Afstudeerdossier Deel1/2 Afstudeerverslag
Hoffelijk registreren

Student naam:	Jeroen Mooijman
Studentnummer:	99009154
Opdrachtgever:	W. Broers (KCC)
Organisatie:	Gemeente Den Haag, Klant Contact Centrum
Plaats:	Den Haag
Examinatoren:	E.M. van Doorn/ W. van Vliet
Opleiding:	Informatica
Datum:	03-10-2011

Referaat

- Gemeente Den Haag
- Registratie klantcontacten
- Rapportage klantcontacten
- Automatische rapportages
- Yii Framework
- PostgreSQL
- Unified Processing

Voorwoord

Een kans om af te studeren terwijl ik werk. Dat is wat ik op mijn huidige werkplek heb gekregen. Ik zou daarom graag het Klant Contact Centrum van de gemeente Den Haag heel erg willen bedanken voor deze kans.

Inhoudsopgave

Inleiding 1

Deel I: Het KCC 2

1. Gemeente Den Haag 3
   1.1. Organisatie algemeen 3
   1.2. De organisatie bij de DPZ en op de afdeling KCC 4
   1.3. Positie binnen het KCC 4

2. Probleem en doelstelling 5
   2.1. Probleemstelling 5
   2.2. Doelstelling 6

Deel II: Het proces 8

3. Opstart van het project 9
   3.1. Methoden en technieken 9
   3.2. Plan van aanpak 11
   3.3. Eisen duidelijk definiëren 14
   3.4. Framework kiezen. 17
   3.5. Vormgeven van de systeemfuncties in UML. 19

4. Ontwerpen van de applicatie 23
   4.1. Gegevensmodel opstellen 23
   4.2. Prototype bouwen en beta-testen 32

5. Deployment in de acceptatieomgeving. 39
   5.1 Gegevensconversie 39
   5.2 Module voor rapportages 42
   5.3 Aanpassingen werking schermen 58
   5.4 Rapporteer module opnemen in de acceptatieomgeving. 61

6. Het in gebruik nemen van de applicatie. 72
   6.1 Het uitrollen van de applicatie. 72
   6.2 Het overdragen van de applicatie. 73

Deel III: De evaluatie 75

7. Evaluatie 76
7.1. Procesevaluatie 76
7.2. Productevaluatie 77
7.3. Beroepstaken 78

Literatuurlijst 81

Verklarende woordenlijst 82

Bijlage zijn aanwezig in afstudeerdossier deel 2 84

Inleiding

Een applicatie waar geen ondersteuning voor is binnen en buiten de organisatie, wat doet een organisatie daarmee? Het komt vaak voor dat het bedrijf de applicatie net zolang blijft gebruiken tot de applicatie niet meer werkt. Daar schuilt een gevaar in, het proces wat gebruik maakt van deze applicatie kan stil komen te liggen als de applicatie niet meer werkt. Om dit voor te zijn wil het Klant Contact Centrum (KCC) van de gemeente Den Haag een oude applicatie de deur wijzen. Deze applicatie geeft de medewerkers de mogelijkheid om te registreren wat voor klantcontact er is geweest. Ook verzorgt de applicatie met tussenkomst van een medewerker de rekeningen die het KCC uitstuurt naar zijn klanten. In het kader van kostenbesparing is er ook behoefte aan de automatisering van de opmaak van rekeningen. Het KCC heeft er daarom voor gekozen een applicatie te laten ontwikkelen binnen de eigen dienst.

Deze opdracht heeft als doel een applicatie te ontwikkelen die een arbeidsintensief proces kan automatiseren. De uiteindelijke applicatie moet de gebruiker de mogelijkheid geven om, klantcontacten te kunnen registeren, deze klantcontacten op te slaan, deze gegevens in een volledige rapportage op te kunnen vragen en zelf de regie te behouden over de applicatie.

Deel I: Het KCC

In dit deel zal ik in gaan op de organisatie van de gemeente Den Haag en het KCC.

1. Gemeente Den Haag

1.1. Organisatie algemeen

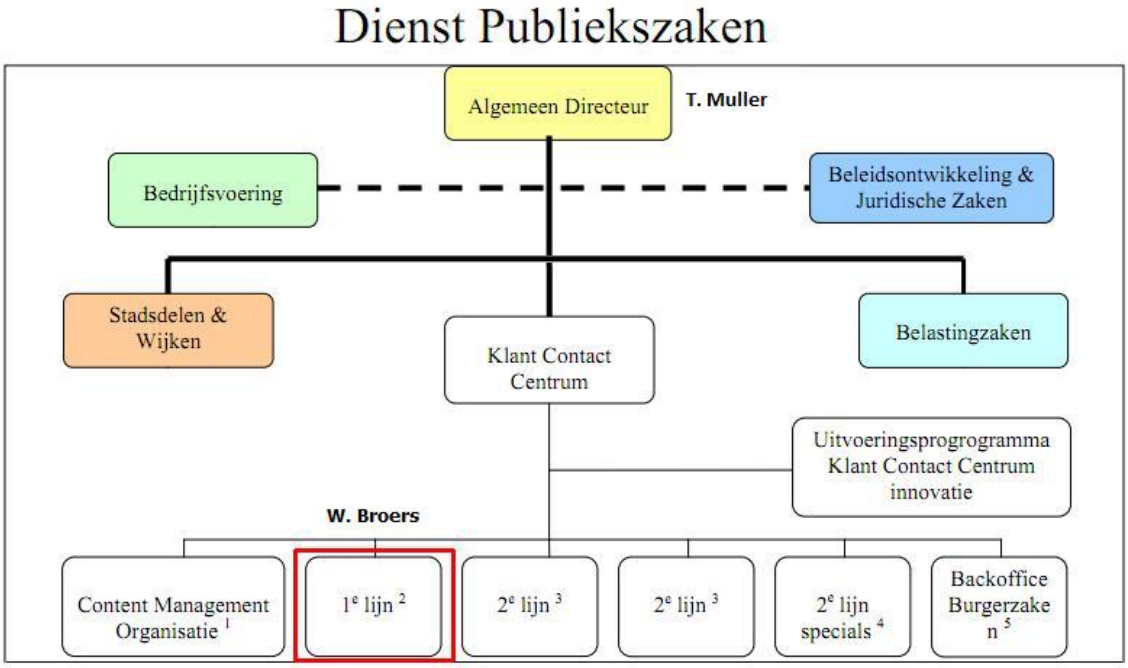
De gemeente Den Haag is verdeeld in tien diensten, die nodig zijn om de interne en externe klanten zo goed mogelijk van dienst te zijn. De volgende diensten zijn er binnen de gemeente:

- Bestuursdienst (BSD)
- Dienst Publiekszaken (voorheen Burgerzaken) (DPZ)
- Gemeentelijke Accountantsdienst (GAD)
- Haeghe Groep (HGR)
- Onderwijs, Cultuur en Welzijn (OCW)
- Openbare Bibliotheek (DOB)
- Dienst Sociale Zaken en Werkgelegenheidsprojecten (SZW)
- Dienst Stadsbeheer (DSB)
- Dienst Stedelijke Ontwikkeling (DSO)
- Intern Diensten Centrum (IDC)

De diensten hebben allemaal een eigen vakgebied. De bestuursdienst geeft ondersteuning aan het bestuur. De dienst stadsbeheer zorg voor een schone en leefbare stad. De Dienst Publiekszaken is de eerste ingang voor de burger om in contact te komen met de gemeente Den Haag.

Elke dienst heeft een eigen organigram. Omdat de afstudeeropdracht plaats vond bij de Dienst Publiekszaken zal alleen van deze dienst een organigram getoond worden.

1.2. De organisatie bij de DPZ en op de afdeling KCC



Afbeelding 1 Organigram DPZ gemeente Den Haag.

De afdeling waar de afstudeerwerkzaamheden plaats vinden, staat aangegeven door middel van de omlijning in afbeelding 1. De afdeling heeft als naam Klant Contact Centrum (KCC) 1e lijn. Voor deze afdeling is er een staf voor de ondersteuning van de organisatie en de processen binnen de afdeling. Voor de aansturing van de medewerkers (agents) zijn er teammanagers. De manager van de afdeling vormt samen met de teammanagers het managementteam (MT).

Wat is het primaire proces binnen het KCC?

Aanspreekpunt zijn voor de burgers van de gemeente Den Haag is het primaire proces. Dit proces gaat over het beantwoorden van vragen over en aan de gemeente Den Haag. Dit kan variëren van een vraag over een vergunning tot een aanvraag om een glascontainer te legen. De vragen kunnen via alle mogelijke kanalen binnenkomen. Denk hierbij aan Internet, telefoon, faxen en brieven.

1.3. Positie binnen het KCC

Ik werk sinds 2008 voor het KCC als Medewerker Informatie Junior. Als Medewerker Informatie Junior beantwoord ik telefonisch vragen van burgers. In deze functie ben ik gevraagd om technische ondersteuning te bieden bij het KCC gezien mijn studieachtergrond. Dit varieert van het maken van rapportages tot het verhelpen van storingen. Op dit moment maak ik deel uit van de staf binnen het KCC.

2. Probleem en doelstelling

2.1. Probleemstelling

Het gemeentelijk Contact Centrum heeft op dit moment een registratie-applicatie voor klantcontact, om in kaart te brengen waar de klanten contact over opnemen. Deze applicatie heeft geen technisch en functioneel beheerder en de leverancier is niet meer bekend binnen de gemeente. De medewerkers van het contact centrum gebruiken deze applicatie dagelijks met ongeveer 40 personen tegelijkertijd. De mogelijkheid om de categorieën van klantcontact binnen de applicatie aan te passen is aanwezig en kan via een webpagina aangepast worden. Dit gaat helaas niet intuïtief en er is geen handleiding beschikbaar. De applicatie is in 2006 ingeregeld binnen de gemeente en heeft daarna nooit meer een update gehad. Omdat de omgeving van het systeem waar de applicatie op geïnstalleerd is sterk is veranderd en de applicatie zelf niet mee veranderd is, geeft dit problemen met de stabiliteit en compatibiliteit.

De applicatie is een CRUD-applicatie voor het aanmaken van klantcontact gegevens. Dit klantcontact bestaat uit de volgende gegevens:

- de manier van binnenkomst
- welke dienst het klantcontact betreft
- over welk onderwerp het klantcontact ging
- de afhandelwijze van dit klantcontact
- welke medewerker het klantcontact heeft afgehandeld

Om inzichtelijk te krijgen wat voor klantcontact er is geweest er waar dit contact over ging, is het mogelijk om rapportages uit te draaien waar een combinatie van de bovenstaande gegevens in gebruikt kan worden. Het aanmaken van een maandrapportage is tijdrovend.

Het contact centrum gebruikt deze rapportages om rekeningen op te maken voor de diensten binnen de gemeente. Om deze rekeningen te maken, dient een medewerker tien verschillende rapportages uit te draaien en naast elkaar te houden om daaruit cijfers te distilleren.

De cijfers komen uit verschillende bronnen:

- De Customer Interaction Client(CIC) van Newtel essence [1]
  Deze bron geeft het volume van het telefoonverkeer weer. Er is informatie beschikbaar zoals het aantal gesprekken dat daadwerkelijk gevoerd is, in welke tijdspanne en hoe lang een klant heeft gewacht voordat hij geholpen werd. De informatie uit deze bron is het startpunt voor de rapportage. Alle cijfers die verder naar voren komen uit andere bronnen, worden handmatig gekoppeld met deze informatie.

[1] http://www.newtelessence.com/

- Het roostersysteem WFM Verint [2]
  Deze bron geeft informatie over welke medewerkers gewerkt hebben, hoe lang en wanneer ze dit gedaan hebben.
- De huidige registratie-applicatie.
  Hierin staan alle eerder genoemde gegevens over het afhandelen van het klantcontact. Verschillende combinaties van deze informatie vormen een onderdeel van de uiteindelijke rapportage. De registratie-applicatie biedt geen mogelijkheid om deze informatie direct te koppelen aan de andere bronnen. Alles dient hier handmatig te worden opgeslagen en daarna kunnen deze rapportages gebruikt worden voor het opstellen van een aantal Excel bestanden die de basis vormen voor de rekeningen.

De cijfers over het aantal gesprekken uit de CIC en gegevens uit de registratie-applicatie komen terug in een Excel bestand. In dit bestand vinden berekeningen plaats om de totalen te krijgen voor het Contact Centrum. Dit Excel bestand dient als basis voor de verdere verdeling per dienst. Elke dienst heeft een eigen Excel bestand waar de gegevens per dienst in staan.
De structuur van dit bestand per dienst is voor elke dienst gelijk, de data die hierin staat niet. Een combinatie van de data uit het Excel bestand van het contact centrum en nieuwe rapportages uit de registratie-applicatie geeft een rekening per dienst. Deze werkwijze zorgt ervoor dat er elke maand één medewerker minimaal voor één dag werk heeft met het opstellen van deze rapportages, Excel bestanden en rekeningen.

Naast deze rekeningen willen de managers op het Contact Centrum ook weten wat het aantal registraties is van de medewerkers en hoeveel gesprekken zij per week/dag/uur voeren. Om dit te realiseren moet er elke week een uitdraai van de individuele gesprekken gemaakt worden uit de CIC, het aantal registraties uit de registratie-applicatie en het aantal werkuren uit het roostersysteem WFM. Dit verwerkt een medewerker tot een geheel. Deze informatie is beschikbaar voor de managers en deze stellen een keer per kwartaal de medewerkers op de hoogte van deze cijfers. Dit is niet frequent genoeg om goed te kunnen sturen op het aantal van de registraties.

2.2. Doelstelling

De doelstelling van deze opdracht is om een nieuwe webbased applicatie te ontwikkelen met de volgende kenmerken:

- het mogelijk maken om klantcontacten te registreren.
- het mogelijk maken om rapportages op te vragen.

[2] http://www.teletrain.nl/software-solutions/performance-management/

- het mogelijk maken rapportages wekelijks en maandelijks automatisch samen te stellen om zo het proces van de rekeningen opmaak te vereenvoudigen en zo kosten te besparen.
- het mogelijk maken voor een functioneel beheerder om de applicatie te beheren doormiddel van een beheerfunctie in de applicatie.
- de oude data moet beschikbaar blijven middels een gegevensconversie.
- baseren op een nieuw systeem binnen de bestaande systemen en omgeving van de gemeente. Stabiliteit staat daarbij voorop.
- de kwaliteit van de registraties proberen te verbeteren door het aantal registraties zichtbaar te maken op het gebruikte wall-display van Quickcom[3]. Als de informatie over het aantal registraties real-time zichtbaar is op het wall-display, is het mogelijk voor de managers om de medewerkers op slecht registreren te wijzen.
- de onderdelen binnen de applicatie dienen gedocumenteerd te worden zodat de applicatie bij de technisch en functioneel beheer groep in beheer kan worden genomen.

[3] http://www.txdigital.com/products/quickcom.php

Deel II: Het proces

In dit deel gaat het over de werkzaamheden die ik tijdens deze periode bij het KCC uitgevoerd heb. Het zal gaan over het proces dat ik heb doorlopen om deze opdracht uit te voeren.

3. Opstart van het project

Voordat het project gestart kan worden ligt de nadruk op het opstellen van een plan van aanpak en het definiëren van de activiteiten die bij deze opdracht horen. Het meeste werk voor de opstart van dit project is gedaan voor het aanvangen van het project. Ik rekende op een doorlooptijd van drie weken, met uitloop naar vier weken. De hoofdactiviteiten bestaan uit het duidelijk definiëren van de opdracht, de te gebruiken methodiek, het kiezen van een framework voor de applicatie en het vormgeven van de systeemfuncties in UML.

3.1. Methoden en technieken

Tijdens mijn opleiding ben ik onderwezen in twee verschillende methodes om software te ontwikkelen: Prince2[4] en (R)UP[5]. Voor dit project heb ik deze twee methodes meegenomen in de overweging voor de te gebruiken methode. Door naar het bedrijf te kijken en daarbij ook andere methodes te betrekken, moet ik een afweging kunnen maken waarbij de nog op te stellen planning ook in de gaten moet worden gehouden.

De gemeente Den Haag is een bedrijf waar iets wat gisteren perfect aansloot op de werkwijze van gemeente en de klantvragen, vandaag al niet meer actueel is of niet meer volgens de wet werkt. Dit inzicht geeft het probleem met de oude applicatie goed aan. De applicatie is een vast punt geworden binnen het contact centrum welke, bij bijvoorbeeld een verandering in werkwijze, niet aangepast kan worden. Het contact centrum moet nu om de applicatie heen werken bij andere en/of nieuwe werkzaamheden. Daaruit kan je concluderen dat het wenselijk is om in een project ruimte open te houden voor doorontwikkeling.

Vaak zijn veranderingen binnen het contact centrum een verbetering van de oude werkwijze en kan er tussentijds geëvalueerd worden of een verandering te goede is om zo de werkwijze aan te passen/te verbeteren. Vaak begint een (nieuwe) werkwijze als proef met een klein groepje medewerkers. Als deze proef goed doorlopen is kan er een werkwijze gemaakt worden voor alle medewerkers met daarin de punten verwerkt die naar voren zijn gekomen tijdens de proef. Deze werkwijze is al meerdere malen toegepast binnen de organisatie. Hier kon ik uit concluderen dat het contact centrum werkt met een vorm van iteraties.

[4] http://nl.wikipedia.org/wiki/PRINCE2
[5] http://en.wikipedia.org/wiki/Unified_Process

Het bedrijf past bij meerdere methodes waaronder de twee onderwezen methodes. Andere methodes zoals Rapid application development (RAD)[6] en Agile[7] passen ook bij het groepsgewijs uitrollen van andere/nieuwe werkwijzen.

Voor de realisering van deze opdracht is de keuze gevallen op UP. Deze keuze heb ik vooral gemaakt omwille van de planning. De tijd die nodig is om een nieuwe methode onder de knie te krijgen, kan in dit geval ook ingezet worden in het uitvoeren van de opdracht. Gezien de overlapping die de meeste methodes hebben zoals het iteratief en object georiënteerd werken (RAD, UP en Agile) zag ik weinig meerwaarde in het leren van een nieuwe methode.

Er is een overweging geweest voor Prince2, waar andere afdelingen van de gemeente Den Haag mee werken. Het contact centrum zelf werkt hier niet mee en de afdelingen die het beheer uitvoeren ook niet. Prince2 past in mijn ogen ook minder goed bij het beeld van applicatie-ontwikkeling. Het is meer een generieke project methode voor een eindig project terwijl er behoefte is aan de mogelijkheid om het project "levend" te houden. Deze opdracht in UP onderdeel maken van een groter project in Prince2 zou het beste geven van beide methodes. Helaas was er geen project waar deze opdracht bij zou kunnen aansluiten.

De keuze voor UP sluit ook aan bij de veranderlijkheid en werkwijze van het contact centrum. Het contact centrum blijft bezig zich te ontwikkelen en te verbeteren, zij het verplicht zij het vrijwillig. Een applicatie in deze omgeving is daardoor ook gevoelig voor veranderingen. De opdrachtgever wil na het beta-testen nog de mogelijkheid hebben om een functionele eis toe te voegen aan de hand van feedback van de testers. In dat geval zou een iteratie deze uitkomst bieden.

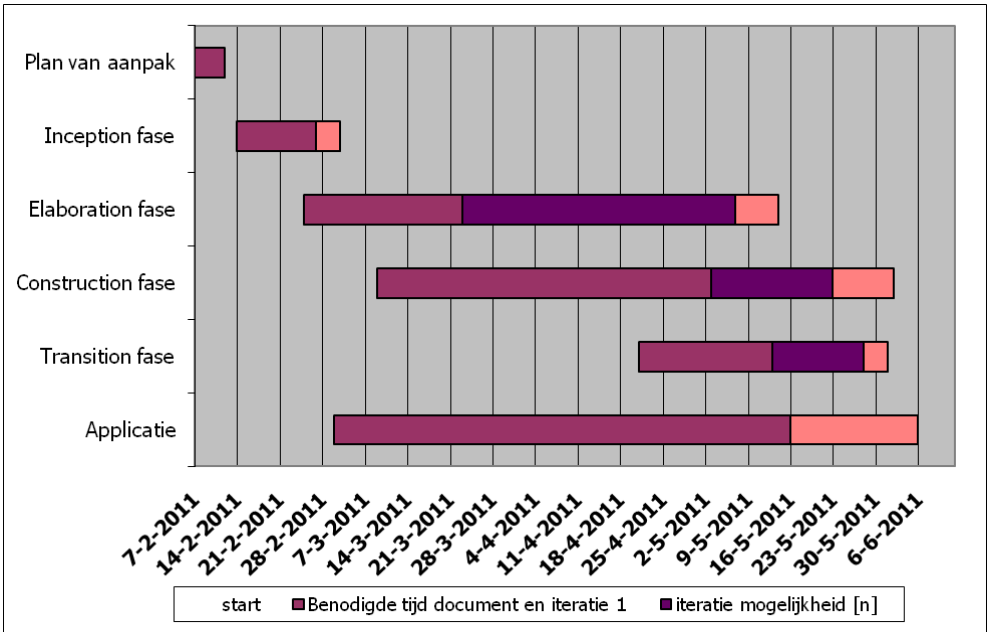
Het werken met UP geeft de opdrachtgever de mogelijkheid om in verschillende stadia te testen. Het testen kan al starten in het begin van de ontwikkeling en daardoor kan de opdrachtgever inschatten of de applicatie lijkt op wat hij voor ogen heeft en sturing/remming geven waar nodig. De opdrachtgever kan hierdoor ook inschatten of er een training nodig is voor de medewerkers die de applicatie gaan gebruiken. Als dat zo is dan kan de trainer betrokken worden bij het project om dit te kunnen verzorgen. De trainer kan zelf ook testen en input leveren om de applicatie nog beter aan te kunnen laten sluiten bij de organisatie. Hiervoor zou RAD natuurlijk ook een goede kandidaat zijn, wat een overweging kan vormen voor deze methode bij de start van een ander project waar meer tijd beschikbaar is.

[6] http://nl.wikipedia.org/wiki/Rapid_application_development
[7] http://nl.wikipedia.org/wiki/Agile-software-ontwikkeling

De applicatie zal voor een groot deel bestaan uit het maken van rapportages, dit weet ik door mijn ervaring met het oude programma, het uitwerken van de inhoud/omvang deze rapportages zal ik in de inception fase doen. Omdat ik denk dat de opbouw van de rapportage in grote lijnen hetzelfde is, zal ik uitgaan van een basis, waar componenten bij ontwikkeld kunnen worden die gebruik maken van deze basis. UP maakt gebruik van componenten in een object georiënteerde omgeving. Zo kan de opdracht in kleinere delen gesplitst worden en per onderdeel getest worden. Er is dan een grotere kans op onafhankelijkheid van componenten binnen het systeem en daardoor is er goed terug te grijpen op vorige versies bij gebruik van een subversie systeem.

3.2. Plan van aanpak

De opstart van dit project heeft de basis gevormd voor het plan van aanpak. Tijdens de opstart van het project was het van belang dat de opdracht duidelijk gedefinieerd werd en dat er keuzes werden gemaakt voor ontwikkelmethodes en technieken. Het project is op te delen in twee delen, het hoe en wat, waarvan de scope duidelijk vorm gegeven moet worden om dit project tot een goed einde te brengen. Het hoe gedeelte is opgenomen in het plan van aanpak. Parallel aan het plan van aanpak verliep het opstellen van het inception rapport. In het inception rapport komen delen van het plan van aanpak terug met daarbij een eerste opzet van applicatie, het wat gedeelte. In het plan van aanpak is onder andere opgenomen wat de globale planning is, welke mijlpalen er verwacht worden in de loop van het project en welke producten er opgeleverd worden aan het einde van het project. De globale planning heeft als basis de fases welke gedefinieerd zijn in UP en ziet er als volgt uit.



afbeelding 2 de globale planning in grafische weergave.

De planning geeft een globaal beeld van de fases die zal doorlopen en deze fases leveren rapporten op die corresponderen met de fases. De op te leveren producten zijn de volgende:

- Plan van Aanpak
- Inception rapport
- Elaboration rapport
- Construction rapport
- Transition rapport
- Geïmplementeerde wensen en eisen lijst
- Handleiding
- Applicatie in code vorm

De globale planning is als volgt in tekst versie verwerkt:

Week 1:	Formuleren opdrachtomschrijving, Plan van Aanpak.
Week 2:	Starten met de Inception fase. Mockup maken.
Week 3:	Starten met de Elaboration fase. Afronden van het Inception rapport.
Week 4:	Starten met de ontwikkeling van de applicatie. Mockup omzetten naar prototype.
Week 5:	Starten met de Construction fase. Het Elaboration rapport opstellen. Eerste beta-testen uitvoeren.
Week 7:	Mogelijkheid bekijken om wens naar eis om te zetten.
Week 8:	Construction fase verder uitwerken.
Week 11:	Construction fase afronden. Testen van het product en Construction rapport opstellen. Transitie fase starten.
Week 15:	Acceptatie testen van het product uitvoeren en de Transitie fase afronden.

Om tot deze producten te komen zal ik ook activiteiten uitvoeren die niet in de globale planning naar voren komen. Deze activiteiten staan in de detailplanning van het plan van aanpak en ziet er als volgt uit:

- Het plan van aanpak schrijven.
  4 werkdagen
- Onderzoek MVC framework : Frameworks onderzoeken waarin deze applicatie ontwikkeld kan worden.
  3 werkdagen

- Onderzoek: De wensen/eisen van de accountmanager en de afdelingsmanager achterhalen en hiervan een vertaling maken naar een applicatieontwerp met behulp van UML.
  7 werkdagen
- Het inception rapport schrijven.
  4 werkdagen
- Onderzoek gegevensmodel:
  - Overleg voeren met de leverancier voor het gebruik van de externe gegevens.
  - De gegevens uit de externe bronnen vergelijken met de gegevens van de interne bron.
  - Een nieuw gegevensmodel opstellen met behulp van UML.

  7 werkdagen
- Het elaboration rapport schrijven.
  6 werkdagen
- Ontwerpen van het systeemdeel voor de externe koppelingen met behulp van UML.
  6 werkdagen
- De gegevensconversie uitvoeren en de kwaliteit ervan controleren.
  5 werkdagen
- Het bouwen van de applicatie met behulp van het gekozen framework.
  Het integreren van het losse systeemdeel voor de externe koppelingen.
  9 werkdagen
- Het construction rapport schrijven.
  5 werkdagen
- Testen:
  - Opstellen van testsets.
  - Het testen van de externe koppelingen.
  - De verwerking van de gegevens binnen de applicatie testen.
  - De applicatie stresstesten.
  - De integratie testen van de systeemdelen in het framework.
  - Een acceptatietest uitvoeren.
  - De testen worden samengesteld met behulp van TMap of de IEEE 829 standaard.

