Relationele databases vs. Document databases in de R&R-web applicatie

Bijlageboek

Xavyr Rademaker, 11077581

Bijlage	Titel
Bijlage A	Plan van Aanpak
Bijlage B	Onderzoek
Bijlage C	Pakketselectie
Bijlage D	Functioneel ontwerp
Bijlage E	Technisch ontwerp
Bijlage F	Conclusies en aanbevelingen
Bijlage G	Afstudeerplan

In dit document zijn bijlages A t/m G terug te vinden.

Bijlage A: Plan van aanpak

Document databases vs. Relationele databases in R&R-web

Xavyr Rademaker

Management samenvatting

Het bedrijf de Vries WFM heeft opdracht gegeven om te onderzoeken waar document oriented databases bij hun bestaande applicatie past, welke document oriented database en wat zijn de gevolgen van het overstappen naar een document oriented database voor hun huidige applicatie.

Het bedrijf is namelijk van plan uit te breiden naar Duitsland. Deze uitbreiding betekend dat er meer klanten zullen zijn en dus meer data in de applicatie. Het bedrijf heeft gehoord over de groeiende populariteit van document databases, en merkt dat andere bedrijven/projecten hier zijn voordelen mee hebben kunnen behalen. Zij vragen zich af of deze vorm van databases ook voordelen voor de R&R web heeft.

Het bedrijf weet echter nog niks over deze vorm van databases, zij hebben enkel gehoord dat de kosten voor het schalen lager zijn dan bij relationele databases, en dat data als ware als losse groepen wordt opgeslagen.

Het doel van de opdracht is dan ook inzicht geven in de mogelijkheden van document databases, maar ook kijken waar deze toegepast kunnen worden binnen de huidige applicatie en hoe deze kan worden toegepast.

Met deze informatie weet het bedrijf wat de voordelen, van de overstap van database, zijn voor hun huidige applicatie en wat de gevolgen ervan zijn en hoe zij zouden moeten omgaan met de nieuwe database (hoe spreek je deze aan bijvoorbeeld).

Dit document beschrijft wie deze opdracht heeft aangenomen, en hoe de opdrachtnemer en de opdrachtgever van plan zijn de opdracht succesvol af te sluiten. Ook wordt er gekeken naar welke risico's er zijn, en welke maatregelen hiertegen getroffen zullen worden.

Binnen het project zijn er verschillende onderdelen die opgeleverd dienen te worden, namelijk:

- Een verslag waarin de krachten en zwakheden van zowel document oriented databases en relationele databases naar voren zullen komen en wat de invloed hiervan is op het huidige systeem
    - Binnen de uitvoering van dit verslag wordt ook een analyse gedaan van de huidige applicatie. Door middel van deze analyse en de achterhaalde krachten en zwakheden van beide soorten databases kan bepaald worden bij welke modules van R&R-web er voordelen te behalen vallen als er overgestapt wordt naar document databases, zonder dat er functionaliteiten van deze module verloren gaan. Ook worden de gevolgen die de overstap van database op deze modules zal hebben benoemd.
- Een pakketselectie waarin naar voren komt welke document oriented database past bij de applicatie
- Het ontwikkelen van een proof of concept dat de beweringen (gemaakt in de eerste twee opgeleverde documenten) aantoont. Het ontwikkelen bestaat uit:
    - User stories zodanig uitwerken dat de opdrachtgever en opdrachtnemer beide weten wat er gebouwd gaat worden.
    - Ontwerpen proof of concept
    - Implementeren proof of concept
    - Testen proof of concept

Een voorbeeld van wat de proof of concept zal kunnen aantonen, is een verbetering in performance. Als in het eerste verslag, de krachten en zwakheden van document databases vs. relationele databases, wordt beweerd dat document databases een hogere performance hebben dan relationele databases, zal dit worden aangetoond in de proof of concept.

Het bouwen van de proof of concept is verder ook nuttig, omdat er hiermee aangetoond kan worden wat het effect van de overstap is op de huidige functionaliteiten en eventuele verantwoordelijkheden die verschuiven van de database naar de applicatie.

Voor het ontwikkelen van de proof of concept is gekozen voor een iteratieve ontwikkelmethodiek, namelijk SCRUM.

In dit document wordt verder ook duidelijk wie welke documenten ontvangt, en met welk doeleinde, wat de verwachte planning zal zijn en hoe de organisatie rondom dit project in elkaar zit.

Inhoudsopgave

1. Opdracht

1.1 Opdrachtgever en opdrachtnemer

De opdrachtnemer voor de in dit plan van aanpak beschreven delen is:

Xavyr Rademaker
Functie: afstudeerder
Email: Xavyr.Rademaker@infosupport.com

De opdrachtgever voor de in dit plan van aanpak beschreven delen is:

Info Support (de Vries WFM)
Kruisboog 42, 3905 TG Veenendaal

Contactpersoon: Rick Megens
Functie: opdrachtgever/ontwikkelaar
Email: rick.megens@devrieswfm.com

De eerste begeleider/examinatoren van de Haagse Hogeschool voor de in dit plan beschreven delen is:

De Haagse Hogeschool
Johanna Westerdijkplein 75, 2521 EN Den Haag

Contactpersonen: G. Mijnarends
Functie: Eerste begeleider/docent
Email: G.A.Mijnarends@hhs.nl

J.J. van der Hoek
Functie: Tweede beoordelaar/docent
Email: J.J.vanderHoek@hhs.nl

1.2 Opdrachtdefinitie

De Vries:

1.2.1 Aanleiding bedrijf

De Vries WFM richt zich met de applicatie R&R-web(Retail&Resultaat) op het optimaliseren van Workforce Management in de Retail sector. Workforce Management houdt volgens de opdrachtgever het volgende in: Het inplannen van mensen op de juiste plaats tegen de juiste loonkosten op het juiste moment.

Hiermee wordt bedoeld: de mensen(werknemers) met de juiste kwalificaties die beschikbaar zijn op de tijden waarop ze nodig zijn om werkzaamheden uit te voeren. Deze mensen worden ingepland op de afdelingen waar zij gekwalificeerd voor zijn. Verder wordt er bij het inplannen van deze mensen op de afdelingen rekening gehouden met het salaris van de mensen ten opzichte van de totale kosten die de werkzaamheden die zij uitvoeren mogen kosten.

De totale kosten die werkzaamheden mogen kosten worden gedaan op basis van een inschatting. Deze inschatting neemt onder andere de normtijden en normtarieven (beide gemeten waarden) en de gemiddelde kosten per werknemer in acht. Daarnaast wordt er ook rekening gehouden met wetten/regels die er gelden. Zo geeft het systeem bijvoorbeeld ook aan dat er op zondag meer uitbetaald moet worden per uur, dus is het misschien verstandig om een andere werknemer(een jongere, goedkopere wellicht) in te plannen.

Op deze manier helpt de applicatie winkelmanagers bij het zo kosteneffectief mogelijk inroosteren van hun personeel. Dit wordt aangetoond in het onderstaande voorbeeld:

Van 9.00 tot 17.00 uur zijn er bijna geen klanten voor brood. Van 17.00 tot 20.00 uur komen er veel klanten voor brood. Het totale loonkostenbudget voor de broodafdeling zijn die dag gezet op 100 euro. Van 9.00 tot 17.00 uur wordt er 1 medewerker ingeroosterd bij de brood afdeling voor 5 euro per uur (40 euro in totaal). Vervolgens wordt er van 17.00 uur tot 20.00 uur 3 medewerkers ingepland (zodat de grotere hoeveelheid klanten beter verwerkt kan worden). 1 van de 3 medewerkers werkt voor 9.50 per uur (28,50 euro in totaal) en 2 van ze werken voor 5 euro per uur (30 euro in totaal).

De totale loonkosten voor de brood afdeling die dag zijn 98,50 euro. Dit is dus binnen het gestelde loonkostenbudget. Verder is er ook rekening gehouden met de drukkere periode (1 meer ervaren persoon die duurder is, en 2 minder ervaren personen die goedkoper zijn) door meer mensen in te roosteren, en worden er geen overbodige mensen ingezet gedurende de rustige periode.

Binnen R&R-web zijn momenteel meerdere modules aanwezig om de klant te helpen hun medewerkers zo kosteneffectief mogelijk in te roosteren. De data uit deze modules wordt momenteel opgeslagen in een SQL-server database. De medewerkers van de Vries WFM vragen zich af of er een voordeel te behalen valt als er overgestapt wordt op Document Oriented Databases en zo ja, bij welke modules is dit voordeel dan te behalen.

De reden dat zij dit afvragen is dat het bedrijf van plan is te groeien, onder andere in Duitsland. Dit houdt in dat zowel het aantal eindgebruikers als de hoeveelheid data in het

systeem zullen toenemen. De medewerkers van de Vries vragen zich af of Document Oriented Databases misschien beter om kunnen gaan met deze hoeveelheden gebruikers en data (denk aan performance en kosten aan server hardware bij het opschalen van de relationele database).

De reden dat de medewerkers van de Vries WFM zich afvragen of document databases de oplossing kunnen zijn, is dat zij hebben gehoord over deze vorm van databases. Hierover hebben zij onder andere gehoord dat de kosten voor het schalen wellicht lager zijn dan bij relationele databases en dat de data als het waren als losse groepen wordt opgeslagen.

Ook is het de medewerkers van de Vries WFM opgevallen dat document databases sneller groeien en populairder zijn dan andere vormen van databases. Uit de groei in populariteit concluderen zij dat andere bedrijven/projecten dus voordelen uit het gebruik van document databases halen. Hierdoor vragen de medewerkers van de Vries WFM zich dus af of er in hun applicatie (R&R web) ook voordelen te behalen valt.

Zij hebben echter helemaal geen kennis over document databases. Zo weten zij bijvoorbeeld niet wat document databases kunnen en hoe je deze zou moeten benaderen. Het bedrijf wilt deze kennis graag opdoen (inclusief het weten of er voordelen te behalen vallen binnen hun applicatie), zodat zij kunnen overwegen of dit de database voor hun is, of dat zij verder moeten zoeken.

Een voorbeeld waarbij er misschien een voordeel valt te behalen is de roostermodule. Momenteel wordt elke dienst binnen de roostermodule opgeslagen als een record. Echter wordt het rooster altijd als geheel getoond. Hierbij zou de opslagstructuur van Document Oriented Databases, het gehele rooster als 1 document, zijn voordelen kunnen hebben. Aangezien er in dit geval maar 1 ‘record’ uitgelezen hoeft te worden, in plaats van elke dienst individueel, zal dit waarschijnlijk sneller gaan.

1.2.2 Probleemstelling bedrijf

Zoals eerder vermeld, wilt de Vries WFM groeien. Zij willen graag meer klanten binnenhalen, wat resulteert in meer data en meer gebruikers. Zij weten echter niet of de SQL-server database dit aan gaat kunnen en of het wellicht meer voordelen biedt als er overgestapt wordt op document databases. Zij willen echter ook niet zomaar zonder er iets van af te weten overstappen op een ander type database, aangezien dit consequenties kan hebben.

De problemen hier zijn dus de onwetendheid van de medewerkers van de Vries over de nieuwe vorm van databases en de hoeveelheid gebruikers/data waar het systeem in de toekomst mee te maken gaat krijgen.

1.2.3 Doelstelling bedrijf

Het doel van de opdracht is dan ook het onderzoeken waar Document Oriented Databases passen binnen de R&R-webapplicatie, welke Document Oriented Database hierbij past en wat eventuele gevolgen zullen zijn van het overstappen zodat de opdrachtgever op basis van deze gegevens kan bepalen of zij willen overstappen op document oriented databases, of dat zij naar een andere oplossing moeten gaan zoeken. Verder wil de opdrachtgever zo veel mogelijk conclusies terugzien in een proof of concept, dat gebaseerd is op functionaliteiten uit een of meerdere modules uit R&R web.

Mocht er een conclusie zijn die niet kan worden aangetoond, wordt er verwezen naar waar de conclusie op gebaseerd is (bijvoorbeeld naar de documentatie van de nieuwe database).

De bovenstaande documenten en de proof of concept hebben als doel het overbrengen van kennis naar de opdrachtgever.

1.2.4 Resultaat bedrijf

Aan het eind van dit project heeft de opdrachtgever een document waar relationele databases vergeleken worden met document oriented databases. Verder hebben zij een document waarin staat welke modules compatible zijn met deze vorm van databases en een advies over welke implementatie van deze vorm van databases het best bij hun applicatie past. Ook beschikken zij op dat moment over een proof of concept die zo veel mogelijk eerder gemaakte beweringen aantoont.

1.2.5 Effect bedrijf

Aan de hand van de resultaten en de proof of concept kan de Vries WFM beslissen of zij wel of niet overstappen op de geselecteerde Document Oriented Database. Mocht de Vries WFM ervoor kiezen over te stappen, dan kunnen zij de overstap beginnen te implementeren. Mocht de Vries WFM ervoor kiezen om niet over te stappen aan de hand van de getrokken conclusies, weten zij dat zij verder moeten zoeken naar een andere oplossing.

De opdrachtnemer:

1.2.6 Aanleiding student

De opdrachtnemer is een vierdejaars student Informatica aan de Haagse Hogeschool. Voor zijn afstuderen wordt van hem verwacht dat hij 18 weken stageloopt, en daar een afstudeerverslag over schrijft. Het eindcijfer wordt bepaald op basis van het ingeleverde verslag, met alle mijlpaalproducten in de bijlage. Gedurende de stageperiode zal hij de bovenstaande opdracht voor de Vries WFM uitvoeren.

1.2.7 Doelstelling student

Het doel van de opdrachtnemer is om voor het verslag en de verdediging samen een 8 of hoger te halen. Mocht dit doel behaald worden is de opdrachtnemer afgestudeerd met lof.

1.2.8 Resultaat student

Aan het eind van de afstudeerperiode heeft de opdrachtnemer een afstudeerverslag geschreven inclusief een externe bijlage waarin alle mijlpaalproducten verwerkt zijn ingeleverd en is hij zijn verdediging doorstaan.

1.2.9 Effect student

Als het resultaat behaald is, is de opdrachtnemer afgestudeerd.

1.3 Afbakening

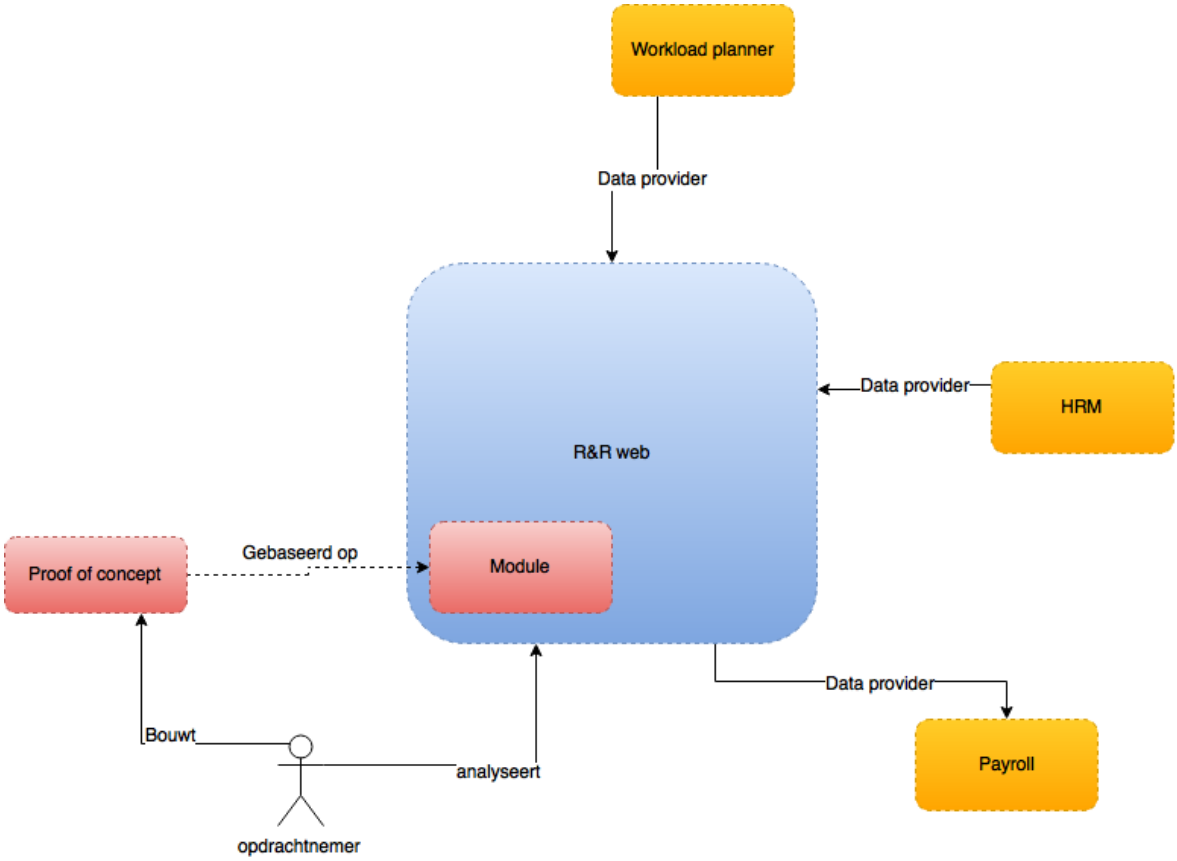
Het project beperkt zich enkel tot de R&R-webapplicatie. Hierbij zullen alle modules geanalyseerd worden, en zal er een proof of concept gebouwd worden op basis van een of enkele modules. Ook zal er alleen gekeken worden naar Document Oriented Databases, niet naar andere vormen van databases met SQL-server als uitzondering. De volgende acties worden wel gedaan binnen het project:

1. Een onderzoek wat antwoord op de vraag: Bij welke modules van R&R-web zijn er voordelen te behalen bij het overstappen naar document oriented databases zonder dat dit ten koste gaat van de huidige functionaliteit, en welke gevolgen heeft deze overstap?
2. Het analyseren van de huidige applicatie
3. Een pakketselectie waaruit blijkt welke database bij het bedrijf past.
4. Een proof of concept bouwen gebaseerd op functionaliteiten uit een module uit de huidige applicatie
    a. Ontwerpen van de proof of concept
    b. Ontwerpen en implementeren van de database van de proof of concept
    c. Bouwen van de proof of concept in C#
    d. Testen van de proof of concept
    e. Achterhalen/specificeren user stories van de proof of concept
5. Het afstudeerverslag
6. Voorbereiden en geven van de presentatie incl. verdediging

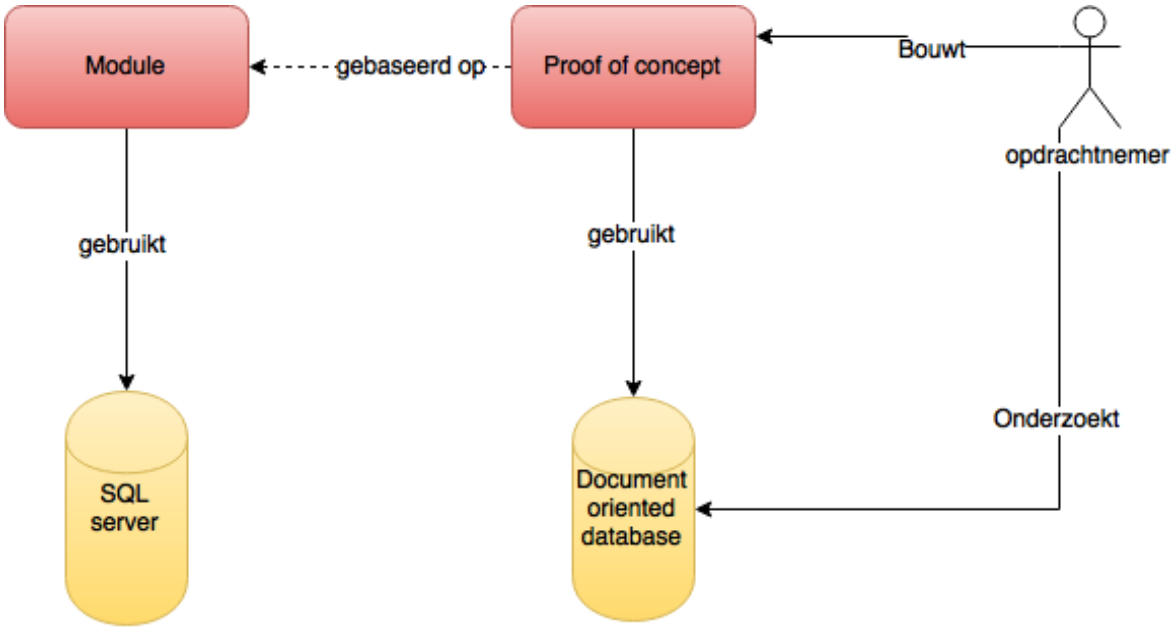
Buiten het project valt het volgende:

1. Overige applicaties van de Vries WFM en Info Support
2. Vormen van databases anders dan Document Oriented Databases, met als uitzondering SQL-server
3. Een volledige applicatie (er wordt enkel een proof of concept gebouwd)
4. Nieuwe, nog niet geïmplementeerde, modules van R&R web
5. Ophalen van nieuwe requirements voor de R&R-webapplicatie
6. Beheer/werken aan de R&R-applicatie
7. De front-end, het hoeft niet op de huidige applicatie te lijken als de functionaliteit er maar in zit
8. R&R time, R&R mobile apps en R&R integration. Het project focust alleen op R&R-web daar dit de core van het systeem is.

De volgende afbeeldingen laten zien waar dit project zit binnen de huidige applicatie:



De proof of concept staat los van de huidige applicatie, maar zal wel gebaseerd zijn op functionaliteiten uit de huidige applicatie. Ingaande op de proof of concept ziet het er als volgt uit:



Zoals in de afbeelding de zien bouwt de opdrachtnemer de proof of concept, en onderzoekt hij welke document oriented database hier het best bij past, en hoe deze het best

geïmplementeerd kan worden bij de proof of concept. De implementatie van de database hoort bij het bouwen van de proof of concept.

1.4 Afhankelijkheden

Om het project succesvol af te ronden zijn er een aantal afhankelijkheden, namelijk:

1. De bereikbaarheid en deelname van de opdrachtgever
2. De kennis en pro activiteit van de opdrachtnemer
3. De begeleiding van de technische begeleider
4. Beschikbare documentatie omtrent de huidige R&R-webapplicatie of toegang tot de applicatie inclusief de database (mag een testdatabase zijn als de structuur maar klopt) en/of een persoon met kennis over de volgende punten:
    a. De database van de applicatie
    b. De functionele en niet functionele eisen aan het systeem en de database
    c. Welke modules gebruiken welke data uit de database
5. De beschikbare documentatie omtrent document oriented databases
6. De begeleiders en examinatoren van de opdrachtnemer
7. Beschikbare kennis (literatuur, personen, blogs, lezingen enzovoorts) omtrent document oriented databases
8. De aanwezigheid en werking van een ontwikkelomgeving
9. Eventuele tools die later nodig blijken te zijn

Toelichting:
Hieronder de toelichting per punt:

1. De opdrachtgever moet bereikbaar zijn, omdat de eisen en wensen vanuit hem moeten komen. Ook is het zo, dat de opdrachtgever moet valideren of er gebouwd wordt wat hij wil hebben. De opdrachtgever moet ook bereikbaar zijn om feedback te geven en om samen met de opdrachtnemer keuzes te maken.
2. De opdrachtnemer dient de benodigde kennis te hebben, of op te doen als dit niet het geval is, om het project succesvol af te kunnen ronden. Verder moet hij een proactieve houding aannemen, zowel de opdrachtgever als de technisch begeleider op de hoogte houden van ontwikkelingen en de opdrachtgever/technisch begeleider aansporen als hier reden toe is.
3. De technische begeleider dient de opdrachtnemer te begeleiden tot het succesvol afronden van het project, hierbij dienen vragen beantwoord te worden en eventueel feedback gegeven als iets beter kan
4. De documentatie, applicatie incl. database en/of een persoon met de juiste kennis omtrent de huidige applicatie is nodig, omdat de opdrachtnemer hiermee zicht krijgt op wat de applicatie, en de modules precies doen.
5. Deze documentatie is nodig om de pakketselectie zo goed mogelijk uit te kunnen voeren. De meer relevante documentatie er beschikbaar is, de meer criteria achterhaald kan worden. Onder relevante documentatie wordt het volgende verstaan:
    a. Beschikbare functionaliteiten binnen de database
        i. Kan zijn stored procedures, triggers, indexen, replica enz.
    b. Manier van querieen binnen de database
    c. Platformen waarop de database kan draaien
    d. Programmeertalen waar drivers/ODM’s (object document mapper) voor beschikbaar zijn. Incl. hoe deze werken.

    e. Installatie handleidingen van de database
6. De begeleiders en examinatoren zijn van belang, omdat zij begeleiden en beoordelen vanuit de school.
7. De kennis omtrent document oriented databases is nodig om het onderzoek: Bij welke modules van R&R-web zijn er voordelen te behalen bij het overstappen naar document oriented databases zonder dat dit ten koste gaat van de huidige functionaliteit, en welke gevolgen heeft deze overstap? Te kunnen uitvoeren
8. Er is een ontwikkelomgeving nodig om de proof of concept te kunnen bouwen
9. -

1.5 Kwaliteitseisen

De volgende kwaliteitseisen worden gesteld aan het project:

Functioneel:

F1. De software die geschreven wordt is gedocumenteerd, inclusief commentaar waar nodig
F2. De code is getest met een code coverage van minimaal 80%
F3. De codeerstandaarden van Microsoft dienen gehanteerd te worden(zie https://msdn.microsoft.com/en-us/library/ff926074.aspx)
F4. Alles wat in de oude applicatie kan, moet nog steeds werken met de nieuwe database.

Reliability:

R1. De data die wordt ingevoerd door de gebruiker moet correct in de database opgeslagen worden

Security:

S1. Klanten mogen geen data van elkaar zien.

De opdrachtgever heeft bepaald dat de codeerstandaarden van Microsoft gehanteerd dienen te worden, en die van Info Support niet per se hoeven van hem. Reden dat dit niet hoeft, is dat deze standaarden niet strikt gehanteerd worden binnen de Vries WFM. De standaarden van Microsoft worden wel gehanteerd.

Wel gaf de opdrachtgever aan dat het wel kan helpen om de striktere regels van Info Support te hanteren, aangezien je daarmee altijd veilig zit.

Naast de bovenstaande kwaliteitseisen zijn er ook nog een aantal eisen vanuit de school. Namelijk:

Selecteren van standaardsoftware (De pakketselectie)
Het betreft het selecteren van een of meerdere applicatie(componenten) voor een simpel bedrijfsproces. Er is een ‘longlist’ beschikbaar of eenvoudig samen te stellen, maar de selectiecriteria zijn dubbelzinnig en spreken elkaar soms tegen.

Ontwerpen, bouwen en bevragen van een database
Het betreft het ontwerpen, bouwen en bevragen van een database die gebruikt gaat worden door vele verschillende groepen gebruikers. De beschikbaarheid van data moet gewaarborgd blijven en de performance is een kritische factor. Daarnaast wordt de kwaliteit van de data bewaakt d.m.v. in het DBMS beschikbare functionaliteiten.

Bouwen applicatie

Het betreft het bouwen van een objectgeoriënteerde applicatie, waarbij geavanceerde concepten van de gebruikte programmeertaal aan de orde komen. Verder wordt rekening gehouden met toekomstige wijzigingen, testbaarheid en hergebruik. Het bouwen gebeurt in een ontwikkelomgeving.

Uitvoeren van en rapporteren over het testproces
Bij het opstellen van het logisch testontwerp wordt gebruik gemaakt van een testontwerptechniek. Er is aandacht voor herhaalbaarheid van de testen. Het betreft hoofdzakelijk het testen van de functionaliteit. Testrapportage betreft het volledige systeem.

Afstudeerverslag
In het afstudeerverslag moet het volgende terugkomen:

- Een referaat, voorwoord, inhoudsopgave en inleiding
- Beschrijving van de organisatie van de opdrachtgever en de plaats van de afstudeerder daarin
- Beschrijving van mogelijke oplossingsmethoden en verdediging gekozen aanpak
- Beschrijving van de werkzaamheden met toelichting op en motivatie van gemaakte keuzes
- Bespreking van afwijkingen ten opzichte van het afstudeerplan met motivatie
- Analyse van de tussen resultaten
- Bespreking van de opgeleverde tussenproducten los van de tussenproducten zelf
- Evaluatie van de gebruikte aanpak tijdens de afstudeerperiode
- Evaluatie van de opgeleverde tussenproducten
- Evaluatie op de uitgevoerde beroepstaken
- Geraadpleegde literatuur
- Afkortingenlijst

1.6 Uitgangspunten

Voor uitvoering van de in dit plan van aanpak beschreven delen zijn de onderstaande uitgangspunten van toepassing:

1. Er wordt gewerkt volgens SCRUM, aangezien het team uit 1 persoon bestaat zal de uitvoering afwijken van de normale scrum uitvoeren. Meer hierover in hoofdstuk 3.1
2. Er wordt gewerkt op de hoofdlocatie van Info Support in Veenendaal
3. Er wordt minimaal eens per 2 weken afgesproken met de opdrachtgever.
4. Sprints duren 2 weken
5. De te ontwikkelen software is een op zichzelf staande applicatie
6. De nieuwe database moet draaien op Windows server 2012 R2, tenzij er hele goede redenen zijn om over te stappen
7. De proof of concept is gebouwd in C#/.NET
8. De applicatie maakt gebruik van REST-Services

1.7 Randvoorwaarden

Aan uitvoering van de in dit plan van aanpak beschreven delen zijn de volgende randvoorwaarden gesteld:

1. Om te kunnen werken dient de opdrachtnemer een werkplek te hebben in Veenendaal, welke voorzien is van internet
2. De opdrachtgever geeft binnen 1 week feedback op documenten waar hij feedback op moet geven
3. De opdrachtgever reageert binnen 3 dagen op mailtjes waarop gereageerd moet worden
4. De opdrachtgever vindt het niet erg als de opdrachtnemer afspraken op school of uren voor school moet maken

2. Risicomanagement

Onderkende risico's met maatregelen ter beheersing
Voor uitvoering van de in dit plan van aanpak beschreven delen zijn de onderstaande risico's onderkend. Bij de risico's zijn de oorzaken en bijbehorende maatregelen ter beheersing opgenomen.

Nr.	Risico omschrijving				
	Oorzaak	Maatregel	P/S	Wie	Wanneer
1	Het project loopt vertraging op/wordt niet binnen de tijd afgerond				
	Opdrachtgever of opdrachtnemer zijn langere tijd afwezig	Via skype en mail contact blijven houden en eventuele afspraken verplaatsen	S	Opdrachtgever en opdrachtnemer	Op de afgesproken momenten en eventueel vaker als het vaker nodig is.
	Documenten gaan verloren	Alle tussenversies van alle documenten mailen en aan het eind van de dag alle documenten op laptop zetten	S	Opdrachtnemer	Aan het eind van elke tussenversie
	Source code gaat tussentijds verloren	Source code op een git-repository zetten	S	Opdrachtnemer	Minimaal 1x per dag(als er geprogrammeerd is)
	De opdrachtnemer beschikt niet over voldoende kennis	Googelen naar informatie, vragen hulp aan de technisch begeleider, volgen cursussen van Info Support	P/S	Opdrachtnemer	Als de opdrachtnemer merkt dat hij ergens tegenaan loopt
	De opdrachtnemer weet niet wat hij(op een bepaald punt binnen het project) moet doen	De opdrachtnemer moet aan de technisch begeleider, of aan de opdrachtgever vragen wat de vervolg stappen zijn, afhankelijk van de situatie	S	De opdrachtnemer	Zodra hij merkt dat hij niet weet wat nu
	De opdrachtgever is niet beschikbaar voor meetings, waardoor niet duidelijk is wat hij van de opdrachtnemer verwacht	De opdrachtgever wordt hierop aangesproken en de begeleiders worden ingeschakeld	S	Begeleiding van de opdrachtnemer, opdrachtgever, opdrachtnemer	Op het moment dat bekend wordt dat de opdrachtgever niet beschikbaar is.

