Afstudeerverslag

Afstudeerdossier deel 1

Student: K.B. Hendriks

Stagebegeleider: T. Cocx

Afstudeerbedrijf: Pamicon

Bedrijfsmentor: N. Noordijk

Plaats: Zoetermeer

Datum: 25-3-2013

Afstudeerstage Pamicon

In opdracht van	De Haagse Hogeschool, Academie voor ICT & Media Bredewater 24 2715 CA, Zoetermeer
Uitgevoerd bij	Pamicon Ranonkelweg 12 2651 MX, Berkel en Rodenrijs
Auteur	K.B. Hendriks
Versienummer	1.0
Datum	25-3-2013

Voorwoord

Mijn naam is Kevin Hendriks en ik ben 21 jaar. Ik volg de opleiding Informatica aan de Academie voor ICT & Media. Dit afstudeerverslag schrijf ik om de examinatoren en mijn begeleider te laten zien wat ik de afgelopen maanden voor werkzaamheden heb verricht tijdens mijn afstudeerstage en wat ik daarvan geleerd heb. Het belangrijkste van dit verslag is aan te tonen dat ik mijn competenties die ik van te voren had opgesteld, heb volbracht.

Aan het eind van de stageperiode kijk ik tevreden terug. Het was even wennen in het begin maar naarmate ik langer bezig was en mijn kennis uitbreidde nam de interesse toe. Het pionieren en daarmee het idee dat je echt met nieuwe dingen bezig bent, trok mij erg aan. Verder vond ik het ook heel leuk om te werken op een echte ICT afdeling in een bedrijf dat haar geld verdient met een product dat zij levert op de ICT markt. Het was eens heel wat anders dan op school werken met een groep studenten. Dit is het echte werk!

Ik hoop dan ook dat met dit eindrapport een goede reflectie wordt gegeven van de werkzaamheden tijdens mijn afstudeerstage en de keuzes die ik tijdens met werkzaamheden heb gemaakt.

Uiteraard wil ik mijn begeleiders bedanken. Als eerst wil ik Tim Cocx bedanken voor de begeleiding vanuit de Academie voor ICT & Media. Daarnaast ook Niels Noordijk en Aad Palms voor het begeleiden vanuit Pamicon en voor het opzetten van de afstudeeropdracht. Ook de andere collega's waar ik mee werkte wil ik bedanken voor de gezelligheid en de leuke vier maanden die ik er heb gehad.

25-3-2013,
Kevin Hendriks

Inhoudsopgave

Samenvatting

Voor mijn afstudeerstage heb ik in 18 weken een afstudeeropdracht uitgevoerd bij het bedrijf Pamicon. De opdracht behelsde het toevoegen van een inkoop en voorraad module aan een kassasysteem genaamd Delivery Manager. Dit diende bereikt te worden door het uitwerken van een functioneel ontwerp, een technisch ontwerp waarin ook is opgenomen wat de gevolgen zijn voor de database structuur, de volledige ontwikkeling en het uitvoeren van een test. Gedurende het uitvoeren van de opdracht moest er rekening gehouden worden met de requirements die vooraf aangeleverd waren en die tijdens het gehele traject veranderingen konden ondergaan.

De eerste weken heb ik mij bezig gehouden met het opzetten van een Plan van Aanpak en het maken van ontwerpen. In het Plan van Aanpak zijn de methodes en standaarden vastgelegd in overleg met de bedrijfsmentor. Tijdens het ontwikkelen heb ik gebruik gemaakt van SCRUM. Deze methodiek sloot het beste aan op de voor Pamicon bekende werkwijze en is ook geschikt als er regelmatig prototypes opgeleverd moeten worden.

Tijdens het ontwerpen zijn de requirements vertaald naar functionaliteiten door middel van het opzetten van Use Cases en schermontwerpen. Deze zijn vervolgens vertaald naar technische ontwerpen. De technische ontwerpen zijn gebruikt tijdens het ontwikkelen van de modules en het opzetten van de achterliggende database.

Het ontwikkelen begon met de eerste sprint die in het teken stond van de stamgegevens. De stamgegevens zijn gegevens die gebruikt worden op meerdere plekken in de applicatie en die beheerd kunnen worden door de gebruiker zelf. De tweede sprint was gericht op de inkoopmodule. Deze module biedt de mogelijkheid orders te kunnen plaatsen en bewerken. Ook kunnen binnengekomen goederen (ingrediënten) worden ingeslagen. Indien nodig moeten ingrediënten ook kunnen worden geretourneerd. Deze acties hebben allemaal op een andere manier invloed op de voorraad. De voorraad was het onderwerp van de derde sprint. De voorraadmodule geeft de gebruiker de mogelijkheid om per ingrediënt de voorraad en de mutatiedetails te bekijken. Ook kunnen er correcties doorgevoerd kunnen worden doormiddel van het invoeren van tellingen. Hier kunnen ook ingrediënten worden afgeboekt van de voorraad. In de laatste ontwikkelsprint zijn er toevoegingen gedaan aan de modules die het mogelijk maken om te werken volgens een FIFO werkwijze.

Om de overdraagbaarheid te verbeteren is de systeemdocumentatie uitgebreid met een klassendiagram. Het doel van het klassendiagram is een beeld te creëren van de toevoegingen en veranderingen die het gevolg zijn van ontwikkelen van de nieuwe modules.

Voor oplevering van de modules zijn deze getest. Tijdens het opzetten van de test is er overleg geweest om te komen tot een goede testmethodiek en techniek die de belangrijkste kwaliteitsattributen test. Hieruit is naar voren gekomen dat functionaliteit en gebruiksvriendelijkheid de belangrijkste attributen waren. Om deze attributen te kunnen testen is gebruik gemaakt van een Procescyclustest. In de laatste weken zijn er een aantal rapportages opgezet die vanuit Delivery Manager gegenereerd worden. De opdracht is afgerond door het opleveren van een gebruikershandleiding.

1. Inleiding

Pamicon is een ICT dienstverlener die zich bezig houdt met het beheren van systemen en het ontwikkelen van software. Het bedrijf ontwikkelt al een aantal jaren een softwarepakket genaamd Delivery Manager. Delivery Manager is een modulair kassa softwarepakket voor restaurants, cafés, bars en bezorg- en afhaalrestaurants zoals pizzeria's, cafetaria's en andere (fast)foodbedrijven met een bezorgdienst. Het team van software ontwikkelaars is samen met de klant continu bezig Delivery Manager te verbeteren met nieuwe functionaliteiten. Hierdoor blijft de bezorgsoftware up-to-date met de ontwikkelingen in de markt.

Aan het pakket dient een module toegevoegd te worden waardoor er inzicht is in de voorraad en inkopen van goederen. Het toevoegen van de modules vormde het onderwerp van mijn afstudeerstage. In dit verslag geef ik weer hoe ik de opdracht heb aangepakt.

In hoofdstuk 2 wordt in het kort het bedrijf Pamicon beschreven, de werkwijze die zij hanteren en mijn positie binnen het bedrijf gedurende mijn afstudeerstage. Vervolgens wordt in hoofdstuk 3 de opdracht beschreven aan de hand van de huidige situatie en de werkzaamheden die verricht moeten worden om te komen tot de producten die opgeleverd moeten worden.

In hoofdstuk 4 worden de uitgevoerde werkzaamheden beschreven. Deze werkzaamheden omvatten het uitwerken van een functioneel ontwerp, een beknopt technisch ontwerp tot het ontwikkelen van de modules en het uiteindelijk testen van deze modules.

Na het beschrijven van alle werkzaamheden zal ik het proces en het opgeleverde product evalueren en zal ik kort terugkijken op de begeleiding die ik gehad heb tijdens het afstuderen. Ook zal ik evalueren of ik de door mij opgegeven competenties heb vervuld in de afgelopen vier maanden.

Uiteindelijk sluit ik dit verslag af met een conclusie. In deel twee van het afstudeerdossier zijn de bijlages te vinden waar in het verslag naar verwezen wordt.

2. Pamicon

2.1 Het bedrijf

In 1977 is Pamicon Software in Rotterdam gestart als software house met als doel het leveren van maatwerk software applicaties. In de loop der jaren is Pamicon gegroeid van primair softwareontwikkelaar naar een allround ICT dienstverlener. De ontwikkelaars houden zich vooral bezig met webbased oplossingen en Microsoft.NET omgevingen.

Een van de producten van Pamicon is Delivery Manager, een modulair kassa softwarepakket voor restaurants, cafés, bars en bezorg- en afhaalrestaurants zoals pizzeria's, cafetaria's en andere (fast)foodbedrijven met een bezorgdienst. Het doel van Pamicon is om zich uiteindelijk te kunnen onderscheiden van de concurrentie. Dit doen zij door software te bieden waar samen met de klant over nagedacht wordt.

Bijkomend gevolg van het goed onderhouden van contacten met klant is dat de software steeds vernieuwd wordt en er veel veranderingen worden doorgevoerd. Hierdoor blijft de software up-to-date met de ontwikkelingen in de markt.

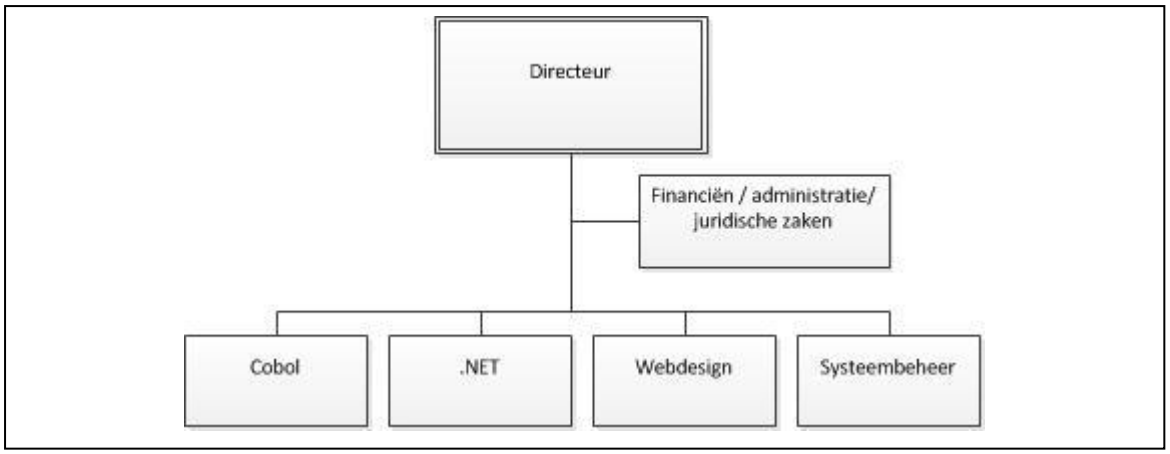
Het motto van Pamicon luidt: "Goede software wordt gemaakt door personen en interacties, eerder dan processen en tools".

2.2 Structuur van bedrijf

Bij Pamicon zijn 14 mensen werkzaam. Hiervan zijn er constant 5 mensen bezig met het ontwikkelen en onderhouden van het softwarepakket Delivery Manager.

Pamicon hield zich van origine bezig met het ontwikkelen van applicaties in Cobol. Veel van de klanten maken nog steeds gebruik van deze applicaties. Om deze reden zijn er nog 3 Cobol programmeurs aanwezig. Daarnaast is er een team van 3 mensen aanwezig dat zich bezighoudt met het ontwikkelen en ontwerpen van websites. Voor de ondersteuning op het gebied van hardware zijn er nog een drietal medewerkers werkzaam bij Pamicon. Deze houden zich bezig met de systemen van de klanten waarop Delivery Manager draait. Een medewerker die de marketing regelt is betrokken bij al deze afdelingen binnen het bedrijf. Hij is verantwoordelijk voor het aantrekken van nieuwe klanten en het onderhouden van de contacten met bestaande klanten.

Ten slotte is er nog een team van twee ontwikkelaars van Delivery Manager. Zij zijn constant bezig met het toevoegen van nieuwe functionaliteiten en het verbeteren van de oude functionaliteiten. Deze ontwikkelaars hebben van Delivery Manager een heel nieuw product gemaakt sinds het opgekocht is door de directeur van Pamicon. Zij hebben in een aantal jaren van Delivery Manager een WPF (Windows Presentation Foundation) applicatie gemaakt en daarbij de bestaande takeaway module uitgebreid met een restaurant-en deliverymodule.



Figuur 2.2: Organogram Pamicon

In het organogram (zie figuur 2.2) is de eerder beschreven structuur binnen Pamicon zichtbaar. Tijdens mijn afstudeerstage heb ik deel uitgemaakt van het .NET team waarbinnen ik mijn eigen opdracht heb uitgevoerd. Tijdens de opdracht heb ik weinig contact gehad met de teams die zich op hetzelfde niveau bevinden maar is er wel naar boven gecommuniceerd.

2.3 Werkwijze

Pamicon werkt niet volgens een vaste planning en maakt geen gebruik van een softwaremethode die bij naam genoemd kan worden. Wel proberen ze op een Agile manier, dus iteratief, te werk te gaan. Daarnaast is het opleveren van werkende software belangrijker dan maken van ondersteunende documentatie, zoals bijvoorbeeld een handleiding voor de klant of systeemdocumentatie van een softwarepakket. Pamicon speelt eerder in op wensen van klanten dan dat zij een plan volgt. Deze flexibele manier van werken is mogelijk door de platte organisatie van Pamicon. Onderling overleg hoeft hierdoor ook niet ingepland te worden in de vorm van een vergadering of bijeenkomst maar wordt gedaan gedurende het werkproces.

3. Opdrachtomschrijving

Allereerst zal er in dit hoofdstuk een beeld geschetst worden van het softwarepakket Delivery Manager. Daarna worden de tekortkomingen van het pakket beschreven, het doel van mijn opdracht en de manier waarop ik het doel bereikt heb.

3.1 Delivery Manager

Delivery Manager (DM) is een kassassysteem voor de horeca. Het product wordt verspreid in verschillende soorten versies. Afhankelijk van de licentie die de klant koopt heeft Delivery Manager functionaliteiten. De basis van DM wordt gevormd door de Takeaway module. Deze module ondersteunt het bedrijfsproces van verwerking van een afhaalorder. Dit behelst het bereiden van een order tot het afronden van de order.

De Takeaway module kan uitgebreid worden met een module die het bezorgproces ondersteunt, de Delivery module. Bij bezorgen kunnen de bezorgorders worden gemonitord tijdens het proces van aanmaken van de order, het klaarmaken van de gerechten, het volgen van de bezorgers tot het uiteindelijk afhandelen en afleveren van de order. Voor deze module staat DM in contact met websites zoals Thuisbezorgd en JustEat. Orders die geplaatst worden op deze websites worden automatisch opgenomen in Delivery Manager.

Naast het bezorgen kan er ook een module restaurant gekocht worden. Deze module zorgt voor een overzicht van de horecagelegenheid. Er kunnen per tafel, bar of bijvoorbeeld zitbank orders opgenomen worden. Van alle orders kan worden gemonitord wat de status is in het proces van aannemen van de order, bereiden in de keuken tot het afhandelen van de order.

3.2 De opdracht

Doel van deze opdracht is het toevoegen van een inkoop en voorraad module aan Delivery Manager. Het moet mogelijk zijn om inkooporders in te voeren die de horeca gelegenheid doet bij groothandel of andere afnemers. De aflevering van een order heeft invloed op de voorraad. De voorraad wordt ook beïnvloed door het verkopen van producten, retourneren, afboeken en correcties naar aanleiding van tellingen.

Bij het verwerken van de verkoop moet rekening gehouden worden met de mogelijkheid dat samengestelde artikelen worden verkocht. Zo moet een menu worden uitgesplitst naar verschillende inkoopartikelen, ook wel ingrediënten. De voorraad zal dus bijgehouden moeten worden op ingrediëntniveau.

Retourneren van ingrediënten kan gewenst zijn indien er een verkeerd product is geleverd, als er schade is geconstateerd bij het geleverde product of wanneer er een andere reden is die er voor zorgt dat het retourneren van ingrediënten nodig is.

Het afboeken gebeurt voornamelijk naar aanleiding van het verstrijken van de THT (Tenminste Houdbaar Tot) datum. De module moet afboekingen mogelijk maken, maar ook inzicht geven in de THT datum van de voorraad en FIFO (First In – First Out) voorraadbeheer mogelijk maken.

Tellingen worden doorgevoerd om de voorraad in het systeem te corrigeren met de voorraad die aanwezig is na het doen van een telling.

Uiteindelijk zal het systeem zelf, wanneer het voorraad niveau onder een ijzeren voorraad zakt, moeten aangeven dat er besteld moet worden. De voorraad zal bestaan uit verschillende partijen met afwijkende THT data. Hierdoor kan er een overzicht worden gegeven van dreigend bederf van ingrediënten.

3.3 Uit te voeren werkzaamheden

De opdracht omvat het uitwerken van een functioneel ontwerp, een technisch ontwerp waarin ook is opgenomen wat de gevolgen zijn voor de database structuur, de volledige ontwikkeling en het uitvoeren van een test. Aan het einde van het traject zullen de modules geïntegreerd worden in Delivery Manager. De modules zullen in eerste instantie ontwikkeld gaan worden in de laatste stabiele ontwikkelaarsversie van Delivery Manager. Hiervoor zal er in een aparte branch (ontwikkelaarstak) gewerkt gaan worden. Later zal deze tak samengevoegd worden met de stabiele branch van Pamicon.

Er zullen tijdens het gehele traject meerdere iteraties worden doorlopen die als uitkomst een prototype opleveren. De requirements zullen gaande het traject veranderingen ondergaan. Nieuwe inzichten hebben tot gevolg dat requirements aangepast worden, afvallen of toegevoegd worden. In overleg met de bedrijfsmentor en de medewerker van de marketing afdeling zullen nieuwe wensen en ideeën besproken worden en mogelijk meegenomen worden voor ontwikkeling. Het uiteindelijke resultaat kan hierdoor afwijken van wat er op voorhand is bedacht. Wanneer na een aantal iteraties de volledige modules zijn ontwikkeld zullen deze getest worden. Hiervoor zal allereerst een testproces opgezet moeten worden. Vervolgens zal het testen resulteren in een rapport waarin de einduitslag beschreven staat en waarin er een advies gegeven wordt over het vrijgeven van de module.

Wanneer de modules volledig getest zijn en er een stabiele versie is opgeleverd, zullen er een aantal rapportages in Crystal Reports opgezet worden. Voor deze rapportages zal er gewerkt moeten worden met de functionaliteiten van het databasemanagementsysteem SQL Server. Het aanmaken en gebruiken van stored procedures zal hiervoor minimaal nodig zijn om rapportages op te zetten. Uiteindelijk zullen de verschillende rapportages geïmplementeerd moeten worden in de stabiele versie van Delivery Manager zodat ze gegenereerd kunnen worden door de gebruiker.

Als laatste zal het geheel opgeleverd worden door middel van een beschrijvende handleiding van de modules en mogelijk ook een presentatie.

3.4 Op te leveren producten

Tussenproducten:
- Plan van Aanpak

- Analyse / Functioneel ontwerp
(use cases en beschrijvingen, schermontwerp)

- Technische ontwerpen
(klassendiagram database, sequentiediagrammen)

- Prototypes: Eindproduct van elke iteratie waarmee de toegevoegde functionaliteiten getoond kunnen worden.

- Testplan, testontwerp en testrapportage

Eindproduct:

- Werkende modules

- Handleiding

3.5 Terugkoppeling en overleg

Tijdens het uitvoeren van de opdracht zal er veel terugkoppeling zijn met de bedrijfsmentor. Keuzemogelijkheden zullen worden besproken zodat er voor de oplossing gekozen wordt waar iedereen zich in kan vinden. Ook voorkomt dit het dat verkeerde geïnterpreteerde wensen en ideeën ontwikkeld worden.

3.6 Testen en evalueren

Tijdens het ontwikkelen zal er gebruik worden gemaakt van het debuggen. De modules worden geëvalueerd aan het einde van elke sprint door het doorlopen van de prototypes. Daarnaast is er na het ontwikkelen nog tijd en ruimte voor het opzetten en uitvoeren van een test. Het streven is het opleveren van werkende en foutloze modules. Als er nog fouten in de modules zitten die de correcte werking niet verstoren kunnen deze naar boven komen door ingebruikname van de klant. Dit is niet erg en moet natuurlijk wel voorkomen worden.

4. Werkzaamheden

In dit hoofdstuk staat beschreven welke stappen zijn doorlopen en op welke manier er te werk is gegaan.

4.1 Plannen en oriënteren

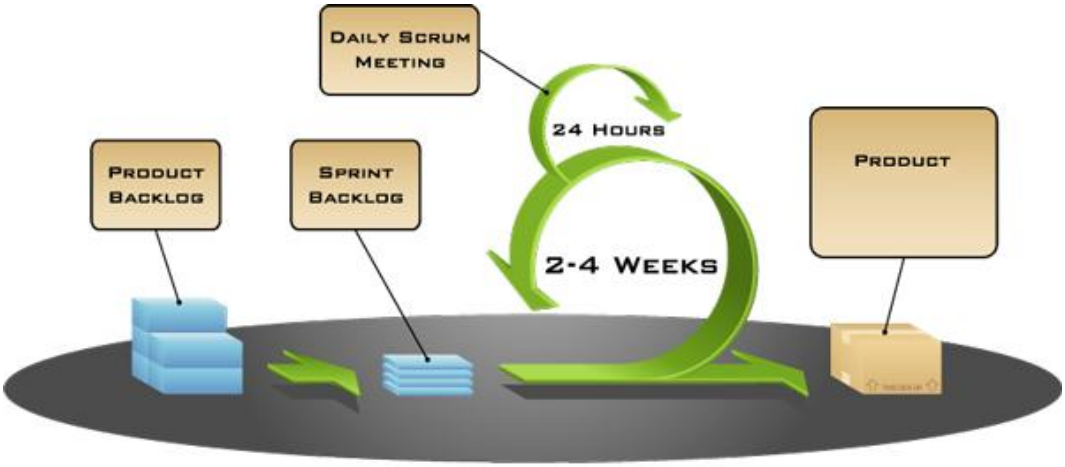
Het plannen en oriënteren heeft als doel te kijken hoe het project het beste aan kan worden gepakt. De producten die in deze fase tot stand zijn gekomen zijn een plan van aanpak (bijlage 2) en een globale planning. Het plan is besproken met de bedrijfsmentor en heeft geresulteerd in een plan waar wij ons beiden in konden vinden. Na het opzetten van een plan is als eerste de huidige situatie in kaart gebracht. Daarna zijn er verschillende ontwerpen gerealiseerd om te komen tot de gewenste situatie.