  8 werkdagen
- Het transition rapport schrijven.
  3 werkdagen
- Een handleiding schrijven, implementeren applicatie en overdacht applicatie
  3 werkdagen
- Opbouwen afstudeerdossier
  15 werkdagen

3.3. Eisen duidelijk definiëren

Het verduidelijken van de opdracht in de inceptionfase heeft via een brainstorm sessie met de opdrachtgever vorm gegeven aan de wensen en eisen. Hierin kwamen punten naar voren zoals welke functionele eisen moet de applicatie hebben en welke niet functionele eisen zijn er. Er is ook besproken waarom en in welke omgeving deze applicatie gebruikt gaat worden. Dit overleg gaf mij een beeld waarom er bepaalde eisen aan de applicatie werden gesteld. De eisen waren nu duidelijk voor mij en geformuleerd aan de hand van dit informele overleg. Om deze eisen zwart op wit vast te stellen heb ik een formeel gesprek met de opdrachtgever aangevraagd. Tijdens dit overleg wilde ik van de opdrachtgever inhoudelijke feedback waarom deze eisen er zijn om zo de eisen te kunnen onderscheiden van de wensen. Dit was nodig omdat ik deze eisen moet verwerken in de planning waardoor het voor mij mogelijk is om een inschatting te maken of ik deze eisen kan verwerken in de applicatie

De eisen die naar aanleiding van deze overleggen en het opstellen van de afstudeeropdracht zijn vastgesteld zijn als volgt:

Functionele eisen voor de initiële oplevering:

- Het invoeren van klantcontact informatie.
- Het opslaan van de klantcontact informatie.
- Het tonen van deze informatie met behulp van de volgende rapportages:
    - Het totaal aantal klantcontacten afgezet tegen de wijze van binnenkomst in een zelf te bepalen periode. Op te vragen per binnenkomst of alle binnenkomst mogelijkheden.
    - Het totaal aantal klantcontacten afgezet tegen de wijze van binnenkomst per dienst gescheiden in een zelf te bepalen periode. Op te vragen per dienst of alle diensten.
    - Het totaal aantal klantcontacten per dienst afgezet tegen de soort afhandeling (intern of extern) in een zelf te bepalen periode. Op te vragen per dienst of alle diensten. Per dienst is de soort afhandeling gespecificeerd in extern of extern en wat er intern of extern gedaan is om de vraag af te handelen.
    - Het totaal aantal klantcontacten per dienst gespecificeerd naar product/onderwerp van deze dienst in een zelf te bepalen periode. Op te vragen per dienst of alle diensten.
    - Het totaal aantal klantcontacten per medewerker gespecificeerd naar binnenkomst in een zelf te bepalen periode. Op te vragen per medewerker of alle medewerkers.
    - Alle rapportages tonen de aantalen zowel absoluut als relatief.
- Het automatisch aanmaken van rapportages (een combinatie van de bovenstaande rapportages) met behulp van externe databronnen zoals de belgegevens van de medewerkers.

- Het beheren van gebruikers mogelijk maken.
- Het beheren van de binnenkomst, diensten, onderwerpen en afhandeling mogelijk maken.

Niet functionele eisen voor de initiële oplevering:

- De cliënt kant van deze applicatie moet onder Windows XP SP3 kunnen werken met Mozilla Firefox en Windows Internet Explorer 8.
- Zowel de afdeling Functioneel Beheer als een agent moet met de applicatie overweg kunnen.
- De server kant van de applicatie moet niet afhankelijk zijn van een bepaalde hardware setup. Het moet mogelijk zijn om de applicatie op een Windows 2003 Server/XP/7 en Linux Fedora[8] systeem te installeren.
- De gegevens uit de oude applicatie, vanaf 2009 tot nu, moeten worden overgezet naar de nieuwe applicatie.

De functionele eis "Het automatisch aanmaken van rapportages met behulp van externe databronnen zoals de belgegevens van de medewerkers." zorgt voor een koppeling met een databron waardoor het rapportage proces geautomatiseerd kon worden. Deze bron bleek bij navraag bij de ICT-afdeling van de gemeente niet in beheer bij de gemeente Den Haag zelf. Om toegang tot deze externe databron te krijgen moet ik in overleg gaan met de leverancier die de databron in beheer heeft.

De niet functionele eis "De gegevens uit de oude applicatie, vanaf 2009 tot nu, moeten worden overgezet naar de nieuwe applicatie" is vastgesteld omdat er een zekerheid is dat de oude data beschikbaar blijft via een databasedump. Hierdoor is het niet nodig om de data vanaf het eerste gebruik van de oude applicatie te importeren in de nieuwe applicatie. De manager gaf aan dat twee jaar genoeg historie zou geven in de nieuwe applicatie. Omdat dit een deel van 2009 zou zijn is afgesproken om vanaf 1-1-2009 de data te importeren.

Nu de eisen vorm hadden gekregen had ik de mogelijkheid om de wensen op te stellen. De opdrachtgever heeft een aantal wensen en geeft ook aan dat dit aantal zou kunnen veranderen/vermeerderen. Om te voorkomen dat er een wens omgezet wordt in een eis die later overbodig blijkt, is er afgesproken dat voor de eerste release er één wens als eis bij kan komen. De omzetting van deze wens naar eis moet voldoende gefundeerd zijn en kan pas na het beta-testen besproken worden. Hierbij is afgesproken dat het managementteam van het contact centrum het unaniem eens moet zijn dat een wens omgezet moet worden naar eis, dan pas is er voldoende onderbouwing. Dit is zo afgesproken om de planning te kunnen bewaken en het zorgt ervoor dat er uniformiteit is bij het managementteam.

[8] http://fedoraproject.org/nl/

De wensen die naar voren zijn gekomen:

- Het aantal registraties zichtbaar maken op het wall-display van Quickcom.
- Het aantal registraties ten opzichte van het aantal gesprekken real-time zichtbaar maken in het systeem zelf.
- Het koppelen van het systeem aan de inkomende gesprekken in de centrale om zo af te dwingen dat iemand zal registreren.

In de inceptionfase zal er een mockup gemaakt worden om de gebruikersinterface te beoordelen. Dit is een deel van de niet functionele eis " Zowel de afdeling Functioneel Beheer als een agent moet met de applicatie overweg kunnen."

In de elaborationfase zijn de volgende functionele eisen verwerkt om te kunnen beta-testen met een prototype:

- Het invoeren van klantcontact informatie.
- Het opslaan van de klantcontact informatie.
- Het beheren van gebruikers mogelijk maken.
- Het beheren van de binnenkomst, diensten, onderwerpen en afhandeling mogelijk maken.

In de elaborationfase kan er ook verder gekeken worden naar de niet functionele eis " Zowel de afdeling Functioneel Beheer als een agent moet met de applicatie overweg kunnen." omdat er nu daadwerkelijk functionaliteiten verwerkt zijn die te toetsen zijn door de agents en functioneel beheerders.

Als de applicatie na het beta-testen goedgekeurd is zal ik in de constructionfase de gegevensconversie uitvoeren en de rapportagemogelijkheden in de applicatie verwerken zodat ik de applicatie kan uitrollen in de acceptatieomgeving van het contact centrum. In deze fase kan ik alle functionele en niet functionele eisen die op dat moment zijn verwerkt in applicatie laten accepteren door de manager.

Als de manager akkoord gaat met de applicatie kan ik deze in de transitionfase uitrollen in de productieomgeving. Hierbij draag ik de applicatie samen met de documentatie over aan de afdeling functioneel beheer. De trainer/managers en functioneel beheerder krijgen van mij een training als dit nodig is.

3.4. Framework kiezen.

Om dit project uit te voeren wilde ik gebruik maken een bestaand framework. De redenen om voor een bestaand framework te kiezen zijn als volgt:

- Het scheelt veel ontwikkeltijd en geld.
- Bij een veel gebruikt framework zijn er al vele mensen bezig met het testen van het framework, wat de kans op kritieke fouten in het framework minder maakt.
- Veel onderdelen zijn al aanwezig zoals databasekoppelingen en authenticatie mogelijkheden.

Het framework moest aan een aantal eisen voldoen om in aanmerking te komen voor de shortlist. De eerste eis was dat het framework op basis van het Model–view–controller architectuur werkt. Dit kan ervoor zorgen dat de applicatie minder complex is en dat er meer flexibiliteit en hergebruik van de code plaatsvindt. Het framework moet meerdere gebruikers tegelijk aankunnen. Het moet werken op elk systeem, zoals beschreven in de eisen voor zowel de server als clientkant, binnen het KCC.

De keuze voor het MVC framework heb ik gemaakt na een vergelijking tussen verschillende pakketten die op de markt aanwezig zijn. De vergelijking is gemaakt via een lijst op Wikipedia[9] en PHP Zone[10]. De keuze voor de pakketten die naar voren zijn gekomen is op basis van de kennis die aanwezig is qua programmeertaal en de mogelijkheid om het framework te laten werken op de bestaande systemen bij de gemeente Den Haag. Deze keuzes zijn gemaakt om de risico's op vertraging zo veel mogelijk uit te sluiten. Door deze lijsten had ik veel keuzes en buiten ASP.NET had ik geen ervaring met deze frameworks, keuzes maken was daardoor ingewikkeld. De shortlist is gevormd door mijn ervaring met ASP.NET, de nummer 1 bij PHP Zone, de winner van een wedstrijd volgens PHP Zone en een vergelijking[11] van een ervaringsdeskundige op Internet waarin staat dat Yii lijkt op ASP.NET.

De frameworks die in de vergelijking naar voren gekomen zijn:

- ASP.NET[12]
- Symfony[13]
- Yii[14]
- Prado[15]

[9] http://en.wikipedia.org/wiki/Comparison_of_Web_application_frameworks
[10] http://www.mustap.com/phpzone_post_73_top-10-php-mvc-frameworks
[11] http://www.sheldmandu.com/php/php-mvc-frameworks/choosing-the-best-php-mvc-framework-part-1
[12] http://www.asp.net/
[13] http://www.symfony-project.org/
[14] http://www.yiiframework.com/
[15] http://www.pradosoft.com/

De keuze voor het uiteindelijke gebruikte framework heb ik in twee punten kunnen onderverdelen.

Het eerste punt betreft de kosten. Behalve ASP.NET werken alle gekozen frameworks met PHP[16]. Deze programmeertaal is vaak samen te vinden met een Apache[17] webserver en ze zijn beide gratis te verkrijgen en te gebruiken. De combinatie van deze twee is al vaak getest en veelvuldig aanwezig op het Internet. ASP.NET maakt gebruik van C#[18] en werkt goed samen met een Windows server met daarop Internet Information Services (IIS)[19]. ASP.NET kan ook samenwerken met Apache. Dit gaat met behulp van een module die geen support meer krijgt, dat brengt een risico met zich mee. Er is geen PC met Windows Server 2008 en IIS aanwezig binnen het contact centrum, wel meerdere met Windows XP. Dit zou betekenen dat er een licentie moet worden aangekocht voor Windows Server 2008 en waarschijnlijk is er ook andere hardware nodig. Het feit dat Apache en PHP gratis zijn en werken op Windows XP heeft de doorslag gegeven om alleen nog maar naar PHP frameworks te kijken. De aanschaf van een apart systeem zou geen goedkeuring krijgen wegens bezuinigingen.

Het tweede punt ging over de ondersteuning en doorontwikkeling van het framework.

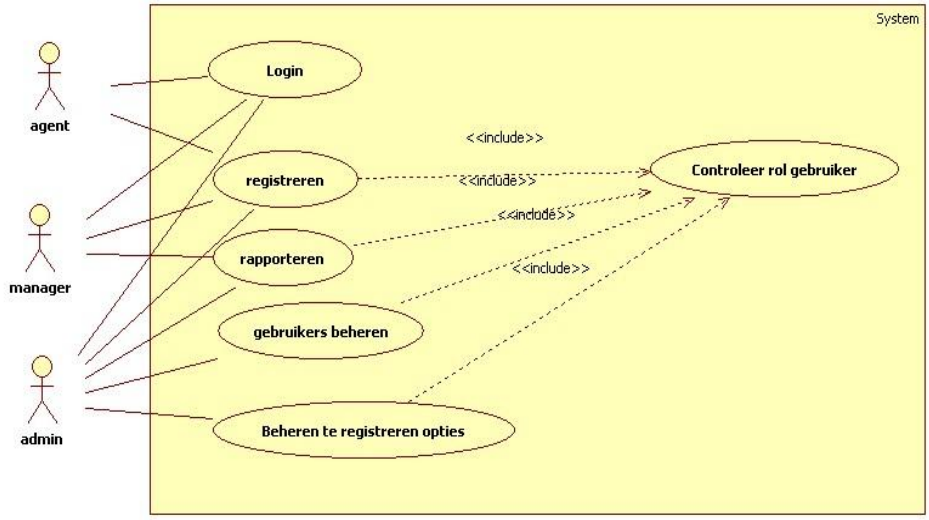
Tijdens het doornemen van de mogelijkheden kwam ik erachter dat Yii een doorontwikkeling is van Prado waarin een aantal veranderingen zijn gemaakt om de werking te versnellen. Door deze informatie was de keuze nog tussen Yii en Symfony. Beide pakketten zijn snel en stabiel en hebben zichzelf al bewezen op de markt. Dit maakt het moeilijk om een goede keuze te maken tussen deze pakketten. De keuze voor het uiteindelijke pakket is gemaakt op basis van de aanwezige documentatie en de snelheid van nieuwe releases van het pakket. Omdat er voor Yii veel documentatie aanwezig is die makkelijk te benaderen is, wat fijn werkt op het moment dat er iets onduidelijk is. De documentatie van Yii is online beschikbaar en makkelijk door te zoeken. Symfony maakt gebruikt van PDF documentatie waardoor doorverwijzen naar andere pagina ervoor zorgt dat je weer moet zoeken naar waar je begonnen was in de documentatie. De periodes waarin Yii nieuwe releases uitgeeft is constanter dan die van Symfony. Als Symfony een nieuwe release van het framework heeft dan is Yii al 3 a 4 releases verder. Dit geeft het idee dat Yii sneller kan inspelen op bugs en beveiligingsfouten. Deze punten hebben ervoor gezorgd dat mijn keuze is gevallen op Yii.

[16] http://php.net/
[17] http://www.apache.org/
[18] http://nl.wikipedia.org/wiki/C%E2%99%AF
[19] http://www.microsoft.com/windowsserver2008/en/us/internet-information-services.aspx

3.5. Vormgeven van de systeemfuncties in UML.

Om een beeld te geven wie de applicatie gebruikt en wat die gebruiker kan doen heb ik gebruik gemaakt van UML (Unified Modeling Language[20]). Het uitwerken van de aanwezig eisen en ideeën in UML geeft een grafische weergave van de applicatie. Het opstellen van de modellen was in de eerste instantie met behulp van "Praktisch UML 2de editie" van Jos Warmer & Anneke Kleppe gedaan. Tijdens een controle van de reeds aanwezige stukken bleek dit boek niet geschikt te zijn voor de huidige versie van de UML standaard. Het boek is geheel in UML 1.4 terwijl de standaard sinds 2005 al op 2.x zit. De modellen die tot dan toe waren gemaakt waren daarom niet correct. De gebruikte web-applicatie was niet toereikend om de modellen op de juiste wijze weer te geven. Omdat het contact centrum niet in het bezit is van een Rational Rose pakket, moest ik zoeken naar een applicatie welke gebruik maakte van de UML 2.0 standaard en welke gratis te gebruiken is, om zo de kosten laag te houden. Het antwoord vond ik in starUML[21]. Alle bestaande modellen zijn omgezet in starUML zodat er een geheel ontstond qua vormgeving in modellen. Dit voorkomt eventuele (overbodige) vragen die de opdrachtgever zou kunnen stellen over de verschillen in de modellen.

Er is een usecase diagram (afbeelding 3) aanwezig in het inceptionrapport met een weergave van het tot dusverre aanwezige idee. Dit diagram geeft de algemene mogelijkheden weer, die tot dan toe zijn geëist door de opdrachtgever. Buiten het usecase diagram zijn er gedetailleerde usecases per functie opgenomen in het elaboration rapport. In dit rapport staan ook een sequentie, toestandsdiagram en een domeinmodel.

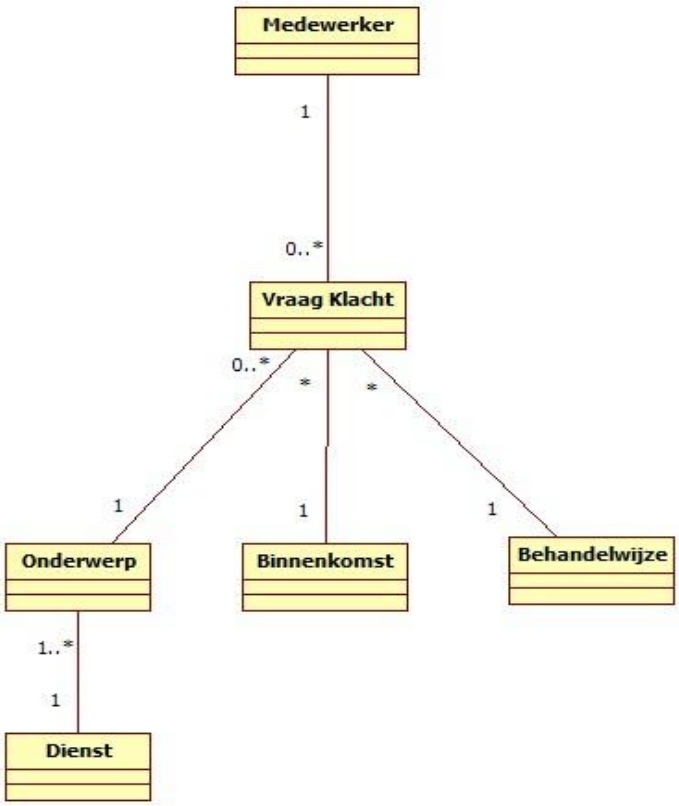


Afbeelding 3 usecase diagram uit het inceptionrapport.

[20] http://nl.wikipedia.org/wiki/Unified_Modeling_Language
[21] http://staruml.sourceforge.net/en/

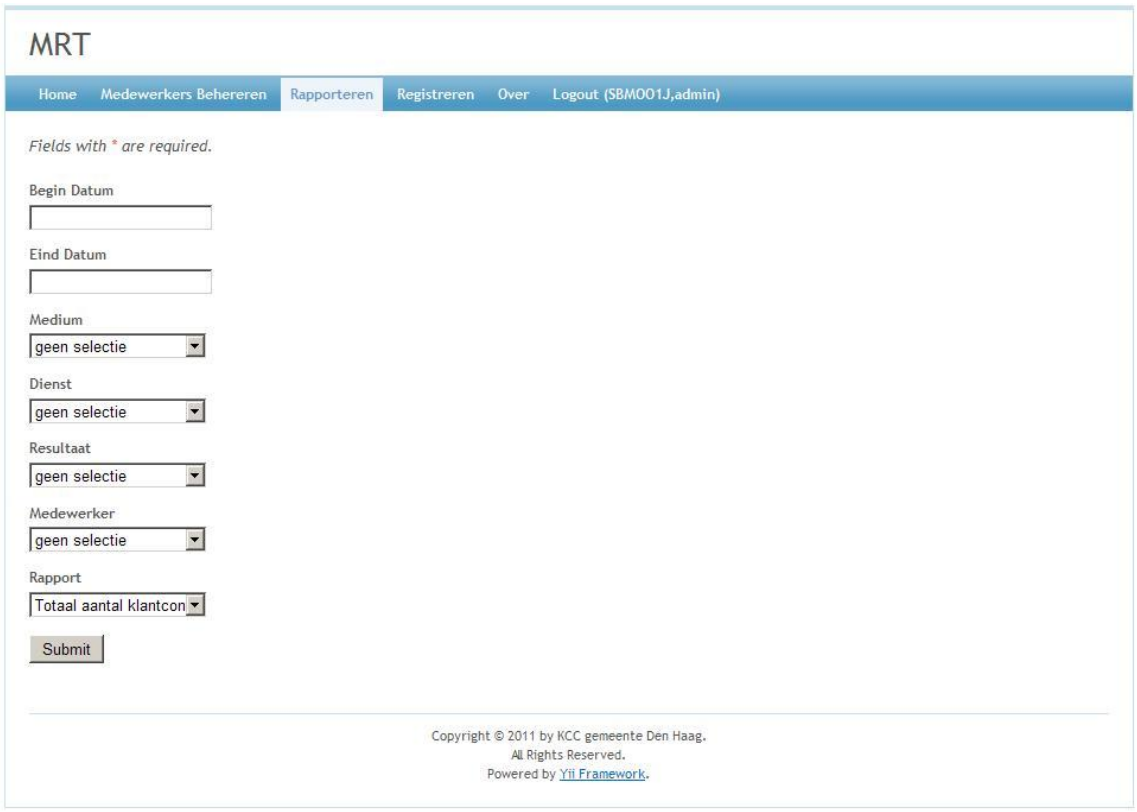
Met behulp van deze diagrammen en een mockup had de opdrachtgever de mogelijkheid om de gebruikersinterface te toetsen aan de hand van de niet functionele eis " Zowel de afdeling Functioneel Beheer als een agent moet met de applicatie overweg kunnen.". In dit stadium was het belangrijk om te weten of de gebruikersinterface zou voldoen. Bij een no go zou ik met de opdrachtgever om de tafel moeten gaan om te kijken wat er te doen was om er toch een go van te kunnen maken. Dit zou een rimpeleffect in de planning geven en er voor zorgen dat ik krap in de tijd zou komen te zitten. Door de opdrachtgever actief te betrekken bij het ontwerpen van de gebruikersinterface en de functies die de applicatie moet krijgen, begreep ik de opdrachtgever beter en wilde ik voorkomen dat ik een verkeerd beeld van de applicatie zou schetsen. Mocht het dan nog voorkomen dat de opdrachtgever niet tevreden is met de gebruikersinterface, was het altijd mogelijk om de teammanagers om een mening te vragen. Het MT zou dan de doorslag moeten geven of de gebruikersinterface juist is.

De opdrachtgever was tevreden met de tot dan toe opgeleverde producten, zowel de gebruikersinterface mockup, als de grafische vormgeving in UML. Dit had als gevolg dat de opdracht zelf uitgewerkt kon worden tot prototype.

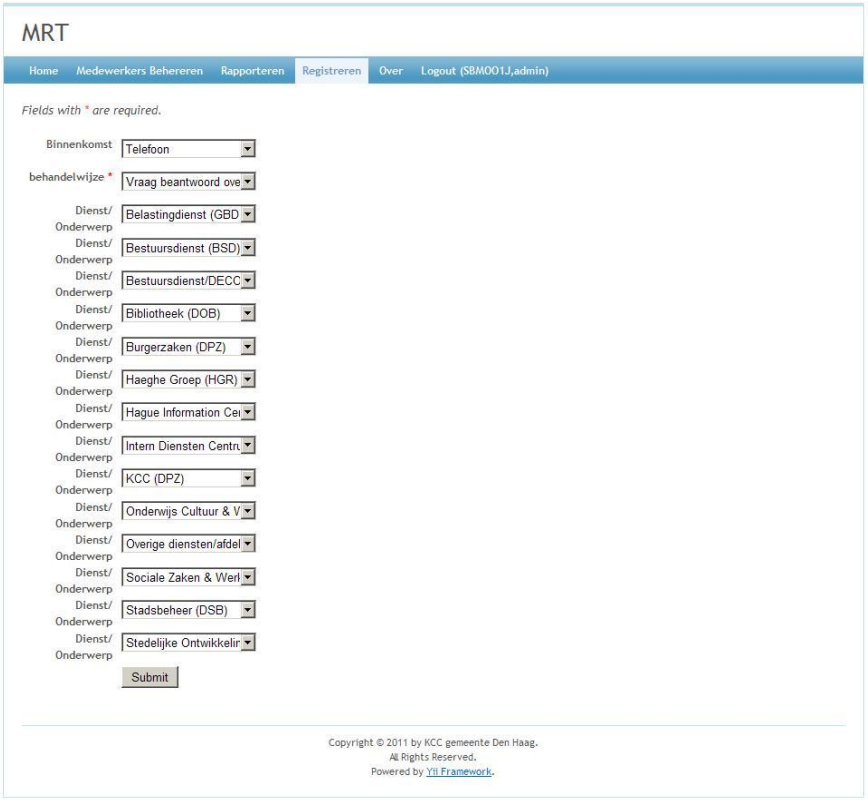


Afbeelding 4 Eerste vereenvoudigde versie klassendiagram

Afbeelding 5 Mockup van het begin scherm



Afbeelding 6 Mockup van het rapporteer scherm.



Afbeelding 7 Mockup van het registreer scherm.

De opbouw van het registreerformulier heb ik gemaakt met behulp van feedback van gebruikers over de huidige applicatie. Deze geeft alle mogelijke onderwerpen weer, wat op dit moment 86 stuks zijn. Dit zorgt ervoor dat de schermen waar de gebruikers mee werken vol staan met informatie en de submit knop onderin alleen via scrollen te bereiken is. Dit word als storend ervaren en als teveel informatie in een keer. Voor gebruikers die er al een tijd mee werken is dit een gewenning worden. Bij nieuwe gebruikers merkt de trainer dat ze schrikken van de hoeveelheid tekst op het scherm. Om te zorgen dat agents makkelijk overweg kunnen met het registratiescherm heb ik gekozen voor een lijst per dienst waarin de onderwerpen van die dienst zijn opgenomen. Dit zorgt voor een rustiger beeld dan bij de huidige applicatie.

4. Ontwerpen van de applicatie

De ruwe versie van de applicatie is geschetst met behulp van het definiëren van de eisen en door dit te verwerken in UML en een mockup. Dit kan na goedkeuring verder uitgewerkt worden tot een prototype zodat de opdrachtgever beta-testen kan uitvoeren. Voordat de applicatie gebouwd kan worden, dienen alle onderdelen die samenhangen in deze applicatie in kaart gebracht te worden, waarna een gegevensmodel opgesteld kan worden. Als er een goede basis is voor de applicatie, is het mogelijk om het geraamte neer te zetten voor de beta-testen.

4.1. Gegevensmodel opstellen

De applicatie zal verschillende data en databronnen gebruiken om te kunnen functioneren. De data die direct beschikbaar is, zal de data zijn die de applicatie verkrijgt door de invoer van de agents van het contact centrum. Deze data zal ook het grootste gedeelte innemen bij alle rapportages en zal daarom ook vaak opgevraagd worden. Deze data hangt samen met de eis " Het tonen van de klantcontact informatie met behulp van de volgende rapportages ".

