deze aanroep maakt het mogelijk om de restaurants op te halen die bij een serienummer horen. Daarnaast wordt ook aangegeven in welke modus de software moet functioneren. Dit is belangrijk voor de afhandeling van orders en de richting van berichten verkeer.

Bij het ophalen van restaurant gegevens zal het systeem de volgende informatie terug geven voor elk restaurant wat aan deze serienummer is gekoppeld:

RestaurantID
Modus
DatabaseID

Indien er geen (geldig) licentie nummer wordt opgegeven zal het systeem reageren met een leeg antwoord.

Het ophalen van basis gegevens van een restaurant (RestaurantID)

Deze aanroep maakt het mogelijk om op basis van een restaurantid alle basis gegevens te ontvangen.

Bij het ophalen van de gegevens zal het systeem de volgende informatie teruggeven:

StoreID	Image	brancheID
StoreName	BankAccountNo	googletekst
BaseStoreID	StoreStartDate	fee
StoreStatusIDoud	Memo	medium
OrdersByFax	CFCheckAccept	flag_delete
OrdersByEmail	FaxOpmerking	invoice_email
OrdersToGet	StoreStatusID	ranking
AcceptsCFChecks	subdomain1	PortalLogin
HtmlText1	subdomain2	PortalPaswd
HtmlText2	emailordersconfirm	debcode
salesbooster	bezorgminuten	

Indien er geen (geldig) restaurantid wordt opgegeven zal het systeem reageren met een leeg antwoord.

Het ophalen van adres gegevens van een restaurant (RestaurantID)

Deze aanroep maakt het mogelijk om op vestigingsgegevens van een restaurant op te halen.

Bij het ophalen van de gegevens zal het systeem de volgende informatie teruggeven:

AddressID	HouseCode	bedrijfsnaam
OwnerID	Zipcode	kostenplaats
StreetName1	City	flag_delete
HouseNo	verwijderd	date_address_inserted

Indien er geen (geldig) restaurantid wordt opgegeven zal het systeem reageren met een leeg antwoord.

Het ophalen van een orders aangeleverd via ifoods.nl (RestaurantID) e.a.

Deze aanroep maakt het mogelijk om orders die op een andere manier binnen zijn gekomen (bijvoorbeeld via Ifoods.nl, de eigen website of via een callcenter) te ontvangen. Deze aanroep bevat niet de orders zelf maar enkel de orderID's. Aan de hand van deze ID's kan de order worden opgehaald.

Bij het ophalen van de gegevens zal het systeem de volgende informatie teruggeven:

StoreID
OrderID
Status

Indien er geen (geldig) restaurantid wordt opgegeven zal het systeem reageren met een leeg antwoord.

Het ophalen van Product gegevens, Product toevoegingen en Product bijproducten (restaurantid)

Deze serrie van oproepen maakt het mogelijk om de productgegevens zoals deze in de database van Ifoods staan te ontvangen.

De serie van oproepen bestaat uit:

Het ophalen van producten
Het ophalen van bijproducten
Het ophalen van producttopping
Het ophalen van de koppelingsgegevens tussen product en bijproduct
het opahalen van de koppelingsgegevens tussen product en producttopping

Het ophalen van een of meerder bezorggebieden van een restaurant (RestaurantID)

Deze aanroep maakt het mogelijk om voor een gegeven restaurant het postcode boek op te halen met daarbij horende gegevens zoals bezorgkosten, minimum order bedrag en reistijd.

Voor elke bekende straat binnen het bezorggebied zal het systeem de volgende informatie teruggeven:

Huinrvan
Huisntot
plaats
postcode
reeksindicatie
straat_lang
deliverycost
maximumorderprice
minimimorderprice

Indien er geen (geldig) restaurantid wordt opgegeven zal het systeem reageren met een leeg antwoord.