	De opdrachtnemer werkt buiten de scope	De opdrachtnemer controleert voorafgaand aan het werken aan een onderdeel of dit binnen de scope valt met behulp van de eerder opgestelde afbakening. Op het moment dat een wens/eis van de opdrachtgever buiten de scope ligt spreekt de opdrachtnemer de opdrachtgever hierop aan en wordt er gezamenlijk gekeken wat ermee gedaan gaat worden	P	Opdrachtnemer	Voorafgaand aan het werken aan een onderdeel, zodra de opdrachtnemer merkt dat hij buiten de scope gaat werken
	De opdrachtnemer werkt buiten de scope	De opdrachtgever spreekt de opdrachtnemer hierop aan	P/S	Opdrachtgever	Zodra de opdrachtgever merkt dat er buiten de scope gegaan wordt.
	De opdrachtnemer werkt onbewust buiten de scope van het project	De technisch begeleider houdt bij wat de opdrachtnemer aan het doen is en spreekt de opdrachtnemer aan als deze buiten de scope dreigt te gaan werken.	S	Technisch begeleider	Op het moment dat ze opmerkt dat er buiten de scope gewerkt word
	De opdrachtnemer wacht te lang met hulp vragen	Hulp vragen aan de technisch begeleider	P	De opdrachtnemer	Zodra de opdrachtnemer ergens tegenaan loopt.
	De opdrachtnemer kan niet bij de virtualisatie komen omdat het domein eruit ligt	Domein werkend krijgen, ondertussen verder werken aan documentatie	S	Systeembeheer, opdrachtnemer	Zo snel mogelijk nadat geconstateerd word dat er iets mis is met het domein, zolang het domein niet hersteld is
	De opdrachtnemer kan niet bij de virtualisatie komen omdat de opdrachtnemer geen	Verstrekken van de juiste bevoegdheden aan de opdrachtnemer	P	Systeembeheer, opdrachtgever	Voorafgaand aan het project. Mochten de bevoegdheden

	bevoegdheid om bij de virtualisatie te komen heeft				halverwege wijzigen, dan zodra het opvalt.
	De virtualisatie ligt eruit	De virtualisatie werkend krijgen, ondertussen verder werken aan documentatie	S	Systeembeheer, opdrachtnemer	Zo snel mogelijk nadat geconstateerd word dat er iets mis is met de virtualisatie, zolang het domein niet hersteld is
2	De opgeleverde producten voldoen niet aan de eisen van school				
	De mijlpaalproducten zijn niet van voldoende kwaliteit	Zowel de technisch begeleider als de opdrachtgever geven feedback op de mijlpaalproducten(de opdrachtgever: is het wat ik wil, de technisch begeleider: is het kwalitatief goed genoeg)	P	Technisch begeleider, opdrachtgever	Bij het opleveren van afgeronde mijlpaalproducten en het reviewen van mijlpaalproducten waaraan gewerkt wordt.
	De mijlpaalproducten zijn niet van voldoende kwaliteit	De eerste begeleider van school voert tussentijdse reviews van het afstudeerverslag incl. bijlage uit.	S	Eerste begeleider van school	Op 25%, 45% en 60% van het project(gebaseerd op tijd, niet voortgang) en 3 weken voor het inleveren van het afstudeerverslag
	Het afstudeerverslag is niet goed opgebouwd	De technisch begeleider reviewed het verslag en geeft feedback aan de opdrachtnemer.	P	Technisch begeleider	Na elk mijlpaalproduct/elke iteratie
	De opdrachtnemer heeft niet kunnen opleveren naar de verwachtingen van de opdrachtgever	De opdrachtnemer wekelijks met de opdrachtgever of de technisch begeleider om de voortgang te bespreken.	P	Opdrachtnemer, Opdrachtgever, Technisch begeleider	Elke week met afwisselend de opdrachtgever en de technisch begeleider
3	De proof of concept komt functioneel niet overeen met functionaliteiten van de huidige applicatie				

	De huidige functionaliteiten zijn onduidelijk bij de opdrachtnemer	1.Bij onduidelijkheden dient de opdrachtnemer duidelijkheid te halen bij de opdrachtgever (vragen stellen), 2.De opdrachtgever stelt voorafgaand documentatie over de desbetreffende module beschikbaar, 3.De opdrachtgever en opdrachtnemer stellen voor elke iteratie af wat er gebouwd gaat worden en hoe dit precies moet werken	P, P, P	De opdrachtnemer, De opdrachtgever, De opdrachtnemer en de opdrachtgever	1.Tijdens het ontwikkelen zodra er onduidelijkheden voorkomen, 2.Voorafgaand aan het ontwikkelen 3.Voorafgaand aan het ontwikkelen
	Verschillende interpretatie van backlogs door opdrachtgever en opdrachtnemer	De backlogs vooraf samen met de opdrachtgever specificeren en achteraf valideren of het juiste is gebouwd met de opdrachtgever	P	Opdrachtnemer	Aan het begin en aan het eind van elke sprint
4	Er ontstaat een conflict tussen de opdrachtgever en de opdrachtnemer waardoor het project niet verder kan				
	De opdrachtgever en opdrachtnemer hebben een verschillend beeld bij wat er moet gebeuren	Als de eisen/wensen van de opdrachtgever niet onmogelijk zijn, heeft de opdrachtgever het laatste woord	P	Opdrachtgever en opdrachtnemer	Als er een conflict ontstaat
5	Er ontstaat een conflict tussen de belangen van school en de belangen van het bedrijf				
	Er zijn verschillende belangen bij het project	De eerste begeleider, de opdrachtgever/begeleiders vanuit het bedrijf en de opdrachtnemer gaan samen proberen een middenweg te vinden	S	De eerste begeleider, de opdrachtgever/begeleiders vanuit het bedrijf en de opdrachtnemer	Zodra het conflict optreedt
6					

P/S = Preventief of Schadebeperkend

3. Aanpak

In dit hoofdstuk zal de aanpak van het project beschreven worden. Hierbij wordt een ontwikkelmethodiek gekozen, met een toelichting waarom er voor deze methodiek gekozen is. Verder wordt er gekeken naar welke fases het project zal hebben. Per fase zal er een korte aanpak beschreven worden, zullen de doelen benoemd worden en welke eindproducten er worden opgeleverd.

3.1 Ontwikkelmethodiek

De ontwikkelmethodiek die in dit project gehanteerd gaat worden is een iteratieve methode. De reden dat het een iteratieve methode is, is dat vooraf niet vaststaat wat het exacte eindproduct zal zijn. Wel staan de tijd en de budget/middelen vast. Bij een iteratieve methode is er de mogelijkheid om gestelde eisen en wensen te wijzigen gedurende het project. Deze mogelijkheid kan resulteren in een eindproduct wat afwijkt van oorspronkelijke ideeën.

De reden dat het mogelijk is om gestelde eisen en wensen gedurende het project te wijzigen, is dat de iteraties kort zijn en herhaald worden. Zo worden er in elke iteratie requirements/user stories opgehaald, ontworpen, gebouwd en getest. In elke iteratie zal dus een klein deel van het volledige systeem geïmplementeerd worden. Omdat het kleine delen betreft, is het wijzigen van de eisen hieraan makkelijker te implementeren dan als het zou gaan om wijzigingen in een volledig gebouwd systeem.

Als methodiek is gekozen voor SCRUM. Reden dat er voor SCRUM is gekozen is, dat het een iteratieve ontwikkelmethodiek is. Verder is het zo dat er in SCRUM alleen gedocumenteerd wordt wat nodig is, en niks overbodigs wat ervoor zorgt dat er geen tijd verspeeld wordt.

Ook zijn er veel opleveringen (shipable products) en omdat het zo kort cyclisch is, is er ook veel feedback. Dit is relevant voor dit project, omdat de opdrachtnemer naast het uitvoeren van het project ook moet leren. Dit gebeurd onder andere door het verkrijgen van feedback op alles wat hij heeft opgeleverd. Aangezien er feedback gegeven wordt op wat er in de korte cyclus is uitgevoerd, is de feedback ook snel te verwerken zonder dat dit een al te grote impact heeft op het verloop van het project.

Naast dat de opdrachtnemer moet leren, is het ook vanuit de opdrachtgever gezien voordelig dat hij de mogelijkheid heeft om kort cyclisch feedback te geven. Hierdoor heeft hij namelijk meer controle op wat het eindproduct uiteindelijk wordt. Ook heeft hij de mogelijkheid snel in te grijpen als er iets anders geïmplementeerd is dan hij verwacht.

Het is ook zo, dat het SCRUM-team zelf zijn acties organiseert, en dat niet gedaan wordt door een manager van buitenaf. Aangezien er binnen dit project sprake is van 1 persoon, de opdrachtnemer, die naast het project voor de opdrachtgever ook verantwoordelijkheden naar zijn school heeft, is het passend dat hij zelf zijn stappen/acties kan organiseren (in overleg met de opdrachtgever en de technisch begeleider).

Tot slot gebruikt het bedrijf zelf ook SCRUM, waardoor de opdrachtgever en de technische begeleider het al gewend zijn om met SCRUM te werken. Dat de het grootste deel van de

betrokkenen (2 van de 3) het gewend zijn om met SCRUM te werken heeft als gevolg dat het niet nodig is om lessen/cursussen te geven in SCRUM. Zij weten namelijk al hoe het werkt, en hoe zij hier het optimale uit kunnen halen. Dit resulteert ook in een betere begeleiding voor de opdrachtnemer.

Het is echter zo dat het ontwikkelteam bestaat uit maar 1 persoon, dus is er gekozen voor een aangepaste vorm van SCRUM. Gedurende het project zullen de volgende onderdelen wel/niet/anders gedaan worden:

Wel:
Er worden sprints aangehouden die zullen leiden tot shipable products. Voorafgaand aan het bouwen wordt de product backlog gevuld met alle op dat moment bekende user stories, voor de te bouwen proof of concept. Aan het begin van elke sprint wordt er samen met de opdrachtgever bepaald welke backlog items in deze sprint uitgewerkt zullen worden (de sprintplanning). De backlog items die hierin voorkomen zullen ook gespecificeerd worden indien er onduidelijkheden zijn.

Er wordt ook een sprint review gedaan. De opdrachtgever zal worden gevraagd hier deel aan te nemen om eventueel feedback te geven. Hierbij wordt ook meteen de retrospective gedaan. De technische begeleider en de procesbegeleider mogen de sprint review en retrospective bijwonen mochten zij op dat moment tijd hebben.

Niet:
Er is geen sprake van een SCRUM-team, de backlogs worden gezamenlijk opgesteld door de opdrachtnemer en de opdrachtgever. En vervolgens wordt alles ontworpen, ontwikkeld en getest door de opdrachtnemer.

Aangepast:
Er zullen geen daily stand ups gehouden worden, omdat dit niet relevant is met 1 deelnemer. Wel wordt er aan het eind van elke dag gekeken naar: wat heb ik vandaag gedaan, hoe ging dit en wat ga ik morgen doen.

Naast SCRUM zal er ontwikkeld worden volgens Test Driven Development(TDD). Dit houdt in dat de tests eerst opgesteld zullen worden op basis van de user stories. Vervolgens worden de user stories gebouwd, tot de test slaagt. Er zijn meerdere reden waarom voor test driven development gekozen is, namelijk:

- De kwaliteit van geschreven code wordt altijd gehanteerd, alles wat gebouwd is, is in principe getest
- Het is een manier van ontwikkelen die niet standaard geleerd wordt op school, het toepassen in de praktijk is een leerzame ervaring.

TDD zal gebruikt worden in combinatie met SpecFlow. SpecFlow is een tool voor .NET waarbij user stories, en bijbehorende voorbeelden, worden omgezet tot tests. De scenario's bij de user stories moeten hiervoor wel in Gherkin Language opgesteld worden. Gherkin Language houdt het volgende in:

- Er wordt aangegeven wat de feature is (user story)
- Er wordt een beschrijving gegeven van wat het scenario moet doen
- De precondities worden aangegeven met: Given … (als er meerdere precondities zijn worden deze gescheiden door And …)
- De actie van de actor wordt aangegeven met: When … (ook hier kan er opgesomd worden met And …)

- De verwachte uitkomst wordt aangegeven met: Then … (ook hier kunnen er meerdere uitkomsten zijn).

Reden dat er gekozen is om SpecFlow te gebruiken is dat ook dit een manier van leren is en omdat het bedrijf zijn tests ook op deze manier uitvoert. Er is overwogen om de applicatie te testen met behulp van TMap Next, aangezien dit al bekend is bij de opdrachtnemer en geen leercurve zal hebben. Hierbij zijn de volgende stappen overwogen:

1. Het gebruik van SpecFlow kan betekenen dat het testen langer gaat duren, omdat de manier van nog aangeleerd moet worden. Dit is niet erg, en zal alleen invloed hebben op het begin van de bouwfase.
2. Het is voor de opdrachtgever iets nieuws om te leren, dus een nieuwe uitdaging
3. Als de opdrachtnemer beide manieren evengoed zou beheersen, zou hij ook voor SpecFlow kiezen omdat:
    a. Het de link tussen requirements en testen als het ware afdwingt. Alle user stories krijgen een aantal scenario's waarin de functionaliteiten getest worden
    b. De tests zijn goed te begrijpen door iedereen, zelfs al is er geen technische kennis aanwezig (wat er getest word is duidelijk)

Het gevolg van het gebruik van SpecFlow is dat de user stories ook voorbeelden/scenario's moeten bevatten. Dit wordt ook wel specification by example genoemd.

3.2 Fasering

Binnen het project zullen er verschillende fases zijn, namelijk de:

1. Opstartfase
    a. Onderzoek: Bij welke modules van R&R-web zijn er voordelen te behalen bij het overstappen naar document oriented databases zonder dat dit ten koste gaat van de huidige functionaliteit, en welke gevolgen heeft deze overstap?
    b. Pakketselectie
2. Bouwfase
    a. Requirements/backlogs
    b. Ontwerpen
    c. Bouwen
    d. Testen
3. Afstudeerverslag
4. Afstudeerverdediging

Per fase zal het volgende worden uitgevoerd:

3.2.1 Opstartfase: Onderzoek: bij welke modules van R&R-web zijn er voordelen te behalen bij het overstappen naar document oriented databases zonder dat dit ten koste gaat van de huidige functionaliteit, en welke gevolgen heeft deze overstap?

In de eerste fase van het project zal er een onderzoek worden uitgevoerd. Het doel van dit onderzoek is antwoord geven op de volgende vraag: `Bij welke modules van R&R-web zijn er voordelen te behalen bij het overstappen naar document oriented databases zonder dat dit ten koste gaat van de huidige functionaliteit, en welke gevolgen heeft deze overstap?'

Om dit te kunnen achterhalen zullen er een aantal dingen moeten gebeuren, namelijk:

1. Uitzoeken wat relationele databases zijn en wat deze kunnen
2. Uitzoeken wat document databases zijn en wat deze kunnen
3. De huidige applicatie analyseren en daar het volgende uit halen:
    a. Welke modules zijn er en waar zijn deze verantwoordelijk voor
    b. Welke functionaliteiten er in de modules van de applicatie zitten
    c. Welke niet functionele eisen er zijn aan de modules en de gehele applicatie
    d. Welke data gebruiken de modules, en hoe is de structuur van deze data
    e. Welke functionaliteiten (stored procedures, triggers enz.) zitten er in de huidige database
4. Uitzoeken wat de krachten en zwakheden van relationele databases zijn
    a. Welke hiervan hebben betrekking op modules van de applicatie
5. Uitzoeken wat de krachten en zwakheden van document databases zijn
    a. Welke hiervan hebben betrekking op modules van de applicatie

Om de eerste 2 punten te achterhalen zullen er bronnen gezocht worden. Deze bronnen worden gefilterd op relevantie, en moeten aan minimaal 1 van de volgende eisen voldoen:

- Onderzoeksrapporten
- Universitaire verslagen
- Veel gebruikte bronnen (dus een bron die in meerdere andere documenten genoemd word)
- Personen/bedrijven met ervaring in het vak en geen belang hebben bij het promoten/afkraken van het onderwerp

Voor punt 3 moet de gehele huidige applicatie geanalyseerd worden. Om dit te doen moeten er verschillende stappen ondernomen worden. Namelijk:

3a) Op de site van de Vries WFM staan de modules inclusief uitleg. Mocht deze uitleg te weinig informatie bevatten, of mochten er onduidelijkheden ontstaan, wordt de opdrachtgever geraadpleegd voor toelichting.

3b) De opdrachtgever stelt een lijst met user stories/requirements aan het systeem beschikbaar. Deze worden bestudeerd en verdeeld over de modules. De verdeling wordt gevalideerd door de opdrachtgever.

3c) De opdrachtgever wordt gevraagd of hier ook een lijst van beschikbaar is. Mochten de niet functionele eisen niet gedocumenteerd zijn, wordt de opdrachtgever gevraagd om deze op te noemen, voor zover hij dit weet.

3d) De opdrachtgever stelt een testdatabase beschikbaar. Aan de hand van deze database kan een implementatiemodel gemaakt worden en deze kan weer worden omgezet naar een diagram. Dit diagram zal verdeeld worden over de modules (dus welke tabellen horen waarbij). De verdeling wordt gevalideerd door de opdrachtgever.

3e) In de testdatabase zijn de gemaakte functionaliteiten op databaseniveau terug te vinden. Mocht dit niet het geval zijn, wordt de opdrachtgever hierom gevraagd.

Voor punt 4 en 5 wordt er weer literatuur geraadpleegd, net zoals bij de eerste 2 stappen. Hier worden de krachten en zwakheden van beide type databases uitgehaald, en er wordt gekeken welke krachten en welke zwakheden betrekking hebben op de huidige applicatie.

Om antwoord op de hoofdvraag te kunnen geven, word elk eerdergenoemde stap een deelvraag. De resultaten van de deelvragen, inclusief de bronnen/uitvoering van de deelvragen worden verwerkt in een document. Elke deelvraag zal dan ook een hoofdstuk zijn, met de volgende structuur:

- 1. Deelvraag
    - Resultaat
    - Uitvoering

In het geval van literatuur raadpleging, is de bronverwijzing de uitvoering.

Het eindproduct van deze fase is dan ook dit document. Deze zal bij zowel de opdrachtgever als de technische begeleider ingeleverd worden.

3.2.2 Opstartfase: Pakketselectie

Als de huidige applicatie bekend is, zal er gekeken worden welke document oriented database het beste bij deze applicatie past. Dit wordt gedaan door middel van een pakketselectie. Met pakketselectie wordt bedoeld: Een onderzoek dat achterhaald welke bestaande software het beste bij de eisen en wensen van de opdrachtgever past.

Gedurende deze fase is de deelname van de opdrachtgever cruciaal, aangezien alle criteria aan de database van hem moeten komen, en er zonder criteria geen database geselecteerd kan worden.

Voorafgaand aan de eerste stap is het onderzoek: Bij welke modules van R&R-web zijn er voordelen te behalen bij het overstappen naar document oriented databases zonder dat dit ten koste gaat van de huidige functionaliteit, en welke gevolgen heeft deze overstap? al uitgevoerd. De resultaten hiervan worden meegenomen als input voor het eerste gesprek (en mogelijk ook voor de latere gesprekken). Verder wordt er voorafgaand aan dit gesprek een lijst met databases opgesteld op basis van randvoorwaarde 6: De nieuwe database moet draaien op Windows server, tenzij er hele goede redenen zijn om over te stappen.

Mocht deze lijst 10 of minder databases bevatten worden stap 1 en 2 overgeslagen en wordt deze lijst als longlist beschouwt. Verder zal er voorafgaand aan elk gesprek een plan voor dat desbetreffende gesprek opgesteld door de opdrachtnemer. Dit plan omvat het volgende:

- Welke vragen komen voort uit de vorige stap
- Welke informatie verwacht de opdrachtnemer uit het gesprek te halen
- Hoe verwacht de opdrachtnemer aan deze informatie te komen

De pakketselectie zal als volgend aangepakt worden:

1. De opdrachtnemer en de opdrachtgever gaan samen zitten voor een gesprek. In dit gesprek wordt met de opdrachtgever een minimale lijst met criteria bepaald.
2. Aan de hand van de in stap 1 opgenomen criteria zal er een longlist opgesteld worden. Verder zullen zowel de genomen stappen, als de toelichting waarom de longlist er zo uit ziet aanwezig zijn. De toelichting geld per database op de longlist
3. De opdrachtnemer en de opdrachtgever gaan weer samen zitten voor een gesprek. Uit dit gesprek dienen weer criteria te komen. Verder dient de criteria ook gewogen te worden.
    a. Voorbeeld: het is belangrijker dat er stored procedures gemaakt kunnen worden dan dat er triggers geschreven kunnen worden.
4. Aan de hand van de criteria uit stap 2 zal er van de longlist tot een shortlist gewerkt worden. De shortlist mag maximaal 3 databases bevatten. Op het

moment dat er meer databases overblijven betekend het dat stap 2 nog een keer uitgevoerd moet worden, met nieuwe criteria voor de overgebleven databases. Ook bij het maken van shortlist zullen de genomen stappen en de toelichting vermeld worden.

5. Als de shortlist bekend is, zal er voor de laatste keer een gesprek plaatsvinden waar criteria opgehaald wordt. Deze keer zal de criteria iets anders opgehaald worden. Er wordt bij deze set criteria namelijk een weging gegeven aan alle criteria. Verder wordt er per criteria ook bepaald hoe er punten aan de database gegeven kunnen worden en wat de punten inhouden.
6. Op basis van de criteria, de weging en de punten wordt per database op de lijst een eindscore berekend. De database met de hoogste eindpunten wordt de uiteindelijk gebruikte database. Ook hier worden alle ondernomen stappen en de toelichting genoteerd.

Het berekenen van de score per criteria gaat als volgt:
In stap 5:
Criterium: wij willen database die het snelst door 10000 records kan zoeken.
Weging: 3
Punten: de snelste krijgt 3 punten, de langzaamste 1 punt, de overige 2 punten.
In stap 6:
Shortlist: DB1, DB2, DB3
Testresultaten: snelste: DB2, langzaamste: DB1, midden: DB3
Puntenverdeling: DB2: 3, DB1: 1, DB3: 2
Weging criterium: 3
Berekening criterium: weging * punten
Score criterium: DB1: 1 * 3 = 3, DB2: 3*3 = 9, DB3: 2 * 3 = 6

Al deze stappen worden gecombineerd tot 1 document welke eindigt met een conclusie. Deze conclusie zal het volgende bevatten:

1. Een tabel met daarin de scores van de databases per criterium, en de totaalscores
2. De benoeming van de gekozen database.

Verder geldt voor alle stappen waar criteria wordt opgehaald: de criteria zal worden opgehaald en verwerkt tot een Word document. Dit document wordt per mail naar zowel de opdrachtgever als de technisch begeleider gemaild. Vervolgens wordt er pas verder gewerkt aan de verwerking van criteria op de databases als er een akkoord is gegeven door de opdrachtgever.

Het kan gedurende de pakketselectie voorkomen dat er onderdelen gebouwd moeten worden om iets aan te tonen. Een voorbeeld hiervan is als een van de criteria performance is, dan zal de performance van de overgebleven databases met elkaar vergeleken moeten worden. Een van de manieren om dit aan te tonen is, een applicatie schrijven, welke bij elke test tegen een andere database praat. Als het enige verschil tussen de tests de gebruikte database is, zal het verschil in performance waarschijnlijk hiervandaan komen.

Voor het verwerken van de criteria op de databases geldt dat alle ondernomen stappen, gebruikte links/documenten en uitgevoerde tests genoteerd moeten worden zodat alle belanghebbende deze stappen kunnen herproduceren, en zodat er gereviewed kan worden waar iets misging als er wat misging.

3.2.3 Bouwfase: Bouwen proof of concept

De bouwfase zal volgens SCRUM gedaan worden. De aanpak hiervan is beschreven in 3.1. Per sprint zal er een requirements, ontwerp, bouw en testfase zijn. Per sprint wordt er een shipable product opgeleverd aan de opdrachtgever. Dit shipable product bestaat uit documenten uit elke fase binnen de sprint, en een demo van de gebouwde backlogs. De fases per sprint staan hieronder beschreven. Verder is het zo, dat voorafgaand aan de eerste sprint, een eerste set backlogs wordt gedefinieerd en geprioriteerd.

De sub onderdelen van de bouwfase zijn aangepast aan de hand van de training projectplanning en aanpak op 24-2-2016. Reden van de wijziging is dat de volgorde van subfases niet overeenkwam met TDD.

Requirements/backlogs:
Aangezien de proof of concept gebaseerd zal worden op een bestaande applicatie, zullen er geen nieuwe requirements/backlogs opgehaald worden. Wat wel wordt gedaan is in het begin van elke sprint met de opdrachtgever overleggen welke backlogs in die sprint thuishoren.

De backlogs zullen geformuleerd worden als user stories. Deze user stories zullen opgesteld worden volgens INVEST:

- Independent, de user stories moeten onafhankelijk van elkaar worden opgesteld
- Negotiatable, tot de sprint waarin de user story uitgewerkt wordt mag de user story wijzigen
- Valuable, de user story moet van waarde zijn voor de eindgebruiker
- Estimable, de grootte van een user story moet altijd ingeschat kunnen worden
- Small, de user story mag niet zodanig groot worden dat het onmogelijk wordt deze te kunnen plannen/prioriteren
- Testable, de user story of de bijbehorende bijschrijving moet testbaar zijn

Verder wordt er per user story een aantal scenario's/voorbeelden opgesteld. Deze scenario's worden volgens de Gherkin taal opgesteld, zodat deze eenvoudig terug kunnen komen in de tests.

Deze fase resulteert elke sprint in een lijst met geplande backlogs(de sprint backlog).

Testen:
Voor het testen van de proof of concept moet aan 2 dingen gedacht worden:

1. Het vaststellen van de kwaliteit van de applicatie
2. Het aantonen dat eerder gemaakte beweringen over de database waar zijn.

Om de kwaliteit van de code vast te stellen zal, zoals eerder verteld, Test Driven Development gehanteerd worden. Aangezien de tests van stukken code opgesteld worden voordat de code geschreven wordt, en de code pas goed is als de test slaagt, wordt de kwaliteit van de code gewaarborgd.

De test zullen, zoals eerder verteld, opgesteld worden volgens Gherkin en zullen uitgevoerd worden met de tool SpecFlow. Op deze manier wordt verzekerd dat de applicatie doet wat ervan verwacht wordt. Mocht de code coverage van de applicatie na het uitvoeren van de hierboven beschreven tests lager zijn dan 80%, worden de tests aangevuld met behulp van unit tests op basis van TMap Next.

Hoe de gemaakte beweringen aangetoond worden is nog niet bekend. Dit is afhankelijk van wat er precies aangetoond moet worden.

Alle tests en de bijbehorende resultaten zullen gedocumenteerd worden en opgeleverd worden per sprint.

Ontwerpen:
Als duidelijk is wat er ontwikkeld gaat worden binnen de sprint, wordt het klassendiagram ontworpen. Hierbij worden in ieder geval de volgende diagrammen gemaakt/bijgewerkt:

1. Klasse diagram van de applicatie
2. Database diagram
3. Database implementatie model
4. Sequence diagram
5. Eventueel activity diagrammen

Dit zijn dan ook de documenten die voortkomen uit elke iteratie van deze fase.
Bouwen:
Het bouwen van de applicatie houdt simpelweg in dat de applicatie gebouwd gaat worden. Dit gebeurd op basis van de backlogs die aan het begin van de sprint zijn overeengestemd.

Uit deze fase komt elke sprint een shipable product, welke dus per sprint uitgebreider is.

Als niet duidelijk is wat een stuk code doet, dient hier commentaar geplaatst te worden.
Beslissingen/aannames document
Naast de 4 fases binnen het bouwen komt er ook een document waarin alle technische beslissingen en aannames gedocumenteerd zullen worden. Op deze manier kunnen er achteraf geen vragen overblijven.

3.2.4 Afstudeerverslag

Gedurende het gehele project is er een overkoepelende fase, namelijk het afstudeerverslag. Dit verslag is voor de opdrachtnemer en de school van belang. Het afstudeerverslag zal ook iteratief geschreven worden. Aan het eind van elke werkdag maakt de opdrachtnemer een log van die dag. Aan het eind van elke fase met uitzondering van de bouwfase zullen deze logs in het afstudeerverslag verwerkt worden. Zo zal het verslag even snel bijgewerkt worden als de producten per fase en loopt er niks achter. In de bouwfase wordt het afstudeerverslag per sprint bijgewerkt.

3.2.5 Afstudeerverdediging

Aan het eind van het traject is er de afstudeerverdediging. Voorafgaand aan de verdediging krijgt de opdrachtnemer een indicatie van zijn cijfer te horen. Deze indicatie kan zijn: Voldoende, Twijfel of onvoldoende. Deze indicatie is gebaseerd op het afstudeerverslag van de opdrachtnemer. Het verloop van de verdediging zelf is gebaseerd op deze indicatie. Per situatie verloopt de verdediging als volgt:

Voldoende:
De afstudeerverdediging is een openbare zitting, waarbij de opdrachtnemer maximaal 10 personen kan uitnodigen voor de zitting. Naast de opdrachtnemer, zullen de eerste begeleider en de examinatoren van de Haagse Hogeschool ook aanwezig zijn samen met de opdrachtgever vanuit het bedrijf. Mocht de technisch begeleider de mogelijkheid hebben om te komen, zal zij ook uitgenodigd worden.

Gedurende deze zitten houdt de opdrachtnemer eerst een presentatie van ongeveer 20 minuten over een onderwerp wat sterk gerelateerd is aan de afstudeeropdracht. Vervolgens zullen de examinatoren vragen aan de opdrachtnemer stellen op basis van zijn afstudeerdossier en de presentatie. Deze vragenronde zal ongeveer 30 tot 40 minuten duren.

Na de vragenronde zal de afstudeercommissie het eindcijfer bepalen, welke vervolgens aan de student zal worden medegedeeld. Het cijfer zal, tenzij de verdediging heel slecht ging, in ieder geval een voldoende zijn.

Twijfel:
De afstudeerverdediging is een besloten zitting. Ook hierbij geeft de student eerst een presentatie van ongeveer 20 minuten, welke sterk gerelateerd is aan de afstudeeropdracht. Vervolgens is er een ondervraging van ongeveer 30 tot 40 minuten, waarbij de examinatoren de vragen zullen stellen. Aan de hand van het afstudeerdossier en de vragenronde wordt vervolgens bepaald of:

- De student een voldoende heeft, inclusief het eindcijfer
- De student een onvoldoende heet, inclusief het eindcijfer
- De student herkansing krijgt

De mogelijkheid tot herkansen wordt alleen gegeven als de afstudeercommissie van mening is dat herkansen zinvol is. Deze mogelijkheid kan ook maar 1 keer gegeven worden. Als herkansen de beoordeling is, wordt er met de examinatoren besproken wat er verbeterd moet worden. Vervolgens heeft de student tot het eerstvolgende ingeplande inleverdatum om het verslag te verbeteren en voor te bereiden op de volgende verdediging.

Onvoldoende:
Als de eerste beoordeling onvoldoende is, vindt er een besloten zitting plaats zonder presentatie. Tijdens deze zitting wordt besloten of de student een herkansing krijgt, of dat de student een onvoldoende krijgt. Het is niet mogelijk om in deze zitting nog een voldoende te krijgen.

Voorbereiding:
Aangezien de gegeven indicatie pas een aantal dagen voor de verdediging bekend is, moeten de presentatie en de verdediging in ieder geval voorbereid worden. Deze voorbereiding ben ik van plan als volgt aan te pakken:

Eerst kies ik een onderwerp wat ik ben tegengekomen tijdens mijn afstudeerperiode. Vervolgens bepaal ik wat ik over dit onderwerp wil vertellen, en wat niet. Mocht ik gedurende de afstudeerperiode niet genoeg informatie over de gekozen onderdelen hebben vergaard, zal ik deze informatie ophalen nadat de onderdelen zijn gekozen.

Vervolgens ga ik de onderdelen van het onderwerp opsplitsen tot een logische presentatie. Met logisch wordt bedoeld: een volgorde van onderwerpen wat te begrijpen is door een buitenstaander. Deze volgorde en opgehaalde informatie worden geverifieerd bij de technisch begeleider.

Als ik de te vertellen informatie in een representatieve PowerPoint heb verwerkt, ga ik zelf op zoek naar al bekende, of juist onbekende dingen van het desbetreffende onderwerp. De reden dat ik hiernaar ga zoeken, is dat er een kans bestaat dat hier vragen over zullen komen. Naast bekende en onbekende dingen over het onderwerp, deel ik ook mijn presentatie met iemand die wel verstand heeft van de branch, ICT, maar niet zozeer kennis

heeft van het onderwerp. Vragen die diegene mij zal stellen zijn ook mogelijke vragen die ik ga krijgen op de verdediging.

Tot slot zal ik een oefen presentatie geven. Deze presentatie zal in ieder geval gegeven worden aan de technisch begeleider en de opdrachtgever. De procesbegeleider, businessunit manager en overige medewerkers van Info Support en de Vries WFM zijn ook welkom om deze presentatie bij te wonen. Van de oefen presentatie verwacht ik feedback en vragen van de bijwonende om zo de daadwerkelijke presentatie te verbeteren.

3.3 Planning

Aan de hand van de inhoud van de fases ben ik met de volgende planning gekomen:

Fase	Aantal weken	Opmerkingen
Opstartfase: onderzoek ' Bij welke modules van R&R web zijn er voordelen te behalen bij het overstappen naar document oriented databases zonder dat dit ten koste gaat van de huidige functionaliteit, en welke gevolgen heeft deze overstap?’	3	Voor dit verslag is een halve week speling genomen voor het geval van uitloop
Opstartfase: Pakketselectie	3	De pakketselectie dient grondig uitgevoerd te worden, waarbij meerdere male contact opgenomen dient te worden met de opdrachtgever. Vooral van de shortlist tot de uiteindelijke winnaar komen kan wat meer tijd kosten.
Bouwfase: bouwen proof of concept	10	Voor het bouwen van de proof of concept zal minimaal 10 weken genomen worden. Dit staat gelijk aan 5 sprints.
Afstudeerverslag	18	Het afstudeerverslag is een overkoepelende fase die gedurende het hele project uitgevoerd zal worden

4. Beheers aspecten

4.1 Organisatie

4.1.1 Normstelling

Binnen de organisatie van dit project zijn er verschillende lagen te herkennen. Deze lagen zijn:

1. De stuurgroep, deze neemt geen inhoudelijke beslissingen overziet alleen het geheel
2. Projectleiding, deze gaat over wat er inhoudelijk binnen het project uitgevoerd zal moeten worden en wat de kwaliteit van deze producten is.
3. Ontwikkelaars, deze houden zich bezig met het leveren van de inhoud van het project
4. Support, de supportafdeling is verantwoordelijk voor het werkend houden van ondersteunende zaken zoals bijvoorbeeld: het opzetten van een werkomgeving.
5. Klankbordgroep, deze bestaat uit iedereen die invloed heeft op het project en erbij betrokken is.

Per laag ziet dit er als volgt uit:

Stuurgroep:
De stuurgroep van dit project bestaat uit de begeleiders en examinatoren vanuit de Haagse Hogeschool. Zij gaan niet zozeer over wat er inhoudelijk moet gebeuren aan het project. Wel gaan zij over het verloop van het proces, en beslissingen die genomen zijn binnen het traject.

Verder zit ook de businessunit manager in de stuurgroep. Hij gaat namelijk ook niet over de inhoud van het project, maar zal wel willen weten hoe het project verloopt.

Projectleiding:
De projectleiding bestaat uit degene die gaan over de inhoud van het project. Dit houdt in zowel de inhoudelijk verwachtingen aangeven en controleren, als de kwaliteit van de inhoud vaststellen en waarborgen. In dit project bestaat deze groep uit 2 personen, namelijk: Rick Megens, de opdrachtgever en Erma van Alebeek, de technische begeleider.

Rick Megens zal zich als opdrachtgever bezighouden met: wat verwacht ik inhoudelijk van de opdrachtnemer, en heb ik gekregen wat ik wilde.
Erma van Alebeek is als technisch begeleider verantwoordelijk voor de inhoudelijke kwaliteit. Zij zal controleren of de inhoudelijke kwaliteit van de opgeleverde producten volstaat, en geeft feedback aan de ontwikkelaars aan de hand van haar bevindingen.

