Afstudeerverslag

Ontwikkeling van de correspondentie module

Gerlof van der Veen
4-6-2014

Voorwoord

Ik wil graag de mensen bedanken die het project mogelijk hebben gemaakt. Allereerst gaat mijn dank uit naar de mensen binnen CCGroup die de mogelijkheid hebben geschapen om deze opdracht uit te kunnen voeren, Bram Vos en Reza Jarehani.

Daarnaast wil ik mijn directe collega's bedanken en dan specifiek Ernie Molenaar, Nice Verseput en Paul Tinkler.

Eveneens wil ik mensen bedanken uit mijn persoonlijk omgeving, in het special Peter Tuik, Benjamin van Eindhoven voor het forceren dat ik wat ga doen. En daarnaast wil ik ook de mensen bedanken die het document voor me doorgelezen hebben.

Rijswijk 4-6-2014

Inhoudsopgave

Inleiding

Het afstudeerverslag wat voor U ligt en het proces en de keuzes van het afstudeerproject beschrijft is opgedeeld in de volgende secties: introductie, opzet, uitvoer van het proces, afronding en conclusie.

De eerste sectie bestaat uit het eerste en tweede hoofdstuk en dient als introductie voor het project. Tijdens de hoofdstukken die gaan over de introductie wordt er een beeld gevormd over het afstudeerproject en het overkoepelende project. Daarnaast wordt uitgelegd wat de beweegredenen zijn om dit ontwikkeltraject te starten. Het eerste hoofdstuk gaat over het stage aanbiedende bedrijf en de markten die zij bediend. Het tweede hoofdstuk gaat over het project, de aanleiding en mijn beoogde rol binnen het project.

De tweede sectie bestaat uit het derde vierde en vijfde hoofdstuk en beschrijft de opzet van het project. Tijdens deze hoofdstukken wordt er uitleg gegeven over de project aanpak en systeem ontwikkelmethode, de uitgangssituatie van de ontwerpstrategie en de uitgangssituatie van de eisen die gesteld worden aan het project.

De derde sectie bestaat uit het zesde tot het elfde hoofdstuk en gaat over de project uitvoer. In deze hoofdstukken wordt per te ontwikkelen element uitleg gegeven over de processen die zijn doorlopen en de keuzes die ik gemaakt heb. Per hoofdstuk worden de volgende zaken besproken: Functionele eisen, het ontwerp voor deze story, de bouw van de backend en/of frontend en de tests die bij dit element horen.

De vierde sectie bestaat uit het twaalfde hoofdstuk en gaat over de projectafronding. In deze sectie worden de acties die nog gedaan moeten worden beschreven.

De laatste sectie bestaat uit het dertiende en veertiende hoofdstuk en vormt de conclusie. In deze hoofdstukken geef ik per element aan wat ik zelf voor mening heb over de keuzes, het proces en wat ik eventueel een volgende keer anders zou doen. In het laatste hoofdstuk pak ik vervolgens de gekozen beroepstaken mee en koppel ik terug welke elementen van de beroepstaak in opgepakt heb.

1. CCGroup

1.1. Missie

Onze missie is om de bedrijfsdoelstellingen van onze klanten te helpen realiseren door geïntegreerde systemen te ontwikkelen, die direct inzetbaar zijn, voldoen aan de verwachtingen en lang mee gaan, zodat de klant een degelijke “return on **I**nvestmen**T**” ervaart.

1.2. Werknemers

De CCGroup is de overkoepelende organisatie van 5 verschillende bedrijven die gezamenlijk onder dezelfde naam naar buiten toe treden. Deze bedrijven hebben binnen de overkoepelende organisatie elk een eigen identiteit en focus punt binnen de markt. De totale organisatie van CCGroup bestaat uit ongeveer 70 werknemers. De platte structuur van de organisatie zorgt ervoor dat er snel geschakeld kan worden.

1.3. Markten

De CCgroup is actief op de volgende markten door de bouw van software producten, de ondersteuning van hardware en het detacheren van personeel:

- Travel(reiswereld)
- Healthcare
- Managed systems
- Detachering

Travel

Op de reismarkt is CCGroup actief met de verkoop van touroperatorpakketen en CMS systemen. Deze pakketten worden gebruik bij de in en verkoop van reisproducten.

Healthcare

In de Healthcare tak van CCGroup is ze actief met de verkoop van twee software pakketten: PassClinical en VMOS.

VMOS is een product dat ontwikkeld is om incident management in ziekenhuizen te ondersteunen. Het kan worden gebruikt om incidenten te registeren en te beheren.

PassClinical is een product ontwikkeld om stamboomonderzoek te ondersteunen. Binnen het software pakket kunnen genetische stambomen aangelegd worden waardoor onderzoek naar specifieke genetische problemen gemakkelijker gemaakt worden.

Managed systems

Veel klanten van CCGroup besteden het onderhoud aan fysieke hardware systemen uit aan CCGroup. Deze tak ondersteunt het onderhoud aan servers en desktop systemen.

Detachering

CCGroup doet aan detachering van opgeleide JAVA specialisten, veelal gespecialiseerd in het JAVA cms systeem Hippo.

2. Het Project

In dit hoofdstuk wordt wat beschreven over het gehele project, later wordt er meer in detail ingegaan op mijn opdracht binnen het algemeen overkoepelend project.

2.1. Aanleiding

Binnen de afdeling Travel van CCgroup worden momenteel twee verschillende touroperatorpakketen ontwikkeld en verkocht. Deze pakketten ondersteunen de in-en verkoop van reizen. Deze twee producten zijn DirectTravel(DT1.0) en Dynatravel. Het belangrijkste verschil tussen deze twee producten. zit hem in de reissoort die de touroperator verhandelen wil, hierbij kan onderscheid gemaakt worden tussen de georganiseerde groepsrondreizen en de individuele pakketreizen. Hierbij sluit Dynatravel momenteel beter aan op de pakketreis en DirectTravel bij de georganiseerde groepsrondreis.

De eisen voor beide softwarepakketten lopen ver uiteen door het verschil in organisatie van de reizen. Voor individuele pakketreizen is het van belang dat een touroperator flexibel om kan gaan met zijn accommodaties (hotels/appartementen) en daar vluchten om heen kan plannen. groepsrondreizen zijn vervolgens afhankelijk van de beschikking van reisleiders, vluchten/binnenlandse vluchten en het vervoer van de deelnemers van en naar de “stopover” accommodaties.

Vanuit een van de bedienende touroperators (Fox vakanties) kwam begin 2012 de vraag of kenmerken van pakketreizen en groepsreizen gecombineerd konden worden tot een backoffice systeem. Het project (DirectTravel 2.0/DT2.0) zou vooral doorbouwen op de kennis die opgedaan is in de ontwikkeling en onderhoud van het groepsrondreizen backoffice systeem DirectTravel. Met deze kennis is in 2012 een projectplan met functioneel ontwerp gemaakt waarin de functionaliteiten zover uitgewerkt zijn als de kennis van toen het toeliet. Dit heeft in 2012 geresulteerd in een functioneel ontwerp dat slechts de basis functionaliteit van het uiteindelijk te ontwikkelen product vormde. De ontwikkeling van DT2.0 is opgesplitst in 10 opeenvolgende thema’s.

Het eerste thema is presales, een thema dat gaat over de opvoer van contracten voor reisonderdelen (vluchten, transfers, accommodaties, excursies, reisleiders). Gevolgd door het combineren van diverse reisonderdelen tot een reisproduct en het berekenen van een prijs voor het reisproduct voor de diverse vertrekdata (afhankelijk van beschikbaarheid van reisonderdelen).

Het tweede thema is Sales, een thema dat gaat over de verkoop van de samengestelde reisproducten, en het afboeken van contractueel afgesproken voorraden.

Het derde thema is de Post-Sales, hierin valt de informatievoorziening richting de kopers van reisproducten en de aanbieders van de diverse reisaanbieders.

De eerste drie thema’s vormen een oplossing om de corebusiness van een touroperator mogelijk te maken, de 7 andere thema’s (Beheer contacten, Financiële afhandeling, marketing, reportage, backoffice, ontsluiting naar externe applicaties en reisleiding) dienen daarbij een direct ondersteunende rol.

Het ontwikkelteam heeft het grootste gedeelte van de pre-sales en de sales inmiddels opgeleverd en werkt aan het afmaken van deze thema's en de ontwikkeling van de post-sales en de ondersteunende thema's.

2.2. Opdracht

Ik ga binnen het DT2.0 project werken aan de ontwikkeling van een module die gericht is op het afhandelen van correspondentie. Deze module is binnen de functionele ontwerpen van DT2.0 beschreven in het functioneel ontwerp van het thema Post-sales. De module dient de communicatie met de klanten en relaties van Fox vakanties te faciliteren. De functionele eisen van de "correspondentiemodule" zijn vanuit het functioneel ontwerp samen te vatten in de volgende userstories.

Userstories

1: Verstuur Queues

Deze userstory gaat over het opbouwen van verzendmomenten en het kunnen opgeven voor verwerking van deze taken. De taken die nog verzonden moeten worden of reeds verzonden moeten doorgezocht kunnen worden. De taken die nog verzonden moeten worden moet het gekoppelde document van vervangen kunnen worden.

2: Selectie print/email

Deze userstory beschrijft de eisen die er gesteld worden aan de communicatie die vanuit de Correspondentiemodule verzonden kan worden. Standaard dient er de mogelijkheid er de mogelijkheid te zijn om te printen of e-mailen.

Deze userstory beschrijft de functionele eis dat de Correspondentiemodule moet kunnen communiceren. Hierin staat beschreven dat er de mogelijkheid moet bestaan voor het afdrukken van pdf's, en het kunnen versturen van pdf's.

3: Standaardopmaak Communicatie

Deze Userstory betreft het kunnen instellen van specifieke opmaak en afhandeling van de taken die aangemeld worden bij de correspondentie module. Hierbij kan gedacht worden aan de opmaak van een email, de selectie van een printer en printerinstellingen en de keuze voor een gestandaardiseerd afhandelingsmoment.

4: Historie

Deze userstory dient het doorzoeken van een queue met afgehandelde zaken mogelijk te maken alsmede het kunnen her-uitvoeren van niet geslaagde of afgeronde taken.

5: Communicatie types

Deze userstory is een overzicht van de verschillende correspondentietypen die afgehandeld moeten kunnen worden door de module.

5: Versturen facturen

Deze userstory is een specificatie van een communicatie type in de uitvoer is deze gevoegd bij de verwerking van andere communicatie types omdat er in verwerking geen verschil is met andere communicatie types.

5: Offerte stadium van boeking

In deze userstory is het uitbrengen van een offerte alvorens een reservering om te zetten in een boeking, dit in verband met het consumenten recht in België. Deze userstory komt uit het Sales thema en de aan de correspondentiemodule gerelateerd eisen zijn in te passen in de userstory communicatie type omdat het een gestandaardiseerde afhandeling onder te brengen is.

Product

Uiteindelijk dient er voor het project een product ontwikkeld te worden die voorziet in de mogelijkheid tot afhandeling van aangemelde taken op 2 fundamenteel verschillende wijzen (afdrukken / emaillen) met een diversiteit aan gebruiker instelbare sub afhandelingen. Hiervoor dient een gebruiker de diverse opties te kunnen beheren, de aangemelde taken te kunnen beheren en de hoeveelheid sub opties en terugval mogelijkheden te kunnen instellen.

3. Plan van aanpak

In het plan van aanpak heb ik beschreven op welke wijze ik bepaalde problemen aan wil gaan pakken, welke planning ik hanteer om zaken als afgerond te kunnen beschouwen en naar welke eindsituatie er gewerkt gaat worden. Hiervoor heb ik gebruik gemaakt van een template voor een Plan van aanpak en hier de stukken uitgehaald die voor het project toegevoegde waarden hebben.

Doelstelling

De doestelling van het plan van aanpak is het inzichtelijk maken van de risico's, de methoden en technieken die gebruikt dienen te worden.

Systeem ontwikkelmethode

De keuze voor een systeem ontwikkelmethode is van te voren in het afstudeerplan opgenomen. Aangezien het project aansluit op een reeds bestaand project en het een onderdeel van een systeem dient te worden wat reeds in ontwikkeling is. Hierdoor is ook gekozen om aan te sluiten bij de systeem ontwikkelmethode van dit project(SCRUM) en de daarbij behorende reeds ingerichte tools en processen te hergebruiken. De keuze om SCRUM op dezelfde wijze toe te passen zorgt ervoor dat dit project meeloopt in de huidige planning, het project toegevoegd kan worden aan de sprintplanning, sprint burndownchart en de backlog van het overkoepelende project. Een ander bijkomende voordeel is dat ik mee kan lopen in de dagelijkse planning, en de sprint specifieke sprintplanningen en daarmee ervaring kan opdoen met deze systeem ontwikkelmethode binnen een groter project. Uiteindelijk zal ik mij gedurende het afstudeertraject exclusief bezighouden met het ontwikkelen van de userstories zoals vooraf in de opdrachtomschrijving gedefinieerd.

3.1. Planning

Ik heb met het maken van de planning samen met de opdrachtgever de keuze gemaakt om de sprints van mijn project parallel te laten verlopen aan de sprints die gekoppeld zijn aan de ontwikkeling van DT2.0. Dit betekend dat de oplevering van mijn tussenproducten gekoppeld is aan de oplevering van de sprints aan de klant. Dit biedt mij eveneens de mogelijkheid gebruik te maken van de diverse scrum ondersteunende tools die reeds ingericht zijn voor dit project.

De volgorde waarin de userstories opgepakt worden wordt vastgesteld voor het verlopen van de sprint en heeft te maken met de volgende factoren:

- De te verwachten complexiteit van een userstory
    - Hogere verwachte complexiteit betekend het eerder oppakken van de userstory
        - het gebruik van methoden en technieken die voor mij onduidelijk zijn
- De kans op project bedreigende factoren
    - Onmisbaar voor de basis functionaliteit(versturen en afdrukken)
    - Onmisbaar voor de wachtrij functionaliteit(verwerkmoment)
- Impact op de rest van het project
- Afhankelijkheid van deze userstory naar de ontwikkeling van andere userstories binnen de correspondentie module

Ik heb opgenomen hoeveel userstories ik per sprint binnen de huidige sprint cycli ga doen. De in de volgende sprint uit te voeren userstories worden een week voor het verlopen van de huidige sprint bepaald. In het begin van het project heb ik in de planning ruimte gereserveerd om de userstories zoals in het afstudeerplan opgenomen na te lopen op onduidelijkheden en onlogische eisen. Voor het

begin van elke sprint is vervolgens ruimte gereserveerd om functioneel de userstories met de stuurgroep en product-owner vast te stellen en bij te schaven.

De standaard duur van de sprints binnen het DT2.0 project zijn vier weken. In deze vier weken werk ik aan Userstories die relevant zijn aan de Correspondentie Module. De volgordelijkheid van de userstories is afhankelijk van de prioritering die aan de Userstories toegekend is. De prioritering wordt door de productowner(en stuurgroep) in overleg met mij bepaalt door een afweging te maken van de bovengenoemde afwegingsfactoren.

Pre-Sprint(week 1-2)

De eerste twee weken van het afstuderen stonden in het teken van de project voorbereiding. Dit kwam omdat het overkoepelend DT2.0 Project aan de tweede week van haar huidige sprint bezig was. Deze eerste halve sprint heb ik mij beziggehouden met de voorbereiding. Hierin zijn de processen doorlopen die de rest van het project kunnen stroomlijnen. In deze fase zijn de volgende milestones opgeleverd:

- Plan van aanpak
- Functioneel ontwerp(basis)
    - Uitwerking van de userstories in het eerste functionele ontwerp.
- Technisch ontwerp(basis)
    - Eerste basis opzet van een globaal klassediagram

Sprint 1(week 3-6)

Voorafgaand aan de eerste vollesprint heb ik samen met de opdrachtgever een keuze gemaakt voor de uitvoer van een proof of concept om te bewijzen dat het mogelijk is om een afdruk te maken op basis van de te verwachtten invoer, en deze vervolgens zonder verder tussenkomst af te drukken op een ingesteld gekoppeld apparaat.

Daarnaast heb ik samen met de opdrachtgever de keuze gemaakt om twee essentiële userstories zover op te pakken dat de applicatie instaat gesteld wordt op een tijdstip een afdruk/email te versturen.

De twee userstories die in de eerste sprint opgepakt worden zijn de verstuur queue en de userstory selectie print/e-mailen. Deze twee userstories vormen volgens de gestelde afwegingseisen de userstories die als eerste opgepakt dienen te worden.

Voor de interne sprintplanning maken de geselecteerde userstories onderdeel uit van de sprint planning en gaan ze uiteindelijk mee in de sprint demonstratie aan het einde van de sprint.

De specifieke milestones die opgeleverd worden vanuit mijn deel van de sprint deze sprint zijn:

- Proof of concept
- De correspondentie module
- Aangescherpt functioneel ontwerp
- Technisch ontwerp van de applicatie tot nu toe
- Test document

Sprint 2(week 7-11)

Aan het einde van de voorgaande sprint heb ik samen met de opdrachtgever de selectie gemaakt van de userstories die in deze en de volgende sprint opgepakt dienen te worden. Gebruik makend van de kennis en de basis die ik in de eerste sprint gelegd heb met de bouw van de voorgaande userstories zijn de userstories voor deze sprint vastgesteld op: Standaard opmaak Communicatie en Historie. In deze sprint heb ik aan het einde eveneens een set scopechanges doorgevoerd in de ontwikkeling van de twee voorgaande userstories.

De specifieke milestones die opgeleverd worden vanuit mijn deel van de sprint deze sprint zijn:

- De correspondentie module
- Aangescherpt functioneel ontwerp
- Technisch detail ontwerp
- Test document

Sprint 3(week 12-16)

In de laatste sprint heb ik de overgebleven userstories opgepakt waarbij de userstories als dusdanig aangepast zijn vanaf het originele functionele ontwerp dat deze kon samenvoegen in een oplossing. Eveneens heb ik in deze sprint een heleboel scopechanges doorgevoerd en diverse tests applicatie omvattende tests uitgevoerd.

De specifieke milestones die opgeleverd worden vanuit mijn deel van de sprint deze sprint zijn:

- Afgeronde applicatie
- Technisch ontwerp
- Test rapport

4. Functioneel ontwerp (basis)

4.1. Uitgangssituatie

DirectTravel 2.0 is opgedeeld in thema's die elk beschreven zijn in een Functioneel ontwerp(FO). Het thema waar de opdracht betrekking over heeft is het postsales thema, het thema dat gaat over de afhandeling van verkochte producten en de communicatie over de verkoop richting de klant en de leveranciers. Het functioneel ontwerp en de userstories die daarin genoemd zijn vormen de leidraad voor het functioneel ontwerp van de correspondentiemodule. Deze userstories zullen worden geanalyseerd, aangevuld en daar waar nodig verhelderd zodat de eisen beter aan sluiten bij reële wensen en mogelijkheden. Daar waar de eisen nog niet uitgebreid genoeg zijn zullen er toevoegingen gemaakt worden. Alle eisen moeten door de productowner en de stuurgroep goedgekeurd worden.

Een voorbeeld van een userstory uit het Postsales FO:

Selectie printen of mail

Er kan in DirectTravel 2.0 worden aangegeven of de communicatie met een klant digitaal (via email) of analoog (via de post) verloopt. Wanneer een klant kiest om de communicatie digitaal af te handelen dan zal er een mail worden verstuurd om aan te geven dat het document klaar staat in het dossier van de klant.