Use Cases & Use Case Scenarios

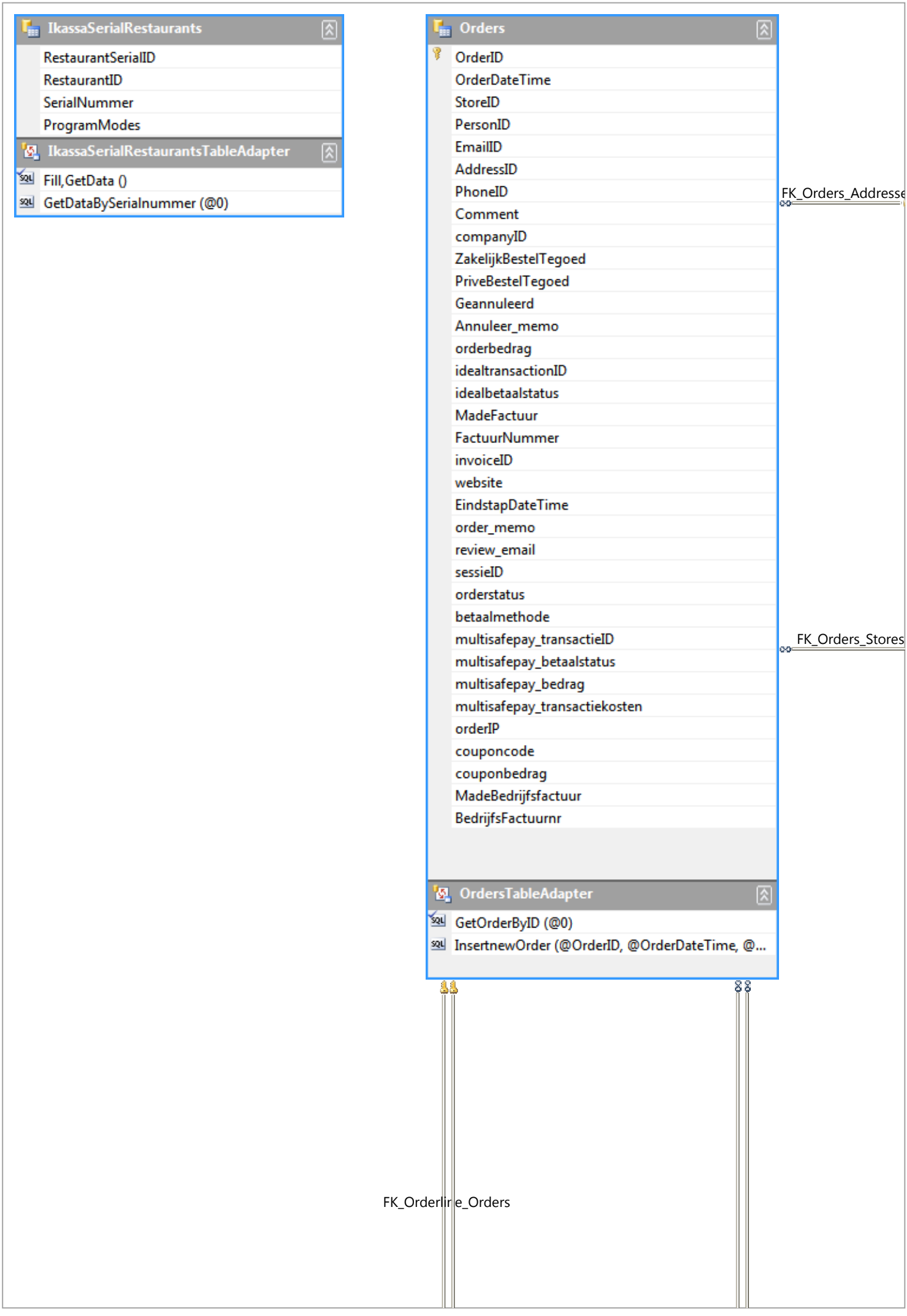
Binnen dit system is er geen interactie tussen system en gebruiker. Er zijn dus geen Use cases of Scenarios te omschrijven