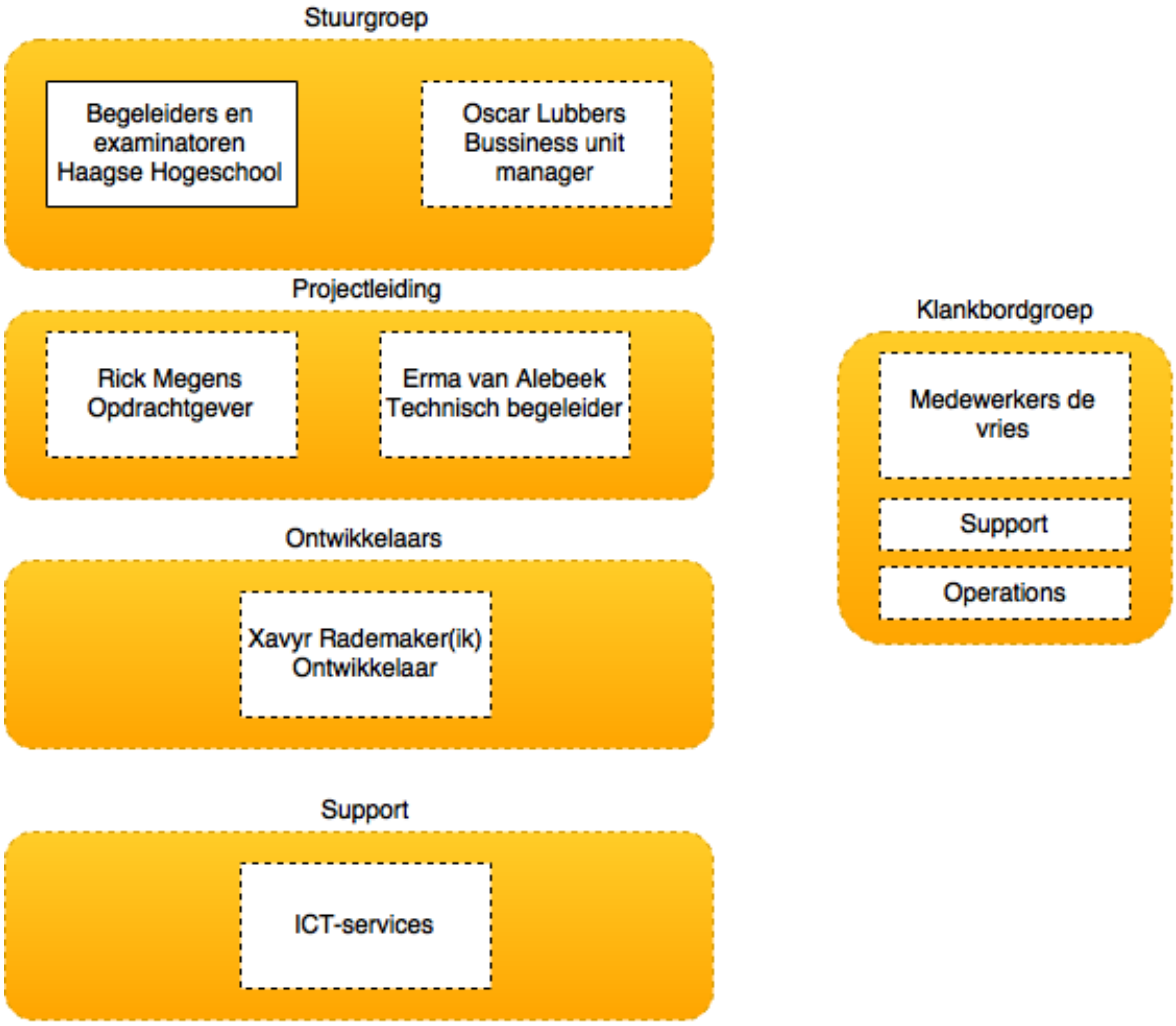
Ontwikkelaars:
De ontwikkelgroep bestaat in dit project uit 1 persoon, namelijk de opdrachtnemer (Xavyr Rademaker). Hij is verantwoordelijk voor het opleveren van de door de opdrachtgever verwachte producten.

Support:
Binnen Info Support bv. is er een afdeling genaamd ICT-services(Systeembeheer). Deze afdeling levert support waar nodig. Dit kan gaan over het opzetten van omgevingen, tot verkrijgen van nieuwe software en licenties.

Klankbordgroep:
De klankbordgroep bestaat in dit project uit de medewerkers van de Vries WFM. De vraag naar dit project komt vanuit hun, en zij zullen ook geïnteresseerd zijn in de resultaten.

Verder kan hun mening van invloed zijn op bepaalde beslissingen die door de stuurgroep en/of de projectleiding genomen worden.

In de volgende afbeelding is de structuur van de organisatie te zien:



4.1.2 Voortgangscontrole

De voortgang wordt gecontroleerd door de procesbegeleider: Marieke Keurntjes. Hiervoor wordt 1maal per 2 weken een afspraak ingepland. Verder zal de opdrachtgever het proces ook controleren.

4.2 Informatie

4.2.1 Normstelling

Alles wat uitgevoerd wordt door de opdrachtnemer wordt gedocumenteerd. Deze documenten zullen een bepaalde status hebben. Deze status houdt het volgende in:

Versie	Toelichting
1	Aanmaken van het document, eerste versie van het document
1.1	Er is een aanleiding geweest om wijzigingen te maken, dit is een tussenversie met de wijzigingen verwerkt
2	De opdrachtnemer denkt dat dit een goede versie is en wacht goedkeuring/feedback af
2.1	Er was feedback, een deel van de feedback is verwerkt maar nog niet alles
3	Alle feedback is verwerkt, als er geen nieuwe feedback is: de laatste versie

Dit kan eindeloos doorgaan, maar het komt erop neer dat:

1. Alle versies met decimalen zijn versies waar nog wijzigingen in doorgevoerd moeten worden (met versie 1 als uitzondering).
2. Alle versies zonder decimalen (met versie 1 als uitzondering) zijn versies waarvan de opdrachtnemer denkt dat het alles omvat en dus een akkoord of niet akkoord omdat verwacht.

Verder zullen alle documenten de volgende opbouw hebben:

1. Voorblad
2. Historie
3. Management samenvatting
4. Inhoudsopgave
5. Inhoud
6. Conclusie
7. Eventuele bijlage

Verder zullen alle documenten ten alle tijden opgestuurd worden naar de technische begeleider. Dit wordt per mail gestuurd en wordt gedaan zodra een onderdeel binnen het document af is. In de mail moet wel vermeld worden wat de status van het document is en wat ik ervan verwacht (wel feedback, geen feedback).

Zodra een onderdeel van een document af is, zal ik deze ook naar de opdrachtgever sturen. Dit zal meer ter informatie zijn, zodat hij niet achteraf een heel groot document hoeft te lezen. Er zullen momenten zijn dat de opdrachtgever feedback, of een akkoord moet geven. Dit zal ik dan ook van hem vragen.

In de onderstaande tabel is te zien wie welke informatie wanneer ontvangt:

Document	Wie	Wanneer	Doel
Plan van aanpak	Opdrachtgever	Na de afronding per onderdeel	Informatief, geven feedback
Plan van aanpak	Opdrachtgever	Eindversie	Geven akkoord
Plan van aanpak	Technisch begeleider	Na de afronding per onderdeel	Informatief, geven feedback
Plan van aanpak	Technisch begeleider	Eindversie	Informatief
Onderzoek: 'Bij welke modules van R&R-web zijn er voordelen te behalen bij het overstappen naar document oriented databases zonder dat dit ten koste gaat van de huidige functionaliteit, en welke gevolgen heeft deze overstap?'	Opdrachtgever	Na de afronding per onderdeel	Informatief
Onderzoek: 'Bij welke modules van R&R-web zijn er voordelen te behalen bij het overstappen naar document oriented databases zonder dat dit ten koste gaat van de huidige functionaliteit, en welke gevolgen heeft deze overstap?'	Opdrachtgever	Eindversie	Geven akkoord
Onderzoek: 'Bij welke modules van R&R-web zijn er voordelen te behalen bij het overstappen naar document oriented databases zonder dat dit ten koste gaat van de huidige functionaliteit, en welke gevolgen heeft deze overstap?'	Technisch begeleider	Na de afronding per onderdeel	Informatief, geven feedback

Onderzoek: 'Bij welke modules van R&R-web zijn er voordelen te behalen bij het overstappen naar document oriented databases zonder dat dit ten koste gaat van de huidige functionaliteit, en welke gevolgen heeft deze overstap?'	Technisch begeleider	Eindversie	Informatief
Analyse huidige applicatie	Opdrachtgever	Na de afronding per onderdeel	Informatief
Analyse huidige applicatie	Opdrachtgever	Eindversie	Geven akkoord
Analyse huidige applicatie	Technisch begeleider	Na de afronding per onderdeel	Informatief, geven feedback
Analyse huidige applicatie	Technisch begeleider	Eindversie	Informatief
Pakketselectie	Opdrachtgever	Na de afronding per onderdeel	Geven akkoord
Pakketselectie	Opdrachtgever	Eindversie	Geven akkoord
Pakketselectie	Technisch begeleider	Na de afronding per onderdeel	Informatief, geven feedback
Pakketselectie	Technisch begeleider	Eindversie	Informatief
Backlogs totaal	Opdrachtgever	Zodra ze zijn opgesteld	Geven akkoord
Backlogs totaal	Technisch begeleider	Zodra ze zijn opgesteld	Geven feedback
Backlogs per sprint	Opdrachtgever	Zodra ze zijn ingedeeld	Geven akkoord
Backlogs per sprint	Technisch begeleider	Zodra ze zijn ingedeeld	Geven feedback
Ontwerpen proof of concept	Opdrachtgever	Elke sprint zodra deze is bijgewerkt met de huidige backlogs	Geven akkoord
Ontwerpen proof of concept	Technisch begeleider	Elke sprint zodra deze is bijgewerkt met de huidige backlogs	Geven feedback
Testendocumenten proof of concept	Opdrachtgever	Aan het eind van elke sprint	Informatief
Testendocumenten proof of concept	Technisch begeleider	Zodra deze gedaan zijn in de sprint	Geven feedback
Beslissingen document	Opdrachtgever & technisch begeleider	Bij de ontwerpdocumenten als er beslissingen zijn genomen	Informatief

Demo applicatie	Opdrachtgever	Aan het eind van elke sprint	Informatief, geven feedback, geven akkoord
Afstudeerverslag	Technisch begeleider	Bij elke wijziging	Geven feedback
Afstudeerverslag	Begeleiders/examinatoren HHS	3 juni	beoordelen

4.2.2 Voortgangscontrole

Zowel de technische begeleider als de opdrachtgever houden de voortgang in de gaten. Hiervoor zijn er eens in de 2 weken afspraken gepland met de opdrachtgever, en eens in de 2 weken met de technisch begeleider. De weken zijn afwisselend, waardoor er dus elke week een controle plaatsvindt. Verder wordt de voortgang ook duidelijk in de sprint reviews.

4.3 Tijd

4.3.1 Normstelling

Voor de eerder benoemde mijlpalen (de op te leveren documenten per fase) gelden de volgende start en einddata:

Document	Verwachte start week	Verwachte start datum	Verwachte eind week	Verwachte einddatum
Onderzoek: Onderzoek: 'Bij welke modules van R&R-web zijn er voordelen te behalen bij het overstappen naar document oriented databases zonder dat dit ten koste gaat van de huidige functionaliteit, en welke gevolgen heeft deze overstap?'	3	15-2-2016	5	4-3-2016
Pakketselectie	6	7-3-2016	8	25-3-2016
Bouwen(incl. alle documentatie)	Totaal :9 Per sprint: 1. 9 2. 11 3. 13 4. 15 5. 17	Totaal: 28-3-2016 Per sprint: 1. 28-3-2016 2. 11-4-2016 3. 25-4-2016 4. 9-5-2016 5. 23-5-2016	Totaal: 17 Per sprint: 1. 10 2. 12 3. 14 4. 16 5. 17	Totaal: 27-5-2016 Per sprint: 1. 7-4-2016 2. 21-4-2016 3. 5-5-2016 4. 19-5-2016 5. 27-5-2016
Afstudeerverslag	1	5-2-2016	18	3-6-2016
Afstudeerzitting*	-	Eind juni	-	Begin juli

*de afstudeerzitting is 1 dag, de exacte datum wordt pas in juni bekend gemaakt.

4.3.2 Voortgangscontrole

Zodra een onderdeel af is en is goedgekeurd wordt de eindweek en de einddatum genoteerd. Deze wordt vergeleken met de bovenstaande tabel, om zo te zien of er vertraging is opgelopen, of er tijd is bespaard of dat er precies op schema wordt gewerkt. Verder houden de technische begeleider, de opdrachtgever, de opdrachtnemer en de procesbegeleider de voortgang ook in de gaten.

4.4 Geld

4.4.1 Normstelling

Voor het project is momenteel geen budget benodigd. Wellicht dat dit in de loop van het project wel benodigd zal zijn. Dat zal tegen die tijd ingepland worden.

4.4.2 Voortgangscontrole

Wordt ingevuld zodra het nodig is.

4.5 Middelen

4.5.1 Normstelling

De volgende middelen/tools zijn benodigd gedurende dit project:

1. Een werkplek met internet
2. De beschikbaarheid en betrokkenheid van de opdrachtgever
3. De vrijheid voor de opdrachtnemer om uren voor school te kunnen maken
4. Een ontwikkelomgeving welke vanaf de werkplek benaderd kan worden
5. De beschikbaarheid en betrokkenheid van de technisch begeleider
6. De beschikbaarheid en betrokkenheid van de opdrachtnemer

Van de benodigde middelen is het volgende minimaal vereist:

Middel	Eis
Werkplek met internet	Er hoort gedurende het gehele project altijd een werkplek te zijn. Verder mag het hooguit 1 dag in de hele periode voorkomen dat er geen internet aanwezig is op de werkplek
Beschikbaarheid en betrokkenheid van de opdrachtgever	Er wordt van de opdrachtgever verwacht dat hij: 1. Geplande persoonlijke afspraken nakomt, of als er iets tussenkomt verplaatst naar een moment binnen 3 dagen later. Een afspraak mag altijd naar voren verplaatst worden en kan ook verplaatst worden door de opdrachtnemer. 2. Als er feedback verwacht wordt, wordt deze binnen een week verwacht 3. Als de opdrachtgever een mail moet beantwoorden/feedback moet geven heeft hij 3 dagen om aan te geven dat hij het gezien heeft 4. Hij binnen werktijden bereikbaar is voor korte vragen, dit kan ook over de mail zijn.
De vrijheid voor de opdrachtnemer om uren voor school te kunnen maken	De opdrachtnemer zal af en toe naar school moeten of tijd moeten inplannen voor afspraken met de begeleider/het maken van zijn afstudeerverslag. Deze tijd mag hij ten alle tijden inplannen, tenzij het project vertraging heeft opgelopen.
Een ontwikkelomgeving welke vanaf de werkplek benaderd kan worden	Vanaf het moment dat de bouwfase begint, dient er een ontwikkelomgeving te zijn. Deze omgeving moet het volgende bevatten: 1. Tools benodigd om te coderen 2. De te gebruiken database 3. Tools om de kunnen ontwerpen 4. Eventueel tools om te testen
De beschikbaarheid en betrokkenheid van de technisch begeleider	Van de technisch begeleider wordt het volgende verwacht: 1. Het controleren van de inhoudelijke kwaliteit van opgestuurde producten 2. Het geven van feedback als er iets heel erg mis is in een product, of als er gevraagd is om feedback

De beschikbaarheid en betrokkenheid van de opdrachtnemer	Van de opdrachtnemer wordt het volgende verwacht: 1. Hij dient proactief te zijn in de gesprekken met de opdrachtgever 2. Hij dient er zelf achteraan te gaan als hij iets nodig heeft 3. Hij dient vragen te stellen indien nodig 4. Hij dient de eindversies van alle gevraagde producten voor/op de uiterlijke einddatum in de leveren bij zowel de technisch begeleider als de opdrachtgever

Toelichting:
De toelichting per benodigd middel is als volgt (de nummers komen overeen met de nummers boven de tabel):

1. De opdrachtnemer heeft een werkplek nodig om dit project uit te kunnen voeren. Voor zowel het ontwikkelen als het onderzoeken is internet nodig, om zo snel bij informatie te komen.
2. Van de opdrachtgever wordt verwacht dat hij beschikbaar is om alle vragen te kunnen beantwoorden die vanuit de opdrachtnemer komen. Ook is zijn beschikbaarheid nodig om feedback te geven op alles waar hij het niet mee eens is.
3. Dit is nodig, omdat de opdrachtnemer naast verantwoordelijkheden naar de opdrachtgever ook verantwoordelijkheden richting school heeft. Deze verantwoordelijkheden moeten uitgevoerd worden om een goed eindcijfer te krijgen vanuit de school.
4. De opdrachtnemer heeft een omgeving nodig waar hij kan ontwikkelen. Deze omgeving moet te benaderen zijn vanaf de werkplek beschreven in punt 1.
5. De technisch begeleider dient beschikbaar te zijn voor het beantwoorden van vragen, geven van feedback op document inhoudelijk punten en overige inhoudelijke begeleiding.
6. De opdrachtnemer dient beschikbaar te zijn om meetings en vragen van de opdrachtgever en de technisch begeleider te beantwoorden. Daarnaast dient hij betrokken genoeg te zijn om niet alleen als student, maar ook als expert te handelen om zo de opdrachtgever te geven wat hij wilt.

4.5.2 Voortgangscontrole

De opdrachtnemer zal in de gaten houden of alle middelen voldoen aan de eisen, met uitzondering van zichzelf, gedurende het project. Gaat er iets mis dan zal hij dat aankaarten bij de verantwoordelijke en de technisch begeleider of de procesbegeleider.
Het middel dat gaat over de opdrachtnemer zelf zal gecontroleerd worden door de opdrachtgever, de technisch begeleider en de procesbegeleider.

4.6 Kwaliteit

4.6.1 Normstelling

Productkwaliteit:
De kwaliteit van de proof of concept wordt vastgesteld door middel van tests. Hiervoor is eerder vastgesteld dat er een minimale code coverage is van 80%, en wordt er gebruik gemaakt van risico gedreven testen. Oftewel, de groter het risico, de beter er getest wordt. Verder zijn er in het hoofdstuk kwaliteitseisen een aantal producteisen gesteld:
Klanten mogen van elkaar geen data zien
Afnemers van de R&R-webapplicatie hebben elk hun eigen deel van de database. Deze klant A mag dus niet de gegevens zien van klant B.
De codeerstandaarden van Microsoft moeten gehanteerd worden
Alle geschreven code moet deze codeerstandaarden aanhouden

Verder wordt van alle documenten verwacht dat ze voldoen aan hbo-niveau.
Proceskwaliteit:
Van het proces wordt verwacht dat alle gemaakte afspraken worden nagekomen. Mochten er onverwachts dingen gebeuren waardoor een afspraak, een feedback of iets dergelijks niet zoals afgesproken kan verlopen dan zal dit vermeld moeten worden inclusief de reden waarom en hoe het opgelost gaat worden (bijvoorbeeld de afspraak verplaatsen).
Gebruikerskwaliteit:
De kwaliteit die de gebruiker van het systeem verwacht wordt vastgesteld in de backlogs en in de criteria voor de pakketselectie, dus zal dit in de loop van het project vastgesteld worden. Verder geeft de gebruiker ook bij elk document waarop hij akkoord of feedback moet geven aan of het aan zijn eisen voldoet.

4.6.2 Voortgangscontrole

Productkwaliteit:
De opdrachtgever zal elk document wat wordt opgeleverd voor een akkoord reviewen en aangeven of het is wat hij wilt of niet. Verder zal de technisch begeleider ook alle documenten nalezen en feedback geven waar nodig.
Proceskwaliteit:
Zowel de opdrachtgever als de procesbegeleider als de technisch begeleider zullen controleren of alle afspraken worden nagekomen. Verder zal ook de opdrachtnemer de begeleiders aan hun afspraken houden
Gebruikerskwaliteit:
De opdrachtgever zal van elk document beoordelen of het is wat hij ervan verwacht. Zo ja zal hij hiermee akkoord gaan, zo nee zal hij feedback geven. Dit geld voor alle documenten, documentatie en demo’s.

4.7 Overlegvormen

De volgende overlegvormen zijn er binnen die project:

Overlegvorm	Met wie	Hoe	Frequentie	Doel
Voortgangsgesprek	technisch begeleider	Persoonlijk /skype	1 keer in de 2 weken	Inhoudelijke voortgang bespreken en mogelijkheid tot beantwoorden van vragen
Voortgangsgesprek	Opdrachtgever	Persoonlijk	1 keer in de 2 weken	Bespreken voortgang met betrekking tot wat er opgeleverd word
Voortgangsgesprek	Procesbegeleider	Persoonlijk	1 keer in de 2 weken	Bespreken hoe het proces verloopt, worden afspraken nagekomen enzovoorts.
Dagelijkse review	Mezelf	Persoonlijk	1 keer per dag	Reviewen hoe de dag ging, wat er beter kan, wat er morgen gebeurd
Sprint planning	Opdrachtgever	Persoonlijk	Begin van elke sprint	Backlogs van de huidige sprint inplannen en specificeren
Sprint review/retrospective	Opdrachtgever	Persoonlijk	Eind van elke sprint	Afgelopen sprint reviewen, en kijken hoe de volgende sprint verbeterd kan worden.
Sprint presentatie	Opdrachtgever	Persoonlijk	Eind van elke sprint	Met de opdrachtgever zitten om te bespreken wat er gedaan is, en het geven van een demo
Feedback vragen/overdragen & op de hoogte stellen van documenten	Technisch begeleider/ Opdrachtgever	Mail	Na afronding van een onderdeel binnen een document	De opdrachtgever en de technisch begeleider op de hoogte stellen van de ontwikkelingen rondom dit project en eventueel vragen om feedback

Voortgangsgesprek	Eerste begeleider	Persoonlijk	-	Voortgang van het stageverslag bekijken, eventueel feedback geven/vragen beantwoorden
Verdediging	Eerste begeleider, examinatoren	Persoonlijk	1 keer gedurende de afstudeerperiode	Afstudeerverslag van de opdrachtnemer beoordelen, inclusief mogelijkheid om vragen te stellen.

4.8 Communicatie

Fase/proces	Ondersteunen	Uitvoeren	Beslissen (goedkeuren)	Gebruiken (informeren)
Onderzoek: Onderzoek: 'Bij welke modules van R&R-web zijn er voordelen te behalen bij het overstappen naar document oriented databases zonder dat dit ten koste gaat van de huidige functionaliteit, en welke gevolgen heeft deze overstap?' - Tussentijdse versies	Technisch begeleider	Opdrachtnemer	Technisch begeleider	Opdrachtgever
Onderzoek: 'Bij welke modules van R&R-web zijn er voordelen te behalen bij het overstappen naar document oriented databases zonder dat dit ten koste gaat van de huidige functionaliteit, en welke gevolgen heeft deze overstap?' - Eindversie	Technisch begeleider	Opdrachtnemer	Opdrachtgever, technisch begeleider	Technisch begeleider
Analyse huidige applicatie - Tussentijdse versie	Technisch begeleider	Opdrachtnemer	Technisch begeleider	Opdrachtgever

Analyse huidige applicatie - Eindversie	Technisch begeleider	Opdrachtnemer	Opdrachtgever, technisch begeleider	Technisch begeleider
Pakketselectie - Tussentijdse versie	Technisch begeleider	Opdrachtnemer	Opdrachtgever, Technisch begeleider	Technisch begeleider
Pakketselectie - Eindversie	Technisch begeleider	Opdrachtnemer	Opdrachtgever, technisch begeleider	Technisch begeleider
Bouwen - Backlogs totaal en per sprint	Technisch begeleider	Opdrachtnemer	Opdrachtgever, technisch begeleider	Technisch begeleider
Bouwen - Ontwerpen tussentijdse (dus niet de definitieve versie van die sprint)	Technisch begeleider	Opdrachtnemer	Technisch begeleider	Opdrachtgever
Bouwen - Ontwerpen eindversie per sprint	Technisch begeleider	Opdrachtnemer	Opdrachtgever, technisch begeleider	Technisch begeleider
Bouwen - Bouwen applicatie	Technisch begeleider	Opdrachtnemer	Opdrachtgever, technisch begeleider	Technisch begeleider
Bouwen - Testen tussentijdse (dus niet de definitieve versie van die sprint)	Technisch begeleider	Opdrachtnemer	Technische begeleider	Opdrachtgever

Bouwen - Testen eindversie per sprint	Technisch begeleider	Opdrachtnemer	Opdrachtgever, technisch begeleider	Technisch begeleider
Afstudeerverslag	Technisch begeleider	Opdrachtnemer	Begeleiders en examinatoren van de Haagse Hogeschool	Technisch begeleider

Toelichting:
Voor alle inhoudelijk ondersteuning is de technisch begeleider bedoeld. Alles wordt uitgevoerd door de opdrachtnemer. De technische begeleider houdt ook bij of de inhoud van alle opleveringen van voldoende kwaliteit is, en moet hier dus ook mee akkoord gaan. Dit akkoord wordt gegeven in de vorm van feedback, er wordt dus gewoon doorgewerkt zonder dat akkoord. De opdrachtgever wordt van alle tussentijdse versies op de hoogte gehouden, zodat de voortgang in de gaten kan houden en niet achteraf hele documenten hoeft te lezen.

Voor de eindversies geld dat zowel de opdrachtgever als de technisch begeleider akkoord moeten gaan. Het akkoord van de opdrachtgever is bindend. Dit wil zeggen dat er gewacht moet worden tot dit akkoord gegeven is om verder te gaan naar de volgende stap. Dit akkoord houdt in: is dit wat ik wil of niet. De technisch begeleider dient akkoord te gaan met de kwaliteit van de eindversies.

Mocht het voorkomen dat een van de twee niet akkoord is gegaan, moet het onderdeel worden verbeterd en moet er nogmaals een akkoord gegeven worden door beide. De reden hiervoor is dat op het moment dat het onderdeel is aangepast, het misschien niet meer van voldoende inhoudelijk kwaliteit is, of niet meer is wat de opdrachtgever wilt.

5. Oplevering

De volgende producten worden gedurende dit project opgeleverd:

Product	**Verwachte (uiterlijke) einddatum**	**Werkelijke einddatum**	**Akkoord technisch begeleider**	**Akkoord opdrachtgever**
Plan van aanpak	12-2-2016	11-2-2016	Akkoord 19-2-2016	Akkoord 25-2-2016
Onderzoek: Onderzoek: 'Bij welke modules van R&R-web zijn er voordelen te behalen bij het overstappen naar document oriented databases zonder dat dit ten koste gaat van de huidige functionaliteit, en welke gevolgen heeft deze overstap?'	4-3-2016	9-3-2016 De weken erna enkel feedback verwerkt wat geen invloed had op de inhoud van het document	Akkoord 30-3-2016	Akkoord 14-3-2016
Pakketselectie	25-3-2016	31-3-2016		
Bouwen(incl. alle documentatie)	27-5-2016			
Afstudeerverslag	3-6-2016	3-6-2016		

Bijlage A: Begrippenlijst

Workforce Management: Het inplannen van de mensen op de juiste plaats tegen de juiste loonkosten op het juist moment.

R&R-web: Retail&Resultaat (voorheen Rooster&Realisatie). De applicatie die de Vries WFM gebruikt om het optimaliseren van workforce management.

Pakketselectie: Een onderzoek dat achterhaald welke bestaande software het beste bij de eisen en wensen van de opdrachtgever past.

Bijlage B: Aangepaste aanpak

In deze bijlage wordt de aangepaste aanpak(hoofdstuk 3) weergeven per mijlpaalproduct waarvan de aanpak gewijzigd is voordat er aan het product gewerkt is.

B.1 Bouwfase: Bouwen proof of concept

De aanpak van de bouwfase (aanpak per sprint) is ligt gewijzigd aan de hand van de training projectplanning en aanpak op 24-2-2016. Wat er wijzigt is de volgorde van de subfases per sprint, vanwege het ontwikkelen volgens TDD.

Requirements/backlogs:
Voor deze subfase zijn er geen wijzigingen.

Testen:
Zoals eerder verteld zal er gebruik gemaakt worden van TDD. Hierdoor zullen de tests opgesteld worden nadat de sprintbacklog klaar is. Hierbij wordt als eerst de SpecFlow test opgesteld(dus 1 scenario). Mocht de scenario groot zijn, wordt er vervolgens een kleinere test opgesteld, die een deel van de scenario test. Deze test moet zodanig opgesteld worden dat hij faalt zodra die geschreven is. De tests worden constant uitgebreid(nieuwe kleine tests opstellen) tot het scenario uit SpecFlow slaagt. Zodra deze is geslaagd, ga ik verder met de volgende scenario.

Mocht het zo zijn dat het scenario klein genoeg is, worden er geen kleine tests uitgevoerd.

Ontwerpen:
In kader van TDD wordt het technisch ontwerp na elke test uitgebreid om die desbetreffende test te laten slagen. Als er bijvoorbeeld een test is die moet valideren, zal er ergens een valideer methode toegevoegd moeten worden. Op het moment dat de test is geslaagd(na het coderen) moet er refactored worden. Het refactoren gebeurd als volgt:

- De geschreven code om te test te laten slagen wordt opgesplitst in losse methodes(indien de methode iets doet waar hij niet verantwoordelijk voor is), maar blijven binnen dezelfde klasse
- De nieuwe methodes worden in het klassendiagram toegevoegd
- Het klassendiagram wordt bijgewerkt waar nodig(splitsen van methodes naar nieuwe klassen en dergelijk)
- De code wordt bijgewerkt aan de hand van het nieuwe ontwerp

Aan ontwerpen worden de volgende diagrammen gemaakt:

- Klasse diagram van de applicatie
- Database diagram
- Database implementatie model
- Sequence diagram
- Eventueel activity diagrammen

Bouwen:
Het bouwen gebeurt na het ontwerp, en zal zo minimaal mogelijk uitgevoerd worden om de test te laten slagen(net als het ontwerp). Op het moment dat de test slaagt, zal de code refactored worden. Dit gaat, zoals hiervoor beschreven, heel nauw samen met het ontwerpen.

Bijlage B: Onderzoek

Bij welke modules van R&R-web zijn er voordelen te behalen bij het overstappen naar document oriented databases zonder dat dit ten koste gaat van de huidige functionaliteit, en welke gevolgen heeft deze overstap?

Xavyr Rademaker

Management samenvatting

In opdracht van de Vries WFM is er onderzocht waar er voordelen te behalen vallen met het gebruik van document databases binnen de huidige applicatie zonder dat dit ten koste gaat van de huidige functionaliteiten.

De reden dat zij dit willen weten is dat zij van plan zijn hun gebruikersgroep van de applicatie R&R-web uit te breiden. Nu vragen zij zich af of het gebruik van document databases deze uitbreiding misschien beter aankan dan de huidige SQL-server database.

Het doel van dit onderzoek is dan ook antwoord geven op de vraag: Bij welke modules van R&R-web zijn er voordelen te behalen bij het overstappen naar document oriented databases zonder dat dit ten koste gaat van de huidige functionaliteit, en welke gevolgen heeft deze overstap?

Met het antwoord op deze vraag, weet het bedrijf waar document databases voordelen bieden, welke voordelen deze bieden en wat de gevolgen van de overstap zijn.

Aan de hand van het onderzoek, kan er geconcludeerd worden dat er voor de 4 grote modules van de applicatie voordelen te behalen valt bij het gebruik van document databases. Een aantal voordelen die er te behalen zijn, zijn:

- Horizontale schaalbaarheid
- Performancewinst

Een van de redenen dat de performancewinst de behalen valt is, dat er binnen de huidige applicatie veel joins uitgevoerd worden op tabellen die enkel gebruikt worden om te filteren. Deze data kan makkelijk embed worden in de documenten, om zo het document zelf te query'en in plaats van meerdere tabellen.

De horizontale schaalbaarheid is een eigenschap van document (en NOSQL) databases, omdat er alle documenten op zichzelf staande dingen zijn. Ze hebben dus geen afhankelijkheden naar andere documenten. Deze onafhankelijkheid maakt het mogelijk de database op te delen over meerdere servers (de data op de verschillende servers zal in principe niet samen opgehaald hoeven worden).

Dit zal wel een aantal gevolgen met zich meebrengen, namelijk:

- De ACID-regels op transacties worden niet gegarandeerd
- De verantwoordelijkheden en functionaliteiten die momenteel op/in de database zitten, moeten worden verplaatst naar de applicatie.

Inhoudsopgave

1. Inleiding

In dit hoofdstuk wordt besproken wat de aanleiding van dit onderzoek is, welk probleem er opgelost moet worden en wat de doelstelling van dit onderzoek is.

1.1 Aanleiding

Het bedrijf de Vries WFM is van plan zijn klantengroep uit te breiden naar Duitsland. Bij deze uitbreiding komt kijken dat er meer klanten, en dus meer gegevens in hun applicatie (R&R web), in hun applicatie komen te staan. Het bedrijf heeft gehoord dat document databases wellicht een betere oplossing zijn voor deze uitbreiding, in plaats van hun huidige SQL-server database.

Reden dat zij dit denken is dat zij merken dat steeds meer bedrijven/projecten overstappen naar document databases. Hieruit concluderen zij dat deze bedrijven/projecten een voordeel konden behalen uit deze vorm van databases. Zij vragen zich dus af of er bij hun applicatie, of in een deel daarvan, ook een voordeel te behalen valt.

Behalve dat steeds meer bedrijven/projecten overstappen, heeft het bedrijf ook gehoord dat de kosten voor schalen lager zijn bij document databases en dat data in document databases als het ware in groepen losse data wordt opgeslagen. Verder weten zij niet wat document databases zijn of hoe deze benaderd moeten worden.

1.2 Probleemstelling

Het probleem is dat de Vries WFM niet weet wat document databases hun kunnen bieden en dus niet weet bij welke modules dit ingezet kan worden om er voordelen uit te behalen, zonder huidige functionaliteiten te verliezen. Het is voor hun namelijk van belang dat zij er op het gebied van functionaliteit niet op achteruit gaan als er overgestapt wordt.

1.3 Doelstelling

Het doel van dit onderzoek is het achterhalen welke modules van R&R-web voordelen kunnen behalen bij het overstappen naar document databases, zonder dat dit ten koste gaat van de huidige functionaliteiten.

Om dit doel te behalen zijn de volgende deelvragen opgesteld:

- Wat zijn relationele databases?
- Wat is NOSQL en welke vormen zijn er hiervan?
- Wat zijn document databases?
- Welke modules zijn er in de huidige applicatie en wat doen deze?
- Wat zijn de kern functionaliteiten van de modules van de applicatie?
- Welke niet functionele eisen zijn er aan het systeem?