Kiest een klant voor het analoog afhandelen van de communicatie dan zullen alle documenten uitgeprint worden om deze vervolgens via de post te versturen. In dit geval zullen de documenten alsnog beschikbaar worden gesteld in het online dossier van de klant

De optie voor 'digitaal' of 'per post' wordt op boekingsniveau ingesteld. Bij een documenttype kan door de touroperator worden aangegeven of het digitaal en per post verstuurd mag worden. Wanneer de klant bij een boeking kiest om alles per post af te handelen zullen enkel de documenten waar dit bij is aangegeven geprint worden en doorgestuurd. De overige documenten zullen digitaal worden verstuurd.

Bijvoorbeeld alleen de reisbescheiden mogen per post worden verstuurd. Wanneer een klant een boeking maakt en aangeeft alles per post te willen ontvangen, dan zal dit enkel gelden voor de reisbescheiden. De overige communicatie, zoals de factuur, zal digitaal gebeuren.

Het is mogelijk om automatisch een korting op te voeren wanneer gekozen wordt om een boeking digitaal af te handelen. Er kan ook automatisch een extra fee opgevoerd worden wanneer er gekozen wordt voor het per post afhandelen.

Bijvoorbeeld bij de keuze 'per post' op een boeking zal er een extra boekingsonderdeel worden opgevoerd van 15 euro. Bij de keuze 'digitale afhandeling' zal er een korting van 15 euro als negatief bedrag op de boeking worden getoond.

Bedrijfsregels selectie printen of mail

Het is mogelijk om in DT aan te geven dat de optie per post wordt geselecteerd, zonder dat hier extra kosten voor verrekend worden.

4.2. Opstellen basis document

Ik ben begonnen met het samenvoegen van alle userstories die vooraf bij het opstellen van de afstudeeropdracht al benoemd waren. Twee van deze userstories waren niet in het postsales FO terug te vinden maar als wijziging aangemeld op de sprint backlog. De andere userstories heb ik bekeken en gekeken wat ontbreekt om de user-story voor mij en de product-owner duidelijker te kunnen maken wat de daadwerkelijke functionele eisen van deze story zijn. Het reeds bestaande functionele ontwerp(opgesteld bij aanvang van het project in 2012) heb ik als leidraad gebruikt voor de functionele eisen.

Uiteindelijk heb ik in deze basis het besluit genomen om enkele zaken anders te verwoorden om de duidelijkheid te vergroten en een aantal inconsistente zaken weg te halen. In het voor traject heb ik het samenstellen vooral gebruikt om voor mijzelf een zo helder mogelijk beeld te creëren over de zaken die meegenomen dienen te worden in het ontwikkeltraject.

4.3. Uitwerking userstories

De userstories waar het project direct betrekking op heeft zijn de userstories die de methodieken van communicatie beschrijven. Hier volgt de korte beschrijving van de essentie waar de userstory uit bestaat.

1: Verstuur Queue

Het printen van de selectie/het versturen van de selectie/ het in SharePoint beschikbaar maken voor de klant van de selectie. Bij het optreden van fouten(in het print/e-mail proces) deze toevoegen aan een fout queue en later via de selectie module opnieuw kunnen “queuen”

2: Selectie printen/mail

- Het automatisch versturen van correspondentie aan de hand van ingesteld parameters(zie standaard opmaak module) of gedefinieerde parameters

3: Standaard opmaak module

- De standaard opmaak module userstory heeft betrekking op het kunnen inregelen van de afhandeling bij het communiceren over een communicatie kanaal(email/post). En geeft daarbij invulling op de instelbaarheid van de volgende zaken voor communicatie via email:
    - Het moment van behandeling(verstuurmoment)
    - Het onderwerp van de email
    - De opbouw van de multipart email(html en platte tekst) en logo’s
- Via de post:
    - Het moment van behandeling(printmoment)
    - De selectie van de printer(per document verschillend)
    - De opbouw van enkele printerinstellingen

4: Historie

- Het inzichtelijk maken van de bij de correspondentiemodule aangemelde taken:
    - Te versturen/printen
    - Aan het versturen/printen

    - Foutieve taken
    - Afgeronde taken(waarin gezocht kan worden op basis van identificatie kenmerken(boekingsnummers, ontvanger gegevens
- Het bieden van de mogelijkheid om taken aan te passen:
    - Aanpassen moment van versturen voor taken die nog verstuurd/geprint dienen te worden
    - Opnieuw toevoegen van taken die foutief uitgevoerd zijn en daarbij de mogelijkheid bieden wijzigingen te maken die zorg dragen dat de uitvoering wel goed gaat(aanpassen printer instellingen/ontvangst adres)
    - Opnieuw toevoegen van uitgevoerde taken en de mogelijkheid bieden wijzigingen te maken.

5: Communicatie types

Offerte stadium van boeking

- Het opstellen van gestandaardiseerde offertes op basis van de aan de module aangeleverde gegevens.

Versturen van facturen

- Het opmaken en versturen van facturen.

Herinneringsmail ontbrekende gegevens

- Het versturen van een herinneringsemail aan de afnemers van reisproducten wanneer er gegevens ontbreken.
-

Ik heb eerst een opzet gemaakt van de userstories zodat het voor mij een duidelijker geheel is, hierna heb ik deze samen met de productowner nog een keer bekeken en daar waar nodig functioneel aangepast. Deze userstories zijn vervolgens nog teruggekoppeld aan de stuurgroep en goedgekeurd.

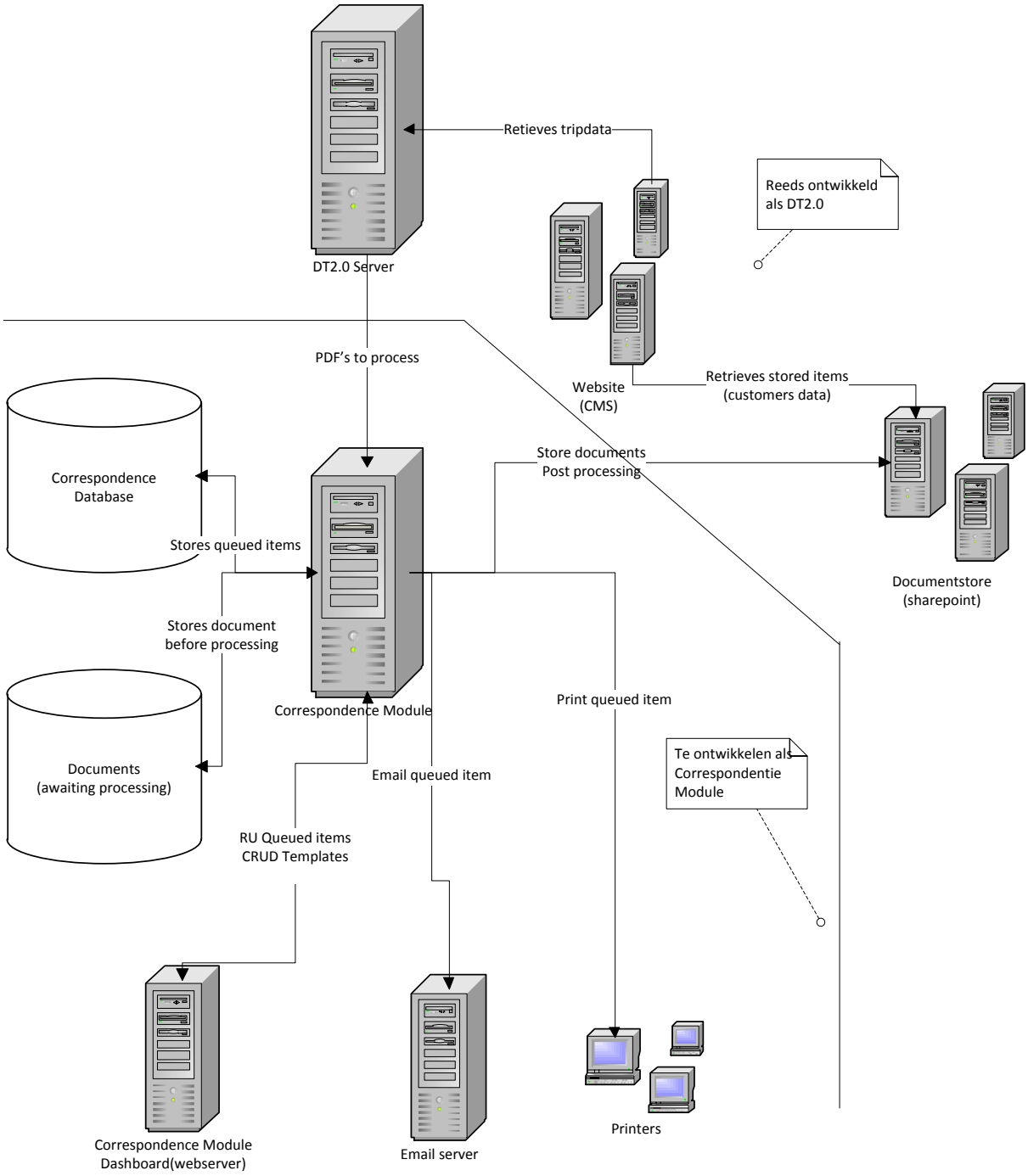
Na het definitief vastzetten van de functioneel eisen heb ik een opzet gemaakt van de basis structuur die in eerste instantie de bedoeling had om de basis functionaliteiten te kunnen realiseren. De structuur die opgesteld had wordt in de hoofdstukken bij de userstory waar het stuk betrekking op heeft beschreven.

5. Technisch ontwerp (basis)

De basis van het technisch ontwerp wordt gevormd door de principes waarop de Correspondentie module gebouwd dient te worden. De architectuur speelt hierin een belangrijke rol alsmede de opbouw van de packages. In de opbouw van de packages wordt gebruik gemaakt van Domain Driven Design. De Systeemarchitectuur is gestoeld op de gedachte dat bepaalde elementen voor andere eindgebruikers anders in te richten zijn.

5.1. Systeem architectuur

De systeem architectuur(figuur 5-1) wordt gekenmerkt door een onafhankelijkheid van de rest van DirectTravel. Er wordt voor de Correspondentie Module gebruik gemaakt van een los gekoppelde architectuur. De afhankelijkheid van andere software systemen wordt daardoor zo klein mogelijk gehouden. Dit biedt als voordeel dat de module door andere software systemen herbruikbaar is. De enige directe afhankelijkheid van een andere software systeem is de documentstore(sharepoint) oplossing. De afhankelijkheid die hieruit ontstaat dient programmatisch zo klein mogelijk gehouden te worden.

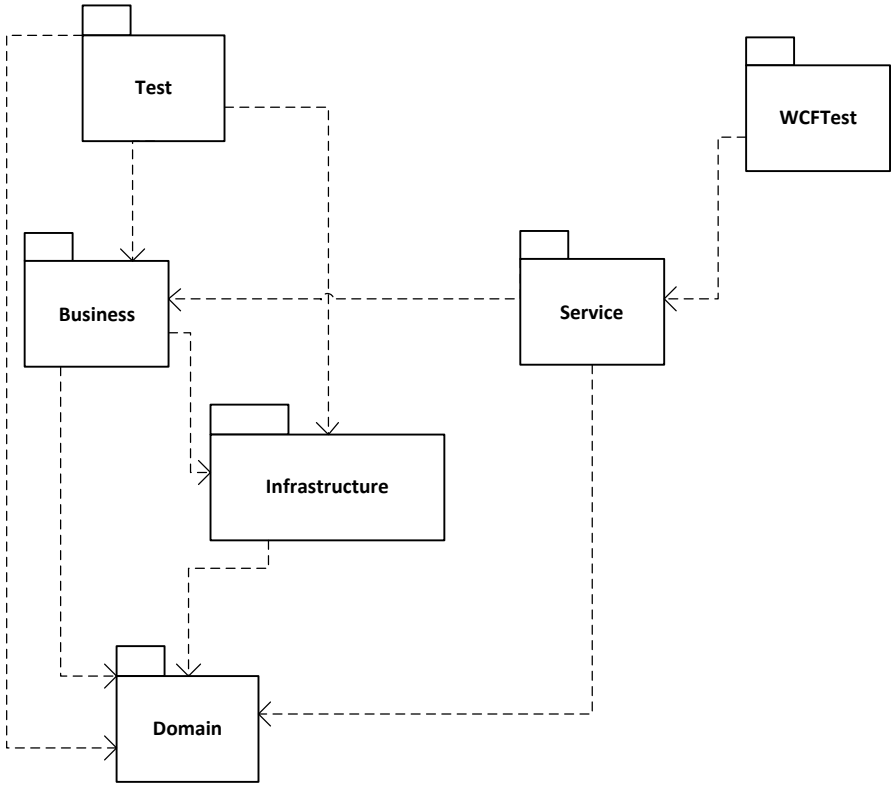


Figuur 5-1

5.2. Ontwerp filosofie

De reeds gebouwde applicatie onderdelen van DirectTravel 2.0 zijn gebaseerd op principes van Domain Driven Design(DDD). De applicatie die ik ontwikkel maakt gebruik van dezelfde principes om de overdracht gemakkelijk te houden.

DDD zoals toegepast op de ontwikkeling van DirectTravel 2.0 maakt gebruik van de volgende scheiding in packages(5-2).



Figuur 5-2

In het domain packages zitten alle objecten die de gegevens van de applicatie vormen, deze objecten worden door andere packages gebruikt om informatie te delen.

De service is waar de applicatie door de buitenwereld te benaderen is, en via bied via “Domain Transfer Objects”(DTO) objecten aan via de WCF service. De DTO zijn vaak transformaties van de domein objecten die getransformeerd worden in de service.

De business package bevat de bedrijfslogica en maakt gebruik van de infrastructure voor communicatie met externe afhankelijkheden (databases, andere packages, utility packages, printers, mailservers).

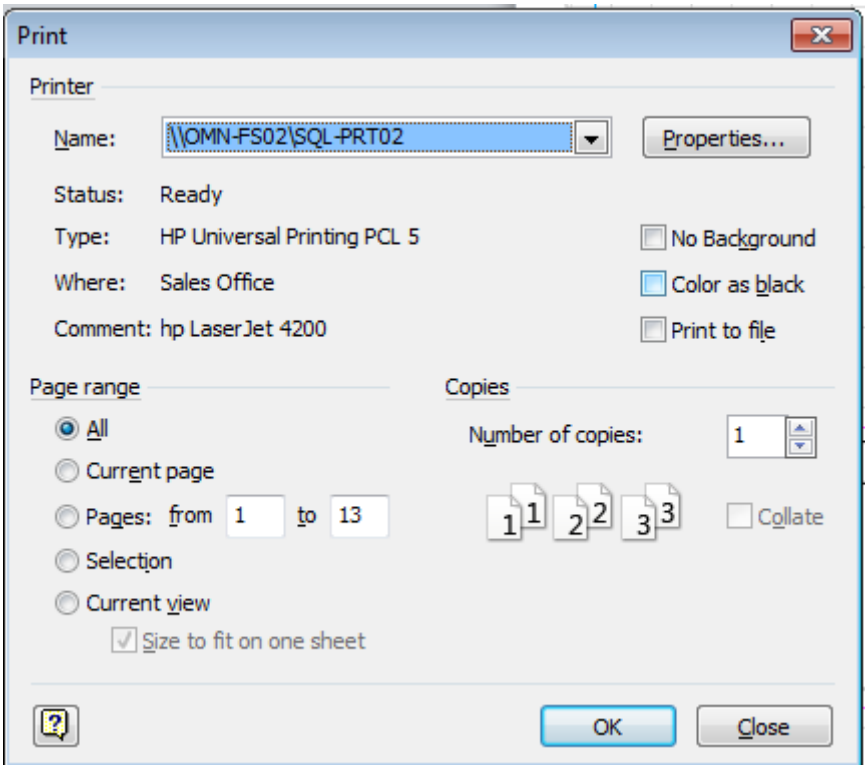
De twee testpackages die gedefinieerd zijn worden gebruikt voor in package testing(create read update delete unit tests, specifieke methoden in de bedrijfslogica). De WCFTest package test de service methoden.

DDD is gebaseerd op de scheiding van domein objecten van de bedrijfslogica en maakt het voor de ontwikkelaar gemakkelijk om een Object Relation Mapper in te zetten voor persistentie van de

domein objecten. Het basis domein model in deze applicatie wordt gevormd door een set interfaces en basis klassen die reeds gedefinieerd zijn in het functioneel ontwerp.

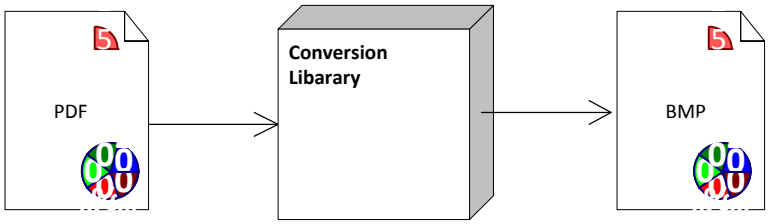
6. Proof of concept afdrukken pdf

Voordat ik kon beginnen aan het technisch ontwerp heb ik eerst in overleg met de opdrachtgever een proof of concept gemaakt. Dit proof of concept moet aantonen dat het mogelijk is een afdruk te maken van een PDF zonder het tonen van een afdrukscherm zoals gebruikelijk. Normaliter dient er een alvorens een afdruk te maken een venster(figuur 6-1) getoond te worden waarin er diverse printerinstellingen aangepast kunnen worden.



Figuur 6-1

Het uiteindelijk doel van het POC is aan te tonen dat het mogelijk is om PDF bestanden van meerdere pagina's af te drukken zonder tussenkomst van een persoon.



Figuur 6-2

Uiteindelijk dient een PDF alvorens deze geprint wordt geconverteerd te worden naar een format wat de printer kan printen. Een standaard format die door elke moderne printer af te drukken valt is de RichTextFile(RTF) of Bitmap(BMP). Waar bij RTF als formaat alleen geschikt is voor teksten. Eventuele logo's en andere plaatjes forceren al snel het gebruik van een BMP.

6.1. Eisen POC

De complexiteit ligt in het automatiseren van een afdruk zonder tussenkomst van het in vrijwel elk programma gebruikelijke printerinstellingen scherm. Aangezien in een latere userstory de mogelijkheid beschreven was om dit in te kunnen stellen moest ik in overleg met de opdrachtgever voor deze proof of concept aantonen dat:

- Een pdf van meerdere pagina's afgedrukt kon worden
- Er een selectie van printers gemaakt kon worden
- De pdf vervolgens per pagina uit de geselecteerde printer komt.
- De afdruk nauwelijks tot geen kwaliteitsverlies heeft geleden te opzichte van het origineel.
- De proporties hetzelfde zijn als op het afdrukvoorbeeld van de pdfviewer.

6.2. Bouw

Aangezien het enkel een proof of concept betreft heb ik gekozen om mij puur op de functionele eisen te richten, hierdoor heb ik gekozen om geen selectie van een document toe te voegen maar de test te baseren op een pdf versie van mijn planning. Deze bestaat uit vier pagina's en is door de rijke kleur, verschillende verticale en horizontale lijnen goed te vergelijken met het afdrukvoorbeeld.

Voor het afdrukken heb ik gekeken naar diverse mogelijkheden om af te drukken. Ik heb gezocht op de mogelijkheden om documenten met een .doc of .odf(word document format en het opendocument format) opmaak af te drukken zonder tussenkomst van een gebruiker. De enige mogelijkheden van afdrukken die ik tegenkwam waren heft afdrukken van richtext documenten(txt) zonder plaatjes of het afdrukken van bitmap objecten(alleen plaatjes).

De zoektocht heeft mij vervolgens doen besluiten mij verder te gaan verdiepen in het afdrukken van bitmap afbeeldingen omdat die dichter bij de eis van het afdrukken van PDF(documenten met plaatjes ligt) dan het afdrukken van alleen tekst. Als eerste mechaniek heb ik de applicatie gebouwd om een bitmap afbeelding op een printer naar mijn keuze af te kunnen drukken. De applicatie bood mogelijkheid tot selectie van een printer door middel van een dropdown van alle geïnstalleerde printers op een computer.