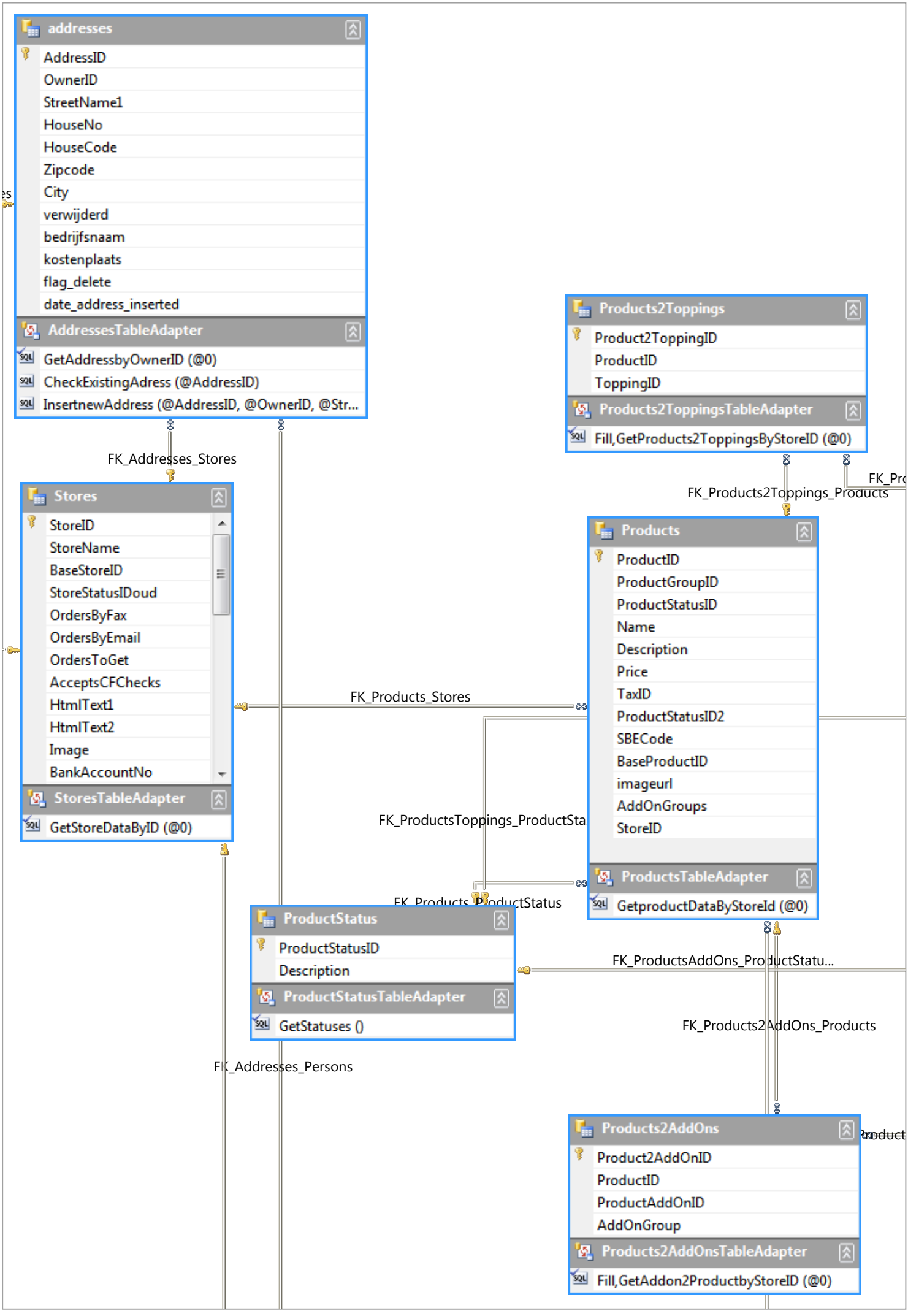
Project plan

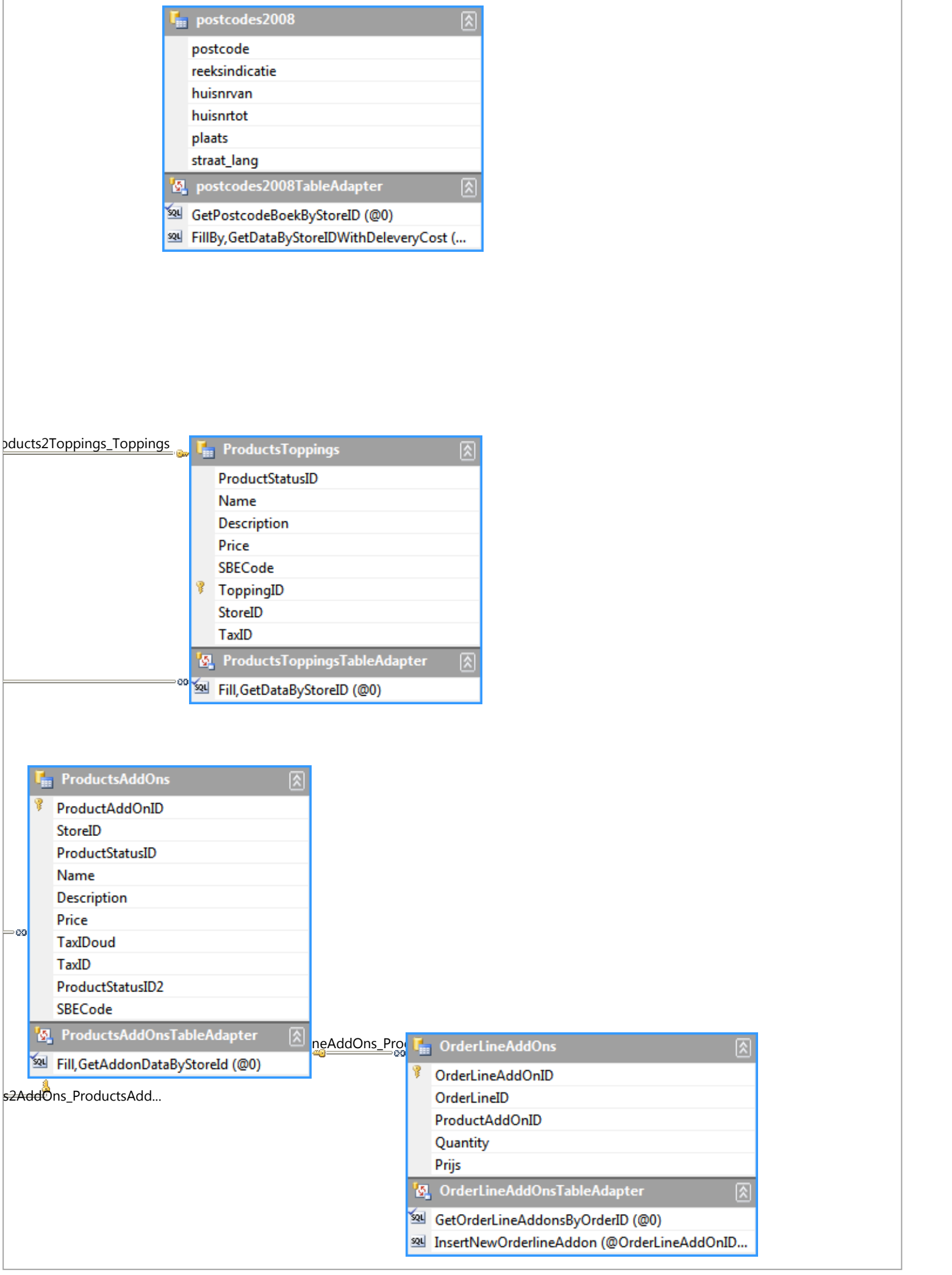
Het project zal worden uitgevoerd zoals het globale plan van aanpak