De applicatie gaat ook gebruik maken van externe databronnen om de functionele eis “Het automatisch aanmaken van rapportages met behulp van externe databronnen zoals de belgegevens van de medewerkers.” te laten functioneren.

Omdat de externe databronnen nog ontsloten moesten worden, is de focus eerst naar de data gegaan die door de agents aangemaakt gaat worden. Deze data bevat de volgende klantcontact kenmerken van het contact centrum:

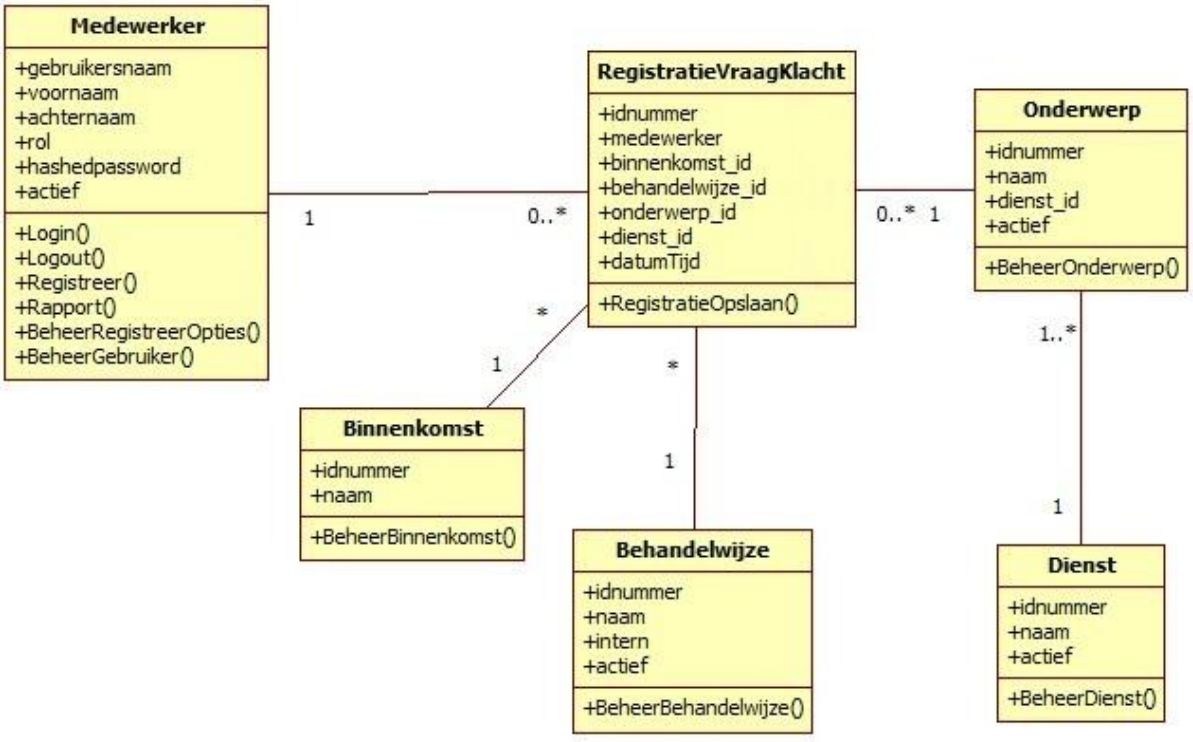
- de manier van binnenkomst
- welke dienst het klantcontact betreft
- over welk onderwerp het klantcontact ging
- de behandelwijze van dit klantcontact
- welke medewerker het klantcontact heeft afgehandeld

Door alleen met deze kenmerken te werken is het moeilijk om een database op te zetten die goed te bevragen is. Het worden allemaal losse rijen met veel redundante gegevens. Er is (praktisch) geen mogelijkheid om unieke klantcontact registraties te filteren voor een rapportage. Omdat deze applicatie afhankelijk is van invoer voor de registraties en van uitvoer voor de rapportages, is het een noodzaak dat er een goed gegevensmodel ontworpen is. Dit gegevensmodel moet zorgen voor een juiste structuur in de database. Deze structuur is voor de ondersteuning in de applicatie belangrijk. Het voorkomt

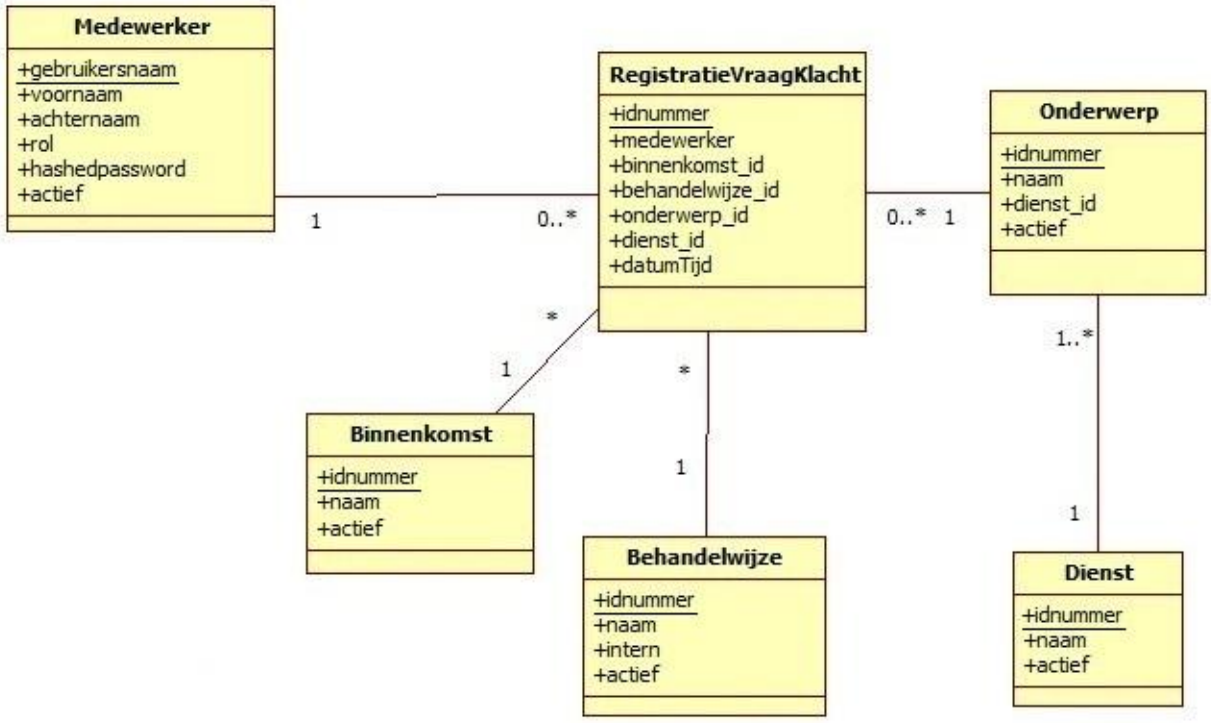
redundantie in de data en maakt de bevraging van de database met SQL makkelijker en sneller.

Het gegevensmodel bestaat uit het vormgeven in UML van een conceptueel klassendiagram. Dit model vormt de basis voor het ERD (entity-relationship diagram). Dit model valt samen met het relationeel representatiemodel, welke weer de basis vormt voor het relationeel implementatiemodel. Met deze modellen is het mogelijk om een database aan te maken. Deze database kan ik gebruiken in het prototype. Door de data in het prototype te verwerken wil ik het mogelijk maken om de snelheid van de combinatie applicatie en database te testen. Hierna kan ik eventuele aanpassingen, waar nodig, maken in/aan de database voordat de applicatie naar de acceptatieomgeving overgezet gaat worden.

De UML modellen en het relationeel representatiemodel zien er als volgt uit.



Afbeelding 8 Conceptueel klassendiagram voor de basis van het ERD (entity-relationship diagram)



Afbeelding 9 entity-relationship diagram

Relationeel representatie model

RegistratieVraagKlacht (IDnummer, *medewerker, binnenkomst_id*, *behandelwijze_id*, *onderwerp_id, dienst_id,* datumTijd)

Medewerker (gebruikersnaam, voornaam, achternaam, rol, HashedPassword, actief)

Binnenkomst (IDnummer, kanaal, actief)

Behandelwijze (IDnummer, naam, intern, actief)

Onderwerp (IDnummer, naam, *dienst_id*, actief)

Dienst (IDnummer, naam, actief)

RegistratieVraagKlacht:
IDnummer is de primaire sleutel, deze mag niet null zijn.
medewerker is een vreemde sleutel die verwijst naar Medewerker.gebruikersnaam, deze mag niet null zijn.
binnenkomst_id is een vreemde sleutel die verwijst naar Binnenkomst.IDnummer, deze mag niet null zijn.
behandelwijze_id is een vreemde sleutel die verwijst naar Behandelwijze.IDnummer, deze mag niet null zijn.

onderwerp_id is een vreemde sleutel die verwijst naar Onderwerp.IDnummer, deze mag niet null zijn.
dienst_id is een vreemde sleutel die verwijst naar Dienst.IDnummer, deze mag niet null zijn.
datumTijd mag niet null zijn.

Medewerker:
gebruikersnaam is de primaire sleutel, deze mag niet null zijn.
voornaam mag null zijn.
achternaam mag null zijn.
rol deze mag niet null zijn.
hashedpassword mag null zijn.
actief mag niet null zijn.

Binnenkomst:
IDnummer is de primaire sleutel, deze mag niet null zijn.
kanaal mag niet null zijn.
actief mag niet null zijn.

Behandelwijze:
IDnummer is de primaire sleutel, deze mag niet null zijn.
naam mag niet null zijn.
intern mag niet null zijn.
actief mag niet null zijn.

Onderwerp:
IDnummer is de primaire sleutel, deze mag niet null zijn.
naam mag niet null zijn.
dienst_id is een vreemde sleutel die verwijst naar Dienst.IDnummer, deze mag niet null zijn.
actief mag niet null zijn.

Dienst:
IDnummer is de primaire sleutel, deze mag niet null zijn.
naam mag niet null zijn.
actief mag niet null zijn.

Afbeelding 10 relationeel representatiemodel

De gegevens van klantcontact bestaan zoals eerder aangegeven uit:

- de manier van binnenkomst
- welke dienst het klantcontact betreft
- over welk onderwerp het klantcontact ging
- de behandelwijze van dit klantcontact
- welke medewerker het klantcontact heeft afgehandeld

Al deze gegevens komen meerdere malen terug in verschillende combinaties. Per dag zijn er gemiddeld 1700 registraties. De kans dat een bepaalde registratie dezelfde kenmerken heeft als een eerdere registratie is groot. Omdat dit niet

wenselijk is, moet er een unieke waarde zijn voor elke registratie. Mede daarom heb ik een tabel aangemaakt (RegistratieVraagKlacht) met een IDnummer als uniek nummer welke dan ook als primaire sleutel zal fungeren. Voor de rapportages is het belangrijk om te weten wanneer de registratie plaats heeft gevonden. Het data-element datumTijd zorgt hiervoor. De data die verder in de tabel RegistratieVraagKlacht komt, heeft verschillende variaties, maar zal altijd meerdere keren per dag gebruikt worden. Omdat dit met 1700 registraties per dag flink kan oplopen, zijn de gegevens per stuk in aparte tabellen opgenomen. Dit voorkomt redundantie en maakt de database efficiënter. Nu zal in de grootste tabel gewerkt worden met cijfers in plaats van lange teksten. Bijkomend voordeel is dat de afdeling functioneel beheer de opties aan en uit kan zetten door actief true of false als waarde mee te geven. De database is aangemaakt in de derde normaal vorm, er bestaan geen transitieve afhankelijkheden binnen een losse tabel.

Ik heb ervoor gekozen om in de tabel RegistratieVraagKlacht de medewerker op te nemen aan de hand van zijn/haar gebruikersnaam in plaats van een idnummer omdat deze gebruikersnaam binnen de gemeente uniek is. Deze waarde komt in elke applicatie terug als uniek kenmerk. De CIC gebruikt deze waarde als loginnaam en WFM Verint ook. Dit zal de gemene deler zijn in de verschillende databronnen. Aan de hand van deze modellen zal ik het relationeel implementatiemodel opstellen.

Het opstellen van een ERD kan met kennis over de gevraagde data worden gedaan, zonder restricties op te leggen aan het gebruikte databasemanagementsysteem (DBMS). Hetzelfde gaat op voor het relationeel representatiemodel. Om te voorkomen dat het relationeel implementatiemodel te algemeen is, heb ik samen met de manager een DBMS gekozen om zo het relationeel implementatiemodel toe te kunnen spitsen op dat model. Dit betekende dat de manager betrokken was bij de keuze voor een DBMS en daardoor inspraak had en een oordeel kon vellen aan de hand van de informatie die ik hem gaf. Er ontstond daardoor een combinatie van de inhoudelijke structurele kennis van de applicatie en de belangen van het bedrijf, welke een keuze zou moeten opleveren waar beide partijen mee uit de voeten kunnen.

Welke criteria waren belangrijk voor het contact centrum om een DBMS te kiezen? Na overleg met de manager ben ik op de volgende punten uitgekomen:

1. Het DBMS moet gratis zijn (in het kader van de bezuinigingen).
2. Het DBMS moet standaard kunnen samenwerken met Yii.
3. Het DBMS moet op Linux (fedora) en Windows (XP SP3 en 7) werken.
4. Het DBMS moet voldoende mogelijkheden hebben om te groeien.

Criteria 2 en 3 zijn mijn eigen inbreng. Als het DBMS hieraan voldoet kan ik de planning aanhouden zoals deze is. Het scheelt tijd als iets in een uur werkt in plaats van zelf een database connector te moeten maken in het framework.

Vanwege de keuze voor Yii en criterium 2 komen er een aantal DBMS-en naar voren als kandidaat uit de documentatie[22] van Yii.

- SQLite[23]
- MySQL[24]
- PostgreSQL[25]
- MS SQL Server[26]
- Oracle[27]

De standaard binnen de gemeente Den Haag is Oracle. Dit pakket is helaas niet gratis en voldoet daarbij dus niet aan punt 1. Dit pakket valt daarom af gelijk af.

Door de documentatie van de andere pakketten te lezen was het mogelijk om een geschikte kandidaat te vinden.

SQLite is gratis en op verschillende systemen te gebruiken. Hierdoor voldoet het aan alle punten. SQLite is geeft duidelijk informatie als het gaat om wanneer je SQLite niet moet gebruiken. In de documentatie " Appropriate Uses For SQLite"[28] staan de volgende regels: " If you have many client programs accessing a common database over a network, you should consider using a client/server database engine instead of SQLite" en " SQLite uses reader/writer locks on the entire database file.". Als het alleen zou gaan om "many client programs" dan had er nog een overweging kunnen zijn om dit pakket te gebruiken gezien de compactheid van SQLite en de hoeveelheid "many" goed te testen is. Waar het contact centrum dan wel rekening mee dient te houden is dat de applicatie een beperking krijgt voor het aantal simultane gebruikers. Helaas is het blokkeren van het databasebestand niet gewenst. Hoe klein die kans ook mag zijn volgens de documentatie, het is een risico waar je niet mee te maken zou willen hebben. Dit was geen criteria op de lijst, maar dit is een punt dat voor zich spreekt bij het aantal medewerkers dat het contact centrum heeft.

MySQL is volgens hun eigen site "The world's most popular open source database". Dit komt waarschijnlijk door de koppeling van MySQL aan Apache in bijvoorbeeld het webserver pakket XAMPP[29]. Voor niet commercieel gebruik is

---

22 http://www.yiiframework.com/doc/guide/1.1/en/database.dao
23 http://www.sqlite.org/
24 http://www.mysql.com/
25 http://www.postgresql.org/
26 http://www.microsoft.com/netherlands/sqlserver/default.aspx
27 http://www.oracle.com/nl/products/database/index.html
28 http://www.sqlite.org/whentouse.html
29 http://en.wikipedia.org/wiki/XAMPP

MySQL gratis, maar deze applicatie gaat ingezet worden voor commercieel gebruik. Om MySQL dan te mogen gebruiken moet er minimaal $2000,- per jaar betaald worden. Het is een populair pakket maar helaas niet gratis in gebruik waardoor het niet voldoet aan punt 1. Hierdoor valt dit pakket ook af.

Het PostgreSQL pakket voldoet volgens de documentatie aan alle punten.
Er zijn geen beperkingen qua hardware opgenomen in de documentatie en de beperkingen voor de database zijn dermate ruim, één tabel mag maximaal 32 TB groot worden, waardoor dit pakket nog lang gebruikt kan worden.

MS SQL Server werkt alleen samen met een Windows installatie. Hierdoor voldoet het pakket niet aan punt 3. Tevens is er een beperking voor de gebruikte hardware en de grootte van de gebruikte database in de gratis versie, de express editie. Dit zou ervoor kunnen zorgen dat er later alsnog betaald moet gaan worden voor de standard editie.

PostgreSQL voldoet aan alle punten en zal daarom het DBMS worden van deze applicatie. Het relationeel implementatiemodel is opgesteld aan de hand van de specificaties die PostgreSQL gebruikt. Afbeelding 12 geeft het relationeel implementatiemodel van de database weer.

Relationeel implementatie model

```
CREATE TABLE RegistratieVraagKlacht (
  IDnummer SERIAL NOT NULL,
  medewerker character varying(7) NOT NULL,
  binnenkomst_id smallint NOT NULL,
  behandelwijze_id smallint NOT NULL,
  onderwerp_id smallint NOT NULL,
  dienst_id smallint NOT NULL,
  datumTijd timestamp without time zone NOT NULL,

        PRIMARY KEY (IDnummer),

        FOREIGN KEY (medewerker) REFERENCES Medewerker (gebruikersnaam)
        ON UPDATE RESTRICT ON DELETE RESTRICT,

        FOREIGN KEY (binnenkomst_id) REFERENCES Binnenkomst (IDnummer)
        ON UPDATE RESTRICT ON DELETE RESTRICT,

        FOREIGN KEY (behandelwijze_id) REFERENCES Behandelwijze (IDnummer)
        ON UPDATE RESTRICT ON DELETE RESTRICT,

        FOREIGN KEY (onderwerp_id) REFERENCES Onderwerp (IDnummer)
        ON UPDATE RESTRICT ON DELETE RESTRICT,

        FOREIGN KEY (dienst_id) REFERENCES Dienst (IDnummer)
```

```
        ON UPDATE RESTRICT ON DELETE RESTRICT
)
CREATE TABLE medewerker (
        gebruikersnaam character varying(7) NOT NULL,
        voornaam character varying (128) DEFAULT NULL,
        achternaam character varying (128) DEFAULT NULL,
        rol character varying (10) NOT NULL DEFAULT 'gebruiker',
        hashedpassword character varying (128) NOT NULL,
        actief boolean NOT NULL,

        PRIMARY KEY (gebruikersnaam)
)
CREATE TABLE Binnenkomst(
        IDnummer SERIAL NOT NULL,
        naam character varying(128) NOT NULL,
        actief boolean NOT NULL,

        PRIMARY KEY (IDnummer)
)
CREATE TABLE Behandelwijze(
        IDnummer SERIAL NOT NULL,
        naam character varying(128) NOT NULL,
        intern boolean NOT NULL,
        actief boolean NOT NULL,

        PRIMARY KEY (IDnummer)
)

CREATE TABLE Onderwerp(
        IDnummer SERIAL NOT NULL,
        naam character varying(128) NOT NULL,
        dienst_id smallint NOT NULL,
        actief boolean NOT NULL,

        PRIMARY KEY (IDnummer),
        FOREIGN KEY (dienst_id) REFERENCES Dienst (IDnummer)
        ON UPDATE RESTRICT ON DELETE RESTRICT
)

CREATE TABLE Dienst(
        IDnummer SERIAL NOT NULL,
        naam character varying(128) NOT NULL,
        actief boolean NOT NULL,

        PRIMARY KEY (IDnummer)
)
```

Afbeelding 11 relationeel implementatiemodel

De externe bronnen moeten ook aansluiting vinden in de applicatie. Deze bronnen zijn in het beheer van twee verschillende partijen die beide een contract bij de gemeente Den Haag hebben voor de applicaties die gebruik maken van deze bronnen. In het contract staat onder andere dat zij eigenaar zijn van de databron en deze ook in beheer zullen hebben. De data die aanwezig is in deze bronnen blijft natuurlijk van het contact centrum, alleen de toegang ertoe gaat via de leverancier. De bedoeling is dat de applicatie direct toegang zal krijgen tot deze bronnen, dit om te voorkomen dat de leverancier ingeschakeld moet worden bij veranderingen in deze applicatie. De andere kant daarvan is dat de leverancier het contact centrum moet inschakelen op het moment dat zij iets veranderen aan de applicatie/datastructuur.

Om toegang te krijgen tot deze bronnen heb ik contact opgenomen met de leveranciers in beheer van de bronnen te weten: "Teletrain" en "Newtel". Omdat het niet mogelijk is om beide partijen tegelijk om de tafel te krijgen, zal ik per leverancier mijn verzoek indienen. Dit zorgt ervoor dat het langer duurt om de externe bronnen te ontsluiten.

Via een contactpersoon bij Newtel kreeg ik de benodigde gegevens om in de werkende productie-omgeving in te loggen op de Oracle database die Newtel gebruikt. Deze gegevens mag ik echter niet gebruiken totdat ik in de testomgeving kan aantonen dat er gedegen koppeling is naar de database. Om dit te kunnen realiseren heb ik de beheerder bij Newtel verzocht om een dump van de database aan te leveren zodat ik deze kan nabouwen en testen op kan uitvoeren.

Teletrain heeft een officieel verzoek via het ticketsysteem van Teletrain gekregen. Er kwamen enkele vragen van Teletrain zoals:"Wil je ook data kunnen wegschrijven naar de database?" en "Kunnen wij een akkoord krijgen van de manager dat jij met deze gegevens aan de slag gaat?". Het wegschrijven van data zou niet nodig zijn in de database gezien er voor rapportages alleen cijfers nodig zijn die zij kunnen aanleveren. Nadat de manager zijn akkoord had gegeven kreeg ik toestemming om een koppeling te maken met de MSSQL database van Teletrain. Bij het aanvragen van de toeganggegevens bleek dat ik wederom met een dump van de database moet aantonen dat ik een gedegen koppeling kan bouwen voordat ik toegang kan krijgen tot de live-data.

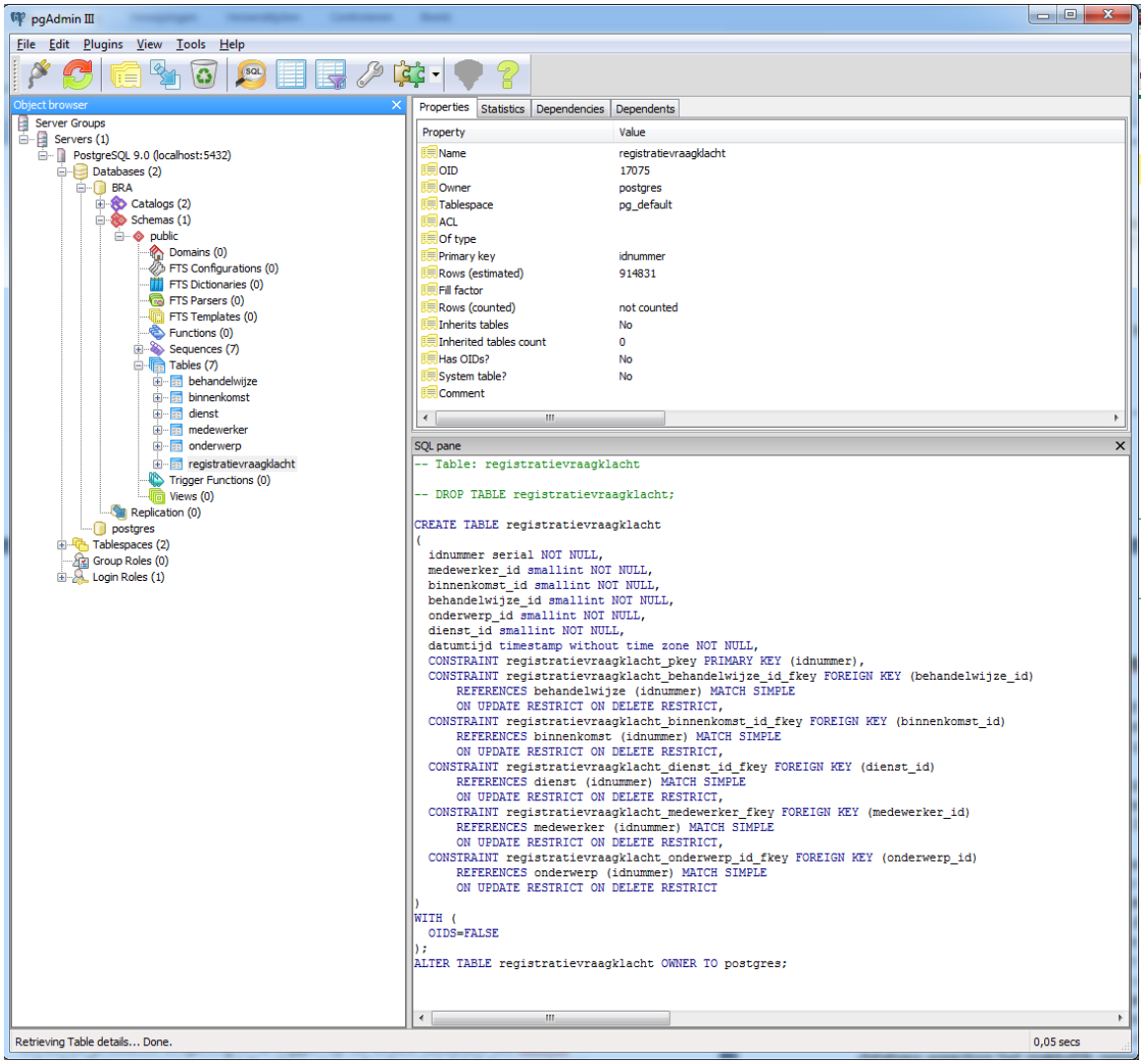
Bij beide leveranciers heb ik aangegeven dat ik alleen leesrechten nodig zou hebben bij deze koppeling, maar dit was de voorwaarde om überhaupt toegang te krijgen tot de databases. Omdat dit traject langer in beslag zou nemen dan verwacht heb ik in overleg met de manager dit deel van de opdracht in de wacht gezet.

4.2. Prototype bouwen en beta-testen

Met het kiezen van een DBMS werd het mogelijk om een prototype te bouwen, waarna dit prototype overgezet zou kunnen worden van de ontwikkelomgeving naar de testomgeving van het contact centrum.