- Welke data gebruiken de modules van de applicatie en wat is de structuur van deze data?
- Welke functionaliteiten (triggers, stored procedures enz.) zitten er in de database van de huidige applicatie?
- Welke tabellen uit de huidige applicatie worden vaker gejoint dan individueel opgehaald?
- Wat zijn de krachten en zwakheden van relationele databases?
    - Welke van deze krachten en zwakheden hebben betrekking op de modules van de huidige applicatie?

Wat zijn de krachten en zwakheden van document databases?

- Welke van deze krachten en zwakheden hebben betrekking op de modules van de huidige applicatie?

2. Wat zijn relationele databases?

In dit hoofdstuk komt naar voren wat relationele databases zijn, en hoe hiermee gewerkt wordt. De informatie die hiervoor gebruikt is komt van bronnen (zie bronverwijzingen) en van kennis die de opdrachtnemer al had.

2.1 Relationele databases

Een database is een opslagplaats voor data waarin informatie opgeslagen kan worden op een manier dat deze informatie weer opgehaald kan worden.[5] De relationele database is een van de meest voorkomende databases.[6] In deze vorm van databases wordt data opgeslagen in tabellen, waarin relaties naar andere (of dezelfde) tabellen voorkomen. Het vermogen om aan elkaar gerelateerde data uit deze verschillende tabellen te halen is de basis voor relationele databases.[5]

In tegenstelling tot wat men denkt komt de naam relationeel niet voort uit de relaties tussen de tabellen, maar uit de term relatie uit de wiskunde. In de wiskunde definieert een relatie namelijk een bepaalde verzameling. In deze zin zijn alle gegevens die opgehaald worden uit de database relaties.

2.2 Datastructuur

Relationele databases slaan de data op in de vorm van tabellen. Een tabel bevat zowel rijen als kolommen. De kolommen staan voor de attributen, en de rijen bevatten de bijbehorende data. Alle data die op dezelfde rij staat hoort bij elkaar. Een voorbeeld hiervan is als volgt:

Tabel: Student

Id	Naam	Geboortedatum	Lengte_in_cm	Hobby
1	Xavyr Rademaker	15-03-1993	173	Voetbal
2	Job Haantjes	02-08-2000	154	Tennis

In de bovenstaande tabel is te zien dat Xavyr Rademaker geboren is op 15 maart 1993, voetbal als hobby heeft en 173 cm lang is. Job Haantjes is geboren op 02-08-2000, houdt van tennis en is 154 cm lang. In het voorbeeld is ook een kolom Id opgenomen. Deze kolom moet een unieke waarde bevatten (in dit geval een uniek nummer) waaraan die desbetreffende rij te herkennen is. Dit wordt ook wel de primaire sleutel (primairy key) genoemd.

Het is ook mogelijk in relationele databases om aan te geven dat er relaties tussen bepaalde tabellen bestaan. Als Xavyr Rademaker en Job Haantjes nu een docent zouden hebben, wordt er verwezen naar de primaire sleutel van hun docent, uit de tabel docent. Dit zou er als volgt uit zien:

Tabel: Student

Id	Naam	Geboortedatum	Lengte_in_cm	Hobby	Docent
1	Xavyr Rademaker	15-03-1993	173	Voetbal	4
2	Job Haantjes	02-08-2000	154	Tennis	2

Tabel: Docent

Id	Naam	In_dienst_sinds	Hobby
4	W.F.M. Vries	1-1-1995	Voetbal
2	I. Support	2-2-1990	Hardlopen

In de bovenstaande tabellen is te zien, dat de studenten een verwijzing naar de primaire sleutel van de docent bewaart (de met rood aangegeven vakken).

Wat opvalt in de vorige voorbeeld is dat zowel de studenten als de docenten hobby's hebben. Ook is het zo dat deze hobby's hetzelfde kan zijn en dus dubbel wordt opgeslagen. Om dit soort dubbele data te voorkomen is er binnen de wereld van relationele databases een techniek genaamd 'normaliseren'. Normaliseren is eigenlijk het verspreiden van data over meerdere tabellen die naar elkaar verwijzen, om zo spaarzamer met opslagruimte om te gaan en om te voorkomen dat data dubbel wordt opgeslagen (zoals de hobby: voetbal in het vorige voorbeeld).

In het geval van het voorbeeld zou het normaliseren van de tabellen tot het volgende leiden:

Tabel: Student

Id	Naam	Geboortedatum	Lengte_in_cm	Hobby	Docent
1	Xavyr Rademaker	15-03-1993	173	1	4
2	Job Haantjes	02-08-2000	154	2	2

Tabel: Docent

Id	Naam	In_dienst_sinds	Hobby
4	W.F.M. Vries	1-1-1995	1
2	I. Support	2-2-1990	3

Tabel: Hobby

Id	Naam
1	Voetbal
2	Tennis
3	Hardlopen

In het voorbeeld is te zien dat de namen van hobby's naar 1 tabel zijn gezet, waardoor er niet 2x de hobby voetbal terug is te vinden in de database. De tabellen docent en student verwijzen naar de primaire sleutels uit deze tabellen zodat nog steeds duidelijk is welke hobby's de studenten en de docenten hebben.

2.3 Regels en functies

Voor (bijna) alle relationele databases geld dat de ACID-eigenschappen gehanteerd worden. ACID staat voor: Atomair(Atomicity), Consistent(Consistency), Geïsoleerd(Isolation), Duurzaam(Durability) en zorgen ervoor dat database transacties correct aankomen in de database.
In het kort houden deze eigenschappen het volgende in[1]:

Atomair:
Alles wat in de transactie voorkomt wordt succesvol gecommit, of niks uit de transactie wordt gecommit.
Consistent:
De transactie maakt een nieuwe valide staat van de data, of de data wordt teruggezet naar de staat voor de transactie.
Isolatie:
Een transactie staat volledig los van een andere transactie. Transacties kunnen geen invloed op elkaar hebben.
Duurzaam:
De gecommitte data is zodanig opgeslagen dat als het systeem uitvalt en opnieuw opgestart moet worden, de data nog steeds beschikbaar is in de juiste staat.

Naast de ACID-eigenschappen, zijn foreign keys, indexen en extra regels die op de data worden gezet(bijvoorbeeld de NOT NULL constraint). Naast deze regels beschikken (veel) relationele databases over functies om de data te manipuleren, zoals optellen, aftrekken enzovoorts.

2.4 Query taal

Alle relationele databases worden bevraagd, en gewijzigd met behulp van SQL (Structured Query Language). Een simpele SQL-statement om data op te halen uit de eerdere student tabel ziet er als volgt uit:

```
SELECT * FROM Student;
```

De bovenstaande statement zegt dat ik alle attributen (aangegeven met *) van alle studenten wil ophalen. Zou ik alleen de studenten willen waar de naam Xavyr Rademaker is zou de query er als volgt uit zien:

```
SELECT * FROM Student WHERE Naam = "Xavyr Rademaker";
```

Het enige wat er in de bovenstaande query bij is gekomen is de WHERE-clausule. Met de WHERE-statement kan je aangeven dat de data uit die tabel wilt filteren.

Het is ook mogelijk om gegevens uit 2 tabellen te combineren, als je bijvoorbeeld wilt weten welke hobby's de studenten hebben. Dit ziet er als volgt uit:

```
SELECT * FROM Student s JOIN Hobby h ON s.Hobby = h.Id;
```

Doordat er in de bovenstaande query een JOIN is opgegeven, combineert(joined) de database de gegevens uit de 2 tabellen tot 1 resultaat (een tabel die informatie van beide bevat). Door na de JOIN een ON te zetten geef je aan welke kolom van student verwijst naar welke kolom van hobby.

Tot slot is het ook mogelijk om data uit 1 tabel op te halen, met filter eisen op een andere tabel waar de eerste tabel een relatie mee heeft. Als ik bijvoorbeeld alle voetballende studenten wil hebben, stel ik de query als volgt op:

```
SELECT * FROM Student WHERE Hobby = (SELECT Id FROM Hobby WHERE Naam = "Voetbal";);
```

In de bovenstaande query wordt als het ware een query binnen een query uitgevoerd. Met de laatste query beginnend, wordt de Id opgehaald van de hobby waarvan de naam 'Voetbal' is. In dit geval is dat 1. Vervolgens wordt het resultaat hiervan (1) ingevuld op de plek waar de query staat. Dit soort query's heten subselects.

Dit zorgt ervoor dat de query er als volgt uitziet: **SELECT** * **FROM** Student **WHERE** Hobby = 1;

De mogelijkheden en diepgang van de query's kan zo eindeloos doorgaan. Zo kunnen joins en subselects gecombineerd worden, en ook meerdere keren gebruikt worden in 1 query (er kunnen meerdere tabellen aan elkaar gekoppeld worden, er kunnen subselects in subselects voorkomen enzovoorts.).

Voor het opslaan, wijzigen en verwijderen van data in de tabellen zijn ook gestandaardiseerde manieren. Het opslaan gaat als volgt:

INSERT INTO Hobby(Id, Naam) **VALUES**(5, "Programmeren");

Nadat je hebt aangegeven aan welke tabel je iets wilt toevoegen, geef je aan in welke volgorde je welke kolommen gaat invullen. Als je alle kollommen gaat invullen in de volgorde waarop ze al zijn ingedeeld, hoeft dit niet. Na values geef je tussen haakjes aan welke waarde in de kolom moet, gescheiden met een ',.

Voor het wijzigen geld het volgende:

UPDATE Hobby **SET** Naam = "Ontwikkelen" **WHERE** Id=5;

De bovenstaande query wijzigt in de tabel hobby de naam naar 'Ontwikkelen' waar de Id 5 is.

DELETE FROM Hobby **WHERE** Id = 5;

De laatste query verwijderd de hobby met het Id 5.

3. Wat is NOSQL en welke vormen zijn er hiervan?

Om deze deelvraag te beantwoorden is de ISKA van 11-11-2015[7] bekeken.

3.1 NOSQL

NOSQL-databases zijn een vorm van databases die niet relationeel zijn, en ook niet perse gebruik maken van SQL als query taal. De term NOSQL staat niet zozeer voor een specifiek soort databases, zoals relationele databases altijd relationeel zijn, maar het is een verzamelnaam voor databases die niet relationeel zijn. Deze vorm van databases hebben een aantal karakteristieken, namelijk:

- Ze hebben geen relationeel model
    - Alle data wordt als geheel, en onafhankelijk stuk data opgeslagen zonder foreign keys
- Het is horizontaal schaalbaar
    - Omdat de data zo onafhankelijk is, kan de database makkelijk verspreid worden over meerdere servers
- Er is geen vast schema, de structuur van de data is dus flexibel
    - De databases hebben geen vooraf vastgestelde schema's wat ervoor zorgt dat je structuur van de ene op de andere dag kan veranderen, indien dit nodig is.

3.1.1 ACID vs. BASE

In het vorige hoofdstuk is uitgelegd waar ACID voor staat. BASE is als het ware de tegenhanger hiervan. Het belangrijkse punt uit BASE is de E(eventual consistency), hier zal dan ook meer aandacht aan besteedt worden. BASE staat voor:

- Basically Available
    - De database is beschikbaar wanneer deze nodig is
- Soft state
- Eventual consistency
    - Replica's van de database hoeven niet meteen bijgewerkt te worden als er een update gedaan wordt op een andere replica.
    - Deze updates kunnen doorgevoerd worden op een ander, wellicht rustiger, moment

Er is geen harde definitie van Basically Available en Soft state. De essentie van BASE ligt dan ook(zoals eerder vermeldt) op de eventual consistency.

Een voorbeeld bij base is:

Gebruiker A geeft een like aan een foto van gebruiker B. Deze like wordt verwerkt in database A. Een paar minuten later bekijkt gebruiker B dezelfde foto, welke wordt opgehaald van database Replica A, maar ziet nog geen nieuwe likes op deze foto (like van gebruiker A is nog niet verwerkt op de replica). 10 minuten later verschijnt de like bij gebruiker B op zijn scherm.

Door de soft state was het mogelijk dat gebruiker B 10 minuten later de like binnen kreeg, ook al werd er op dat moment geen update uitgevoerd op database Replica A. de eventual consistency zorgde ervoor dat de data uiteindelijk consistent gemaakt werd.

Het grote verschil tussen ACID en BASE zit hem in deze consistentie. Op het moment dat een database ACID zou hanteren, zou gebruiker B de like meteen binnenkrijgen. Dit komt doordat alle replica's meteen consistent gemaakt worden. De keerzijde hiervan is, dat de delen van database A en database B waar de like invloed op heeft een kort moment niet beschikbaar zouden zijn, omdat deze consistent gemaakt zou worden. Op het moment dat replica B niet beschikbaar zou zijn, zou dit dus betekenen dat de delen in database A pas beschikbaar zijn als B consistent gemaakt is, oftewel als B weer live staat en bijgewerkt is.

3.1.2 Aggregaten

Aggregaten in NOSQL zijn iets anders dan aggregaten in relationele databases. In NOSQL wordt een los stuk samengestelde data(bijvoorbeeld een document) als aggregaat gezien.

NOSQL-databases worden over het algemeen niet genormaliseerd. Het idee erachter is dat de data wordt opgeslagen zoals het binnenkomt, oftewel zoals de applicatie ermee werkt.

Voordeel hiervan is dat de database precies hetzelfde is ingericht als de applicatie. Het is niet zo dat, zoals bij relationele databases wel het geval is, de data in de database op een bepaalde manier is opgeslagen (genormaliseerd naar aparte tabellen per model als het ware) en de data in de applicatie in elkaar wordt bijgehouden (in JSON: { naam: "Xavyr Rademaker", docent: { naam: "W.F.M. Vries" } }). Omdat het op dezelfde manier is opgeslagen, is er geen laag meer nodig voor het vertalen van data uit de database naar de data gebruikt in de applicatie.
Het nadeel hiervan is wel dat er bepaalde data dubbel in de database kan voorkomen. Als docent "W.F.M. Vries" namelijk 2-maal zou voorkomen bij studenten, zou het er als volgt uitzien:

{ naam: "Xavyr Rademaker", docent: { naam: "W.F.M. Vries" } }, { naam: "Job Haantjes", docent: { naam: "W.F.M. Vries" } }).

3.1.3 CAP-theorem

Het verschil tussen ACID en BASE is wellicht beter te begrijpen door het CAP-theorem te begrijpen. Het CAP-theorem zegt dat er 3 aspecten zijn, namelijk:

- Consistency
  - Alle clients hebben altijd dezelfde data
- Availability
  - Alle clients kunnen altijd lezen en schrijven
- Partition tolerance
  - Het systeem werkt goed, ook als er fysieke netwerk partities zijn/uitvallen.

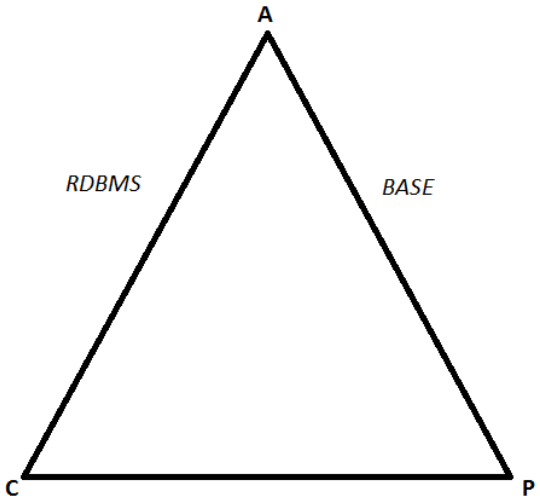
Wat het CAP-theorem over deze aspecten zegt is dat een systeem maar aan 2 van de 3 aspecten kan voldoen. Reden hiervoor is als volgt:

Op het moment dat een systeem altijd consistent is en altijd beschikbaar is kan hij niet partitie tolerant zijn, omdat het uitvallen van een partitie ervoor zorgt dat het systeem niet meer beschikbaar is (het systeem kan de partities niet consistent maken, dus moet het of inconsistent doorgaan of niet beschikbaar zijn tot alles consistent is).

Op het moment dat een systeem altijd beschikbaar is en partitie tolerant is kan hij niet consistent zijn, omdat het systeem beschikbaar moet blijven als er een partitie wegvalt. Oftewel, hij kan niet wachten om de partities consistent te maken.

Op het moment dat een systeem partitie tolerant wilt zijn en consistent is, kan hij niet altijd beschikbaar zijn, omdat het systeem moet wachten tot alle partities consistent zijn, zelfs als er een partitie uitvalt (er wordt dus niet verder gewerkt tot deze partitie weer up is, en consistent gemaakt is).

De relationele databases voldoen aan de aspecten consistentie en beschikbaarheid. Hierdoor kan de database niet zomaar op meerdere partities verdeeld worden. De NOSQL-databases (BASE) zitten voornamelijk tussen de beschikbaarheid en de partitie tolerantie. Omdat zij consistentie als het ware hebben opgegeven voor partitie tolerantie, is het mogelijk om deze databases horizontaal te schalen (verspreiden over meerdere servers). Hoe relationele databases(ACID) en BASE ten opzichte van het CAP theorem staan is op de volgende afbeelding te zien:



3.1.4 Voor –en nadelen NOSQL

Een aantal voordelen die gelden voor (bijna) alle NOSQL-databases zijn:

- Het is horizontaal schaalbaar vanwege de onafhankelijkheden tussen data (geen foreign keys)
- Het coderen gaat makkelijker, het schema wordt aangegeven in de structuur van de applicatie, in plaats van dat de structuur van de database de structuur van de applicatie aangeeft.*
- Er is geen schema, dit resulteert in meer flexibiliteit.

De nadelen die erbij komen zijn:

- Over het algemeen wordt ACID niet gehanteerd, wat resulteert in minder consistentie in de data, en kan zelfs ten koste gaan van de integriteit van de data
- Het is nog niet zo standaard/veel gebruikt als relationele databases, dus niet alle tools kunnen ermee samenwerken
- Er is geen schema, dit kan resulteren in een rommelige database
- De databases zijn simpeler van aard, wat ervoor zorgt dat verantwoordelijkheden naar de applicatie verplaatst moeten worden.
    - Met simpeler wordt bedoeld dat, de database minder tot geen complexe constraints en schema's hebben. De data wordt opgeslagen zoals het

binnenkomt(niet normaliseren/verdeeld over tabellen) en wordt op dezelfde manier opgehaald.

*Het codeert makkelijker, omdat je bij een wijziging van de structuur van data niet de database hoeft bij te werken. Als Persoon A bijvoorbeeld een attribuut erbij krijgt, of er juist een attribuut weggaat, hoeft je niet te bedenken hoe deze verwerkt gaat worden in de database. Dit komt doordat het schema van de database gedefinieerd wordt vanuit de applicatie(oftewel als je data wegschrijft met een attribuut meer of minder, wordt deze data zodanig weggeschreven).

Ook is het zo dat als je data wilt ophalen uit een relationele database, je de resultaten van de query moet omzetten naar de objecten die gebruikt worden in de applicatie.

Als er geen gebruik gemaakt wordt van een ORM, zou dit ophalen betekenen dat de attributen van een object 1 voor 1 gevuld moeten worden met data uit de juiste kolommen uit de database(uit de resultaten van de query). Als er wel gebruik gemaakt zou worden van een ORM, zou dit betekenen dat de mapping files aangepast moeten worden bij het toevoegen/verwijderen van een attribuut. Wat een kleine, maar een extra handeling is.

3.2 Vormen van NOSQL

Binnen de NOSQL-wereld zijn er een aantal categorieën. Deze categorieën zijn:

- Key-value
- Document database
- Column-based database
- Graph database

3.2.1 Key-value store

Een key-value store is, zoals de naam zegt, een structuur waar data wordt opgeslagen aan de hand van een key. Deze key kan elke waarde hebben die de ontwikkelaar opgeeft bij het opslaan van de data. Over het algemeen kan de key dus ook een string zijn(bijvoorbeeld een naam + datum aanmaak enz.), zolang deze maar uniek is en je de data ermee kan terugvinden(als je de key zelf niet kan onthouden of zodanig formuleert dat deze niet meer terug te vinden is door de applicatie, kan de bijbehorende data niet meer teruggehaald worden).

Het is in deze vorm van databases belangrijk dat de key onthouden wordt, omdat dit de enige manier is om de data weer terug te halen. In tegenstelling tot relationele databases, is het hier dus niet zo dat er op elk attribuut gezocht kan worden.

In de value kan alles opgeslagen worden. Van simpele strings tot volledige objecten, of zelfs hele sessies. Omdat het zo is dat alles in de value kan, is het belangrijk dat de applicatie die de data ophaalt uit de database weet hoe deze data opgebouwd is. Als ik bijvoorbeeld een object opsla, en daarbij de attributen scheidt met een ; en de waardes van de attributen aangeef met : dan zal de value er als volgt uitzien:

Key	Value
1	Naam: Xavyr Rademaker; Geboortedatum: 15-03-1993; Lengte_in_cm: 173

De applicatie die dit uitleest zal moeten weten hoe de structuur van de value is, om deze terug te kunnen converteren naar een object zoals de applicatie het kent.

Krachten:
De krachten van een key-value database zitten hem in het opslaan en ophalen van kleine hoeveelheden data. Dit komt doordat de database alleen de keys kent, en verder ook geen indexen neerzet. Op het moment dat er grote hoeveelheden data, met complexe structuren in de database opgeslagen wordt, betekent dit dat het altijd als geheel opgehaald kan worden, wat niet altijd wenselijk is. Verder is het zo dat alles in de value kan, ongeacht de structuur en grootte.

3.2.2 Document store

Document databases zijn uitgebreide vormen van key-value stores. Er is nog steeds sprake van een key, maar de opgeslagen data (value in key-value) is gestructureerd. Over de structuur wordt in de volgende deelvraag meer verteld.

Het grootste verschil met key-value stores, naast de structuur, is dat er ook op elk attribuut geïndexeerd kan worden, en er dus ook op elk attribuut gezocht kan worden.

3.2.3 Column family store

Column family stores zijn ook uitgebreide key-values stores. Dit is ook de enige vorm van NOSQL-databases waar er deels een schema van tevoren wordt gedefinieerd. Een column family store werkt als volgt:

Er is een row key (dit is altijd de PK), meerdere column families (Soort overkoepelende kolom waarin er meerdere kolommen met bij elkaar horende data zitten), en kolommen. Dit ziet er als volgt uit:

Persoon	
Naam	**geboortedatum**
Xavyr Rademaker	15-3-1993

Docent	
Naam	**Vak**
Leraar	Biologie

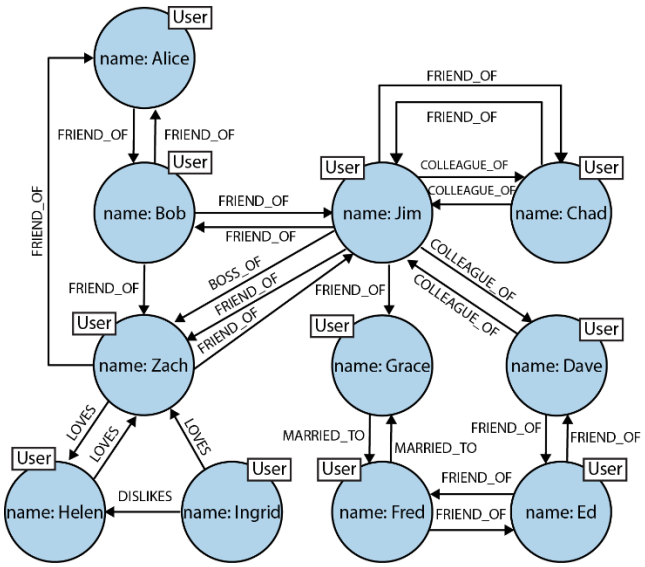
In het bovenstaande voorbeeld zijn de column families: Persoon en Docent. De kolommen zijn: Naam, geboortedatum, naam en vak. En de waardes zijn Xavyr Rademaker, 15-3-1993, leraar en biologie. We gaan er in dit voorbeeld van uit dat de row key voor de values 1 is, omdat het allemaal op dezelfde row zit.

Over het algemeen geld voor Column family stores, dat zowel de column families als de rows te verspreiden zijn over verschillende servers. Dit maakt horizontale schaalbaarheid mogelijk. De hoeveelheid columns die in een row kunnen staan zijn eindeloos. Ook is het zo dat in een cell (de waarde op row x in column y) ook meerdere versies van een waarde bewaard kunnen worden.

3.2.4 Graph database

Een graph database is te herkennen aan nodes en relaties. Dit type database gaat om de gegevens van een relatie, niet zozeer een entiteit zelf. Een voorbeeld hiervan is: wie heeft het artikel geschreven, wie heeft het gelezen, waar gaat het over enzovoorts. Er kan ook gedacht worden aan social media bij dit soort databases (vrienden met, vrienden van, vrienden van vrienden van vrienden van).

Alle entiteiten in de database zijn nodes, deze kunnen ook properties bevatten. Tussen deze nodes liggen relaties, welke ook properties kunnen krijgen. Het idee van Graph databases is, het bevragen van deze relaties. De relaties worden dan dus ook als data opgeslagen. In de volgende afbeelding is te zien hoe de opslag van een graph database eruitziet:



4. Wat zijn document oriented databases?

In dit hoofdstuk wordt besproken wat document databases zijn. De informatie die hiervoor gebruikt is komt van bronnen (zie bronverwijzingen) en van kennis die de opdrachtnemer al had.

4.1 Document oriented databases

Document Oriented Databases zijn een vorm van NOSQL-databases. Zoals de naam al aangeeft, slaan Document Oriented Databases hun data op in de vorm van een document. Het formaat hiervan kan zijn JSON, XML, BSON (binaire JSON), YAML en nog veel meer. Dit formaat staat wel vast per implementatie. 1 implementatie zal dus niet zowel JSON als XML gebruiken om documenten op te slaan.

4.2 Datastructuur

Zoals eerder gezegd kan de opgeslagen data worden bewaard in verschillende formaten. Hier is dus geen standaardformaat voor. Wel is het zo, dat een bepaalde implementatie van deze vorm van databases 1 formaat gebruikt. Het zal dus niet zo zijn dat een specifieke database afwisselt van bestandsformaat. In de onderstaande voorbeelden heb ik gekozen voor het JSON-formaat, omdat dit naar mijn mening het eenvoudigst is. Het principe is hetzelfde bij de andere formaten, alleen de manier van noteren is anders.

Als dezelfde studenten als in het voorbeeld van relationele databases uitgewerkt zouden worden naar de document oriented manier van opslaan, zou dit er als volgt uitzien:

```
{
    {
      Id: 1,
      Naam: “Xavyr Rademaker”,
      Geboortedatum: 15-03-1993,
      Lengte_in_cm: 173,
      Hobby: “Voetbal”
    },
    {
        Id: 2,
      Naam: “Job Haantjes”,
      Geboortedatum: 02-08-2000,
      Lengte_in_cm: 154,
      Hobby: “Tennis”
    }
}
```

Voor iedereen die ooit wat met Javascript gedaan heeft zal dit er bekend uitzien. Voor iedereen die dit nog nooit heeft gedaan ziet de tabelvorm er waarschijnlijk prettiger uit.

Het grijs gemaakte gebied in het voorbeeld is 1 document. Vergeleken met de tabel uit het vorige hoofdstuk, zou je kunnen zeggen dat een document een equivalent is van de rij.

De met geel aangegeven accolades zijn de grenzen van de verzameling. Een verzameling is in principe een soort tabel in relationele termen. Het houdt bepaalde documenten bij. Let er wel op dat er in tegenstelling tot bij de relationele databases, de documenten in een verzameling niet allemaal hetzelfde formaat hoeven te hebben. De rijen in relationele databases moeten dit wel.
Als er nu bij Xavyr Rademaker een attribuut toegevoegd zou worden die Job niet heeft, zal het systeem er niet moeilijk om doen. Een voorbeeld hiervan is:

```
{
    {
       Id: 1,
       Naam: "Xavyr Rademaker",
       Geboortedatum: 15-03-1993,
       Lengte_in_cm: 173,
       Hobby: "Voetbal",
       Talenten: ["Is heel snel aan de bal", "Call of Duty: Black Ops 3"]
    },
    {
        Id: 2,
       Naam: "Job Haantjes",
       Geboortedatum: 02-08-2000,
       Lengte_in_cm: 154,
       Hobby: "Tennis"
    }
}
```

Zoals de met grijs aangegeven rij laat zien, hoeven documenten in een verzameling niet dezelfde structuur te hanteren en kunnen er ook arrays met data opgeslagen worden. Documenten binnen een collectie kunnen wel dezelfde structuur hanteren, dit moet dan afgedwongen worden op applicatieniveau. Er zijn dus geen constraints op de structuur en invoer van data.

Verder is het best-practice om vergelijkbare documenten te groeperen in 1 verzameling. Zo is er in het voorbeeld uit hoofdstuk 2 een verzameling met studenten, en een verzameling met docenten.

Om relaties te weergeven in document databases zijn er 2 opties: embedding en references. Een reference werkt in principe hetzelfde als bij relationele databases, er wordt een verwijzing naar de andere document bijgehouden. Embedding daarentegen houdt in dat het gehele document binnen een ander document wordt bijgehouden. Als de docenten nu toegevoegd zouden worden zou het er als volgt uitzien in beide situaties:

Id	Naam	Geboortedatum	Lengte_in_cm	Hobby	Docent
1	Xavyr Rademaker	15-03-1993	173	Voetbal	4
2	Job Haantjes	02-08-2000	154	Tennis	2

Tabel: Docent

Id	Naam	In_dienst_sinds	Hobby
4	W.F.M. Vries	1-1-1995	Voetbal
2	I. Support	2-2-1990	Hardlopen

References:[4]

Als er gekozen is voor references ziet het voorbeeld van de studenten en docenten er als volgt uit:

Verzameling: Student

```
{
    {
      Id: 1,
      Naam: "Xavyr Rademaker",
      Geboortedatum: 15-03-1993,
      Lengte_in_cm: 173,
      Hobby: "Voetbal",
      Talenten: ["Is heel snel aan de bal", "Call of Duty: Black Ops 3"],
      Docent: 4
    },
    {
        Id: 2,
      Naam: "Job Haantjes",
      Geboortedatum: 02-08-2000,
      Lengte_in_cm: 154,
      Hobby: "Tennis" ,
    }
}
```

Verzameling: Docent

```
{
      {
            Id: 4,
            Naam: "W.F.M. Vries ",
            In_dienst_sinds: 1-1-1995,
            Hobby: "Voetbal"
      },
      {
            Id: 2,
            Naam: "I. Support",
            In_dienst_sinds: 2-2-1990,
            Hobby: "Hardlopen",
            Studenten: [2]
      }
}
```

In het bovenstaande voorbeeld is te zien dat student Xavyr Rademaker docent W.F.M. Vries als docent heeft. Dit wordt aangegeven door middel van het Id (gemarkeerd met groen).

Om aan te tonen dat de verwijzing ook op de andere kant kan staan is deze voor student Job Haantjes bijgehouden bij zijn docent. Bij docent I. Support wordt namelijk een array bijgehouden met verwijzingen naar de id's van zijn studenten (met grijs aangegeven). In dit geval is er maar 1 student.

Een voordeel van het gebruik van references is, dat de schrijfsnelheid van het document waar de references in staan hoger is. Echter gaat dit wel ten koste van de leessnelheid aangezien er meerdere documenten opgehaald moeten worden om alle informatie te verkrijgen.[2]

Een nadeel van het gebruik van references is ook dat, in tegenstelling tot relationele databases, de integriteit in de meeste document databases niet gewaarborgd wordt. Het is in het bovenstaande voorbeeld mogelijk om docent: W.F.M. Vries te verwijderen zonder dat de reference in student verwijderd wordt.

Embedding:[4]

In het geval van embedding ziet het voorbeeld van studenten en docenten er als volgt uit:

Verzameling: Student

```
{
    {
      Id: 1,
      Naam: "Xavyr Rademaker",
      Geboortedatum: 15-03-1993,
      Lengte_in_cm: 173,
      Hobby: "Voetbal",
      Talenten: ["Is heel snel aan de bal", "Call of Duty: Black Ops 3"],
      Docent: {
                  Id: 4,
                  Naam: "W.F.M. Vries ",
                  In_dienst_sinds: 1-1-1995,
                  Hobby: "Voetbal"
        }
    },
    {
    ,
    }
}
```

Verzameling: Docent

```
 {
    {
            Id: 2,
            Naam: "I. Support",
            In_dienst_sinds: 2-2-1990,
            Hobby: "Hardlopen",
            Studenten: [{
            Id: 2,
            Naam: "Job Haantjes",
            Geboortedatum: 02-08-2000,
            Lengte_in_cm: 154,
                  Hobby: "Tennis"
        }]
      }
}
```

In het bovenstaande voorbeeld wordt embedding weergeven. In de verzameling student is te zien dat Xavyr Rademaker W.F.M. Vries als docent heeft. Deze docent staat niet in zijn eigen verzameling, maar staat als volledig document binnen het document van Xavyr Rademaker.

In het grijs is bij de docent I. Support aangegeven dat het ook andersom kan. Dus docenten kunnen ook een array bijhouden van volledige documents (van in dit geval studenten).

Een voordeel van embedding is de verhoogde leessnelheid ten opzichte van references, omdat alle data in 1 document staat. Het betekent echter wel dat documents snel groot kunnen worden, wat ten koste kan gaan van de schrijfsnelheid van het document.[2]

Een ander nadeel wat kan voorkomen is dat gegevens meerdere malen worden opgeslagen (als dezelfde docent in meerdere situaties voorkomt bijvoorbeeld)

References of embedding:
In de volgende tabel[2] wordt weergeven in welke situatie er voor welke opslagstructuur gekozen zou moeten worden:

Embedding is beter voor	**References zijn beter voor**
Kleine subdocumenten	Grote subdocumenten
Data dat niet regelmatig wijzigt	Data wat vaak wijzigt
Documenten die niet snel groeien*	Documenten die snel groeien*
Data wat vaak als geheel opgehaald moet worden	Data wat vaak niet getoond word in de resultaten
Hoog leessnelheid	Hoog schrijfsnelheid

*met snel groeien wordt bedoeld: documenten waar veel attributen/veel documenten in embed worden. Denk bijvoorbeeld aan de volgende situaties:

- Als bol.com per product bijhoudt hoeveel mensen het kopen, en deze personen embedded opslaat in de producten, zullen de producten snel heel veel embedded documents bevatten, wat de product document dus groot maakt. Oftwel het document groeit snel.