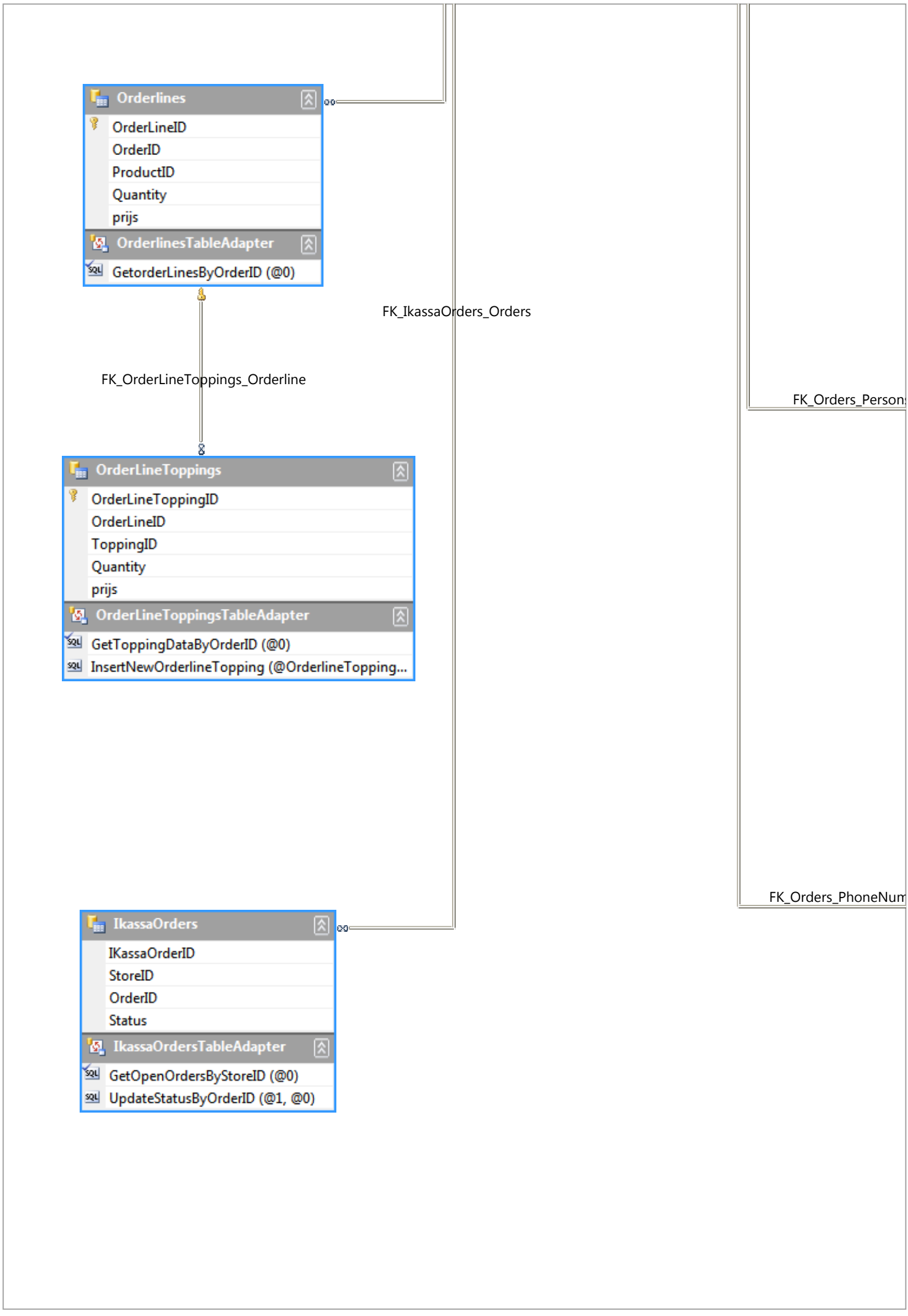
Prototypen

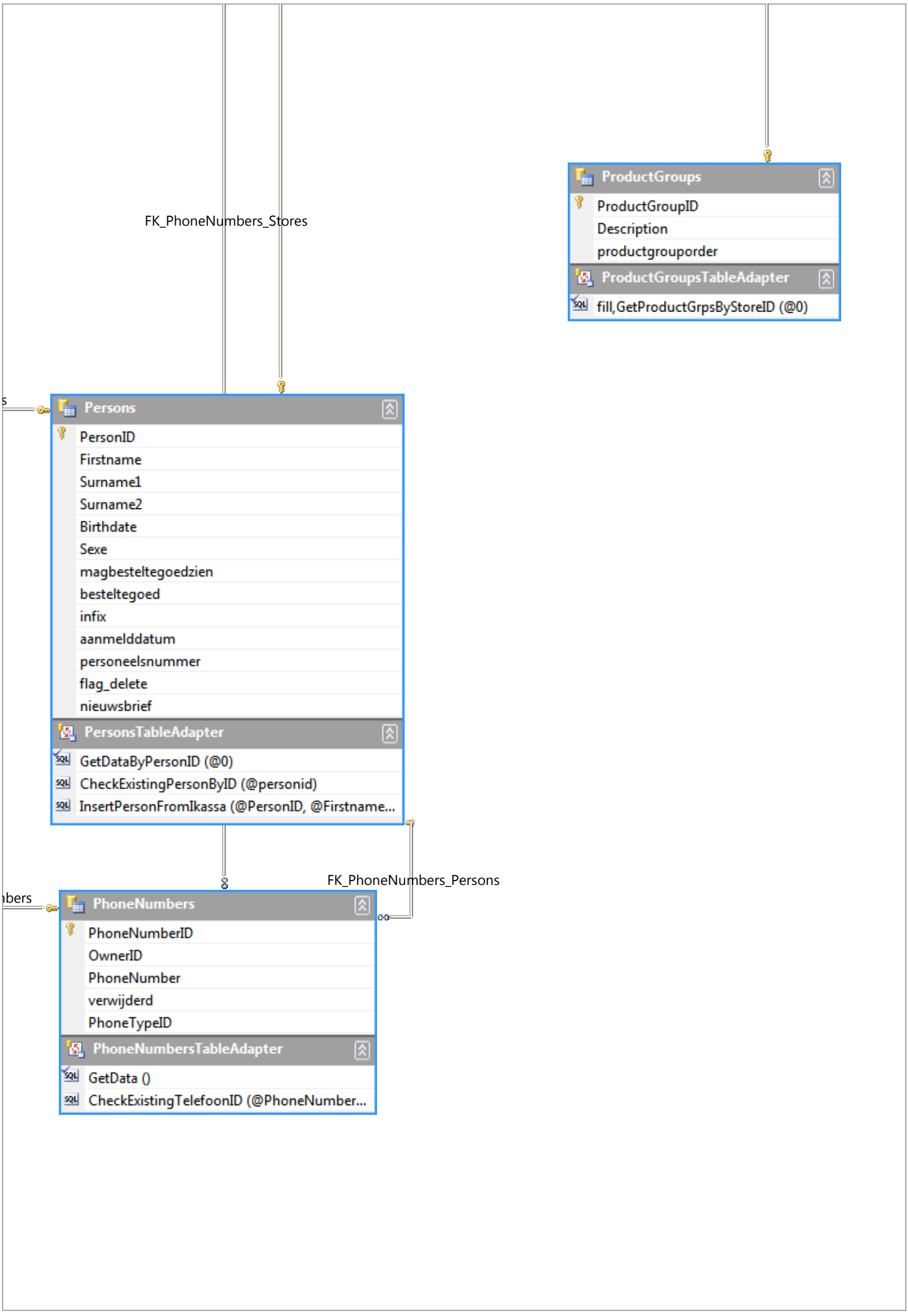
Er is geen interactie met de gebruiker. Daarom zijn er geen interface prototypes gemaakt


IkassaSerialRestaurants
RestaurantSerialID
RestaurantID
SerialNummer
ProgramModes
IkassaSerialRestaurantsTableAdapter
Fill,GetData ()
GetDataBySerialnummer (@0)
Orders
OrderID
OrderDateTime
StoreID
PersonID
EmailID
AddressID
PhoneID
Comment
companyID
ZakelijkBestelTegoed
PriveBestelTegoed
Geannuleerd
Annuleer_memo
orderbedrag
idealtransactionID
idealbetaalstatus
MadeFactuur
FactuurNummer
invoiceID
website
EindstapDateTime
order_memo
review_email
sessieID
orderstatus
betaalmethode
multisafepay_transactieID
multisafepay_betaalstatus
multisafepay_bedrag
multisafepay_transactiekosten
orderIP
couponcode
couponbedrag
MadeBedrijfsfactuur
BedrijfsFactuurnr
OrdersTableAdapter
GetOrderByID (@0)
InsertnewOrder (@OrderID, @OrderDateTime, @...
FK_Orders_Addresse
FK_Orders_Stores
FK_Orderline_Orders