Selectie van conversie bibliotheek

Aangezien er op het internet diverse oplossingen geboden worden voor het converteren van PDF documenten naar andere formaten met bepaalde voor en nadelen heb ik meerdere van deze bibliotheken getest. De volgende bibliotheken zijn in het proof of concept gebruikt:

- LibPDF
- PDFRasterizer
- Bytescout PDF renderer
- Aspose PDF
- Ghostscript

Deze bibliotheken zijn degene die ik via google gevonden heb die mij instaat stelden een conversie van PDF naar BMP te doen in het geheugen van een machine, deze bilbiotheken konden vanaf een bytestream een conversie uitvoeren en hoefden daarbij tijdens de conversie geen gegevens naar disc(harde schijf) te schrijven, en vervolgens van disc te halen na conversie.

Ik heb een selectie gemaakt van eisen en deze vervolgens aan de opdrachtgever voorgelegd. De eisen zoals door de opdrachtgever goedgekeurd zijn:

- Conversie kwaliteit
    - De conversie moet zo plaats vinden dat er visueel geen verlies in kwaliteit van de PDF plaatst vindt.
- Snelheid
    - De snelheid dient van dusdanig niveau te zijn dat de conversie niet zorgt voor vastlopende threads.
- Externe installatie
    - Gewenst is om een oplossing te zoeken zonder extra installatie
- Consistentie conversie
    - De conversie dient consistent te zijn(andere pagina volgorde moet hetzelfde resultaat bieden)
- Stabiliteit
    - De bibliotheek dient stabiel te zijn, bij grote parallelle operaties dient de bibliotheek te blijven functioneren

De eerste test met de 5 bibliotheken vielen bytescout en libpdf af omdat het resultaat wat ze genererende vertekend werd met de pagina breedte, hierdoor werd de breedte paramater van de af te drukken pagina niet goed weergegeven en kwam er een niet goede afdruk.

In de tweede test heb ik de pagina's handmatig omgedraaid en beleven de overgebleven 3 bibliotheken zonder problemen functioneren.

Bij de snelheidstest heb ik gekeken naar het converteren van een vrij groot document naar bitmap(20 pagina's) deze heb ik vervolgens naast de profiler van Microsoft visual studio gehouden en gekeken hoeveel systeem resources ze verbruikten, en hoelang de conversie duurde. De pdf rasterizer bleef hierbij achter in snelheid. Het converteren van dit bestand duurde voor de rasterizer bijna 4 seconden, terwijl Aspose PDF en Ghostscript beide onder de 2 seconden bleven in de conversie. Deze test heb ik ook gebruikt om te bepalen of de bibliotheken stabiel bleven, en geen van beide was in deze een probleem met stabiliteit(20x deze taak simultaan uitvoeren).

Uiteindelijk was voor Aspose pdf de doorslag dat een andere deel van de bibliotheek al gebruikt werd in de bibliotheek het doorslaggevende argument om daarvoor te kiezen. De licentie kosten moesten dus al voldaan worden indien het andere product gebruikt wordt.

Uiteindelijk had ik een Proof of concept dat bitmap afbeeldingen kon printen, en een methode om pdf documenten per pagina goed om te zetten naar een bitmap afbeelding.

7. Userstory 1: Verstuur queue

7.1. Functionele eisen

Voor alle communicatie kan worden aangegeven op welk moment deze verstuurd moet worden.

- Voor alle communicatie die verstuurd/uitgeprint dient te worden door de correspondentie module kan handmatig een moment van versturen opgegeven worden.
    - Een afnemer van de functionaliteit van de Correspondentie Module wordt instaat gesteld een taak aan te maken met een ingestelde tijd.
- Voor alle gegenereerde communicatie die verstuurd dient te worden door de correspondentiemodule kan een ingesteld moment van versturen ingesteld worden.
    - Het instellen van het verzend moment is afhankelijk van tijdstip van aanbieden bij de correspondentie module en de instelling die opgegeven is als verzendmoment. De volgende momenten zijn instelbaar:
        - Direct versturen(toegevoegd aan de eerst volgende batch die verstuurd dient te worden.
        - Dagelijks versturen op tijdstip x(b.v. dagelijks om 6uur versturen)
        - Wekelijks versturen op tijdstip x(b.v. elke maandag om 5 uur)
    - Het verzend moment wordt opgehaald uit een set kenmerken die bij het aanmelden van een taak meegegeven worden. Deze kenmerken zijn instelbaar.

De CM biedt dus de mogelijkheid om bij aanroep een tijd op te geven voor verwerking of gebruik te maken van de mogelijkheid om dit geautomatiseerd erbij te zoeken.

In de CM is het mogelijk een overzicht op te vragen van taken in de wachtrij, dit kan gebruikt worden om een overzicht te genereren van taken, hun verwerkingsmoment en de methode van verwerking.

Het is mogelijk om een het verwerkingsmoment van een taak aan te passen.

Het is mogelijk om de documenten die gekoppeld zijn aan een taak op te vragen en indien een gebruiker deze aan past het reeds gekoppelde document te overschrijven met een aangepast document.

Er kan gezocht worden in de correspondentie die momenteel in behandeling is alsmede de communicatie die reeds doorgevoerd is.

De communicatie die in de queue staat kan gefilterd worden om snel een specifiek onderdeel te vinden. Het filteren kan gebeuren op:

- Klantnummer
- Naam
- Dossier nummer
- Type communicatie

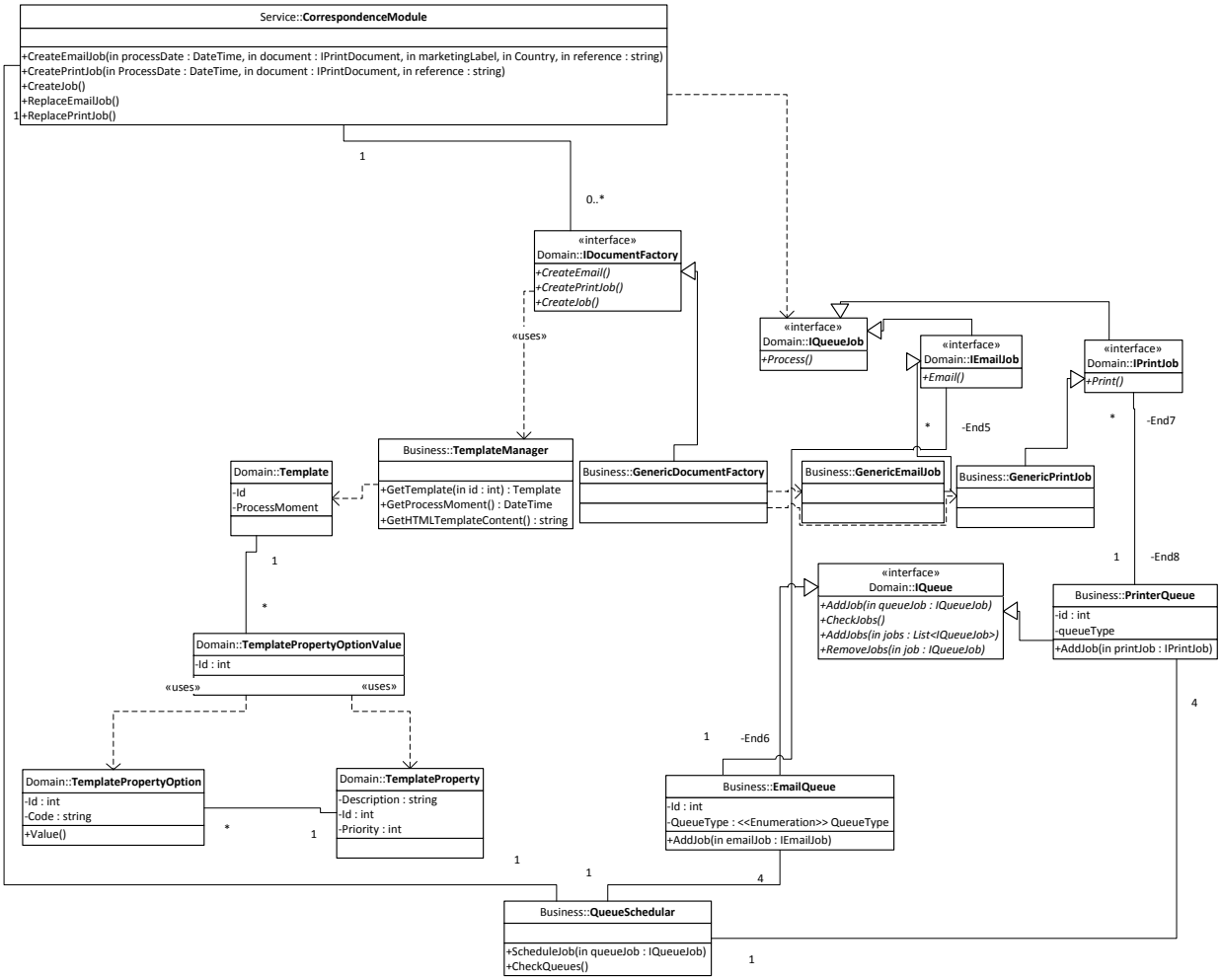
7.2. Globaal ontwerp

Deze userstory vormt het functionele hart van de applicatie, hierdoor is het van belang om het aanmaken en verwerken van taken zo abstract mogelijk te laten verlopen. In deze userstory wordt nog niet het daadwerkelijk verwerken van taken opgepakt, er wordt slechts gekeken naar de benodigdheden om de taken van afhandelingsstatus te laten veranderen na “virtuele” verwerking.

De keuze voor het gebruik van Domain Driven Design(DDD) heeft direct invloed op de opbouw van de klassen en de verantwoordelijkheden van een klasse. Ik heb omdat ik de keuze gemaakt heb voor DDD heeft het klasse diagram wat hierop volgt de volgende package indeling:

Domain Package: alle basis klassen en interfaces

Business Package: alle klassen waar bedrijfslogica in opgenomen is.



Figuur 7-1

Service Package: Correspondence Module, DataTransferObjecten(DTO) van de domain objecten. DTObjecten zijn serieliseerbare data objecten die gebruikt worden in de communicatie met de service en die de service teruggeeft als een complex object terug de service te delen.

Dit klassediagram(7-1) toont de eerste opzet van de applicatie, er zijn in dit klassediagram een aantal keuzes terug te vinden die de verdere ontwikkeling van de applicatie mogelijk moeten maken.

De keuze voor een enkele service, voor het toevoegen van de taken aan de wachtrij, er is hier gekozen voor een enkele klasse boven een functionele scheiding van services.

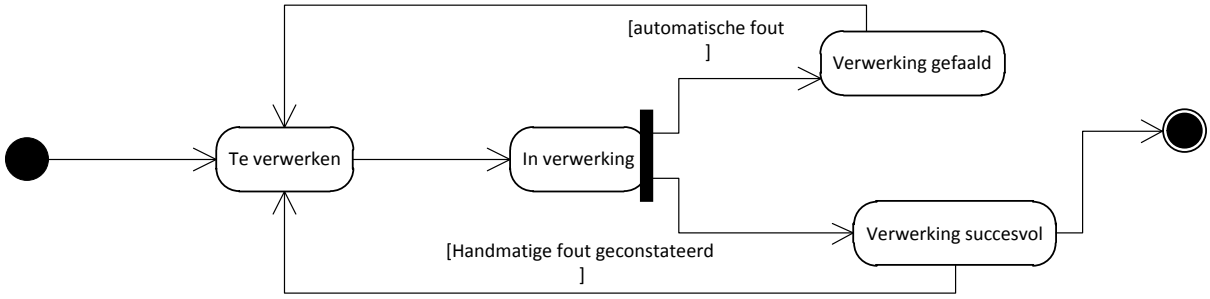
De keuze voor een Singleton design pattern voor de QueueScheduler, de specifieke keuze hier voor komt door twee redenen. Omdat ik de hoeveelheid verschillende wachtrijen wil minimaliseren, zodat het mogelijk is om de wachtrijen apart van elkaar te beheren. De tweede reden komt grotendeels door de voorkennis die ik heb opgedaan tijdens de bouw van het proof of concept, de vertraging die een taakuitvoer kan genereren betekend dat de taken vanuit een centraalpunt te benaderen moeten zijn indien er een splitsing tijdens de uitvoer gebruikt moet worden.

Ik heb eveneens gekozen om in eerste instantie een AbstractFactory pattern te gebruiken voor het aanmaken van de taken om de flexibiliteit te kunnen waarborgen dat het later in het project mogelijk is om correspondentie typen toe te voegen. Getoond wordt in het globale klassediagram hierboven de implementatie met een GenericDocumentFactory, GenericPrintJob en een GenericEmailJob.

7.3. Technisch ontwerp

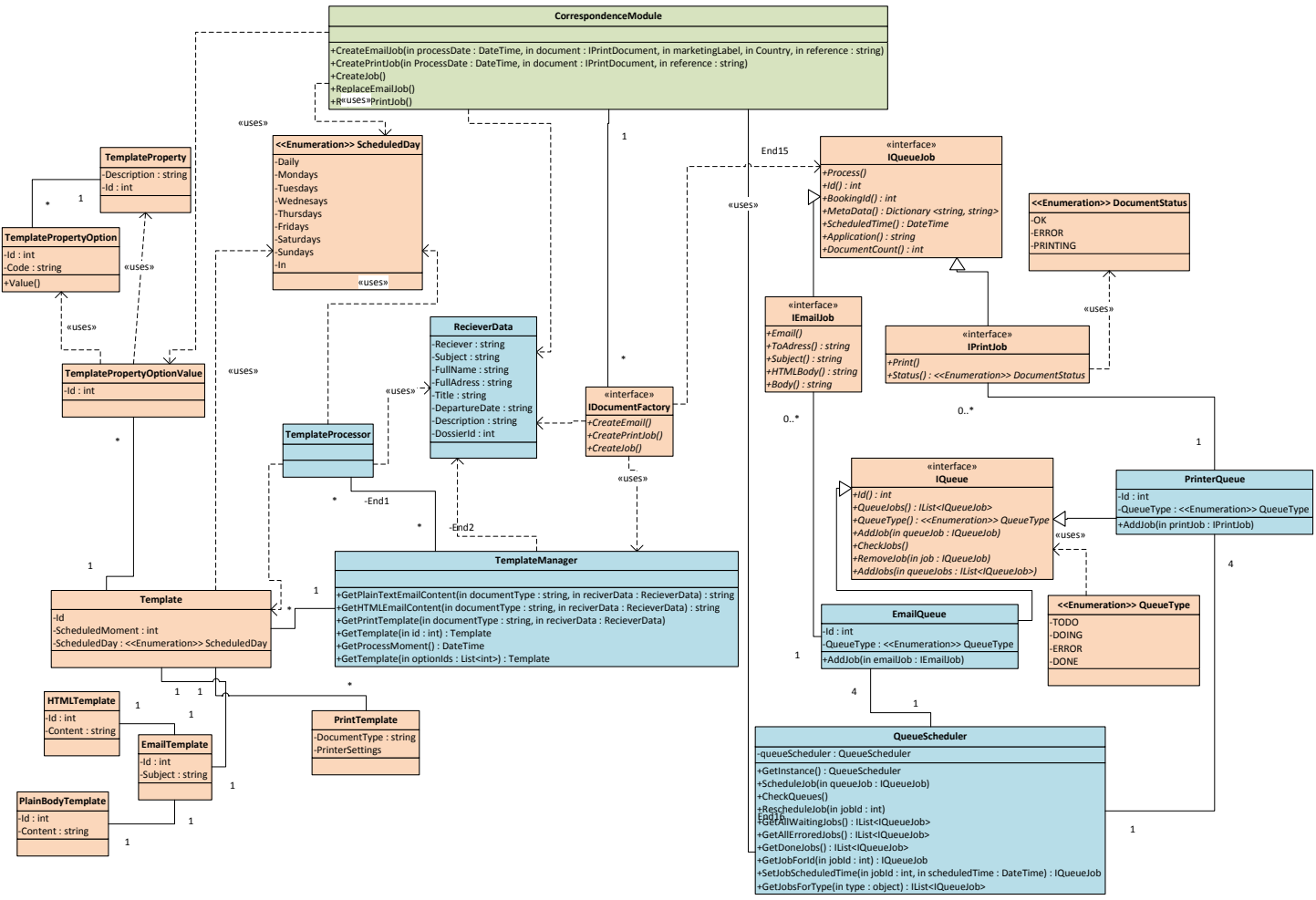
De userstory Verstuur queue gaat over het klaar zetten van communicatie naar de eindklant en het inzichtelijk maken van de taken die gedaan moeten worden. Hiervoor moet het mogelijk zijn om de taken in verschillende stadia van verwerking bij te houden. Technisch zijn deze stadia op te breken in de volgende stadia:

- Te verwerken
    - Taken die nog verwerkt moeten worden en een verwerk moment in de toekomst hebben.
- In verwerking
    - Taken die momenteel in behandeling zijn
- Succesvol verwerkt
    - Taken die verwerkt zijn zonder problemen
- Gefaalde taken
    - Taken waarbij er problemen zijn opgetreden



Figuur 7-2

Figuur 7-2 toont de verschillende statussen die een taak kan hebben in de loop van de applicatie. In Zoals in figuur 7-3 is de verwerkingsstatus van een taak(IQueueJob) niet afhankelijk van de status van het object zelf maar afhankelijk van de Wachtrij(IQueue) waarin deze taak zich momenteel bevindt. De QueueScheduler treed op als “beheerder” van de wachtrijen en verzorgt het veranderen van de status door het wijzigen van de wachtrij waarin een taak zich bevindt.



Figuur 7-3

Figuur 7-3 is het klassendiagram met alle klassen, methoden en attributen om de eisen horende bij deze userstory te kunnen voldoen. Er is gebruik gemaakt van kleuring om de DDD gerelateerde packages aan te geven. De groene klasse behoort tot de service package, het is de ingang naar de rest van de klasse. De blauw gekleurde klassen horen bij de BusIness package. De roze gekleurde klassen horen bij de domain package.