4.3 Regels en functies

Elke document database heeft zo zijn eigen functies. Sommige implementaties hebben de mogelijkheid om functies in de database te maken (vergelijkbaar met stored procedures), sommige hebben rekenfuncties enzovoorts. Dit is dus volledig afhankelijk van de database zelf.

Qua regels komen er een aantal punten terug (of juist niet) in de meeste document databases, namelijk:

- De meeste document databases houden geen rekening met de integriteit van de data
- Niet alle document databases houden zich aan de ACID-eigenschappen[3]

4.4 Query taal

Onder de NOSQL-databases is geen gestandaardiseerde query taal vastgelegd. Daar document databases onder deze categorie van databases valt geldt dit ook hiervoor. Elke implementatie van document databases hebben zo ook hun eigen query taal.

Hoewel de syntax van query'en niet gestandaardiseerd is, zijn er wel een aantal mogelijkheden kenmerkend voor het query'en in document databases. Over het algemeen geldt namelijk, dat er op alle attributen van een document gezocht kan worden. Ook is het in een aantal databases mogelijk alleen een deel van het document op te halen (bijvoorbeeld alleen de naam van de studenten) en bestaat er in sommige databases de mogelijkheid om indexen toe te voegen op veel gezochte attributen.

Aangezien relaties niet expliciet aangegeven kunnen worden (een document kan referenties bevatten naar documenten die niet bestaan) is er over het algemeen geen mogelijkheid tot joinen. Om de data te combineren moet er een 2^{e} query uitgevoerd worden.

5. Welke modules zijn er in de huidige applicatie en wat doen deze?

In de beantwoording van deze deelvraag komt het volgende naar boven: welke modules zijn er in de huidige applicatie, en wat doen deze modules. De informatie gebruikt om deze vraag te beantwoorden komt van de website van de Vries WFM. Verder zijn er vragen gesteld aan de opdrachtgever over onduidelijkheden.

5.1 Modules R&R-web

Binnen de R&R-webapplicatie zijn er een aantal modules, namelijk:

- Normeringen
  - Kan extern gedaan worden
  - Valt eigenlijk onder prognose
- Prognose
- Rooster
- Realisatie
- Uitbetaling
- Rapportages
  - Zijn meer een ondersteunende module.

De flow van de huidige applicatie verloopt ook in de bovenstaande volgorde. De modules zijn verantwoordelijk voor het volgende:

5.1.1 Normeringen

De module normeringen is verantwoordelijk voor het in kaart brengen van alle werkzaamheden binnen de winkel. Hieronder valt onder andere:

- Het scannen van artikelen bij de kassa
- Het vullen van de schappen
- Het schoonmaken van de vloer

Deze werkzaamheden worden voorzien van een normtijd en een normtarief. De uitkomsten van de normering wordt gebruikt als input voor het maken van de planning en worden alleen aangepast als de klant daar behoefte aan heeft.

De normtijden en normtarieven worden in de winkel gemeten. Dit kan gedaan worden door medewerkers van de Vries, maar kan ook gedaan worden door een externe partij. Als het bedrijf al normeringen heeft, kunnen deze ook in het systeem verwerkt worden.

De normtijden worden uitgezet in kwartieren.

5.1.2 Prognose

In de prognose module wordt er een voorspelling gedaan omtrent de omzet, klanten bezoek, benodigde werkkracht enz. voor een geselecteerde week. De voorspelling wordt gedaan op basis van (geselecteerde)realisaties uit het verleden. Hierbij wordt ook rekening gehouden met de door de retailer ingevoerde geplande weekomzet.

Het resultaat van de berekening geeft aan, hoeveel tijd er voor welke werkzaamheden ingeruimd moet worden. Deze tijdsindeling wordt gekoppeld aan het loonkostenbudget om de kosten van werkzaamheden in kaart te brengen.

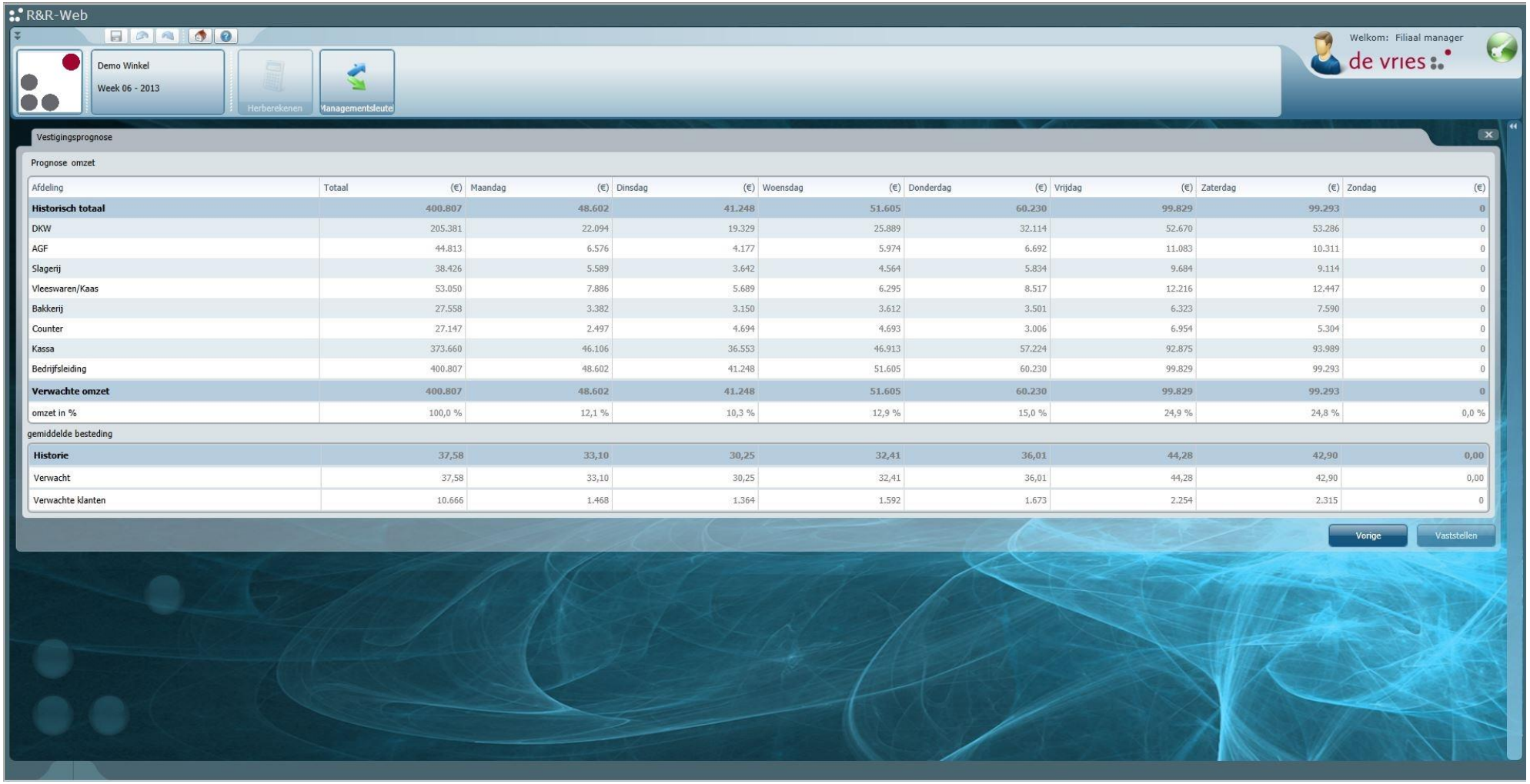
Het loonkostenbudget is gebaseerd op: normeringsuren * gemiddelde kosten per werknemer voor de uitvoerende taak.

Door middel van een koppeling met het kassasysteem in de winkel, kan de winkel de klantbezoekpatronen analyseren. Door middel van dit patroon, weet de applicatie precies wanneer er meer of minder personeel ingezet moet worden op welke afdelingen. Als er bijvoorbeeld een piek aan klanten is aan het eind van de dag, zullen er ook meer kassamedewerkers moeten zijn.

Het kassasysteem is overigens geen onderdeel van het systeem. De data van de kassa's kan aan het eind van de dag ingevoerd/geïmporteerd worden in het systeem.

De prognose wordt geeft uit deze gegevens dus een inschatting voor de komende week.

De afbeelding op de volgende pagina laat een prognose zien. De cijfers uit de prognose zijn gebaseerd op al afgeronde weken. Op basis van deze weken is er een gemiddelde berekent.


R&R-Web
Demo Winkel
Week 06 - 2013
Herberekenen
Managementsleute
Welkom: Filiaal manager
de vries
Vestigingsprognose
Prognose omzet
Afdeling | Totaal | (€) | Maandag | (€) | Dinsdag | (€) | Woensdag | (€) | Donderdag | (€) | Vrijdag | (€) | Zaterdag | (€) | Zondag | (€)
Historisch totaal | 400.807 | 48.602 | 41.248 | 51.605 | 60.230 | 99.829 | 99.293 | 0
DKW | 205.381 | 22.094 | 19.329 | 25.889 | 32.114 | 52.670 | 53.286 | 0
AGF | 44.813 | 6.576 | 4.177 | 5.974 | 6.692 | 11.083 | 10.311 | 0
Slagerij | 38.426 | 5.589 | 3.642 | 4.564 | 5.834 | 9.684 | 9.114 | 0
Vleeswaren/Kaas | 53.050 | 7.886 | 5.689 | 6.295 | 8.517 | 12.216 | 12.447 | 0
Bakkerij | 27.558 | 3.382 | 3.150 | 3.612 | 3.501 | 6.323 | 7.590 | 0
Counter | 27.147 | 2.497 | 4.694 | 4.693 | 3.006 | 6.954 | 5.304 | 0
Kassa | 373.660 | 46.106 | 36.553 | 46.913 | 57.224 | 92.875 | 93.989 | 0
Bedrijfsleiding | 400.807 | 48.602 | 41.248 | 51.605 | 60.230 | 99.829 | 99.293 | 0
Verwachte omzet | 400.807 | 48.602 | 41.248 | 51.605 | 60.230 | 99.829 | 99.293 | 0
omzet in % | 100,0 % | 12,1 % | 10,3 % | 12,9 % | 15,0 % | 24,9 % | 24,8 % | 0,0 %
gemiddelde besteding
Historie | 37,58 | 33,10 | 30,25 | 32,41 | 36,01 | 44,28 | 42,90 | 0,00
Verwacht | 37,58 | 33,10 | 30,25 | 32,41 | 36,01 | 44,28 | 42,90 | 0,00
Verwachte klanten | 10.666 | 1.468 | 1.364 | 1.592 | 1.673 | 2.254 | 2.315 | 0
Vorige
Vaststellen

5.1.3 Rooster

Als er een prognose is gesteld, komt de fase waarin het rooster gemaakt moet worden. Hiervoor dient een manager 1-malig een standaard rooster in te voeren in het systeem. Dit standaardrooster zal elke week gebruikt worden, maar de manager heeft de vrijheid om zelf wijzigingen aan te brengen per week.

Per week wordt er in de rooster module gekeken naar: welke medewerkers zijn er beschikbaar, welke kwalificaties hebben deze medewerkers en welk van deze medewerkers kunnen ingezet worden bij welke werkzaamheden. Bij het inzetten van de medewerkers wordt dus rekening gehouden met de beschikbaarheid en kwalificaties van de medewerkers en naar de verwachte werkzaamheden uit de prognose. Als uit de prognose blijkt dat er op een bepaald moment 3 kassamedewerkers nodig zijn, wordt er dus ook aangegeven dat er 3 medewerkers ingepland moeten worden.

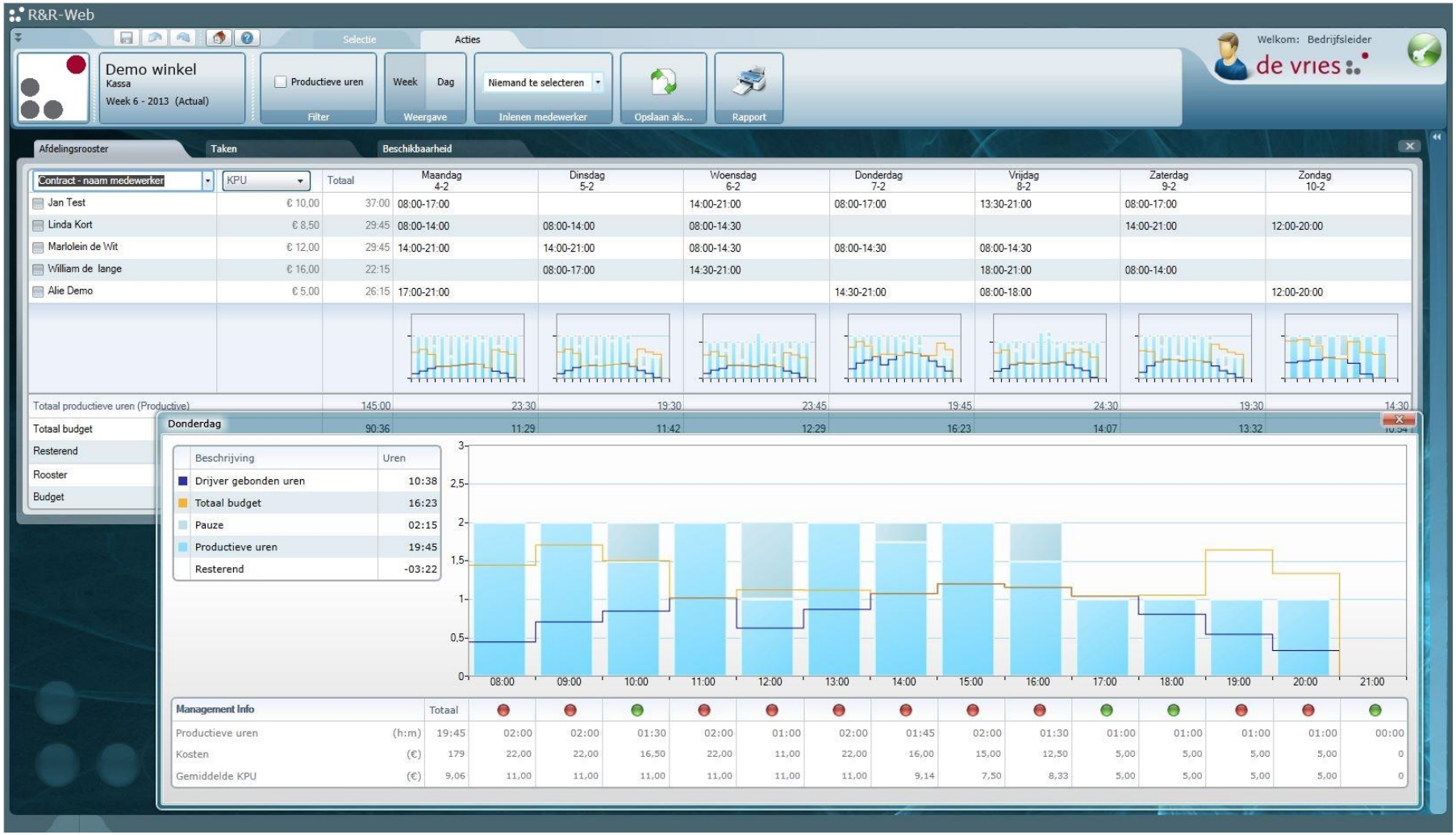
Ook wordt er gekeken naar wanneer welke medewerkers het best op welke momenten ingezet kunnen worden. Op drukke momenten zullen er dus meer kassières nodig zijn en op rustige momenten kunnen de schoonmakers beter hun werk doen.

Aangezien er bij de prognose rekening wordt gehouden met het loonkostenbudget, komt dit ook naar voren in de roostermodule. Ook wordt er rekening gehouden met de CAO en de wetgevingen.

Hier wordt rekening mee gehouden door middel van controles van voorwaarde die staan aangegeven in het CAO en in de wetgevingen. Mocht iets niet aan deze voorwaarde voldoen worden er waarschuwingen aan de gebruiker gegeven. Deze waarschuwingen kunnen informatief zijn, maar ook blokkerend (er mag niet verder gegaan worden tot het is aangepast).

De wetgevingen en Cao’s kunnen worden aangegeven in de instellingen. Mochten de eisen van het Cao niet in de instellingen verwerkt worden, dan wordt de code van deze instelling aangepast.

Ook kunnen er medewerkers van andere afdelingen en winkels worden ‘geleend’.
Op de volgende pagina een screenshot van een rooster te zien.


R&R-Web
Selectie
Acties
Welkom: Bedrijfsleider
de vries
Demo winkel
Kassa
Week 6 - 2013 (Actual)
Productieve uren
Filter
Week
Dag
Weergave
Niemand te selecteren
Inlenen medewerker
Opslaan als...
Rapport
Afdelingsrooster
Taken
Beschikbaarheid
Contract - naam medewerker
KPU
Totaal
Maandag 4-2
Dinsdag 5-2
Woensdag 6-2
Donderdag 7-2
Vrijdag 8-2
Zaterdag 9-2
Zondag 10-2
Jan Test € 10,00 37:00 08:00-17:00 14:00-21:00 08:00-17:00 13:30-21:00 08:00-17:00
Linda Kort € 8,50 29:45 08:00-14:00 08:00-14:00 08:00-14:30 14:00-21:00 12:00-20:00
Marlolein de Wit € 12,00 29:45 14:00-21:00 14:00-21:00 08:00-14:30 08:00-14:30 08:00-14:30
William de lange € 16,00 22:15 08:00-17:00 14:30-21:00 18:00-21:00 08:00-14:00
Alie Demo € 5,00 26:15 17:00-21:00 14:30-21:00 08:00-18:00 12:00-20:00
Totaal productieve uren (Productive) 145:00 23:30 19:30 23:45 19:45 24:30 19:30 14:30
Totaal budget 90:36 11:29 11:42 12:29 16:23 14:07 13:32
Resterend
Rooster
Budget
Donderdag
Beschrijving
Uren
Drijver gebonden uren 10:38
Totaal budget 16:23
Pauze 02:15
Productieve uren 19:45
Resterend -03:22
08:00 09:00 10:00 11:00 12:00 13:00 14:00 15:00 16:00 17:00 18:00 19:00 20:00 21:00
Management Info
Totaal
Productieve uren (h:m) 19:45 02:00 02:00 01:30 02:00 01:00 02:00 01:45 02:00 01:30 01:00 01:00 01:00 01:00 00:00
Kosten (€) 179 22,00 22,00 16,50 22,00 11,00 22,00 16,00 15,00 12,50 5,00 5,00 5,00 5,00 0
Gemiddelde KPU (€) 9,06 11,00 11,00 11,00 11,00 11,00 11,00 9,14 7,50 8,33 5,00 5,00 5,00 5,00 0

5.1.4 Realisatie

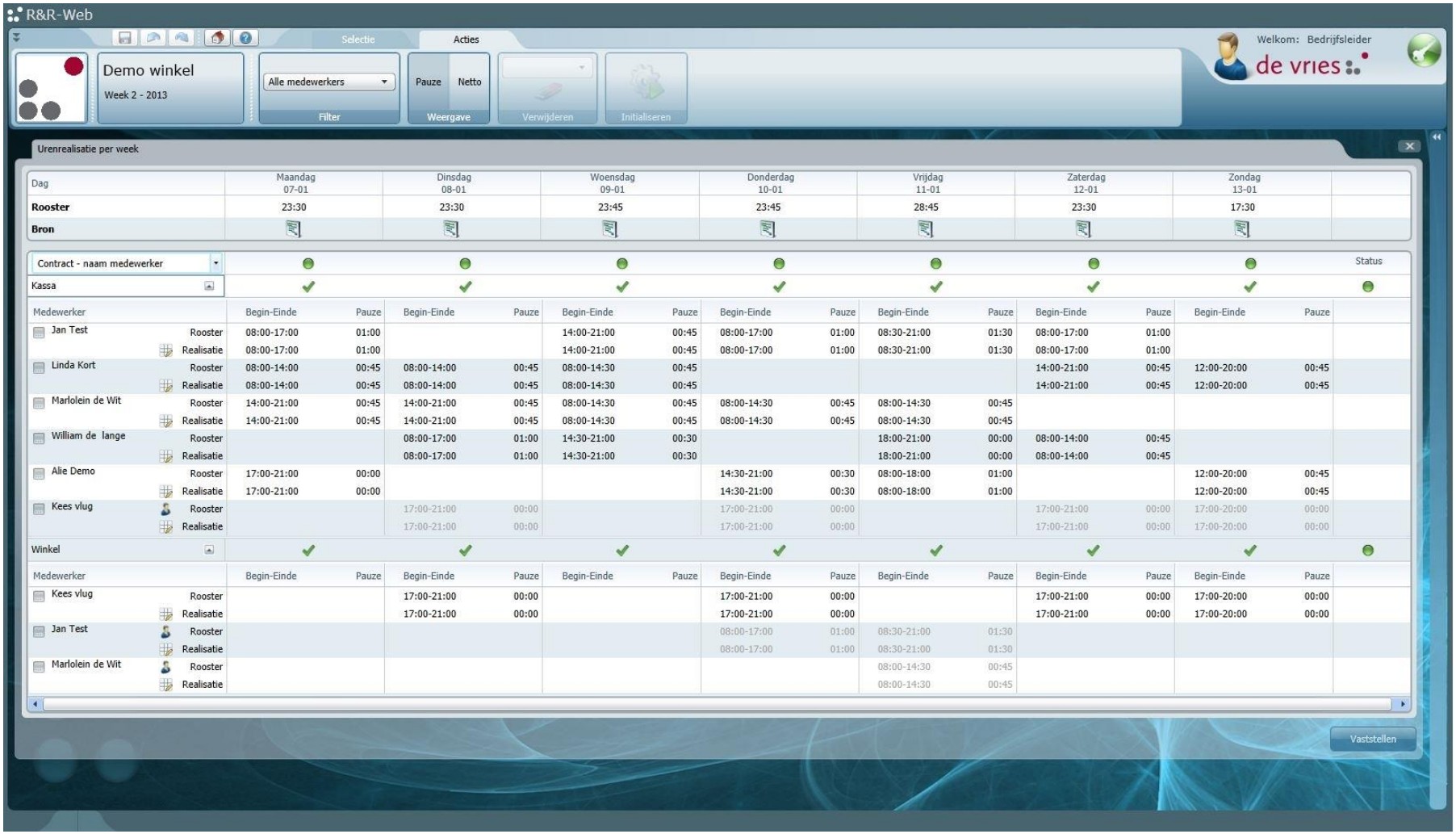
De realisatie is de module verantwoordelijk voor de daadwerkelijk gemaakte uren. In de prognose en het rooster zijn bepaalde planningen gemaakt omtrent de werkzaamheden. In de realisatie komt naar voren wanneer medewerkers aanwezig geweest zijn, en op welke afdeling zij zijn geweest.

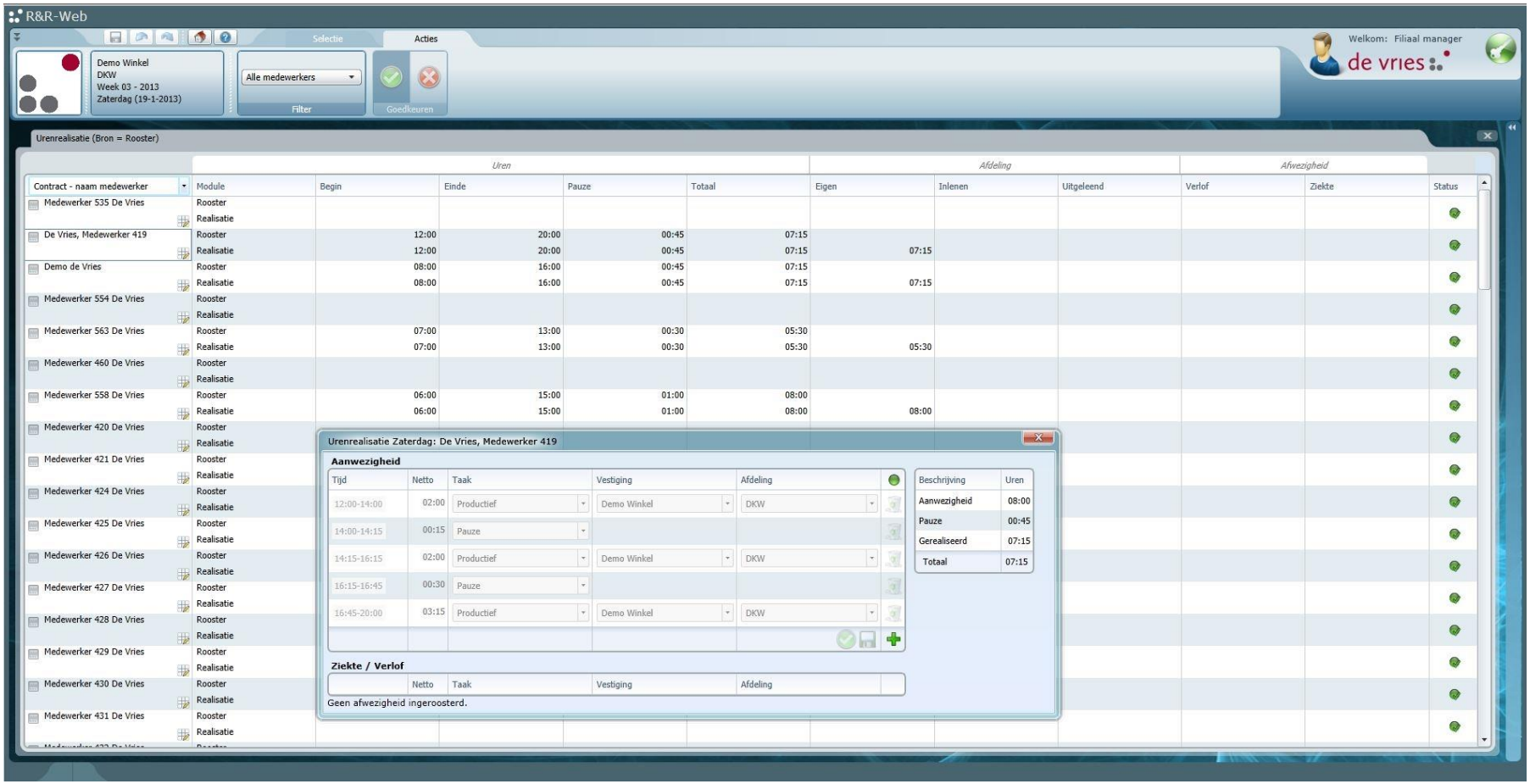
De gemaakte werkuren, dus de start en eindtijd, per ingeroosterde werknemer, per dag kunnen handmatig door de manager ingevoerd worden. Daarnaast biedt deze module een integratie met het R&R-time systeem (een kloksysteem gebaseerd op in en uitklokken van medewerkers met pasjes) om de werktijden te verkrijgen.

Zowel bij het handmatig invoeren als bij de integratie met R&R-time moet de manager zelf controleren of deze tijden kloppen. Dit is belangrijk omdat de uren uit de realisatie mee worden genomen naar de uitbetaling.

Het is overigens niet altijd zo dat alle geklokte tijden uitbetaald worden. Het komt namelijk voor dat er in Cao's staat dat er per kwartier wordt afgerond. In dat geval kan het dus voorkomen dat iemand 8.55 binnenkomt en 12.13 weggaat, en betaald krijgt van 9 tot 12.

De afbeeldingen op de volgende pagina's laten zien hoe zo een realisatie eruit ziet. In dit voorbeeld is de realisatie echter wel perfect overeenkomend met de geroosterde tijden, dit kan ook afwijken en aangepast worden.


R&R-Web
Selectie
Acties
Welkom: Bedrijfsleider
de vries
Demo winkel
Week 2 - 2013
Alle medewerkers
Filter
Pauze
Netto
Weergave
Verwijderen
Initialiseren
Urenrealisatie per week
Dag
Maandag 07-01
Dinsdag 08-01
Woensdag 09-01
Donderdag 10-01
Vrijdag 11-01
Zaterdag 12-01
Zondag 13-01
Rooster
23:30
23:30
23:45
23:45
28:45
23:30
17:30
Bron
Contract - naam medewerker
Status
Kassa
Medewerker
Begin-Einde
Pauze
Jan Test
Rooster
Realisatie
08:00-17:00 01:00
14:00-21:00 00:45
08:00-17:00 01:00
08:30-21:00 01:30
08:00-17:00 01:00
Linda Kort
08:00-14:00 00:45
08:00-14:30 00:45
14:00-21:00 00:45
12:00-20:00 00:45
Marlolein de Wit
14:00-21:00 00:45
08:00-14:30 00:45
William de lange
08:00-17:00 01:00
14:30-21:00 00:30
18:00-21:00 00:00
08:00-14:00 00:45
Alie Demo
17:00-21:00 00:00
14:30-21:00 00:30
08:00-18:00 01:00
Kees vlug
17:00-21:00 00:00
Winkel
Vaststellen


R&R-Web
Selectie
Acties
Demo Winkel
DKW
Week 03 - 2013
Zaterdag (19-1-2013)
Alle medewerkers
Filter
Goedkeuren
Welkom: Filiaal manager
de vries
Urenrealisatie (Bron = Rooster)
Uren
Afdeling
Afwezigheid
Contract - naam medewerker
Module
Begin
Einde
Pauze
Totaal
Eigen
Inlenen
Uitgeleend
Verlof
Ziekte
Status
Medewerker 535 De Vries
Rooster
Realisatie
De Vries, Medewerker 419
Rooster 12:00 20:00 00:45 07:15
Realisatie 12:00 20:00 00:45 07:15 07:15
Demo de Vries
Rooster 08:00 16:00 00:45 07:15
Realisatie 08:00 16:00 00:45 07:15 07:15
Medewerker 554 De Vries
Medewerker 563 De Vries
Rooster 07:00 13:00 00:30 05:30
Realisatie 07:00 13:00 00:30 05:30 05:30
Medewerker 460 De Vries
Medewerker 558 De Vries
Rooster 06:00 15:00 01:00 08:00
Realisatie 06:00 15:00 01:00 08:00 08:00
Medewerker 420 De Vries
Medewerker 421 De Vries
Medewerker 424 De Vries
Medewerker 425 De Vries
Medewerker 426 De Vries
Medewerker 427 De Vries
Medewerker 428 De Vries
Medewerker 429 De Vries
Medewerker 430 De Vries
Medewerker 431 De Vries
Urenrealisatie Zaterdag: De Vries, Medewerker 419
Aanwezigheid
Tijd
Netto
Taak
Vestiging
Afdeling
12:00-14:00 02:00 Productief Demo Winkel DKW
14:00-14:15 00:15 Pauze
14:15-16:15 02:00 Productief Demo Winkel DKW
16:15-16:45 00:30 Pauze
16:45-20:00 03:15 Productief Demo Winkel DKW
Beschrijving
Uren
Aanwezigheid 08:00
Pauze 00:45
Gerealiseerd 07:15
Totaal 07:15
Ziekte / Verlof
Netto
Taak
Vestiging
Afdeling
Geen afwezigheid ingeroosterd.

5.1.5 Uitbetaling

De uitbetalingsmodule is verantwoordelijk voor het exporteren van alle loongegevens. Hierbij worden de cijfers uit de realisatie gebruikt, in combinatie met het CAO en de arbeidstijdenwet inclusief de toeslagen. In de uitbetalingsmodule worden dus ook regels als: 'de uitbetalingen worden afgerond op kwartieren' toegepast. Deze gegevens worden geëxporteerd naar het payroll systeem van de winkel. De uitbetalingen zelf worden niet gedaan door R&R-web.

5.1.6 Rapportages

De rapportage module is verantwoordelijk voor het genereren van rapporten op basis van data uit het systeem. Deze module kan onder andere de volgende rapportages genereren:

- Realisatie per medewerker, wekelijks en dagelijks
- Urenverschillenrapportage
- Realisatie vs. planning per medewerker
- Afwezigheid
- Ziekteverzuim per afdeling
- Verlofsaldo
- Kloktijden uit het tijdsregistratiesysteem
- Jaaroverzicht per medewerker

6. Wat zijn de kern functionaliteiten van de modules van de applicatie?

Om deze vraag te beantwoorden heb ik gekeken naar de functionaliteiten(user stories/use cases) die te vinden waren op de portal en op augurk. Van deze functionaliteiten heb ik de kernfunctionaliteiten eruit gehaald. Deze heb ik vervolgens gevalideerd bij de opdrachtgever.

Verder zijn er ook een aantal flowcharts gemaakt om de flow van de functionaliteiten in kaart te brengen. Deze flowcharts zijn de vinden in bijlage G: Flowcharts.

De use cases behorende bij de kern-functionaliteiten zijn te vinden in bijlage C: use case scenario's. Verder zijn alleen de functionaliteiten van de 4 grote modules uitgewerkt, daar de kern van de applicatie zich hierbinnen bevind. Van deze modules vormen de volgende functionaliteiten de kern:

6.1 Prognose

De volgende kernfunctionaliteiten zijn te vinden in de prognose module.

UC nr	Use case	functionaliteiten	Funct. nr
UC009	Als winkel manager wil ik een voorspelling voor de winkel initialiseren		
		Beheer openingstijden	F047
		Initialiseer voorspelling voor de winkel	F123
		Beheer 'main driver' op winkel niveau(per week/dag)	F044
		Keur voorlopige voorspelling van de winkel goed	F058
UC010	Als afdelingsmanager wil ik de voorspellingen per afdeling beheren	Beheer 'main driver' per normcluster	F078
		Verdeel de 'main driver' van de afdeling over dagen	F079
		Keur de voorspelling van een afdeling goed	F120
		Beheer werktijden	F086
UC011	Als winkelmanager wil ik winkel voorspelling reviewen	Laat een winkel voorspelling terugkeren(revert)	F063
UC077	Als afdelingsmanager wil ik de voorspelling van mijn afdeling initialiseren	Beheer taken	F053

*main driver = instelbaar, maar is vaak de omzet.

6.2 Rooster

De volgende kernfunctionaliteiten zijn te vinden in de rooster module.