addresses
AddressID
OwnerID
StreetName1
HouseNo
HouseCode
Zipcode
City
verwijderd
bedrijfsnaam
kostenplaats
flag_delete
date_address_inserted
AddressesTableAdapter
GetAddressbyOwnerID (@0)
CheckExistingAdress (@AddressID)
InsertnewAddress (@AddressID, @OwnerID, @Str...
Products2Toppings
Product2ToppingID
ProductID
ToppingID
Products2ToppingsTableAdapter
Fill,GetProducts2ToppingsByStoreID (@0)
FK_Addresses_Stores
FK_Products2Toppings_Products
Stores
StoreID
StoreName
BaseStoreID
StoreStatusIDoud
OrdersByFax
OrdersByEmail
OrdersToGet
AcceptsCFChecks
HtmlText1
HtmlText2
Image
BankAccountNo
StoresTableAdapter
GetStoreDataByID (@0)
Products
ProductID
ProductGroupID
ProductStatusID
Name
Description
Price
TaxID
ProductStatusID2
SBECode
BaseProductID
imageurl
AddOnGroups
StoreID
ProductsTableAdapter
GetproductDataByStoreId (@0)
FK_Products_Stores
FK_ProductsToppings_ProductSta
FK_Products_ProductStatus
ProductStatus
ProductStatusID
Description
ProductStatusTableAdapter
GetStatuses ()
FK_ProductsAddOns_ProductStatu...
FK_Products2AddOns_Products
FK_Addresses_Persons
Products2AddOns
Product2AddOnID
ProductID
ProductAddOnID
AddOnGroup
Products2AddOnsTableAdapter
Fill,GetAddon2ProductbyStoreID (@0)


postcodes2008
postcode
reeksindicatie
huisnrvan
huisnrtot
plaats
straat_lang
postcodes2008TableAdapter
GetPostcodeBoekByStoreID (@0)
FillBy,GetDataByStoreIDWithDeleveryCost (...
oducts2Toppings_Toppings
ProductsToppings
ProductStatusID
Name
Description
Price
SBECode
ToppingID
StoreID
TaxID
ProductsToppingsTableAdapter
Fill,GetDataByStoreID (@0)
ProductsAddOns
ProductAddOnID
StoreID
ProductStatusID
Name
Description
Price
TaxIDoud
TaxID
ProductStatusID2
SBECode
ProductsAddOnsTableAdapter
Fill,GetAddonDataByStoreId (@0)
neAddOns_Pro
OrderLineAddOns
OrderLineAddOnID
OrderLineID
ProductAddOnID
Quantity
Prijs
OrderLineAddOnsTableAdapter
GetOrderLineAddonsByOrderID (@0)
InsertNewOrderlineAddon (@OrderLineAddOnID...
s2AddOns_ProductsAdd...


Orderlines
OrderLineID
OrderID
ProductID
Quantity
prijs
OrderlinesTableAdapter
GetorderLinesByOrderID (@0)
FK_IkassaOrders_Orders
FK_OrderLineToppings_Orderline
FK_Orders_Person
OrderLineToppings
OrderLineToppingID
OrderLineID
ToppingID
Quantity
prijs
OrderLineToppingsTableAdapter
GetToppingDataByOrderID (@0)
InsertNewOrderlineTopping (@OrderlineTopping...
FK_Orders_PhoneNum
IkassaOrders
IKassaOrderID
StoreID
OrderID
Status
IkassaOrdersTableAdapter
GetOpenOrdersByStoreID (@0)
UpdateStatusByOrderID (@1, @0)


FK_PhoneNumbers_Stores
ProductGroups
ProductGroupID
Description
productgrouporder
ProductGroupsTableAdapter
fill,GetProductGrpsByStoreID (@0)
Persons
PersonID
Firstname
Surname1
Surname2
Birthdate
Sexe
magbesteltegoedzien
besteltegoed
infix
aanmelddatum
personeelsnummer
flag_delete
nieuwsbrief
PersonsTableAdapter
GetDataByPersonID (@0)
CheckExistingPersonByID (@personid)
InsertPersonFromIkassa (@PersonID, @Firstname...
FK_PhoneNumbers_Persons
PhoneNumbers
PhoneNumberID
OwnerID
PhoneNumber
verwijderd
PhoneTypeID
PhoneNumbersTableAdapter
GetData ()
CheckExistingTelefoonID (@PhoneNumber...

Plan van aanpak Ikassa

Inhoudsopgave

1.0 Inleiding

In het kader van het afstuderen van de opleiding Informatica, is dit plan van aanpak geschreven. Hierin is in grote lijnen aangegeven hoe de opdracht uitgevoerd zal worden, volgens welke methodieken, en de verschillende fases die de opdracht zal doorlopen.

Het plan van aanpak is voor de opdrachtgever, de opdrachtnemer en de keurende docenten binnen de Haagse Hoge School.

2.0 Betrokkenen

2.1 Opdrachtgever

Bedrijf:	Ifoods
Postbus:	9210 Alkmaar
Telefoon:	020-8208331
Contact Persoon:	Rolf van Alten
Functie:	ICT Manager

2.2 Opdrachtnemer

Naam:	John Renooij
Functie:	Ontwikkelaar
Opleiding:	Informatica (4de Leerjaar)

2.3 Locatie van uitvoering

Plaats:	Heemstede
Adres:	Industrieweg 4B 2102LH

3.0 Opdracht omschrijving

3.1 Probleemstelling / Huidige situatie

Voor veel restaurants is het aanschaffen van een horeca kassasysteem een investering die niet opweegt tegen de voordelen die de software doet beloven. Om deze reden wordt er in deze restaurants vaak gewerkt met losse papiertjes of worden bestellingen gewoon uit het hoofd bereid en eventueel bezorgd.