4.1.1 Plan van aanpak

Om aan te sluiten op de werkwijze van Pamicon kon het best gebruik worden gemaakt van een iteratieve ontwikkelmethode. De keuze voor de ontwikkelmethode was aan mij. Pamicon zelf werkt nu een tijdje volgens de agile werkwijze maar gebruikt geen specifieke ontwikkelmethode. In overleg met mijn bedrijfsmentor heb ik ervoor gekozen gebruik te maken van SCRUM omdat dit het beste aansluit op de voor Pamicon bekende werkwijze. SCRUM is een geschikte methodiek omdat ik zal gaan werken door middel van oplevering van verschillende prototypes. Elke iteratie, in SCRUM sprint genoemd, zal resulteren in een prototype.

4.1.2 SCRUM

Deze ontwikkelmethode vereist het ontwikkelen in sprints. Hierbij gaat het om iteraties van een vaste lengte van één tot vier weken. Aan het einde van een sprint dient altijd software opgeleverd te worden. Daarnaast zijn samenwerking, communicatie en werken als een team zeer belangrijk. Elke sprint begint met het vaststellen van de onderdelen die behandeld gaan worden. Dit heb ik gedaan in overleg met de bedrijfsmentor en een medewerker van de marketing afdeling. De onderdelen staan beschreven op de productbacklog. Aan de productbacklog kunnen ieder moment nieuwe onderdelen worden toegevoegd. Per taak wordt bepaald hoeveel tijd er nodig is. Omdat een sprint een bepaalde doorlooptijd heeft aan uren, kan er precies gepland worden. De opzet van SCRUM is te vinden in figuur 4.1.2.1.



Figuur 4.1.2.1: Opzet van SCRUM

SCRUM kent een aantal fases waarin een taak zich kan bevinden, zoals weergegeven in figuur 4.1.2.2:

To do	In progress	To check	Ready
In deze fase staan de producten die in afwachting staan om geproduceerd te worden.	De producten in deze fase worden geproduceerd.	De producten in deze fase zijn gereed. Correctheid wordt gecontroleerd.	Als er gecontroleerd is en er geen fouten zijn gevonden beland het product in de ready-fase.

Figuur 4.1.2.2: Te definiëren fases van taken in SCRUM

De dagelijkse voortgang in een sprint is in gesprekken van ongeveer tien minuten doorgesproken met de bedrijfsmentor. Aan het eind van elke sprint is het opgeleverde prototype geëvalueerd met de bedrijfsmentor en de medewerker van de marketingafdeling. Hieruit ontstonden nieuwe inzichten die werden verwerkt op de productbacklog. In de evaluatie wordt ook de nieuwe sprint ingepland.

Team Foundation Server (TFS)

Er is gebruik gemaakt van Team Foundation Server voor het plannen van de sprints en het bijhouden van de productbacklog. TFS is een samenwerkingsplatform waarmee versiebeheer kan worden toegepast en rapportages kunnen worden gegenereerd. Daarnaast kunnen er ook werkzaamheden worden ingepland en kan de voortgang worden gemonitord. Het platform biedt door een downloadbare plug- in de mogelijkheid gebruikt te worden in combinatie met SCRUM. Voorafgaande aan het ontwikkelen is de gehele productbacklog ingevoerd. Tijdens het ontwikkelen is voor elke sprint een sprintbacklog opgesteld waarin de items van de productbacklog zijn afgewerkt.

Alternatieven voor SCRUM

Er is voor SCRUM gekozen omdat de andere bekende methodieken niet volledig aansloten op de opdracht en de werkwijze van Pamicon. De methodieken zijn afgevallen om de volgende redenen:

Extreme Programming (EP):
Deze best practice verwacht dat er voor het ontwikkelen tijd wordt besteed aan het opzetten van Unit Tests. Daarnaast moet er worden gewerkt in paren. Ook dient de programmacode continu geïntegreerd te worden. Omdat er tijdens het afstuderen alleen gewerkt moet worden en omdat er gekozen is voor een aparte testfase na het ontwikkelen was deze methodiek niet geschikt genoeg.

Adaptive Software Development (ASD):
Wanneer er volgens ASD gewerkt wordt moet er een schema opgesteld worden waarin de ontwikkelcycli worden opgesteld. Dit idee komt overeen met SCRUM. Echter wordt er hierbij vanuit gegaan dat een ontwikkelcyclus tussen de vier tot acht weken duurt. Dit heeft als gevolg dat er minder prototypes opgeleverd worden en er minder vaak teruggekoppeld kan worden. Dat was voor mijn ontwikkeltraject van acht weken te weinig.

Dynamic Systems Development Method (DSDM):
DSDM heeft veel overeenkomsten met SCRUM. Een verschil tussen de technieken is dat er bij DSDM gewerkt wordt met verschillende fases. Het gaat hier om de feasability fase (bepalen van de geschiktheid van het systeem a.d.h.v. kosten en benodigde tijd) en de business study fase (de primaire functionaliteit bepalen). Deze fases zijn overbodig omdat de primaire functionaliteiten al bekend zijn en inzicht in de kosten en duur geen toevoeging is voor het project. Daarnaast is het niet duidelijk of er bij DSDM ook gebruik wordt gemaakt van het structureel afwerken van een productbacklog.

4.1.3 Modelleertaal: Unified Modeling Language (UML)

Modelleertalen zijn ontwikkeld om een exacte specificatie van systemen te maken, zodat allerlei betrokkenen een beter begrip kunnen krijgen van de gemodelleerde systemen. Daarnaast wordt het ook gebruikt voor de specificatie van systeem benodigdheden, structuren en gedrag van een systeem. Als modelleertaal tijdens het afstuderen is er gebruik gemaakt van Unified Modeling Language (UML) omdat deze standaard bij mij bekend is en altijd zeer goed is bevallen. Daarnaast beschikt het ook over een breed arsenaal aan ondersteunende gereedschappen en ondersteunt het veel verschillende diagramtechnieken.

Tijdens het uitvoeren van mijn afstudeerstage zou ik te maken krijgen met veel aspecten (denk aan data en functionaliteit) van een informatiesysteem. UML is erg bruikbaar voor het grafisch weergeven van deze aspecten.

4.1.4 Globale planning

De gehele afstudeerperiode bedraagt 17 weken. Deze weken zijn verdeeld in een oriëntatiefase, ontwikkelsprints en een testfase.

Oriënteren
In de oriëntatiefase heb ik in samenspraak met de bedrijfsmentor het volgende opgeleverd:

- Plan van aanpak
- Huidige situatie (klassendiagram database)
- Functionele ontwerpen (use cases, schermontwerp)
- Technische ontwerpen (klassendiagram database, sequentie diagrammen)
-Productbacklog

Sprint 1: Stamgegevens
Doel van deze sprint is het ontwikkelen van de stamgegevens. Stamgegevens zijn gegevens die gebruikt worden in de applicatie maar daarnaast ook beheerd kunnen worden door de gebruiker. In deze sprint maak ik me ook vertrouwd met het bestaande programma en de gebruikte programmeertalen.

Sprint 2: Inkoopmodule
Het toevoegen van de inkoopmodule staat centraal in deze sprint.

Sprint 3: Voorraadmodule
De voorraadmodule zal in de sprint gerealiseerd worden.

Sprint 4: FIFO werkwijze
Als laatste worden in deze sprint de functionaliteiten toegevoegd die er voor zorgen dat er gewerkt kan worden volgens een FIFO werkwijze.

Testen
In deze fase wordt er een testplan opgesteld. Vanuit het testplan worden er vervolgens ontwerpen gemaakt voor het uitvoeren van de test. Uiteindelijk resulteert het in een rapportage waarin de bevindingen worden beschreven en er een advies wordt gegeven voor vrijgave van de software.

Rapportages
Opzetten van rapportages met behulp van Crystal Reports die opgevraagd kunnen worden in Delivery Manager.

Opleveren
Er zal aan het bedrijf een gebruikershandleiding worden geleverd van de modules waarin alle toegevoegde functionaliteiten staan beschreven.

4.1.5 Requirements

De requirements voor de modules stonden vooraf vast. Door het opleveren van prototypes tijdens het ontwikkelen kunnen er nieuwe inzichten ontstaan waardoor het eindresultaat kan afwijken van de requirements. De requirements betreffen het volgende:

- **Inkoop**
  - Invoeren inkooporder.
  - Aanmaken retouren.

- **Voorraad**
  - Overzicht per inkoopartikel met mogelijkheid door te klikken naar een mutatieoverzicht.
  - Overzicht per inkoopartikel met mogelijkheid door te klikken naar een overzicht van de aanwezige partijen met THT datum.
  - Signalering bij te lage voorraad is gewenst.
  - Tellingen invoeren waaruit voorraadverschillen duidelijk worden.
  - Signaleren bij dreigend bederf.
  - Afboeken met opgaaf van reden.
  - Binnen melden inkooporder. Voorraad zal gemuteerd moeten worden en THT datum moet vastgelegd worden.

- **Verkoop**
  - Vrijgeven van de verkoop van een artikel alleen als alle ingrediënten aanwezig zijn.
  - Automatisch muteren bij verkoop van een verkoopartikel

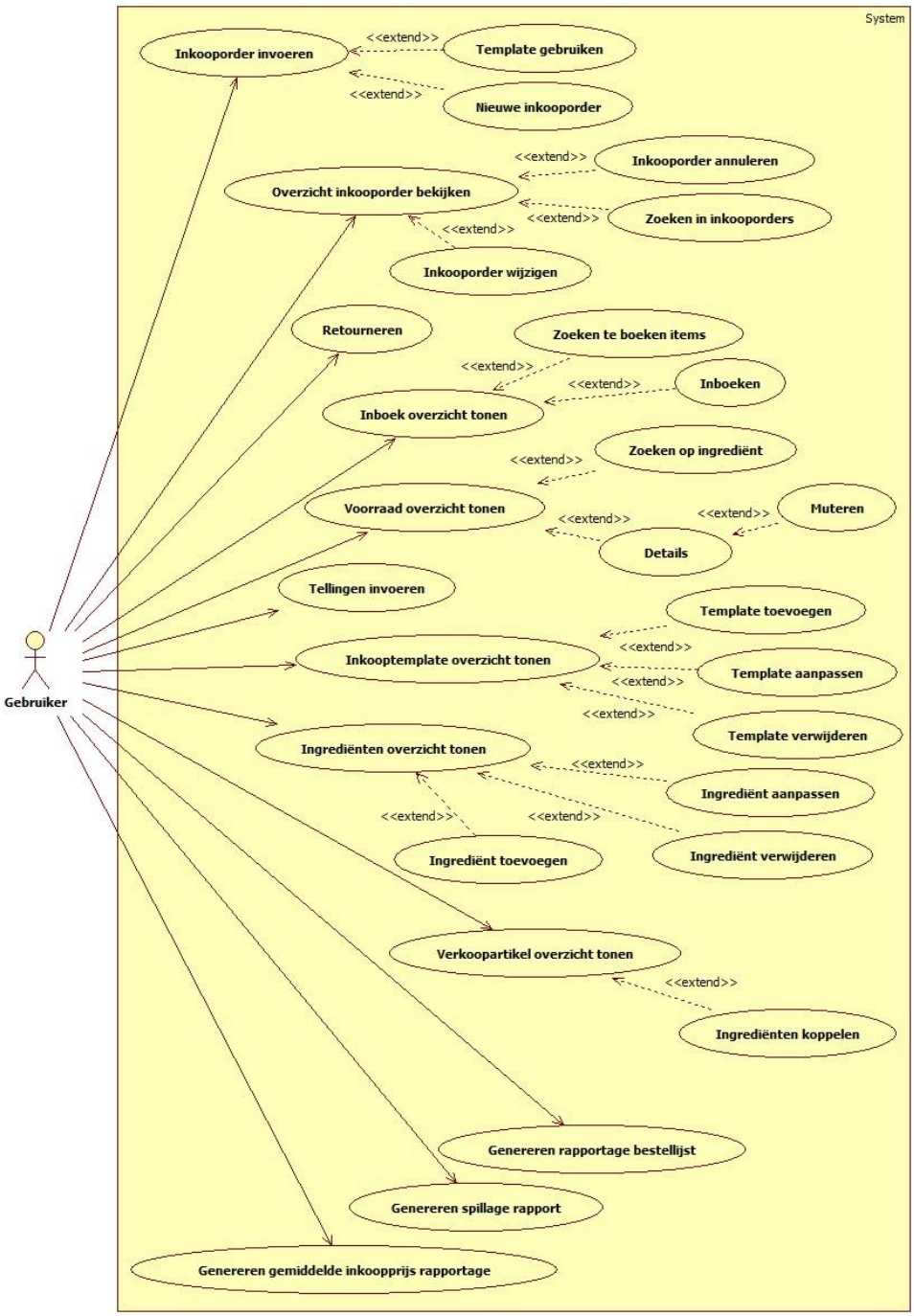
- **Stamgegevens**
  - Vastleggen uit welke ingrediënten een verkoopartikel bestaat. Zorgen voor onderscheid in verkoopartikelen en inkoopartikelen.
  - Informatie over leveranciers.
  - Mogelijkheid tot vastleggen van inkoopeenheden.

- **Rapportage (voorbeelden)**
  - Overzicht van spillage per dag/periode en per medewerker.
  - Bestellijsten genereren.
  - Verwachte verkoop berekenen.

4.2 Ontwerpen

4.2.1 Use Cases en beschrijvingen

Het doel van de use cases is om een duidelijker beeld te krijgen van de interactie tussen de gebruiker en Delivery Manager. Het diagram is opgesteld aan de hand van de requirements. Voor elke case is er ook een beschrijving gemaakt. Deze geven een invulling aan de functionaliteiten die te zien zijn in het onderstaande diagram.



Figuur 4.2.1.1: Use case aan de hand van de requirements

De beschrijvingen van de Cases hebben veel veranderingen ondergaan. Samen met de bedrijfsmentor en ook de medewerkers van de marketing afdeling is er gediscussieerd over de verschillende functionaliteiten. Dit had als doel te controleren of de beschreven requirements allemaal mee waren genomen in het ontwerp. Ook heeft het geleid tot nieuwe functionaliteiten. Een voorbeeld hiervan is het invoeren van een inkooplijst(template). Dit is heel erg handig voor bedrijven die regelmatig dezelfde bestellingen doen. Ze kunnen dan een standaard inkooplijst aanmaken en deze kan dan gebruikt worden bij het inkopen van goederen bij één bepaalde leverancier. Deze oplossing zorgt ervoor dat er sneller en op een efficiëntere manier bestellingen kunnen worden gedaan.

Naast deze functionaliteit zijn er nog tal van andere functionaliteiten terug te vinden in bijlage 4.

4.2.2 Klassendiagram database

Na het beschrijven van de use cases ben ik ontwerpen op gaan zetten om de operationele database uit te bereiden. In eerste instantie is de operationele database in kaart gebracht. De tabellen die relevant zijn voor het uitvoeren van de opdracht zijn weergegeven in een klassendiagram, zie bijlage 6. De bestaande (operationele) tabellen die gebruikt dienen te worden als ondersteuning voor de inkoop-en voorraadmodule zijn niet aangepast.

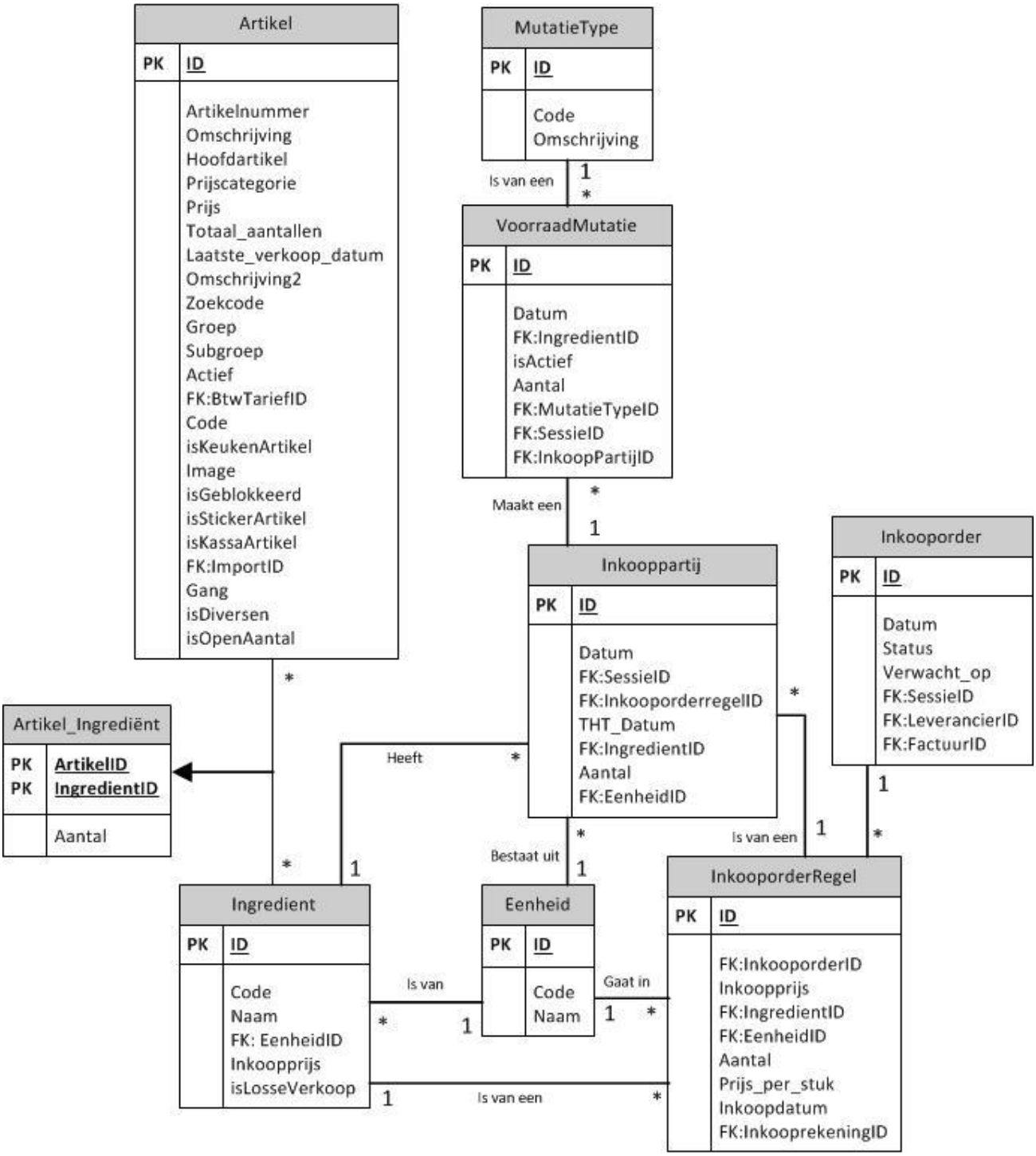
Met het opstellen van een klassendiagram voor de nieuwe situatie moest rekening worden gehouden met de requirements (figuur 4.2.2.1). Een aantal interessante punten waren:

1. Onderscheid maken tussen *inkoopartikelen*, *verkoopartikelen* en artikelen die één op één doorverkocht worden (zoals een blikje cola).
2. Inzicht verschaffen in de verkoopbare *eenheden* van alle *ingrediënten*.
3. *Voorraad* weergeven per THT datum als er volgens FIFO is verbruikt.

1. Ingrediënten kunnen onderdeel zijn van één of meer verkoopartikelen. Een relatie tussen de al bestaande artikel tabel en een toegevoegde ingrediënt tabel zorgt voor de vervulling van de requirement. Omdat het gaat om een veel op veel relatie is er een koppeltabel nodig om bij te kunnen houden welke ingrediënten verwerkt zitten in een verkoopartikel en daarbij ook de hoeveelheid.

2. Bij een ingrediënt dienen een aantal gegevens bijgehouden te worden. Zo heeft een ingrediënt een code, naam, inkoopprijs en een verwijzing naar een eenheid. Door een eenheid op te slaan bij een ingrediënt kan er in het later ontwikkelde systeem overal waar gegevens van het ingrediënt nodig zijn ook de eenheid worden getoond. Hier is voor gekozen om de tweede requirement op een valide manier te vervullen. Daarbij is nog wel nodig om bij te kunnen houden wat dan de aantallen zijn van de eenheden. Om dit op te lossen is de volgende databasestructuur bedacht.