UC nr	Use case	functionaliteiten	Funct. nr
UC012	Als winkelmanager wil ik het winkel rooster (her)initialiseren	Initialiseer rooster	F060
UC013	Als afdelingsmanager wil ik het afdelingsrooster beheren	Beheer geroosterde activiteiten van medewerkers per dag	F034
		Review details van activiteiten van medewerkers	F013
		Inlenen medewerker	F029
		Toon management informatie	F018
		Toon uurloon/verlofsaldo/kwalificaties	F016
		Sla op als template afdelingsrooster	F064
		Keur afdelingsrooster goed	F005
UC014	Als winkelmanager wil ik een winkel rooster reviewen	Toon waarschuwingen per afdeling	F021
		Keur periode rooster van afdeling af	F009
		Rond winkelrooster af (inclusief afronden periode rooster van afdeling)	F028
UC016	Als gebruiker wil ik de template afdelingsrooster beheren	Beheer template voor activiteiten medewerkers	F095
		Toon berichten per medewerker /week/dag	F019
UC017	Als gebruiker wil ik standaard beschikbaarheid beheren	Beheer template voor beschikbaarheid medewerkers	F101

6.3 Realisatie

De volgende kernfunctionaliteiten zijn te vinden in de realisatie module.

UC nr	Use case	functionaliteiten	Funct. nr
UC018	Als winkelmanager wil ik de realisatie uren per dag beheren	Beheer realisatie van medewerkers	F039
		Af/goedkeuren realisatie van medewerkers per dag	F002
		Af/goedkeuren realisatie van medewerkers per week	F001

UC019	Als winkelmanager wil ik handmatig de realisatie per dag beheren	Beheer realisatie per dag	F118
UC061	Initialiseer winkel realisatie	Initialiseer winkel realisatie(1 of meer dagen)	F030
UC062	Als winkelmanager wil ik de winkel realisatie(per week) beheren	Handmatig initialeren uren realiseren(1 of meer dagen)	F087
		Goedkeuren realisatie uren van alle medewerkers van alle geselecteerde afdelingen	F006
UC075	Verwerk kloktijden	Importeren kloktijden(geen actor, gebeurd automatisch op de achtergrond)	F092

6.4 Uitbetaling

De volgende kernfunctionaliteiten zijn te vinden in de uitbetaling module.

UC nr	Use case	functionaliteiten	Funct. nr
UC022	Beheer betalingen	Beheer winkel betalingen op feestdagen	F011
		Beheer betalingen van medewerkers per week	F037
		Beheer betalingen van medewerkers per week: beheer activiteiten van medewerkers	F049
		Beheer betalingen van medewerkers per week: beheer aanwezigheidspatroon van de medewerkers	F031
		Beheer betalingen van medewerkers per week: Reset betaling van medewerker(realisatie terugdraaien)	F061
		Beheer betalingen van medewerkers per week: Goed/afkeuren betaling medewerker	F004
		Toon vertrek balans	F017
UC074	Initialiseer winkel betaling	Initialiseer betaling	F088

7. Welke niet functionele eisen zijn er aan het systeem?

Aan het huidige systeem gelden geen harde niet functionele eisen. Zo is er geen eis die zegt: het systeem moet roosters ophalen binnen 1 seconden. In plaats daarvan wordt er ingespeeld op commentaar en feedback van de klanten. Als een klant bijvoorbeeld aangeeft dat iets niet snel genoeg gaat, wordt hiernaar gekeken.

Wel zijn er binnen het bedrijf een aantal processen rondom dit systeem, namelijk:

- Hotfixen kunnen live gezet worden zonder het systeem down te halen
- Tijdens onderhoudt mag de server wel down zijn.
- Er worden dagelijks back-ups gemaakt van de database
- Sommige users mogen alleen lezen in de database, sommige lezen en schrijven en slechts enkele mogen ook de structuur van de database aanpassen
    - Dit geld voor het deployment team
- Klanten mogen geen data van elkaar zien
    - Momenteel heeft elke klant een eigen database
- Back-ups van de live omgeving worden op de acceptatie omgeving gezet
    - Database dump live omgeving, vervolgens op acceptatie omgeving inladen
    - Op de test omgeving wordt de data geanonimiseerd opgeslagen, zodat er geen klantgegevens gelezen kunnen worden
- Er worden stored procedures gebruikt voor de query's
    - Er werd eerst een ORM gebruikt, maar hier was er te weinig controle op de manier van query'en. De ORM zelf genereerde te grote query's, vandaar dat er nu stored procedures gebruikt worden
    - Zo worden SQL-injections ook tegengegaan
- De database is goed te monitoren, er zijn error logs enz.
- Als er back-ups op de live omgeving worden ingelezen, duurt het maximaal 3uur voor het systeem weer live staat
- Er is een schaduwomgeving van de live omgeving, voor als de live omgeving down gaat
- Er kan op performance gestuurd worden door middel van indexen enz. (op database niveau).
- Er wordt gequeried op meerdere attributen, niet alleen op de PK

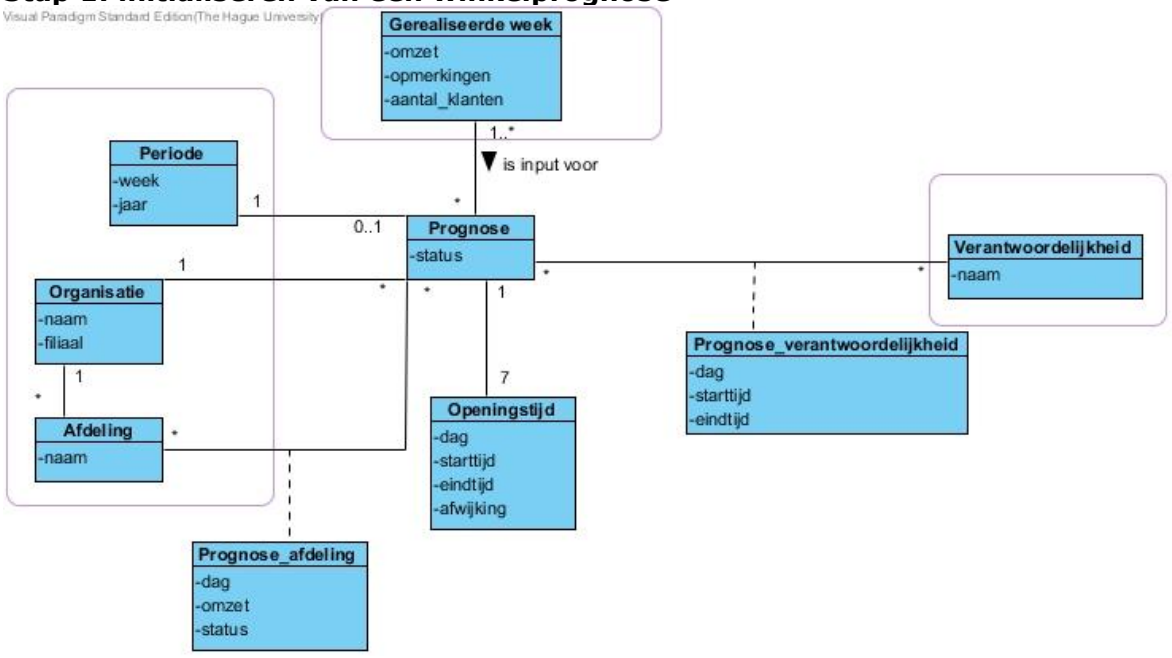
8. Welke data gebruiken de modules en hoe zit de structuur van deze data in elkaar?

De tabellen in dit hoofdstuk zijn slechts contextueel, om een beeld te krijgen bij de data die gebruikt wordt bij de flow uit hoofdstuk 6. De stappen zijn dan ook afkomstig van de stappen uit de flow. De klassen waar een vierkant omheen staat komen van buiten de module af.

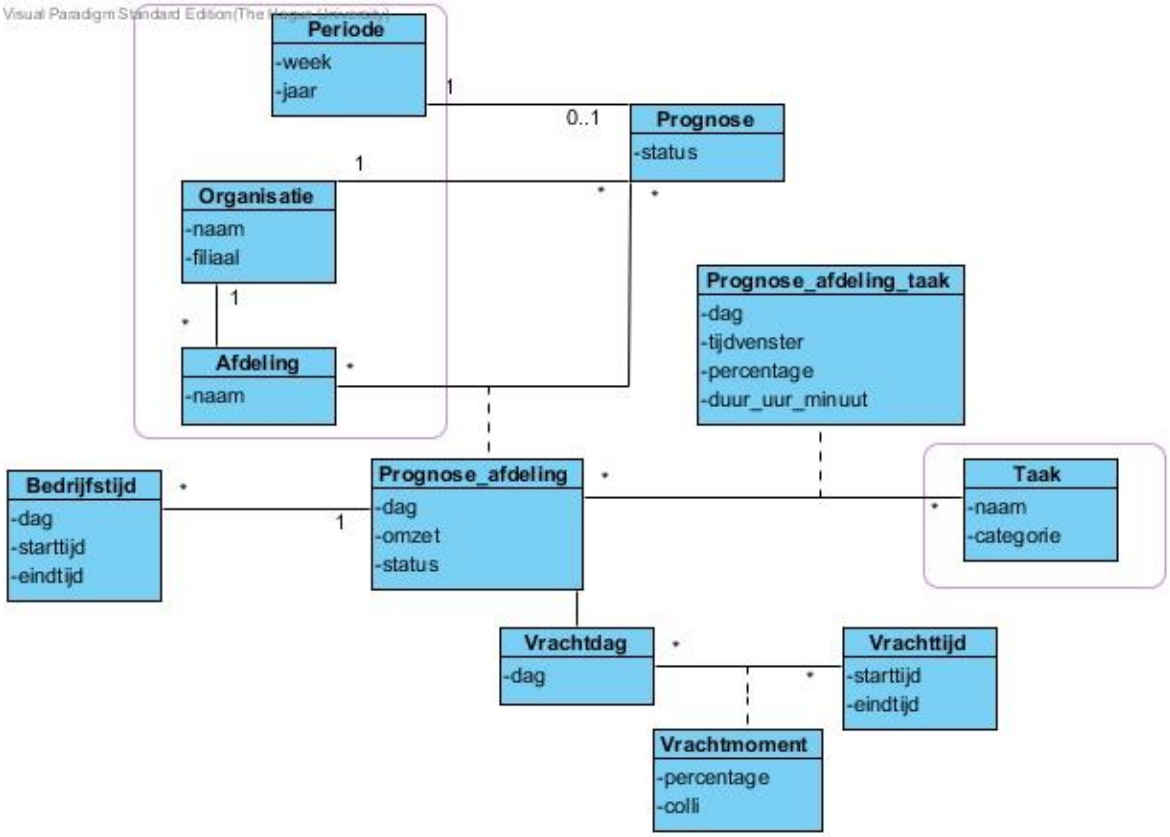
Voor de technische(gedetailleerde) diagrammen zie bijlage F.

8.1 Prognose

Stap 1: initialiseren van een winkelprognose



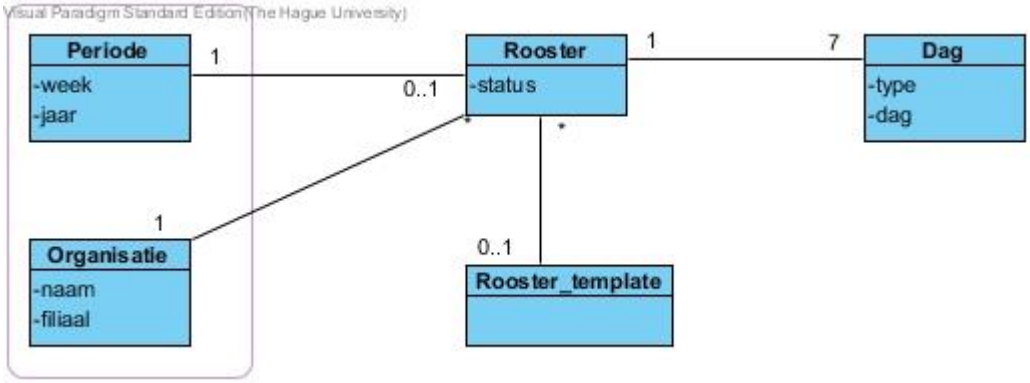
Stap 2: beheren/vaststellen van de afdelingsprognose



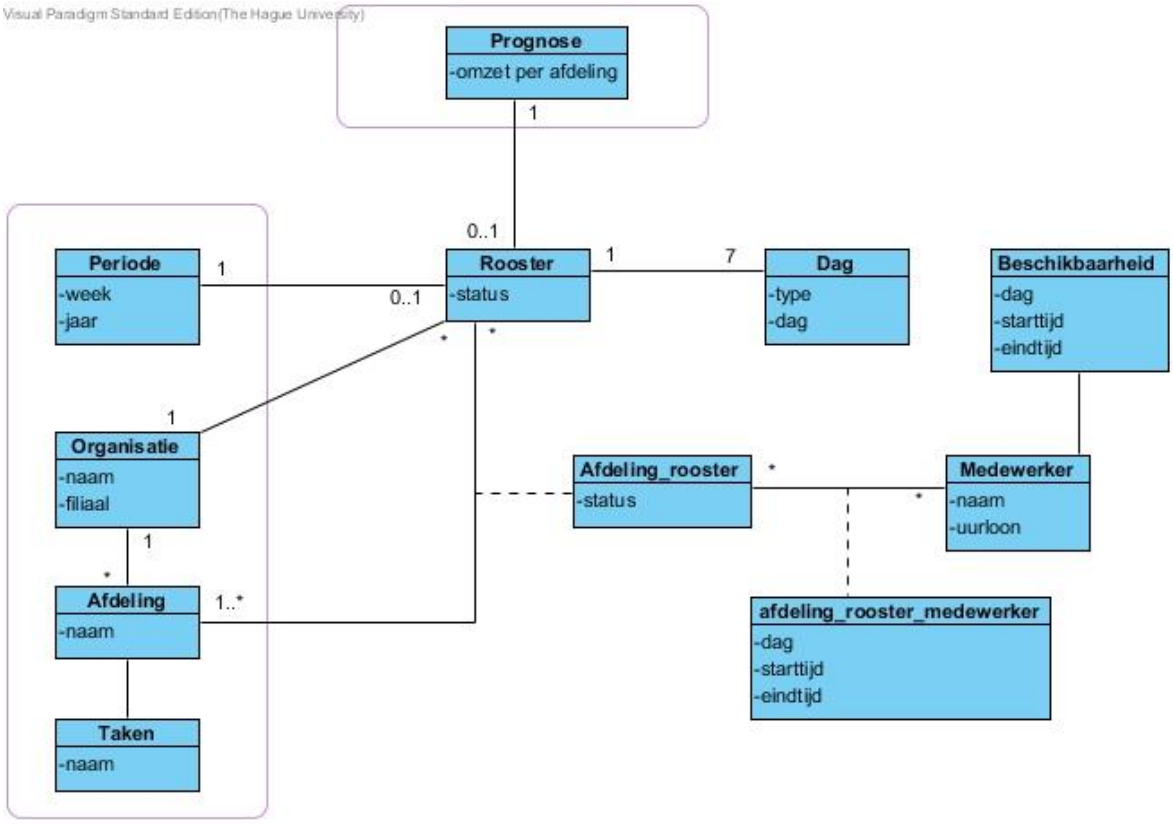
Stap 3: Vaststellen prognose
Hier komt geen nieuwe data bij kijken.

8.2 Rooster

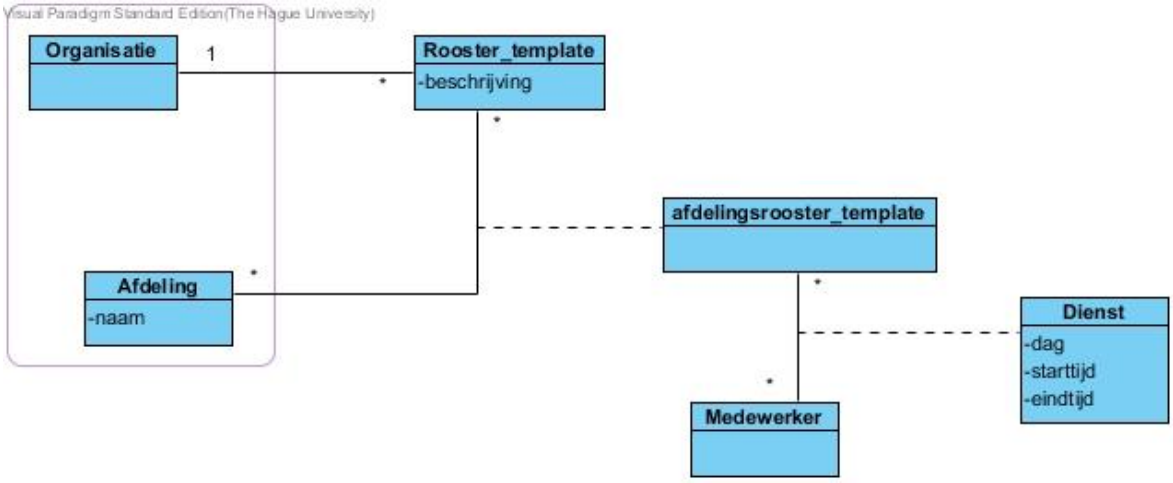
Stap 1: initialiseren winkelrooster



Stap 2, 3 en beheer beschikbaarheid: beheren/vaststellen afdelingsrooster, vaststellen winkelrooster en Beheren standaard beschikbaarheden

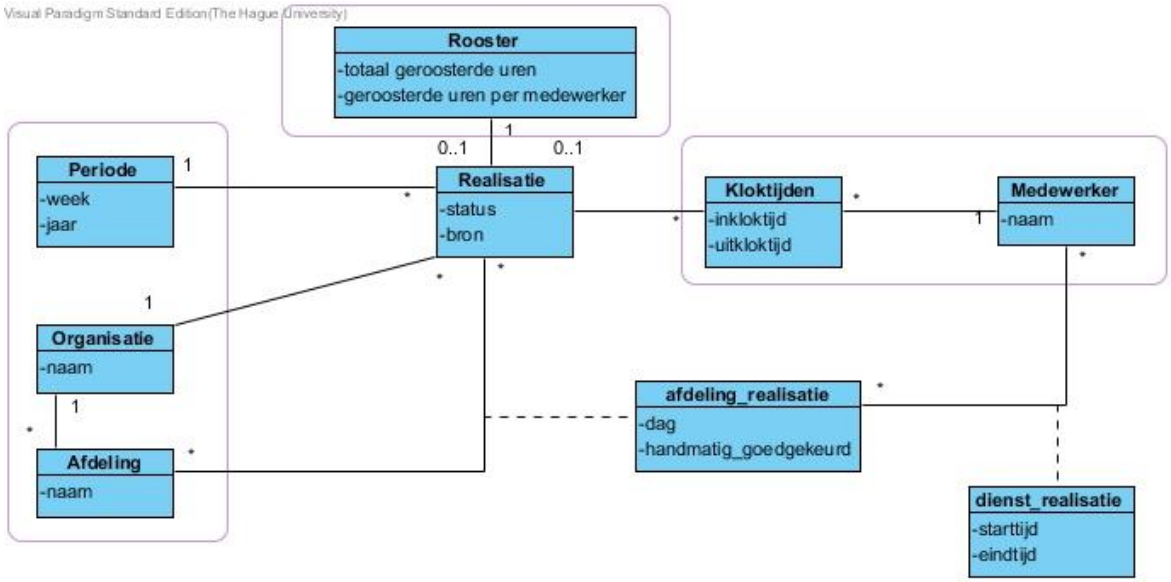


Beheer winkel/afdelingsrooster:

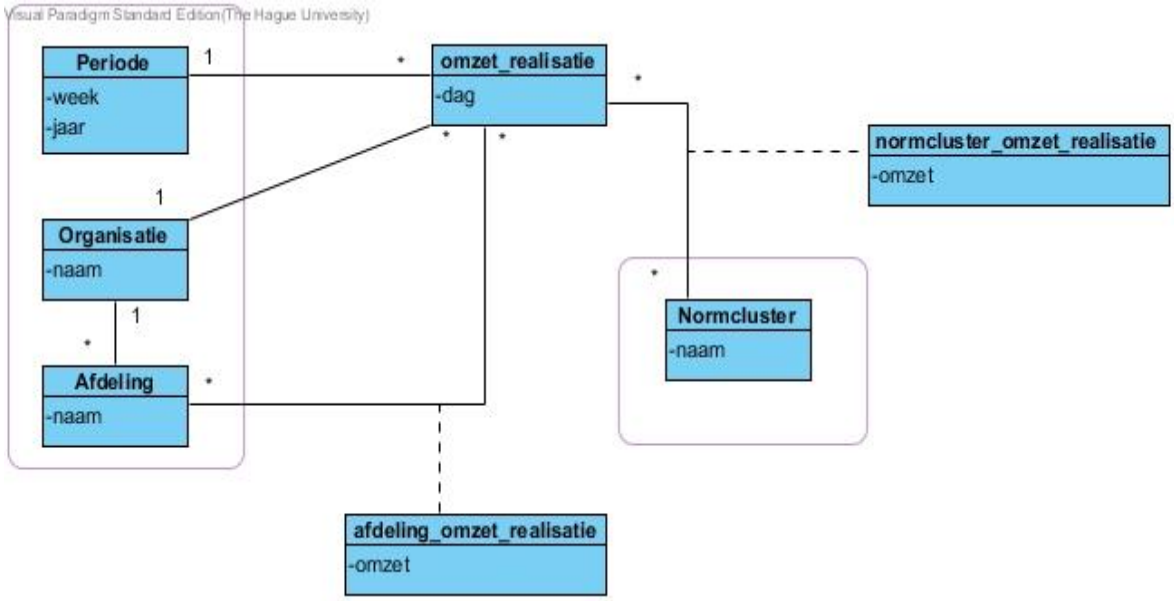


8.3 Realisatie

Handeling 1: invullen gerealiseerde uren per week:

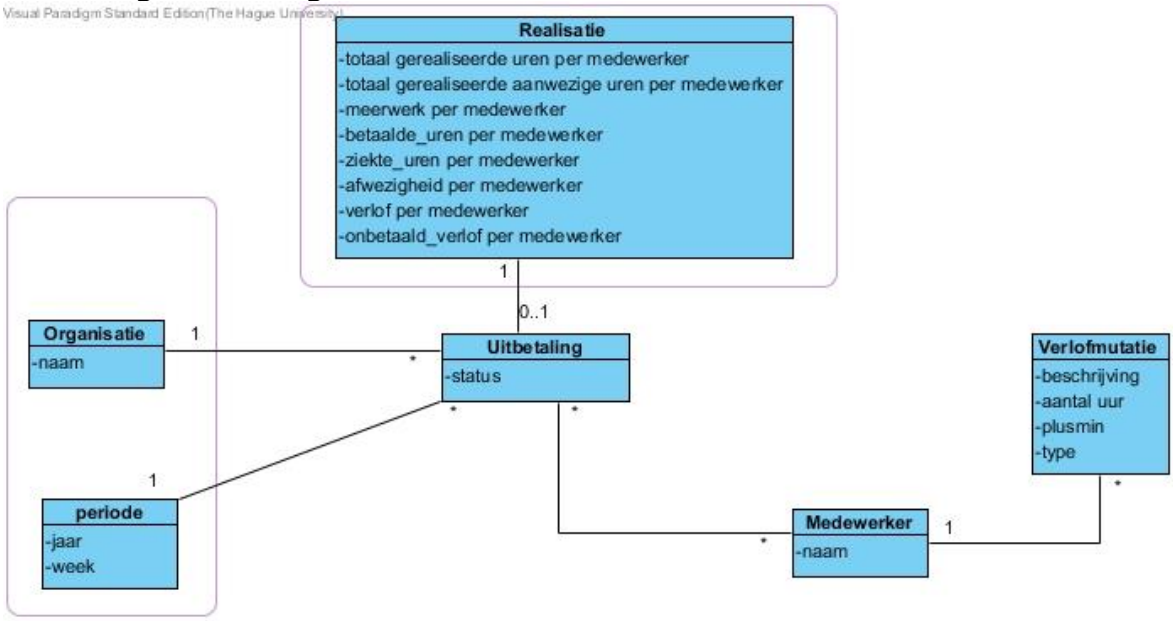


Handeling 2: gerealiseerde omzet/klanten



8.4 Uitbetaling

Handeling 1: betalingen beheren



Handeling 2:
Het exporteren van data is, zoals de naam zegt, enkel het exporteren van data die ergens binnen de applicatie gegenereerd/ingevuld is. Hier komt dus geen nieuwe data bij kijken.

9. Welke functionaliteiten zitten er in de database van de huidige applicatie?

De volgende functionaliteiten zijn gevonden in de database van de huidige applicatie:

- Triggers
- Custom functions
- Stored procedures
- Indexen
- Constraints

9.1 Stored procedures

Aangezien de applicatie geen gebruik maakt van een ORM, is er besloten om alle query's in stored procedures op te slaan. Op deze manier wilt het bedrijf onder andere SQL-injections tegengaan. Een aantal van deze query's zullen naar voren komen in het volgende hoofdstuk.

9.2 Custom functions

In de modules: realisatie en prognose zijn een aantal custom functions geschreven op database niveau.

In de realisatie zitten de volgende functionaliteiten:

- GetLastActiveOrganizationLink
  - Haalt de laatste actieve organizationlink op en geeft de PK hiervan terug
- GetLastInitializationDate
  - Haalt de laatste initialisatiedatum van een realisatie op en retourneert deze.
- GetRealizationDayStatus
  - Haalt de status van de realisatie van een medewerker op een dag op. Retourneert de status van het akkoord gegeven door de manager (Yes of No)

In de prognose module zitten de volgende functionaliteiten:

- GetLaborBudgetCosts
  - Haalt het budget voor arbeidskosten op, op basis van de winkel, de periode en de arbeidskosten van de winkel.
- GetGrossProfitPercentage
  - Haalt het brutowinstpercentage op van de prognose op basis van de periode en de winkel, en retourneert deze.
- fn_GetMainDriver
  - Haalt de primaire driver van een winkel op, op basis van de winkel, en retourneert deze.
- fn_GetMainDriverValue
  - Haalt de waarde van de primaire driver (bijvoorbeeld de omzet) op, op basis van de winkel en de periode, en retourneert deze.

9.3 Triggers

Er zijn een aantal tabellen waar triggers in verwerkt zitten.

Een aantal van deze triggers staan op de volgende tabellen:

- Forecast.Budget
  - Na het invoeren of updaten van een row in deze tabel wordt er gekeken of de waarde van BUT_Type 'HPT' is. Zo ja, dan wordt er gekeken of de attributen PVN_ID en HPT_IsTransferred null zijn. Zijn deze kolommen null dan wordt er een passende error message getoond. Vervolgens wordt er gekeken of BUT_Type 'PBT' is. Zo ja, kijkt de trigger of PRD_ID null is. Is dit het geval dan wordt er een passende error message getoond.
- Forecast.NormCluster
  - Na het invoeren van een row in de tabel wordt er een row in Forecast.NormClusterPriority ingevoerd met een FK naar de nieuwe row in NormCluster en de waarde 0 als prioriteit. Vervolgens wordt deze prioriteit gewijzigd op basis van het ID.
- Realization.RawActualHistory
  - Na het invoeren van een row in de tabel wordt de waarde van CreateDate van deze row gewijzigd naar de huidige datum.

Naast deze triggers zijn er nog meer te vinden in verschillende tabellen.

9.4 Indexen

Naast de Primaire en Vreemde sleutels, zijn de volgende indexen terug te vinden in de database:

- Unique
- Clustered indexes
- Nonclustered indexes

9.5 Constraints

Binnen de database zijn ook een aantal constraints gelegd op tabellen. De meest voorkomende constraints zijn:

- NOT NULL constraints
- UNIQUE constraints (soms ook op meerdere kolommen)
- Kolommen waarvan de waarde alleen 'Y' of 'N' mogen zijn
  - Yes en no
- Kolommen waarvan de waarde beperkt is tot een aantal mogelijkheden.
- Foreign keys
- Primairy keys
- Default waardes (huidige datum, 0 enz.)
- Waarde is groter dan x
- Waarde is kleiner dan x
- Waarde ligt tussen x en y

10. Welke tabellen uit de huidige applicatie worden vaker samen opgehaald dan dat ze individueel opgehaald?

Aangezien er veel stored procedures zijn, waarin er soms meerdere queries per procedure zijn, is er besloten deze deelvraag met de opdrachtgever te bespreken. Hieruit bleek dat (bijna) alle tabellen in de database gebruik maken van joins om de data te kunnen filteren. Als er bijvoorbeeld een rooster van een bepaalde periode opgehaald moet worden, wordt er gejoint op de tabel met periodes, om een where uit te kunnen voeren op data uit de reference. Dit geld ook voor de tabel organizationlink, waarop alleen gefilterd wordt.

Een van de weinige uitzonderingen hierop is de tabel met instellingen. Deze wordt over het algemeen alleen bijgewerkt en alleen opgehaald.

11. Wat zijn de kracht en zwakheden van relationele databases en welk van deze krachten/zwakheden hebben betrekking op de huidige applicatie?

Om de krachten en zwakheden van relationele databases te achterhalen is gebruik gemaakt van bronnen. Daarnaast zijn een aantal krachten al naar voren gekomen tijdens een gesprek met mijn technisch begeleider en worden er een aantal krachten/zwakheden genoemd in de ISKA. [7]

11.1 Krachten/voordelen

Relationele databases zijn goed in de volgende punten:

- Data 1-malig opslaan [8]
    - Dit bespaard opslagruimte en het meerdere malen wijzigen van dezelfde data
- Er kan complex gequeried worden [8]
    - De gebruiker kan zelf bepalen welke combinatie van data hij op dat moment nodig heeft
- Het is mogelijk de tabellen te beveiligen [8]
    - Rechten binnen de databases
- De consistentie van de database wordt gewaarborgd door de aanwezigheid van constraints en ACID
- De integriteit van de data wordt gewaarborgd door de aanwezigheid van constraints en ACID
- Het wordt al jaren als standaard manier van data opslaan gebruikt
    - Veel externe partijen werken ermee/kunnen ermee werken
- Er is een standaard query taal(SQL) [9]
    - Inclusief functies (triggers, stored procedures enz.)

11.2 Zwakheden/nadelen

De zwakheden van relationele databases zijn:

- Het is niet goed in staat om horizontaal te schalen (database verspreiden over meerdere servers)
    - Het is mogelijk maar gaat ten koste van de performance
- Er is een vast schema
    - Als een applicatie willekeurige data heeft, of de datastructuur veranderd moet de database ook veranderd worden
    - Als alleen bepaalde types in de tabel een waarde hebben, wordt de database gevuld met NULL-waardes
- De data heeft een hele andere structuur dan de applicatie die met de data moet werken
- De constraints en joins van tabellen gaan ten koste van de performance

11.3 Betrekking op huidige applicatie

De volgende krachten en zwakheden hebben betrekking op de huidige applicatie.

11.3.1 Betrekking van de voordelen/krachten

De volgende voordelen/krachten hebben betrekking op de huidige applicatie:
Er kan complex gequeried worden:
Omdat de data in de huidige database genormaliseerd is, is het een groot voordeel dat deze data in verschillende combinaties op te halen is. Zo is op het ene scherm data uit tabel A en B nodig, en in het andere scherm data uit tabel A en C.

De beveiliging:
Binnen de huidige databases zijn verschillende users met verschillende rollen. Op deze manier wordt voorkomen dat mensen onnodig bij data kunnen waar zij niks mee te maken hebben.

De aanwezigheid van ACID:
Hierdoor wordt er gegarandeerd dat de data in de database voldoet aan de consistentie –en integriteitseisen die aan de data gesteld wordt.

Er is een standaard query taal:
Alle ontwikkelaars en databasebeheerders weten hoe zij data in en uit de database kunnen stoppen/ophalen

11.3.2 Betrekking van de nadelen/zwakheden

Hiernaast hebben de volgende nadelen/zwakheden betrekking op de huidige applicatie:
Het is niet goed in staat horizontaal te schalen:
Aangezien de Vries WFM van plan is hun klantengroep uit te breiden naar het buitenland, zal de database meer en meer records gaan bevatten. Op een bepaald punt zal de huidige server dus niet voldoende capaciteit bevatten om alle data te kunnen bewaren. Met hun huidige database kunnen zij deze database wel verticaal schalen (meer/krachtigere hardware op de huidige server zetten) maar dit wordt op een bepaald punt duur of zelfs niet te doen (de hoeveelheid hardware die een server aankan is gelimiteerd).

De joins/constraints gaan ten koste van de performance:
Er worden in de huidige applicatie heel veel joins gedaan om de juiste (combinatie van) data te verkrijgen. Het kost zowel tijd als kracht om deze data bij elkaar te krijgen, oftewel performance.

Ook zijn er constraints op bepaalde kolommen/tupels/tabellen die de performance van het opslaan omlaaghalen. Het is namelijk zo dat al deze checks uitgevoerd moeten worden voordat de data opgeslagen/gewijzigd mag worden.

De database heeft een hele andere structuur dan de data die de applicatie gebruikt:
Dit is nadelig voor de huidige applicatie, omdat de opgehaalde data altijd omgezet moet worden naar een object wat gebruikt wordt door de applicatie. Ook moet er altijd rekening

gehouden worden met de structuur van de database op het moment dat er iets nieuws ontwikkeld wordt.

12. Wat zijn de krachten en zwakheden van Document Oriented databases en welk van deze krachten/zwakheden hebben betrekking op de huidige applicatie?

Om de krachten en zwakheden van document databases te achterhalen is gebruik gemaakt van bronnen. Daarnaast worden er een aantal krachten/zwakheden genoemd in de ISKA. [7]

12.1 Krachten/voordelen

Document databases zijn goed in de volgende punten:

- Het is makkelijk te schalen, ook horizontaal [9]
  - Omdat de documenten niet aan elkaar gelinkt zijn door constraints en documents embed worden binnen andere documents in plaats van de FK
- Het is sneller in zowel het ophalen als het opslaan van data [9]
  - Vanwege het ontbreken van constraints, het ontbreken van joins en omdat de data precies is opgeslagen als in de applicatie
- Er is geen vast schema [7]
  - Data kan flexibel opgeslagen worden
- De data wordt op dezelfde manier opgeslagen als hoe de applicatie met de data werkt [7]
- Het ontwikkelen van de applicatie wordt makkelijker [7]
  - De database is precies hetzelfde ingericht als de applicatie
- De meeste implementaties werken met JSON, BSON of XML [7]
  - Dit is web vriendelijk aangezien de meeste browsers hier ook mee werken

12.2 Zwakheden/nadelen

De zwakheden van document databases zijn:

- Er is geen standaard query taal [9]
- Niet alle document databases garanderen ACID-transacties [9]
- Het in nog niet zo volwassen als relationele databases [9]
  - Niet alle externe systemen zijn erop gebouwd
  - Er zijn minder tools beschikbaar
- Er is geen vast schema [7]
  - De database kan een rommel worden als de applicatie(s), die gebruik maken van de database, onjuist in elkaar zit(ten)
- De applicatie(s), die gebruik maken van de database, wordt/worden complexer omdat de database als het ware versimpeld is [7]
- Het kan voorkomen dat data meerdere malen is opgeslagen in de database [7]
  - Als deze data moet worden geüpdatet moet dit op meerdere plaatsen

12.3 Betrekking op huidige applicatie

De volgende krachten en zwakheden hebben betrekking op de huidige applicatie.