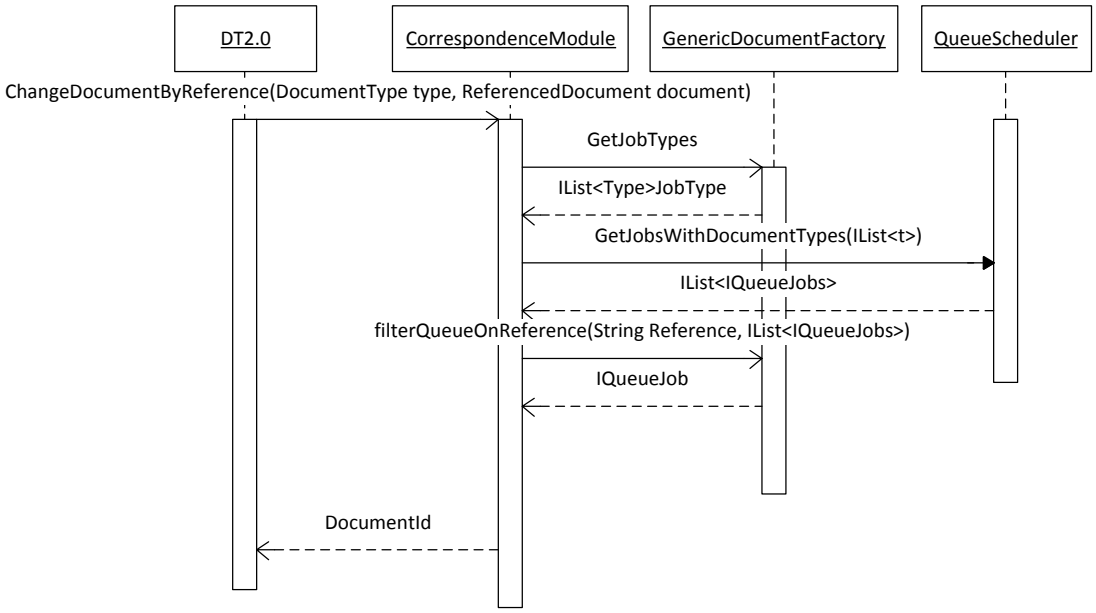
In figuur 7-3 in het Domain package bij de klasse Template staan de attributen ScheduledMoment(int) en ScheduledDay(<<enum>> ScheduledDay). Deze dienen binnen de applicatie gebruikt te worden om het standaard verwerkingsmoment van een taak die gebruik maakt van deze template vasttestellen. Het scheduledMoment kan gebruikt worden om een moment na de ScheduledDay vast te stellen waarop de taak plaats moet vinden. ScheduledDay wordt vervolgens gebruikt om de dag vast te stellen waarbij de volgende opties gebruikt kunnen worden:

- Dagelijks
    - De taak dient dagelijks uitgevoerd te worden op de hoeveelheid seconden in scheduledMoment na 00:00. Indien de taak die dag reeds uitgevoerd is(taak wordt om 16:00 toegevoegd, standaard vind afhandeling plaats om 15:00) dan wordt deze de volgende dag uitgevoerd.
- Wekelijks op dag(maandag-zondag)
    - De taak wordt wekelijks uitgevoerd op de hoeveelheid seconden in ScheduledMoment na 00:00. Indine de Taak deze week reeds uitgevoerd is dan wordt deze de week erna uitgevoerd.
- Over een tijd
    - De taak dient uitgevoerd te worden na ScheduledMoment seconden vanaf nu.

Het bij de TemplateManager opvragen van de tijd betekend dat het template door een Klasse uit de Infrastructure package (klasse is niet opgenomen in het overzicht) opgehaald wordt uit de database. Vervolgens dient er door de TemplateProcessor een verwerkingsmoment gedestilleerd te worden op basis van de twee variabelen in de Template en de huidige tijd. En deze wordt vervolgens terug gegeven en door de verwerkende instantie van de IDocumentFactory op de taak(IQueueJob) ingesteld als verwerkingsmoment(ScheduledMoment).

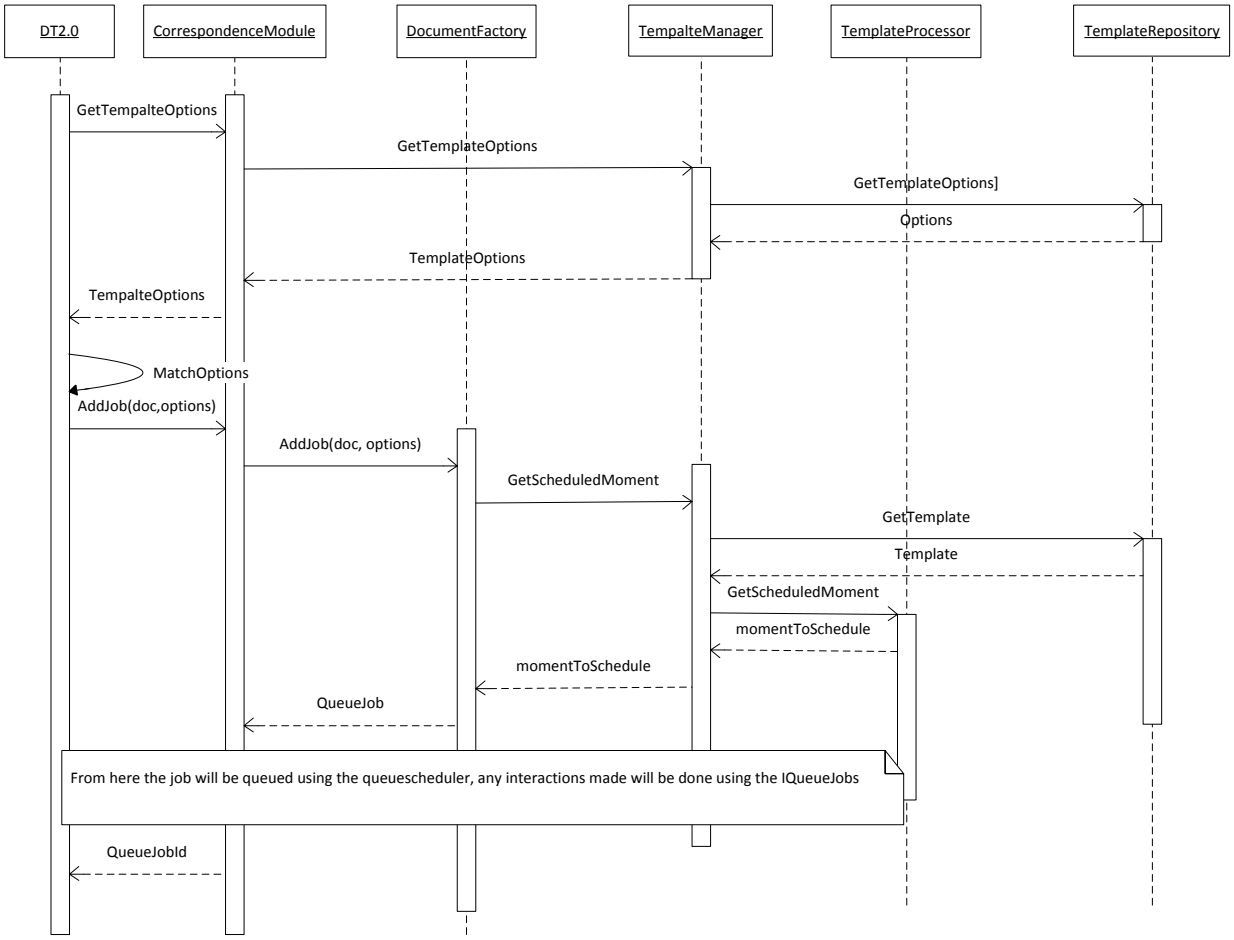
Deze IQueueJob wordt vervolgens teruggegeven aan de Service die hem door de QueueScheduler laat toevoegen aan een wachtrij. De wachtrij controleert geregeld te taken en verwerkt deze zodra deze zijn gescheduled. Ik heb hierin gekozen om de status van een taak afhankelijk te maken van de wachtrij waarin ze opgenomen is. Deze keuze is gebaseerd op de mogelijkheid van het gebruik van een lazy loading algoritme in de persistentie structuur van de data. Hierdoor kan je een wachtrij ophalen en in het geheugen laden zonder de achterliggende taakinformatie te hoeven op halen uit de database. Dit zorgt ervoor dat ik taken bij een status gegroepeerd kan ophalen en verwerken.

In diagram(figuur7-3) dat betrekking heeft op de deze userstory, staan niet opgenomen de DataTransferObjecten(DTO) die nodig zijn om aanroep van de service mogelijk te maken, de aanroep van de methoden van de service staan hier beschreven met de Domainobjecten die in de service gekoppeld zijn aan de DTO. Eveneens staan in dit overzicht Infrastructure klassen niet opgenomen omdat deze het geheel te onoverzichtelijk kunnen maken.



Figuur 7-4

In figuur 7-4 staat aangeven wat de flow van een document is als deze vervangen dient te worden. Hierbij wordt op basis van een document referentie een document vervangen. Dit biedt de mogelijkheid tot het handmatig bewerken van een document en het vervolgens overschrijven van het reeds aangemelde document.



Figuur 7-5

Dit diagram(figuur 7-5)diagram beschrijft het gebruik van een infrastructure element(TemplateRepository). De repository maakt gebruik van de ORM mapper voor communicatie naar de database. Voor het toevoegen van een taak is het nodig dat de gebruiker van der service weet welke mogelijkheden er zijn voor de keuze van communicatie kenmerken. De afnemer van de service bepaalt vervolgens welke kenmerken van toepassing zijn op deze taak. Hierna kan de template opgehaald worden, de tijd gegenereerd en vervolgens kan deze taak toegevoegd worden aan de wachtrij.

7.4. Bouw

In eerste instantie heb ik het skelet voor de applicatie moeten opzetten. Hiervoor heb ik dus een specifiek Domain, Infrastructure, business en Service package aangemaakt.

Uiteindelijk heb ik de klassen vanuit het TO aangemaakt en de methoden geïmplementeerd. Hierna heb ik gekeken naar de WCF service en de diverse objecten die nodig zijn bij afhandeling van de requesten en heb ik een basis versie gemaakt van de IDocuementFactory zodat ik op z'n minst Jobs kon toevoegen aan de QueueScheduler.

Een van de eerste problemen na het opzetten van de structuur en het in het geheugen kunnen opslaan van taken in een wachtrij structuur was de duur van de uiteindelijke verwerking. Ervaring die ik tijdens de bouw van de Proof of concept op had gedaan met het verwerken van documenten die afgedrukt dienen te worden betekende dat ik een redelijke inschatting kon maken van de duur van de verwerking van een taak. Dit betekend dat ik gekeken heb naar een aantal verschillende mogelijkheden om eventuele verwerking op een zo veilig mogelijke manier te kunnen laten verlopen zonder dat de beschikbaarheid van de applicatie er onder zou kunnen komen te leiden.

In andere software projecten had ik reeds ervaring opgedaan met alternatieve verwerkingsmethoden. De mogelijke oplossing staan hier benoemt, ik ga hierna dieper in op de werking van de oplossingen

- Een single thread oplossing
- Een multithread oplossing
- Multithreaded fork-join oplossing

Er is een oplopende complexiteit in de genoemde potentiele oplossingen.

De naam voor de singlethreaded oplossing betekend niet dat de executie in dezelfde thread plaatsvind als de rest van de applicatie. Het betekend dat na het verstrijken van de delay, en sub sequentiële ticks van de timer er een nieuwe thread uit de threadpool gehaald wordt waar in de methode die uitgevoerd moet worden uitgevoerd wordt. Dit betekend dat de controle op de taken in de wachtrij plaatsvindt in een singlethread. Potentieel zou het zo kunnen zijn dat er een extra thread gecreëerd wordt tijdens de controle van de huidige thread. In deze oplossing zou er op basis van een wachtrij een database lock aangevraagd moeten worden. Dit kan bij verwerkingsduur boven de verwerkingsinterval leiden tot deadlock situaties.

De tweede oplossing maakt gebruik van dezelfde methode omtrent aanroep van de methode, echter in de methode zelf wordt er eerste de taken opgehaald die verwerkt moeten worden, vervolgens worden de taken in uitvoering gezet en de transactie afgesloten, vervolgens worden deze taken verwerkt in een aparte thread. Waarna de methode doorgaat naar de volgende controle, zo wordt er voor elke queue in de eerste thread de taken overgezet waardoor de wachtrijen het kortste transactioneel niet beschikbaar zijn. De verwerking verwerkt vervolgens de taken een voor een en voegt bij elk van de taken die verwerkt zijn het verwerkingsmoment. In totaal worden er meerdere threads gestart uit de threadpool, die na executie allemaal eindigen. Per thread waarin de taken opgenomen zijn wordt er een lock op een specifieke hoeveelheid taken verkregen en dus geen lock op de gehele wachtrij, dit betekend dat de wachtrijen gedurende het ophalen van de taken die verwerkt is gelockt is waarna de object die verwerkt worden naar een andere queue gezet worden en

dan op taak niveau gelocked worden voor verdere verwerking. De opbouw van het verkrijgen van locks leidt uiteindelijk tot het verminderen van de impact van dirty reads(eventuele verwerkte taken die in verwerking staan worden een later moment alsnog naar verwerkt gezet), en het verminderen van de kans op deadlocks.

De Fork-join oplossing is gebaseerd op het parallel kunnen verwerken en vervolgens na uitvoer terug te koppelen naar een enkele thread. Het is vooral toepasbaar in een omgeving waarbij centrale resourcse parallel verwerkt(berekeningen) kunnen worden. De toepasbaarheid in bovenop een database met transactionele beperkingen kan leiden tot deadlock situaties waarbij twee threads simultaan wachten op het vrijkomen van een andere transactionele databaselock. De potentiele problemen die kunnen ontstaan wegen niet op tegen de voordelen van het parallelle gebruik van de resources.

Uiteindelijke heb ik gekozen voor de enige oplossing die de database het kortste belast met locks en die de verwerking zoveel mogelijk naar threads waarin de prioriteit lager gezet kan worden dirigeert.

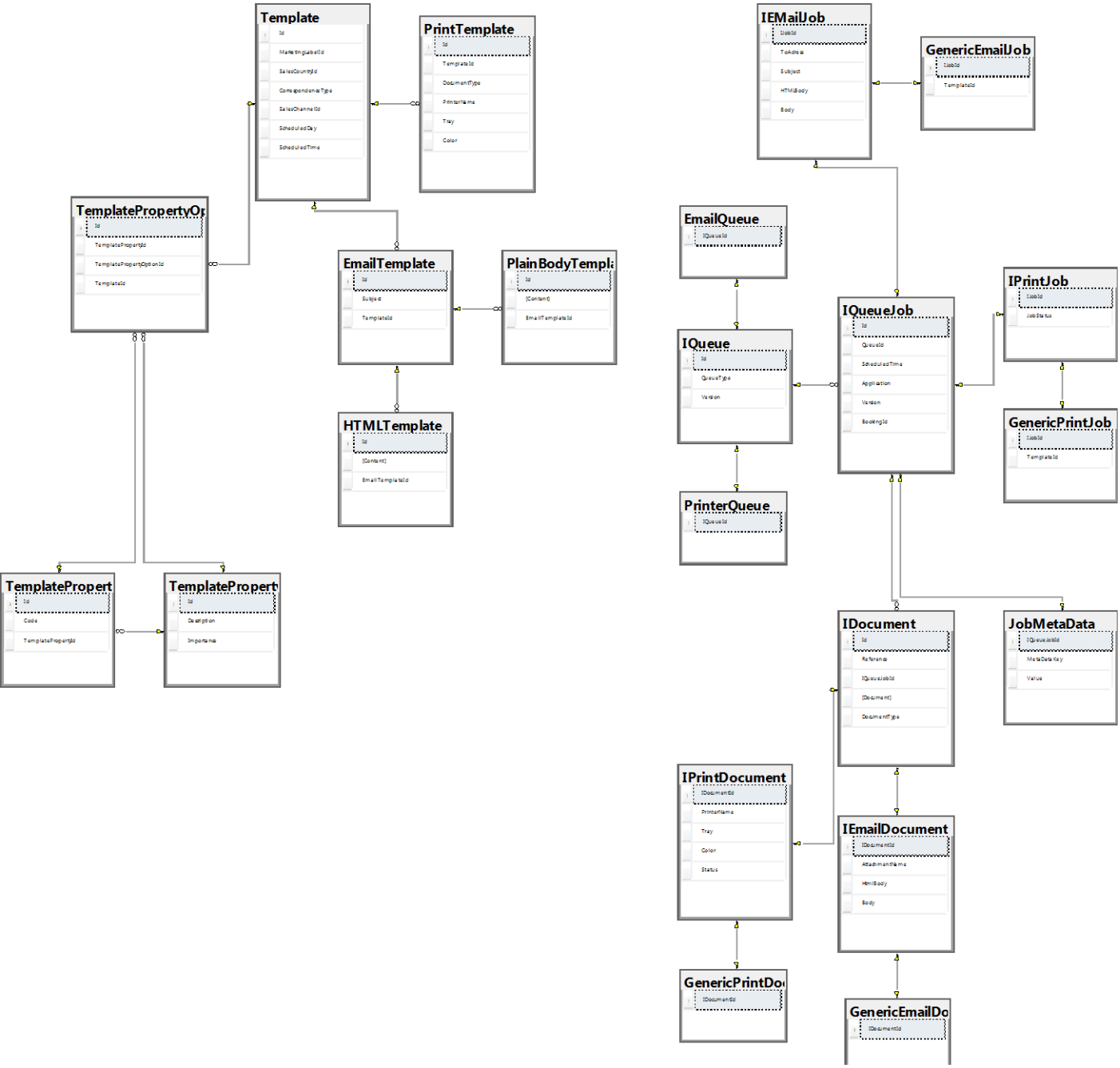
De periode waarin de controle uitgevoerd wordt heb ik vervolgens instelbaar via de configuratie files gemaakt.

Na dat het algemene queuemechanisme en de verwerking ervan in de userstory print/selectie uitgevoerd was heb ik de persistentie laag gebouwd. Hierdoor wordt bij eventueel uitvallen van het systeem de taken in de wachtrij gepersisteerd en kan de applicatie na herstart verder waar het gebleven was. De mogelijkheden voor persistentie van de domein objecten op transactionele basis zijn vrij beperkt, gebruik makend van de kennis die reeds binnen het ontwikkeltraject zit heb ik gekozen voor NHibernate voor persistentie. Geen enkele andere tool voldoet aan de inherente eisen die ik gesteld heb voor persistentie(transactioneel opgebouwd, SQL injection safe, omgang met blobs en gebruikmakend van Microsoft SQL server). De enige optie die mogelijk een oplossing had kunnen bieden was het gebruik van Stored Procedures vanwege de snelheid, maar de afhankelijkheid van bedrijfslogica in de database wordt daarmee te groot.

Uiteindelijk heb ik dus gekozen voor de Object Relation Mapper(ORM) die binnen het software traject reeds in gebruik is. Een ander voordeel hiervan is dat de code overdraagbaarheid hierdoor is vergroot. Ik heb hiervoor eveneens gebruik gemaakt van de zelfde methode die binnen het DirectTravel 2.0 systeem gebruikt was om de Object mappings op te zetten. Hiervoor heb ik mijn objecten beschreven in een XML structuur. Voor het maken van de interface structuur van de basis taak en document generatie heb ik gebruik gemaakt van aparte tabellen. Een andere mogelijk was het gebruik van een discriminator, een aparte veld bij een tabel die aangeeft om welke daadwerkelijke klasse het gaat. De toepassing met aparte tabellen heeft weliswaar meer overhead maar is wel beter in de uitbreidbaarheid in de toekomst, ik heb om die reden dan ook gekozen voor extra tabellen.

Voor de database heb ik een aparte database opgezet voor de correspondentie module en de software(nHibernate) de mappings vervolgens in een aparte database tabellen later genereren, dit betekent dat zodra ik aanpassingen doe in het domeinmodel en ik deze wijzigingen meeneem in de mappings de database tabellen handmatig aangepast moeten worden om de situatie op elkaar aangesloten te houden, mogelijk moet dan alle data verschoont worden.

Het model(figuur 7-6) is een overzicht van de opbouw van de data. Dit model is de database structuur na ontwikkeling van de gehele applicatie en is grotendeels een continuering van de klassen en relaties zoals in het DomainPackage. Een duidelijke splitsing in de twee stukken van het domein is terug te vinden(template en wachtrijen)



Figuur 7-6

7.5.Tests

Selectie van tests

Vanwege de samenhang tussen de verschillende userstories en de samenhang van testbare onderdelen heb ik testgemaakt voor deze userstory die in latere userstories uitgebreid zijn om de eisen van de userstory te koppelen aan de reeds bestaande tests.