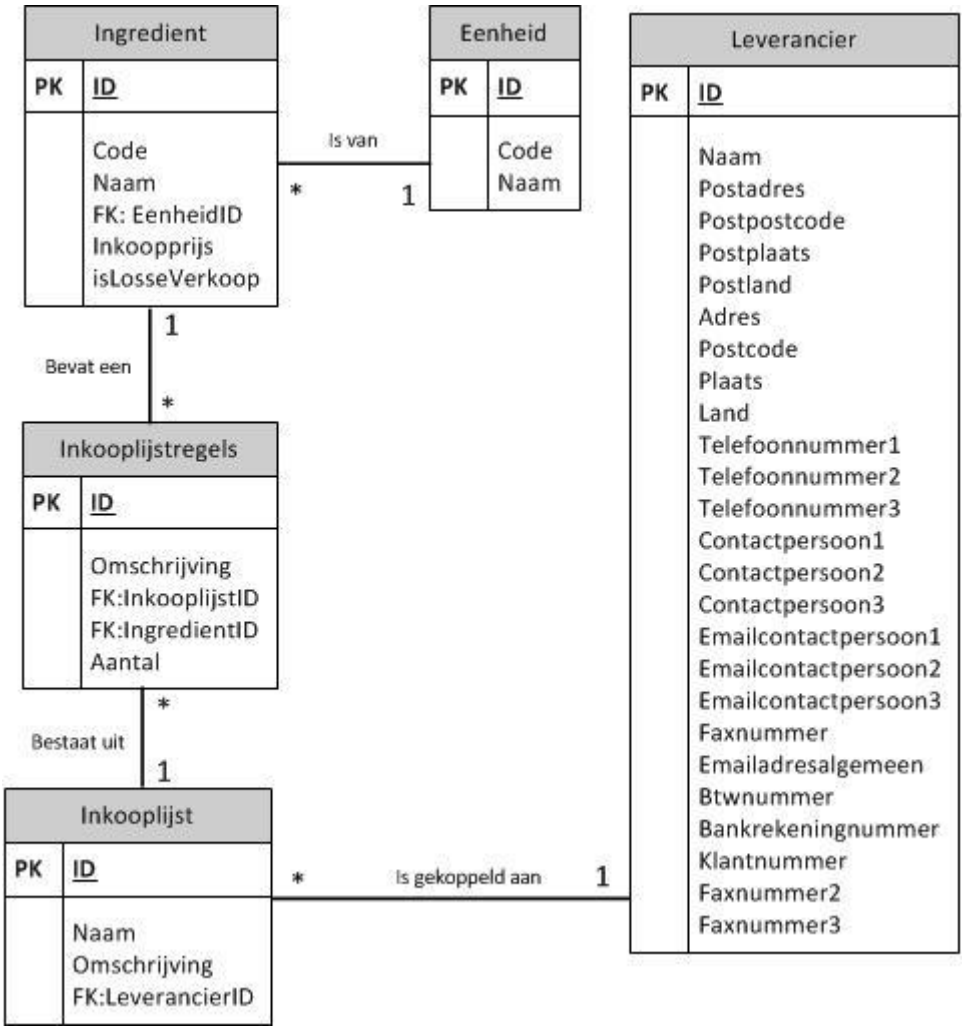
3. Ingrediënten moeten ingekocht kunnen worden om bij binnenkomst toegevoegd te kunnen worden aan de voorraad. De voorraad moet op THT datum bij te houden zijn. Dit is opgelost door per ingrediënt uit een order vast te leggen hoeveel er binnenkomt, van welke order het ingrediënt afkomstig is en wat de THT datum is. Dit wordt opgeslagen in de tabel inkooppartij. Het is op deze manier ook mogelijk om van de verschillende ingrediënten één of meerdere partijen te hebben van verschillende houdbaarheidsdata. Het berekenen van de totale voorraad gebeurd door alle veranderingen (mutaties) bij te houden van elk ingrediënt. De voorraad zou op te maken kunnen zijn uit de tabel inkooppartij, namelijk het optellen van alle aantallen van de inkooppartijen. Er is echter gekozen voor een oplossing die wat meer informatie geeft over de totstandkoming van de voorraad door te kijken naar de verschillende mutaties op een ingrediënt. Door een berekening uit te voeren van alle mutaties kan de voorraad berekend worden en is er precies vastgelegd welke mutaties er doorgevoerd worden op een ingrediënt.



Figuur 4.2.2.1: Klassendiagram database - afdekken van de 3 requirements

Bij het opstellen van de use cases was naar voren gekomen dat in het systeem gebruik gemaakt moest gaan worden van inkooplijsten. Om dit mogelijk te maken is stukje van het klassendiagram in figuur 4.2.2.2 nodig.

De eerder besproken tabellen ingrediënt en eenheid worden hier weergeven. Ingrediënten kunnen ingekocht worden door middel van het plaatsen van een order waarin zij gekoppeld zijn aan een inkooporderregel. Het principe van een inkooplijst werkt ongeveer hetzelfde. Zo moet het mogelijk zijn om een inkooplijst aan te maken die ook bestaat uit verschillende inkooplijstregels. Deze regels zijn gevuld met gegevens voor een bestelling, namelijk een ingrediënt, aantal en de lijst waar de regel bij hoort. Zo kan er bij het aanmaken van een order later gekozen worden om gebruik te maken van een Inkooplijst die bestaat uit verschillende van te voren opgestelde orderregels. Een inkooplijst is gekoppeld aan de al bestaande tabel Leverancier. Dit zorgt ervoor dat een inkooplijst maar aan één leverancier gekoppeld kan worden. Dit was immers de wens bij het bespreken van dit nieuwe idee.

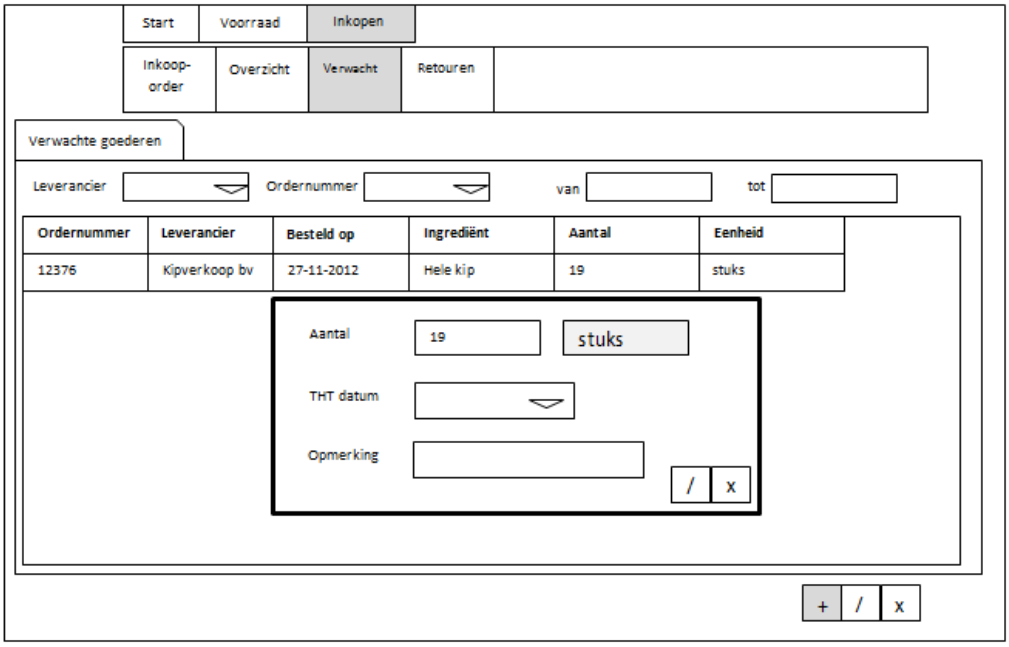


Figuur 4.2.2.2: Klassendiagram database inkooplijsten

Een volledig overzicht van het klassendiagram is toegevoegd als bijlage in deel 2 van het afstudeerdossier.

4.2.3 Ontwerp van schermen

Om bij het ontwikkelen direct de schermen op te kunnen zetten zijn er in de oriëntatiefase ook prototypes opgezet voor schermen (figuur 4.2.3.1). Tijdens het ontwerpen was het belangrijk dat de standaard van Delivery Manager aangehouden zou houden. Door de huidige schermen van Delivery Manager te bekijken is inspiratie opgedaan voor de schermen en is de standaard duidelijk geworden. Tijdens het ontwerpen is er meerdere malen overleg geweest met de bedrijfsmentor. Het doel hiervan was om samen te valideren of de schermen consistent zijn met de beschreven use cases en of de standaard gewaarborgd blijft. Alle ontwerpen zijn te vinden in bijlage 5 in deel 2 van het afstudeerdossier.

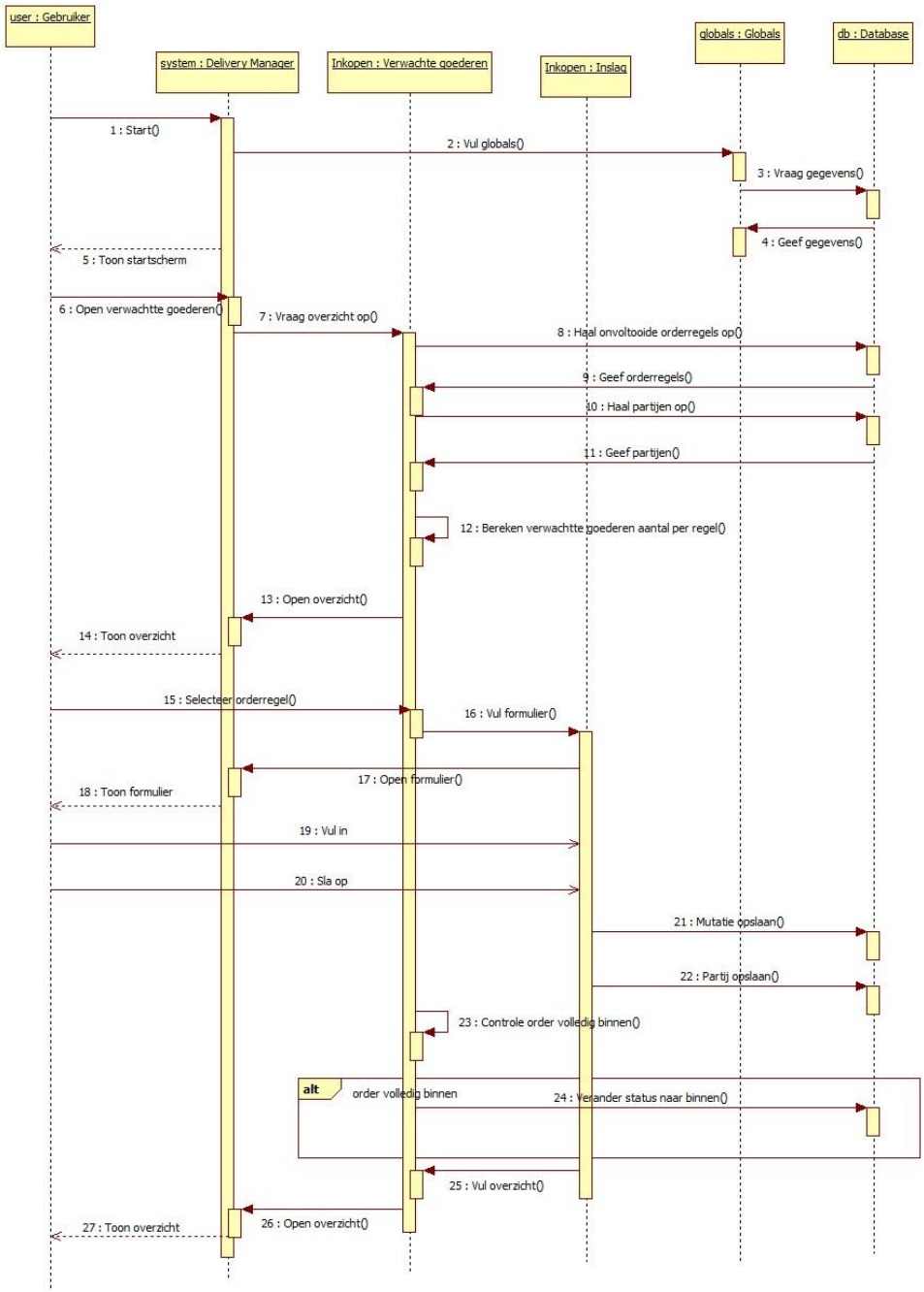


Figuur 4.2.3.1: Schermontwerp inslag van ingrediënten

4.2.4 Sequentiediagrammen

Het doel van de sequentiediagrammen is het vertalen van de requirements (wat) naar functionaliteit (hoe). De diagrammen geven de boodschappen tussen objecten weer. Veel van deze boodschappen zullen worden gemodelleerd gaan worden tot operaties in een class.

In de diagrammen is stap voor stap te zien wat er gebeurt bij een bepaalde input van de gebruiker. Een terugkerende operatie is het laden van de Globals bij het opstarten van Delivery Manager. De Globals zijn gegevens die bestaan uit stamgegevens, lokale instellingen en netwerk instellingen. Deze gegevens worden op meerdere plaatsten in de applicatie gebruikt. Om performance redenen worden deze eenmalig ingeladen bij het opstarten van het Delivery Manager. Als de Globals opgevraagd en geladen zijn wordt het startscherm getoond aan de gebruiker die vervolgens acties kan uitvoeren. De diagrammen vormen samen een vertaling van de use cases (wat moet het systeem kunnen) naar hoe het systeem op deze acties reageert. Op de volgende pagina is een voorbeeld te zien (figuur 4.2.4.1) van een sequentie diagram die het inslagproces van een ingrediënt beschrijft.



Figuur 4.2.4.1: Sequentiediagram inslag van goederen

Na het verplicht inladen van de Globals kan de gebruiker er voor kiezen naar de verwachte goederen te kijken in het overzicht. Zoals te zien is hier interactie voor nodig met de database. Het systeem heeft de verschillende orderregels en de partijen nodig. Met deze gegevens kan het systeem per orderregel berekenen hoeveel er binnen is en hoeveel er nog verwacht wordt. Na deze berekening gedaan te hebben kan het scherm getoond worden aan de gebruiker. De gebruiker kan vervolgens een regel selecteren uit het overzicht en het formulier invullen. Als deze naar wens is ingevuld drukt de gebruiker op de knop opslaan en schrijft het systeem de aanpassingen weg. Er wordt een nieuwe partij aangemaakt, en er wordt een nieuwe mutatie toegevoegd. Wanneer dit gedaan is toont het

systeem het overzicht opnieuw, maar met de doorgevoerde wijzigingen van hierboven. Het is belangrijk te controleren of een order compleet is of niet. Een order kan namelijk in meerdere keren binnenkomen of kan bestaan uit verschillende THT data waardoor er meerdere partijen aangemaakt dienen te worden. Pas zodra er helemaal geen goederen meer verwacht worden kunnen er geen partijen meer worden aangemaakt.

Alle andere sequentie diagrammen zijn te vinden als bijlage(8) in deel 2 van het afstudeerdossier.

4.2.5 Documentatie

Gaande het ontwikkelen zijn de eerder genoemde ontwerpen steeds bijgewerkt. Dit met het doel de nieuwe inzichten te verwerken en zo te zorgen voor consistentie tussen de ontwerpen en het ontwikkelde stuk software.

4.2.6 Productbacklog

De productbacklog bevat alle taken die uitgevoerd moeten gaan worden om de functionaliteiten van de modules toe te voegen aan Delivery Manager. Door het opzetten van use cases en scherm ontwerpen is duidelijk *wat* er ontwikkeld moet worden en door de toevoeging van de sequentiediagrammen is ook duidelijk *hoe* dit moet gebeuren. De productbacklog is niet definitief. Als gevolg van het opleveren van prototypes zullen de wensen veranderen. Hierdoor worden er nieuwe functionaliteiten toegevoegd of veranderingen aan functionaliteiten aangebracht. De complete productbacklog bij aanvang van de eerste sprint is te vinden als bijlage(9). Per sprint is ook een sprintbacklog opgezet om de vooruitgang te monitoren en te toetsen of alle opgestelde eisen worden meegenomen tijdens het ontwikkelen.

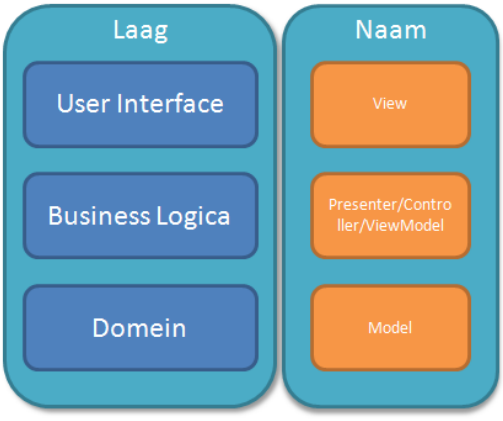
4.3 Eerste sprint: Stamgegevens

Het doel van de eerste sprint was het opzetten van de *database* en het ontwikkelen van de *stamgegevens*. Om dit te bereiken is er in deze sprint eerst kennis gemaakt met Delivery Manager, Visual Basic, Windows Presentation Foundation en het MVVM design pattern door een testproject aan te maken. Ook moesten eerst alle ontwerpen van de schermen opgezet worden in Delivery Manager.

4.3.1 Kennis vergaren

Allereerst was het noodzakelijk in te lezen en te verdiepen in Visual Basic, een voor mij onbekende programmeertaal. Al snel bleek dat Visual Basic niet heel anders is als bijvoorbeeld C# wat voor mij wel bekend is. Wat opvalt, is dat de syntax anders is. Het ontwikkelen van Delivery Manager gebeurt in Visual Studio. Dit programma biedt de mogelijkheid tot ontwikkelen met Windows Presentation Foundation (WPF). WPF maakt het ontwerpen van besturingselementen (denk aan knoppen, keuzelijsten en dialoogvensters) mogelijk. Het visuele deel van de applicatie wordt opgezet in XAML (wat staat voor Extensible Application Markup Language). XAML is een declaratieve taal van Microsoft die gebaseerd is op XML. De taal wordt gebruikt om gestructureerde waarden en objecten te initialiseren. XAML wordt uitgebreid gebruikt in het.NET Framework, in het bijzonder in WPF en Silverlight. Daarin dient het als taal voor de gebruikersinterface. In WPF wordt XAML typisch gebruikt om grafische interfaces te beschrijven. Alles wat aangemaakt is met XAML kan met .NET talen uitgedrukt worden, waarvan Visual Basic er één is.

Delivery Manager wordt ook ontwikkeld door middel van het MVVM (Model - View- ViewModel) design pattern. MVVM is specifiek bedoeld voor WPF. Het is een variant op het oudere MVC (Model - View- Controller) pattern. Ze gaan beide uit van hetzelfde principe, namelijk het scheiding van de userinterface, de business logica en het domeinmodel in drie lagen, zie figuur 4.2.1. Iedere laag heeft zijn eigen verantwoordelijkheid en heeft alleen kennis van de laag die zich onder hem bevindt. De UI (View) laag bevat alle grafische componenten zoals tekstvelden, knoppen en lijsten. De domein laag (Model) bevat de classes met daarin de objecten. De laag daartussen, de logica laag (ViewModel), koppelt de andere twee aan elkaar en bevat de logica.



Figuur 4.2.1: De drie verschillende lagen in MVVM

Als een applicatie op deze manier gestructureerd is kunnen lagen later aangepast worden zonder grote impact op de rest van het systeem. Het gebruik van deze structuur samen met WPF zorgt er voor dat er gebruik kan worden gemaakt van ‘bindings’.

Binding zorgt ervoor dat de elementen op het scherm gevuld worden met gegevens die ergens anders vandaan komen. Met andere woorden: een tekstveld bevat een tekst die niet in de definitie van het scherm staat maar die een andere oorsprong heeft. Om dit te bereiken is het volgende in WPF van belang.

Als eerste is er de content van de control die wordt bijgehouden in een property. Bij een tekstveld is dit bijvoorbeeld een stuk tekst, bij een vinkje is dit een boolean en bij een knop een commando. Deze properties kunnen gekoppeld worden met binding. Een binding geeft aan hoe de data opgehaald moet worden. De data zelf kan van alles zijn. In XAML ziet een binding van een tekstveld er als volgt uit:

```
<TextBox Name="Naam"  Text="{Binding ProductNaam}" />
```

In plaats van in de code een tekst te plaatsen in het tekstveld, vertelt de WPF nu dat er ergens in de Datacontext (klasse) een property is met de naam ProductNaam. De inhoud daarvan moet in dit tekstveld te zien zijn. De data staat in de Business Logica, in MVVM de ViewModel. Deze data kan door de ViewModel opgehaald worden uit de domein laag maar kan ook ingevoerd worden door de gebruiker.

Testproject

Met de kennis die ik had vergaard, heb ik geprobeerd een testproject op te zetten om deze te beheersen. Het testproject was een simpel project waarmee ik data kon ophalen uit een database en vervolgens deze data kon tonen in een overzicht. De volgende stap was het toevoegen van data, aanpassen van data en verwijderen van data. Met deze kennis kon ik gaan beginnen aan het ontwikkelen van de stamgegevens. Allereerst was het zaak de schermen te ontwikkelen in XAML.

4.3.2 Schermen ontwikkelen

Voor de ontwerpen van de schermen is gebruik gemaakt van de toolbox in Visual Studio. De code is vervolgens handmatig aangepast omdat afstanden en formaten precies moesten kloppen met de Delivery Manager standaard. Ook het geven van logische namen en het maken van bindings moet in de code zelf gebeuren.

De schermen zijn na het ontwikkelen gevalideerd door de bedrijfsmentor. Hieruit bleek dat de opbouw van de schermen en de positionering van de elementen op het scherm in orde was maar dat de formaten niet altijd klopte met de andere schermen van Delivery Manager. Deze punten zijn gewijzigd en het opzetten van de schermen was hiermee afgerond.

4.3.3 Opzetten van de database

De databases van Pamicon betreffen SQL Server Databases. Hiervoor dient gebruikt gemaakt te worden van Microsoft SQL Server 2008. Voor het ontwikkelen wordt er bij Pamicon gebruik gemaakt van een ontwikkeldatabase die gelijk is aan de databases die gebruikt worden bij de klanten. Wanneer het project volledig afgerond is kan bij de klanten de database worden aangepast naar deze ontwikkeldatabase zodat de nieuwe modules van Delivery Manager ook bij de klanten kunnen werken. Het klassendiagram wat ik eerder ontworpen had heb ik verwerkt in de database. Alle nodige tabellen zijn gemaakt met behulp van de designer die SQL Server biedt voor het aanmaken en beheren van tabellen. Hier onder een voorbeeld van de tabel Inkooppartij.

PAMSERVER9.vvh_...bo.Inkooppartij*

Column Name	Data Type	Allow Nulls
ID	int	
Datum	date	
SessieID	int	
InkooporderregelID	int	
THT_datum	date	
EenheidID	int	
IngrediëntID	int	
Aantal	float	
Opmerking	varchar(255)	

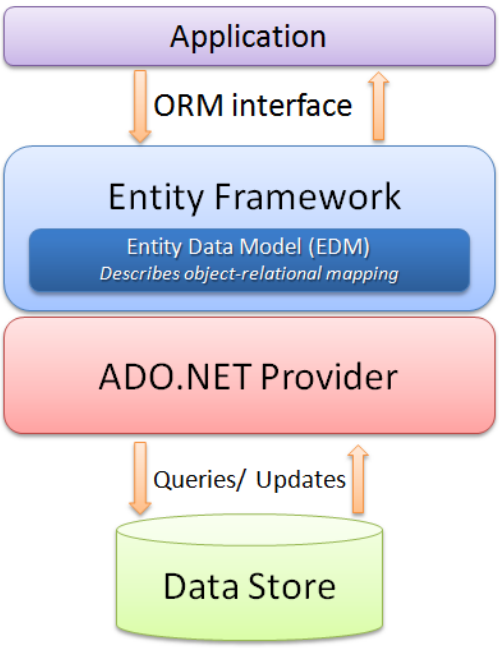
Figuur 4.3.3.1: Tabel Inkooppartij

Constraints

Normaliter worden er aan de databasekant ook constraints toegevoegd. De werkwijze van Pamicon wijkt hier echter af van hetgeen ik gewend ben. Zij kiezen ervoor om deze acties niet door de database af te handelen maar dit af te handelen aan de applicatiekant. In deze werkwijze ben ik meegegaan ondanks dat mijn voorkeur ligt bij het opzetten van constraints op databaseniveau. De reden hiervoor is dat er tijdens het ontwikkelen gelet moet worden op het toevoegen van constraints en dit zal dan ook nog eens op meerdere plekken moeten gebeuren. Dit leidt tot meer werk en is foutgevoelig.