12.3.1 Betrekking van de voordelen/krachten

De volgende voordelen/krachten hebben betrekking op de huidige applicatie:
Het is makkelijk te schalen:
Aangezien de Vries WFM van plan is hun klantengroep uit te breiden naar het buitenland, zal de database meer en meer records bevatten. Op een bepaald punt zal de huidige server dus niet voldoende capaciteit bevatten om alle data te kunnen bewaren. Met hun huidige database kunnen zij deze database wel verticaal schalen (meer/krachtigere hardware op de huidige server zetten) maar dit wordt op een bepaald punt duur of zelfs niet te doen (de hoeveelheid hardware die een server aankan is gelimiteerd).

Door middel van horizontale schaalbaarheid kunnen deze koste beperkt blijven en is de mogelijkheid om op te schalen onbeperkt, daar er altijd een nieuwe servers geplaatst kan worden en de database verspreid kan worden over deze servers. Ook hoeven de servers niet al te dure hardware te hebben, omdat zij slechts een deel van de database aan moeten kunnen.

Sneller ophalen/opslaan van data:
Binnen de database van R&R-web zijn er een aantal tabellen welke enkel gebruikt worden om te filteren (bijvoorbeeld Reference.Period en Organization.OrganizationLink). De data in deze tabellen wijzigen niet, er wordt alleen soms toegevoegd. Het is dus niet erg als deze gegevens embed zouden worden in de tabellen die momenteel een FK naar dit soort tabellen hebben.

Als gevolg hiervan, hoeven er niet zoveel joins uitgevoerd te worden bij het ophalen van de data en zal het ophalen dus sneller gaan dan momenteel.

De database is hetzelfde ingericht als de applicatie/het ontwikkelen van de applicatie wordt makkelijker:
Dit voordeel spreekt voor zich, de database zal dezelfde structuur hebben als de applicatie. Hierdoor hoeft er tijdens het ontwikkelen geen rekening gehouden te worden met de structuur van de database.

De meeste implementaties werken met JSON, BSON of XML
Daar R&R-web een webapplicatie is, kan het dus omgaan met deze formaten. Ook het versturen van de data naar de database en het ophalen ervan is mogelijk in een webapplicatie.

12.3.2 Betrekking van de nadelen/zwakheden

Hiernaast hebben de volgende nadelen/zwakheden betrekking op de huidige applicatie:
De complexiteit van de applicatie gaat omhoog:
Functionaliteiten die momenteel in de database zitten (constraints, functies enz.) moeten deels verplaatst worden naar de applicatie. Het is in de meeste implementaties van

document databases nog wel mogelijk om bepaalde functies te schrijven voor de database, maar daar blijft het bij.

ACID wordt niet gegarandeerd:
Het is mogelijk dat transacties niet volledig aankomen, of dat ze de consistentie van de database beïnvloeden. Ook hier moet rekening mee gehouden worden op applicatie niveau.

Het is nog niet zo volwassen:
Door de onvolwassenheid van document databases, zijn er nog minder tools en mogelijkheden beschikbaar.

13. Conclusie

Om tot een conclusie te kunnen komen, worden de resultaten uit de laatste 2 deelvragen naast elkaar gezet. Hier komt het volgende uit:

Voordelen/krachten met betrekking tot de huidige applicatie:

Relationeel	Document
Er kan complex gequeried worden	Het is makkelijk te schalen
De beveiliging	Sneller ophalen/opslaan van data
Er is een standaard query taal	De database is hetzelfde ingericht als de applicatie/het ontwikkelen van de applicatie wordt makkelijker
De aanwezigheid van ACID	De meeste implementaties werken met JSON, BSON of XML

Op basis van het aantal voordelen die betrekking hebben op de huidige applicatie gaat het gelijk op. Maar als de voordelen tegen elkaar opgewogen worden blijkt dat document databases toch de overhand hebben.

Het is namelijk zo, dat het 1e voordeel (complexe query's) van een Relationele database over het algemeen ook niet nodig zal zijn vanwege onder andere het embedden van documenten.

De aanwezigheid van een standaard query taal een voordeel, omdat iedereen deze begrijpt. Maar bij document databases wordt vaak JSON, BSON of HTTP gebruikt om query's uit te voeren. Oftewel, er is geen standaard maar de gebruikte query taal is ook niet iets nieuws.

Wat er dan overblijft zijn de beveiliging en ACID voor relationele databases. Aan de kant van document databases zijn de belangrijkere voordelen: de performance, de schaalbaarheid en de inrichting van de database structuur.

Dit zijn de belangrijkere voordelen, omdat performance iets is wat de klant direct merkt, aangezien hun applicatie sneller zal gaan. De inrichting van de database versimpeld zoals eerder gezegd het ontwikkelen van de applicatie wat in ontwikkeltijd en kosten zal schelen, en de horizontale schaalbaarheid zorgt ervoor dat de Vries zijn klantengroep kan blijven uitbreiden, zonder dat dit gelimiteerd wordt door servercapaciteit.

Dit alles resulteert in de volgende 'relevante' voordelen:

Relationeel	Document
De aanwezigheid van ACID	Schaalbaarheid
De beveiliging	Performance
	Inrichting van de database structuur

Nadelen/zwakheden met betrekking tot de huidige applicatie:

Relationeel	Document
Het is niet goed in staat horizontaal te schalen	De complexiteit van de applicatie gaat omhoog
De joins/constraints gaan ten koste van de performance	ACID wordt niet gegarandeerd
De database heeft een hele andere structuur dan de data die de applicatie gebruikt	Het is nog niet zo volwassen

Als er ook hier gekeken wordt naar de hoeveelheid nadelen, gaat het weer gelijk op. Als we kijken naar waar de nadelen invloed op zullen hebben, zal het er als volgt uitzien:

Het beperkt horizontaal schalen van de relationele database, zorgt ervoor dat de hoeveelheid dat er geschaald kan worden gelimiteerd is. Dit betekent dat de hoeveelheid data, oftewel hoeveelheid klanten, die het bedrijf kan aannemen uiteindelijk ook gelimiteerd zal zijn. Dit limiet is wellicht momenteel nog niet heel relevant. Wat wel relevant is, is dat verticaal schalen op den duur ook duur kan worden. Betere hardware in je server zetten kan op een bepaald punt duurder worden dan meerdere goedkopere servers.

De performance is al besproken bij de voordelen, dus komt het neer op de structuur. Dit heeft op zich niet een heel groot impact op het bedrijf, de ontwikkelaars moeten hier gewoon rekening mee houden en de data converteren naar objecten.

Voor de document databases geld dat er meer complexiteit naar de applicatie gaat. Dit betekend dat de architectuur van de applicatie zodanig opgesteld zal moeten worden dat dit mogelijk is, zonder dat de applicatie op zich te complex wordt.

De onvolwassenheid is op zich niet heel erg, daar dit voornamelijk het gevolg heeft dat er weinig tot geen tools beschikbaar zijn en dat niet alle externe systemen met deze database om kunnen gaan. Aangezien R&R-web een REST API heeft waarmee gepraat wordt, hoeven de externe systemen niks van de database af te weten. Het niet garanderen van ACID is wel een belangrijke, aangezien dit de consistentie en de integriteit van de data in de database garandeert (in combinatie met constraints).

Om de nadelen in kaart te brengen zal het er dus zo uitzien:

Relationeel	Document
Performance	Geen ACID
Schaalbaarheid	Applicatie kan complex worden.

13.1 Conclusie

Op basis van de effecten van de voor –en nadelen (krachten en zwakheden) van beide type databases, ben ik tot de conclusie gekomen dat alle 4 de grote modules van R&R-web voordelen kunnen behalen uit het overstappen naar document databases. De huidige functionaliteit zal nog steeds werken, maar functies uit de database moeten verplaatst

worden naar de applicatie. Het enigste wat niet gegarandeerd is, is dat er rechten op de database toegekend kunnen worden. Dit zal dus binnen het bedrijf geregeld moeten worden.

13.2 Gevolgen overstap

Een aantal gevolgen die de overstap zal hebben op de huidige applicatie zijn:

- De applicatie zal meer data validatie moeten doen, om de huidige constraints op tabellen te vervangen
- De bestaande functies in de database moeten naar de applicatie gezet worden
- De structuur van models in de applicatie gaan bepalen hoe de database eruit zal zien
    - Embedding vs. referencing
    - Hier moet aan gedacht worden bij het maken van de models
- De performance zal omhooggaan
- Het zal mogelijk worden horizontaal te schalen
- De data laag gaat aangepast moeten worden.

13.2.1 Voorbeelden

Hieronder een aantal voorbeelden waar deze gevolgen invloed op hebben binnen de R&R-web applicatie.

Meer validatie op applicatie niveau:
Een voorbeeld hierbij is een constraint die in de huidige database voorkomt: starttijd moet voor de eindtijd zijn. Daar document databases dat niet valideren moet de applicatie dit nu doen.

Functies uit de database naar de applicatie:
Een functie die in de database staat haalt het Id op van een periode op basis van beschrijving en jaar. Deze functie zal in de nieuwe situatie in de applicatie moeten zitten.

De structuur van models bepaald structuur database:
Waar hier rekening mee gehouden moet worden is wanneer er embed wordt en wanneer een reference gebruikt wordt. Een voorbeeld van data wat prima embed kan worden zijn periode gegevens. Deze gegevens wijzigen nooit, en worden enkel aangevuld met nieuwe rows(of in de toekomst nieuwe documents). Daar deze data niet wijzigt is het niet erg als het op 100en plaatsen voorkomt.

Een voorbeeld van data wat liever gereferenced wordt kan zijn de gegevens over een dienstverband. Een dienstverband bevat namelijk informatie wat vaker kan wijzigen. Zo veranderen contracturen en uurlonen van personen toch wel eens. Daar een dienstverband ook elke week op minimaal 1 rooster voorkomt, zou er per wijziging minimaal (1*aantal weken in dienst) documents geüpdatet moeten worden.

Performance zal omhooggaan:
Omdat er niet zoveel complexe joins meer gedaan hoeven te worden, zal de performance van de database waarschijnlijk omhoog gaan.

Het wordt mogelijk horizontaal te schalen:
Op het moment dat de database servers het niet meer aan gaan kunnen(met oog op de uitbreiding naar Duitsland) kunnen er nieuwe servers neergezet worden waar delen van de database op kunnen draaien. Zo kan een database bijvoorbeeld opgedeeld worden per module van R&R-web, of zelfs per collection.

De data laag gaat aangepast worden:
Daar de database veranderd, zal de datalaag mee moeten veranderen.

14. Literatuurlijst

Nr	Auteur	Titel	Link	Hfst.
1	Margaret Rouse	ACID (atomicity, consistency, isolation, and durability)	http://searchsqlserver.techtarget.com/definition/ACID	Geheel artikel
2	Kristina Chodorow	MongoDB the Definitive Guide	http://usuaris.tinet.cat/bertolin/pdfs/mongodb_%20the%20definitive%20guide%20-%20kristina%20chodorow_1401.pdf	8
3	Ameya Nayak, Anil Poriya, Dikshay Poojary	Type of NOSQL Databases and it's comparison with relational databases	http://research.ijais.org/volume5/number4/ijais12-450888.pdf	3
4	Harley Vera, Wagner Boaventura, Maristela Holanda, Valeria Guimaraes, Fernanda Hondo	Data modeling for NoSQL Document-Oriented Databases	http://ceur-ws.org/Vol-1478/paper17.pdf	3
5	Oracle	A Relational Database Overview	https://docs.oracle.com/javase/tutorial/jdbc/overview/database.html	Geheel artikel
6	Joseph V. Homan, Paul J. Kovacs	A COMPARISON OF THE RELATIONAL DATABASE MODEL AND THE ASSOCIATIVE DATABASE MODEL	http://iacis.org/iis/2009/P2009_1301.pdf	RELATIONAL DATABASE MODEL
7	Niels Nagle, Hylke Peek	Dive into NoSQL with Azure	https://www.youtube.com/watch?v=S3z5xnUHo-Q	Gehele film
8			http://www.teach-ict.com/as_as_computing/ocr/H447/F453/3_3_9/database_design/miniweb/pg8.htm	
9	Ameya Nayak, Dikshay Poojary, Anil Poriya	Type of NOSQL Databases and its Comparison with Relational Databases	http://research.ijais.org/volume5/number4/ijais12-450888.pdf	3

Bijlage A: Keuze bronnen

In deze bijlage zijn de bronnen te vinden welke wel/niet gebruikt zijn voor de hoofdstukken: 3, 4, 11 en 12. Per bron wordt er aangegeven of hij wel of niet gebruikt wordt, waarom hij wel of niet gebruikt wordt en voor welk onderdeel/hoofdstuk hij gebruikt wordt.

Bron: http://www.ascent.tech/wp-content/uploads/documents/mongodb/10gen-top-5-nosql-considerations.pdf
Wel/niet: Niet

Indien wel, voor welk onderdeel/welke hoofdstukken:

-

Toelichting:
De onderdelen omtrent document oriented databases in het document is te gericht op MongoDB. Aangezien de auteur er belang bij heeft om MongoDB te promoten, zie ik dit niet als een betrouwbare bron.

Bron: http://www.datasciencecentral.com/profiles/blogs/9-lessons-picking-the-right-nosql-tools
Wel/niet: Wel

Indien wel, voor welk onderdeel/welke hoofdstukken:

- Voor het hoofdstuk: wat zijn document oriented databases
    - Lesson 1: What Are Your Options? NOSQL/ NewSQL
        - De opsomming, punt 2

Toelichting:
Het artikel is geschreven door William Vorhies, een Amerikaanse man met meerdere jaren ervaring in het analyseren van data. Databases vallen onder data. Verder heeft hij geen belang bij het promoten/afkraken van de document oriented databases.

Bron: http://ceur-ws.org/Vol-1478/paper17.pdf
Wel/niet: Wel

Indien wel, voor welk onderdeel/welke hoofdstukken:

- Voor het hoofdstuk: wat zijn document oriented databases, structuur
    - 3 Data Modeling For Document-Oriented Database

Toelichting:
Het artikel is geschreven door 5 studenten aan de universiteit van Brasília. Dit maakt het artikel een universitair verslag, wat een van de minimale eisen is wat ik aan mijn bronnen heb gesteld.

Bron: http://cs.ulb.ac.be/public/_media/teaching/infoh415/student_projects/couchdb.pdf
Wel/niet: Wel

Indien wel, voor welk onderdeel/welke hoofdstukken:

- Voor het hoofdstuk: wat zijn document oriented databases
  - 2 What is a document database
- Voor het hoofdstuk: wat zijn de krachten en zwakheden van document oriented databases
  - 2.1.2 Why choose NoSQL?
  - 2.1.3 Performance overview of different databases

Toelichting: Het is een universitair verslag geschreven door 1 student. Verder is het recentelijk geschreven, 12 oktober 2015.

Bron: http://research.ijais.org/volume5/number4/ijais12-450888.pdf
Wel/niet: Wel

Indien wel, voor welk onderdeel/welke hoofdstukken:

- Voor het hoofdstuk: wat zijn document oriented databases
  - 2.3 Document store Databases
- Voor het hoofdstuk: wat zijn de krachten en zwakheden van document oriented databases
  - 3 NOSQL Databases v/s RDBMS

Toelichting:
Het is een universitair verslag geschreven door 3 studenten aan de universiteit van Mumbai. Er moet rekening gehouden worden dat het document bijna 3 jaar oud is, en mogelijk (deels) verouderd kan zijn.

Bron: http://www.christof-strauch.de/nosqldbs.pdf
Wel/niet: Niet

Indien wel, voor welk onderdeel/welke hoofdstukken:

Toelichting:
Het is een goed artikel en valt ook onder de category universitair verslag, maar gezien de al geaccepteerde bronnen biedt het geen meerwaarde voor mijn verslag.

Bron: http://www.teach-ict.com/as_as_computing/ocr/H447/F453/3_3_9/database_design/miniweb/pg8.htm
Wel/niet: Wel

Indien wel, voor welk onderdeel/welke hoofdstukken:

- Voor het hoofdstuk: wat zijn de krachten en zwakheden van relationele databases
    - Volledig artikel

Toelichting:
Het artikel is afkomstig van een site wat zich bezighouden met overbrengen van kennis(leren). De sectie binnen de site waar het artikel te vinden is, is bedoeld als leermateriaal voor studenten die zich willen kwalificeren.

Bron: http://iacis.org/iis/2009/P2009_1301.pdf
Wel/niet: Wel

Indien wel, voor welk onderdeel/welke hoofdstukken:

- Voor het hoofdstuk: wat zijn relationele databases
    - Relational Database Model

Toelichting:
Het artikel is geschreven door 2 studenten aan de Robert Morris University, een privé-universiteit in Amerika. Hierdoor volstaat het als universitair verslag.

Bron: http://sais.aisnet.org/2013/Nance.pdf
Wel/niet:

Indien wel, voor welk onderdeel/welke hoofdstukken:

-

Toelichting:

Bron: https://docs.oracle.com/javase/tutorial/jdbc/overview/database.html
Wel/niet: Wel

Indien wel, voor welk onderdeel/welke hoofdstukken:

- Voor het hoofdstuk: wat zijn relationele databases
    - Gehele document

Toelichting:
Het is een artikel staat op de site van Oracle, een grote speler in de IT-wereld, welke zelf ook zijn eigen databasemanagementsystemen heeft. Verder is dit artikel puur informatief waarin niks gepromoot/afgekraakt wordt.

Bron: http://searchsqlserver.techtarget.com/definition/ACID
Wel/niet: Wel

Indien wel, voor welk onderdeel/welke hoofdstukken:

- Voor het hoofdstuk: wat zijn relationele databases, uitleg acid
    - Gehele document

Toelichting:
Het is een kort artikel wat uitlegt waar ACID voor staat. Daar dit gewoon begripsuitleg is, stel ik geen hoge eisen aan de schrijver of de context hiervan.

Bijlage B: Glossary

De volgende termen zijn in dit document gebruikt:

Prognose: de naam van een van de modules van de applicatie.

Forecast: de naam van de prognose module zoals deze in het systeem wordt gebruikt.

Voorspelling: de vertaling van forecast, in de beschrijvingen van de prognose module, en bijbehorende functionaliteiten enz. wordt de Nederlandse term gebruikt.

Werkzaamheden/taak/taken: onderdelen van het rooster, dingen die medewerkers van de winkel moeten uitvoeren. Deze worden als hetzelfde gezien.

Rooster: de naam van een van de modules van de applicatie.

Realisatie: de naam van een van de modules van de applicatie.

Uitbetaling: de naam van een van de modules van de applicatie.

Schedule: de naam van de rooster module zoals deze in het systeem wordt gebruikt.

Realization: de naam van de realisatie module zoals deze in het systeem wordt gebruikt.

Payment: de naam van de uitbetaling module zoals deze in het systeem wordt gebruikt.

Bijlage C: Use case scenario's

De use case scenario's uit dit hoofdstuk horen bij de kern-functionaliteiten uit hoofdstuk 6. Deze scenario's zijn onderverdeeld in de modules waar ze bij horen.

C.1 Prognose

UC009:

Voor UC009 geld de volgende flow:

1. Het systeem toont het scherm 'Initialize shop forecast'
2. De winkel manager heeft de volgende opties:
    a. Initialiseer voorspelling voor de geselecteerde week in de 'Ribbon'
        i. De winkel manager selecteert de weken waarop de prognose gebaseerd zal worden.
        ii. Het systeem toont de gemiddelde omzet van de geselecteerde weken
        iii. Het systeem start sub flow: 'Initialize shop forecast'
    b. Wijzig openingstijden
        i. De winkel manager activeert de sectie: openingstijden
        ii. Het systeem start sub flow: 'wijzig openingstijden'
    c. Wijzig tijdspan van een taak
        i. De winkel manager activeert de sectie: openingstijden
        ii. Het systeem start sub flow: wijzig taken
    d. Wijzig verantwoordelijkheden
        i. De winkel manager activeert de sectie: openingstijden
        ii. Het systeem start de sub flow: Wijzig verantwoordelijkheden

Subflow: Initialize shop forecast:

De subflow start als de winkel manager een winkel voorspelling wilt initialiseren en de weken heeft geselecteerd:

1. Tijdens het selecteren van weken, berekent het systeem de gemiddelde omzet van deze weken en toont dat op het scherm.
2. De winkel manager geeft aan dat het initialiseringsproces kan beginnen op basis van de geselecteerde weken
3. Het systeem controleert of het geselecteerd aantal weken groter of gelijk is aan het aantal geconfigureerde historische weken (de setting: minimum).
4. Als het geselecteerd aantal weken niet kleine is dan het geconfigureerde aantal, checkt het systeem of er 1 of meerdere feestdagen voorkomen in de geselecteerde weken
5. Als er geen feestdagen zijn doet het systeem het volgende:
    a. Veranderd de status van de winkel voorspelling in: 'Tune' of 'Tune vacation'
    b. Het berekent de te berekenen items en toont de resultaten in het scherm: 'Initialize shop forecast-results'.
6. De winkel manager keurt de resultaten goed
7. Het systeem voert de volgende acties uit
    a. Slaat de voorspelling van de week op
    b. Veranderd de status van de shop voor de week naar 'Provisional' of 'Provisional Vacation'
    c. Veranderd de voorspellingsstatus van de afdelingen van de winkel naar 'Tune'
    d. Sluit het scherm.

Einde subflow.

Subflow: wijzig openingstijden:

De subflow start als de winkel manager de openingstijden van de geselecteerde winkel wilt wijzigen.

1. Het systeem toont het scherm: 'Opening hours' met de bijbehorende informatie
2. De winkel manager wijzigt de openingstijden en geeft aan de wijzigingen op te willen slaan
3. Het systeem slaat de nieuwe openingstijden op en her-berekent de omzet waarde per normcluster per nieuwe tijdsinterval
    a. Als je dezelfde omzet wilt behalen maar minder lang open bent moeten de waardes per uur hoger zijn dan oorspronkelijk berekent

Subflow: wijzig taken:

De subflow start als de afdelingsmanager de tijdspan van taken wilt wijzigen.

1. Het systeem toont het scherm: 'Taken' met de bijbehorende informatie
2. De afdelingsmanager wijzigt de tijdspan van de taken van zijn afdeling en geeft aan deze waarde op te willen slaan
3. Het systeem slaat de nieuwe tijdspannen op.

Subflow: wijzig verantwoordelijkheden:

De subflow start als de winkel manager ervoor kiest de tijdspan van verantwoordelijkheden te wijzigen.

1. Het systeem toont het scherm: 'Verantwoordelijkheden' met de bijbehorende informatie
2. De winkel manager wijzigt de tijdspan van verantwoordelijkheden en geeft aan te willen opslaan
3. Het systeem slaat de gegevens op

C.2 Rooster

UC012: Zonder template

De main flow start als de winkel manager aangeeft dat hij een rooster wilt initialiseren.

1. Het systeem toont het initialiseer rooster scherm en een lijst van winkel rooster templates
2. De winkelmanager selecteert geen template
3. De winkel manager geeft aan het winkelrooster te initialiseren
4. Als de geselecteerde week nog niet geïnitialiseerd, het systeem start het initialisatie proces en wijzigt de status van het rooster naar 'Initializing
5. Als het initialisatie proces klaar is veranderd het systeem de status automatisch naar 'Initialized'

UC012: Met template

De main flow start als de winkel manager aangeeft dat hij een rooster wilt initialiseren.

1. Het systeem toont het initialiseer rooster scherm en een lijst van winkel rooster templates
2. De gebruiker selecteert een rooster template om te initialiseren
3. Het systeem gebruikt de geselecteerde template om het winkelrooster te initialiseren
4. Het systeem doet het volgende per medewerker als het toepasselijk is voor de medewerker:
    a. Kopieert het rooster per dag van de template naar de rooster voor de geselecteerde week
    b. Verwerkt verlof verzoeken
    c. Verwerkt ziekmeldingen
    d. Pas arbeidsovereenkomst toe

e. Verwerk indeling van de werktijden
f. Bereken verlofrechten en reserveringen
g. Bereken arbeidskosten

UC013:
De main flow start als de afdelingsmanager aangeeft dat hij de afdelingsrooster wilt beheren.

1. Het systeem toont scherm 'Department schedule –Empty'
2. De afdelingsmanager selecteert een winkel, afdeling en een periode, en geeft aan dat hij de data van het rooster voor die afdeling wilt zien
3. Het systeem toont de data van het rooster van de afdeling
4. Als de status van het rooster 'New', 'Approved' of 'Actual' is, start het systeem de subflow: 'Check scheduled labor costs' voor elke getoonde medewerker
5. De afdelingsmanager selecteert een medewerker en wijzigt/vult de dienst in voor een specifieke dag
6. De afdelingsmanager bevestigd de wijzigingen
7. Het systeem start de subflow: 'Check scheduled labor costs' voor de geselecteerde medewerker
8. De afdelingsmanager geeft aan de wijzigingen op te slaan
9. Het systeem slaat de wijzigingen op
10. Het systeem start de subflow: 'Create Schedule file'

Subflow: Check scheduled labor costs:
Deze subflow beschrijft hoe het systeem de geroosterde arbeidskosten en de maximale arbeidskosten bepaald en hoe de waarschuwingen getoond worden.

1. Het systeem (her)berekend of de totaal geroosterde arbeidskosten van de medewerker voor elke maand wordt behaald in de geselecteerde week
2. Het systeem voert de volgende acties uit om de maximaal inroosterbare arbeidskosten van elke maand wordt behaald in de geselecteerde week:
    a. Bepaal welk contracttype van de medewerker geldig is op de laatste dag van de maand, of de laatste contract dag van de medewerker als de medewerker geen contract lopend tot het eind van de maand heeft
    b. Controleer of die contract type een maximaal inroosterbare arbeidskosten heeft op de laatste dag van de geselecteerde maand
3. Als dit het geval is vergelijkt het systeem de totaal geroosterde arbeidskosten van elke maand die betrekking heeft op de maximaal inroosterbare arbeidskosten van die maand
    a. Is het niet het geval doet het systeem niks en eindigt de subflow
4. Het systeem stelt een van de volgende opties vast:
    a. De geroosterde arbeidskosten zijn groter dan de maximaal inroosterbare arbeidskosten
        i. Het systeem toont bericht 'MSG1' naast de medewerkers naam
        ii. Het systeem toont bericht 'MSG2' naast alle geroosterde en getoonde dagen vanaf de dag waar de arbeidskosten de maximaal inroosterbare dagen voorbijgaan tot de laatste dag van de maand.
    b. De geroosterde arbeidskosten zijn niet hoger dan de maximaal inroosterbare arbeidskosten
        i. Het systeem doet niks

Subflow: Create Schedule file:

1. Het systeem controleert of de winkel Speakap gebruikt en of de status van het rooster 'Actual' is
    a. De winkel gebruikt SpeakAp en de status van het rooster is 'Actual'

i. Het systeem het rooster bestand (Schedule file) in .JSON formaat
ii. Het systeem upload het rooster bestand naar een Speakap service

UC014:
De main flow start als de winkelmanager aangeeft dat hij een rooster wilt afronden.

1. Het systeem toont 'Finalize schedule' en toont alle afdelingen die actief zijn in de geselecteerde week
2. Het systeem toont aanvullende informatie en status per afdeling
3. De winkelmanager geeft aan dat hij alle afdelingsroosters goedkeurt
4. Het systeem start subflow: 'Create schedule file'

Subflow: Create schedule file:

1. Het systeem controleert of de winkel Speakap gebruikt.
    a. De winkel gebruikt Speakap
        i. Het systeem maakt het rooster bestand (Schedule file) in .JSON formaat
        ii. Het systeem upload het rooster bestand naar een Speakap service

UC016:
De main flow start als de gebruiker aangeeft dat hij de templates van het afdelingsrooster wilt beheren.

1. Het systeem start template afdelingsrooster
2. Het systeem toont de lijst met winkels
3. De gebruiker selecteert een winkel
4. De gebruiker selecteert een afdeling en geeft aan dat hij de template van de afdelingsrooster wilt openen
5. Het systeem toont de template van de geselecteerde afdeling
6. De gebruiker reviewt de diensten
7. De gebruiker wijzigt de diensten
8. De gebruiker geeft aan dat hij akkoord gaat met de template
9. Het systeem heeft bepaald dat er geen geroosterde diensten zijn die conflicteren met de WHA
    a. (Het WHA is het Workinghoursact. Dit zijn de regels die onder andere vertellen dat er pauzes moeten plaatsvinden tussen werkblokken, je niet op zaterdag en zondag mag werken enz.)
10. Het systeem slaat de data op

Alternative flows:
Er is een blokkerende WHA-conflict gedetecteerd:
Als het systeem een WHA-conflict detecteert met een dienst en de setting van het conflict moet behandeld worden als blokkerend is actief dan:

1. Het systeem markeert de dienst in het rooster waarbij het conflict voorkomt en geeft een uitleg over de regel die de blokkeren veroorzaakt
2. De gebruiker corrigeert de conflicterende dienst
3. Het systeem merkt:
    a. Dat er nog steeds een blokkerend conflict is en begint bij stap 1 van deze alternative flow
    b. Er zijn geen blokkerende conflicten meer en het systeem slaat de shift op.

Er is een informatieve WHA-conflict gedetecteerd:
Als het systeem een WHA-conflict detecteert met een dienst en de setting van het conflict moet behandeld worden als informatief is actief dan:

1. Het systeem markeert elke dienst in het rooster waarbij het conflict met de WHA-regels als informatief is aangegeven en geeft informatie over de desbetreffende regels
2. Het systeem staat toe dat de data opgeslagen mag worden

UC017:

De main flow start als de gebruiker aangeeft dat hij de standaard beschikbaarheid wilt beheren.

1. Het systeem start 'Manage default availability'.
2. De gebruiker selecteert een winkel/locatie, selecteert een afdeling en geeft aan dat hij de standaard beschikbaarheid van alle medewerkers die voor die afdeling in die winkel/locatie werken wilt zien.
3. Het systeem toont de medewerkers en hun standaard beschikbaarheid
4. De gebruiker reviewt de standaard beschikbaarheid
5. De gebruiker past de standaard beschikbaarheid aan
6. De gebruiker geeft aan de standaard beschikbaarheid op te slaan
7. Het systeem slaat de data op

C.3 Realisatie

UC018:

1. Het systeem toont het scherm 'Realization per day' met alle medewerkers van de geselecteerde afdeling
2. De winkelmanager kan het volgende doen:
    a. Beheren realisatie uren
        i. Het systeem start subflow 'Manage realization per day'
3. Het systeem slat de wijzigingen op

Subflow: Manage realization per day:

1. Het systeem toont het scherm 'Maintain realization' met alle medewerkers van de geselecteerde afdeling
2. De afdelingsmanager wijzigt de bestaande diensten of voegt nieuwe diensten toe. De volgende gegevens kunnen gewijzigd worden: start/eindtijd, taken, winkel, afdeling
3. De winkelmanager geeft aan dat hij de gewijzigde/toegevoegde diensten wilt opslaan
4. Het systeem slaat de diensten op

UC019:

Pre-conditie: de winkelmanager heeft een winkel, afdeling en dag geselecteerd.

1. Het systeem toont het scherm 'Department&normcluster realization' met de volgende informatie:
    a. Een lijst van afdelingen die actief zijn in de organisatiestructuur van de geselecteerde winkel gedurende de geselecteerde week.
    b. Een lijst van normclusters die actief zijn in de organisatiestructuur van de geselecteerde winkel tijdens de geselecteerde week
2. De winkelmanager vult het volgende in:
    a. De omzet voor elke afdeling en normcluster
    b. Het aantal klanten van de hele winkel
3. De winkelmanager geeft aan de ingevulde omzet en aantal klanten op te slaan

UC061:

De use case wordt gestart door de automatische initialisatie van de uren realisatie of door de handmatige initialisatie van de uren realisatie door de gebruiker in R&R-web.

1. Het systeem bepaald de week waarvan de uren realisatie van een dag geïnitialiseerd moet worden
2. Het systeem controleert of het rooster voor die week afgerond is

3. Per dag moet het volgende geïnitialiseerd worden:
    a. Het systeem controleert of de winkel een 'ClockTimeSystem – Terminal times' gebruikt op die dag
    b. Het systeem controleert of de 'Shop-day source of hour realization' gelijk is aan 'ClockTime'.
    c. Per medewerker:
        i. Het systeem controleert of de medewerker de setting 'Employee source of hour realization = ClockTime' heeft
            1. Het systeem maakt kloktijden aan op basis van terminal tijden per terminal
        ii. Het systeem controleert of de setting 'ApplyBoundaryPolicy' op 'Yes' staat
            1. Het systeem past BoundayPolicy toe
        iii. Het systeem kopieert afgeronde kloktijden naar urenrealisatie, hierbij wordt gekeken naar:
            1. Ingeklokte uren in de winkel waar de medewerker werkt
            2. Ingeklokte uren in winkels waar de medewerker is uitgeleend (andere filialen)
        iv. Het systeem verwerkt verlofverzoeken
        v. Het systeem verwerkt ziektemeldingen
        vi. Het systeem past het arbeidsovereenkomst toe
        vii. Het systeem keurt de uren realisatie van de medewerker die dag goed.

Alternative flows:
De winkel gebruikt geen kloktijdensysteem:
Per dag wordt het volgende geïnitialiseerd per medewerker:
1. Het systeem kopieert alle diensten van activiteiten die aangegeven zijn ('CountInContract = Yes') van de medewerker van het actuele rooster naar uren realisatie van die dag.
2. Het systeem keurt de realisatie van de medewerker voor die dag goed.