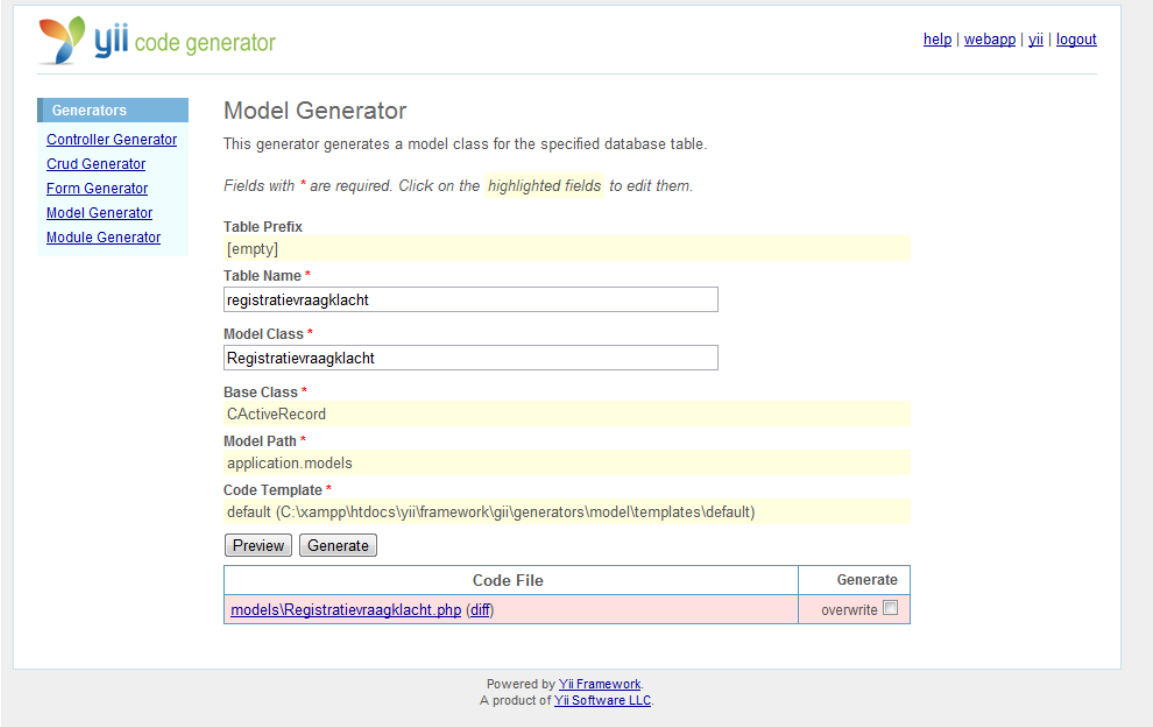
Het prototype kan de opdrachtgever inzicht in de applicatie geven, de gebruikers (managers, agents en beheerders) kunnen beta-testen met de applicatie en ik kan testen of de applicatie werkt zoals beschreven is.

Het bouwen van het prototype is in fases gegaan. Als eerste heb ik de database aangemaakt in PostgreSQL met behulp pgAdmin III en het relationeel implementatiemodel zodat de tabellen in Yii te gebruiken zijn.



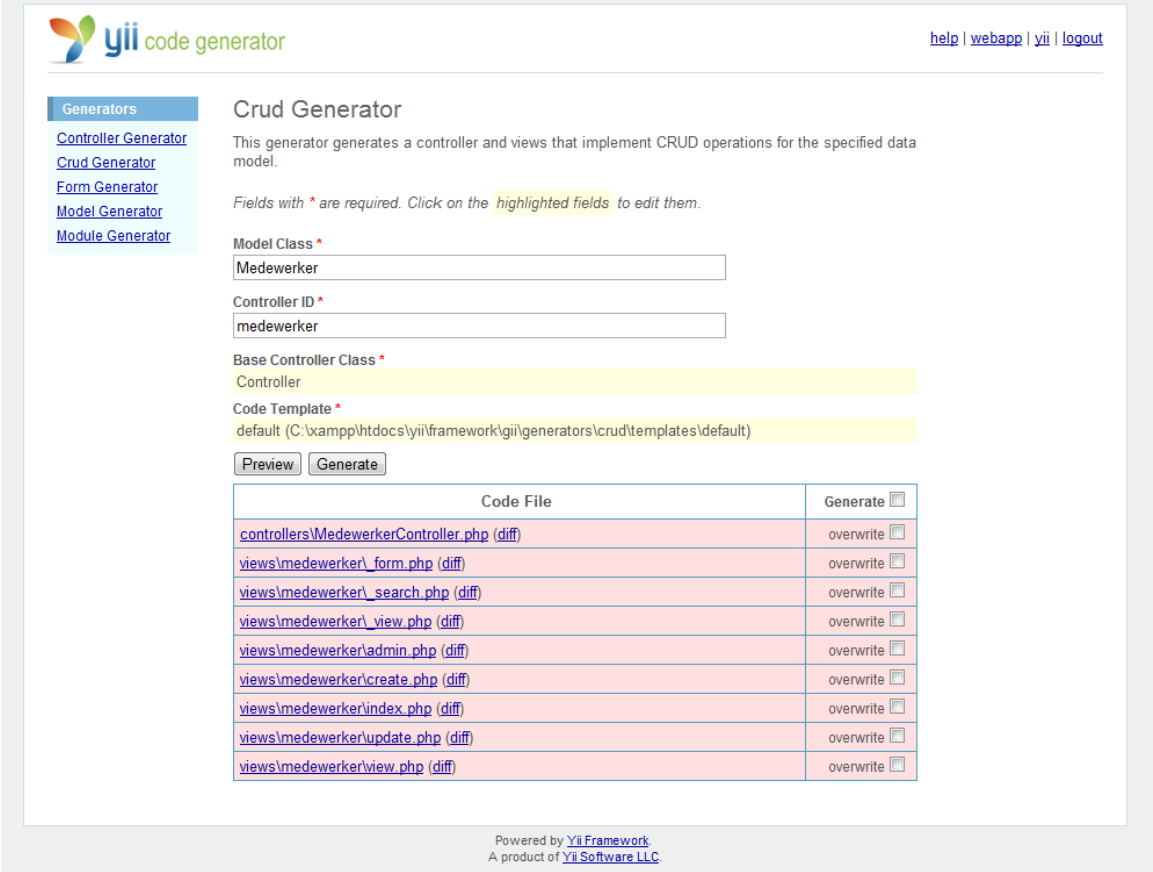
Afbeelding 12 Overzicht van de tabel RegistratieVraagKlacht in pgAdmin III

Yii heeft een functie geïmplementeerd, Gii, waarmee je via de webbrowser op basis van de database PHP code kan laten genereren welke te gebruiken is in Yii (zie afbeelding 13 en 14).



afbeelding 13 De model generator in de Gii functie

Deze functie heeft onder andere een Model en Crud Generator voor een database. Deze generatoren maken direct een verbinding met de gebruikte database om zo te controleren of de gebruikte tabel aanwezig is. Door gebruik te maken van deze generatoren is het zeker dat Yii kan werken met de tabellen die in gebruik zijn in de database. Alle kolommen die gebruikt worden staan gedeclareerd in de PHP code met de kenmerken uit de database waardoor het makkelijk gemaakt is om de invoer in de applicatie te vergelijken met het gewenste formaat in de database.



afbeelding 14 de CRUD generator in de Gii functie

Zoals op de afbeeldingen te zien is, bestaan de bestanden al in de applicatie en zijn ze anders dan hoe de Gii functie ze aanmaakt. Voor de niet functionele eis "Zowel de afdeling Functioneel Beheer als een agent kunnen met de applicatie overweg." heb ik een aantal aanpassingen doorgevoerd in de teksten van de applicatie en functies in de modellen, controllers en views. De aanpassingen zorgen ervoor dat er zo min mogelijk zelf tekst kan worden getypt tijdens het aanmaken van nieuwe medewerkers. Voor het registreren was het mogelijk om alle invoer te baseren op een lijst waar de agent een keuze kan maken uit de actieve mogelijkheden die de database bevat. Deze lijsten zullen zichzelf vullen.

Met alle tabellen aanwezig in de applicatie via Gii is het mogelijk om een inlogpagina te maken die gebruik maakt van waardes in de database. Het contact centrum wil graag zelf de controle hebben over de applicatie met zo min mogelijk technische en financiële afhankelijkheden van andere afdelingen. Daarom zal Yii de authenticatie van de gebruikers controleren in plaats van dit te laten doen via een centraal punt in de gemeente. De gebruikers krijgen een wachtwoord toegewezen van de afdeling functioneel beheer en deze zal met een SHA1 versleuteling [30] opgeslagen worden in de database. Het vergelijken van het

[30] http://nl.wikipedia.org/wiki/SHA-familie

wachtwoord zal binnen de applicatie gebeuren waardoor er alleen een versleuteling van het wachtwoord over het netwerk zal worden verzonden. Er komen geen regels/procedures om het wachtwoord om de zoveel tijd te veranderen. De lengte van het wachtwoord moet wel minimaal 4 karakters zijn. De inlogtijd voor de applicatie is zolang de sessie duurt. Dit zijn wensen van de manager die naar voren kwamen tijdens het beta-testen. Daar is de applicatie dan ook op aangepast omdat deze wensen geen grote impact hadden op de planning. Dit valt samen met de niet functionele eis "Zowel de afdeling Functioneel Beheer als een agent kunnen met de applicatie overweg."

Met het inloggen en koppelen aan de database is het ook mogelijk om de rechten in de applicatie te integreren. Ik heb gekozen voor een dynamisch menu bovenaan in de applicatie. Dit menu is standaard voorzien van een “Home”, “Over” en “Login” knop. Bij het inloggen is er een controle in de applicatie die ervoor zorgt dat de juiste menuknoppen erbij komen. Afhankelijk van de rol zijn dat de volgende knoppen:

- Medewerkers beheren: zichtbaar voor admin
- Opties beheren: zichtbaar voor admin
- Rapporteren: zichtbaar voor admin en manager
- Registreren: zichtbaar voor admin, manager en agent

De ontwikkelomgeving voor het bouwen van de applicatie komt sterk overeen met het systeem waar de applicatie op zou gaan werken. De weergave aan de cliëntkant en werking daarvan kan ik daarom goed op het ontwikkelsysteem bekijken en testen. De testomgeving van de gemeente Den Haag is helaas anders, deze werkt op een versie van Windows Server 2003 waardoor ik rekening moet houden met verschillen in de werking op de test pc en het daadwerkelijke doelsysteem van de applicatie. Dit kan ik in kaart brengen op het moment dat de applicatie de acceptatietest heeft doorstaan en op het systeem geïnstalleerd staat van het contact centrum. Deze situatie kan ik niet vermijden. De manager is hiervan op de hoogte en gaf aan dat deze situatie al bestaat zolang als de oude applicatie bestaat. Hij zal er rekening mee houden dat de gebruikersacceptatietest op het acceptatiesysteem een ander beeld kan geven dan een gebruikersacceptatietest in de productiesituatie bij het contact centrum. Mocht dit beeld 180 graden draaien dan moet er een oplossing worden bedacht door de afdeling functioneel beheer DPZ. Mijn oplossing voor dit moment is de applicatie thuis ook te testen op een systeem dat grotendeels overeenkomt met het productiesysteem. Werkt de applicatie niet naar behoren op het productiesysteem dan krijgt de afdeling functioneel beheer DPZ de vrijheid om via het IDC een nieuw systeem aan te vragen welke zij naar wens kunnen inrichten. Dit is geen ideale oplossing, maar wel een werkbare waar de manager akkoord mee is gegaan.

Naast de OTAP[31] straat van de gemeente zal ik de applicatie ook op mijn eigen andere systemen testen en laten draaien om zoveel mogelijk systemen te beta-testen. Deze systemen hebben de volgende besturingssystemen geïnstalleerd staan: Windows XP SP1 en SP3 32 Bit, Windows 7 SP1 64 Bit en Linux BusyBox[32].

In de testomgeving zou moeten blijken of de applicatie werkt zoals deze ontworpen is. In deze omgeving was het helaas nog niet mogelijk om met meerdere cliëntsystemen tegelijk te testen. Dit was voorbehouden aan de acceptatieomgeving binnen de OTAP straat van de gemeente.

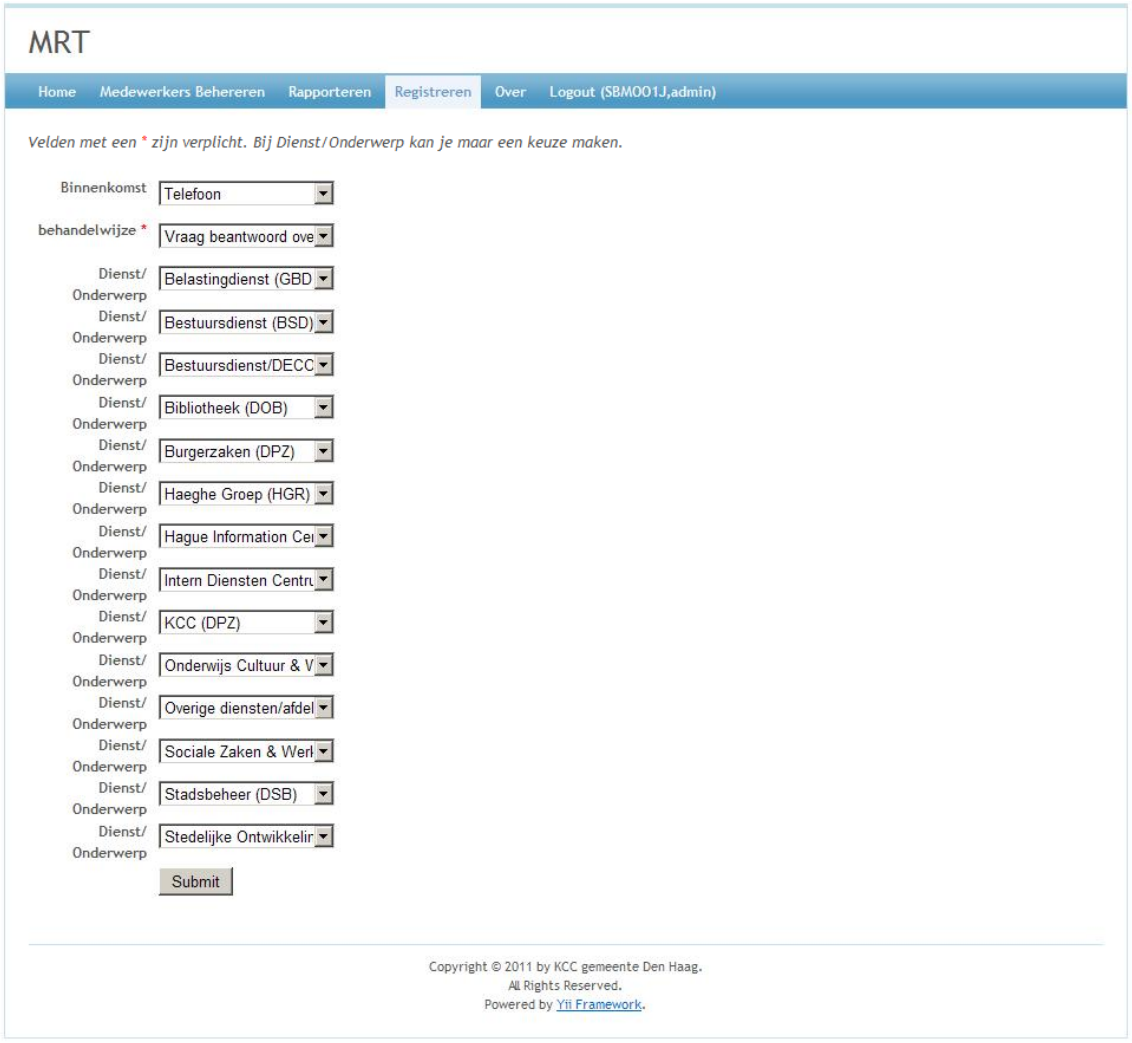
De inlogfunctie in het prototype was geen eis maar wel belangrijk om de volgende eisen te kunnen verwerken:

- Het invoeren van klantcontact informatie.
- Het opslaan van de klantcontact informatie.
- Het beheren van gebruikers mogelijk maken.
- Het beheren van de binnenkomst, diensten, onderwerpen en afhandeling mogelijk maken.

Het menu maakt deze eisen zichtbaar, de views waar deze eisen in verwerkt zijn, werken nog niet allemaal. Er zijn voorbeelden van de rapportagemogelijkheid zichtbaar zonder logica er aan vast, deze voorbeelden komen voort uit de mockup. De bovenstaande eisen zijn wel verwerkt in dit prototype. Voor het beheren van de gebruikers en opties maak ik gebruik van de Gii CRUD generator waar ik aanpassingen in verwerk. Deze aanpassingen variëren van toegangscontrole tot het aanpassen van de teksten zodat ze in het Nederlands worden weergegeven.

De opbouw van het registreerformulier had ik al gemaakt voor de mockup en heeft nu ook de logica erachter die ervoor zorgt dat er een registratie gemaakt kan worden.

[31] http://nl.wikipedia.org/wiki/Ontwikkeling,_test,_acceptatie_en_productie
[32] http://www.busybox.net/



Afbeelding 15 Weergave registreerscherm prototype

Het installeren en opstarten van de applicatie op de testomgeving kan gestart worden. Er zijn een paar punten die ik moet afwerken om het geheel te laten werken. Er moet een database aangemaakt worden en Xampp moet gebruiksklaar gemaakt worden. De database is in de ontwikkelomgeving aangemaakt en een back-up daarvan zal dienen als testdatabase in de testomgeving. Hiermee verzeker ik de opdrachtgever en mijzelf ervan dat de twee omgevingen gelijk lopen qua database en applicatie.

Omdat er geen registratiedata in de database aanwezig was, is deze eerst aangemaakt om te kunnen testen of de registratiemogelijkheid correct werkt. Tijdens het testen heeft de manager een paar wensen kenbaar gemaakt met betrekking tot het inloggen en uitloggen welke omgezet zijn in eisen zoals eerder vermeld vanwege de lage impact op de planning. Het beta-testen heeft een aantal fouten in de applicatie aan het licht gebracht in de applicatie zoals het omzeilen van de beveiliging door een directe link te gebruiken. Een andere fout

was tekstueel. Sommige foutmeldingen bestonden uit een combinatie van Nederlandse en Engelse woorden. Deze fouten waren na het achterhalen van het probleem redelijk snel op te lossen door bepaalde bestanden in het Framework aan te passen.

Iets wat ik niet kan testen is de werking van de rapportages, deze functie zal in de acceptatieomgeving geïmplementeerd worden. Hiervoor heb ik veel gegevens in de database nodig om de werking te kunnen controleren als er veel data aanwezig is. De data uit de oude applicatie heb ik daarom opgevraagd bij het IDC. Om deze testen uit te kunnen voeren zal ik de oude data in de nieuwe database zetten. Met het overzetten van de oude data kan ik gelijk ook de stap zetten om de gegevensconversie uit te voeren. Met deze gegevens kan ik verder ontwikkelen om zo de applicatie geschikt te maken voor de acceptatieomgeving.

De afspraak over de te implementeren wens "De omzetting van deze wens naar eis moet voldoende gefundeerd zijn en kan pas na het beta-testen besproken worden. Hierbij is afgesproken dat het managementteam van het contact centrum het unaniem eens moet zijn dat een wens omgezet moet worden naar eis, dan pas is er voldoende onderbouwing." heeft de implementatie van een wens tegengehouden. Het management kan het op dit moment niet eens worden welke wens zij graag terug willen zien in de applicatie waardoor de wensen niet verder zijn meegenomen bij de uitwerking van de opdracht. Deze zijn in de wacht gezet voor een volgende release.

5. Deployment in de acceptatieomgeving.

De eerste testen resulteerden in een positief geluid van de manager. Hij wilde er ook zeker van zijn dat de oude data gebruikt kan worden en dat de applicatie met meerdere gebruikers overweg kan. Om dit te realiseren is de database gevuld met historische data en is de applicatie verder ontwikkeld om zo te kunnen testen in de acceptatieomgeving met meerdere gebruikers.

5.1 Gegevensconversie

De manager heeft aangegeven dat hij graag gebruik zou willen maken van de data die via de oude applicatie aangemaakt is. Dit is een eis " De gegevens uit de oude applicatie, vanaf 2009 tot nu, moeten worden overgezet naar de nieuwe applicatie.". Om tot de kern van deze eis te komen heeft de manager nagedacht welke data op dit moment nog relevant is. Tijdens het vaststellen van deze eis in de inceptionfase is besloten dat de data vanaf 01-01-2009 gebruikt gaat worden. Er kan veel data bewaard worden, 31TB per tabel in het geval van PostgreSQL dat voor minimaal 10 jaar voldoende zal zijn. De data zelf zal nu niet veel ruimte in beslag nemen als het contact centrum groter gaat worden en als ook andere onderdelen, zoals de baliemedewerkers, gaan aansluiten, dan kan de data snel groeien. Daarom heb ik een schatting gemaakt dat 10 jaar het minimale termijn is om PostgreSQL te vullen.

Deze schatting heb ik kunnen maken door in de huidige applicatie de aantallen van de registraties per jaar vanaf 2006 op te vragen. Het totaal hiervan kwam uit op 1973729. Met in het eerste jaar (van 11 maanden) 326584 registraties en in 2010 waren er 424388 registraties. Als ik deze groei omgezet in procenten komt dit neer op 19% groei in 5 jaar tijd. Door de database te vullen met 1973729 registraties kan ik berekenen hoe lang deze groei kan aanhouden voordat PostgreSQL aan het limiet komt van 31 TB. De data neemt na een willekeurige vulling 162 MB aan ruimte in beslag. Stel dat het contact centrum op dezelfde voet doorgaat, wat onwaarschijnlijk is omdat er steeds meer contacten bijkomen, dan kan PostgreSQL nog 980 jaar werken voordat het limiet behaald is. De groei van het aantal contacten moet dan minimaal 98 maal verdubbelen om deze 10 jaar niet te halen.

Er is data beschikbaar vanaf februari 2006 tot het moment dat de nieuwe applicatie daadwerkelijk in gebruik is genomen. Deze data is opgeslagen in een Oracle database waar het contact centrum voor de gebruikte ruimte en resources moet betalen. Deze data moet geëxporteerd worden naar de PostgreSQL database in de nieuwe applicatie. Op het moment dat dit gedaan is, kan het contact centrum de kosten voor de Oracle database weer inzetten voor het jaarbudget en daarmee ook een bezuiniging realiseren. Er is met de afdeling IDC afgesproken dat zij de database afsluiten op het moment dat iedereen binnen

het contact centrum gebruik maakt van de nieuwe applicatie. Er is ook afgesproken dat het IDC een dump van deze database zal bewaren in het geval dat het contact centrum deze wil hebben.

De oude data is ondergebracht in een oracle (versie 9) database. De gegevens zijn opgeslagen in een propriëtair bestandsformaat. Om dit om te kunnen zetten naar de PostgreSQL database moet er een aantal handelingen uitgevoerd worden, PostgreSQL heeft hier een wikipagina[33] voor aangemaakt. Het contact centrum wil deze data niet over laten zetten door een bedrijf of via een betaalde applicatie wegens de bezuinigingen. Omdat PostgreSQL anders omgaat met de verwerking van sommige datatypes dan Oracle en er veel tijd in het overzetten van de data kan zitten, probeer ik te voorkomen dat hier veel tijd in gaat zitten. Ik moet deze procedure namelijk twee keer uitvoeren. Een keer voor de acceptatieomgeving en na de acceptatie van de applicatie in de productieomgeving. Door gebruik te maken van de ESF Database Migration Toolkit Standard Edition[34](ESF) applicatie vang ik een foutieve omzet van datatypes al af. Dit zorgt er ook voor dat ik meer speling krijg voor de kwaliteitscontrole van deze conversie.

Het omzetten van de data zal ik realiseren via een installatie van de Oracle database waarop ik een kopie van de huidige database zet. Deze database zal via een probeerversie van ESF gekoppeld worden aan de PostgreSQL database. Zo zal de data een op een worden overgezet. Omdat ESF een probeerversie is, zet dit programma de data over met een voorwaarde. Er komt bij elke tekstwaarde in de database een 'T' te staan in plaats van de daadwerkelijk gebruikte letter. Omdat ik in Oracle kon zien dat de tabel die de historische data bevat maar één kolom heeft met teksten, schat ik de tijd die er nodig is om dit om te zetten laag in. Dit zal ik via een update SQL queries in de PostgreSQL database corrigeren. Omdat deze tekstwaarde altijd zou beginnen met SB van de gebruikersnaam was het corrigeren minder werk dan de data zelf overzetten.

Toen alle veranderingen in de data verwerkt waren, was er een opschoning van de gegevens nodig om zo vanaf 2009 de registraties juist te kunnen bevragen. Deze opschoning is nodig omdat sommige rijen en waardes overbodig waren geworden of ze hadden een ander IDnummer gekregen in de PostgreSQL database. Dit geeft een vervuiling in de data en dat kan resulteren in foutieve rapportages. Het omzetten van deze IDnummers doe ik met behulp van een Excel bestand en pgAdmin III. In Excel zet ik de oude IDnummers tegenover de nieuwe IDnummers om zo de verandering in de nieuwe database bij te houden. Dit zal per dienst, onderwerp, binnenkomst en behandelwijze moeten gebeuren. Met de gegevens in Excel heb ik een overzicht waarmee ik kan voorkomen dat ik een huidige gebruikte waarde zal overschrijven met oude gegevens en zo de

[33] http://wiki.postgresql.org/wiki/Oracle_to_Postgres_Conversion
[34] http://www.easyfrom.net/

verkeerde gegevens zal opvragen bij een rapportage. In pgAdmin III zal ik per kolom de gegevens bewerken met behulp van update queries. De huidige applicatie is daarin leidend met de rapportages, de gegevens die daarin verwerkt zijn moeten hetzelfde zijn in de nieuwe applicatie.
Deze bewerking zal plaatsvinden in de tabellen Registratievraagklacht en onderwerp. Er bestaan onderwerpen die niet gebruikt worden maar toch weer op actief moeten worden gezet in deze applicatie. Deze onderwerpen staan alleen op actief in bepaalde seizoenen omdat ze dan actueel zijn. Vanwege de paginaopbouw van de huidige applicatie worden deze altijd weggehaald op het moment dat het seizoen voorbij is. Met de nieuwe applicatie zou dit niet meer nodig zijn omdat de nieuwe applicatie anders omgaat met de paginaopbouw.