Entity Framework

Delivery Manager maakt gebruikt van een .NET toepassing genaamd Entity Framework. De doelstelling is het verminderen van de hoeveelheid code en verbeteren van de onderhoudbaarheid voor objectgeoriënteerde toepassingen. Het Framework maakt het mogelijk de toegang tot de gegevens toepassingen te creëren door het programmeren tegen een conceptueel applicatie model (Entity Data Model) in plaats van programmeren direct tegen een relationele opslag (Data Store), zie figuur 4.3.3.2. De provider zorgt voor de koppeling tussen de twee lagen.



Figuur 4.3.3.2: Entity Framework

In Visual Studio kan het conceptueel model van een database automatisch gegenereerd worden. Andersom kan er van het model een database gegenereerd worden. Bij gebruik van het Entity Framework worden er queries uitgevoerd met LINQ (Language Integrated Query). Hiermee worden gegevens opgehaald om deze als objecten te kunnen manipuleren.

4.3.4 Stamgegevens

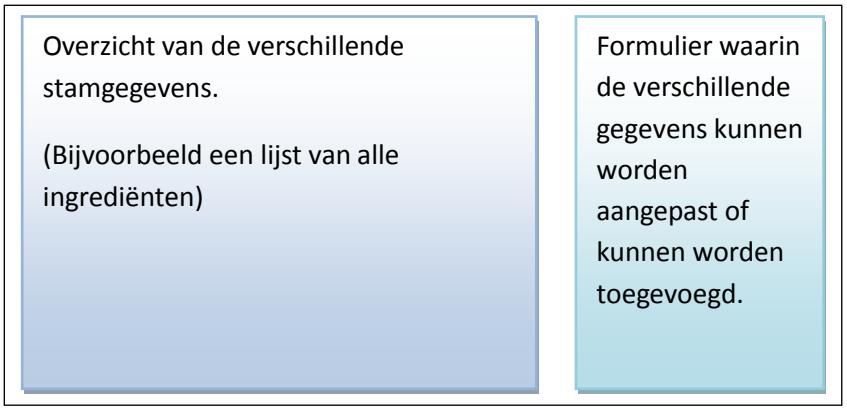
De stamgegevens zijn gegevens die gebruikt worden op meerdere plekken in de applicatie en die beheerd kunnen worden door de gebruiker zelf. Voorbeelden zijn verkoopartikelen en ingrediënten.

De stamgegevens die toegevoegd moesten worden waren:
- Ingrediënt
- Eenheid
- Inkooplijst (ordertemplate)

Ook moest er een bestaand stamgegeven worden aangepast, namelijk:
- Artikel

In de situatie zoals die door Pamicon is opgezet gebruiken alle stamgegevens dezelfde basisclass. Een stamgegeven is een afgeleid object. Omdat alle stamgegevens dezelfde operaties bevatten is er hier gebruik gemaakt van polymorfisme. Aan de hand van het stamgegeven worden de operaties ingevuld en kunnen er operaties worden toegewezen aan dat specifieke stamgegeven. Door deze oplossing wordt voorkomen dat voor elk van de stamgegeven het geheel opnieuw ontwikkeld moet worden. Dit zou ook zorgen voor redundantie met als gevolg dat bij het aanpassen of toevoegen van de code er op meerdere plekken wijzigingen gedaan moeten worden.

Door het gebruik van één class is er ook één User Interface (User Control). De opzet van de lay-out ziet er als volgt uit:



Figuur 4.3.4.1: De standaard lay-out van de stamgegevens

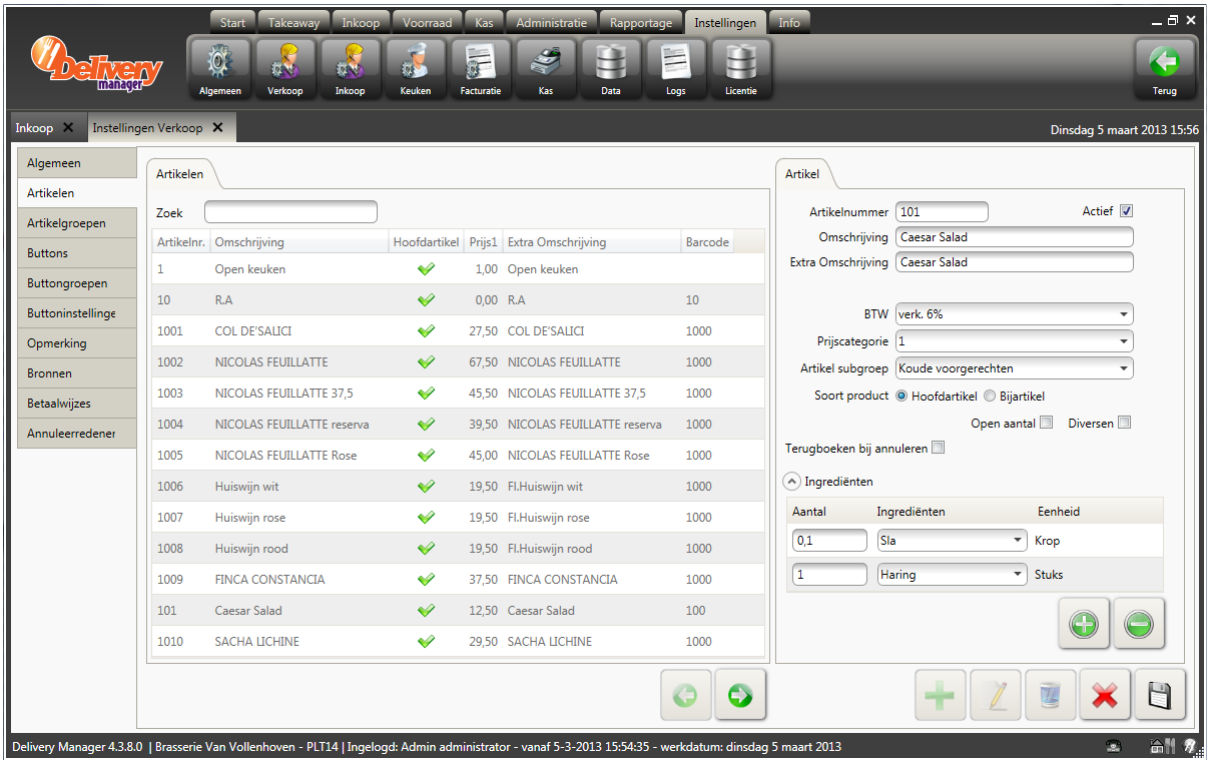
Links is een overzicht te zien in de vorm van een lijst. Rechts wordt er een formulier getoond waarmee er nieuwe stamgegevens toegevoegd of aangepast kunnen worden. Het formulier bestaat uit tekstvelden en selectieboxen die door de gebruiker kunnen worden ingevuld. Omdat de gegevens per stamgegeven verschillen is rechterdeel van het scherm is een losstaande User Control.

Alle stamgegevens konden opgezet worden volgens de beschreven opzet met uitzondering van inkooplijsten en artikelen. Beide hebben dezelfde toevoeging, namelijk dat er een *lijst* met gegevens gekoppeld moet worden aan het stamgegeven.

- Inkooplijsten kunnen aangemaakt worden. Hierbij worden een aantal gegevens ingevuld zoals een naam, een opmerking maar vooral ook belangrijk de leverancier. De lijst wordt opgezet om te gebruiken voor één bepaalde leverancier. Aan de lijst moeten ingrediënten toegevoegd worden waarvan er een bepaald aantal gekoppeld zal worden aan de bestellijst.

- Artikelen bestaan uit verschillende ingrediënten. Van de ingrediënten moeten de stamgegevens worden toegevoegd door de gebruiker. De gebruiker moet de mogelijkheid krijgen om één of meerdere ingrediënten te koppelen aan een artikel door deze toe te voegen aan een *lijst* (zie figuur 4.3.4.2: Overzicht van stamgegevens artikelen).

De objecten wijken af van de basis klasse van de stamgegevens doordat er een lijst getoond moet worden en er hier aanpassingen gedaan kunnen worden door de gebruiker. Er zijn meer operaties nodig dan dat er van de standaard stamgegevens geërfd wordt. Ook de User Control wijkt af van de andere stamgegevens.



Figuur 4.3.4.2: Overzicht van stamgegevens artikelen.

4.3.6 Plannen volgende sprint

De toegevoegde stamgegevens zijn getoond en besproken met de bedrijfsmentor. Hieruit kwam naar voren dat alles naar wens ontwikkeld was. In de volgende sprint ben ik mij gaan richten op een nieuw stuk van de applicatie, namelijk de inkoopmodule. Voor aanvang van deze sprint zijn alle taken betreffende inkoopmodule ingepland en verwerkt tot een sprintbacklog. Zie hiervoor bijlage 9, sprintbacklog sprint 2.

4.4 De tweede sprint: Inkooporders

De tweede sprint stond in het teken van het invoeren van de inkooporders. Orders moeten geplaatst kunnen worden doormiddel van het toevoegen van een nieuwe order of door het gebruiken van bestaande inkooplijsten. Daarnaast moeten ze altijd te bewerken zijn. Wanneer ingevoerde orders geleverd worden moeten deze ingeslagen kunnen worden, wat resulteert in partijen. Verder moeten ingrediënten geretourneerd kunnen worden.

4.4.2 Overzicht van inkooporders

De applicatie moet de mogelijkheid bieden een overzicht te kunnen tonen van alle inkooporders. Het scherm was al opgezet in de eerste sprint, de logica (ViewModel) nog niet.

Het overzicht bestaat uit een lijst van de inkooporders die in het scherm (user control) wordt getoond in een datagrid. De data die getoond moet worden in de datagrid moet verzameld worden in de ViewModel. Door de ViewModel te koppelen aan een user control in de Resource Dictionary weet de user control welke ViewModel voor hem verantwoordelijk is.

Om de gegevens op te halen is er in de ViewModel een methode die de gegevens verzamelt en verwerkt tot een collectie van data. Deze methode wordt aangeroepen in de constructor zodat hij uitgevoerd wordt bij het aanroepen van het object. De data is gekoppeld aan de datagrid in de user control door middel van binding. Een property van het datatype Observerable Collection bevat de gegevens van alle inkooporders. Een Observerable Collection is vergelijkbaar met een List. Een Observerable Collection heeft de mogelijkheid om een binding te hebben naar de User Control, dit is niet mogelijk met een List.

Omdat de collectie van gegevens gevuld moet worden met gegevens van verschillende objecten is er een class aangemaakt die deze informatie kan tonen (figuur 4.4.2.1). Het object inkooporder bevat in sommige gevallen enkel het id wat verwijst naar een object. De class zorgt ervoor dat niet de verwijzing getoond wordt maar een relevant attribuut van het object waar naar verwezen wordt. Een voorbeeld is een leverancier. Een inkooporder bevat een id van de leverancier. In de class kan aan de hand van het id het gehele object leverancier worden opgehaald. Hierdoor kan er in het overzicht een naam getoond worden van de leverancier i.p.v. het id.

```
254         Public Class clsInkopen
255             Inherits WorkspaceViewModel
256             Private _Inkoop As Inkooporder
257             Private _Leverancier As leveranciers
258             Public Property Inkoop As Inkooporder
259                 Get
260                     Return _Inkoop
261                 End Get
262                 Set(ByVal value As Inkooporder)
263                     _Inkoop = value
264                     NotifyPropertyChanged("Inkoop")
265                     Leverancier = (From l In Globals.Leveranciers Where l.ID = Inkoop.LeverancierID).FirstOrDefault
266                 End Set
267             End Property
268             Public Property Leverancier As leveranciers
269                 Get
270                     Return _Leverancier
271                 End Get
272                 Set(ByVal value As leveranciers)
273                     _Leverancier = value
274                     NotifyPropertyChanged("Leverancier")
275                     NotifyPropertyChanged("LeverancierNaam")
276                 End Set
277             End Property
278             Public ReadOnly Property Type As String
279                 Get
280                     If Inkoop.isRetour = True Then
281                         Return "Retour"
282                     Else
283                         Return "Inkoop"
284                     End If
285                 End Get
286             End Property
287             Public ReadOnly Property LeverancierNaam As String
288                 Get
289                     If Leverancier Is Nothing Then
290                         Return ""
291                     Else
292                         Return Leverancier.naam
293                     End If
294 
295                 End Get
296             End Property
297         End Class
```

Figuur 4.4.2.1: Klasse voor verzameling van objecten

4.4.3 Aanmaken inkooporders

Dit scherm bestaat uit een formulier met algemene gegevens van de inkooporder en een overzicht (datagrid) van alle orderregels. In het overzicht en formulier met algemene gegevens van de order moet de gebruiker gegevens kunnen invoeren. Daarnaast moet de gebruiker regels toe kunnen voegen aan en verwijderen uit de lijst. Na het doorvoeren van wijzigingen moet de order opnieuw opgeslagen kunnen worden.

De ViewModel achter de User Control kan omgaan met het bewerken van bestaande orders en toevoegen van nieuwe orders. Als er een parameter (object inkooporder) wordt meegestuurd bij het aanmaken van het object weet het dat er een order bewerkt moet worden. De parameter wordt bepaald aan de hand van de inkooporder die de gebruiker wil aanpassen. Het gehele object van de inkooporder wordt meegeven waardoor het niet nodig de gegevens van de inkooporder op te vragen om het formulier met de algemene gegevens van de inkooporder te vullen.

Als er een parameter wordt meegegeven wordt er met behulp van een methode een lijst verzameld van de inkooporderregels. Wanneer de parameter leeg is wordt de inkooporder User Control leeg getoond aan de gebruiker.

4.4.4 Inkooporders a.d.h.v. inkooplijsten

Orders moeten in te voeren zijn door het gebruik van een inkooplijst. Tijdens het aanmaken van een inkooporder moet er een mogelijkheid zijn een lijst te selecteren. Indien er al een leverancier gekozen is bij de inkooporder, is het alleen mogelijk om lijsten te selecteren van deze leverancier. Als deze keuze nog niet is gemaakt en er een lijst wordt gekozen moet het systeem zelf de leverancier van de inkooplijst beschouwen als de leverancier van de inkooporder.

Voor deze functionaliteit wordt er in de User Control aan de rechterkant in het scherm een lijst getoond met de verschillende inkooplijsten. Elke van deze lijsten heeft een naam en een standaard leverancier. Door een bij elk van deze lijsten weergegeven importeerknop kan de gebruiker een lijst importeren naar de inkooporder.

Voor het vullen van de inkooporder met een inkooplijst is er een methode in de ViewModel. Deze wordt uitgevoerd op het moment dat een gebruiker een lijst importeert uit de verzameling met inkooplijsten. Bij de algemene gegevens van de lijst worden de regels opgehaald die de inkooplijst vormen. Vervolgens worden deze regels toegevoegd aan de inkooporder. Waar als laatst nog op gecontroleerd moet worden is of er een leverancier geselecteerd is. Als dit niet het geval is wordt de geselecteerde leverancier voor de inkooporder vastgezet op de leverancier van de inkooplijst. Hierdoor verandert ook de keuze op te importeren inkooplijsten.

4.4.5 Partijen aanmaken

Geleverde orders moeten worden ingeslagen. Dit resulteert in verschillende partijen met een houdbaarheidsdatum. Elk van deze regels worden getoond in het overzicht wanneer deze nog niet volledig is ingeslagen. De gebruiker moet de regels in zijn geheel of gedeeltelijk in kunnen slaan.

De User Control is zo opgezet dat de gebruiker kan zoeken in de lijst met verwachte goederen. Dit is nodig omdat het aantal goederen wat besteld kan worden al snel op kan lopen bij een grote onderneming. Condities als een leverancier, datum en ordernummer kunnen zo invloed uitoefenen op de data die getoond moet worden in het overzicht. Vanuit het overzicht kan de gebruiker een regel selecteren om van deze regel goederen in te slaan. Hierbij kan een aantal opgegeven worden en een THT datum.

De logica in de ViewModel moet bij het filteren van de gegevens om kunnen gaan met de verschillende condities. De methode die verantwoordelijk is voor het ophalen van de juiste gegevens filtert de gegevens op datum wanneer de methode aangeroepen wordt. Op het moment dat deze aangeroepen wordt, wat gebeurt bij het laden van het scherm, zijn de parameters begin en einddatum al ingevuld met een standaard waarde. De andere condities zijn nog onbekend en de methode zal opnieuw aangeroepen worden op het moment dat één van deze condities veranderd. De collectie van gegevens zal veranderen en hierdoor wordt aan de User Control duidelijk gemaakt dat de gegevens opnieuw ingeladen moeten worden.

Voor het daadwerkelijk inslaan van goederen is een andere methode verantwoordelijk. Deze wordt uitgevoerd op het moment dat de gebruiker de partij aanmaakt door hem op te slaan. Voordat de gebruiker een partij aan kan maken wordt er eerst gecontroleerd of het gekozen aantal überhaupt wel in te slaan is.

Dit wordt berekend door de aantallen van de partijen die al aangemaakt zijn van deze order bij elkaar op te tellen. Als het ingevoerde aantal niet het complete te verwachte aantal betreft wordt er een partij aangemaakt met het ingevoerde aantal. Hierdoor ontstaan er deelpartijen wat uiteindelijk resulteert in een compleet ingeboekte order.

4.4.6 Leveranciers

De gebruikers van Delivery Manager doen hun inkopen bij verschillende leveranciers. Orders en inkooplijsten behoren toe aan één bepaalde leverancier. Leveranciers zijn net als ingrediënten en artikelen een stamgegeven. Die optie tot het beheren van leveranciers is niet terug te vinden onder de stamgegevens maar in de inkoopmodule. Dit is naar wens van de bedrijfsmentor.

4.4.7 Retourneren

Het doen van een retourorder is vergelijkbaar met het maken van een inkooporder. De opzet van het scherm is ook vergelijkbaar met dat van het maken van een inkooporder. Het verschil is echter dat er bij het retourneren van een order wel een mutatie gedaan moet worden op de voorraad terwijl dit bij het aanmaken van een inkooporder niet gebeurd.

In de User Control is de gebruiker verplicht om een leverancier te kiezen waar de goederen naar geretourneerd worden. Hierbij kan er een algemene opmerking worden ingevuld. Verder kan er net als bij het inkoopscherm een lijst worden opgezet van ingrediënten die geretourneerd moeten worden. Als dit lijstje opgeslagen wordt zal er door de ViewModel een aantal acties uitgevoerd worden.

In eerste instantie zal er een mutatie aangemaakt worden voor elk ingrediënt uit de lijst. Daarnaast zal een andere functie de gegevens van retourorder wegschrijven naar de database. Voor het opslaan van deze data zijn verschillende manieren mogelijk. Er is bij een order en een retourorder sprake van overeenkomsten in de classes. Om de data zo efficiënt mogelijk op te slaan is er gekeken naar de verschillende mogelijkheden. Er zijn drie verschillende manieren waarop deze situatie aan de databasekant kan worden opgelost.

1. Iedere klasse een aparte tabel
2. Iedere concrete klasse in de hiërarchie een tabel
3. Alle klassen in de hiërarchie in één enkele tabel

Mijn voorkeur ging uit naar optie 1. Dit omdat er op deze manier geen redundantie optreedt en ook geen ruimte wordt verspild door niet ingevulde attributen. De voorkeur van de bedrijfsmentor ging echter uit naar de derde optie. Deze oplossing wordt door de gehele operationele database gebruikt en door deze manier te gebruiken blijft de implementatiewijze consistent. Wel is het gevolg dat hier gegevens redundant worden opgeslagen en dat er ruimte wordt verspild door het niet volledig vullen van de records in de tabel.

4.4.8 Bespreking prototype

Als afsluiting van deze sprint is er een oplevermoment geweest met de bedrijfsmentor en de medewerker van de marketing. Bij het doorlopen van het prototype zijn de volgende nieuwe *wensen* ontstaan:

Inkoopmodule:
- Ingrediënten moeten gekoppeld worden aan een btw categorie. Bij het aanmaken van een inkooporder moet de huidige categorie worden opgeslagen om zo later nog precies te kunnen zien hoeveel btw er is betaald bij die inkooporder.

- Een stamgegeven reden toevoegen. Hiermee moet bereikt worden dat er standaard redenen aangemaakt kunnen worden en tijdens het invullen van een reden bij het uitvoeren van een actie er uit een standaard selectie gekozen moet worden. Dit heeft tot gevolg dat er een nieuwe tabel toegevoegd moet worden aan de database, er een User Control opgezet moet worden met daarbij een ViewModel.

- Bij verwachte goederen tonen hoeveel er besteld is, hoeveel er binnen is en hoeveel er nog verwacht wordt. De User Control moet aangepast worden en er zal een berekening gedaan moeten worden in de logica.

- Inkooporders bij annuleren niet verwijderen uit de database om inzicht te houden in geannuleerde orders.

- Verandering van de status van alle orderregels als deze afgehandeld zijn. Hierdoor kan uiteindelijk de gehele order als volledig ontvangen worden beschouwd.

Voorraadmodule:
- Afboeken van een verkoopartikel. Een verkoopartikel in z'n geheel kunnen afboeken. Dit houdt in dat de voorraad van alle ingrediënten gemuteerd wordt.

4.4.9 Plannen volgende sprint

De volgende sprint stond in het teken van de voorraadmodule. Bij deze sprint zijn ook de eerder beschreven inzichten verwerkt. De complete sprintbacklog is te vinden als bijlage in deel 2 van het afstudeerdossier (bijlage 9, sprintbacklog sprint 3).