Bestellingen van buiten de winkel worden veelal doorgebeld (door de klant of diensten als Ifoods en Just-eat) of komen binnen doormiddel van fax en / of email. Doordat er verschillende kanalen zijn gebeurt het meer dan eens dat bestellingen te laat opgemerkt worden of gewoon verdwijnen waardoor de klant langer moet wachten of zelfs helemaal vergeten wordt.

Ook is het lastig voor de eigenaar een goed overzicht te hebben aan het eind van de dag met betrekking tot de omzet, kosten, gewerkte uren en andere administratieve cijfers.

Om het bestel proces te vergemakkelijken en eenduidig te laten verlopen alsmede het inzichtelijk maken van diverse financiële gegevens is Ifoods op zoek naar een hard en softwareoplossing. Deze oplossing moet de administratieve meerwaarde bieden zodat restaurant de oplossing graag willen implementeren alsmede een 2-ways interface bieden naar de systemen van ifoods.nl en eventueel andere diensten zoals just-eat.

3.2 Oplossing / Gewenste situatie

Ifoods is voor zijn klanten opzoek naar een oplossing waarbij op een eenvoudige en eenduidige interface bestellingen kunnen worden ingevoerd. Voor management doeleinde moet er een mogelijkheid zijn om per dag of per maand een overzichten van oa bestellingen uit te draaien ter behoeve van de administratie.

Naast dat bestellingen moeten kunnen worden ingevoerd, moet er een interface bestaan die het mogelijk maakt om orders aangeleverd door derde partijen zoals Ifoods, Just-Eat en andere aanbieders.

De interface van de nieuw te ontwikkelen omgeving dient geschikt te zijn om bediend te worden met zowel een muis en toetsenbord combinatie alsmede een Touch screen. Ook moet het mogelijk zijn om verschillende terminals te voorzien van dezelfde informatie.

3.3 Project doelstelling

De doelstelling is het ontwikkelen van een modulaire applicatie ter bevordering en structurering van het bestel proces binnen verschillende restaurants. De applicatie moet eenvoudig te bedienen zijn en intuïtief aanvoelen.

Door de intuïtieve interface en eenvoudige werkwijze moet het voor de gebruiker een hulpmiddel worden voor de administratieve aspect van het restaurant.

4.0 De ondersteuning

Om dit project met succes af te ronden, bestaan verschillende hulpmiddelen die wij ter ondersteuning zullen gebruiken.

4.1 Technieken en Technologieën

Rational Unified Process (RUP)
Voor het ontwikkelen van de applicatie zal gewerkt worden volgens de RUP ontwikkelmethode. Om dit project met succes af te ronden zal ook gewerkt worden volgens de project management richtlijnen van RUP

Unified Modeling Language (UML)
UML zal gebruikt worden voor het maken van Use Cases en ontwerpen van diverse diagrammen.

Visual C# .net 4.0 en WPF
Voor het ontwikkelen van de applicatie zal gebruik worden gemaakt van C# 4.0. Voor de interactie met de gebruiker zal ook gebruikt worden van de “nieuwe” WPF technologie ipv de forms interfacing

4.2 Tools

De onderstaande tools zullen gebruikt worden tijdens dit project voor het ontwikkelen en projectbeheer doeleinden:

- MS Office 2007
- MS Visual Studio 2010
- MS SQL Management Studio 2010
- SQL Server 2010 express edition

4.3 Betrokken personen

- Opdrachtgever
- Keurende docenten
- Ondersteund programmeur

5.0 Afbakening van de opdracht

5.1 De wensen van de opdrachtgever

- Intuïtieve interface voor dagelijkse bediening
- Ontvangen van bestellingen van Ifoods en andere aanbieders
- Stand alone kunnen werken in geval van internet uitval
- Overzichten uitdraaien tbv administratie
- Software moet ook gebruikt kunnen worden voor het aanleveren van orders aan Ifoods (ter behoeve van verder afhandeling)

5.2 Wat de opdrachtgever niet wilt

Alles omvattende interface. Verschillende functies mogen verdeeld worden onder meerdere programma's als hierdoor de interface intuïtiever wordt.

5.3 De wensen van de opdrachtnemer

Minimaal 1 maal per week overleg met de opdrachtgever ter behoeve van goede doorstroming van het project. Op deze wijze kunnen problemen snel worden benoemd en opgelost.