De verschillende testtypes die in eerste instantie gekozen zijn om te kunnen garanderen dat wat opgeleverd gaat worden voldoet aan de functionele eisen:

- Geautomatiseerde tests
    - Unittests die de Create Read Update Delete van domein entiteiten testen(whitebox)
        - Gebruik makend van de bedrijfslogica
    - Unittests die de functionaliteit van de service testen(blackbox)
        - Gebruik makend van de WCF service
- Handmatige tests
    - Testen of de webUI functioneert naar behoren
        - Juiste invoer
        - Foutieve invoer
        - Functionele elementen
        - Aanmaken, inzien, wijzigen en verwijderen

Naast de functionele tests dient er ook nog een set tests uitgevoerd te worden om de maximale belasting bij gelijktijdige verwerking te achterhalen alsmede tests die de performance in gesimuleerde situaties in kaart probeert te brengen.

- Geautomatiseerde tests
    - Een automatische test die bestanden van 10 pagina's klaar zet voor verwerking waarbij de helft van de bestanden via de email verstuurd dient te worden en de ander helft via de printer verwerkt moet worden.
        - Verwerking van de e-mails gebeurt via een lokaal benaderbare exchange server.
        - Verwerking van printer opdrachten gebeurd afwisselend via een software printer(PDFCreator) en een fysieke printer.
    - Een automatische test die bestanden toevoegt aan de service terwijl er gelijktijdig verwerking van bestanden plaatsvind, bestanden die toegevoegd worden hebben een totale grote van 5 pagina's en worden toegevoegd in het tempo van eens per 5 seconden verwerking vind plaats in batches van 60 documenten via de email en 60 documenten via de printer.

De keuze voor deze tests komt vooral door het beschikbaar zijn van een testteam die de functionele testen voor het grootste gedeelte uitvoeren. De keuze om perforamance en stress tests uit te voeren komt voort uit de wens een uitspraak te kunnen doen over belastingen op de verschillende onderdelen en te kunnen voldoen aan de verwachtte piekbelasting voor het gebruik van de correspondentiemodule, daarnaast is het nodig om wat te kunnen zeggen over de maximale verwerkingstijd van taken in de wachtrij.

Uitvoer van tests

Stress test

De stress test heeft tot doel het ontdekken van bottlenecks en de maximale verwerkingssnelheid die deze bottlenecks geven.

Potentiele bottlenecks

Netwerk

Het eerste potentiele bottleneck is de verwerkingsnelheid van het netwerk, aangenomen dat alle onderdelen in het netwerk aangesloten zijn op een gigabit netwerkmodule zonder parallelle verwerking en van deze verbinding gebruik maakt voor het downloaden van de documenten uit de database, het verwerken van documenten naar de printers/exchange server. De Maximale verwerkingssnelheid van een gigabit netwerkmodule is in theorie 125mb/s. In de praktijk is deze snelheid eerder lager en kunnen we beter uitgaan van een reële situatie waarin de helft van deze snelheid gehaald kan worden(60mb/s). De verwerkingsgrote van mijn document is 1.5mb maar het document wordt dubbel over de lijn verwerkt, een keer van de database naar de server en een keer van de server naar de verwerkende instantie. In totaal zou het dus mogelijk moeten zijn om 20 documenten per seconde via het netwerk te laten verwerken.

Printer verwerking

In het proof of concept heb ik al een vooropzet gemaakt van de te verwachtten verwerkingsduur van het printen van PDF objecten , op basis hiervan kan ik een extrapolatie maken naar de verwachtte maximale hoeveelheid te verwerken documenten. Een groot document van 10 pagina's voor de printer verwerken duurt ongeveer 2 seconden. Gedurende de verwerkingsinterval die ingesteld staat op 5 minuten zou het mogelijk moeten zijn om 30x5 = 150 documenten voor een totaal van 1500 pagina's af te drukken. Een simulatie voor 5 minuten zou daarnaast eveneens gebruik moeten maken van meerdere printers om deze hoeveelheid af te kunnen drukken. Een bovengemiddelde printer is in staat 60 pagina's per minuut af te drukken, 300 pagina's in de test periode. De simulatie gaat uit van vier printers(de hoeveelheid printers waarbij bij de klant afgedrukt kan worden). Hierdoor is de maximale hoeveelheid af te drukken documenten volgens de verwachting niet gelimiteerd door de verwerkingssnelheid van documenten in de wachtrij maar door de verwerkingssnelheid van de printers. De hoeveelheid documenten die op basis van deze gegevens als bovengrens genomen kan worden is dus dan ook 120 documenten ofwel 1200 pagina's.

Email verwerking

Ik verwacht geen problemen met de verwerking van e-mails. De snelheid van verwerking is gelimiteerd op de netwerk bottleneck, en die zou instaat moeten zijn om voldoende netwerksnelheid te hebben voor verwerking.

Schrijven van de test

De test is zo geschreven dat deze automatisch uitvoerbaar is, er wordt dan ook gebruik gemaakt van de test package(whitebox performance test) waarin de test beschreven is. Deze package maakt het mogelijk direct de business logica aan te roepen van de applicatie. Bij het aanmaken wordt er gebruik gemaakt van een loop die een aantal taken toevoegt aan wachtrijen. Vervolgens worden de taken opgehaald die verwerkt moeten worden en wordt in de eerste thread de verwerkingsmethode

aangeroepen(hierdoor wordt er geen gebruik gemaakt van de multithreading en kan er een uitspraak gedaan worden over de verwerkingstijd).

De test run

De test wordt na het runnen uitgevoerd met steeds hoger wordende waarden voor de hoeveelheid taken die toegevoegd worden totdat de totale executie op 5 minuten komt.

Gebruikmakend van de aanname die ik al had gemaakt ben ik begonnen met 90 om 90 documenten en heb ik dit vervolgens uitgebouwd naar 110 om 110 documenten waarna ik problemen begon te krijgen met de printers. Ik heb vervolgens nog een run gedaan op 100 te printen documenten om 150 te e-mailen documenten.

Conclusie

Uiteindelijk vormt de printer nog de bottleneck maar het verspreiden van de load over meerdere printers zal snel leiden tot de volgende theoretische bottleneck. Daarnaast loopt de applicatie gewoon door buiten deze 5 minuten met het verwerken van taken. Indien er daadwerkelijk meer performance vereist gaat worden is het volgende bottleneck de interne verwerkingssnelheid. Een oplossing die daarvoor geboden moet worden is het splitsen van email en print opdrachten over twee systemen.

Performance test

De performance test heeft als doel aantonen dat de applicatie blijft functioneren onder een oplopende load op de backend van het systeem. Hiermee dient aangetoond te worden dat het systeem responsief blijft en de applicatie die gebruik maakt van de module niet vertraagd doordat de module druk bezig is met verwerking.

Verwachte uitvoer

De te verwachten uitvoer voor deze test is dat er wel een verschil ontstaat in reactiesnelheid indien het systeem idle is tegenover wanneer het systeem bezig is met taak verwerking. Dit verlies in performance zal leiden tot vertraging bij het aanmaken van de taak van 0% tot 15% langere verwerkingstijd.

De prioriteit van de executie thread is normaal, de prioriteit van de verwerkingsthread is Below normaal. Dit betekend dat zodra de taken opgehaald worden en klaargezet voor verwerking de prioriteit normaal is en afhankelijk van beschikbare processor capaciteit er vertraging optreed. Indien de taak bezig is met het verwerken in de lagere prioriteit thread het performance verlies minimaal is. De duur van verwerking gaat wel omhoog zodra er meer taken toegevoegd worden.

Uitvoer van de test

De test is zo opgezet dat er simultaan een load gezet wordt op de voorkant bij het invoeren als aan de achterkant het laten lopen van de verwerking. Hiervoor wordt er gebruik gemaakt van de WCF service om een flink aantal jobs te laten genereren met een gelijktijdig verzendmoment. Vervolgens zal op het verzendmoment de test weer gestart worden om taken aan te maken.

De twee tijden die benodigd zijn om de taken aan de wachtrij toe te voegen via de service worden vergeleken, en op basis hiervan kan ik een uitspraak doen over de performance onder load tegenover de statische performance.

De test run

Uiteindelijk heb ik de test gedraaid met het verwerkingsmoment van de template op 14:00 uur. Vervolgens heb ik een aantal jobs klaar gezet voor verwerking, om de druk zo goed mogelijk na te kunnen bootsen heb ik voor verwerking een test gedaan met verwerking naar een printer en met verwerking naar een email. Uiteindelijk voeg ik voor de verwerking 60 taken van 1 document toe aan de emailwachtrij en 60 taken van 1 document aan de printwachtrij. Vanuit de performance test bestaat de verwachting dat ik hierdoor een verwerkingstijd heb van ongeveer 2 minuten. Dit betekend dat ik 2 minuten onder dubbele load zou kunnen testen.

Het verwerkingsmoment eerste verwerkingsmoment is 15 seconden na het starten van de service. Om dit te kunnen testen heb ik vervolgens dus een wait ingebouwd van 15 seconden waarna hij opnieuw gaat toevoegen aan de wachtrij. Hierna zet ik de service af tot na het verwerkingsmoment en worden de taken verwerkt op het zelfde moment dat de verwerking gaat lopen dus is er zeker dat de twee tegelijk bezig zijn.

Conclusie

De eerste run, het toevoegen zonder gelijktijdige automatische verwerking was sneller dan de tweede run. De eerste run voegde de taken binnen 45 seconden toe aan de wachtrij(120 taken) de tweede run deed hier langer over 65 seconden. Een tweede run waarbij ze beide wel dezelfde code gebruikten(de 15 seconden wait) was de eerste run in 62 seconden klaar en was de tweede run 66 seconden klaar. Onder aan de streep is de verwachtte vertraging minder dan verwacht. Een derde run waarbij de wait opgeschroefd is naar 20 seconden(om de gelijktijdige lock op de wachtrijen bij het verwerken en het toevoegen van de eerste taken uit te sluiten) zakte het verschil naar 2 seconden. Hiermee blijft performance onder load goed te noemen.

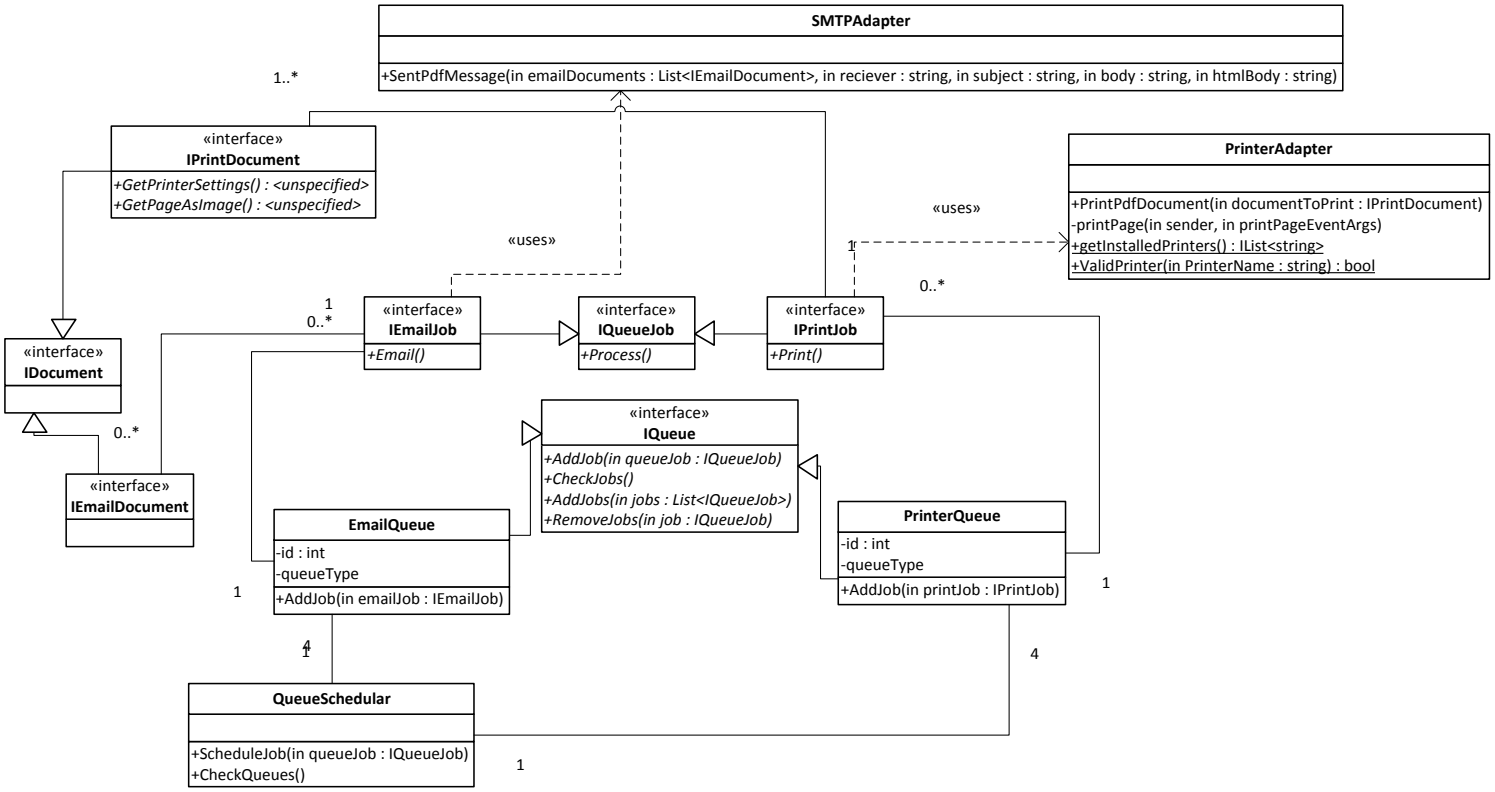
8. Userstory 2: Selectie printen/e-mailen

8.1. Functionele eisen

De correspondentie module voorziet inde mogelijkheden om analoog(via de post) en digitale(via de email) correspondentie af te handelen. Er wordt voor DT2.0 de mogelijkheid gecreëerd om correspondentie te verwerken op basis van communicatietypes.

- De combinatie communicatietype (reisbescheiden etc.) marketinglabel en verkoopland zijn leidend bij de selectie voor print of email.
    - De standaard afhandeling dient ingesteld te kunnen worden(user story standaard opmaak).
- Het verwerken van digitale items gebeurd door het versturen van de documenten via de email
    - Hierbij dient rekening gehouden te worden met de maximale verstuurgrote van bijlagen bij emailberichten
    - De tekstuele inhoud van het emailbericht wordt opgesteld op basis van een template(zie standaard opmaak communicatie)
- Voor een specifieke boeking moet het mogelijk zijn om de standaard methode van versturen te overschrijven (alle communicatie wordt digitaal verstuurd maar klant wil alles analoog hebben)
- Voor het afhandelen van analoge communicatie kan er per documenttype in het communicatietype een selectie van printerinstellingen opgegeven worden bijvoorbeeld:
    - factuur die verstuurd dient te worden via de post met een begeleidende brief(printer 1 A4 lade 1), factuur pagina met aangehechte acceptgiro(A4 met acceptgiro aanhechting(printer 3 Lade 2)
        - Er dient dan specifiek voor een documenttype, marketinglabel en verkoopland een printtemplate voor het afdrukken aangemaakt waarin de documenten gespecificeerd worden

8.2. Globaal ontwerp

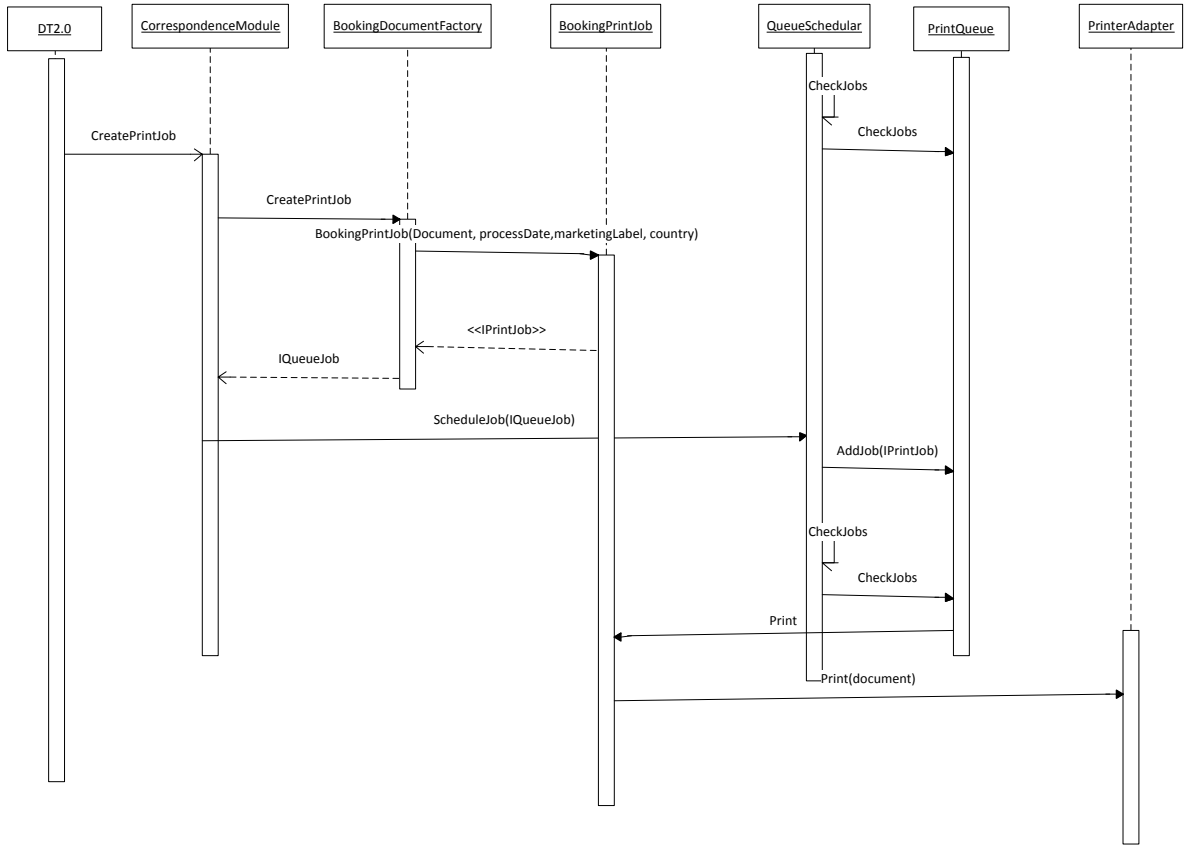


Figuur 8-1

Voor het functioneel ontwerp van deze userstory is het functioneel klasse diagram uitgebreid met functies die het mogelijk moeten maken om op het verwerkingsmoment de taken te gaan verwerken. De adapters voor de afhandeling van het printen/e-mailen hebben een specifieke methode die door de verwerkende taak aangeroepen dient te worden. De QueueSchedulerdient in “ticks” de CheckQueues uit te voeren. De kennis die ik reeds opgedaan bij het proof of concept voor het verwerken van printer items heb ik gebruikt om functioneel de PrinterAdapter te kunnen omschrijven.

8.3. Technisch ontwerp

Door de gecombineerde ontwikkeling van de userstories verstuur queue en selectie printen/e-mailen en de reeds opgedane kennis van de Proof of concept is er binnen het klasse diagram ten opzichte van het functionele klasse diagram voor de twee objecten die deze userstory nog betrekking op heeft geen wijziging te vinden.



Figuur 8-2

Het sequentiediagram is toepasbaar op zowel het printen alsmede het emailen van een taak, met de enige wijziging dat het laatste object in plaats van de PrinterAdapter de SMTPAdapter is. En de methode andere genoemd is. Belangrijk hierin is het losstaande toevoegen en uitvoeren van taken, de taak wordt toegevoegd en dan pas opgepakt zodra de tijd waarop de taak staat om uitgevoerd te worden verstreken is.