4.5 De derde sprint: Voorraad

Deze sprint stond in het teken van de voorraad module. Deze module moet de gebruiker de mogelijkheid geven om per ingrediënt de voorraad en de mutatiedetails te bekijken. Ook moeten er correcties doorgevoerd kunnen worden naar aanleiding van tellingen. Ook moet de gebruiker een ingrediënt kunnen afboeken, wat ook resulteert in het doorvoeren van een mutatie.

Deze sprint is als eerst begonnen met het compleet maken van de inkoopmodule door de nieuwe wensen door te voeren die ontstaan waren bij het bespreken van het prototype van de inkoopmodule.

4.5.1 Nieuwe inzichten inkoopmodule

De ideeën betreffende de inkoopmodule die ontstaan zijn bij het bespreken van de tweede sprint zijn eerst verwerkt.

- Het stamgegeven BTW wat al bestaat in Delivery Manager moet een koppeling krijgen met ingrediënt. Hiervoor moest het stamgegeven ingrediënt aangepast worden. Zowel de User Control, ViewModel en databasetabel zijn veranderd. In de User Control kan de gebruiker een BTW percentage selecteren. In de database wordt er bij elk ingrediënt een verwijzing weggeschreven naar het bijbehorende BTW percentage.

- Voor het nieuwe stamgegeven Reden is er een User Control aangemaakt en een ViewModel. Daarnaast is er in de database een tabel toegevoegd waar deze stamgegevens opgeslagen worden.

- Voor een gedetailleerder beeld bij de verwachte goederen zijn er veranderingen doorgevoerd op het niveau van de User Control en ViewModel. De datagrid in de User Control heeft twee nieuwe kolommen gekregen. De ViewModel zorgt ervoor dat deze kolommen gevuld worden met het aantal goederen wat nog verwacht wordt en het aantal wat al binnen is.

- Voor het behouden van de gegevens van de inkooporders is een wijziging gedaan op het niveau van de database en de ViewModel. Een kolom met een bit waarde wordt aangepast bij het annuleren van een order.

- Ook het bijhouden van een status heeft veranderingen te weeg gebracht op databaseniveau. Een kolom status in de tabel inkooporderregel wordt gevuld aan de hand van het te verwachten aantal. Als alle orderregels volledig binnen zijn kan ook de status van de order worden aangepast en is het duidelijk dat de order volledig ontvangen is.

4.5.2 Voorraad overzicht

De eerste toevoeging voor deze module was het opzetten van een overzicht van de voorraad per ingrediënt. De voorraad wordt per ingrediënt berekend met behulp van de mutaties. Van elk ingrediënt moet ook de laatste besteldatum in het overzicht worden getoond. Ook moet de gebruiker vanuit dit scherm ingrediënten kunnen afboeken. Dit kan bijvoorbeeld gewenst zijn als van een ingrediënt de THT datum is verstreken.

In de User Control wordt een overzicht getoond van de verschillende ingrediënten. Aan de rechterkant wordt een formulier getoond wat dient om ingrediënten af te boeken van de voorraad. Tevens is er een knop te vinden die door zal verwijzen naar de mutatiedetails van het ingrediënt. Om het vinden van het juiste ingrediënt makkelijker te maken voor de gebruiker is er ook een zoekveld in te vullen in dit scherm. Zo kan de gebruiker gericht op ingrediënt zoeken door de naam in te voeren.

De ViewModel bij de User Control is verantwoordelijk voor het tonen van de voorraad per ingrediënt en het afhandelen van de afboeking. Bij het aanmaken van het scherm wordt er een methode aangeroepen die voor elk ingrediënt wat bekend is in de stamgegevens de voorraad berekend. Een andere methode zorgt ervoor dat voor alle ingrediënten ook de laatste besteldatum opgehaald wordt. Deze gegevens kunnen opgevraagd worden door de laatste orderregel op te vragen van het ingrediënt. Wat er vervolgens nog afgehandeld moet worden is het opvragen van de kortst houdbare THT datum. Voor elk ingrediënt wordt de deze datum opgehaald uit de tabel inkooppartij. Als al deze gegevens zijn opgehaald kan het overzicht getoond worden.

Voor het zoeken in het overzicht is er een functie toegevoegd aan het ViewModel die ervoor zorgt dat er gezocht kan worden naar ingrediënten. Omdat het aantal ingrediënten een grote collectie van gegevens betreft is er een zoekfunctie geschreven die hiermee rekening houdt. De filterfunctie zoekt in de verzameling van ingrediënten naar de ingevulde gegevens in het zoek veld. Dit gebeurd alleen als de gebruiker is gestopt met tikken. Het daadwerkelijk doorvoeren van de methode die de lijst filtert wordt pas uitgevoerd als de gebruiker voor 0,3 seconde geen toets meer heeft ingedrukt.

Naast het ophalen van de juiste gegevens om te tonen in het overzicht is de ViewModel ook verantwoordelijk voor het doorvoeren van een afboeking. Een methode die aangeroepen wordt wanneer de gebruiker een afboeking opslaat maakt een mutatie aan van het ingrediënt.

4.5.3 Mutatiedetails per ingrediënt

Om het overzicht te houden in mutaties moet er per ingrediënt een mogelijkheid zijn een overzicht op te vragen van de mutaties. Dit kan dienen als een gedetailleerde log voor de gebruiker waar elke aanpassing van de voorraad terug te vinden is. In dit overzicht is te zien per dag en tijdstip hoeveel er gemuteerd is van het ingrediënt en hoe deze mutatie tot stand is gekomen.

De User Control voor het scherm toont aan de gebruiker een overzicht van alle mutaties van het ingrediënt. Het aantal van de mutaties blijft toenemen en daardoor ook het aantal records in de database. Dit zorgt ervoor dat het ophalen van de mutaties na verloop van tijd langer gaat duren. Daarom worden in het scherm altijd alleen de actieve mutaties getoond. Actieve mutaties zijn

mutaties die uitgevoerd zijn na de laatste telling. Deze filter op informatie kan aangepast worden in het scherm indien het gewenst is. Ook kan er gefilterd worden op datum en mutatietype.

De ViewModel bij die de logica achter het scherm voor zijn rekening neemt voert bij het aanmaken van het object een methode uit die een lijst van mutaties ophaalt. Verder wordt er pas een volgende operatie uitgevoerd als de gebruiker een filterconditie wijzigt. De lijst van mutaties zal dan opnieuw worden geladen waarbij de condities mee worden genomen.

4.5.4 Telling van ingrediënten

Om direct correcties door te voeren op de voorraad moet de gebruiker de mogelijkheid krijgen een telling door te voeren. De gebruiker moet de voorraad hier per ingrediënt aan kunnen passen en indien gewenst ook van meerdere ingrediënten tegelijk. In het scherm moeten ook de gevolgen zichtbaar zijn die optreden bij het doorvoeren van een telling. Om deze reden moet in het scherm getoond worden hoeveel er volgens het systeem nog op voorraad is en wat voor effect het doorvoeren van de telling heeft.

De User Control van het scherm heeft in grote lijnen dezelfde opzet als die van het voorraad overzicht. Er wordt een overzicht getoond van alle ingrediënten met het aantal dat er op voorraad is. Daarnaast kan bij elk van de ingrediënten in een tekst vak het getelde aantal ingevuld worden. Door hier een aantal in te vullen zal het systeem bij dat ingrediënt het telverschil weergeven.

De logica achter de User Control haalt per ingrediënt de mutaties op en berekent hiermee de huidige voorraad. Dit gebeurd bij het aanmaken van het object wanneer het scherm door de gebruiker geopend wordt. Bij elke input in het scherm wordt er een methode aangeroepen die voor het geselecteerde ingrediënt het telverschil berekent. Vervolgens wordt het telverschil getoond in het scherm. Het invullen en later aanpassen van het aantal getelde eenheden kan per ingrediënt herhaald worden. De ingevoerde waardes van alle ingrediënten blijven bewaard tot het moment het geheel opgeslagen wordt. De methode voor het opslaan filter als eerst de lijst van ingrediënten op een ingevoerde telling wat tot gevolg heeft dat er een lijst ontstaat van ingrediënten waar een mutatie voor aangemaakt moet worden.

Het opslaan van de telling heeft tot gevolg dat er een mutatie wordt aangemaakt. Deze waarde zal vanaf het telmoment als de nieuwe voorraad worden beschouwd. Dit wordt bereikt door alle mutaties die bekend zijn van het ingrediënt op inactief gezet. Daarna wordt er een mutatie doorgevoerd waarin het telverschil wordt opgeslagen. Als laatst wordt er een mutatie aangemaakt waarbij de getelde waarde wordt opgeslagen. Deze mutatie is wel actief. Vanaf dit moment worden de mutaties op de telling berekend.

4.5.5 Afboeken verkoopartikel

Dit is een nieuw idee wat ontstaan is bij het inplannen van de derde sprint. Er moet een mogelijkheid zijn tot het afboeken van een artikel. Hierdoor zal er een mutatie doorgevoerd moeten worden op alle ingrediënten die in het artikel verwerkt zijn. De gebruiker moet de mogelijkheid hebben tot het afboeken van meerdere ingrediënten in één handeling. Daarom moet het aantal af te boeken ingrediënten in te vullen zijn. Als de actie wordt doorgezet moet het systeem zelf de juiste aantallen afboeken van elk ingrediënt.

De User Control bestaat uit een overzicht van de verschillende artikelen. Een gedeelte van het scherm is gereserveerd om de ingrediënten te tonen die verwerkt zitten in de artikelen. Deze gegevens zullen aan de hand van het geselecteerde artikel veranderen. Daarnaast is er een formulier wat ingevuld kan worden om een artikel af te boeken. Als laatst kan er vanuit dit scherm een overzicht worden opgevraagd van de afboekingen van een ingrediënt.

De ViewModel achter het scherm zorgt ervoor dat het overzicht van de verschillende artikelen wordt opgehaald bij het aanmaken van het object. De ingrediënten worden pas opgehaald op het moment dat er een artikel geselecteerd wordt. Doordat de collectie van ingrediënten veranderd bij het selecteren van een artikel wordt de User Control daarop aangesproken waardoor de juiste ingrediënten worden getoond.

Als de gebruiker een aantal heeft ingevoerd wat afgeboekt moet worden van het geselecteerde artikel en vervolgens de actie doorvoert door op te slaan zal het systeem een mutatie wegschrijven voor elke ingrediënt. De methode berekent het aantal wat afgeboekt moet worden aan de hand van het ingevoerde aantal van de gebruiker en het aantal wat verwerkt zit in het artikel.

4.5.6 Oplevering prototype

Bij het evalueren van het prototype zijn de volgende nieuwe inzichten naar voren gekomen:

- Order annuleren
Verkooporders moeten door een gebruiker geannuleerd kunnen worden. Wanneer dit gebeurd moet er een keuzescherm getoond worden waarin de gebruiker kan kiezen of het nodig is om de artikelen af te boeken of dat het terug de voorraad in kan omdat het gerecht nog niet is klaargemaakt. Omdat er standaard artikelen zijn die terug de voorraad in kunnen, moet er bij de stamgegevens van het verkoopartikel een gekozen kunnen worden wat de actie is bij annulering.

- Mutaties op ingrediënten link naar aard mutatie
Bij de mutatiedetails van een ingrediënt was het nuttig een link te leggen naar de aard van de mutatie. Zo moet het mogelijk zijn om als de mutatie een gevolg is van het verkopen van een artikel, het orderscherm kan worden opgeroepen. Wanneer het zou gaan om een retour zou het retourscherm opgeroepen moeten worden. Wanneer het gaat om een inslag van goederen moet het detailscherm van de partij getoond worden.

- Voorraadoverzicht kortst houdbare THT datum tonen
In overzicht bij elk ingrediënt tonen wat de korst houdbare THT datum is van aanwezige partij.

- Informatie van inkooporder bij inslag
Bij het inslaan van goederen moet er meer informatie getoond worden over de inkooporder. Daarnaast moet een THT datum optioneel zijn. Dit voor als de gebruiker van het systeem niet volgens de FIFO wijze wil gaan werken.

- **Mutatiedetails partijen**
Van alle partijen moet het mogelijk zijn de mutaties te bekijken. Dit is vergelijkbaar met de mutatiedetails van ingrediënten.

- **Voorraad muteren bij verkooporder**
Bij het verkopen van verkoopartikelen moeten de ingrediënten worden gemuteerd. Dit dient ook op partijniveau gedaan te worden.

4.4.7 Plannen volgende sprint

De volgende sprint zou in het teken staan van het implementeren van de FIFO werkwijze. Een compleet overzicht van de ingeplande onderdelen in de sprint is te vinden als bijlage in deel 2 van het afstudeerdossier (bijlage 9, sprintbacklog sprint 4).

4.4.8 Aanpassingen database

De nieuwe inzichten hebben veranderingen vereist aan de database. Deze zijn doorgevoerd in het klassendiagram. De volgende wijzigingen zijn gedaan (zie bijlage 6, nieuwe situatie versie 2):

- Tabel reden is toegevoegd
- Ingrediënt heeft een koppeling met bestaande tabel btw
- Koppeling gemaakt tussen voorraad mutaties en bestaande tabel verkooporderregel
- Bestaande tabel verkooporder is toegevoegd

4.6 De vierde sprint: Uitbouwen voorraad met FIFO werkwijze

Deze sprint stond in het de FIFO (First In – First Out) werkwijze. Dit hield in dat de bestaande functionaliteiten uitgebreid zijn zodat deze te gebruiken zijn om te werken volgens de FIFO werkwijze indien dat gewenst is. Ook zijn in deze sprint de nieuwe ideeën meegenomen die tot stand zijn gekomen bij het bespreken van het vorige prototype.

4.6.1 Voorraad tonen partijen

In het voorraadscherm moest het mogelijk zijn van de ingrediënten de verschillende partijen te bekijken. Hierdoor is er een goed overzicht voor de gebruiker die op de FIFO manier werkt hoeveel er nog aanwezig is van elke partij en hoeveel er ingeslagen is bij het aanmaken van de partij. Vanuit het overzicht van partijen moet het ook mogelijk zijn om het mutatieoverzicht te tonen van een partij.

De bestaande User Control voor het overzicht is als eerst aangepast. Aan de datagrid die de voorraad toont per ingrediënt is een RowDetailTemplate toegevoegd. Deze template wordt alleen getoond als de regel in de bovenliggende datagrid geselecteerd is. In de template is een datagrid toegevoegd waarin de detailinformatie getoond kan worden van partijen. Per partij is er een verwijzing toegevoegd naar een nieuw scherm om de mutaties te tonen van de partij.

Om de detailinformatie op te halen is er een functie toegevoegd aan de ViewModel. Een methode zorgt voor het ophalen van de partijen die horen bij het geselecteerde ingrediënt. Deze methode wordt uitgevoerd wanneer er een ingrediënt geselecteerd wordt uit de voorraad. Voor de verwijzing naar de mutaties van de partijen moest een nieuwe User Control en ViewModel worden toegevoegd.

In de User Control is algemene informatie te vinden van de inkooppartij. Daarnaast wordt er een overzicht getoond van alle mutaties van de partij. Het scherm is vergelijkbaar met het mutatiescherm van een ingrediënt, echter worden hier de mutaties op partijniveau getoond.

De ViewModel zorgt ervoor dat dat bij een partij alle mutaties kunnen worden opgehaald en dat alle algemene partijinformatie getoond kan worden in de User Control. In figuur 4.6.1.1 is een voorbeeld van een ingrediënt met daaronder de partijen te zien.

Figuur 4.6.1.1: Partijen van ingrediënt ketchup

4.6.2 Mutatie op partij bij afboeken van een ingrediënt

Om de FIFO werkwijze te implementeren en het voorraadoverzicht te kunnen gebruiken op partijniveau was het van belang dat alle mutaties die gedaan worden door de gebruiker gedaan worden op een bepaalde partij. In het geval van het afboeken van een ingrediënt zou de gebruiker de keuze moeten krijgen om van een partij af te boeken waaruit het ingrediënt afkomstig is.

Aan de User Control hoefde bijna geen aanpassingen gedaan te worden. Omdat er al ingrediënten afboekt konden worden hoefde hier alleen nog maar een toevoeging gedaan te worden zodat er partijen geselecteerd kunnen worden om op af te boeken. Hiervoor is een selectielijst toegevoegd waarin de verschillende partijen getoond worden.

Aan de ViewModel moest er wel meer worden toegevoegd en aangepast. Zo is er een functie toegevoegd die er voor zorgt dat de partijen worden opgehaald bij een geselecteerd ingrediënt. Deze partijen worden in een lijst getoond in de User Control. Daarnaast moest er een functie toegevoegd worden die controleert of het af te boeken aantal wel aanwezig is in de geselecteerde partij. Het ingevulde aantal wordt vergeleken met het aantal dat er nog aanwezig is bij de partij. Wanneer het aantal niet afgeboekt kan worden wordt er een melding getoond en kan er niet afgeboekt worden (figuur 4.6.2.1).

Figuur 4.6.2.1: Ingevoerd aantal is niet af te boeken.

Deze controle gaat niet op wanneer de gebruiker geen partij selecteert. Om de gebruiker niet te verplichten een partij te kiezen en dus te werken volgens de FIFO werkwijze, is er ook de mogelijkheid de optie leeg te laten. In het geval van het niet aangeven van een partij, voert het systeem zelf de mutaties door op de partijen. Dit zal gebeuren door de partij te muteren met de kortst houdbare THT datum. Wanneer er meer afgeboekt wordt dan dat er in die partij aanwezig is, moet de daaropvolgende partij met de kortste THT datum gemuteerd worden. Deze stap zal zich herhalen tot het aantal wat afgeboekt moet worden is weggewerkt.

4.6.3 Mutaties bij retourneren

Als er een retourorder wordt gedaan moet er een mogelijkheid komen voor de gebruiker om een partij te kiezen waarvan er geretourneerd moet worden. Ook hier kan de gebruiker er voor kiezen geen partij te selecteren en dan zal het systeem het zelf afhandelen. Voor retouren is er ook een controle ingebouwd. Per te retourneren ingrediënt zal bekeken worden of het aantal wat geretourneerd dient te worden niet groter is als het huidig aanwezige aantal. Wanneer dit wel zo is bij één van de ingrediënten kan heel de retour niet doorgevoerd worden en wordt er ook een melding getoond bij de ingrediënten waardoor deze uitzondering wordt veroorzaakt.

Voor deze toevoeging moest ook de User Control worden aangepast. Per regel in de lijst met te muteren ingrediënten is een selectielijst toegevoegd waar de gebruiker kan kiezen voor een partij om de retour op te muteren indien gewenst.

De ViewModel controleert of het af te boeken aantal aanwezig is in de gekozen partij. Dit wordt gedaan voor elk te retourneren ingrediënt. Wanneer bij één van deze ingrediënten de retouractie niet uit te voeren is, kan de gehele retouractie niet doorgevoerd worden. Wanneer er bij een ingrediënt geen partij is geselecteerd zorgt het systeem ervoor dat er weer op de juiste partijen wordt gemuteerd.

4.6.4 Tellingen op partijen

Het belangrijkste en ook één van de ingewikkeldste functionaliteiten aan het systeem is het tellen op partijniveau. Als er volgens de FIFO werkwijze wordt gewerkt is het noodzakelijk dat er geteld kan worden op partijniveau. Om dit te bereiken moest er een wijziging worden gedaan aan de User Control en de ViewModel.

In de User Control voor dit scherm is ook een RowDetailTemplate toegevoegd om de partijen van de ingrediënten in te tonen. Hier is echter de functionaliteit van de template uitgebouwd zodat er meerdere RowDetailTemplates tegelijkertijd kunnen worden getoond. Standaard is het niet mogelijk dit te bereiken omdat alleen bij de geselecteerde regel de details worden getoond. Bij elke regel is er een knop toegevoegd die ervoor zorgt dat de details getoond worden. Door een functie toe te voegen die het openen en sluiten van de details afhandelt kunnen er meerdere details tegelijkertijd getoond worden. Dit zorgt ervoor dat er meer overzicht is bij het invoeren van de tellingen.

In de ViewModel zijn de bestaande functies uitgebreid en extra functies toegevoegd die het tellen op partijniveau of op ingrediëntniveau kunnen afhandelen. Als eerste wordt er gecontroleerd of er op partij geteld is of op ingrediënt. Als er op ingrediëntniveau geteld is moet het systeem zelf de telling verwerken en de aantallen per partij wijzigen. Er zijn drie verschillende scenario's bij het tellen op ingrediëntniveau.

- Telling waaruit geen telverschil optreedt, dit heeft weinig gevolgen.
- Telling gedaan waaruit blijkt dat er meer op voorraad is dan er volgens het systeem aanwezig is, dit is een positieve telling.
- Telling gedaan waaruit blijkt dat er minder op voorraad is dan er volgens het systeem aanwezig moet zijn, dan gaat het om een negatieve telling.

Het systeem zal zelf de telverschillen moeten verwerken. Dit gebeurd op de volgende manier:

Positieve telling

Bij een positieve telling (figuur 4.6.4.1) is er meer geteld dan dat er aanwezig is volgens het systeem. Er moet daarom een nieuwe partij worden aangemaakt. Er wordt voor alle partijen een telverschil opgeslagen om vervolgens de telling door te voeren. Hierbij wordt er een mutatie weggeschreven voor de nieuwe partij.

Positieve telling

Ingrediënt: Friet	Eenheid: Kilogram
Op voorraad volgens systeem	11
Geteld	12
Telverschil	1

Partijen	Aantal	Nieuw
Partij 1	2	2
Partij 2	5	5
Partij 3	2	2
Partij 4	2	2
Partij 5		1

Partij	Aantal	
1	0	telverschil
1	2	telling
2	0	telverschil
2	5	telling
3	0	telverschil
3	2	telling
4	0	telverschil
4	2	telling
5	1	telling

Figuur 4.6.4.1: Voorbeeld positieve telling

Negatieve tellen

Bij een negatieve telling (figuur 4.6.4.2) moeten er partijen worden gecorrigeerd. De partijen met de kortst houdbare THT datum worden aangepast. Dit wordt ook weggeschreven in de mutatietabel.