Om te controleren of de conversie succesvol verloopt zal ik verschillende rapportages met verschillende soorten data uit de huidige applicatie en de nieuwe database met elkaar vergelijken. De nieuwe rapportages bestaan nu nog uit losse platte cijfers, er zijn nog formules nodig om dezelfde cijfers weer te geven. Deze formules zal ik in de nieuwe applicatie verwerken, maar om te zorgen dat ik de juiste cijfers naar voren krijg, test ik het eerst met de losse cijfers. Hier mag geen afwijking in zitten. Zou dit wel het geval zijn, dan moet ik de uitgevoerde acties in de nieuwe database weer terugdraaien. Dit risico zal ik ondervangen door verschillende testversies bij te houden van de nieuwe database in mijn subversiesysteem. Op deze manier zal er niet teveel tijd, moeite en werk verloren gaan bij een foute conversie en kan ik de stappen stuk voor stuk terughalen. De bewaking van de planning is op dit moment het belangrijkst omdat er een aantal projecten tegelijkertijd lopen waardoor de beschikbare tijd voor dit project schaars aan het worden is.

De uitvoer van de database controleren met de rapportages van de huidige applicatie kost veel tijd. Dit is echter een noodzakelijke procedure om de opdrachtgever te kunnen garanderen dat alle historische gegevens correct aanwezig zijn. Het controleren van de kwaliteit heeft daarom ook een ruime planning gekregen van drie werkdagen. Deze werkdagen werden er vier omdat ik een snelheidswinst had geboekt met ESF. De controles vereisen ook een goede concentratie omdat een fout in een klein hoekje kan zitten. Het vergelijken kostte uiteindelijk 3 werkdagen en omdat het stap voor stap is gegaan waren de fouten die er naar voren kwamen snel verholpen. Dit waren in alle gevallen typefouten bij de update query of bij de select query. De ene fout kostte daardoor meer hersteltijd dan de andere fout.

Nu de oude data op de juiste manier aanwezig is in de nieuwe applicatie is het mogelijk om daadwerkelijk de rapportagemodule te ontwerpen, maken en testen. Er zijn al vergelijkingen geweest met de oude applicatie en de nieuwe database-uitvoer. Het is daarna mogelijk om een syntactische en semantische test te doen. Het testen van de applicatie zal gedaan worden op de

acceptatieomgeving waarvoor ik de rapportagemodule zal ontwerpen en integreren in het al aanwezige prototype.

5.2 Module voor rapportages

Yii zorgt voor een koppeling met de database, maar kan geen rapportages maken. Voordat ik dit onderdeel wil ontwerpen zal ik de huidige database testen op de snelheid van de query verwerking.

Ik had bij het ontwerp van de database de keuze gemaakt voor het gebruiken van medewerker in de tabel RegistratieVraagKlacht. Medewerker zal de gebruikersnaam zijn van de gebruiker die registratie maakt. Dit is een unieke waarde binnen de gemeente en voldoet als primaire sleutel in de tabel Medewerker. Deze waarde komt zoals eerder aangegeven ook terug in de andere databronnen. Omdat bijna alle waardes in de tabel RegistratieVraagKlacht uit IDnummer's bestonden wilde ik weten of het logischer/sneller zou zijn om ook medewerker als medewerker_id op te nemen.

De database heeft nu echte historische data om dit te kunnen testen. Door dit te testen is het mogelijk dat ik de rapportagefuncties nog efficiënter maak. Ik heb twee versies van de database aangemaakt:

- een versie zoals origineel ontworpen
- een versie met in RegistratieVraagKlacht het attribuut medewerker_id welke een vreemde sleutel is die verwijst naar Medewerker.IDnummer. In de tabel Medewerker is het nieuwe attribuut IDnummer de primaire sleutel geworden. De waarden van medewerker zijn via een update aangepast naar de juiste corresponderende waarde in de tabel Medewerker.

Het eerste verschil is de grootte van de tabel RegistratieVraagKlacht, dit is 72MB in het originele ontwerp en 65 MB in het nieuwe ontwerp. Gezien de conclusie van minimaal 10 jaar gebruik die ik tijdens de gegevensconversie kon trekken over de tijd die het kost om aan het limiet van PostgreSQL te komen, heeft dit verschil weinig invloed. Het tweede verschil is de snelheid van bepaalde queries. Als ik alle waardes opvraag in RegistratieVraagKlacht is het nieuwe ontwerp 3 seconden sneller klaar (40312 ms en 37328 ms).
Deze query zal ik niet gebruiken in de rapportagefunctie, een query die ik elke week, elke maand, elk kwartaal en elk jaar zal gebruiken heeft meer waarde.

```
SELECT DISTINCT medewerker.gebruikersnaam as medewerker,
binnenkomst.naam as medium, count(binnenkomst_id) as aantal
FROM binnenkomst, registratievraagklacht, medewerker
```

```
WHERE datumtijd between '01-01-2009 00:00:00' and '01-01-2011 23:59:59'
and binnenkomst_id = binnenkomst.idnummer and medewerker.idnummer =
registratievraagklacht.medewerker_id
GROUP BY medewerker.gebruikersnaam, binnenkomst.naam
ORDER BY medewerker.gebruikersnaam, binnenkomst.naam ASC
```

Afbeelding 16 SQL Query om het klantcontact per agent op te vragen met de nieuwe structuur

Deze query gaat over een periode van twee jaar en heeft een verschil van 172 ms in het voordeel van het originele ontwerp (594 ms en 766 ms). Het nieuwe ontwerp is sneller als alles opgevraagd wordt en het originele ontwerp is sneller bij rapportagespecifieke queries. De winst die de database haalt bij het oude ontwerp is verwaarloosbaar als de resultaten groeien. Bij een query die 4 jaar aan data opvraagt, is het verschil 15 ms in het voordeel van het originele ontwerp (1129 ms en 1144 ms). Mede door dit resultaat kan ik concluderen dat deze wijziging vooral in de toekomst met meer data winst kan opleveren. Omdat het verschil nu verwaarloosbaar is zal dit verschil niet merkbaar zijn in de applicatie. Met deze reden heb ik het database model aangepast naar het volgende model.

Relationeel representatie model

RegistratieVraagKlacht (IDnummer, *medewerker_id, binnenkomst_id*, *behandelwijze_id*, *onderwerp_id, dienst_id,* datumTijd)

Medewerker (IDnummer, gebruikersnaam, voornaam, achternaam, rol, HashedPassword, actief)

Binnenkomst (IDnummer, naam, actief)

Behandelwijze (IDnummer, naam, intern, actief)

Onderwerp (IDnummer, naam, *dienst_id*, actief)

Dienst (IDnummer, naam, actief)

RegistratieVraagKlacht:
IDnummer is de primaire sleutel, deze mag niet null zijn.
medewerker_id is een vreemde sleutel die verwijst naar Medewerker.IDnummer, deze mag niet null zijn.
binnenkomst_id is een vreemde sleutel die verwijst naar Binnenkomst.IDnummer, deze mag niet null zijn.
behandelwijze_id is een vreemde sleutel die verwijst naar Behandelwijze.IDnummer, deze mag niet null zijn.
onderwerp_id is een vreemde sleutel die verwijst naar Onderwerp.IDnummer, deze mag niet null zijn.

dienst_id is een vreemde sleutel die verwijst naar Dienst.IDnummer, deze mag niet null zijn.
datumTijd mag niet null zijn.

Medewerker:
IDnummer is de primaire sleutel, deze mag niet null zijn
gebruikersnaam mag niet null zijn.
voornaam mag null zijn.
achternaam mag null zijn.
rol deze mag niet null zijn.
hashedpassword mag null zijn.
actief mag niet null zijn.

Binnenkomst:
IDnummer is de primaire sleutel, deze mag niet null zijn.
naam mag niet null zijn.
actief mag niet null zijn.

Behandelwijze:
IDnummer is de primaire sleutel, deze mag niet null zijn.
naam mag niet null zijn.
intern mag niet null zijn.
actief mag niet null zijn.

Onderwerp:
IDnummer is de primaire sleutel, deze mag niet null zijn.
naam mag niet null zijn.
dienst_id is een vreemde sleutel die verwijst naar Dienst.IDnummer, deze mag niet null zijn.
actief mag niet null zijn.

Dienst:
IDnummer is de primaire sleutel, deze mag niet null zijn.
naam mag niet null zijn.
actief mag niet null zijn.

afbeelding 17 het nieuwe relationeel representatie model

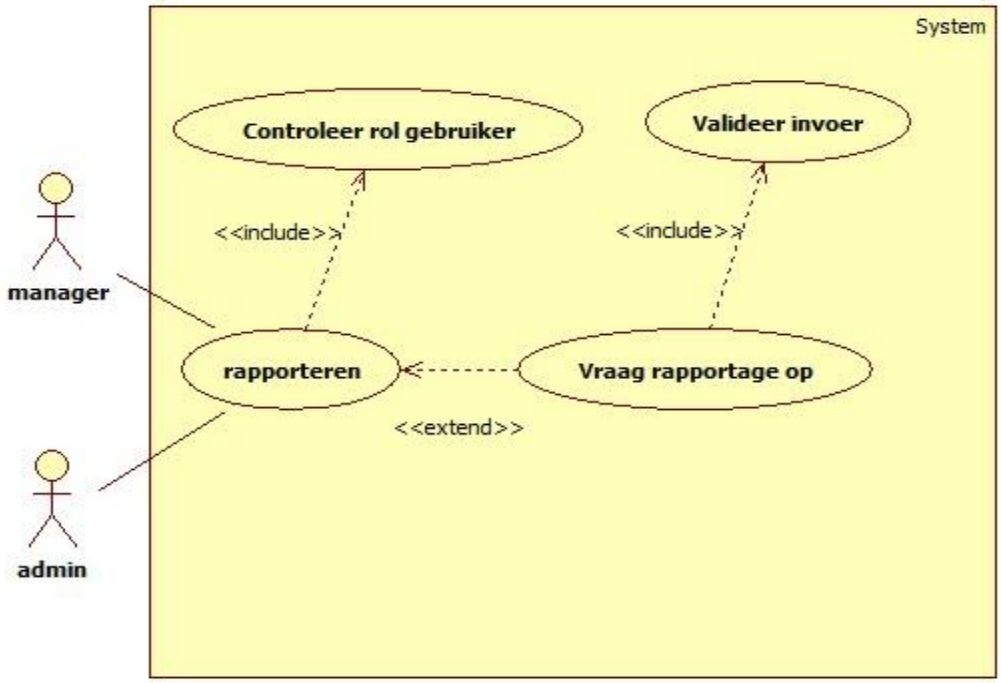
De queries voor de resultaten van de 10 rapportages die gekozen kunnen worden heb ik in pgAdmin III getest en voldeden aan mijn gestelde voorwaarde tijdens de gegevensconversie "Hier mag geen afwijking in zitten ten opzichte van de rapportages uit de huidige applicatie". De cijfers uit de rapportages kloppen, de opbouw van de view aan de hand van de queries die ik gebruik nog niet. De

applicatie zal nu niets doen met de resultaten die de verschillende queries terug kunnen geven.

Wat is belangrijk voor een rapportage binnen deze applicatie? Er moet een begin en einddatum zijn, welke beide ook dezelfde dag mogen zijn. Afhankelijk van welke rapportage er gekozen is zal er ook een extra waarde meegegeven moeten worden. De volgende rapportages, met daarbij de kenmerken die er meegegeven kunnen worden, zal ik implementeren.

- Het totaal aantal klantcontacten afgezet tegen de wijze van binnenkomst in een zelf te bepalen periode. Op te vragen per binnenkomst of alle binnenkomst mogelijkheden.
    - Meegeven: begindatum, einddatum en binnenkomst bij de per binnenkomst optie.
- Het totaal aantal klantcontacten afgezet tegen de wijze van binnenkomst per dienst gescheiden in een zelf te bepalen periode. Op te vragen per dienst of alle diensten.
    - Meegeven: begindatum, einddatum en dienst bij de per dienst optie.
- Het totaal aantal klantcontacten per dienst afgezet tegen de soort afhandeling (intern of extern) in een zelf te bepalen periode. Op te vragen per dienst of alle diensten. Per dienst is de soort afhandeling gespecificeerd in extern of extern en wat er intern of extern gedaan is om de vraag af te handelen.
    - Meegeven: begindatum, einddatum en dienst bij de per dienst optie.
- Het totaal aantal klantcontacten per dienst gespecificeerd naar product/onderwerp van deze dienst in een zelf te bepalen periode. Op te vragen per dienst of alle diensten.
    - Meegeven: begindatum, einddatum en dienst bij de per dienst optie.
- Het totaal aantal klantcontacten per medewerker gespecificeerd naar binnenkomst in een zelf te bepalen periode. Op te vragen per medewerker of alle medewerkers.
    - Meegeven: begindatum, einddatum en medewerker bij de per medewerker optie.

De invoer is beperkt tot totaal zes variabelen. De invoer voor de data staat vrij, drie variabelen komen uit de database en de rapportages staan vast in de applicatie. Met deze informatie kan ik een aantal UML diagrammen maken om de module vorm te geven.



Afbeelding 18 Usecasediagram rapporteermodule

De actoren in deze usecase zijn de manager en admin. De gebruikers met deze rollen kunnen gebruik maken van de rapportage functie. De applicatie zal dit controleren bij het aanvragen van de rapportagepagina door gebruik te maken van een al gemaakte functie in de applicatie. De invoer voor een rapportage zal gevalideerd worden voordat er een eventueel resultaat in beeld komt.

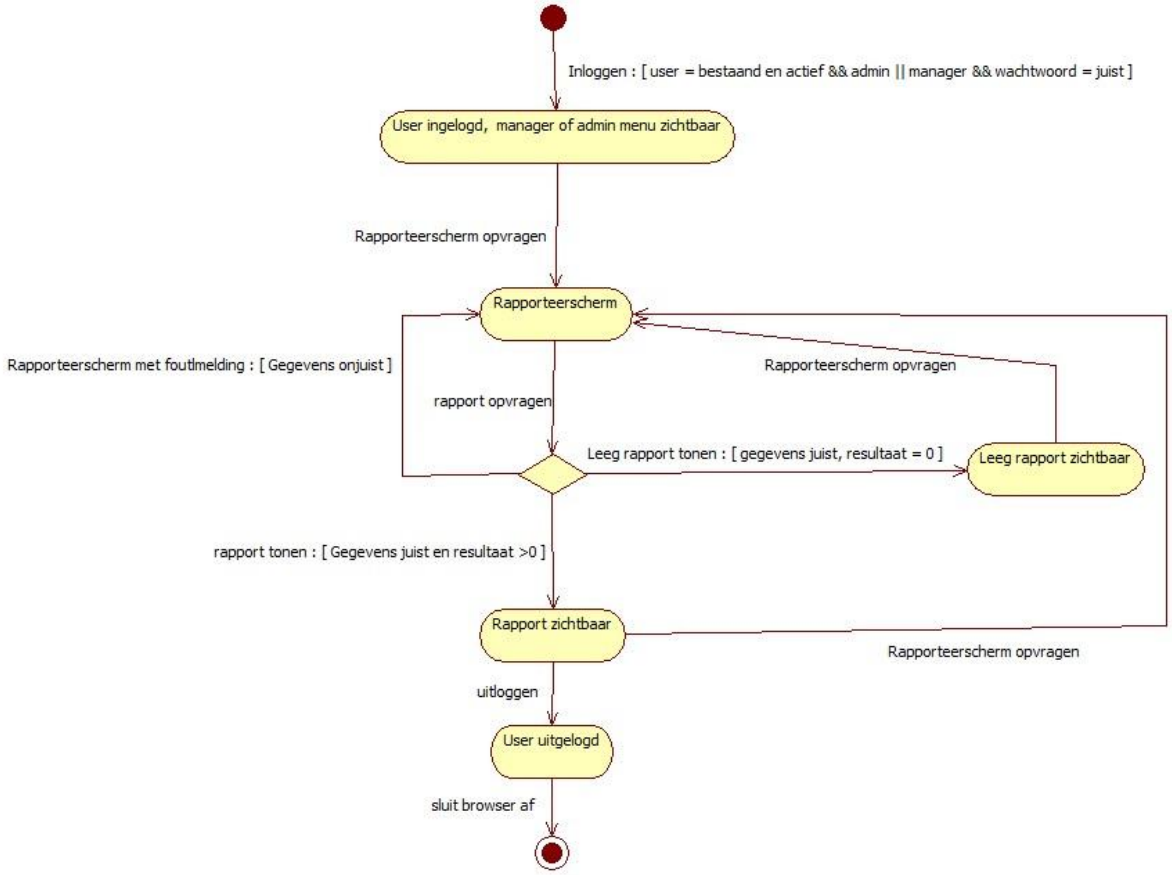
Dit vertaalt zich naar de volgende usecases.

Naam	Rapporteren
Samenvatting	De gebruiker kan hier rapportages opvragen.
Actoren	manager/admin
Aannamen	De gebruiker moet ingelogd zijn als manager of admin.
Beschrijving	De gebruiker heeft hier de mogelijkheid om rapportages op te vragen in het systeem. Dit kan door het ingeven van data, over welk onderdeel de rapportage moet gaan en wat voor vooraf gedefinieerd rapport het moet zijn.
Uitzondering	De gebruiker is niet ingelogd als manager of admin.
Resultaat	De gekozen rapportage komt zichtbaar in het scherm.

Naam	Vraag rapportage op
Samenvatting	De gebruiker kan hier rapportages opvragen.
Actoren	manager/admin
Aannamen	De gebruiker moet ingelogd zijn als manager of admin en de

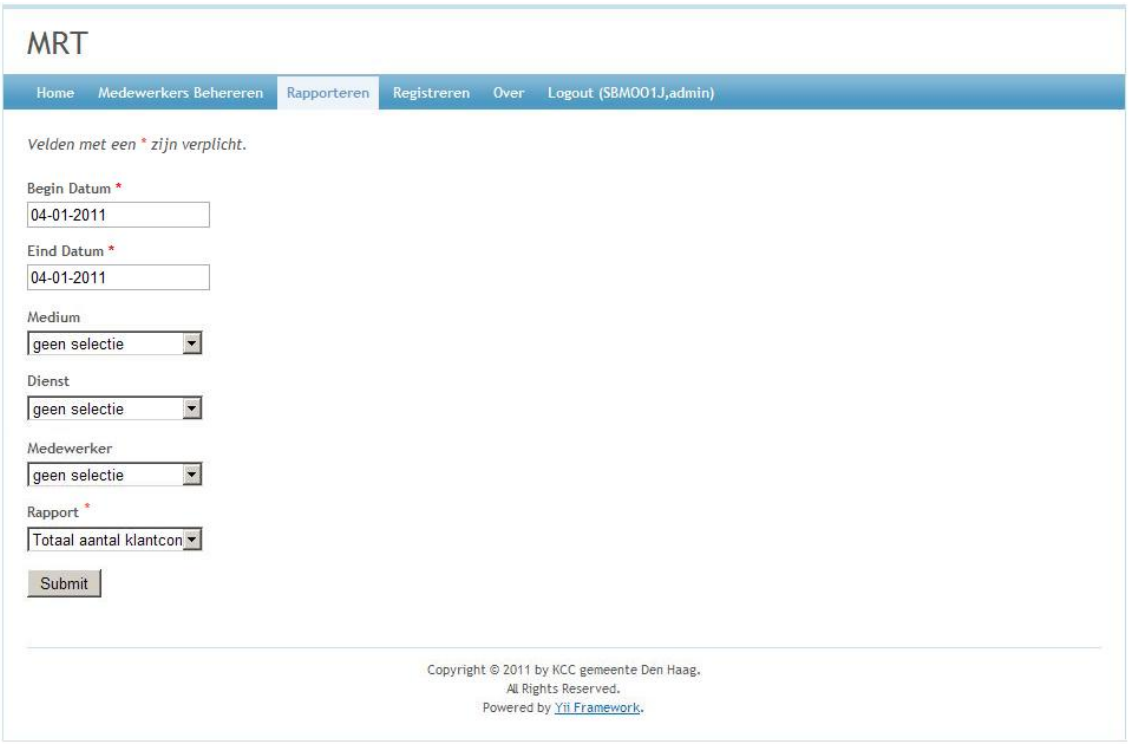
	rapportagescherm open hebben staan.
Beschrijving	Bij de juiste data invoer (datums, soort rapport en eventueel specifieke dienst/binnenkomst/medewerker) kan de gebruiker het rapport opvragen.
Uitzondering	De gebruiker is niet ingelogd als manager of admin.
Resultaat	De gekozen rapportage komt zichtbaar in het scherm.

De toestanden waar het rapporteer gedeelte zich in kan bevinden zijn als volgt.



Afbeelding 19 Toestandsdiagram van de rapporteermodule

Hoe verwerk ik de waardes voor de rapportage in de logica met de kleinste kans op foutieve invoer?
De waardes uit de database (diensten, medewerkers en binnenkomsten) en de rapportages kan ik in een lijst zetten, zodat er niet nagedacht hoeft te worden welke keuzes er bestaan. Het rapporteerscherm laat daarbij zien welke waardes verplicht zijn en geeft bij de datum een voorbeeld van het formaat. Dit voorkomt al fouten aan het begin, stel dat er dan toch een fout gemaakt is dan kan ik dat laten controleren in de applicatie.



Afbeelding 20 Het rapportagescherm met logica

De controle op de waardes wil ik via het RegistratieVraagKlacht model laten uitvoeren. Zo is de kans op foutieve invoer bijna uitgesloten omdat Yii de invoer zal vergelijken met het type dat de database vraagt bij die invoer. Deze vergelijking is echter niet overal mogelijk vanwege de koppeling die het model met de database heeft. Door deze koppeling zijn de waardes niet te controleren als deze niet in de database als attribuut zijn opgenomen. Dit model/deze tabel heeft één datum en automatische controle zal dan ook alleen op één datum plaatsvinden. Het controleren van de waardes is belangrijk in verband met de eis "Zowel de afdeling Functioneel Beheer als een agent moet met de applicatie overweg kunnen.". Feedback van de applicatie door middel van een foutmelding komt netter en begrijpelijker over. Het voorkomt dat een gebruiker de handleiding er bij moet pakken of hulp moet vragen aan de afdeling functioneel beheer.

Om de rapportagefunctionaliteit toch goed aan te kunnen bieden en te controleren zal ik in de database een rapporteer tabel aanmaken. Deze tabel zal de namen van de rapportages bevatten. Deze tabel zorgt in combinatie met Yii ook voor de mogelijkheid van invoercontrole. De afdeling functioneel beheer kan in deze tabel via pgAdmin III een rapportage op deactief zetten.

Het model ziet er als volgt uit.

Relationeel representatie model

Rapporteer (IDnummer, rapportagenaam, medewerker_id, binnenkomst_id, behandelwijze_id, onderwerp_id, dienst_id, datumTijd1, datumTijd2, actief)

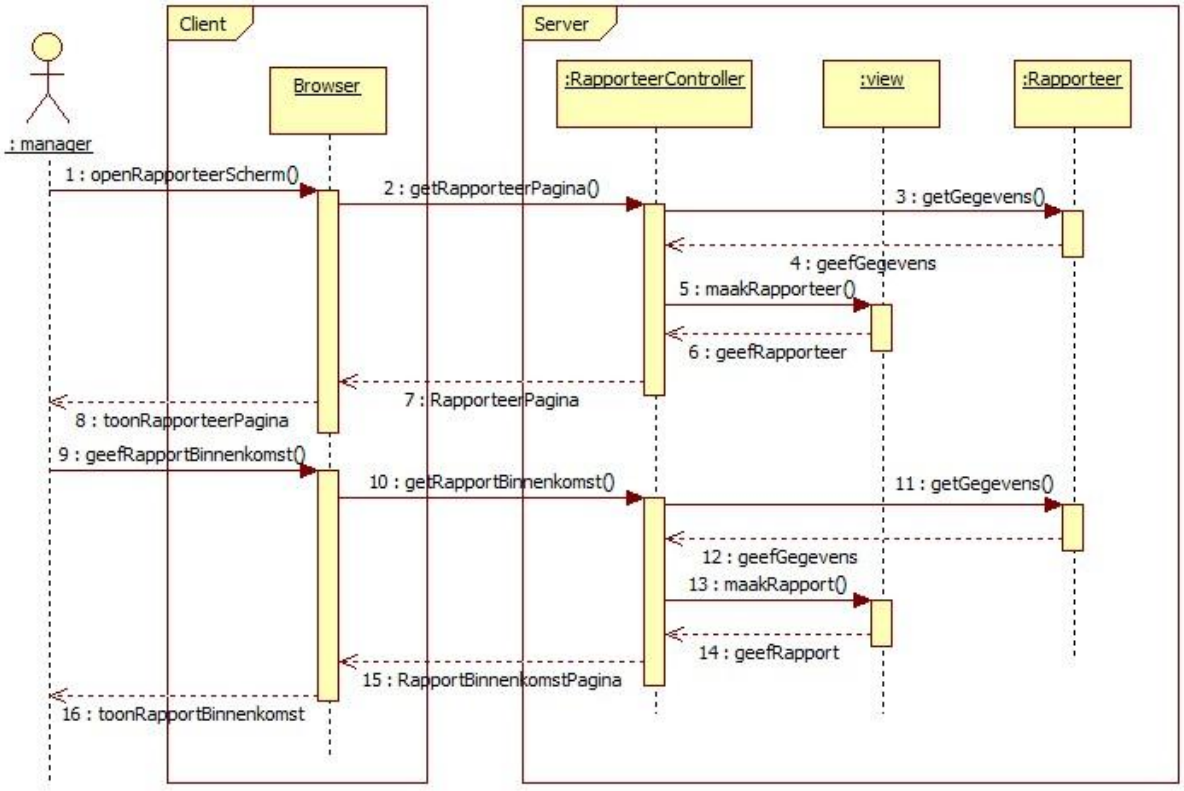
Rapporteer:
IDnummer is de primaire sleutel, deze mag niet null zijn.
rapportagenaam mag niet null zijn.
medewerker _id mag niet null zijn.
binnenkomst_id mag niet null zijn.
behandelwijze_id mag niet null zijn.
onderwerp_id mag niet null zijn.
dienst_id mag niet null zijn.
datumTijd1 mag null zijn.
datumTijd2 mag null zijn.
actief mag niet null zijn.