De selectie van de printerinstellingen indien het een taak is die geprint dient te worden is niet opgenomen in dit sequentiediagram. De TemplateManager kan op basis van diverse mogelijke aangeboden types van template opties een template selecteren voor de af te handelen tijd. Deze template heeft een relatie met een set PrintTemplates die te identificeren valt op basis van een documentType.

8.4. Bouw

De Proof of concept zoals ik reeds al had gemaakt heb ik niet gebruikt, de proof of concept was om aan te tonen dat het concept mogelijk was en welke conversie bibliotheek ik het beste kan gebruiken.

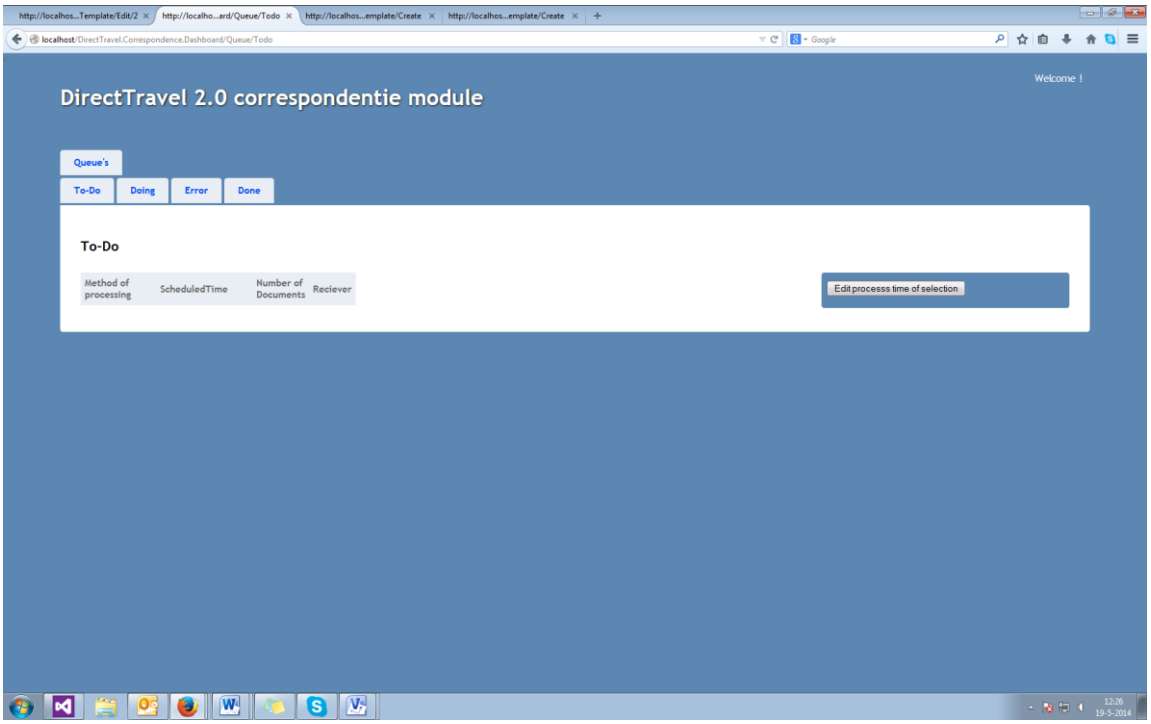
De bouw van deze userstory zit hem voor het grootste gedeelte in het ontwikkelen van een Object georiënteerde oplossing voor het printen met behulp van de Aspose bibliotheek. Omdat de eerste twee userstories in dezelfde sprint zitten heb ik deze ook gelijktijdig opgepakt. Hierdoor kon ik direct de taken de verwerkt moesten gaan worden gebruiken om te verwerken zodat ik ook een demonstreerbare applicatie heb.

In de bouw van deze userstory neem ik echter de bouw van de User Interface op, dit is een specifieke eis dat pas later tijdens sprint twee daadwerkelijk genoemd is. Hiervoor heb ik de service die tot nu toe puur was voor het toevoegen van taken uitgebreid met mogelijkheden voor status controle van taken. Om de wachtrijen en templates inzichtelijk te maken heb ik de volgende opties overwogen:

- Een set schermen binnen het overkoepelende project(winforms oplossing)
- Een set schermen die los staat van het huidige project en eventueel geïmporteerd kan worden(winforms oplossing)
- Een web interface (Asp.net MVC oplossing)

Uiteindelijk is de keuze niet gevallen op een winforms oplossing omdat dit het toekomstig hergebruik zou kunnen beperken. Indien de Correspondentie module door andere projecten hergebruikt zou worden dient het mogelijk om zonder afhankelijkheid van DirectTravel gebruikt te worden. Een winforms oplossing maakt het of direct afhankelijk van DT2.0 of het maakt een extra winforms applicatie voor weergave van de wachtrijen een eis. Het gebruik van een web gebaseerde oplossing biedt de mogelijkheid tot integratie door een Browser user control in de reeds bestaande applicatie. Eveneens is het gebruik van een web oplossing een oplossing die beter hergebruikt zou kunnen worden in andere projecten.

De keuze voor ASP.Net in combinatie met het MVC framework heeft uiteindelijk geleid tot het ontwikkelen van een dashboard applicatie die gebruik maakt van de WCF service van de Correspondentie Module voor de data. De dashboard oplossing maakt gebruik van diverse HTML5 en JQuery oplossingen om het responsief te laten verlopen. Het biedt inzicht in de taken die uitgevoerd moeten worden, die in uitvoering zijn, die succesvol afgerond zijn en niet succesvol afgerond zijn.



Figuur 8-3

Functioneel was in eerste instantie niet opgenomen dat er controle plaatsvond op taken die in verwerking zijn. Deze taken werden vanuit verwerking automatisch op verwerkt gezet. In een later stadium is hier controle op de uitvoer bijgekomen, deze controle zorgt ervoor dat taken als verwerkt of als foutief aangemerkt kunnen worden. De controle betekend het controleren op fouten die tijdens het verwerken kunnen ontstaan, je kan hierbij denken aan verbindingsproblemen met de exchange server, printer fouten of pagina fouten.

Het belangrijkste element hiervan is de controle op het printen, het printproces kan gecontroleerd worden door gebruik van een in het .net framework ingebouwde printerstatus package. In deze package kan je door de printer op te zoeken een controle uitvoeren op de taken. Deze controle is beperkt tot het controleren van de op de volgende zaken:

- Fouten tijdens het printen
    - Verstopping van het papier
    - Niet kunnen printen door inkt problemen
- Printer problemen die ontstaan zijn waarvan het ontstaan niet aan een bepaalde printopdracht gekoppeld kan worden
    - Printers die uitgezet zijn/uit het printernetwerk gehaald zijn

Eventuele andere problemen kunnen niet door deze detectie vastgesteld worden. Dit is helaas een beperking van deze controle.

Hiervoor heb ik uiteindelijk een extensie binnen de printeradapter geschreven die op basis van document naam en geselecteerde printer kan controleren of de controleerbare gefaalde condities al dan niet zijn opgetreden.

8.5. Tests

Bij de vorige userstory heb ik de stress en performance testen omtrent het verwerken van taken in de wachtrij beschreven, deze testen gaan hand in hand met de performance die in deze userstory vereist wordt in het daadwerkelijk verwerken van deze taken. De tests bij de vorige userstory zijn dan ook gekoppeld aan zowel het testen van de verwerking en dus ook de multithreading alsmede de daadwerkelijke verwerking zoals die hoort bij deze userstory.

De tests die exclusief bij deze userstory horen zijn de tests die betrekking hebben op het gebruik van de web interface. Het gaat hier om de syntactische en semantische tests omtrent het opnieuw toevoegen van taken.

Syntactische tests

Het wijzigen van de verwerkdatum van taken in de te verwerken status. Bij het wijzigen van de verzend datum dient de datum en tijd gecontroleerd te worden op validiteit.

Semantische test

Het wijzigen van de verwerkingsdatum en tijd van een taak dient de daadwerkelijke verwerking van deze taak ook op dit tijdstip plaatst te vinden.

Acceptatie testen

Printen

Het printen dient gebruik te maken van de ingestelde printer en lade. Eveneens dient het print resultaat gecontroleerd te worden door de opdrachtgever of het van voldoende kwaliteit is. Het specificeren van een kwaliteitseis gebeurt door de pdf via adobe af te drukken en vervolgens via de correspondentie module af te drukken en te zoeken naar visuele afwijkingen. Hiervoor wordt gebruik gemaakt van een printertestpagina, dit zijn afdrukken die gebruikt kunnen worden om de kleur saturatie vast te kunnen stellen, het verloop van lijnen en de afdruk kwaliteit op kleine tekens. Aan de hand van een printertestpagina kan er een uitspraak worden gedaan over de kwaliteit van het geprinte object.

E-mailen

Het vaststellen van acceptatiecriteria van de e-mails is simpeler dan printopdrachten. Bij het e-mailen is er geen kwaliteitsverlies. De enigste eis die er aan een e-mail gesteld kan worden is dat de bijlage zelf geopend kan worden, dat bij meerdere documenten in een taak elk van de bijlagen toegevoegd is aan de e-mail en dat het ontvangen wordt door de ontvanger.

9. Userstory 3: Standaard opmaak communicatie

9.1. Functionele eisen

Alle communicatie is gekoppeld aan een standaard opmaak. De standaard opmaak is per product groep in te stellen. De standaard opmaak is te beheren via tekstblokken. In de tekstblokken wordt door middel van een code aangegeven waar dynamische gegevens worden ingevuld. *Bijvoorbeeld de naam van de reis moet getoond worden in een template. Hier kan de code {Reis} worden neergezet. Deze zal bij het genereren van de communicatie worden vervangen door de naam van de reis waar de communicatie over gaat.*

De gegevens die onder andere dynamisch ingevuld kunnen worden zijn gegevens die in DT zijn opgevoerd:

- Klant gegevens:
    - Aanhef
    - Naam
    - Adres
    - Etc.
- Boekingsgegevens:
    - Geboekte reis
    - Vertrekdatum
    - Dossier nummer
    - Vluchtschema
    - Etc.

Voor alle communicatie die verstuurd wordt vanuit de correspondentie module kan een template worden aangemaakt en beheerd. Het is mogelijk om de gegenereerde communicatie die niet direct wordt verstuurd aan te passen voor versturen.

Het beheren (aanpassen) van de templates zal gebeuren in tekstblokken.

Alle digitale communicatie vanuit DT wordt voorzien van een HTML emailbody en een platte tekst(voor specifieke emailontvangers) dit wordt gegenereerd aan de hand van contentblokken.

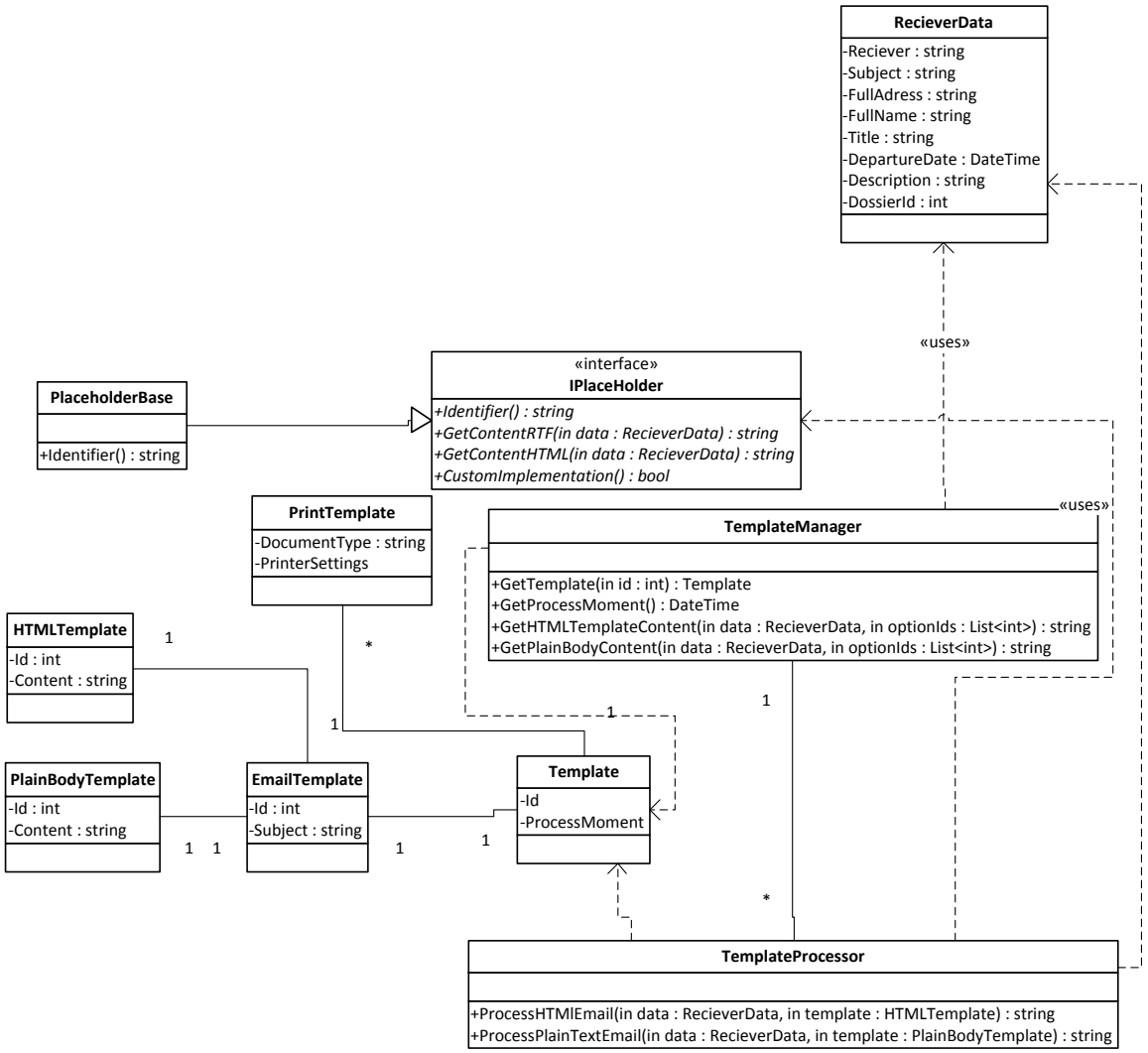
De analoge Communicatie maakt ook gebruik van instellingen om specifieke printerinstellingen voor documenttype(b.v. factuurpagina) binnen communicatietype (factuur)

Bedrijfsregels voor standaard opmaak communicatie

Een template wordt gekoppeld aan één communicatie type, één marketing label, één verkoopland, één verkoopkanaal en de specifieke communicatie.
Bijvoorbeeld voor de e-mail die verstuurd wordt wanneer de reisbescheiden voor product groep familie reizen beschikbaar zijn in het online dossier zal een specifieke template beschikbaar zijn. Voor dezelfde e-mail bij een singlereis zal een andere template gebruikt worden voor deze e-mail.

Wijzigingen aan een template worden niet aangepast op gereedstaande communicatie in de verzend queue. Bij nieuwe communicatie die gebruikt maakt van de aangepaste template zal de wijziging worden toegepast.
Wanneer gereedstaande communicatie wordt aangepast moet het mogelijk zijn de aanpassingen op de template op te slaan zodat deze voor alle nieuwe gegenereerde communicatie gebruikt wordt.

9.2. Globaal ontwerp



Figuur 9-1

De template processor dient de mogelijkheid te bieden de placeholders die in de html en plainbody templates opgenomen zijn eruit te filteren en te vervangen voor de data uit het data object(recieverData). De implementaties van de IDocumentFactories vroegen al aan de TemplateManager de verzendmomenten waarop deze taak afgehandeld dient te worden, die implementaties moeten al desgewenst de TemplateManager nu vragen om de twee delen van de multipart emailbody samen te stellen.

9.3. Technisch ontwerp

Om deze userstory in te kunnen passen in het reeds gebouwde en daarnaast onderhoudbaar te houden heb ik gekeken naar de mogelijkheden om een deel van de business logica anders aan te pakken. Een belangrijk element daarvan is het mogelijke gebruik van templates voor het inpassen van teksten, deze zouden onderhouden kunnen worden en gekoppeld kunnen worden op basis van exacte matches(zoals reeds gebouwd voor de tijdsafhandeling) of op basis van een zo goed mogelijk overeenkomende template. Daarnaast heb ik in deze userstory het beheer gedeelte van de templates ontworpen. Het beheer gedeelte van de templates, een onderdeel van het dashboard, dient de mogelijkheid te hebben de templates aan te maken in te zien en aan te passen.

9.4. Bouw

Voor deze userstory heb ik de afhandeling van templates uitgebreid met mogelijkheden om vervangende teksten te laten vervangen door aan de taak meegegeven placeholders. Eveneens heb ik voor deze userstory de reeds gebouwde wachtrij interface uitgebreid met de mogelijkheid om templates te beheren.

Voor de uitbereiding van de template structuur zodat deze bij taken op basis van de meegegeven property(kenmerk) waardes moest er eerst een aanpassing komen aan het zoeken naar de templates. Het moet namelijk zo zijn dat indien er geen exacte match mogelijk zou zijn er een best passend alternatief geselecteerd wordt. De strategie om te zoeken naar een best passende alternatief dient dus te weten wat de waarde van elke kenmerk is en vervolgens te zoeken naar een template die het best past bij het meest zwaar wegende template net zo lang totdat de template het beste past bij de meest zwaar wegende kenmerken. hiervoor heb ik een algoritme geschreven wat de kenmerk waardes eerst ordent op basis van deze weging. Vervolgens wordt er voor het zwaarst wegende kenmerk de templates erbij gezocht die dit kenmerk met deze waarde heeft. Daarna wordt voor elk van de opvolgende minder zwaar wegende kenmerken de templates er bij gezocht die in de collectie van huidige templates zitten. Hierna zijn er meerdere opties:

- Een template over na controle van een kenmerk waarde
- Geen templates over na controle van een kenmerk waarde
- Alle kenmerken zijn gecontroleerd meer dan een template over
- Alle kenmerken gecontroleerd een template over

Indien er nog maar een resultaat terug komt uit de selectie wordt er gecontroleerd of de template exact de hoeveelheid kenmerken heeft als er tot nu toe gecontroleerd zijn(het is een exacte match tot nu toe), dit template wordt terug gegeven. De template heeft meer kenmerken als er tot nu toe gecontroleerd zijn, er wordt gecontroleerd of die kenmerken een match zijn met de kenmerken die gecontroleerd moeten worden. Indien een exacte match wordt dit template teruggegeven, indien geen exacte match, de selectie van voor selectie van het laatste kenmerk wordt er bij gehaald er wordt gekeken of er een template is met de hoeveelheid tot nu toe gecontroleerde kenmerken - 1(template is afgevallen om de we meer gespecialiseerd zijn gaan zoeken) indien dat het geval is wordt die template teruggegeven, indien hieruit geen template volgt recursieve aanroep waarbij de twee minst zware kenmerk waardes weg gedaan worden en de functie opnieuw uitgevoerd wordt.