Negatieve telling

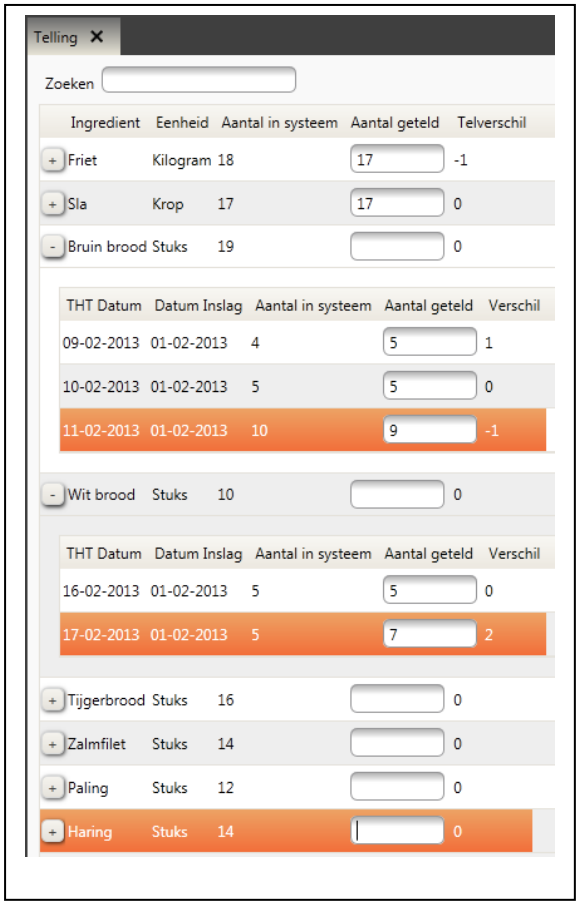
Ingrediënt: Friet	Eenheid: Kilogram
Op voorraad volgens systeem	11
Geteld	6
Telverschil	-5

Partijen	Aantal	Nieuw
Partij 1	2	2
Partij 2	5	4
Partij 3	2	0
Partij 4	2	0

Partij	Aantal	
1	0	telverschil
1	2	telling
2	-1	telverschil
2	4	telling
3	-2	telverschil
3	0	telling
4	-2	telverschil
4	0	telling

Figuur 4.6.4.2: Voorbeeld negatieve telling

Als een gebruiker volgens de FIFO methode werkt, zal deze waarschijnlijk op partijniveau willen tellen. Wanneer de gebruiker handmatig per partij de waardes invoert, worden de mutaties van de gebruiker doorgevoerd. Hiervoor hoeft het systeem zelf geen berekening te doen. De ingevoerde tellingen worden doorgevoerd op de partijen en er worden ook mutaties toegevoegd om de huidige voorraad gelijk te laten zijn aan de telling. Onderstaand figuur toont het eindresultaat van het scherm waarin de tellingen kunnen worden doorgevoerd.



Figuur 4.5.5: Tellingen in Delivery Manager

4.6.5 Mutaties partijen bij afboeken van verkoopartikel

Het doorvoeren van deze mutaties is vergelijkbaar met het afboeken van ingrediënten, met dit verschil dat er hier door de gebruiker gekozen wordt voor een verkoopartikel wat in zijn geheel afgeboekt moet worden. Bij het afboeken van een verkoopartikel moet er voor elk ingrediënt op partij worden gemuteerd.

Aan de User Control is geen aanpassing gedaan. In de ViewModel is de functie die het afboeken afhandelt wel aangepast. Voor elk ingrediënt wat verwerkt is in het artikel wordt er van de kortst houdbare partij het aantal afgeboekt. Dit is het gevolg van een aanname die gedaan is waarbij er vanuit wordt gegaan dat er volgens FIFO wordt gewerkt en er dus van de kortst houdbare partij moet worden afgeboekt. Wanneer er niet volgens FIFO wordt gewerkt maakt het niet veel uit van welke partij er wordt afgeboekt.

4.6.6 Oorsprong mutatie

Mutaties hebben een oorsprong, zoals verkoop van artikel of afboeken van ingrediënt. Om meer inzicht te krijgen in de oorsprong van de mutatie moest er een mogelijkheid zijn om deze te kunnen bekijken vanuit het mutatieoverzicht. De volgende (kleine) functionaliteiten zijn toegevoegd.

Inslag: Bij alle types inslag kan het scherm van de partijdetails worden opgevraagd. Vanuit dit scherm kan de gebruiker de inkooporder openen.

Retour: Bij alle types retour kan het retourneerscherm worden geopend. De gehele order wordt opgehaald met alle geretourneerde ingrediënten.

Verkoop: Bij alle types verkoop wordt er verwezen naar het order informatiescherm. Hierin is de gehele verkooporder te zien.

4.6.7 Mutaties op partijen bij verkoop van verkoopartikelen

Bij het verkopen van artikelen moeten de ingrediënten die er in verwerkt zijn ook worden gemuteerd op partijniveau. Ook hier wordt net als bij het afboeken van een artikel gebruik gemaakt van de aanname dat de korst houdbare partij moet worden gemuteerd.

De User Control hoefde niet aangepast te worden, de ViewModel wel. Op het moment van het muteren van de voorraad (afronding van de order) wordt er gekeken wat de oudste partij is om daar vervolgens de mutatie op door te voeren. Wanneer er nog een aantal open staat wordt de daaropvolgende partij gemuteerd, dit tot er niks meer te muteren is.

Bij het aanpassen van een bestaande order moet het systeem een aantal extra acties uitvoeren. Als eerst wordt er een tegenmutatie gedaan van alle aangepaste artikelen. Dit gebeurd op de oorspronkelijke partijen waar op de mutatie in eerste instantie was doorgevoerd. Hiermee wordt bereikt dat alle eerdere aanpassingen ongedaan worden gemaakt maar deze nog wel inzichtelijk zijn. De nieuwe mutatie wordt weggeschreven op mogelijk een andere partij.

Bij het verwijderen van een artikel uit de order wordt er alleen een tegenmutatie doorgevoerd op de partij waar op de order van toepassing was.

4.6.8 Signalering bij bederf van ingrediënten

In het voorraad overzicht moest een melding getoond worden wanneer een partij van een ingrediënt dreigde te bederven. Om dit aan te geven kan de THT datum van een partij getoond worden in drie verschillende kleuren:

- Groen voor als de partij een THT datum heeft van 5 of meer dagen vanaf de huidige dag.
- Oranje bij een THT datum die van een verschil heeft van 0 tot en met 5 dagen vanaf de huidige datum.
- Rood als de THT datum is verstreken.

Op dit moment is er voor gekozen om deze waardes vast te leggen in de code. Eventueel kan er in een later stadium gekozen worden om hiervoor ook een stamgegeven te maken. Dan zouden gebruikers zelf de status kunnen beheren. Omdat er per ingrediënt ook de oudste THT datum wordt getoond, kan deze ook in de bijbehorende kleur worden getoond.

In de ViewModel is er een functie toegevoegd die de kleur van de tekst kan aanpassen aan de hand van de THT datum. Er wordt gekeken wat het verschil is van tussen de THT datum van de partij met de huidige datum. Afhankelijk van het verschil wordt de kleur bepaald. De kleur kan gebind worden in de User Control aan de kleureigenschap van het stukje tekst wat de THT datum weergeeft. Het resultaat is te zien in figuur 4.6.8.1.

Datum Inkoop	Datum Inslag	Aantal	Huidige aantal	THT Datum	Details
28-01-2013	28-01-2013	6	4,8	5-2-2013	
28-01-2013	28-01-2013	6	4,9	6-2-2013	

Figuur 4.6.8.1: Voorbeeld van de werking

4.6.9 Signaleren bij te lage voorraad

Evenals bij de signalering op dreigend bederf moest deze signalering getoond worden in het voorraadscherm. De gebruiker heeft hier inspraak in het moment dat de melding getoond moet worden. Een toevoeging bij de stamgegevens van ingrediënten zorgt ervoor dat de gebruiker zelf een ‘ijzeren voorraad’ in kan vullen. Wanneer de huidige voorraad onder deze voorraad gaat zal er een melding getoond worden (figuur 4.6.9.1).

Paling	4	Stuks	30-1-2013	30-1-2013	
Haring	3	Stuks	24-1-2013	26-1-2013	

Figuur 4.6.9.1: Signalering bij te lage voorraad

4.6.10 Annuleren verkooporder

Als een verkooporder wordt geannuleerd kan dit op verschillende momenten gebeuren. Zo kan de gehele order al zijn bereid, deels zijn bereid of helemaal nog niet bereid. Deze verschillende mogelijkheden kunnen invloed hebben op de voorraad. Het kan namelijk zo zijn dat er een order wordt geannuleerd, die al is gemuteerd op de voorraad, maar nog wel terug genomen kan worden in de voorraad. Om dit op te vangen zijn er een aantal toevoegingen gedaan.

Allereerst kan de gebruiker aangeven in de stamgegevens van een artikel wat de actie moet zijn bij het annuleren van een order waar het artikel in zit. Dit vereiste kleine aanpassingen aan de User Control en de ViewModel van het stamgegeven.

Er is vervolgens een nieuwe User Control opgezet. Deze wordt aangeroepen op het moment dat er een order geannuleerd wordt. De artikelen die niet standaard terug kunnen worden genomen bij de voorraad worden hier in een lijst getoond. Hier kan voor deze artikelen alsnog gekozen worden of deze terug genomen kunnen worden in de voorraad.

De ViewModel verzamelt alle artikelen in een lijst. Als een artikel terug genomen kan worden in de voorraad wordt er een tegenmutatie doorgevoerd op de partij. Wanneer een artikel niet kan worden teruggenomen wordt er geen actie ondernomen.

4.6.11 Afsluiting sprint

Het ontwikkelen van de modules was bij het opleveren van dit prototype voltooid. Dit laatste prototype is doorgenomen en hieruit zijn geen nieuwe ideeën ontstaan of fouten naar voren gekomen. Het geheel was nu klaar om getest te worden.

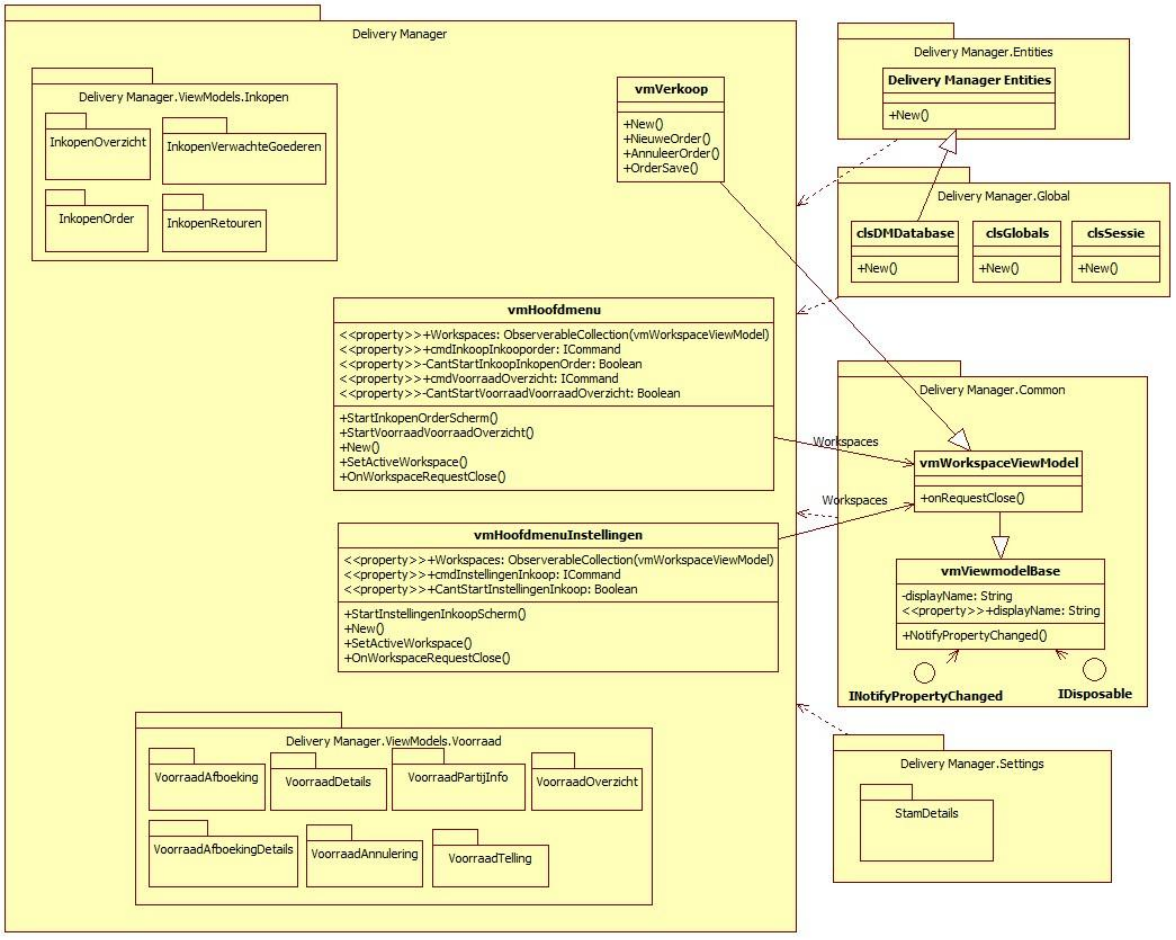
4.7 Klassendiagram applicatie

Na het ontwikkelen leek het mij, vanwege onder ander de overdraagbaarheid, handig om de systeemdocumentatie vollediger te maken met een klassendiagram. Het doel van het klassendiagram is een beeld te creëren van de toevoegingen en veranderingen die het gevolg zijn van ontwikkelen van de nieuwe modules.

4.7.1 Namespaces Delivery Manager

Om het klassendiagram overzichtelijk te houden is het in verschillende delen opgedeeld. Het klassendiagram kan het beste gelezen worden door eerst het Delivery Manager overzicht te bekijken (figuur 4.7.1.1). Hierin zijn meerdere packages te zien die de verschillende namespaces (verzameling van classes) in het project voorstellen. Vervolgens zijn er een aantal van die packages uitgewerkt. Dit zijn de namespaces waarin ik wijzigingen heb aangebracht of betrekking hebben op die wijzigingen.

Het overzicht van verschillende namespaces binnen Delivery Manager ziet er als volgt uit:



Figuur 4.7.1.1: Overzicht van namespaces Delivery Manager

De namespace Delivery Manager is afhankelijk alle andere namespaces. De ontwikkelde stukken voor de inkoop-en voorraadmodule bevinden zich in de namespaces Delivery Manager en Settings.

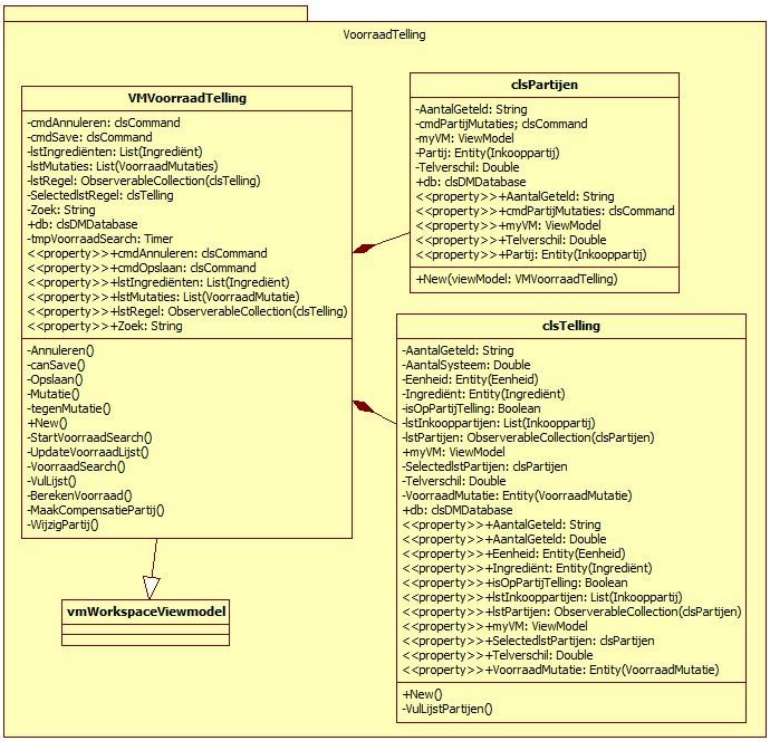
In de namespace Delivery Manager zijn de classes te vinden die de inkoop- en voorraadmodule vormen. In Settings zijn de stamgegevens toegevoegd die deze modules ondersteunden.

De namespace Delivery Manager bevat een class vmHoofdemenu. Vanuit deze class worden Workspaces aangemaakt, dit zijn de schermen die getoond worden. De classes die zijn toegevoegd om de zijn Workspaces. Al deze Workspaces zijn van het type vmWorkspaceViewmodel, wat in de namespace Common te vinden is. De ViewModel vmWorkspaceViewModel is van het type vmViewmodelBase. Deze class zorgt er voor dat er eigenschappen van het scherm aangepast kunnen worden, bijvoorbeeld de naam van het scherm. Verder worden door deze class de interfaces INotifyPropertyChanged en IDisposable gebruikt. Hierdoor kunnen deze services in alle schermen (Workspaces) aangeroepen kunnen worden.

De namespace Globals is ook van belang voor de modules. In deze namespace bevind zich de class clsGlobals. Deze class zorgt ervoor dat de stamgegevens worden ingeladen bij het opstarten van de applicatie. Daarnaast is de class clsSessie verantwoordelijk voor het bijhouden van sessies. Sessies worden gestart wanneer de applicatie wordt opgestart door een gebruiker. Deze class zorgt ervoor dat er gegevens kunnen worden bijgehouden van de gebruiker en van acties die uitgevoerd worden. Als laatst is er de class clsDMDatabase. Deze class zorgt voor een verbinding met de database. Hiervoor is het afhankelijk van de Enitities die te vinden zijn in de namespace Entities.

4.7.2 Package op detailniveau

Alle packages uit het overzicht zijn verder uitgewerkt. Figuur 4.7.2.1 laat hier een voorbeeld van zien.



Figuur 4.7.2.1: Uitwerking van de Voorraad Telling

Wat nog een keer verduidelijkt wordt is dat de class VMVoorraadTelling(workspace) een instantie is van de class vmWorkspaceViewmodel. Verder heeft de class een compositierelatie met een aantal andere classes. De compositierelatie houdt in dat de classes alleen kunnen bestaan wanneer de class daarboven bestaat en dat het maar tot één geheel behoord. Dit is precies wat er ook van deze classes wordt gevraagd. Ze worden alleen maar in de ViewModel gebruikt. De classes zorgen ervoor dat het mogelijk is om data van verschillende objecten te kunnen tonen. Als voorbeeld hiervoor kan een ObserverableCollection genoemd worden. Deze is vaak gebruikt om data weer te geven in een datagrid waarbij er van verschillende objecten data getoond moet worden.

Het volledige klassendiagram is te vinden als bijlage (7) in deel 2 van het afstudeerdossier.

4.8 Testen

Allereest zal er ingegaan worden op het opzetten van de test, waarbij als eerst het testplan aan bod komt. Daarna wordt er wat verteld over de verschillende testontwerpen die zijn opgezet aan de hand van de bevindingen uit het testplan. De testontwerpen bevatten de logische testontwerpen, die vertellen *wat* er getest moet worden. Daarnaast zorgen de fysieke testgevallen ervoor dat het duidelijk wordt *hoe* er getest moet worden. Vervolgens is nog beschreven hoe het testen zelf is verlopen wat zal resulteren in een vrijgaveadvies voor de software.

4.8.1 Testplan

Het testplan (bijlage 10) kan gezien worden als een plan van aanpak voor het testtraject waarin aangegeven is op welke wijze de testprocedures en verantwoordelijkheden worden verdeeld en beheerst. Het testplan is bedoeld voor iedereen die betrokken is bij de uitvoering van het project en bij de beoordeling en goedkeuring van de ontwikkelde software. In dit geval zijn dat mijn bedrijfsmentor en ikzelf. Het plan is opgezet doormiddel van Testgoal te gebruiken als leidraad. Testgoal is bekend en daardoor kon er snel begonnen worden aan het testtraject.

Het plan begint met opdrachtformulering. Hier wordt beschreven wat er bereikt moet worden met het testtraject. Dit houdt in dat er beschreven staat wat er aan het einde van het testtraject verwacht wordt aan producten, wat de randvoorwaarden zijn, welke uitgangspunten er zijn en wie de belanghebbenden zijn van de test. Daarnaast is er vastgelegd dat de volgende documenten opgeleverd zullen worden.

- Testontwerp uit te voeren test
- Eindrapportage testuitvoering:
  - Wat is er getest?
  - Wat zijn de resultaten?
  - Wat zijn de risico's bij het in productie nemen?

Om tot een testtechniek te komen om het gewenste resultaat te bereiken is er een Testrisicoanalyse uitgevoerd en zijn de algemene kwaliteitsattributen geprioriteerd door de belanghebbende van de test.

Testrisicoanalyse

Er is gebruik gemaakt van een eendimensionale Testrisicoanalyse(TRA). De input voor deze TRA zijn de requirementsspecificaties (sprintbacklogs) van het systeem. Bij het tot stand komen van de eendimensionale TRA zijn er een aantal stappen doorlopen, deze worden hieronder beschreven.

1. Vaststellen van de functies en aandachtsgebieden

Allereest moet er bij een Testrisicoanalyse de risicogebieden in kaart gebracht te worden. Omdat dit alleen maar kan nadat er duidelijkheid is over de functies en aandachtspunten is een Testboom opgesteld. De Testboom geeft een overzicht van de verschillende functionaliteiten en aandachtsgebieden.

2. Bepalen van het relatief belang
In deze stap zijn de functionaliteiten van de Testboom voorzien van een prioritering, met als doel vast te leggen welke functies een belangrijker zijn voor het systeem dan andere. Uit deze sessie komt naar voren wat door de verschillende belanghebbende als belangrijkste wordt beschouwd. De prioriteit wordt bepaald door de volgende punten te verdelen over de Testboom. De punten zijn overgenomen van de prioritering die door Testgoal wordt gebruikt.

9 punten: Functie is van cruciaal belang
5 punten: Belangrijke functie
3 punten: Niet cruciaal aandachtsgebied
1 punt: Aandachtsgebied is niet noodzakelijk voor werking van het systeem.

3. Verwerken van gegevens
In de laatste stap worden de gegevens uit de laatste stap verwerkt. De gegevens van de verschillende belanghebbende zijn hier verzameld en gesorteerd naar relatief belang. Deze risicogebieden zijn samengevoegd in risicocategorieën. Deze risicocategorieën zijn het eindproduct van de TRA.