Relationeel implementatie model

```
CREATE TABLE rapporteer
(
  IDnummer SERIAL NOT NULL,
  rapportagenaam character varying(200) NOT NULL,
  medewerker_id boolean NOT NULL DEFAULT false,
  binnenkomst_id boolean NOT NULL DEFAULT false,
  behandelwijze_id boolean NOT NULL DEFAULT false,
  onderwerp_id boolean NOT NULL DEFAULT false,
  dienst_id boolean NOT NULL DEFAULT false,
  datumtijd1 timestamp without time zone,
  datumtijd2 timestamp without time zone,
  actief boolean NOT NULL DEFAULT true,

  PRIMARY KEY (IDnummer)
)
```

Afbeelding 21 Database modellen rapporteer tabel

Door deze opbouw kan er per rapportage aangegeven worden welke optie id's er nodig zijn voor de rapportage. DatumTijd1 krijgt als datum 01-01-2009 omdat er vanaf die datum cijfers aanwezig zijn voor deze rapportages. De applicatie zal gebruik maken van IDnummer, rapportagenaam, datumTijd1, datumTijd2 en actief in de nieuw aan te maken RapporteerController, Rapporteermodel en views. Dit idee heb ik inzichtelijk gemaakt met een sequencediagram.



Afbeelding 22 Sequencediagram rapporteermodule

De stappen van de applicatie in het sequencediagram zijn als volgt:
Stap 2 t/m 7:

2. in de controller komt er een aanvraag voor het rapportagescherm->
3. verzoek naar het model voor de vooringevulde informatie die aanwezig is in de database->
4. geef deze gegevens in een of meerdere arrays terug aan de controller->
5. de controller stuurt deze array(s) naar de view van het rapportagescherm->
6. de view maakt er een pagina voor de browser van en stuurt deze naar de controller->
7. de controller stuurt de pagina door naar de browser.

Dezelfde stappen maakt de applicatie bij het opvragen van een rapport waarbij er nog een controle op de invoer plaats zal vinden.

De SQLqueries waarmee ik de uitvoer van de database had getest zijn ondergebracht in het model Rapporteer. De controller van Rapporteer zorgt dat deze gegeven aangevraagd kunnen worden en verwerkt worden in de views. De views van de rapporten maken gebruik van de arrays, die als resultaat terugkomen, om de pagina op te bouwen. De opbouw van de pagina wil ik met tabellen doen, welke pas aangemaakt worden als er resultaten zijn. Zo kan ik

inspelen op een verandering van de hoeveelheid diensten, onderwerpen, binnenkomsten, behandelwijzen en medewerkers.

Het omzetten van één array naar één tabel is niet ingewikkeld. Een voorbeeld hiervan ziet er als volgt uit voor het rapport " Totaal aantal klantcontact per binnenkomst apart":

```
<table border=3>
<TR>
        <TD>Binnenkomst</TD>
        <TD>Aantal</TD>
 </TR>



<?php foreach($resultaat as $t) {



                        $aantal = $t['aantal'];
                        $medium = $t['naam'];
                        echo "<TR>";
                        echo "<TD>";
                        echo $medium;
                        echo "</TD>";
                        echo "<TD>";
                        echo $aantal;
                        echo "</TD>";
                        echo "</TR>";

                }
?>

</table>
```

Afbeelding 23 een deel de opbouw van de rapportage Totaal aantal klantcontact per binnenkomst apart

Het resultaat "$resultaat" bestaat uit een array waar het aantal klantcontacten voor de bepaalde binnenkomst in staat. Via een foreach doorloopt de applicatie dit array en maakt er rijen en kolommen omheen. Als er meerdere arrays zijn gaat het omzetten naar tabellen niet zomaar met de losse arrays, deze moeten in een multidimensionaal array verwerkt worden voordat deze mooi in een tabel gezet kunnen worden. De opbouw ziet er dan als volgt uit bij de rapportage "Totaal aantal klantcontact alle diensten naar binnenkomst".

```
<?php       $teller_arr1 = 1;

            foreach ($resultaat1 as $t) {
                  //$resultaat1 heeft de namen van de binnenkomsten als
                  naam en de totalen bij die namen als totaal in de periode die
                  gekozen is
                  //Hierin komen de kolommen voor de soort binnenkomst
                  //dit kan van 0 tot in principe oneindig gaan

                        echo "<TD>";
                        echo $t['naam'];
                        echo "</TD>";

                  //vul hier een array met alle soorten binnenkomsten die
                  gevonden zijn in de periode die gekozen is

                        $arr1[$teller_arr1] = $t['naam'];

                  //vul hier een array met de totalen per binnenkomst die
                  gevonden zijn in de periode die gekozen is

                        $arr3[$teller_arr1] = $t['totaal'];

                        $teller_arr1 ++;
                  }
?>
```

Afbeelding 24 het eerste deel van de opbouw van de rapportage Totaal aantal klantcontact alle diensten naar binnenkomst

Deze opbouw geeft de tabel in de view namen boven de kolommen en vult een array met de totalen van deze namen. Omdat deze namen kunnen variëren van nul tot nu zes maar in de toekomst misschien meer moet er een multidimensionaal array gemaakt worden per dienst, welke ook weer kunnen variëren in aantal, waar alle mogelijke gevonden binnenkomsten apart in staan. Omdat de hoeveelheden van beide resultaten niet vast staan, bouw ik de array pas op als er een resultaat terug is gekomen. Deze dynamische opbouw heb ik gekozen om te voorkomen dat er resultaten buiten beeld vallen als de tabellen in de view te kort schieten aan ruimte. Nu maakt het niet uit of er één resultaat is of er vijf zijn. De opbouw past zich hier op aan. Daarvoor zorgt het volgende stuk in de opbouw.

```
<?php       $teller_arr = 0;
            //$resultaat2 heeft de namen van de diensten als naam en de
            totalen bij die namen als totaal in de periode die gekozen is

            foreach ($resultaat2 as $t) {
            //vul in dit array het totalen per dienst in die gevonden zijn in de
            periode die gekozen is

            $arr4[] = $t['totaal'];

            //vul hier op positie 0 in het multidimensionale array de naam van
            de dienst in die gevonden zijn in de periode die gekozen is

            $test[$teller_arr][0] = $t['naam'];

            //maak hier voor elke dienst een array binnen het array aan met op
            positie 0 de dienst en voor elke binnenkomst een key in dat array
            beginnend bij 1 tot een max van count($arr1)

                    for($i=1; $i <= count($arr1); $i++ ){
                            $test[$teller_arr][$i] = $arr1[$i];
                            }

                    $teller_arr ++;
            }
?>
```

Afbeelding 25 het tweede deel van de opbouw van de rapportage Totaal aantal klantcontact alle diensten naar binnenkomst

Nu bestaan er drie arrays waarmee de applicatie verder moet werken $arr3, $arr4 en $test. Deze bevatten het totaal per dienst, het totaal per binnenkomst en alle gevonden diensten met de mogelijke binnenkomsten. Het multidimensionale diensten array moet nog gevuld worden met de individuele cijfers. Daarvoor zorgt het volgende deel. Deze cijfers komen uit het derde en laatste array ($resultaat3) wat de controller meegeeft. Als dit verwerkt is in de applicatie dan zijn alle cijfers die de database kan aanleveren verwerkt in de arrays.

```
<?php       $totaal = 0;
            //$resultaat3 heeft de namen van de dienst als dienst en de namen
            van de binnenkomst als medium en het totaal per binnenkomst per
            dienst als aantal in de periode die gekozen is

            foreach($resultaat3 as $t) {
            //zet de waardes om naar variabelen

                  $dienstID = $t['dienst'];
                  $binnenkomstID = $t['medium'];
                  (int)$aantal = $t['aantal'];

            //dit zal uiteindelijk het totaal aantal registraties zijn die er gemaakt
            zijn in de periode die gekozen is
                        $totaal += $aantal;

            //met deze structuur doorloop je het multidimensionale array om
            alles in te vullen waar nodig
                  for ($row = 0; $row < (count($test)); $row++){

                        $teller = 0;

                        foreach($test[$row] as $key => $value){

                              if ($dienstID == $test[$row][0] && $dienstID
                                    == $value){
                              //doe niets dit is de dienst
                                    }
                              else if ($dienstID == $test[$row][0] &&
                                          $binnenkomstID == $value){
                              //vul hier het aantal in op de plaats van $value
                              want deze klopt hiermee overschrijf je de
                              naam van de binnenkomst voor het aantal
                                    $test[$row][$teller] = $aantal;
                                    }

                                    $teller++;
                              }
                        }
            }
```

Afbeelding 26 het derde deel van de opbouw van de rapportage Totaal aantal klantcontact alle diensten naar binnenkomst

```
//doorloop het multidimensionale array door om alles op het beeld te zetten
	for ($row = 0; $row < (count($test)); $row++){
		//de groote van het array zorg hier voor genoeg rijen
			echo '<tr>';

		//per rij doorloop je de foreach en vul je de cellen op de plaats van
		dienst en de soort binnenkomst
		foreach($test[$row] as $key => $value){
			echo '<td>';

			//controleer of de waarde niet op plaats $test[$row][0] en
			value geen waarde is om getallen in te vullen achter de
			dienst als de waarde op plaats $test[$row][0] staat print je
			de dienstnaam
				if($key != $test[$row][0] && !is_numeric($value)){
					echo '0';
				}
				else{
					echo $value;
				}
			echo '</td>';

			}

		//vul hier het totaal voor deze dienst in
		echo '<td>';
		echo $arr4[$row];
		echo '</td>';
		echo '<td>';
		//bereken hier het percentage ten opzichte van het $totaal in 4
		cijfers achter de komma
		echo round($percentage =(($arr4[$row] * 100) /$totaal),4) . " %";
		echo '</td>';
		echo '</tr>';
	}
```

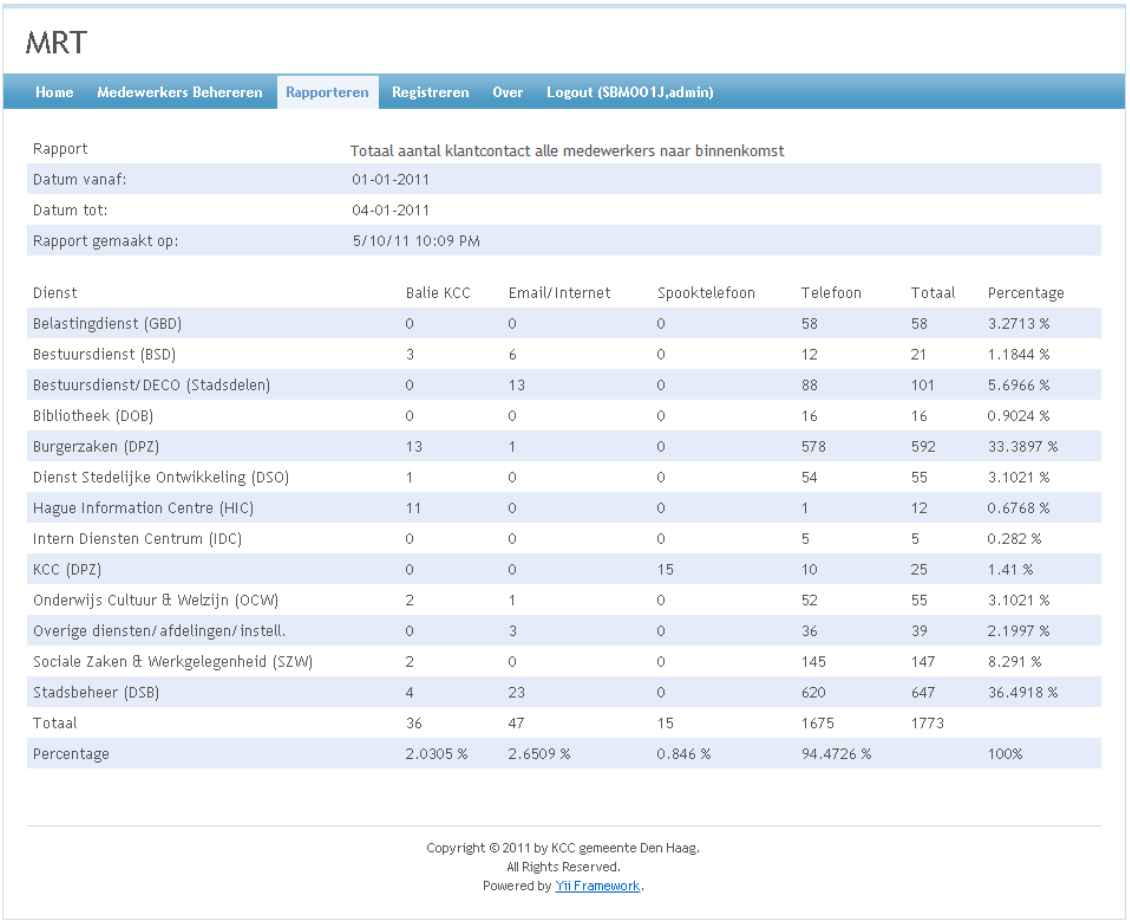
Afbeelding 27 het vierde deel van de opbouw van de rapportage Totaal aantal klantcontact alle diensten naar binnenkomst

Het vierde deel zorgt ervoor dat alle cijfers van de diensten zichtbaar zijn op de pagina. Omdat het ook belangrijk is om de totalen per binnenkomst te zien en de percentages daarbij heeft de tabel nog twee rijen nodig met deze gegevens. Daar zorgt het vijfde en laatste deel voor.

```
<TR>
	<TD>Totaal</TD>
<?php
	foreach($arr3 as $t){
		echo '<td>';
		echo $t;
		echo '</td>';
		}
	echo '<td>';
	echo array_sum($arr3);
	echo '</td>';
?>
<TD>
</TD>
</TR>
<TR>
 <TD>Percentage</TD>
<?php
	foreach($arr3 as $t){
		echo '<td>';
		echo round((($t * 100) / (array_sum($arr3))),4) . " %";
		echo '</td>';
		}
	echo '<td>';
	echo '</td>';
?>
			<TD>100%</TD>
 </TR>
```

Afbeelding 28 het vijfde deel van de opbouw van de rapportage Totaal aantal klantcontact alle diensten naar binnenkomst

Samen met de koptekst in de rapportageview zorgen deze delen voor de opbouw van een rapportage waar verschillende waarden tegenover elkaar uitgezet worden. Deze methode gebruik ik ook in andere rapportages soms in iets andere vorm, maar het idee blijft hetzelfde. Het beeld wat uit deze opbouw komt is het volgende.



MRT

Home | Medewerkers Behereren | Rapporteren | Registreren | Over | Logout (SBM001J,admin)

Rapport	Totaal aantal klantcontact alle medewerkers naar binnenkomst
Datum vanaf:	01-01-2011
Datum tot:	04-01-2011
Rapport gemaakt op:	5/10/11 10:09 PM

Dienst	Balie KCC	Email/Internet	Spooktelefoon	Telefoon	Totaal	Percentage
Belastingdienst (GBD)	0	0	0	58	58	3.2713 %
Bestuursdienst (BSD)	3	6	0	12	21	1.1844 %
Bestuursdienst/DECO (Stadsdelen)	0	13	0	88	101	5.6966 %
Bibliotheek (DOB)	0	0	0	16	16	0.9024 %
Burgerzaken (DPZ)	13	1	0	578	592	33.3897 %
Dienst Stedelijke Ontwikkeling (DSO)	1	0	0	54	55	3.1021 %
Hague Information Centre (HIC)	11	0	0	1	12	0.6768 %
Intern Diensten Centrum (IDC)	0	0	0	5	5	0.282 %
KCC (DPZ)	0	0	15	10	25	1.41 %
Onderwijs Cultuur & Welzijn (OCW)	2	1	0	52	55	3.1021 %
Overige diensten/afdelingen/instell.	0	3	0	36	39	2.1997 %
Sociale Zaken & Werkgelegenheid (SZW)	2	0	0	145	147	8.291 %
Stadsbeheer (DSB)	4	23	0	620	647	36.4918 %
Totaal	36	47	15	1675	1773	
Percentage	2.0305 %	2.6509 %	0.846 %	94.4726 %		100%

Afbeelding 29 het resultaat van de opbouw van de rapportage Totaal aantal klantcontact alle diensten naar binnenkomst

De invoer voor een rapportage zal nu gecontroleerd worden door Yii, de uitvoer was al getest met de nieuwe database en rapportages uit de huidige applicatie. Nu de uitvoer is omgezet in beeld in de nieuwe applicatie ga ik deze uitvoer wederom vergelijken met de rapportages uit de huidige applicatie om te controleren of deze nog steeds overeenkomt.

De uitvoer in de view komt nog steeds overeen met de uitkomsten van de database en van de huidige applicatie. Dit soort views moeten voor deze eis " Het automatisch aanmaken van rapportages in Excel formaat." ook verwerkt worden in Excel. Dit zal ik later oppakken omdat er nu zoals eerder vermeld meerdere projecten gestart zijn waar mijn manager mijn inzet voor nodig heeft. Omdat de externe databronnen om mee te testen pas laat aangeleverd zijn, heb ik een afspraak gemaakt met de manager. Ik kan alleen meewerken aan de andere projecten op voorwaarde dat de koppeling met de andere databronnen uit de huidige eisen wordt gehaald. Hij is hiermee akkoord gegaan, de tijdsdruk woog zwaarder dan de noodzaak voor deze extra cijfers op dit moment.

5.3 Aanpassingen werking schermen

De invoervensters van de applicatie zijn nu overal aanwezig en krijgen een controle op de invoer. Dit werkt goed en zorgt voor duidelijkheid, maar de vensters zijn nog te druk en gevoelig voor meer invoer dan nodig. Bij registreren is het bijvoorbeeld mogelijk om bij elke dienst een waarde te kiezen en dat accepteert de applicatie niet. Deze wil één invoer voor de dienst en één voor het onderwerp. Als alle lijsten ingevuld zijn met een onderwerp komt er een foutmelding dat er maar één waarde gekozen mag worden. Hoewel dit scherm er al overzichtelijker uit ziet dan de huidige applicatie kreeg ik signalen van de testgroep en de manager dat het niet fijn werkt als er verkeerd geklikt wordt. Door dit punt kom ik weer terug op de eis " Zowel de afdeling Functioneel Beheer als een agent kunnen met de applicatie overweg.". Er moet een verbeterslag komen in de view van registreren en rapporteren om deze eis goed in te kunnen vullen.

De afvang van een verkeerde invoer via een model is niet overal voldoende, deze afvang werkt pas na het klikken op submit. Er moet een flow in het venster komen waarbij het praktisch onmogelijk is om een verkeerde keuze te maken.

Yii heeft een ondersteuning voor AJAX[35] wat de mogelijkheid geeft voor een interactieve pagina waar een verandering (klik op een button/keuze in een lijst) kan leiden tot het veranderen van iets anders zoals een lijst of tekst. Deze AJAX integratie is met behulp van jQuery[36] in Yii aanwezig.

In jQuery zijn er veel mogelijkheden om interactie met de gebruiker tot stand te brengen. De interactie die ik wilde gebruiken maakt gebruik van een keuze in één lijst. Er zijn op dit moment 14 diensten met totaal 86 onderwerpen. Dit geeft bij het registratiescherm 14 lijsten met daarin de naam van de dienst en de onderwerpen die bij deze dienst horen. Daar moet één keuze gemaakt worden maar zoals aangegeven is het mogelijk om bij alle 14 een keuze te maken. Dit wil ik terugbrengen naar één lijst voor de diensten en als daar een keuze gemaakt is dan zal er één lijst onderwerpen gevuld worden die bij die dienst horen. Hiermee voorkom ik dat er meer dan één keuze gemaakt kan worden, waardoor de flow van het registeren beter werkt en het scherm heeft daardoor minder informatie dan in het prototype.

Om dit te implementeren is er een lijst "onderwerp" welke afhankelijk is gemaakt van de dienst die gekozen gaat worden. Het is voor mij de eerste keer dat ik gebruik maak van jQuery. De mogelijkheden van jQuery zijn enorm en bleken toereikend voor mijn idee om een lijst dynamisch te vullen.

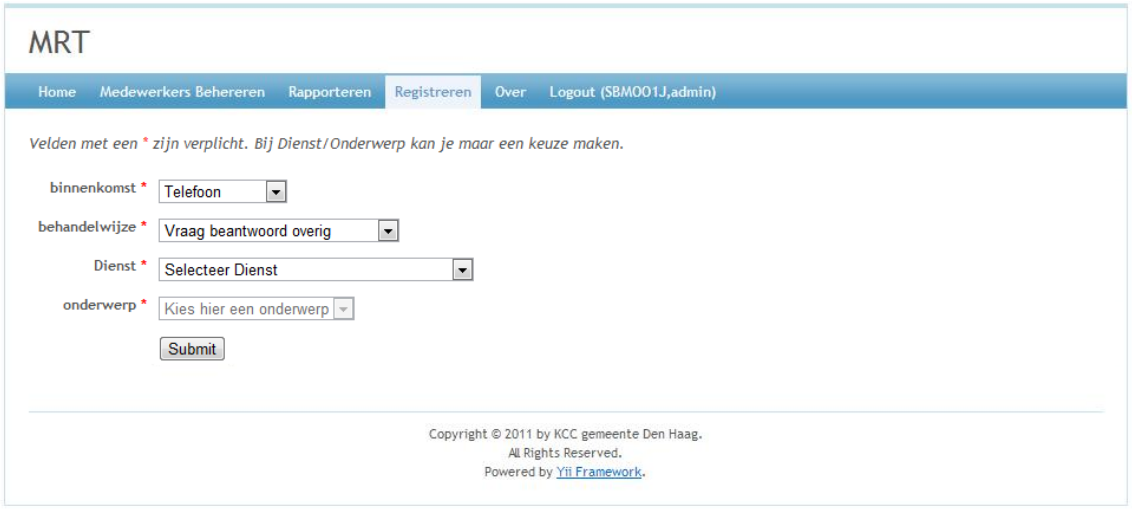
[35] http://nl.wikipedia.org/wiki/Asynchronous_JavaScript_and_XML
[36] http://jquery.com/

Deze opbouw heb ik daarom ook toegepast in het rapporteerscherm. De structuur die ik hiervoor gebruik is als volgt:

```
echo "<div class=\"row\">";

echo $form->labelEx($model,'dienst_id');
echo $form->dropDownList($model,'dienst_id' ,
   $dienst,
   array('style'=>'width: auto',

      'empty'=>'Selecteer Dienst',
      'ajax' =>
      array(
         'type'=>'POST', //request type
         'url'=>CController::createUrl('site/VulOnderwerpen'), //url to call.

         'update'=>'#RegistratieVraagKlacht_onderwerp_id', //selector to update

         'data'   => 'js:"dienst_id="+jQuery(this).val()',
         'success'=>'js:function(data){

       jQuery("#RegistratieVraagKlacht_onderwerp_id").empty();

jQuery("#RegistratieVraagKlacht_onderwerp_id").removeAttr("disabled");
            jQuery("#RegistratieVraagKlacht_onderwerp_id").append(
               data
            );
         }',
      )
   )
);
```

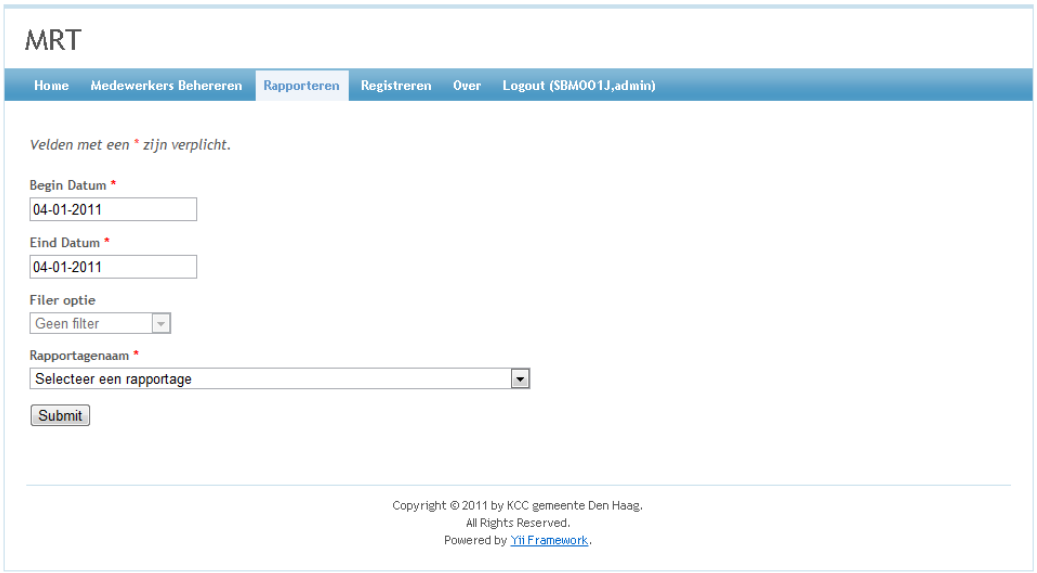
Afbeelding 30 jQuery voorbeeld om een lijst dynamisch te vullen

Het gebruik van jQuery voegt een meewaarde toe aan de applicatie omdat er nu meer interactie is terwijl de gebruiker minder keuzes moet maken voor deze interactie. Dit vertaald zich nu in een overzichtelijk scherm en alleen een foutmelding als er niets gekozen wordt in het registreerscherm. Het rapporteerscherm heeft met behulp van jQuery in plaats van zes invoermogelijkheden er maar vier. Een daarvan zal pas actief worden met relevante data als er een rapportage gekozen is waar een extra invoer voor nodig is. Door deze verandering samen met de rapporteermodule te integreren in de applicatie in de acceptatieomgeving wil ik de eisen en wensen die nu aanwezig zijn laten testen in een acceptatietest. De verandering die ik zal implementeren werkt als volgt door in de views:



Afbeelding 31 nieuw registreerscherm met deactive lijst onderwep

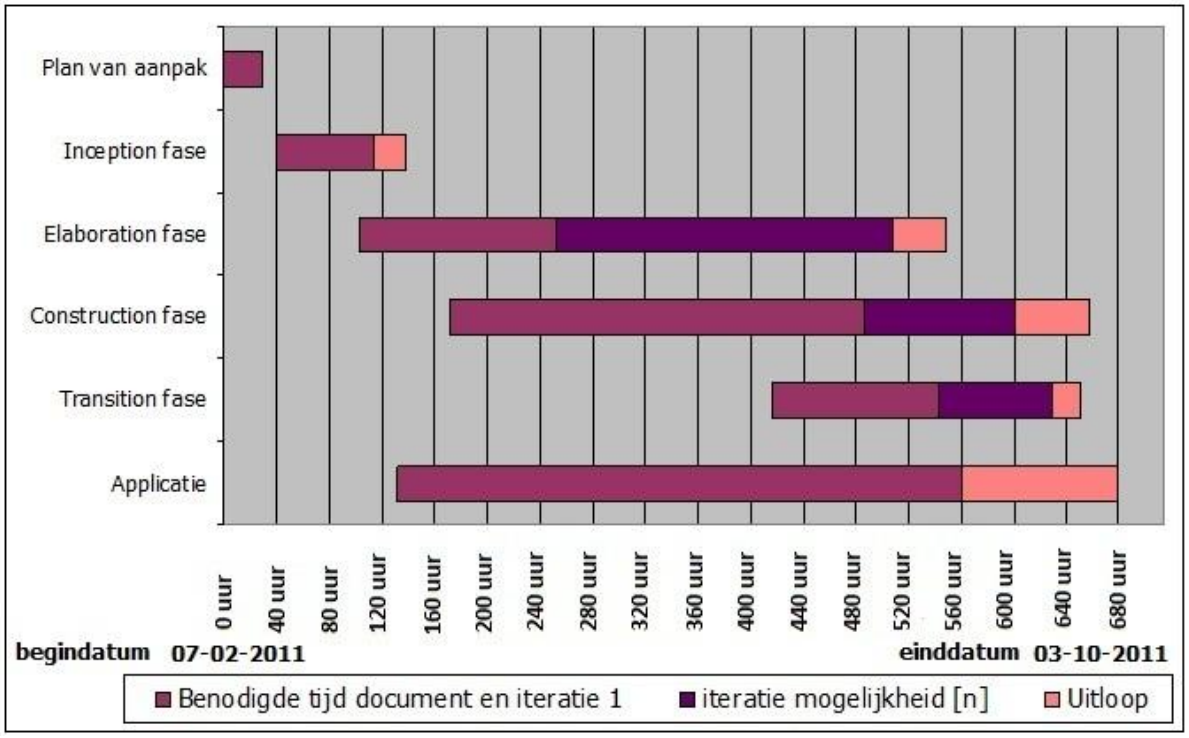
Afbeelding 32 nieuw registreerscherm met actieve lijst bij onderwerp



Afbeelding 33 nieuw rapportagescherm met deactieve filter optie

5.4 Rapporteer module opnemen in de acceptatieomgeving.

De acceptatieomgeving is nu voorzien van het prototype van de applicatie. Hier zijn nog niet alle eisen in verwerkt. Met het ontwerpen van de rapportagemodule zijn er een aantal verandering doorgevoerd in de applicatie. Er is ook een wijziging in de planning gemaakt. Dit heeft effect op de eis van de koppeling van externe bronnen gehad zoals eerder aangegeven en het effect op de planning is dat ik de eerst gestelde 6-6-2011 niet zal halen. De planning heb ik daardoor aan moeten passen. De gelijke verdeling van uren over de gestelde weken bleek een te hoge belasting te veroorzaken omdat mijn inzet nodig is bij andere projecten binnen het contact centrum. Voor deze projecten was het niet mogelijk om een ander persoon in te zetten en mede daardoor heb ik de planning aangepast naar 03-10-2011 als einddatum. In plaats van de begindata van de weken op de x-as te noteren is het realistischer om uren als waarde te nemen bij de x-as. Hierdoor kan ik de tijd flexibel inzetten tussen de begin en einddatum. De planning ziet er daardoor als volgt uit.