Indien er geen templates gevonden worden gebeurt zoals in bovenstaande dat er gekeken wordt naar het vorige gecontroleerde kenmerk, hieruit wordt gekeken of er een match is met een template

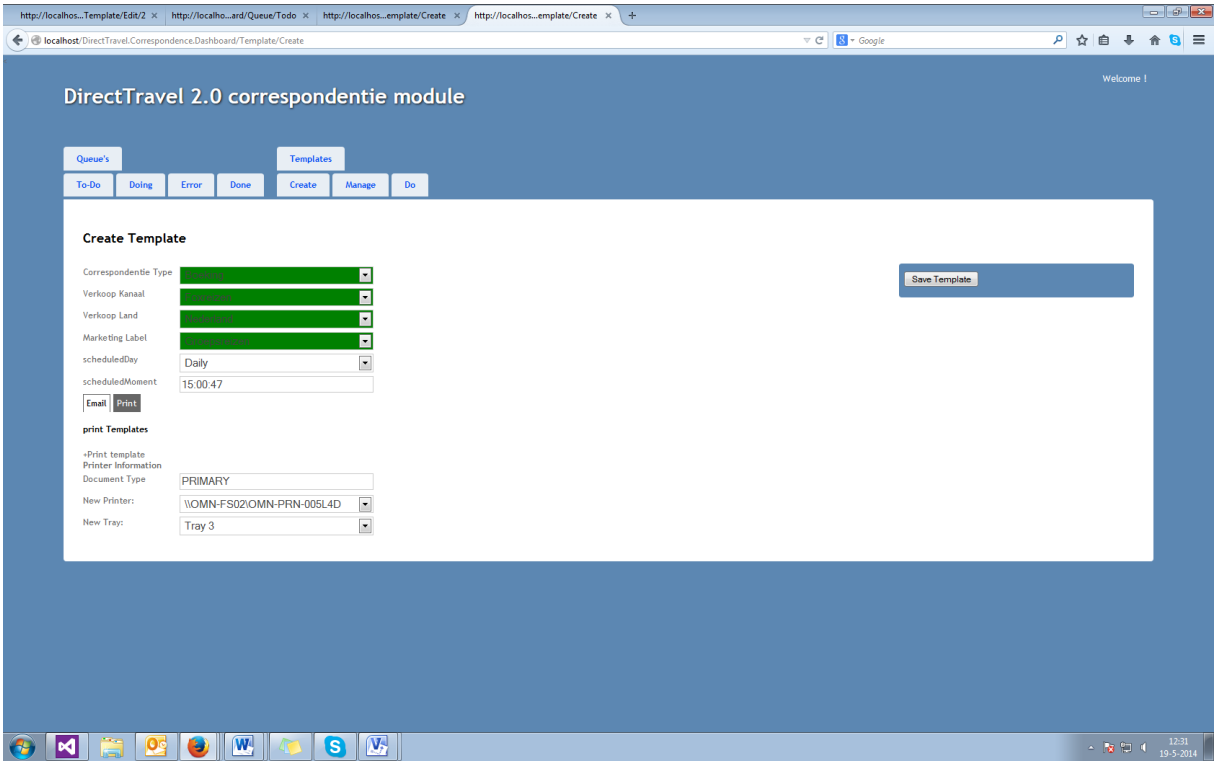
die de hoeveelheid gecontroleerde kenmerken -1. Indien dat het geval is wordt dat template teruggegeven, , indien hieruit geen template volgt recursieve aanroep waarbij de twee minst zware kenmerk waardes weg gedaan worden en de functie opnieuw uitgevoerd wordt.

Indien alle kenmerken gecontroleerd zijn en er is meer dan een template over wordt er gezocht naar een template waarbij er een match is met de hoeveelheid tot nu toe gecontroleerde kenmerken en die terug gegeven, indien er geen match is op de hoeveelheid wordt er gekeken naar het daarvoor gecontroleerde kenmerk en een match op de hoeveel gecontroleerde kenmerken -1. Indien dat het geval is wordt dit template teruggegeven indien er daaruit ook geen match is wordt de functie recursief aangeroepen en de twee minst zwaar wegende kenmerken weggelaten.

Indien er na controle van alle kenmerken nog maar een template over is wordt dezelfde routine toepast als er nog maar een template over is.

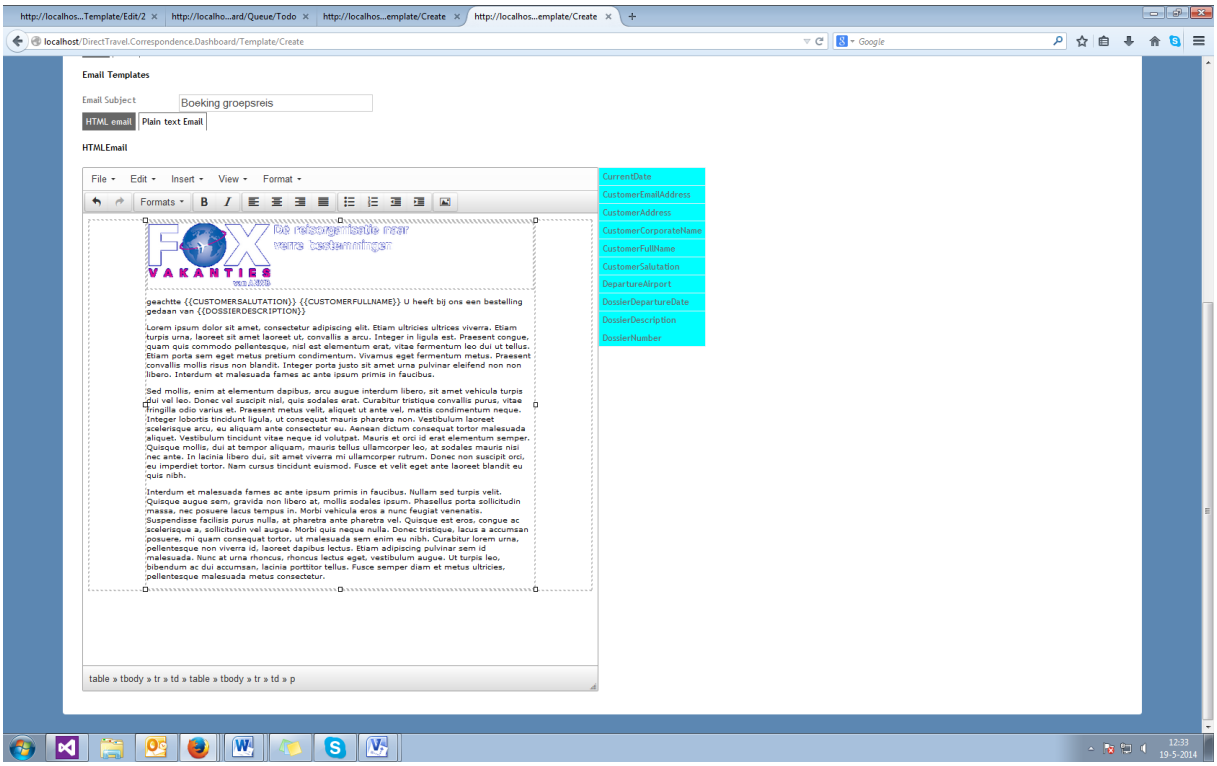
Als er uiteindelijk voor het meeste zwaarwegende template geen waarde te vinden is dan wordt er een exceptie geworpen, de module is dan niet goed ingeregeld of de aanroep is onjuist.

Het twee gedeelte van bouw van deze userstory bestaat uit de interface voor manipulatie van de template object, het inzichtelijk maken welke templates er zijn, het kunnen aanmaken van nieuwe unieke templates, en het kunnen bewerken de bestaande templates.



Figuur 9-2

Aanmaken van een nieuwe template de selectie van printers voor specifieke documenttypen



Figuur 9-3

De selectie van de HTML Email met de placeholders die op de cursorpositie worden toegevoegd en die door de templateprocessor vervangen kunnen worden door de tekst die op die plaats wenselijk is.

9.5. Tests

Een groot deel van deze tests wordt gedaan in het CRUDTemplate test, hier wordt gekeken of de test template gebruikt kan worden om de definieerde placeholders in het brondocument te vervangen.

Een deel van de test gaat over de invoer van de placeholders in het tempalte invoerscherm.

10. Userstory 4: Historie

10.1. Functionele Eisen

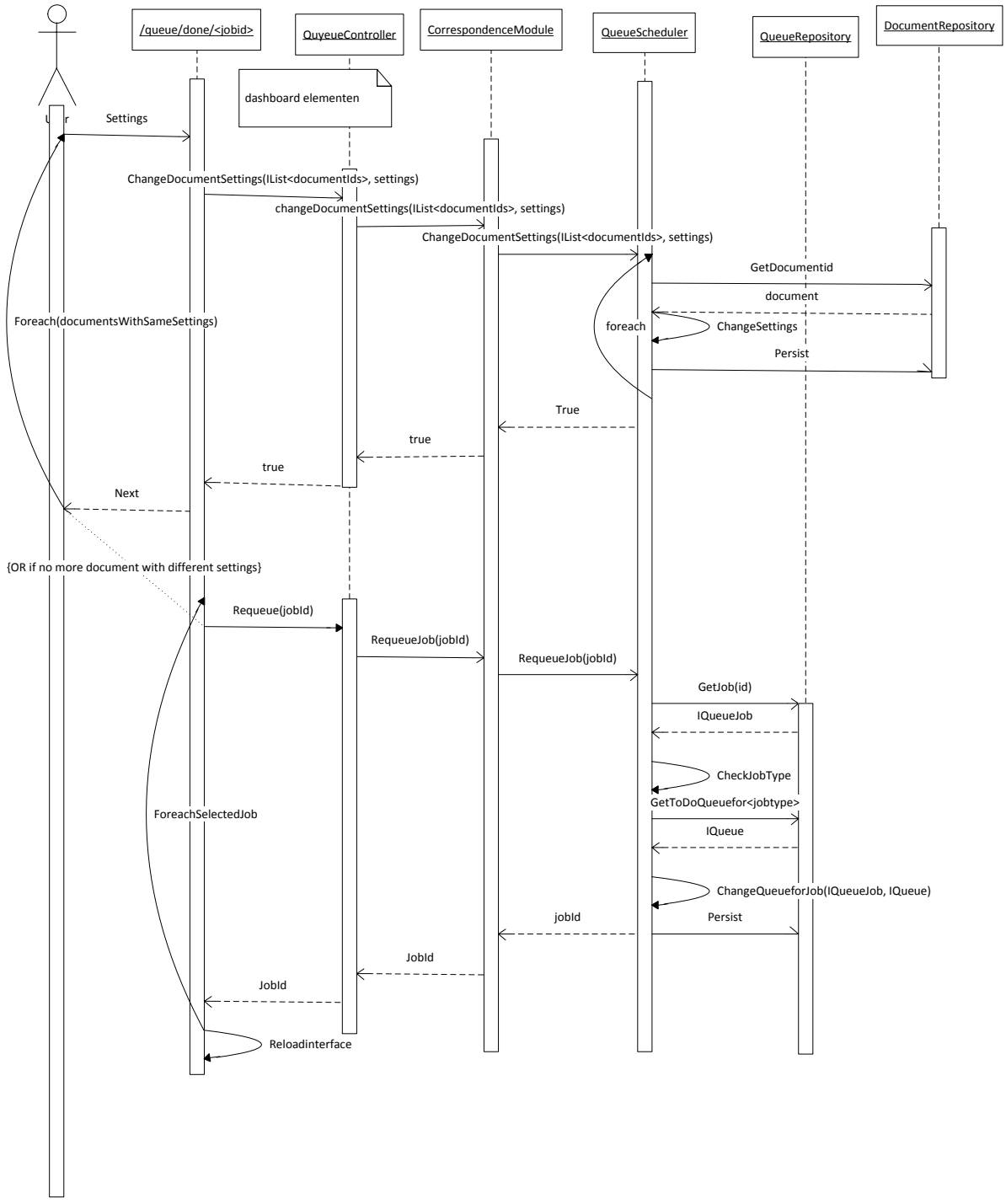
In de correspondentie module is een overzicht terug te vinden van alle afgehandelde communicatie. De verstuurde documenten worden opgeslagen in een documentmanagement systeem(Sharepoint)

De verzonden/geprinte documenten worden niet meer getoond in de queue en kunnen teruggezocht worden aan de hand van boekingskenmerken, moment van verwerken.

- Communicatie die verstuurd is kan weer in de verstuur queue worden geplaatst. Wanneer dit gebeurd zal de standaard opmaak van de communicatie opnieuw gegenereerd worden en wordt de communicatie verder op de dezelfde manier afgehandeld.

10.2. Technisch Ontwerp

De toegevoegde eis die voor de historie ontwikkeld moet worden is het kunnen her toevoegen van afgeronde taken. De eis voor het regenereren van de communicatie is niet van toepassing op de e-mails, hiervoor dient de reeds gegenereerde bestaande body hergebruikt te worden. De documenten worden uit de documentstore gehaald en vanuit daar opnieuw toegevoegd. Indien het een taak is voor de printer kunnen de printerinstellingen opnieuw ingesteld worden.



Figuur 10-1

De interactie van een gebruiker op het dashboard bij het opnieuw aan de taakwachtrij toevoegen van een of meerdere taken met een of meerdere printerinstellingen.

10.3. Bouw

De bouw van het Historie zit hem voor een klein deel in de backend, het doorzoeken van taken op basis van een set eisen gebeurt in de persistentie laag dit wordt vervolgens weer terug omhoog gebracht naar de UI. De uitdaging van deze Userstory zit hem in het kunnen tonen van deze gegevens zonder een pagina te hoeven verversen. Waar dit van zelfsprekend is in een winforms applicatie, moet hier voor in een webomgeving wat meer werk verzet worden.

JQuery

Binnen het project ben ik voor de web omgeving al een paar keer eerder terug gevallen op het gebruik van jquery zonder deze keuze expliciet toe te lichten. Jquery is een javascript framework wat het gebruik van gestandaardiseerde elementen over meerdere browsers mogelijk maakt. Het is de standaard bibliotheek om interactieve webpagina's te bouwen. Voor deze userstory heb ik op meerdere manieren gebruik gemaakt van javascript om gegeven van de view naar de controller laag te versturen en van de controller terug naar de view en vervolgens te verwerken. Met behulp van jquery kan ik dan vervolgens HTML elementen vervangen/uitbreiden met data die terug komt vanuit de server. Dit wordt onder andere gebruik bij het ophalen van historische gegevens, de zoekterm wordt verwerkt en op basis daarvan komt een object(model) terug bij de view. Samen met jquery wordt er vervolgens een basis object uitgebreid met de data van het model en op de juiste positie in de pagina terug geplaatst.

10.4. Testen

Semantische tests

Bij het opnieuw toevoegen van een artikel(foutief/reeds uitgevoerd) aan de wachtrij kan er gebruik gemaakt van de worden mogelijkheid de printer en lade instelling voor een of meerdere taken te vervangen. Bij het vervangen van de instellingen bij meerdere taken of bij een enkele taak met meerdere documenten moeten de documenten gegroepeerd op printer en lade door de gebruiker opnieuw instelbaar zijn. Daarnaast dient de verwachtte uitvoer, documenten worden geprint op de ingestelde printer en lade, gecontroleerd te worden.

Bij het doorzoeken van de historie van de taken kan er gezicht worden op taaknummer(jobId), boekingsnummer(Booking Id), of op het ontvanger adress. De tests die heribij horen gaan over het doorzoeken van een bestaand taak nummer, bestaand boekingsnummer en en bestaand ontvangeradress. Eveneens wordt er getest op niet bestaand nummers en adressen.

11. Userstory 5: Communicatie types

11.1. Functionele Eisen

De communicatie die via de correspondentie module wordt verstuurd is onderverdeeld in verschillende typen:

- Facturen
- Reisbescheiden
- Reminder email
    - Reminder na boeken
    - Reminder x aantal dagen voor vertrek
- Visum aanvraag
- Vanuit het systeem verstuurde brieven

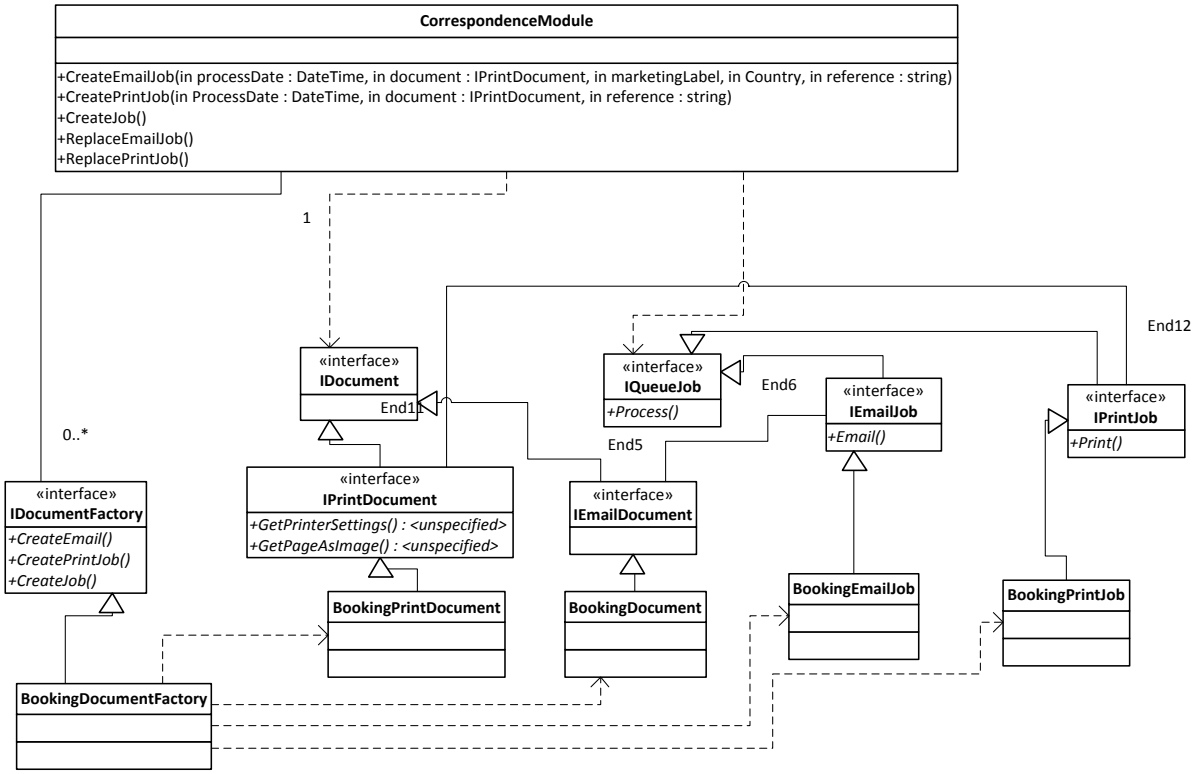
  Bijvoorbeeld een ontvangst bevestiging van een klacht vanuit de afdeling Customer Care

- Rapporten

Per communicatietype kan worden aangegeven of de documenten zowel per post als digitaal voor de klant beschikbaar gesteld kunnen worden.

De communicatie die via de correspondentie module afgehandeld wordt betreft de communicatie die uit DT gegenereerd wordt.