6.0 Randvoorwaarden

- Het project zal worden uitgevoerd in de periode die aangegeven is in de globale en detailplanning.
- Snelle en duidelijke feedback van de opdrachtgever.
  Omdat opdrachtgever en nemer niet allebei inpandig zijn, is het een voorwaarde dat op vragen van de opdrachtnemer snel wordt gereageerd ter behoeve van de doorstroming van het project

7.0 Risicofactoren

7.1 Risicofactoren

1. Er is sprake van ziekte
2. Een gebrek aan kennis
3. Er is sprake van een inschattingsfout binnen de planning
4. Het product voldoet niet aan de eisen van de opdrachtgever

7.2 Hoe risico's herkend worden en de gevolgen

1. Een van de betrokken personen is langdurig afwezig / niet bereikbaar en kan niet meewerken aan de geplande werkzaamheden.
2. Omdat er gewerkt wordt met nieuwe technieken en verspreide architectuur kan het voorkomen dat er problemen voordoen die door de opdrachtnemer niet in korte tijd op zijn te lossen
3. De deadlines worden niet gehaald en hierdoor zal de opdrachtnemer extra uren moeten investeren of in overleg met opdrachtgever de functionaliteit moeten verdelen over een langere ontwikkelingstijd.
4. De opdrachtgever is niet tevreden met de release van de final versie aan het einde van de planning. Het gevolg hiervan is dat het project als mislukt kan worden afgeschreven

7.3 Het voorkomen van risico's en de negatieve gevolgen

1. In het geval van ziekte bij de opdrachtgever zal het werk worden overgedragen aan een ander persoon die ook nauw betrokken is bij het project. In het geval van ziekte bij de opdrachtnemer zal er zoveel mogelijk doorgewerkt worden vanuit huis. Daarnaast is in de planning enige uitloop opgenomen om te voorkomen dat het project niet op tijd af komt.
2. Naast het gebruik maken van de beschikbare bronnen zoals internet en diverse boeken, zal bij problemen waarbij niet binnen een korte periode geen oplossing gevonden kan worden gebruik worden gemaakt van de beschikbare ondersteuning in het netwerk van zowel de opdrachtgever als opdrachtnemer.
3. De planning is dusdanig opgebouwd dat er een ruimte is voor uitloop. In het geval dat het eind product in gevaar dreigt te komen zal worden overlegd om bepaalde functies later toe te voegen .
4. Door wekelijks contact met de opdrachtgever te hebben en van te voren een aantal test criteria vast te leggen is dit risico te beperken tot een minimum.

8.0 Projectorganisatie

8.1 afspraken

Omdat het project door 1 persoon wordt uitgevoerd zijn er binnen de project organisatie geen afspraken gemaakt. Wel zijn er met de opdrachtgever afspraken gemaakt over terugkoppeling en wekelijks evaluatie.

8.2 Documentatie

De opdrachtgever heeft ten alle tijde inzage in de projectdocumentatie en kan eventueel nog wijzigingen aanbrengen gaande het project.

8.3 Taakverdeling

Omdat het project door 1 persoon wordt uitgevoerd is er geen sprake van een taakverdeling.

9.0 Wijze van rapporteren

9.1 Hoe de projectgroep verslag doet

Elke week wordt er overleg gepleegd met de opdrachtgever. Hierbij wordt de voorgang besproken en eventueel project in overleg bijgestuurd worden.

10.0 Globale planning

Week 1 (14 nov 2011)	Project bespreking met opdrachtgever Opdracht omschrijving Plan van Aanpak Bijwerken Project dagboek
Week 2 (21 nov 2011)	Plan van Aanpak Bijwerken Project dagboek
Week 3 (28 nov 2011)	Inrichten omgeving en documenteren van bestaande omgevingen (welke relevant zijn) Bijwerken Project dagboek
Week 4 (5 dec 2011)	Testen haalbaarheid nieuwe technieken (WCF en WPF) Bijwerken Project dagboek
Week 5 (12 dec 2011)	Requirements uitwisselingen services (ontvangen) Use case model + use case scenario's uitwisselingen servers (ontvangen) Business case uitwisselingen services (ontvangen) Projectplan uitwisselingen services (ontvangen) Bijwerken Project dagboek
Week 6 (19 dec 2011)	Ontwikeling uitwisselingen services (ontvangen)
Week 7 (26 dec 2011)	Requirements Frondend interface Use case model + use case scenario's Frondend interface Business case uitwisselingen services Projectplan Frondend interface Bijwerken Project dagboek
Week 8 (2 Jan 2012)	Ontwikkeling Frondend interface Bijwerken Project dagboek
Week 9 (9 Jan 2012)	Ontwikkeling Frondend interface Bijwerken Project dagboek
Week 10 (16 jan 2012)	Uitloop moment Bijwerken Project dagboek
Week 11 (23 jan 2012)	Requirements uitwisselingen services (verzenden) Use case model + use case scenario's uitwisselingen servers (verzenden) Business case uitwisselingen services (Verzenden) Projectplan uitwisselingen services (Verzenden) Acceptatietest frondend Bijwerken Project dagboek
Week 12 (30 jan 2012)	Ontwikkeling uitwisselingen services

	(Verzenden) Bijwerken Project dagboek
Week 13 (6 feb 2012)	Use case model + use case scenario's Installer Business case Installer Projectplan Installer
Week 14 (13 feb 2012)	Ontwikkelen Installer
Week 15 (20 Feb 2012)	Acceptatie test installer Eind documentatie schrijven
Week 16 (28 feb 2012)	Eind documentatie schrijven
Week 17 (5 maart 2012)	Eind documentatie schrijven