Afbeelding 34 aangepaste planning

Het inplannen van werkzaamheden zal nu gebeuren op uurbasis, waar 40 werkuur voor een werkweek zal staan. De globale planning in het plan van aanpak staat in weken aangegeven. Omdat er nu meer weken beschikbaar zijn zullen die weken ook op basis van werkuren geteld worden. Zo kan ik nog grip houden op de oplevering van de producten door de gewerkte uren bij elkaar op te tellen en zo de werkweek in de planning bepalen.

Met het implementeren van de veranderingen in de applicatie op de acceptatieomgeving voldoet de applicatie bijna helemaal aan de aangepaste gestelde eisen.

De acceptatieomgeving heeft nog een versie van de database die geen historische data heeft en nog volgens het oude model is opgebouwd. Het nieuwe model overzetten is de eerste actie die ik moet uitvoeren om de nieuwe functionaliteiten aan te kunnen bieden. PostgreSQL biedt hiervoor een optie aan om van een database back-up te maken. Deze back-up is over te zetten naar een ander systeem door de database aan te maken op dat nieuwe systeem. Dit hoeven niet de tabellen te zijn maar alleen de databasenaam aanmaken is voldoende om deze back-up terug te zetten. Door deze constructie verzeker ik de manager en mijzelf ervan dat de ontwikkel-, test- en acceptatieomgeving gelijk lopen wat betreft de data. Dit geeft ook de zekerheid dat de gebruikte data geen fouten bevat als deze juist op de ontwikkelomgeving aanwezig is. De initiële database zal zich daarom bevinden op de ontwikkelomgeving en vanaf daar zal ik een back-up maken.

De applicatiebestanden zijn opgenomen in een versiebeheersysteem van tortoiseSVN[37]. Met deze software kan ik een export opvragen van de meest recente versie van de applicatie om zo ook te garanderen dat op elk systeem dezelfde versie staat. De bestanden zijn al getest in de testomgeving met de juiste database en daar heb ik geen fouten gevonden. Zodra de applicatie overgezet is naar de acceptatieomgeving kan ik de manager de mogelijkheid bieden om te testen met meerdere medewerkers tegelijkertijd. In de tijd dat de manager zijn testen aan het uitvoeren is kan ik de eis "Het automatisch aanmaken van rapportages in Excel formaat." uitwerken.

Het uitrollen van de applicatie op de acceptatieomgeving leverde een probleem op met de firewall die actief was. Dit zorgt ervoor dat ik de applicatie niet met meerdere gebruikers kan laten testen. Dit is nodig om de opdrachtgever de mogelijkheid tot een acceptatietest te bieden. Om deze blokkade op te laten heffen zal ik verzoek doen bij het IDC welke binnen één week afgehandeld moet zijn. In deze week richt ik mij daarom op de Excel eis om zo min mogelijk tijd verloren te laten gaan.

Om deze eis te laten werken moet het systeem elke week, maand, kwartaal en jaar een rapportage aanmaken. De weekrapportage gaat over maandag tot en met vrijdag. De maandrapportage van de 1e tot en met de 28e, 29e, 30e of 31e van de maand. De kwartaalrapportage van de 1e van de 1e maand in dat kwartaal tot en met de 30e of 31e van de 3e maand in dat kwartaal. De

[37] http://tortoisesvn.net/

jaarrapportage van de 1e van de 1e maand tot en met de 31e van de 12e maand. Het spreekt voor zich dat het om deze periodes gaat, het inzichtelijk maken ervan legt echter het probleem bloot van de frequentie. Er is niet op voorhand vast te leggen op welke dagen je deze rapportages wil laten creëren zonder een kalender voor bijvoorbeeld vijf jaar op te nemen. De verschillende periodes zal ik dynamisch moeten achterhalen en dan de rapportagefunctie laten starten. Het contact centrum heeft openingstijden van 08:00 tot 18:00 en de rapportages binnen die tijden creëren is niet wenselijk, dat zou kunnen leiden tot verstoringen. Om dit te voorkomen heb ik ervoor gekozen om een geplande taak aan te maken die elke avond om 22:00 gestart zal worden. Deze taak zal een batchbestand[38] aanroepen welke in de volgende map van de applicatie staat: "C:\xampp\htdocs\mrt\protected\".
In dit bestand zal ik de command-line versie van Yii (Yiic) aanroepen met het volgende commando "Go". Dit commando kijkt in de map "C:\xampp\htdocs\mrt\protected\commands" of daar PHP bestanden staan om uit te voeren.

De reden om deze taak elke dag uit te voeren, is om uit te sluiten dat er een fout kan optreden door alles vast te zetten in de applicatie. Nu zal er elke dag gekeken worden naar de logica die ik moet verwerken in het bestand in de map " C:\xampp\htdocs\mrt\protected\commands". Deze logica moet bepalen of het een dag is waarop een applicatie gemaakt moet worden.

De logica bouw ik op in de orde van grootte van de tijdsperiode. De logica kijkt eerst welke dag het vandaag is. Met deze waarde in het geheugen is de eerste controle of het een zaterdag is. Dan is er een controle of het de eerste van een maand is. Dan is er een controle of het de eerste van een kwartaal is. De laatste controle kijkt of het de eerste van de eerste maand is. Omdat zaterdag altijd dezelfde dag in de week is kan de weekrapportage zich daarop baseren. De logica zet de datum van die zaterdag om in een getal waarvan je altijd vijf (dagen) op kan verminderen. Dit getal zet ik daarna weer om in een datum en daarmee kan ik de database aanroepen. Dat ziet er als volgt uit.

```
//haalt de gegevens van de dag op voor de maand/kwartaal/jaren
$today = getdate();

//functie om de begindatum van deze week te krijgen en het weeknummer als het
zaterdag is om de rapportage van de medewerkers aan te maken
if($today['weekday'] == 'Saturday'){
        //geeft het huidige weeknummer terug aan $week
        $week = date('W', $today['0']);

        //datum van vandaag
        $vandaag = date("N");
```

[38] http://nl.wikipedia.org/wiki/Batchbestand

```
//komt uit op 5 als het zaterdag is maar als de rapportage naar een andere dag
zal veranderen dan kan je dit nog steeds gebruiken
$dagen_vanaf_maandag = $vandaag - 1;

//zet hier de datum van afgelopen maandag in
$maandag = date("d-m-Y", strtotime("- {$dagen_vanaf_maandag} Days"));

print_r("Het weeknummer van deze weekrapportage is ". $week."\n");
print_r("De datum van deze maandag is ". $maandag."\n");
//logica gaat hieronder nog verder in de applicatie
```

afbeelding 35 diensten rapportage sql functie

De reden dat ik alleen de datum van die maandag wil weten en doorsturen is omdat volgens de documentatie van PostgreSQL er een functie is die "date_trunc('week', *attribuut in de database*) heet. Deze functie zorgt ervoor dat de query kijkt naar alle dagen van die week en daarvan het resultaat geeft. Ditzelfde doe ik voor de maanden, kwartalen en jaren. Hiermee is het niet nodig om een einddatum vast te stellen van die periode. Daardoor voorkom ik het risico op verkeerde datums, het zal altijd de eerste dag van een week, maand, kwartaal en jaar zijn waarmee PostgreSQL de juiste waarden erbij kan zoeken. Ik heb hiervoor gekozen omdat er al een functie bestaat, dan is het niet nodig om deze te ontwerpen. Het zelf ontwerpen zou een verlies van tijd betekenen wat niet wenselijk is.

De logica om de rapportages op te starten moet de database gaan bevragen om de cijfers te krijgen voor de rapportages. Deze rapportages zijn anders dan de rapportages die de managers nu kunnen opvragen. Dit komt omdat deze rapportages een samenvoeging vormen van de rapportages die de applicatie nu kan aanmaken in de browser. Het gaat nu om combinaties van al deze cijfers. Het verwerken hiervan in Excel zal aan een bepaalde opbouw voldoen, afhankelijk van de rapportage. Deze opbouw is vastgezet door het contact centrum. De reden hiervoor is uniformiteit richting de andere diensten, zij weten daardoor welke gegevens waar staan in de rapportage. De rapportages geven een cijfer van het totaal klantcontact (voor alle diensten of alleen die dienst) en daaronder de verdeling van deze contacten. De verdeling gaat over de binnenkomst, dienst (bij alle), onderwerp (bij de dienst), behandelwijze (intern/extern) en staat in procenten weergegeven. Om deze rapportages in Excel mogelijk te maken zal ik gebruik maken van de plugin PHPExcel[39]. Deze plugin kan ik samen laten werken met Yii en zorgt ervoor dat ik de Excel documenten kan opmaken zoals het contact centrum dat graag zou willen. Mocht er in de toekomst een verandering komen in het uiterlijk van de rapportages dan is dit makkelijk aan te passen in de structuur.

[39] http://phpexcel.codeplex.com/

Het opbouwen van de bestanden doe ik hier ook weer met arrays zoals ik ook bij de rapportages in de browser doe. Het verwerken in arrays zorgt voor een gelijke opbouw van elke soort rapportage wat goed om te zetten is naar beeld. In de verwerking van deze gegevens heb ik wel een wijziging aangebracht door het model Rapporteer alle data in één array te laten zetten in plaats van verschillende losse arrays. Hierdoor geven de functies die de dienstrapportages opvragen multidimensionale arrays door. Dit ziet er als volgt uit.

```
public function getRapportDienstMaand($datumtijd1)
	{

	//Queries voor de dienst rapportage
	//hiermee haal je de diensten en hun totalen op voor de periode van de vorige maand
	$aantal = Yii::app()->db->createCommand()
	->selectDistinct('dienst.naam as naam, dienst.idnummer as dienst_id,
	count(dienst.idnummer) as aantal')
	->from('registratievraagklacht, dienst')
	->where('date_trunc(\'month\', datumtijd) = \'' .$datumtijd1. '\' and
	registratievraagklacht.dienst_id = dienst.idnummer')
	->group('dienst.naam, dienst.idnummer')
	->order('dienst.naam asc')
	->query();

	$teller_arrDiensten = 0;
	$arrDiensten = array();

	//hiermee sla je alle diensten op volgens een vaste volgorde waardoor het uitlezen
	gemakkelijker is
	foreach($aantal as $t){
	$arrDiensten[$teller_arrDiensten][0] = $t['naam'];
	$arrDiensten[$teller_arrDiensten][1] = $t['dienst_id'];
	$arrDiensten[$teller_arrDiensten][2] = $t['aantal'];
	$teller_arrDiensten ++;
	}

	//De volgende for gaat elke dienst af die in het array staat en vraagt 3 rapportages per
	dienst op om alle cijfers te krijgen
	for ($row = 0; $row < (count($arrDiensten)); $row++){

	//groene gedeelte in cijfers niet in %
	$groen = Yii::app()->db->createCommand()
	->selectDistinct('binnenkomst.naam as medium, count(binnenkomst_id) as aantal')
	->from('dienst, binnenkomst, registratievraagklacht')
	->where('date_trunc(\'month\', datumtijd) = \'' .$datumtijd1. '\' and
	registratievraagklacht.binnenkomst_id = binnenkomst.idnummer and
	registratievraagklacht.dienst_id = '.$arrDiensten[$row][1].' ')
	->group('dienst.naam, binnenkomst.naam, binnenkomst_id')
	->order('binnenkomst.naam ASC')
	->query();

	//geel in cijfers niet in %
```

```
$geel = Yii::app()->db->createCommand()
->selectDistinct(' onderwerp.naam as onderwerp, count(onderwerp_id) as aantal')
->from('dienst, onderwerp, registratievraagklacht')
->where('date_trunc(\'month\', datumtijd) = \'' .$datumtijd1. '\' and
registratievraagklacht.dienst_id = '.$arrDiensten[$row][1].' and
registratievraagklacht.onderwerp_id = onderwerp.idnummer and dienst.idnummer =
'.$arrDiensten[$row][1].'')
->group('onderwerp.naam')
->order('onderwerp.naam ASC')
->query();

//oranje in cijfers niet in %
$oranje = Yii::app()->db->createCommand()
->selectDistinct('behandelwijze.naam as naam, count(behandelwijze_id) as aantal,
behandelwijze.intern')
->from('dienst, behandelwijze, registratievraagklacht')
->where('date_trunc(\'month\', datumtijd) = \'' .$datumtijd1. '\' and behandelwijze_id =
behandelwijze.idnummer and dienst.idnummer = '.$arrDiensten[$row][1].' and
registratievraagklacht.dienst_id = '.$arrDiensten[$row][1].'')
->group('dienst.naam, behandelwijze.naam, behandelwijze.intern')
->order('behandelwijze.intern DESC')
->query();

//de verdeling per binnenkomst in het array verwerken in procenten
$arrDiensten[$row][] = 'binnenkomst';
        foreach($groen as $t){
                $arrDiensten[$row][] = $t['medium'];
                $arrDiensten[$row][] = round((($t['aantal'] * 100)
                /$arrDiensten[$row][2]),4);
                }

//de verdeling per onderwerp in het array verwerken in procenten
$arrDiensten[$row][] = 'onderwerp';
        foreach($geel as $t){
                $arrDiensten[$row][] = $t['onderwerp'];
                $arrDiensten[$row][] = round((($t['aantal'] * 100)
                /$arrDiensten[$row][2]),4);
                }

$intern = 0;
$extern = 0;
$arrDiensten[$row][] = 'behandelwijze';

//de verdeling intern/extern in het array verwerken in procenten
        foreach($oranje as $t){
                //aantal en intern
                if($t['intern'] == true){
                $intern += $t['aantal'];
                        }
                if($t['intern'] == false){
                $extern += $t['aantal'];
                        }
                }
```

```
                    $arrDiensten[$row][] = round((($intern * 100)
                    /$arrDiensten[$row][2]),4);
                    $arrDiensten[$row][] = round((($extern * 100)
                    /$arrDiensten[$row][2]),4);
            }

            //stuur het multidimensionale array terug
            return $arrDiensten;
    }
```

afbeelding 36 diensten rapportage sql functie

Het verwerken van de maand rapportages van de diensten tot Excel bestanden gaat met behulp van PHPExcel. Dit is een zeer veelzijdige extensie die aan Yii toegevoegd kan worden. Deze extensie kan de applicatie Adobe pdf, Word, Excel en HTML bestanden lezen en schrijven. Door de extensie in de map "mrt/protected/extensions" te plaatsen en de instructies te volgen van een "How to use[40]" handleiding op de Yii website kan PHPExcel gebruikt worden. De functie voor de rapportages zal, afhankelijk van de datum, een aantal aanroepen naar het model Rapporteer versturen. Tijdens het testen van deze functies liep ik tegen een probleem op wat te herleiden was naar de "How to use" handleiding. Hier is aangegeven dat je het volgende moet opnemen in de code " /* Turn off our amazing library autoload */ spl_autoload_unregister(array('YiiBase','autoload'));". Dit zorgt ervoor dat ik geen modellen meer kan aanroepen totdat ik deze autoload weer registreer. Omdat ik voor het overzicht de queries en PHPExcel logica per rapportage wil opbouwen is dit geen werkbare situatie. Als bijkomend risico kan de batch verwerking daardoor vastlopen en open blijven staan zonder de autoload weer te registreren. Dit kan veroorzaken dat de applicatie niet meer zal reageren bij de gebruikers.
Ik wil deze fouten voorkomen en volgens de FAQ van PHPExcel is hier een oplossing voor. Er is een bestand "Autoloader.php" beschikbaar[41] welke zorgt voor een goede samenwerking van de Yii autoloader en de PHPExcel autoloader. Hierdoor kan ik terwijl ik PHPExcel gebruik ook de modellen van Yii aanroepen. Deze autoloader.php voorkomt conflicten (minder risico) en dat scheelt veel werk in het omzeilen van de logica die er als volgt uit ziet.

```
//voor per diensten is het de dienstnaam versie
$rapportageDienstMaand = Rapporteer::model()->getRapportDienstMaand($datumtijd1);

//controleer of er gegevens zijn zo niet dan maakt de applicatie geen rapportage
if(count($rapportageDienstMaand) > 1)
        {

        //doorloop de gegevens van de dienst
```

[40] http://www.yiiframework.com/wiki/101/how-to-use-phpexcel-external-library-with-yii/
[41] http://www.mrsoundless.com/post/2011/04/23/PHPExcel-in-Yii.aspx

```
for ($row = 0; $row < (count($rapportageDienstMaand)); $row++){

// maak een nieuw PHPExcel object
echo date('H:i:s') . " Create new PHPExcel object\n";
$objPHPExcelDienstMaand = new PHPExcel();

// Set eigenschapppen van het excelbestand
echo date('H:i:s') . " Set properties\n";
$objPHPExcelDienstMaand->getProperties()->setCreator("Contact Centrum 14070");
$objPHPExcelDienstMaand->getProperties()->setLastModifiedBy("Contact Centrum
14070");
$objPHPExcelDienstMaand->getProperties()->setTitle('Maandrapportage
'.$rapportageDienstMaand[$row][0].' Maand '.$maand.' jaar '.$jaar);
$objPHPExcelDienstMaand->getProperties()->setSubject('Maandrapportage
'.$rapportageDienstMaand[$row][0].' Maand '.$maand.' jaar '.$jaar);
$objPHPExcelDienstMaand->getProperties()->setDescription('Maandrapportage
'.$rapportageDienstMaand[$row][0].' Maand '.$maand.' jaar '.$jaar);


// Voeg data toe aan de sheet
echo date('H:i:s') . " Add some data\n";
$objPHPExcelDienstMaand->setActiveSheetIndex(0);
$objPHPExcelDienstMaand->getActiveSheet()->getColumnDimension('A')-
>setAutoSize(true);
$objPHPExcelDienstMaand->getActiveSheet()->getColumnDimension('B')-
>setAutoSize(true);
$objPHPExcelDienstMaand->getActiveSheet()->SetCellValue('A1', 'Maand '.$maand);
$objPHPExcelDienstMaand->getActiveSheet()->SetCellValue('A3',
$rapportageDienstMaand[$row][0]);
$objPHPExcelDienstMaand->getActiveSheet()->SetCellValue('B3', 'Aantal');
$objPHPExcelDienstMaand->getActiveSheet()->SetCellValue('A4', 'Totaal aantal
klantcontact');
$objPHPExcelDienstMaand->getActiveSheet()->SetCellValue('B4',
$rapportageDienstMaand[$row][2]);

//gebruik deze waarden om de onderwerpen en binnekomsten variabel op te kunnen
bouwen in Excel
$waardeOnderwerp = array_keys($rapportageDienstMaand[$row], "onderwerp");
$waardeBinnenkomst = array_keys($rapportageDienstMaand[$row], "binnenkomst");



$objPHPExcelDienstMaand->getActiveSheet()->SetCellValue('A6', 'Verdeling
klantcontact');
$objPHPExcelDienstMaand->getActiveSheet()->SetCellValue('B6', 'procent');

//vul de gegevens van de binnenkomsten dynamisch
$teller_excel1 = 7;
for ($row1 = ($waardeBinnenkomst[0] + 1); $row1 < $waardeOnderwerp[0];
$row1++){

$objPHPExcelDienstMaand->getActiveSheet()->SetCellValue('A'.$teller_excel1,
$rapportageDienstMaand[$row][$row1]);
```

```
$objPHPExcelDienstMaand->getActiveSheet()->SetCellValue('B'.$teller_excel1,
$rapportageDienstMaand[$row][($row1 = $row1 +1)]);
$teller_excel1 ++;
        }


$teller_excel1 ++;
$objPHPExcelDienstMaand->getActiveSheet()->SetCellValue('A'.$teller_excel1, 'Verdeling
onderwerpen');
$objPHPExcelDienstMaand->getActiveSheet()->SetCellValue('B'.$teller_excel1, 'procent');
$teller_excel1 ++;

//vul de gegevens van de onderwerpen dynamisch
for ($row2 = ($waardeOnderwerp[0] + 1); $row2 <
(count($rapportageDienstMaand[$row])-4); $row2++){

$objPHPExcelDienstMaand->getActiveSheet()->SetCellValue('A'.$teller_excel1,
$rapportageDienstMaand[$row][$row2]);

$objPHPExcelDienstMaand->getActiveSheet()->SetCellValue('B'.$teller_excel1,
$rapportageDienstMaand[$row][($row2 = $row2 +1)]);
$teller_excel1 ++;
                        }

//vul de gegevens van de behandelwijze (intern of extern) dynamisch
$teller_excel1 ++;
$objPHPExcelDienstMaand->getActiveSheet()->SetCellValue('A'.$teller_excel1, 'Verdeling
intern/extern');
$objPHPExcelDienstMaand->getActiveSheet()->SetCellValue('B'.$teller_excel1, 'procent');
$teller_excel1 ++;
$objPHPExcelDienstMaand->getActiveSheet()->SetCellValue('A'.$teller_excel1, 'intern');
$objPHPExcelDienstMaand->getActiveSheet()->SetCellValue('B'.$teller_excel1,
$rapportageDienstMaand[$row][(count($rapportageDienstMaand[$row]) - 2)]);
$teller_excel1 ++;
$objPHPExcelDienstMaand->getActiveSheet()->SetCellValue('A'.$teller_excel1, 'extern');
$objPHPExcelDienstMaand->getActiveSheet()->SetCellValue('B'.$teller_excel1,
$rapportageDienstMaand[$row][(count($rapportageDienstMaand[$row]) - 1)]);
$teller_excel1 ++;

// Hernoem het blad naar de maand
echo date('H:i:s') . " Rename sheet\n";
$objPHPExcelDienstMaand->getActiveSheet()->setTitle('maand '.$maand);

// Sla het Excel 2007 bestand op
echo date('H:i:s') . " Write to Excel2007 format\n";
$objWriter = new PHPExcel_Writer_Excel2007($objPHPExcelDienstMaand);
//vul hieronder de schijflocatie in waar het bestand moet komen te staan en met welke
naam
$objWriter->save('G:/DPZ/KCC/Rapportages/Maandrapportage
'.$rapportageDienstMaand[$row][0].' Maand '.$maand.' jaar '.$jaar.'.xlsx');

// Echo done
```

```
        echo date('H:i:s') . " Done writing file.\r\n";

                }

}
```

Afbeelding 37 diensten rapportage naar Excel functie

Deze functies en omzettingen naar Excel zijn nu aanwezig in de applicatie. Er bestaan zeven rapportages:

- Weekrapportage_medewerkers
- Maandrapportage 14070
- Maandrapportage per dienst
- Kwartaalrapportage 14070
- kwartaalrapportage per dienst
- Jaarrapportage 14070
- Jaarrapportage per dienst

De rapportages 14070 zijn van het contact centrum algemeen. Dit is het nummer waarop het contact centrum bereikbaar is. De rapportage is er omdat niet alleen de diensten willen weten hoeveel contact er voor hun dienst is geweest. Dit pakket aan rapportages moet elke relevante periode aangemaakt worden en middels de combinatie geplande taak, Yiic en PHPExcel is dit mogelijk gemaakt.

Door dit onderdeel in de acceptatieomgeving op te nemen zijn alle eisen en wensen voor de initiële oplevering aanwezig. Zodra het probleem met de firewall opgelost was had de opdrachtgever de mogelijkheid om te testen. Mijn eigen testen waren al met positief resultaat afgerond in de testomgeving. De opdrachtgever heeft een week kunnen testen met de applicatie en was tevreden met het resultaat van zowel de simpliciteit die de registreerpagina en rapporteerpagina bieden als de uitvoer van de rapportages in de browser. De rapportages die automatisch aangemaakt worden heb ik als voorbeeld aangeleverd om deze ook te laten controleren. Hier zal nog nazorg voor zijn vanuit de afdeling functioneel beheer DPZ zodra er een nieuwe automatische rapportage verkeerd blijkt te zijn. Om vast te leggen dat de opdrachtgever akkoord gaat met de opgeleverde applicatie heb ik de opdrachtgever een document laten ondertekenen waarin de geïmplementeerde eisen uiteengezet staan. Dit document zal bij de afdeling functioneel beheer DPZ in het archief bewaard worden en bij en volgende release aangepast worden naar de nieuwe situatie.