De volgende risicogebieden zijn te onderscheiden:
Kritisch: Belangrijkste 10% van de risicogebieden
Hoog: Daaropvolgende 20% van de risicogebieden
Midden: Daaropvolgende 30% van de risicogebieden
Laag: Daaropvolgende 40% van de risicogebieden

Kwaliteitsattributen

Naast de TRA is de prioritering van de kwaliteitsattributen ook belangrijk. De lijst met kwaliteitsattributen is opgesteld om te definiëren waar een kwalitatief goed systeem aan moet voldoen. Door deze lijst in te laten vullen door de belanghebbende kan opgemaakt worden welk kwaliteitsattribuut het belangrijkst is. Hier kan een test op afgestemd worden.

Uit de ingevulde lijst kwam naar voren dat *functionaliteit* en *gebruiksvriendelijkheid* de belangrijkste kwaliteitsattributen zijn. Dit betekent dat het doel van de test zich moet richten op het testen van de mate waarin het systeem correct gedrag vertoont. Om dit te testen moet er gebruik gemaakt worden van een blackboxtest. Bij een blackboxtest wordt de code niet ingezien en het systeem wordt getest vanaf de buitenkant. Het testen wordt voor een groot gedeelte aan de hand van al ontwikkelde systeeminterfaces gedaan. De volgens Testgoals geschikte techniek voor het testen van de procesflow en daardoor het testen van de functionaliteit is:

Procescyclustest (PCT)

Deze testmethode richt zich op het testen van de organisatorische processen. De procescyclustest is uitgevoerd als een *branch coverage* test. Dit houdt in dat er bij het opzetten van de procesdiagrammen rekening gehouden is met het feit dat alle statements minimaal één keer aan bod komen. Hierdoor moeten bepaalde beslismomenten meerder keren worden doorlopen waardoor er elke keer een ander pad bewandeld wordt van het procesdiagram. Alle functionaliteiten worden op deze manier onder verschillende omstandigheden en met verschillende doeleinden getest.

Alternatieve testtechnieken

Naast de PCT testtechniek zijn er nog meerdere testtechnieken die door Testgoal worden voorgeschreven.

Oorzaak en gevolg:
Cause Effect graphing (C/E): Bij deze techniek worden de oorzaak en het gevolg gemodelleerd, gebruikmakend van beslissingstabellen. Deze testtechniek is zeer uitgebreid en tijdrovend.

State Transition: Techniek waarbij testgevallen worden ontworpen om toestandsovergangen te testen. Is goed voor het testen van procestoestanden en testen van schermovergangen.

De C/E techniek bleek niet geschikt door de tijd die er nodig is voor het opzetten en uitvoeren van een test. Daarnaast gaat het testen met de State Transitioning niet diep genoeg in op het testen van functionaliteit. Hier worden alleen de overgangen in kaart gebracht en getest op de juistheid.

Testen op performance, stabiliteit en betrouwbaarheid:
Load- testen: Systeem wordt belast met een representatieve constante belasting. De verwerkingstijden van de tijd kritische transacties of bedrijfsprocessen worden gemeten om te zien of deze snel genoeg zijn om te voldoen aan de performance- eisen.

Stress- testen: Bij stress-testen wordt het systeem belast met een toenemende belasting. Hierdoor kan bepaald worden onder welke belasting het systeem faalt en op welke manier dit gebeurd. Ook hier wordt het aspect functionaliteit niet getest en is dus ook niet geschikt.

Reliability- testen: Bij reliability-testen wordt de betrouwbaarheid (of stabiliteit) van het systeem getest door het systeem gedurende een langere tijd te laten werken onder een stabiele, representatieve belasting.

Bovenstaande technieken testen het systeem op aspecten die niets te maken hebben met de juistheid en correctheid van de functionaliteit. Hierdoor zijn deze technieken onbruikbaar om het test doel te bereiken.

Testen op input van de gebruiker:
Syntax test: Wordt gebruikt om te controleren op beperkingen die gesteld worden aan de invoer goed zijn geïmplementeerd.

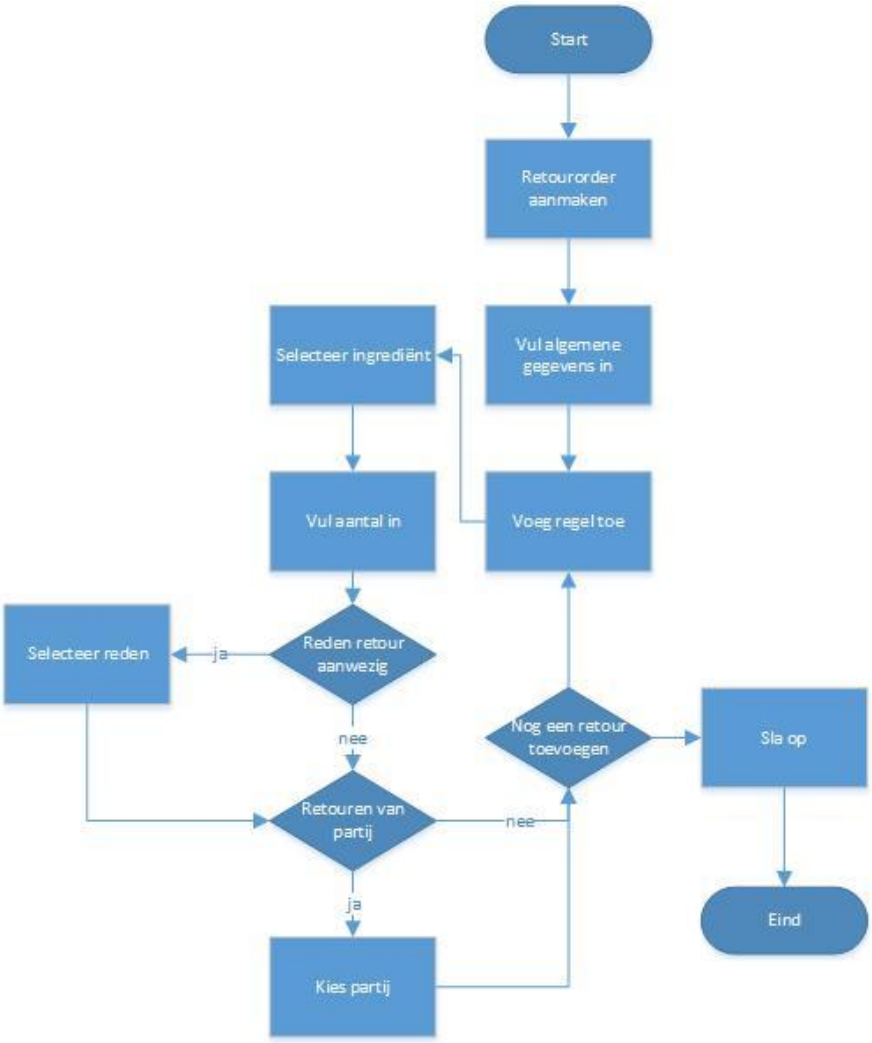
Equivalentieklassen: Een equivalentieklasse is een verzameling van mogelijke invoerwaarden die tot een zelfde soort verwerking leiden. Is goed om te controleren op invoervelden op de user interface en de functionele beslissingen die gemaakt worden aan de hand van de invoerdata.

Grenswaardeanalyse: Testgevallen worden verkregen aan de hand van grenswaarden. Deze grenswaarden bevatten de waarden die net onder, op of net boven de grens van de equivalentieklasse vallen. Er wordt onderscheid gemaakt tussen geldige en ongeldige grenswaarden.

De laatste drie technieken richten zich op de input van de gebruiker. Dit zou na het testen van functionaliteit het belangrijkste aspect zijn maar valt buiten de scope van het te bereiken doel (testen van de functionaliteit door het testen van de mate waarin er correct gedrag vertoond wordt).

4.8.2 Logisch testontwerp

Een logisch ontwerp laat zien wat er getest moet worden. Pas bij het opzetten van de fysieke ontwerpen wordt er meer op details ingegaan en wordt er uitgelegd hoe er getest moet worden. De logische testontwerpen van een PCT test bestaan uit procesdiagrammen (figuur 4.8.2.1) van het systeem. Waar op gelet moet worden is dat alle beslispunten meegenomen worden. Hierdoor ontstaat er een aaneenschakeling van acties die een gebruiker kan uitvoeren. Door de beslismomenten kunnen deze acties elke keer op een andere manier doorlopen worden. Een procesdiagram is niet meer als een aaneenrijging van de use cases. De procesdiagrammen zijn daarom ook aan de hand van de use cases gemaakt. Hier onder is een voorbeeld te zien van een logisch testontwerp. De logische testontwerpen zijn toegevoegd als bijlage(11) in deel 2 van het afstudeerdossier.



Figuur 4.8.2.1: Procesdiagram logisch testontwerp

4.8.3 Fysiek testontwerp

Het fysiek ontwerp laat zien hoe er getest moet worden. De fysieke ontwerpen zijn opgemaakt uit de logische testontwerpen. Doordat bekend is welke processen er zijn en welke beslismomenten invloed hebben op de uitkomst van een actie van de gebruiker, kunnen er fysieke tests opgezet worden. De fysieke testgevallen geven een precieze uitleg hoe er getest moet worden (zie figuur 4.8.3.1). Voor het opzetten van de fysieke testgevallen is er gebruik gemaakt van de standaard die wordt beschreven in Testgoal.

Testgeval	14

Doel van de test	Inkooporder aanmaken met meerdere orderregels en verschillende ingrediënten.
Pre conditie	Testdata ingrediënten en leveranciers ingevoerd Ingrediënt / leverancier testgevallen succesvol afgerond.
Post conditie	Inkooporder is toegevoegd.

nr	testactie	verwachte systeemreactie	opmerking	OK	NOK
1	In het hoofdmenu selecteer [Inkoop]	Submenu wordt aan de gebruiker getoond			
2	In het submenu selecteer [Inkooporder]				
3	Selecteer orderdatum (datum vandaag)	Orderdatum wordt ingevuld			
4	Selecteer leverdatum (datum vandaag + 1 dag)	Leverdatum wordt ingevuld			
5	Selecteer leverancier	Leverancier wordt geselecteerd. Lijsten selectie veranderd			
6	Vul omschrijving in				
7	Selecteer eerste ingrediënt (tijgerbrood)	Tijgerbrood geselecteerd - Eenheid wordt getoond			
8	Voer valide aantal in (14)				
9	Pas prijs per stuk aan (2)				
10	Voeg regel toe	Nieuwe regel wordt toegevoegd aan de order			
11	Selecteer tweede ingrediënt (gehakt)	Friet geselecteerd - Eenheid wordt getoond			
12	Voer valide aantal in (10)				
13	Voeg regel toe	Nieuwe regel wordt toegevoegd aan de order			
14	Selecteer derde ingrediënt (cola)	Haring geselecteerd - Eenheid wordt getoond			
15	Voer valide aantal in (8)				
16	Voeg regel toe	Nieuwe regel wordt toegevoegd aan de order			
17	Selecteer vierde ingrediënt (sla)	Sla geselecteerd - Eenheid wordt getoond			
18	Voer valide aantal in (9)				
19	Opslaan [Save]	Gegevens worden opgeslagen			

Figuur 4.8.3.1: Fysiek testgeval

Elk testgeval heeft een nummer. Er staat verder beschreven wat het doel is van de test, wat de pre conditie is als die van belang is en wat er verwacht wordt na het doorlopen van het testgeval. Tijdens het testen kunnen voor elke stap opmerkingen ingevuld worden en kunnen de stappen afgevinkt worden. Wanneer de stap zonder problemen genomen kon worden is er een vinkje geplaatst bij OK. Wanneer er problemen zijn geweest zal dit moeten gebeuren bij NOK. Alle testgevallen uit het fysieke testontwerp dekken de beslismomenten die er zijn beschreven in de procesdiagrammen van het logisch ontwerp. De fysieke ontwerpen doorlopen op deze manier alle verschillende paden. De fysieke testontwerpen zijn te vinden in bijlage 11.

4.8.4 Uitvoeren van de test

Voor er getest kon worden moest er eerst testdata in de database worden gezet. Welke testdata er toegevoegd moet worden staat beschreven in het testontwerp. Het gaat vooral om stamgegevens die tijdens het testen van processen gebruikt worden. Tijdens het testen zijn de bevindingen bijgehouden door het invullen van de bevindingenregistratietemplate uit het ontwerp. Door deze template in te vullen is het na de test nog duidelijk wat het probleem is en wanneer het optreedt. Elke bevinding heeft een uniek nummer en heeft ook een bepaald niveau wat bepaald hoe kritisch het testgeval is voor de test. De omschrijving bevat de bevinding zoals deze tijdens het testen naar voren is gekomen. Indien nodig kunnen in het vakje commentaar de vervolgstappen beschreven worden die het probleem zouden kunnen oplossen.

4.8.5 Testrapportage

Het doel van dit rapport is het geven van een vrijgaveadvies. Het testrapportage (zie bijlage 12) is een sober en duidelijk verslag waaruit gemakkelijk af te lezen is wat het resultaat is van de test. Grafieken zijn gebruikt om te laten zien hoeveel bevindingen er zijn gedaan en in welke module deze bevinding zijn gedaan. Aan de hand van de bevindingen die er gedaan zijn wordt het vrijgaveadvies opgesteld.

In het vrijgaveadvies wordt de status van het product vergeleken met het beoogde resultaat. Dit wordt weergegeven aan de hand van de openstaande risico's en de nog uit te voeren testen. Het verschil tussen de beoogde kwaliteit en de gerealiseerde kwaliteit levert een quality gap op. Onderstaand vrijgaveadvies is hierop gebaseerd.

Vrijgaveadvies

Door de bevindingen te analyseren is er tot een *conditioneel* vrijgaveadvies tot stand gekomen. Het advies is om de software pas vrij te geven als er aan de genoemde condities is voldaan. De beschreven condities betreffen het minimaal oplossen van drie bevindingen waarvan de werking vereist is om de processen allemaal te kunnen doorlopen met een goede uitkomst. Verdere toelichting is te vinden in het testrapport.

4.9 Rapportages

Na het ontwikkelen van de modules zijn er een aantal rapportages opgezet in Crystal Reports. Aan de databasekant zijn er stored procedures geschreven om de juiste data snel en efficiënt te kunnen tonen. Daarnaast is ervoor gezorgd dat de rapportages te benaderen en te genereren zijn vanuit Delivery Manager.

4.9.1 Stored procedures

In eerste instantie zijn er stored procedures opgezet. Het voordeel van een stored procedure is dat ze draait op de databaseserver. Daardoor heeft de procedure direct toegang tot de gegevens die deze moet manipuleren en hoeft alleen de resultaten terug te sturen. Dit vermijdt het over en weer sturen van gegevens.

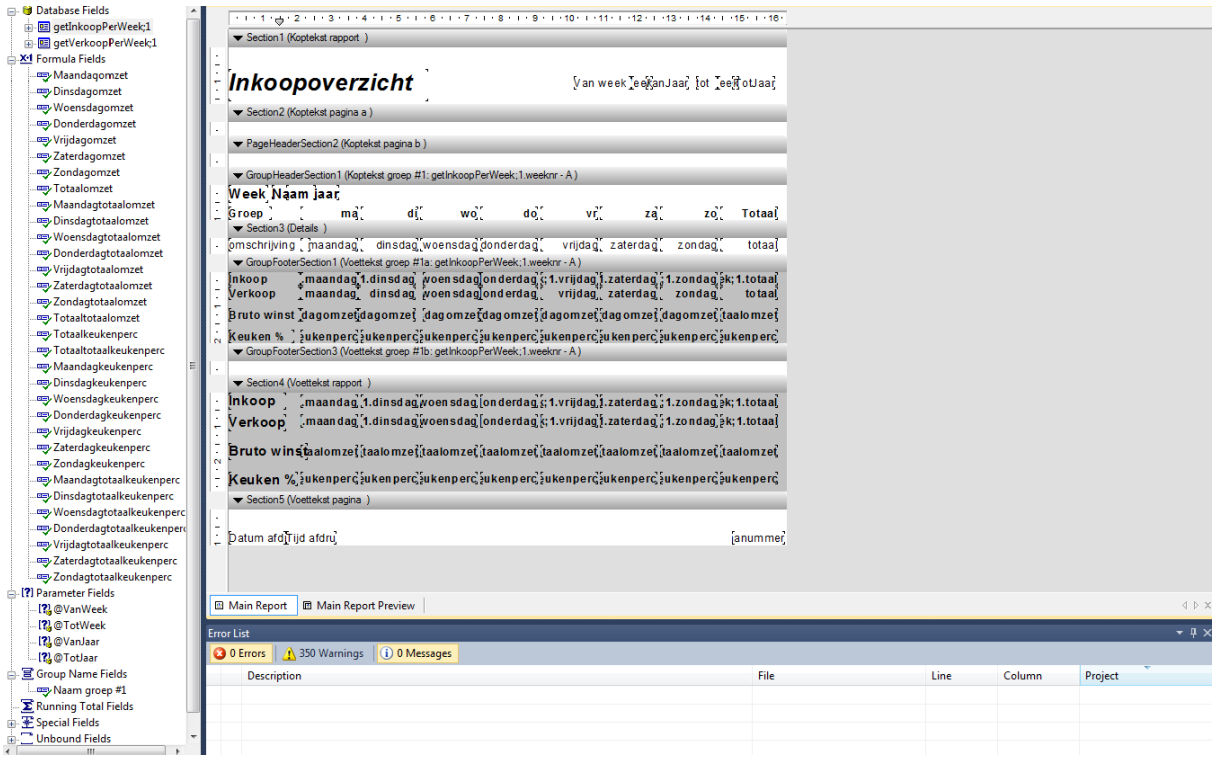
De stored procedure kunnen gebruik maken van een aantal parameters. Deze dienen meegegeven te worden door de gebruiker bij het genereren van een rapportage. Een voorbeeld van een stored procedure is te zien in figuur 4.9.1.1. In deze stored procedure wordt er een parameter IngrediëntGroepID gebruikt. De waarde van de parameter beïnvloedt de uitkomst van de query en dus de informatie die getoond wordt in het rapport.

```
ALTER PROCEDURE [dbo].[getHuidigeVoorraad] (@IngredientGroepID as integer)
AS
BEGIN
    SET NOCOUNT ON;
DECLARE @VanGroepID as integer;
DECLARE @TotGroepID as integer;

if @IngredientGroepID = 0
BEGIN
SET @VanGroepID = 0;
SET @TotGroepID = 2147483647;
END
Else
BEGIN
SET @VanGroepID = @IngredientGroepID;
SET @TotGroepID = @IngredientGroepID;
END

SELECT IngredientGroep.Omschrijving as IngredientGroep, IngredientGroep.ID as IngredientGroepID, Ingredient.ID as IngredientID,
Ingredient.Naam as Ingredient, Eenheid.Naam as Eenheid,
SUM(VoorraadMutatie.Aantal) as Aantal, Ingredient.inkoopprijs as Inkoopprijs,
SUM(VoorraadMutatie.Aantal) * Ingredient.inkoopprijs as WaardeVVP,
SUM(isnull(Inkooporderregel.prijs_per_stuk,0) * isnull(VoorraadMutatie.Aantal,0)) as WaardeInkoop
FROM IngredientGroep left outer join Ingredient on
IngredientGroep.ID = Ingredient.IngredientGroepID Left outer join VoorraadMutatie on
Ingredient.ID = VoorraadMutatie.IngredientID left outer join Eenheid on
Eenheid.ID = Ingredient.EenheidID left outer join Inkooppartij on
Inkooppartij.ID = VoorraadMutatie.InkooppartijID left outer join Inkooporderregel on
Inkooporderregel.ID = Inkooppartij.inkooporderregelID
WHERE VoorraadMutatie.isActief = 1 AND IngredientGroep.ID > = @VanGroepID and IngredientGroep.ID <= @TotGroepID
GROUP BY Ingredient.Naam, IngredientGroep.Omschrijving, Ingredient.inkoopprijs, Eenheid.Naam, IngredientGroep.ID, Ingredient.ID
ORDER BY IngredientGroep.Omschrijving asc, Ingredient.Naam asc
END
```

Figuur 4.9.1.1: Stored procedure – Rapportage Huidige Voorraad

4.9.2 Rapporten ontwerpen

De rapportages zijn opgezet met een business intelligence-applicatie (Crystal Reports) die direct vanuit Visual Studio te gebruiken is. Crystal Reports wordt gebruikt om rapporten vanuit diverse gegevensbronnen te ontwerpen en te produceren. In het geval van de rapportages worden de stored procedure(s) als gegevensbron gebruikt. Figuur 4.9.2.1 geeft een voorbeeld van een opzet van rapportage.