11.2. Globaal Ontwerp

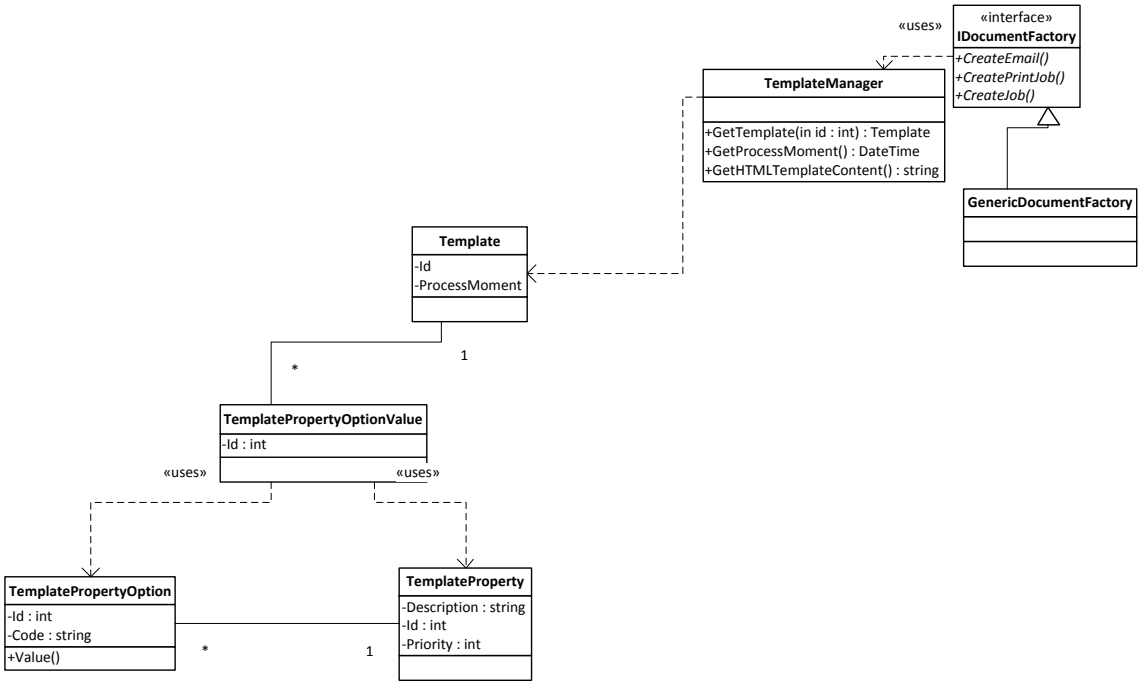


Figuur 11-1

Deze Userstory stond eerst uit een abstractFactory design pattern met voor elk communicatie type een implementatie van een IDocumentFactory, IPrintDocument, IEmailDocument, IEmailJob en IPrintJob. De keuze hiervoor heb ik gemaakt in het ontwerp van de basis zodat er per communicatietype op basis van specifieke bedrijfslogica bepaalde opmaak en afhandeling in kan stellen.

11.3. Technisch Ontwerp

Door voortschrijdend inzicht is de basis gelijk gebleven maar zijn de communicatie types vervangen opgenomen als property in de templatepropertyoptionvalues klasse als database geregistreerde entiteit.



Figuur 11-2

Door het communicatie type een Template Property te maken met de hoogste prioriteit kan dezelfde functionaliteit gehaald worden, en wordt de flexibiliteit van de toepassing hoger. Indien er in de toekomst een eis komt die specifieke logica voor een correspondentietype vereist kan er gebruik gemaakt worden van de structuur die opgezet is door de AbstractFactory Omtrent de IDocumentFactory uit te bereiden.

11.4. Bouw

De bouw van deze userstory bestond uit de bouw van de gebruikers interface en de achterliggende code voor het beheer van de diverse template kenmerken en die opties die er zijn voor deze kenmerken.

Deze is toegevoegd aan het dashboard project en maakt hierbij gebruik van de correspondentiemodule voor het toevoegen, wijzigen en verwijderen van de mogelijke kenmerken en hun opties.

Ik heb hier gekozen om opties standaard toe te voegen als laatste in de volgordelijkheid, deze kunnen vervolgens omhoog geschoven worden totdat ze op de juiste plek staan. Zodra er templates bestaan met andere selectie criteria voor volgordelijkheid dient de gebruiker zelf templates bij te voegen voor deze nieuwe optie/volgorde van opties.

11.5. Testen

In de CRUDjobtest voor taken is een random opgenomen die bij elke doorloop van de test een wisselende communicatie type uit de templateopties pikt en deze gebruikt bij het toevoegen van een taak.

Daarnaast wordt de template selectie binnen de CRUDTemplate test aan de tand gevoelt en wordt er een controle gedaan op het kunnen terugvinden van een “absolute” match van een template met een volledige set kenmerken(random selectie van een template en het daarna terugzoeken) eveneens wordt er test gedaan met het terug zoeken van een “closest” match.

Daarnaast dient bij het invoeren de volgorde van template opties overeen te komen met de volgordelijkheid van de opties, dit is een semantische test.

12. Afronding

Oplevering

Zoals gewoonlijk bij CCGroup wordt er eerst een interne acceptatietest gedaan door het beheer team van het opgeleverde product. Na deze acceptatie test wordt er nog klant acceptatietest. Pas daarna is de feitelijke oplevering op de productie omgeving van de klant.

Presentatie

Het opgeleverde product dient nog aan de keyusers gepresenteerd te worden. In deze presentatie wordt het gebruik van de module, het aanmaken van templates en het opvoeren van kenmerken besproken.

Verdere koppelingen

De koppeling met DirectTravel2.0 is al deels gerealiseerd maar vanuit DT2.0 komen er nog functionaliteiten bij die gebruik gaan maken van deze module. De opzet van de module is zo gedaan dat deze toevoegingen weinig voeten in de aarde hebben.

De opzet van de module is daarnaast zo dat de mogelijkheid geschapen is om de module te gebruiken voor het afhandelen van zaken in andere software pakketten in ontwikkeling bij de CCGroup. Alle web oplossingen en cloud gehoste omgevingen kunnen baat hebben bij deze module.

Nu het product af is kan de koppeling met DirectTravel2.0 gemaakt gaan worden en eventueel ook in andere producten. Het product van de afstudeeropdracht biedt in gebruikers in een centraal overzicht van e-mails die verstuurd dienen te worden en taken die geprint dienen te worden. Het biedt de gebruikers de mogelijkheden communicatie te standaardiseren en is eventueel gemakkelijk uitbreidbaar voor andere soorten van communicatie. De correspondentie module is door zijn opzet bruikbaar om gekoppeld te worden aan een cloud gehoste dienst die dan vervolgens de mogelijkheid krijgt om lokale printers aan te sturen.

13. Evaluatie

Project organisatie

Het overnemen van een deel van de gebruiken van het stage bedrijf heeft geleidt tot beter integratie van het project in de bedrijfscultuur, ik ben van mening dat het bijdraagt aan acceptatie van een project.

De systeem ontwikkel methodiek, hoewel afwijkend van de “normale” methode, past bij een informele organisatie zoals CCGroup. Het gebruik van “zware” overhead methodieken binnen deze organisatie zou alleen maar leiden tot nog meer weerstand tegen de methodiek. Een punt waaruit de organisatie lering zou kunnen trekken vanuit mijn optiek is het beter afschermen van het ontwikkelteam van de grillen van de klant/opdrachtgever. Zodra er van uit het ontwikkelteam een commitment afgegeven voor een sprint en de userstories die in deze sprint gerealiseerd worden dienen de eisen niet meer aangepast te worden tot de evaluatie aan het einde van de sprint. De userstories worden te flexibel opgesteld en gedurende de sprint kunnen de specificaties van een userstory nog wel eens wijzigen met alle gevolgen voor het reeds ontwikkelde stuk.

Ik heb binnen mijn project ook last gehad van userstories waarbij de eisen steeds meer in elkaar overgingen lopen en waarbij de eisen binnen twee userstories elkaar aanvulden, met het gevolg dat vanuit een ontwikkelperspectief het onderscheid tussen de stories niet goed vast te stellen is.

Proof of concept

Het gehele proof of concept was in mijn optiek een goede zet. De bouw ervan heeft mij een beter beeld gegeven van de gehele bedoeling van de opdracht en heeft zeker meegeholpen aan de beeldvorming van wat de opdracht inhoudt richting de opdrachtgever.

Duidelijke verbeter punten van het gebruik van een proof of concept is het onderstrepen dat richting de opdrachtgever dat het een proof of concept betreft. Ik heb nu en demonstratie gegeven van een proof of concept, het getoond en gedemonstreerd en de opdrachtgever was van mening dat wat er daar getoond wordt rechtstreeks geïmplementeerd kon gaan worden.

Userstory 1: Verstuur queue

De keuze om eerste een algemene wachtrij te implementeren en hierna de methoden voor verwerking hierin onder te brengen is een goede keuze gebleken. Hierdoor kon ik eerst onderzoek doen naar een oplossing waarin ik de verwerking lostrok van het toevoegen en dit in een minder complexe omgeving met stubs de afwerking ervan bekijken.

Deze userstory heeft mij echter door bug in de persistentie laag wel extra tijd gekost. Het verwerken van updates op de datastructuur kwam door een combinatie van een fout in de OR Mappings en een bug in de overwrite van de update methode niet door in de database. Ik heb hierdoor wat tijd verdaan bij het debuggen van het onderliggende systeem dat e persistentie afhandelt. Uiteindelijk heb ik hierdoor in de userstory wel tijd verloren.

Eveneens heb ik in deze userstory een structuur opgezet voor het afhandelen van diverse bedrijfslogca elementen die ik uiteindelijk via het Template designpattern opgelost heb in plaats van via het Abstract Factory Pattern. Hierdoor heb ik extra tijd gebruikt om implementerende klassen oor communicatiemethoden op te zetten die ik later op een andere manier verwerkt heb. Dit is een fout

in het ontwerp proces waarbij ik niet voldoende rekening gehouden heb met de mogelijk grote hoeveelheid verschillende communicatie types die in de afhandeling nauwelijks verschillen.

Userstory 2: Selectie printen/e-mailen

De implementatie van de verwerkingsmethoden heb ik aansluitend aan de bouw van de wachtrijen en verwerking aansturende onderdelen gedaan. De keuze hierin betekende voor mij dat ik vanuit het Proof of concept de ideeën kon implementeren op een Object georiënteerde manier. Dit had voor mij als voordeel dat ik het proof of concept ook daadwerkelijk als POC kon gebruiken.

Voor deze userstory ben ik tevreden met het opgeleverde resultaat en hoe dit tot stand gekomen is. Er is vrij weinig wat ik aan dit essentiële deel van de opdracht anders zou doen. In de loop van alle tests die ik gedaan heb is keer op keer gebleken dat de verwerking stabiel verliep en er altijd een correct resultaat teruggegeven werd.

Userstory 3: Standaard opmaak communicatie

De keuze voor het gebruik van een webinterface voor het aanmaken en muteren van templates is achteraf gezien hoewel het de flexibiliteit te goede komt niet een handige keuze. Voor het daadwerkelijk bouwen van de webinterface heb ik naar mijn menig teveel gebruik moeten maken van Javascript. Het gebruik hiervan van deze taal komt in mijn optiek de herbruikbaarheid en de code overdraagbaarheid niet ten goede. Het is lastig om dit om Javascript op een nette manier te construeren vanwege de extreme flexibliteit en het ontbreken van Object Orientatie in de taal zelf.

De oplossing omtrent de templates en het doorzoeken naar het juiste template of een template die een net een stap generiekere kenmerken heeft is wel goede uitgepakt. Het matchen hiervan had achteraf gezien wel meer gebruik kunnen maken van het verschuiven van de load van het zoeken naar de database, met minder query's op de database.

Userstory 4: Historie

Over deze userstory is het lastig wat te zeggen, het grootste gedeelte van het denkwerk van deze story was al op andere momenten opgepakt. Het opnieuw toevoegen van taken aan een wachtrij met het wijzigen van kenmerken had ik achteraf gezien wel meer tijd aan willen besteden. Deze eis was origineel ook niet opgenomen en is pas aan het einde via een wijziging in het functioneel ontwerp toegevoegd. Als ik deze eis had meegenomen had ik een andere oplossing ervoor kunnen ontwikkelen, een oplossing waarbij ik op een lager niveau dit had kunnen oplossen.

Userstory 5: Communicatie types

Deze userstory bestond in eerste instantie uit losse story's met vrijwel gelijkwaardige eisen. Er werd in deze userstory een stel communicatie types genoemd zonder verdere implementatie details, dit heb ik opgevat als iets wat later specifieke bedrijfslogica aan toegevoegd dient te worden en dat dus ook later ingevuld dient te worden. Op basis van deze niet teruggekoppelde aanname heb ik een deel van het ontwerp zo opgezet dat het gemakkelijk te pas gemaakt kan worden. Door de mutatie aan de eisen werd echter duidelijk dat het niet ging om specifieke implementatie logica maar om logica voor een veelvoud aan types, een het kunnen toevoegen en wegenemen van deze types.

14. Beroepstaken

14.1. Beroepstaak 3.2: Ontwerpen systeemdeel

Filosofie

Er is gekozen voor een DomainDrivenDesign filosofie. Dit betekend dat er een scheiding in het ontwerp zit tussen uitvoerende en uitgevoerde objecten. De domein objecten zijn opgezet met het idee dat ze verwerkt worden door een object met businesslogica. Deze filosofie maakt gebruik van de basis principes van ObjectOrientatie.

Het klassediagram is daarnaast opgezet met een waarbij de basis van het domein en stukken bedrijfslogica ligt in interface. Een abstractie laag die ligt op de implementerende klassen waardoor de code onderhoudbaarheid en uitbreidbaarheid gegarandeerd kan worden. Deze filosofie verplicht het verkleinen van de koppeling tussen de klassen en verhoogt de interne cohesie van een klasse. Daarnaast maakt de view automatisch gebruik van elementen uit de service en wordt er dus altijd gestreefd naar een maximale scheiding tussen view en model.

Design patterns

In het ontwerp worden diverse design pattens gebruikt, de eerst opvallende is de Abstract factory voor het aanmaken van taken en documenten bij deze taken. Dit design pattern zorgt voor een uitbreidbare business logica voor taak typen.

Daarnaast zijn er ook minder opvallende patterns gebruikt. Het singleton pattern is gebruikt voor de queuescheduler. Hierdoor wordt deze nooit meervoudig geinitialiseerd en loopt er altijd maar een taak controlerende timer. Een prototype wordt gebruikt voor de generalisatie van de interfaces die door de abstractfactory gebruikt worden(IQueueJob -> IEMailJob -> GenericEmailJob).

Daarnaast zijn er ook structural patterns terug te vinden. De diverse Infrastructure elementen die opgezet zijn als adapter naar de verwerkende systemen(SMTPAdapter, printerAdapter, documentStoreAdapter). Composite patterns zijn toegepast op de subonderdelen van de email/templates.

En als laatste de behavioral patterns. Binnen het vervangen van stukken van teksten met placeholders wordt gebruik gemaakt van dynamische implementatie van de vervanging methode., afhankelijk van het te vervangen element wordt een andere vervangingsmethode toegepast, dit is een strategy pattern.

Complexiteit

De complexiteit van het ontwerp zit hem in de hoeveelheid en de functionele splitsing van deel functionaliteiten. Een deel van het systeem is functioneel verantwoordelijk voor het aanmaken van taken een deel van het systeem is verantwoordelijk voor het verwerken van taken en een deel van het systeem is verantwoordelijk voor het voorzien van de juiste informatie om de twee aan elkaar te koppelen.

14.2. Beroepstaak 3.3: Bouwen applicatie

Taal

De applicatie is opgezet in het .net 4.5 framework en maakt gebruik van C#. Binnen CCGroup wordt er voor Microsoft systemen enkel en alleen gewerkt in deze taal, hierdoor is de code overdraagbaarheid hoger.

Daarnaast is er nog een sub project, het Dashboard. Dit is een MVC project die veel gebruik maakt van Javascript voor viewmanipulatie.

OTAP

Binnen CCGroup worden applicaties ontwikkeld in een OTAP Straat. Hierdoor dient tussen verschillende omgevingen te schakelen zijn, hiervoor dienen diverse onderdelen instelbaar gemaakt te zijn(database connectie strings, instellingen die tussen ontwikkel en test en die per klant kunnen verschillen). Daardoor maakt de applicatie gebruik van los instelbare configuratie files.

S' nachts draait er op de automatische build server een proces waarin alle via TeamFoundationServer ingecheckte projecten gebuild worden, en vervolgens alle unittests op gedraaid worden. Dit proces garandeert dat de code functioneert op andere systemen als die van de ontwikkelaars en dat er tijdig gesignaleerd kan worden of er door updates aan de code fouten in andere delen van de code geslopen zijn.

Bouw

Bibliotheken

Specifiek voor de Correspondentie Module heb ik enkele bibliotheken geïmplementeerd. Een aantal van deze heb ik in een proof of concept tegenover elkaar getest om de voor de applicatie best functionerende bibliotheek te kunnen gebruiken. Een aantal complexe mogelijkheden van deze bibliotheken zijn gebruikt om de conversie van PDF format naar Bitmap te doen.

Koppelingen

De software is gekoppeld met diverse systemen:

- SMTP server(Exchange en andere alternatieven) voor het e-mailen
- Printers (voor afdrukken)
- Sharepoint(documentstore oplossing waarvoor deze klant voor gekozen is)

Ook heb ik de keuze gemaakt om bepaalde software systemen juiste niet te koppelen om herbruikbaarheid van de applicatie intact te houden. Zo is er expliciet niet gekoppeld met DirectTravel zodat de Correspondentie module niet afhankelijk is van dat softwarepakket.

Complexiteit

De complexiteit van de applicatie ligt op diverse vlakken. Het is een vrij grote applicatie die over veel klassen(ongeveer 110 klassen) vrij veel code heeft(gemiddeld 90 regels code per klasse). Daarnaast is er vrij veel gebruik gemaakt van vaak complexere technieken om performance verbeteringen te kunnen realiseren. Hierbij kan gedacht worden aan het verwerken van taken, het invoeren van HTML tekst met placeholders, het bepalen van de toe te passen templates en diverse javascript oplossingen. Daarnaast dient het geheel aan te sluiten op de eisen die gesteld kunnen worden aan

communicatie vanaf een reisorganisatie systeem, waarbij de verschillende communicatie typen, afhandelingen en verwerkingen uiteen kunnen lopen van het verwerken van een simpele reminderemail naar een klant tot het versturen van management rapportages. Van het printen van een document tot het printen van een document met printer en lade eisen enerzijds en printer en lade eisen anderzijds tot complexe pagina voor pagina batch operaties met vastgezette volgordelijkheid.

14.3. Beroepstaak 3.5: Uitvoeren van en rapporteren over het testtraject

Filosofie

Binnen DT was er al een keuze gemaakt voor een deel van de opzet van testen. Het gros van de testen gebeurde al zoveel mogelijk geautomatiseerd. De ontwikkelomgeving was hier speciaal op ingericht. De keuze om een deel van de testdiepte te halen door geautomatiseerde testen uit te voeren was daarom vrij gemakkelijk.

Keuze van tests

Per userstory heb ik een set testen geselecteerd en gebouwd indien het unittests waren of indien het invoertesten betreft overgedragen aan de test afdeling.

Daarnaast heb ik gekozen om een aantal testen buiten de directe scope toch mee te nemen omdat het behalen van een bepaalde performance een niet genoemde maar wel gewenste vereiste is. Deze extra wensen zijn wel uitgevoerd maar niet meegenomen in de eindoplevering aan de klant. Deze nummer bieden een indicatie van de te verwachtten bottlenecks en kunnen gebruikt worden als referentie kader van wat de grenzen van “normaal” gebruik.

Daarnaast heb ik nog een set handmatige testen over de interface gedaan die de functionele eisen die aan de applicatie gesteld worden kunnen aftesten. Met deze tests kan wat gezegd worden over de implementatie. De geautomatiseerde unittests vormen de regressiebarrière, zodat bij code wijzigingen altijd getest worden of deze wijziging in een deel van de code mogelijk een bug geïntroduceerd hebben.

Rapportage van tests

Alle tests die uitgevoerd zijn gebundeld in een rapport dat opgeleverd is aan de opdrachtgever. De conclusie bevat aanbevelingen over het verwerken van de technische tests als zijnde “safe working limits”, en accordeert de functionele tests voor verdere evaluatie door de acceptatie testers.

Bijlagen

Plan van aanpak	Bijlage I
Functioneel ontwerp	Bijlage II
Technisch ontwerp	Bijlage III
Test Rapport	Bijlage IV