De opdrachtgever wilde de nieuwe applicatie snel geïnstalleerd hebben in de productieomgeving. Tijdens het maken van de afspraak wanneer dit zal zijn gaf de opdrachtgever nog wel te kennen dat hij er vanuit ging dat het uiterlijk van de applicatie in de productieomgeving in lijn zou zijn met de gemeentelijke standaard. Dit is een eis die nooit kenbaar is gemaakt, maar ik heb er zelf ook

niet expliciet om gevraagd. Om te voorkomen dat de opdrachtgever met een ontevreden gevoel de applicatie in gebruik zou nemen heb ik deze aanpassing nog verwerkt in de applicatie voor de overzetting naar de productieomgeving. Deze eis heb ik daardoor niet kunnen verwerken in de lijst met geïmplementeerde eisen. In de volgende release zal deze eis officieel opgenomen worden.

6. Het in gebruik nemen van de applicatie.

De manager heeft de applicatie aan een gebruikersacceptatietest onderworpen en een akkoord afgegeven voor een installatie in de productieomgeving. Hierbij is wel een extra eis toegevoegd omtrent het uiterlijk van de applicatie. Deze is verwerkt in de test en acceptatieomgeving. De applicatie moet nu volledig overgezet worden naar de productieomgeving. Als dit gedaan is zal ik alle relevante documenten en code van de applicatie overdragen aan het contact centrum en de afdeling functioneel beheer DPZ.

6.1 Het uitrollen van de applicatie.

De applicatie kan in gebruik worden genomen en moet daarvoor overgezet worden naar de productieomgeving. In deze omgeving zal het uiterlijk overeen moeten komen met de huisstijl van de gemeente Den Haag. Alle huidige en historische gebruikers dienen aanwezig te zijn in de database van de applicatie en de url verwijzing naar de applicatie dient aangemaakt worden. De historische data dient aanwezig te zijn vanaf 01-01-2009.

Al deze eisen zijn nu verwerkt in de acceptatieomgeving en getest door de opdrachtgever of zij voldoen. Dit bleek het geval en daarom kan ik de applicatie één op één overzetten naar de productieomgeving. Omdat ik de applicatie al vaker van de test naar acceptatieomgeving had overgezet voorzie ik geen verassingen tijdens deze implementatie. De applicatie in de productieomgeving krijgt een directe kopie van de applicatie code en database in de acceptatieomgeving. Om het probleem met de firewall in de acceptatieomgeving niet te krijgen heb ik op voorhand het IDC gevraagd dit probleem op te lossen om zo tijd te besparen.

Voordat ik de applicatie over kan zetten zal ik nog een aanpassing in het uiterlijk maken. Ik wil de verwerking van de huisstijl in de applicatie laten doorwerken in alle pagina's. Yii werkt met CSS[42] om de opmaak van de applicatie te verwerken. Ik heb nooit gewerkt met CSS, alleen met HTML waar deze aanpassingen ook mogelijk zijn per pagina. Ik heb de keuze gemaakt om alles aan te passen in de CSS. De reden daarvoor is dat het nu een mooi herbruikbaar resultaat geeft en in de toekomst zal dit betekenen dat alle andere pagina's ook deze huisstijl kunnen verwerken door de CSS te hergebruiken. Omdat er weinig tijd over was heb ik een snelcursus CSS[43] via de website w3schools.com doorlopen. Hierdoor had ik snel de kennis die ik nodig had om de relevante stukken in de CSS aan te

[42] http://nl.wikipedia.org/wiki/Cascading_Style_Sheets
[43] http://www.w3schools.com/css/

passen. Met behulp van Firebug[44] kon ik de elementen in CSS opsporen die verantwoordelijk waren voor de kleuren en layout van de applicatie. Deze aanpassing in de applicatie viel niet onder de gestelde eisen maar daar was ik deels verantwoordelijk voor door ze niet expliciet op te laten nemen. Alhoewel dit extra tijd heeft gekost, welke kostbaar was, was het een aanpassing waarmee ik de opdrachtgever het gevoel gaf dat ik alles eraan had gedaan om deze opdracht tot een goed einde te brengen.

De applicatie is nu volledig in de huisstijl aangepast en kan opgeleverd worden in de productieomgeving. De applicatie zal ik op een 1e van een nieuwe maand actief zetten in verband met het gelijk lopen van de registraties in de oude applicatie. Het installeren van de applicatie kan eerder plaatsvinden, waardoor de manager nog een mogelijkheid heeft om een gebruikersacceptatietest uit te voeren. Deze test zal niet bepalen of de applicatie in gebruik zal worden genomen, dat is al besloten tijdens de gebruikersacceptatietest in de acceptatieomgeving. Er is afgesproken, tijdens het beta-testen, dat de verantwoording van de applicatie na de gebruikersacceptatietest in de acceptatieomgeving bij de afdeling functioneel beheer DPZ zal liggen. Mocht de applicatie niet voldoende functioneren dan neemt de afdeling functioneel beheer DPZ dit op zich. Na het actief zetten van de applicatie zal er nog één gegevensconversie plaatsvinden om de meest actuele data uit de oude applicatie over te hevelen.

De applicatie heb ik op vrijdag 23 september geïnstalleerd en getest in productie. Als dit niet zou werken, dan had ik geen tijd meer om het te repareren. Ik was wel voorbereid op problemen door het gebruik van mijn eigen testomgeving. Vanwege mijn afspraak met de manager en de afdeling functioneel beheer DPZ over de verantwoordelijkheid na de release, was ik ook gerust gesteld dat de afhandeling hiervan in goede handen was. De manager heeft zelf getest in de week na de installatie en alles is geaccepteerd zoals het in de eisen verwerkt was. Dit heeft een groot deel te maken met de testomgeving die ik thuis had nagemaakt. Door dit probleem zo te ondervangen toonde ik de opdrachtgever wederom dat ik alles wilde doen om de opdracht tot een goed einde te brengen. Dit heeft ervoor gezorgd dat de initiële release mogelijk gemaakt wordt per de 1e van oktober. Ik zal in de eerst week erna ondersteuning bieden samen met de afdeling functioneel beheer DPZ.

6.2 Het overdragen van de applicatie.

De afdeling functioneel beheer DPZ zal alle documenten krijgen die als onderbouwing en basis dienen van de huidige release van de applicatie. Zij zullen ook de huidige revisie van de applicatie en database aangeleverd krijgen. Deze

[44] http://getfirebug.com/

zijn al geïnstalleerd op de productieomgeving. Het contact centrum zal de handleidingen krijgen voor de managers en agents. De managers kunnen de documentatie van de applicatie opvragen bij de afdeling functioneel beheer DPZ. Al deze handelingen zorgen ervoor de applicatie een brede ondersteuning heeft binnen de gemeente Den Haag. Dit was met de oude applicatie niet het geval en door de applicatie nu intern te laten beheren wilde het contact centrum dit voorkomen.

De applicatie komt in beheer bij de afdeling functioneel beheer DPZ en zal eigendom zijn van het contact centrum. De afdeling functioneel beheer DPZ valt onder dezelfde dienst als het contact centrum en geeft technische ondersteuning aan het contact centrum. Als er een wijziging moet worden gemaakt in de applicatie zal dit altijd via de afdeling functioneel beheer DPZ aangevraagd moeten worden. Kleine wijzigingen in bijvoorbeeld gebruikers of opties zullen direct verwerkt worden. Grote wijzigingen zoals een nieuwe rapportagevorm zal een RFC (Request For Change[45]) moeten worden, waarna de afdeling functioneel beheer DPZ een onderzoek kan doen naar de haalbaarheid, impact en de benodigde tijd. Door deze constructie kan het contact centrum direct ondersteuning krijgen op de applicatie mocht dit nodig zijn.

Ik heb de documentatie en applicatie voor de ingebruikname digitaal overgedragen, binnen de gemeente Den Haag geldt namelijk een papierarm beleid. De documentatie omvat de volgende documenten die ook als bijlage aanwezig in dit verslag.

- Plan van Aanpak
- Inception rapport
- Elaboration rapport
- Construction rapport
- Transition rapport
- Geïmplementeerde wensen en eisen lijst
- Handleidingen (onderdeel van het transitionrapport)
- Applicatie in code vorm (op cd)

[45] http://nl.wikipedia.org/wiki/Change_request

Deel III: De evaluatie

In deel III is er de ruimte om te reflecteren op het doorlopen proces en het product wat daar uit voort komt. Er is een opdeling gemaakt in de evaluatie. Het proces zal eerst besproken worden, dan het product en als laatste de beroepstaken.

7. Evaluatie

7.1. Procesevaluatie

Het proces is met behulp van UP voltooid. Tijdens deze periode ben ik erachter gekomen dat alleen UP gebruiken misschien niet voldoende is om een applicatie goed en snel te kunnen ontwikkelen. Op sommige punten is het handig om in een hele korte periode al een prototype te kunnen bouwen. Omdat dit natuurlijk niet zonder documentatie kan een gulden middenweg een oplossing bieden. Door zelf een prototype te maken ging ik voor een deel uit het pad van UP. Het komt er op neer dat ik toch een mengeling van de UP en Agile methodes toegepast heb, AUP[46] komt in de buurt van het proces dat ik doorlopen had.

De tijdverdeling in het proces was evenwichtig opgebouwd in de planning zodat de opdrachtgever en ik precies wisten waar wij aan toe waren. Toch bleek gaandeweg dat het opstellen van een planning lastig is in dagen. Dit komt omdat de planning moeilijk in dagen te verdelen is als dit project niet het enige project is waar je verantwoordelijk voor bent. Het vertalen naar werkuren paste beter bij de planning en mijn werkschema. Op het moment dat er geen uren beschikbaar zijn dan staat de ontwikkeling stil terwijl volgens de planning het werk door moet gaan. Hier kwam ik later achter. Ik had geen uren beschikbaar door extra taken die bovenop mijn dagelijkse werkzaamheden waren gekomen. Het aanpassen van de globale planning van dagen naar uren was hiervan een logisch gevolg. Hierdoor was het voor mij beter mogelijk om de gestelde uren te halen omdat ik ze kon verdelen over meerdere weken in plaats van 40 uur per week. De verdeling in werkweken gaf minder druk dan kalenderweken.

Dit heeft ook gevolgen gehad voor de eisen die de opdrachtgever stelde aan de applicatie. De opdrachtgever vond mijn participatie in de andere projecten belangrijker dan de externe datakoppelingen waardoor deze eis in de wacht is gezet voor een volgende release. De volgende keer zal ik explicieter de tijden vastleggen waarin ik kan werken aan een opdracht om een herhaling van deze situatie zo veel mogelijk te voorkomen. Omdat nu de werkzaamheden al door de opdrachtgever toebedeeld waren aan mij moest ik werken met de tijd die ik overhad.

Het vaststellen van de eisen en de wensen op de manier zoals gedaan heeft ervoor gezorgd dat de wensen niet aan bod zijn gekomen tijdens deze opdracht. Omdat het MT het namelijk niet eens kon worden wat een belangrijk wens was. Deze oplossing was planningtechnisch goed omdat er hierdoor tijd vrijkwam om andere onderdelen te verbeteren of te maken. Een dergelijke toepassing kan in

[46] http://nl.wikipedia.org/wiki/Agile_Unified_Process

de toekomst ook zijn vruchten afwerpen. Het zorgt ervoor dat alle neuzen dezelfde kant op moeten wijzen en creëert een draagvlak voor die wens(en).

Door mijn ervaringen met het verkeerd plannen heb ik er voor gezorgd dat er keuzes gemaakt werden om de planning niet verder uit te laten lopen. De opdrachtgever heeft mij een aantal keuzes zelf laten maken zoals het opzetten van een eigen testsysteem om later te voorkomen dat de applicatie niet werkt op de productieomgeving. De keuze om juist een eis te verwerken die niet opgesteld was tijdens het begin van de opdracht heb ik gemaakt om de opdrachtgever een mooi product te geven welke goed past binnen de organisatie. In de toekomst zal ik wel het uiterlijk van een applicatie expliciet opnemen in de eisen om discussies (en mogelijk extra werk) te voorkomen.

Het proces doorlopen was niet altijd even makkelijk door de tijdsdruk maar dit heeft gelukkig geen impact gehad op mijn werk en het resultaat dat ik opleverde. Ik ben er altijd voor de volle 100% voor gegaan en wilde ook een werkend en goedgekeurd resultaat afleveren. De volgende keer wil ik de mogelijkheden van AUP gaan verkennen en gebruiken Een striktere planning maken met gereserveerde uren is ook een onderdeel wat ik wil toepassen.

7.2. Productevaluatie

De opgeleverde applicatie met database voldoet volgens de meest recente eisen aan alle gestelde eisen. De opdrachtgever heeft hiervoor ook zijn handtekening gezet, dat gaf mij nogmaals het idee dat alles voldoet. De in het begin gestelde eis voor de koppeling met de externe databronnen is niet geïmplementeerd in deze versie van de applicatie wegens andere verplichtingen bij het contact centrum. Door de opbouw van de huidige vorm van de automatische rapportages zal bij het ontsluiten van deze databronnen het makkelijker zijn om deze te integreren in de rapportages. Dat is het voordeel van de manier van opbouw die ik gebruikt heb. Hier is de opdrachtgever tevreden mee omdat dit betekend dat er nog steeds een aansluiting kan plaatsvinden. Omdat ik deze eis niet kon verwerken in de applicatie ben ik blij dat ik wel de eis met betrekking tot de huisstijl in de applicatie heb kunnen verwerken als extra eis.

De applicatie neemt nu veel werk uit handen. Elke periode dat er rapportages nodig zijn neemt de applicatie deze rapportages op zich, waar voorheen een medewerker dit moest doen. Het contact centrum kan nu € 7520,- per jaar besparen op personeelskosten. Deze automatische rapportages zorgen ook voor consistente rapportages, de berekening staat vast en de applicatie zal altijd op dezelfde manier rekenen. Om deze totaalcijfers te kunnen ondersteunen is het altijd mogelijk om handmatig rapportages aan te maken in het rapporteerscherm.

Het werken met de applicatie voldoet aan het KISS[47] principe (Keep it Simple & Straightforward). Er kan altijd iets fout gaan tijdens het registreren en rapporteren maar dit is zo veel mogelijk ingeperkt met de foutcontrole. Iedereen die om kan gaan met een computer en internetpagina's zou de applicatie kunnen gebruiken. De applicatie is gebruiksvriendelijk, wat het doel vanaf het begin ook was. De agents kunnen nu met minder klikken en scrollen registreren. De managers kunnen met minder klikken en typen rapporteren. Er hoeven geen handleidingen bij de hand gehouden worden, de applicatie zal melden wat gedaan moet worden en wat er mis is als iets fout gaat. Er kan beheer plaatst vinden wat al een verbetering is ten opzichte de oude situatie. Dit beheer is ook zo ingeregeld dat er weinig fout kan gaan en als het wel fout gaat dan komt er een duidelijk melding voor in beeld.

De applicatie is snel, stabiel, te beheren en correct in data in en uitvoer. De manager is tevreden met resultaat en wilde zo snel mogelijk met de applicatie gaan werken. Dit samenvattend ben ik trots op de afgeleverde applicatie.

Naast de applicatie zijn er documenten (tussenproducten) gemaakt die de basis van deze applicatie vormen. Deze zijn voor, tijdens en na het ontwikkelen gemaakt. Ze zijn gemaakt om de structuur van de applicatie te beschrijven, de verschillende keuzes die gemaakt zijn te verduidelijken, de eventuele aanpassing die verwerkt zijn te beschrijven en om de nieuwe gebruikers en beheerders te informeren wat voor mogelijkheden de applicatie bezit. Alle documenten zijn opgeleverd aan de opdrachtgever. Ik ben tevreden met de documenten die ik in deze periode heb gemaakt.

7.3. Beroepstaken

2.1 Opstellen gegevensmodel voor database (niveau 3)

Voor het gegevensmodel heb ik in kaart gebracht welke onderdelen belangrijk zijn voor een registratie en deze verwerkt in verschillende modellen. Deze gegevens heb ik in de derde normaalvorm verwerkt. Dit gegevensmodel is zo gemaakt dat het ondersteuning geeft aan de historische data die al aanwezig is. De historische objecten die overbodig zijn in de nieuwe applicatie zullen niet meer voorkomen in het nieuwe model, het ontwerp van het nieuwe model zorgt dat deze niet meer nodig zijn. Later heb ik de modellen aangepast omdat er een vorm van redundantie in de database zat en dit past niet bij een genormaliseerde database. Bij deze aanpassing zijn ook weer verschillende modellen gemaakt.

2.2 Ontwerpen, bouwen en bevragen van een database (niveau 3)

[47] http://nl.wikipedia.org/wiki/KISS-principe

Het nieuwe gegevensmodel heb ik omgezet naar een vorm die geïmplementeerd is in het databasemanagementsysteem PostgreSQL. De keuze voor PostgreSQL is gemaakt aan de hand van een aantal eisen van de manager en mij (zie bladzijde 27). Bij het omzetten zijn constraints gebruikt om verschillende waardes te definiëren onder andere de primaire sleutels, standaard waardes, of een waarde NULL mag zijn en de onderlinge relaties tussen de tabellen voor bewaking van de kwaliteit en integriteit. Door slim gebruik van de combinatie Yii en PostgreSQL zorgen zij voor een bewaking van de invoer door de controleren of de gegeven waarde voldoet aan de gevraagde waarde.

Om de verschillende rapportages op te vragen maak ik gebruik van complexe queries. Deze queries zijn zo optimaal mogelijk gemaakt om de database zo snel mogelijk te laten werken. Een voorbeeld van een complexe query met de verwerking ervan staat op bladzijde 66 (afbeelding 36). Hier is de uitvoer van drie queries afhankelijk van een query die als basis dient. Deze vier queries zijn nodig om het juiste resultaat te verkrijgen in een rapportage voor één dienst. Do De database is ook stabiel bij meerdere gebruikers. Tijdens het opvragen van een rapportage in de browser kunnen er tegelijkertijd minimaal 40 agents registreren blijkt uit stresstesten. De PostgreSQL database kan dus meerder verbindingen aan, 100 connectie tegelijkertijd aan volgens de documentatie. PostgreSQL werkt volgens de ACID[48] regels wat zorgt voor betrouwbare transacties.

2.3 Uitvoeren gegevensconversie (niveau 3)

Het nieuwe gegevensmodel komt voort uit de bestaande historische database gecombineerd met de nieuwe data-invoer. Het omzetten van de data is met behulp van een database migratie tool gedaan waardoor alle types juist geconverteerd zijn naar de juiste tabellen en attributen. Er zijn een aantal conversies doorgevoerd in de id's van de verschillende tabellen. Dit is met behulp van SQL gedaan in de pgAdmin III applicatie en door versiebeheer te gebruiken was er geschiedenis aanwezig mocht het fout gaan. De database krijgt in de nieuwe applicatie automatisch aanvragen vanuit de applicatie om rapportages aan te maken. Omdat er geen controle meer plaatst zal vinden op deze rapportages heb ik dat gedaan met behulp van de oude applicatie. Er zijn verschillende rapporten aangemaakt en deze zijn vergeleken met de platte data in de nieuwe database. Op basis van deze controle kon ik de conclusie trekken dat het overzetten en de conversie (op een paar tikfouten na) juist verlopen was.

[48] http://nl.wikipedia.org/wiki/ACID

3.2 Ontwerpen systeemdeel (niveau 3)

Met behulp van StarUML heb ik de rapporteermodule ontworpen in UML. Deze module is als apart gedeelte opgenomen in de applicatie. Ik heb een apart datamodel gemaakt om de module te kunnen ondersteunen in de controle van de gegevens. Het scherm voor de rapportages blijft eenvoudig in het ontwerp en in lijn met de applicatie en eis dat iedereen er mee overweg moet kunnen. Het scherm is daarom overzichtelijk gemaakt en kan zonder scrollen gebruikt worden. Er zal een lijst actief gevuld worden als er een bepaalde rapportages gekozen is, dit voorkomt onnodige lijsten in beeld. De verdeling van de functionaliteiten is volgens de MVC structuur opgebouwd. Er is hergebruik aanwezig in de vorm van gebruikers controle. Deze vangt af of de gebruiker aanwezig mag zijn op die pagina. Voor het tonen van een lege rapportage is er één view voor alle rapportages wat hergebruik stimuleert. Vanuit elke toestand in het rapportagescherm kan je alle andere functionaliteiten kiezen. De module voor het rapporteren zal in de basis van een prototype opgenomen worden in het ontwikkel/test/acceptatiesysteem via versiebeheer. In hoofdstuk 5.2 staan veel details van het ontwerp.

3.3 Bouwen applicatie (niveau 3)

De basis van de applicatie is het Yii framework met een extensie (PHPExcel) voor de Excel rapportages. De applicatie is opgenomen in een subversiesysteem waarin ik elke verandering kon bijhouden. Tijdens het ontwikkelen is de MVC structuur aangehouden. Veel onderdelen heb ik hergebruikt in verschillende rapportages. Het traject is van een release doorloopt de ontwikkel/test/acceptatiesystemen waar via versiebeheer gegarandeerd is dat de situatie overal gelijk is. De gegevens die ingevoerd worden ondergaan een controle op fouten via een koppeling met de database.

Voor de automatische rapportages heb ik een combinatie gemaakt van geplande taken met een commandline bestand welke via een aantal algoritmes rapportages in Excel kan aanmaken. Voor het verwerken van de gegevens maak ik gebruik van dynamische (multidimensionale) arrays die ik kan hergebruiken in toekomstige rapportages.
Om inzichtelijk te houden wat de algoritmes doen zijn deze van commentaar voorzien om een andere programmeur te laten weten wat er gedaan is.

De applicatie is als een werkend geheel opgeleverd.

Literatuurlijst

Boeken

- 'Praktisch UML', 2de editie, Jos Warmer / Anneke Kleppe
- 'Databases en SQL', Ton de Rooij
- 'Testen volgens TMap', 2e druk, Martin Pol / Ruud Teunissen / Erik van Veenendaal

Websites

- Verschillende artikelen op wikipedia
- http://en.wikipedia.org/wiki/Unified_Modeling_Language
- http://en.wikipedia.org/wiki/Unified_Process

- Documentatie van PostgreSQL
- http://www.postgresql.org/docs/

- Documentatie van PHP
- http://www.php.net/docs.php

- Documentatie van Yii
- http://www.yiiframework.com/doc/

Verklarende woordenlijst

De meeste verklaring zijn van wikipedia, de pagina's waar de teksten vandaan komen staan erbij.

AJAX - **Asynchronous JavaScript And XML (AJAX)**
is een term voor het ontwerp van interactieve webpagina's waarin asynchroon gevraagde gegevens worden opgehaald van de webserver. Daardoor hoeven dergelijke pagina's niet in hun geheel ververst te worden.
bron:
http://nl.wikipedia.org/wiki/Asynchronous_JavaScript_and_XML

CSS - **Cascading Style Sheets (CSS),**
stijlbladen, zijn een mogelijkheid om de vormgeving van webpagina's los te koppelen van hun feitelijke inhoud en centraal vast te leggen. Het Engelse "style" heeft de betekenis van "opmaak", niet van schrijfstijl. Het begrip "cascading" (als een waterval) verwijst naar de mogelijkheid van het vererven van opmaak-eigenschappen.
De CSS-informatie voor de vormgeving van het document wordt toegevoegd aan de HTML-code ervan. Die informatie mag in het HTML-bestand zelf staan, maar ook in een apart bestand waar het html-document naar verwijst. Een dergelijk extern bestand wordt ook wel stylesheet genoemd.
bron: http://nl.wikipedia.org/wiki/Cascading_Style_Sheets

MVC - **Model-view-controller (MVC)**
is een ontwerppatroon ("design pattern") dat het ontwerp van complexe toepassingen opdeelt in drie eenheden met verschillende verantwoordelijkheden: datamodel (model), datapresentatie (view) en applicatielogica (controller). Het scheiden van deze verantwoordelijkheden bevordert de leesbaarheid en herbruikbaarheid van code. Het maakt ook dat bijvoorbeeld veranderingen in de gebruikersinterface niet direct invloed hebben op het datamodel en vice versa.
bron: http://nl.wikipedia.org/wiki/Model-view-controller-model

PHP - **PHP: Hypertext Preprocessor (PHP)**
is een scripttaal, die bedoeld is om op webservers dynamische webpagina's te creëren. PHP is in 1994 ontworpen door Rasmus Lerdorf, een senior software engineer bij IBM. Aanvankelijk stonden de letters PHP voor Personal Home Page (de volledige naam was Personal Home Page/Forms Interpreter, PHP/FI). Sinds PHP 3.0 is de betekenis een recursief acroniem geworden: PHP: Hypertext Preprocessor. Deze naam geeft aan waar de taal meestal voor gebruikt wordt: informatie verwerken tot hypertext (meestal HyperText Markup Language (HTML) en Extensible HyperText Markup Language (XHTML).
bron: http://nl.wikipedia.org/wiki/Php

SQL - **Structured Query Language(SQL)**
is een ANSI/ISO-standaardtaal voor een relationeel 'database management systeem' (DBMS). Het is een gestandaardiseerde taal die gebruikt kan worden voor taken zoals het bevragen en het aanpassen van gegevens in een relationele databank. SQL kan met vrijwel alle moderne relationele databankproducten worden gebruikt. SQL is een vierde-generatie-taal (G4-taal) omdat ze niet imperatief maar declaratief is, zoals Prolog.
bron: http://nl.wikipedia.org/wiki/SQL

UML - **Unified Modeling Language(UML)**
is een modelmatige taal die is ontworpen om objectgeoriënteerde analyses en ontwerpen voor een informatiesysteem te kunnen maken. Sinds 1997 bestaat er een standaard voor UML. Kenmerkend is dat de UML modellen een grafische weergave zijn van bepaalde aspecten van het informatiesysteem.
bron: http://nl.wikipedia.org/wiki/Unified_Modeling_Language

UP - **Unified Processing(UP)**
is een iteratief software-ontwikkelingsproces. De engelse uitleg is als volgt. “The Unified Process (UP) is a use-case-driven, architecture-centric, iterative and incremental development process framework that leverages the Object Management Group's (OMG) UML and is compliant with the OMG's SPEM. The UP is broadly applicable to different types of software systems, including small-scale and large-scale projects having various degrees of managerial and technical complexity, across different application domains and organizational cultures.”
bron: http://www.methodsandtools.com/archive/archive.php?id=32

Bijlagen zijn aanwezig in afstudeerdossier deel 2