Figuur 4.9.2.1 Opzet van rapportage Inkoopoverzicht

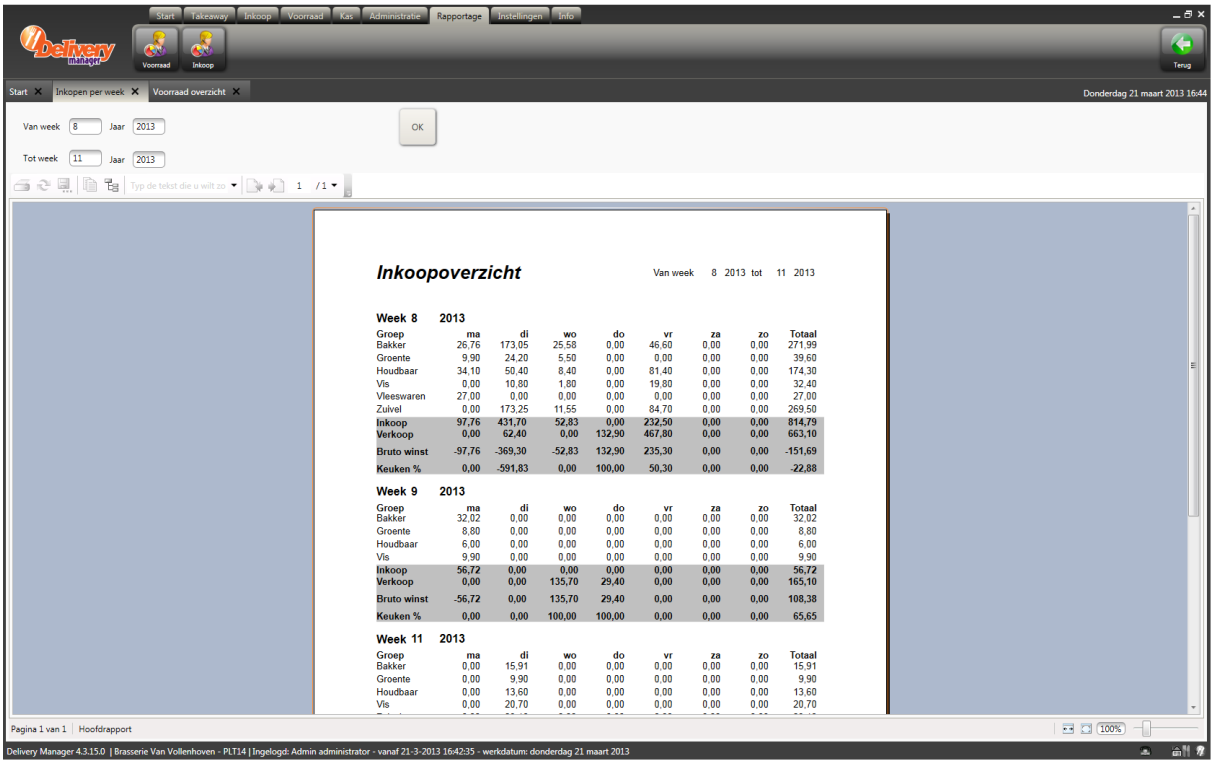
De velden die door de stored procedure opgevraagd worden (geselecteerde kolommen van tabellen in de query) kunnen in het rapport worden getoond. Daarnaast kunnen er berekeningen worden uitgevoerd met de databasevelden om tot een totaal, gemiddelde of percentage te komen. Een voorbeeld van een berekening is te vinden in figuur 4.9.2.2. Ook de parameters kunnen gebruikt worden in het rapport.

```
if ({@Maandagtotaalomzet}) <> 0 and (SUM ({getVerkoopPerWeek;1.maandag})) <> 0  then
({@Maandagtotaalomzet} / SUM ({getVerkoopPerWeek;1.maandag})) * 100
else  0
```

Figuur 4.9.2.2: Berekening voor totaal keukenpercentage (bruto winst gedeeld door omzet)

4.9.3 Rapportages in Delivery Manager

Alle rapportages zijn te benaderen en te generen vanuit Delivery Manager (voorbeeld in figuur 4.9.3.1). Voor alle rapportages is een User Control en een ViewModel aangemaakt. Indien de gebruiker input heeft op het te genereren rapportages door het kiezen van parameters zijn deze in te vullen in de User Control. De ViewModel is verantwoordelijk voor het ophalen en doorgeven van de parameters bij het genereren van een rapportage.

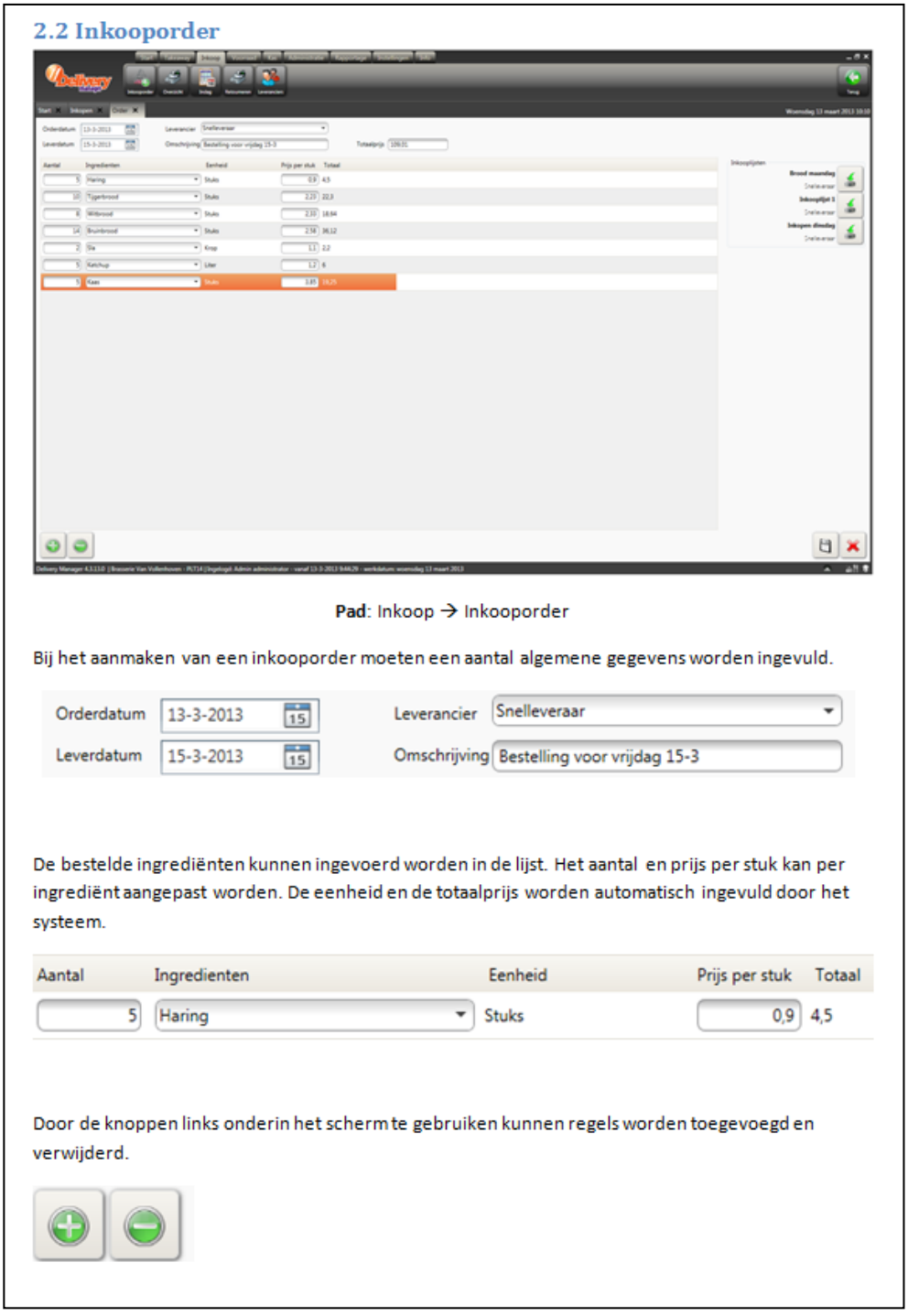


Figuur 4.9.3.1: Rapportage Inkoopoverzicht in Delivery Manager

In deel 2 van het afstudeerdossier zijn ter illustratie een aantal gegenereerde rapportages toegevoegd als bijlage(13). Ook de stored procedures die gebruikt worden bij het genereren van de rapportages zijn daar te vinden.

4.10 Handleiding

Ter ondersteuning van de modules is een handleiding opgeleverd. In de handleiding staan alle schermen weergegeven en de functionaliteit die het biedt beschreven. Een voorbeeld wordt getoond in figuur 4.10.1. De handleiding gaat niet in op de technische aspecten van de modules maar alleen op de functionaliteit die het biedt voor de gebruiker en hoe deze benut kunnen worden.



2.2 Inkooporder

Pad: Inkoop → Inkooporder

Bij het aanmaken van een inkooporder moeten een aantal algemene gegevens worden ingevuld.

De bestelde ingrediënten kunnen ingevoerd worden in de lijst. Het aantal en prijs per stuk kan per ingrediënt aangepast worden. De eenheid en de totaalprijs worden automatisch ingevuld door het systeem.

Door de knoppen links onderin het scherm te gebruiken kunnen regels worden toegevoegd en verwijderd.

Figuur 4.10.1 Handleiding - Inkopen van ingrediënten

5. Evaluatie

In dit hoofdstuk kijk ik eerst terug op de begeleiding die ik tijdens het afstuderen heb gehad. Verder worden het product en het proces geëvalueerd.

De organisatie waar ik afgestudeerd heb heeft tevens ook een evaluatieformulier ingevuld. Dit is te vinden als bijlage (15) in deel 2 van het afstudeerdossier.

5.1 Begeleiding tijdens het afstuderen

Vanuit de opleiding was er tijdens het afstuderen indien gewenst de mogelijkheid op begeleiding. Aangezien ik van deze mogelijkheid gebruik wilde maken, zijn er meerdere momenten geweest waarop ik met mijn begeleider heb afgesproken om de voortgang te bespreken. Aan het begin van de afstudeerperiode is er een ontmoeting geweest met mijn begeleider van school en de bedrijfsmentor van Pamicon. Bij deze ontmoetingen werd geëvalueerd hoever ik met mijn opdracht was, of er problemen waren en hoe Pamicon dacht over mijn werk en de positie binnen het bedrijf. Ook werd definitief de opdracht vastgelegd. Daarnaast heb ik regelmatig contact gezocht met de begeleider van school om mijn voortgang te bespreken.

De begeleiding van uit Pamicon was goed geregeld. Ik kon altijd ergens terecht met mijn vragen. In eerste instantie was dat bij mijn bedrijfsmentor. Door het werken met SCRUM en omdat Pamicon een platte organisatie is, had mijn begeleider ook inzicht in mijn werkzaamheden. Door elke ochtend en vaak ook aan het eind van de dag een kort overleg te houden kon ik vertellen waar ik mee bezig was, of alles volgens plan liep en of ik vast zat. Al met al was de begeleiding van school en vanuit de Pamicon goed geregeld. Ik ben zeer dankbaar voor deze begeleiding.

5.2 Productevaluatie

Over de toevoegingen aan het product Delivery Manager ben ik tevreden. De terugkoppeling naar de bedrijfsmentor en de medewerker van de marketing heeft er voor gezorgd dat er in het product rekening is gehouden met de wensen van de klanten en dat het product is ontwikkeld naar de wens van Pamicon.

Over de opgeleverde tussenproducten ben ik ook tevreden. In het begin van de afstudeerstage heb ik een plan van aanpak opgeleverd. De bedoeling van dit plan was om de opdracht in kaart te brengen, een planning op te zetten en het één en ander af te bakenen. Dit plan heeft bij mij en Pamicon een beeld gecreëerd van de invulling van de afstudeerperiode.

De documenten betreffende de technische en functionele ontwerpen die zijn meegenomen in het proces zijn naar mijn idee van goede kwaliteit. Dit omdat tijdens het ontwerpen regelmatig terugkoppeling en overleg is geweest. Hiermee is bereikt dat alle wensen op een valide manier meegenomen zijn in de ontwerpen.

Over de documenten die tijdens het testtraject tot stand zijn gekomen ben ik tevreden. Het werken met Testgoal heeft ervoor gezorgd dat er een duidelijke structuur is gekomen in het traject maar ook

in de documentatie. Hierdoor is het systeem ontwikkeld naar wens van de bedrijfsmentor en ook afdoende getest.

Als laatst zijn de toegevoegde rapportages aan de module een zeer nuttige toevoeging om zaken als voorraad en kosten inzichtelijk te krijgen. De handleiding die opgeleverd is aan het eind van het afstuderen geeft een goed beeld van de functionaliteiten die te gebruiken zijn in de modules.

5.3 Procesevaluatie

Over het algemeen ben ik tevreden over het gehele afstudeerproces. In het begin van het traject is er de tijd genomen om de requirements duidelijk te krijgen en te vertalen naar verschillende ontwerpen. Door het neerzetten van een goede basis aan het begin van het traject kon ik me tijdens het ontwikkelen en testen volledig richten op de activiteiten die daarbij komen kijken.

Het gebruik van SCRUM heeft als gevolg gehad dat er verschillende prototypes zijn opgeleverd en dat er regelmatig terugkoppeling is geweest. Hierdoor is er getoetst of ontwerpen die de requirements afdekten ook op de juiste manier ontwikkeld zijn. Ook zijn er hierdoor nieuwe inzichten ontstaan. Deze kunnen door de flexibele ontwikkelmethode gemakkelijk meegenomen worden voor ontwikkeling.

Over het ontwikkeltraject ben ik tevreden. Wel heb ik het idee dat er te weinig ruimte was om mijn ideeën naar voren te brengen.

Ondanks dat alles naar mijn idee goed is verlopen tijdens het testen en er een goed traject is doorlopen ben ik te weten gekomen dat ik het testen een minder leuke activiteit vind. Documenteren vind ik niet erg en het uitstippelen van een plan en ontwerp is vaak interessant. Dit heb ik niet zo gevoeld tijdens het testen. Ondanks dat heb ik het niet afgeraffeld en heb ik een goed gevoel overgehouden aan de uitvoering en het resultaat van de test.

5.4 Functioneren in het bedrijf

Ik heb ervaren dat men bij Pamicon vooral geïnteresseerd is in resultaten en minder in de manier waarop die resultaten tot stand komen. Om mijn competenties te vervullen en om niet zomaar ergens aan te beginnen had ik er goed aan gedaan meer een middenweg te vinden tussen wat ik wilde bereiken met mijn afstudeerstage en wat Pamicon wilde bereiken. Ik heb het idee dat ik deze middenweg niet helemaal gevonden heb en mij teveel heb laten leiden door het werkproces van Pamicon.

Omdat Pamicon een platte organisatie is, was het contact leggen met mensen niet moeilijk. Hierbij komt ook nog eens dat ik gedurende de afstudeerstage eigenlijk maar met 3 tot 4 mensen contact heb gehad. Dit is wel fijn omdat je vaak direct al terecht kan met problemen of vragen.

6. Competenties

In dit hoofdstuk staat beschreven of en hoe de vooraf aangegeven beroepscompetenties vervult zijn tijdens de afstudeerstage.

6.1 Analyseren en adviseren

6.1.1 Uitvoeren analyse door definitie van requirements

Deze competentie is eigenlijk als een rode draad door het hele project aanwezig geweest. Tijdens het ontwerpen, ontwikkelen en opleveren is er constant overleg geweest tussen mij en de bedrijfsmentor. Tijdens het ontwerpen zijn alle ontwerpen meerdere malen besproken en daardoor zo opgezet dat deze aan de wensen voldoen van Pamicon.

Voor het ontwikkelen geldt eigenlijk hetzelfde. Ook hier is elke twee weken bij het afsluiten van de sprints, maar soms ook tussendoor, overleg geweest over de manier waarop de modules in Delivery Manager zouden moeten gaan werken, welke functionaliteiten er wel of niet gewenst zijn en hoe de software zo opgezet kan worden dat het voor de klanten aantrekkelijk is. Bij het bespreken van nieuwe ideeën is er altijd nagegaan of het haalbaar was en of het een nuttige toevoeging was.

6.2 Database ontwerp en- ontwikkeling

6.2.1 Ontwerpen, bouwen en bevragen van een database

Naast de functionele ontwerpen heb ik ook een aantal technische ontwerpen opgesteld. De eerste opzet van het klassendiagram heeft er voor gezorgd dat er een goede basis is neergezet waarop gebouwd kon worden. De totstandkoming van deze basis is goed verlopen door de huidige situatie in kaart te brengen en de aanpassingen voor de nieuwe situatie hieraan toe te voegen.

Het bouwen van de database is gedaan in SQL Server. Het aanmaken en indien wijzigen van tabellen is allemaal gedaan door gebruik te maken van het DBMS. Doordat er niet gewerkt is met constraints aan de databasekant is het bouwen beperkt gebleven.

Door het gebruik van het Entity Framework wordt er met de database gecommuniceerd via een applicatie model. Door niet rechtstreeks te communiceren met de relationele opslag maar met het conceptueel model is er gebruik gemaakt van LINQ. Bij het opzetten van de rapportages is er gebruik gemaakt van een andere manier om gegevens te verzamelen. Om voor elk rapportage de juiste gegevens op te halen zijn een aantal stored procedures geschreven. Zo heb ik op verschillende manieren gecommuniceerd met de database en gegevens verzameld en aangepast.

6.3 Software ontwerp en- ontwikkeling

6.3.1 Ontwerpen systeemdeel

Mijn idee is dat het ontwerpproces bij Pamicon gezien wordt als overbodig of iets wat maar beter snel gedaan kan zijn zodat er daarna met het echte werk begonnen kan worden. Ik heb echter geleerd dat dit niet de beste manier is om te werken. Daarom wilde ik mijn kennis gebruiken om dit anders te doen.

Ik heb gekozen om gebruik te maken van UML omdat deze modeleertaal voor mij zeer vertrouwd is. Tijdens de studie heb ik hier meerder malen gebruik van gemaakt en dat is mij altijd erg goed bevallen. Om een beeld te creëren van de functionaliteiten heb ik gebruik gemaakt van Use Case diagrammen. Deze diagrammen heb ik zo gedetailleerd mogelijk omschreven in de begeleidende beschrijvingen. Daarnaast zijn er nog ontwerpen van schermen gerealiseerd. Dit was zeer nuttig om te gebruiken bij het opzetten van de schermen tijdens het ontwikkelen.

Het technisch ontwerp waarbij de functionele ontwerpen als uitgangspunt waren genomen zijn een goede leidraad geweest tijdens het ontwikkelen. Hoewel het ontwerp voor mij voldoende was uitgewerkt is de documentatie nog uitgebreid met een klassendiagram. Hiermee wilde ik de overdraagbaarheid van de ontwikkelde delen verbeteren. Ik heb getracht dit te bereiken door duidelijk te maken wat er is toegevoegd aan de bestaande applicatie en hoe de toegevoegde modules zijn opgezet.

6.3.2 Bouwen applicatie

Omdat ik geen kennis had van WPF en geringe kennis van.NET was het in het begin lastig om hiermee gewend te raken. Daarnaast was het ook vrij nieuw voor mij om aan een bestaand pakket te werken en niet van het begin de software te ontwikkelen. De programmeertaal Visual Basic had ik snel onder de knie omdat deze weinig verschilt van de voor mij al bekende programmeertaal C#. Al met al denk ik dat het gehele traject van ontwikkelen van de applicatie soepel is gegaan. De eerste weken had ik het soms lastig met het werken in aan Delivery Manager omdat het al een ver gevorderd product is. Af en toe raakte ik verdwaald in de applicatie die al vele functionaliteiten kent.

Tevreden kijk ik terug naar doorlooptijd van het ontwikkeltraject. Hierbij heb ik de planning weten aan te houden en is er geen vertraging opgelopen. Ik vind het speciaal dat ik binnen de geplande tijd de applicatie ook helemaal af heb gekregen.

Om er voor te zorgen dat de opgeleverde module later nog verder ontwikkeld kan gaan worden heb ik geprobeerd zoveel mogelijk de standaard aan te houden van Pamicon. Omdat er geen echte code conventie is zal ik nooit helemaal aan die standaard kunnen voldoen maar dit is ook geen vereiste van Pamicon.

6.3.3 Uitvoeren van en rapporteren over het testproces

Het gehele testtraject was betrekkelijke kort en heeft meer bestaan uit het opstellen van een testplan, het maken van ontwerpen en het rapporteren over de test dan het uitvoeren van de test zelf. Ook heb ik veel moeten investeren in het volgen van de theorie die beschreven staat in Testgoal. Daarvoor moest ik mij goed inlezen en tijdens het opstellen van de documenten ook goed blijven concentreren op de werkwijze die in Testgoal wordt beschreven.

Over de kwaliteit van de documenten die het testen heeft opgeleverd ben ik zeer tevreden. Het testplan was een goed begin voor het testtraject waarmee ik de bedrijfsmentor bij het testen heb kunnen betrekken. Omdat Pamicon zelf niet of nauwelijks doet aan testen van de ontwikkelde software vond ik het leuk om hem erbij te betrekken. Dit is naar mijn idee goed gelukt door de bedrijfsmentor te betrekken bij het invullen en bespreken van de Testrisicoanalyse en de kwaliteitsprioritering. Hierdoor is ook een test uitgevoerd die de kwaliteitseisen van Pamicon, namelijk functionaliteit en gebruiksvriendelijkheid, afdoende heeft getest.

Wat mij tegenviel aan het testtraject is dat het opzetten van een plan en het opzetten van de ontwerpen veel tijd in beslag neemt. Het testen zelf is echter heel snel gegaan. Toch begrijp ik dat het belangrijk is om een goed plan te hebben en gestructureerd te werk te gaan. Dit voorkomt dat er dingen vergeten worden of juist werk onnodig gedaan wordt.

7. Conclusie

Over de gehele afstudeerperiode van vier maanden heb ik het erg naar mijn zin gehad. Het was in het begin een beetje wennen om in een echte organisatie te werken die een eigen werkwijze en manier van denken heeft. Naar verloop van tijd voelde ik mij wel steeds meer op mijn gemak. De opdracht zoals ik hem van te voren had beschreven is volledig en correct afgehandeld. Het bedrijf is blij met mijn opgeleverde producten en zal de modules zeker gaan gebruiken.

Het feit dat je bij je afstudeerstage in het diepe wordt gegooid en jezelf moet bewijzen door de kennis te gebruiken die je de afgelopen jaren hebt opgedaan, was zeer spannend. In het begin was ik hier een beetje bang voor maar toen het eenmaal zover was ging het allemaal vanzelf. Door van te voren een goed duidelijk plan te maken en daarin al een aantal zaken vast te leggen was het makkelijker om een begin te vinden tijdens het afstuderen.

Door in iteraties te werken met de ontwikkelmethodiek SCRUM, heb ik een duidelijke manier van werken voor mezelf gecreëerd. Daarbij zorgde het ook voor inzicht in mijn werkzaamheden bij het bedrijf en zijn er regelmatig prototypes opgeleverd. Het evalueren van deze prototypes zorgde voor nieuwe inzichten. In deze evaluatie heb ik mijn ideeën ook naar voren kunnen brengen en heb ik mij kunnen mengen in de discussie.

Hoewel het testtraject mij minder goed bevallen is ben ik wel tevreden met het eindresultaat. Ook ben ik blij dat het systeem afdoende is getest en dat het doel van de test is bereikt.

De van te voren beschreven competenties die ik wilde gaan vervullen tijdens mijn stage zijn allemaal aan bod gekomen, de ene competentie wel wat meer dan de andere. Al met al ben ik er van overtuigd dat de competenties op een voldoende niveau aan bod zijn gekomen tijdens het afstuderen.

Ik vind dat ik een geslaagde afstudeerperiode echter de rug heb. Ik heb veel geleerd van het bedrijfsleven en heb verder ook een hoop kennis opgedaan op een heel ander gebied dan wat in mijn opleiding behandeld is.