UC062:
De main flow start als de manager aangeeft dat hij de urenrealisatie per week wilt beheren.
1. De manager kiest het menu realisatie, urenrealisatie per week
2. Het systeem toont het scherm 'Hour realization per Week(Selection)'
3. De manager selecteert een winkel, een week/jaar en alle afdelingen (of 1 afdeling) en geeft aan de realisatiedata te tonen
4. Het systeem controleert de status van de winkelrealisatie op de geselecteerde week en toont scherm 'Hour realization per Week(Actions)'
    a. De status is 'New': het systeem start subflow: 'New realization data'
    b. De status is 'Initialized', 'Tune' of 'Approved': Het systeem start subflow: 'Review realization data'
    c. De status is 'Finalized': het systeem start subflow: 'View only realization data'
5. Het systeem slaat de gewijzigde data op.

Subflow: new realization data:
1. Het systeem toont scherm: 'Initialize realization hours' met de dagen van de week, de geroosterde uren en de geklokte uren (als deze beschikbaar zijn)
2. Voor elke dag van de week kiest de manager als bron voor de uren "Schedule" en gebruikt de knop "Initialize" om de uren te initialiseren
3. Het systeem initialiseert de uren, voor alle dagen van de week

4. Het systeem wijzigt de status van de realisatie van de week naar ‘Tune’

Subflow: view only realization data:

1. Het systeem toont het scherm ‘Hour realization per week(Actions)’ met gerealiseerde data voor elke afgeronde dag
2. De manager kan alle data zien, maar kan niks wijzigen

Subflow: review realization data:

1. Het systeem toont scherm: ‘Hour realization per week(Actions)’.
2. De manager selecteert als filter: “Not approved” employees
3. De manager reviewt de getoonde data per medewerker per dag
4. Per dag:
    a. De manager keurt de data goed zoals deze wordt weergeven
    b. Het systeem slaat de gewijzigde data op en wijzigt de status van de uren realisatie naar “Approved”
5. Per afdeling:
    a. De manager rond per afdeling de realisatie van die afdeling af
    b. Het systeem wijzigt de status van de urenrealisatie van de afdeling naar “Finalized”
6. De manager drukt de knop “Finalize” om de uren realisatie van de winkel van de week af te ronden
7. Het systeem wijzigt de status van de urenrealisatie van de winkel van de week naar afgerond.

Bijlage D: Niet-kern functionaliteiten

De functionaliteiten uit deze bijlage zijn aangegeven als niet-kern functionaliteiten.

D.1 Normering

UC nr	Use case	functionaliteiten	Funct. nr
UC001	Bereken wekelijks budget	-	-
UC002	Beheer norm basis set		
UC003	Beheer norm gebaseerd op taak tijden		
UC004	Beheer norm tariff		
UC005	Beheer simulatie sets		
UC006	Simuleer norm		
UC007	Raadpleeg norm model		
UC008	Beheer norm modellen		
UC051	Review norm basis set		
UC055	Beheer arbeidskosten budget		
UCxxx1	Vergelijk norm set		
UCxxx2	Raadpleeg norm op specifiek moment		
UCxxx3	Bepaal wekelijks budget		
UCxxx4	Verkrijg jaar budget driver waarden van een specifieke week		
UCxxx5	Beheer normclusters		
UCxxx6	Beheer organisatiestructuur		
UCxxx7	Beheer taken en handelingen		
UCxxx7	Beheer jaar budget driver waarde		
UCxxx8	Beheer jaar budget uren		
UCxxx9	Overzicht norm sets		

D.2 Prognose

UC nr	Use case	functionaliteiten	Funct. nr
UC009	Als winkel manager wil ik een voorspelling voor de winkel initialiseren	Selecteren weken uit de geschiedenis	F065
		Gebruik jaar budget	F068
		Beheer verantwoordelijkheden	F052
		Beheer 'main parameter' (gemiddelde uitgave)	F045
		Beheer management sleutel	F046
		(her)bereken winkel prognose data (voor alle afdelingen)	F059
		Initialiseer voorspelling voor de afdelingen	F102
		Initialiseer rooster	F060
		Beheer 'main driver' per afdeling(week)	F121
UC010			
		Beheer parameter 1(gemiddelde uitgave) van de afdelingen per dag	F080
		Beheer parameter 2(gemiddelde unit waarde) van de afdelingen per dag	F081
		Herbereken de prognose data van de afdeling	F82
		Verdeel budget uren per taak in % over de dagen	F083
		Verwissel budget uren tussen grid en grafisch	F084
		Beheer vrachtschema per normcluster/voor alle normclusters van een afdeling	F085
UC011	Als winkelmanager wil ik winkel voorspelling reviewen	Beheer afdelingsvoorspelling	F032
		Keur afdelingsvoorspelling af	F008
		Beheer management sleutel	F046
		Rond winkel voorspelling af	F027
UC077	Als afdelingsmanager wil ik de voorspelling van mijn afdeling initialiseren	Initialiseer afdelingsvoorspelling	F102
	Nieuw	Uit augurk	
AUGP1	Voorspellen periode informatie		
AUGP2	Ophalen driver waarden per dag		

D.3 Rooster

UC nr	Use case	functionaliteiten	Funct. nr
UC000	Labor agreement old	-	-
UC012	Als winkelmanager wil ik het winkel rooster (her)initialiseren	Herinitialiseer rooster	F110
UC013		Beheer medewerkersdata	F033
		Toon medewerkers data	F012
		Toon berichten per medewerker per week en dag	F019
		Toon grafisch gerelateerd budget uren driver en andere taken en geroosterde uren	F014
		Sorteer Medewerker	F055
		Filter productieve uren	F023
		Wissel van dag naar week schema en andersom	F067
		Toon vertrek(icoon)	F071
		Toon ziekte(icoon)	F072
		Toon te betalen feestdagen(icoon)	F073
		Toon ingeleende medewerker(s)	F075
		Open rapport 'R04. Schedule'	F096
		Review afdelingstaken (= budgeturen per taak per dag)	F010
		Sorteer/filter kwalificatie type	F056
		Sorteer/filter taak type	F057
		Beheer beschikbaarheid medewerkers per week	F054
		Beheer beschikbaarheid medewerkers per dag	F035
		Toon management informatie	F018
UC014		Beheer periode rooster van afdeling	F069
		Toon per afdeling: 'there are warnings'	F070
		Open rapport 'R06. Afronden rooster'	F111
UC015	Als gebruiker wil ik de template winkel rooster beheren	-	-
UC016		Review details van de activiteiten van medewerkers	F013
		Inlenen medewerker	F029
		Beheer medewerkers data	F033
		Toon medewerkers data	F012
		Toon management informatie	F018
		Sorteer medewerker	F055
		Filter productieve uren	F023
		Wissel van dag naar week schema en andersom	F067
		Open rapport 'R19. Template department schedule'	F093

UC017		Toon management informatie	F018
	Nieuw	Uit augurk	
AUGS1	Herbereken winkelrooster		
AUGS2	Herinitialiseer winkelrooster		
AUGS3	Haal organisatie informatie op		
AUGS4	Sla de afdelingsrooster op als template afdelingsschema		
AUGS5	Rooster periode informatie		
AUGS6	Pas regelementen toe op rooster		
AUGS7	Rond winkelrooster af		
AUGS8	Neem medewerker van een ander filiaal aan in het rooster		
AUGS9	Initialiseer winkelrooster		
AUGS10	Beheer afdelingsrooster		
AUGS10.1		Haal prognose informatie op	
AUGS11	Beheer activiteiten van medewerkers		
AUGS12	Beheer template van activiteiten van medewerkers		
AUGS13	Beheer schoolroosters		
AUGS14	Beheer template van afdelingsrooster		
AUGS15	Overzicht schoolroosters		
AUGS16	Winkelrooster		
AUGS17	Medewerker schoolrooster		
AUGS18	Beheer medewerkersbeschikbaarheid per periode		

D.4 Realisatie

UC nr	Use case	functionaliteiten	Funct. nr
UC000	Labor agreement	-	-
UC018		Filter afgekeurde/alle medewerkers	F022
		Beheer medewerker data	F033
		Sorteer medewerkers	F055
UC019		Bereken realisatie gebaseerd op budget uren en budget arbeidskosten	F122
		Sla realisatie per dag op	F119
UC020	Als winkelmanager wil ik winkel uren realisatie reviewen	-	-
UC062		Toon geplande/gerealiseerde uren/bron uren realisatie per dag	F020
		Reset winkel realisatie(verwijder uur realisatie voor 1 of meer dagen)	F007
		Beheer medewerkers realisatie: wijzig begintijd, eindtijd en pauze	F042
		Beheer activiteiten van medewerkers(alle dage)	F077
		Beheer activiteiten van medewerkers(alle dagen: per dag toevoegen/wijzigen/verwijderen van data)	F040
		Beheer activiteiten van medewerkers(alle dagen: per dag goedkeuren van de dag)	F041
		Beheer activiteiten van medewerkers(alle dagen: selecteer volgende/vorige medewerker)	F038
		Selecteer tonen van pauze of netto tijden	F066
		Beheer medewerkers data	F033
		Toon medewerkers data	F012
		Filter afgekeurde/alle medewerkers	F022
		Sorteer medewerkers	F055
		Toon verloof (icoon)	F071
		Toon ziekte(icoon)	F072
		Toon waarschuwing niet verlopen ziekmeldingen	F074
		Toon ingehuurde medewerkers	F075
		Toon lijst van geautoriseerde afdelingen	F076
		Afronden afdelingsuren realisatie	F024

		Afronden afdelingsrealisatie	F025
		Af/goedkeuren uren realisatie van alle medewerkers per dag	F001
UC076	Verwerk realisatie data(BudgetFact)	Verwerken realisatie data	F097
UC079	Labor agreement(new)	Uren classificatie	F106
		Uren specificatie	F107
		Balans toename	F109
		Dag vergoeding	F108
	Nieuw	Uit augurk	
AUGR1	Toepassen regelementen		
AUGR1.1	Toekennen weekrounding op afdelingen		
AUGR2	Bereken klanten op checkout normclusters		
AUGR3	Afronden afdelingsrealisatie		
AUGR4	Initialiseren winkelrealisatie		
AUGR5	Beheren medewerkers activiteiten		
AUGR6	Beheren medewerkersrealisatie		
AUGR7	Verwerken terminal tijden		
AUGR8	Realisatie periode informatie		
AUGR9	Herberekenen medewerkersrealisatie		
AUGR10	Herbereken winkel realisatie		
AUGR11	Reset winkel realisatie		
AUGR12	Winkel realisatie		

D.5 Uitbetaling

UC nr	Use case	functionaliteiten	Funct. nr
UC022		Beheer winkel betalingen op feestdagen(uren te betalen per feestdag)	F036
		Beheer data van medewerkers	F033
		Toon data van medewerkers	F012
		Sorteer medewerkers	F055
		Beheer betalingen van medewerkers per week: selecteer begintijd, eindtijd en pauze	F050
		Beheer betalingen van medewerkers per week: Selecteer volgende/vorige medewerker	F051
		Goed/afkeuren betaling medewerker	F048
		Goed/afkeuren betaling van alle medewerkers	F003
		Review werknemers loon(tonen uur specificaties, incl. reserveringen en arbeidskosten)	F015
		Review werknemers loon op een dag	F116
		Corrigeer verlof balans van een medewerker(kunnen ook correcties per week zijn)	F043
		Afronden betaling	F026
		Terugdraaien betaling	F062
UC023	Als winkelmanager wil ik data definities exporteren	Her-exporteer data	F091
UC079	Arbeidsovereenkomst	Uur classificatie	F106
		Uur specificatie	F107
		Stijgingen in het balans	F109
		Dag lonen	F108
UC080	Her-bereken medewerkers betalingen	Her-bereken medewerkers betalingen	F115
	Nieuw	Uit augurk	
AUGU1	Toepassen regelementen		
AUGU1.1	Toekennen weekrounding op afdelingen		
AUGU2	Corrigeren medewerkers verlofbalans		
AUGU3	Initialiseren winkel betaling		

AUGU4	Beheer activiteiten van medewerker		
AUGU5	Beheer medewerkers aanwezigheidspatroon		
AUGU6	Beheer medewerkers betaling		
AUGU7	Beheer feestdagen betalen van de winkels		
AUGU8	Betaalperiode informatie		
AUGU9	Her-bereken medewerkers betalingen		
AUGU10	Her-bereken winkel betaling		
AUGU11	Reset medewerkersbetaling		
AUGU12	Review medewerkerssalaris		

D.6 Rapportages

UC nr	Use case	functionaliteiten	Funct. nr
ER01	Exporteren van het rapport: betaalbare uren		
ER02	Exporteren van het rapport: betaalbare salarissen		

De volgende rapporten kunnen gegenereerd worden binnen R&R-web.

Rapport nummer	Rapport
R05	Verantwoordelijkheden
R1	Afrondingsrapport
R10	Realisatie per week
R11	Betaalbare uren
R13	Kloktijden
R14	Salaris per medewerker
R15	Roosteren door salaris
R16	Niet juist verwerkte artikelgroepen
R17	Verlofkaart
R19	Template rooster
R2	Prognose winkel
R20	Afwezigheidsrooster
R21	Verlofbalans
R23	Afdelingsafwezigheid
R24	Jaaroverzicht medewerker
R27	Ziekte kaart
R28	Kloof in uren
R3	Taken rapport
R30	Winkelstatus
R4	Rooster
R6	Afronding rooster
R7	Goedkeuring realisatie
R8	Realisatie per medewerker
R9	Realisatie per dag

Bijlage E: Use cases niet-kern functionaliteiten

In deze bijlage zijn de use cases te vinden die horen bij de niet-kern functionaliteiten van R&R-web. Deze use cases zijn onderverdeeld in de modules waar ze bij horen.

E.1 Prognose

AUGP1:

De feature beschrijft de voorspellingsfunctionaliteit voor een periode selectie van jaren met bijbehorende weken. Als de functionaliteit aangeroepen wordt, wordt er een lijst met beschikbare periodes gepresenteerd en er is altijd een standaard periode(jaar en week). De selectie van de periode informatie heeft de volgende specifieke regels voor de prognose module:

- Periodes die opgevraagd kunnen worden voor winkelvoorspellingen zijn allemaal periodes vanaf de startdatum van de winkel tot de laatst beschikbare periode, of de einddatum van de winkel. De standaard periode van de winkel voorspelling is de eerste met status: "NEW", "TUNE" of "TUNE VACATION", "PROVISIONAL" of "PROVISIONAL VACATION", "READYFORFINALIZATION" of "FINALIZED"
- Periodes die opgevraagd kunnen worden voor afdelingsvoorspellingen zijn allemaal periodes vanaf de startdatum van de winkel tot de laatst beschikbare periode, of de einddatum van de winkel. De standaard periode van de winkel voorspelling is de eerste met status: "NEW", "TUNE" of "TUNE VACATION", "PROVISIONAL" of "PROVISIONAL VACATION", "READYFORFINALIZATION" of "FINALIZED"
- Tellen in appartementen kan niet gevraagd worden voor afdelingsvoorspellingen totdat alle afhankelijke afdelingen zijn afgerond

AUGP2:

Voorspelling omzet waarde worden volhardt in periodes van een uur. Deze ophaal feature haalt de waarde op van de 4 standaard drivers op geaggregeerd per dag. De 4 standaard drivers zijn:

- Omzet
- Eenheden
- Klanten
- Collo

E.2 Rooster

UC015:

De main flow start als de gebruiker aangeeft dat hij de templates van winkelroosters wilt beheren.

1. Het systeem toont 'Shop schedule template'
2. De gebruiker selecteert een bestaande template
3. De gebruiker wijzigt de naam van de template
4. Het systeem slaat de template data op

Alternative flows:

Create new shop schedule template:

Als de gebruiker een naam voor de nieuwe template toevoegt en op de + knop drukt maakt de winkel een nieuwe winkelrooster template en slaat deze op met de aangegeven naam. Ga verder met stap 1(main flow)

Copy and create new shop schedule template:

Als de gebruiker aangeeft dat hij de nieuwe winkelrooster wilt kopiëren dan:

1. Het systeem haalt de diensten voor afdelingen die aan de volgende criteria voldoen op:
    a. Bestande afdelingen die eindigen, die zijn begonnen voor de huidige datum en eindigen na de huidige datum
    b. Bestaande afdelingen die niet eindigen, die zijn begonnen voor de huidige datum en eindigen na de huidige datum of zonder einddatum
    c. Nieuwe afdelingen die in de toekomst actief worden waarbij de begindatum na de huidige datum is en de einddatum na de huidige datum is of leeg is
2. Het systeem kopieert de geselecteerde diensten naar een nieuwe winkelrooster template met de door de gebruiker aangegeven naam
3. Ga verder met stap 1(main flow)

Verwijder winkelrooster template zonder inroosterde diensten:
Als de gebruiker aangeeft dat een geselecteerde winkelrooster template verwijderd moet worden en het systeem bepaald heeft dat er geen ingeroosterde diensten in de winkelrooster zitten dan:
1. Het systeem verwijderd de geselecteerde template
2. Ga verder met stap 1(main flow)

Verwijder winkelrooster template met ingeroosterde diensten:
Als de gebruiker aangeeft dat een geselecteerde winkelrooster template verwijderd moet worden en het systeem bepaald heeft dat er ingeroosterde diensten in de winkelrooster zitten dan:
1. Het systeem vraagt voor bevestiging om te verwijderen
2. Als de gebruiker ja kiest
    a. Het systeem verwijderd de template
3. Ga verder met stap 1(main flow)

Herinitialiseer winkelrooster:
Door het verstrekken van een winkel en een week zal de rooster voor die winkel herberekend worden voor de gespecificeerde week en alle weken daarna.

Herinitialiseer winkelrooster:
Een winkelrooster met de status “Initialized” kan herinitialiseerd worden, optioneel met een template rooster.

De herinitialisatie bestaat uit:
1. Optioneel: selecteren van template rooster als bron
2. Selecteren voor het behouden van, of handmatig verwijderen van wijzigingen na de eerste initialisatie en de roosteractiviteiten aanmaken
3. Als de optie verwijderen is geselecteerd zal het rooster geinitialiseerd worden zonder de handmatige toevoegingen na de eerste initialisatie.
4. Als er gekozen is om de wijzigingen te behouden zal de rooster geinitialiseerd worden en handmatig toegevoegde wijzigingen zijn behouden na de herinitialisatie. Als de handmatige wijzigingen conflicteren met de gebruikte template, zullen de activiteiten van de template niet meegenomen worden.
5. De regelementen worden toegepast op de aangemaakte activiteiten.

Na het herinitialiseren van de rooster wordt de status van deze week voor de winkel en al zijn afdelingen op “Initialized” gezet.

Haal organisatie informatie op:

Deze functie beschrijft de organisatie informatie die nodig is om een winkelrooster op te halen in de beheer afdelingsrooster use case. De benodigde informatie gaat over: winkel openingstijden, afdelingsopeningstijden en verlofbalansen van medewerkers.

Sla de afdelingsrooster op als template afdelingsschema:

-

Rooster periode informatie:
Deze functie beschrijft welke periodes(jaar en week) beschikbaar zijn voor het selecteren en welk jaar en week de standaard periode is. De selectie van de periode informatie heeft de volgende specifieke regels voor de rooster module waarin het gebruikt wordt:

- Periodes die opgevraagd kunnen worden voor het initialiseren van ee rooster zijn: periodes die een prognose status "Finalized" hebben en de winkel rooster is nog niet afgerond.
- De standaard periode voor de te initialiseren rooster is:
    - De eerst gevonden periode welke een prognose van "Finalized" heeft en een roosterstatus van "New" of "Concept"
    - Als deze er niet is dan de eerste periode waarvan de prognose "Finalized" is en de rooster status "Initialized"
    - Als er geen valide periodes zijn is de lijst met periodes en de standaard periode leeg
    - Als de standaard of gekozen week de status "Initializing" heeft dan is de select knop "INACTIVE"
- Periodes die gevraagd kunnen worden voor afdelingsroosters zijn: alle periodes tussen de start en einddatum van de winkel
- De standaard periode voor afdelingsroosters zijn:
    - De huidige periode
    - Als die niet gevonden wordt, dan de eerste week van de winkel waarvan "SHP002 SchedulingStartDate" in de toekomst ligt
    - Als die niet gevonden wordt, dan de laatste week van de winkel waarvan ShopEndDate in het verleden ligt
    - Als de gekozen of standaard week de status "Initializing" heeft dan is de select knop "INACTIVE"
- Periodes die opgevraagd kunnen worden om het rooster af te ronden zijn: periodes waarvan het winkelrooster status "INITIALIZED" of "FINAL" is.
- De standaard periode voor het afronden van een rooster is:
    - De eerste periode met als winkel rooster status "INITIALIZED"
    - Als deze niet gevonden wordt dan de laatste periode met winkelroosterstatus "FINAL"
    - Als er geen valide periodes zijn dan is de lijst met periodes en de standaard periode leeg.

Pas regelementen toe op rooster:
Deze feature past een aantal regelementen toe in een vaste volgorde, waarna het de resultaten volhard en de arbeidskosten berekend.

De volgende stappen worden genomen in deze use case:

- Controleer beschikbaarheid
- Past de geconfigureerde pauze profiel toe
- Voer de WHA uit(working hours act)

- Voer de labor agreement uit(arbeidsovereenkomst)
- Als het is ingeschakeld, volhard de logging welke terugkomt van het arbeidsovereenkomst
- Bereken arbeidskosten
- Als het is ingeschakeld, voer maximale arbeidskosten per maand uit
- Volhard indicatieve balans mutaties
- Volhard activiteiten
- Volhard salarissen
- Volhard alle waarschuwingen en error's

Als de uitvoering van de arbeidsovereenkomst resulteert in een functionele exception, wordt de data die hiervan terugkomt niet volhard voor de medewerker die de exception veroorzaakte. Een functionele error zal opnieuw worden geregistreerd. Nadat het probleem is opgelost, moet de use case opnieuw uitgevoerd worden om een consistente staat te behouden.

Rond winkelrooster af:
Een winkelrooster kan alleen afgerond worden als:
- De status ervan "initialised" is
- De prognose van de winkel voor de corresponderende week "finalized" is

Als de winkelrooster is afgerond, zijn alle afdelingen ook afgerond en worden alle activiteiten gekopieerd naar een actueel rooster.

Neem medewerker van een ander filiaal aan in het rooster:
Deze functionaliteit is deel van: beheer medewerkers activiteiten en beschrijft de functionaliteit als een medewerker is ingeroosterd in een andere winkel

Initialiseer winkelrooster:
Deze functionaliteit resulteert in een versie van het rooster waar activiteiten geroosterd kunnen worden voor medewerkers. Het winkelrooster kan (optioneel) ook geïnitialiseerd worden met een template rooster. Initialisatie kan alleen plaatsvinden als het rooster nog niet geïnitialiseerd is.

Initialisatie bestaat uit:
1. Optioneel. Selecteer het template rooster en het zal gebruikt worden als bron van de initialisatie
2. Maak de initiële rooster activiteiten
3. Als de initialisatie gedaan is met een template rooster worden alle activiteiten van deze template gekopieerd naar het initiële rooster.
4. Pas regelementen toe op het rooster op de aangemaakte activiteiten

Na het initialiseren van het rooster wordt de status van het rooster gezet op "Initialized".

Beheer afdelingsrooster:
Om het rooster te kunnen beheren voor 1 of meerdere afdelingen, krijgt de gebruiker informatie per dag, van alle medewerkers behorend bij de afdeling en alle medewerkers die gehuurd zijn en tenminste 1 dienst gedraaid hebben op een afdeling te zien. De informatie is over geroosterde activiteiten, template rooster activiteiten, rooster arbeidskosten, beschikbaarheid en berichten.

Roosters kunnen de volgende statussen hebben: "Initieel", "Finalized" en "Actual".

Haal prognose informatie op:

Deze feature zorgt ervoor dat prognose informatie die nodig is om een winkelrooster op te halen in beheer afdelingsrooster. De benodigde informatie gaat over: omzet, budget uren, afdelingstaken en budget arbeidskosten.

Beheer activiteiten van medewerkers:
Vanaf de beheer afdelingsrooster kunnen activiteiten van medewerkers beheerd worden door deze toe te voegen, te wijzigen en/of te verwijderen. Nadat de activiteiten voor een of meer dagen gewijzigd zijn, worden deze gerund door "Apply regulations to schedule" gecombineerd met de rest van de activiteiten voor de week.

- Activiteiten zijn beperkt tot een tijdsframe van 00:00 tot 23:59 op 1 dag. Activiteiten kunnen momenteel nog niet over meerdere dagen verspreid worden
- Om data op te slaan moet de gebruiker lees en schrijfrechten hebben op de rooster activiteiten resource.

Beheer template van activiteiten van medewerkers:

-

Beheer schoolroosters:
Om schoolroosters van medewerkers in een afdeling te beheren, worden schoolroosters opgehaald. Dit wordt gedaan zolang de einddatum niet verstreken is. Alleen schoolroosters met de status "Pending" en waarvan de einddatum nog niet verstreken is kunnen verwerkt of verwijderd worden.

Beheer template van afdelingsrooster:

-

Overzicht schoolroosters:
Deze feature gaat over het ophalen en tonen van de opgehaalde schoolroosters.

Winkelrooster:
Om het rooster van de medewerkers in een afdeling te beheren wordt de gebruiker de volgende informatie per dag getoond:

- Totale actuele ingeroosterde tijd voor alle productieve activiteiten
- Alle actieve medewerkers behorend tot de afdelingen
- Medewerkers die zijn ingehuurd en ten minste 1 dienst hebben bij een afdeling
- Elke medewerker heeft een lijst van actueel geroosterde activiteiten.

Medewerker schoolrooster:
Schoolroosters worden gebruikt om de manager te informeren dat de medewerker naar school is, en dus niet kan komen werken. Met de schoolrooster kan de manager controleren of de werkuren van de medewerker het maximaal toegestane hoeveelheid werkuren volgens de WHA overschrijdt.

Beheer medewerkersbeschikbaarheid per periode:
Alle beschikbaarheden van medewerkers kunnen beheert worden voor specifieke periodes. Door deze te wijzigen voor een periode wordt de standaard beschikbaarheid niet gewijzigd.

- Om data op te slaan moet de gebruiker lees en schrijfrechten hebben op de rooster activiteiten resource.

E.3 Realisatie

UC020:

Pre-condities: de winkelmanager heeft een winkel en week geselecteerd

1. Het systeem toont het scherm 'Shop realization' met de volgende informatie:
    a. Voor elke dag van de geselecteerde week de geplande hoeveelheid aan uren en de geklokte hoeveelheid uren(indien deze er zijn)
    b. Een lijst met afdelingen die actief zijn in de organisatiestructuur van de geselecteerde winkel in de geselecteerde week
2. De winkelmanager voert een van de volgende acties uit:
    a. Initialiseren winkel realisatie: het systeem start subflow: 'Initialize shop realization'
    b. Beheer winkel realisatie: het systeem start subflow: 'Manage shop realization'
    c. Afronden winkel realizatie: het systeem start subflow: 'Finalize shop realization'

Subflow: initialize shop realization

De subflow start als de winkelmanager de realisatie van winkeluren wilt initialiseren.

1. De winkelmanager selecteert een realisatie bron voor elke dag die hij wilt initialiseren
2. De winkel manager geeft aan deze dagen te willen initialiseren
3. Het systeem initialiseert de geselecteerde dagen. Klok tijden die zijn geklokt op inactieve afdelingen worden opgeslagen met de status 'Not processed'

Subflow: manage shop realization:

De subflow start als de winkelmanager de realisatie uren wilt beheren.

1. Het systeem toont het scherm 'Manage shop realization' met alle afdelingsmedewerkers
2. De winkelmanager selecteert een medewerker en geeft aan de uren realisatie van deze medewerker te beheren
3. Het systeem toont het scherm 'Manage employee realization' met de diensten van de geselecteerde medewerker in de geselecteerde week
4. De winkelmanager wijzigt de bestaande diensten of voegt een nieuwe dienst toe. De volgende data kan gewijzigd worden: start/eindtijd, taken, winkel, afdeling
5. De winkelmanager geeft aan de gewijzigde of toegevoegde dienst op te slaan
6. Het systeem slaat de dienst op

Toepassen regelementen:

Deze functionaliteit voert optioneel de volgende regelementen uit:

- boundaryPolicy
- AutoApprovePolicy
- LaborAgreement

En verwerkt de resultaten. Het zorgt ervoor de voor een set aan uren de volgende componenten in volgorde worden uitgevoerd:

- Uitvoeren Boundary Policy indien nodig
- Uitvoeren Auto Approve Policy indien nodig
- Aanmaken tijdelijk aanwezigheidspatroon, als nodig
- Uitvoeren Labor Agreement indien nodig
- Volharden van de logging van Labor Agreement(gemarkeerd als realisatie)
- Uitvoeren Assign Week Rounding To Department
- Volharden van balans mutaties
- Volharden van activiteiten
- Volharden van salarissen
- Volharden van waarschuwingen en error's van Laboragreement

- Uitvoeren van Labor Costs

De arbeidsovereenkomst moet altijd uitgevoerd worden met een volledige week als input. Dit betekent dat de gerealiseerde dagen aangevuld moeten worden met de geroosterde uren voor de rest van de week. De output hiervan moet alleen volhard worden voor de gerealiseerde dagen.

Als de totale uren van het aanwezigheidspatroon van de medewerker kleiner zijn dan de contract uren van het contract van de laatste werkdag, wordt er een specifieke realisatie aanwezigheidspatroon aangemaakt. Dit wordt voor die specifieke week gedaan en begint als volgt: 8 uur per dag, beginnend op maandag tot alle contract uren zijn verdeeld over 1 of meerdere weekdagen.

Als de contract uren meer dan 56 uur per week zijn, wordt de LaborAgreement niet uitgevoerd en wordt er een error gegeven. Andere regelementen kunnen toegepast worden op elk hoeveelheid uren.

Als de arbeidsovereenkomst resulteert in een functionele error, worden alleen de nieuwe activiteiten volhard, de data van de medewerker zal niet volhard worden en een functionele error zal gegeven worden.

Alleen de indicatieve balans mutaties van het type "Increase" worden volhard.

Toekennen weekrounding op afdelingen:
Weekroundings worden bepaald door de geregistreerde uren van gerealiseerde dagen, en aangevuld met de geroosterde uren voor de rest van de week.

Uren die terugkomen van LaborAgreement en geclassificeerd zijn als weekRounding worden opgeslagen op de winkel en afdeling waarvan de meeste uren geregistreerd zijn voor de medewerker in een week. Als er evenveel gewerkte uren voor meerdere afdelingen zijn wordt de weekrounding toegekend aan de afdeling waar de medewerker als laatst gewerkt heeft.

Bereken klanten op checkout normclusters: (ignore tag)
Deze functie berekent de hoeveelheid klanten op de checkout normcluster. Dit wordt per dag gedaan en per uur.

De reden dat dit berekent wordt is dat als een winkel meer verkooppunten heeft, het niet altijd mogelijk is de hoeveelheid klanten per verkooppunt te leveren. Alleen de totale hoeveelheid klanten van de winkel wordt dan geleverd. De berekening voor de checkout normcluster is de totale hoeveelheid klanten van de winkel min de maximum hoeveelheid klanten, van alle verkooppunten.

De aanname is dat in de winkel alleen 1 normcluster als checkout heeft. Daardoor kunnen klanten die bij 1 checkout geweest zijn, niet ook bij een andere voorkomen.

Afronden afdelingsrealisatie:
Als alle gerealiseerde dagen voor een afdeling zijn geinitialiseerd en goedgekeurd, en er zijn geen open ziektes, kan de afdelingsrealisatie afgerond worden. Als alle afdelingen zijn afgerond, wordt de winkelrealisatie automatisch afgerond.

Initialiseren winkelrealisatie:
Dit kan gedaan worden op basis van kloktijden of gebaseerd worden op geroosterde activiteiten. Sommige medewerkers hebben een setting dat zij altijd geinitialiseerd moeten worden op basis van het rooster, zelfs al is de dag gebaseerd op kloktijden.

Als de gebruiker wilt initialiseren op een gesloten dag, zal het systeem automatisch initialiseren gebaseerd op kloktijden, tenzij de klok geen kloksysteem heeft. In dat geval wordt er gebaseerd op rooster.

De initialisatie gebaseerd op kloktijden bestaat uit:

1. Selecteer wat te initialiseren van de bron
2. Ga alle medewerkers langs die actief in de winkel waren die dag
3. Neem alle kloktijden met badges die horen bij deze medewerkers
4. Maak de initiële realisatie activiteiten aan vanuit de kloktijden.
5. Opvolgende kloktijden op dezelfde afdeling worden samengevoegd
6. Pauzes worden alleen geinitialiseerd als er daarvoor een productieve activiteit heeft plaatsgevonden
7. Als de property "SHP011 ProductiveTimeRegistration" van de winkel de winkels kloktijden is ingeschakeld dan worden de kloktijden op de standaard afdeling van de medewerker gezet
8. De regelementen worden toegepast

De initialisatie gebaseerd op geroosterde activiteiten bestaat uit:

1. Selecteer wat te initialiseren van de bron
2. Alle activiteiten van medewerkers die actief in de winkel waren die dag
3. Maak de initiële realisatie activiteiten aan vanuit de geroosterde activiteiten.
4. Includeer alleen activiteiten die als standaard, pauze of betaalde pauze geclassificeerd zijn
5. De regelementen worden toegepast

Beheren medewerkers activiteiten:
Om de activiteiten van de medewerkers te beheren krijgt de gebruiker een lijst van alle geregistreerde activiteiten voor die week te zien. Na het bewerken en opslaan van de geregistreerde activiteiten voor een dag, worden ze geleverd aan de "Toepassen regelementen". De gebruiker kan ook kiezen om een werkdag goed te keuren, wat betekend dat alle activiteiten van die dag ook worden opgeslagen.

Beheren medewerkersrealisatie:
De realisatie van een medewerker kan per geinitialiseerde dag gedaan worden. beheren van de realisatie door de wijzigen, toevoegen of te verwijderen van de start –en eindtijd of duur van de pauze betekent dat de duur van de activiteiten per winkel en afdeling en het moment van pauze niet relevant zijn.

Verwerken terminal tijden:
Door gebruik te maken van een kloksysteem, kan er precies gekeken worden wanneer tot wanneer een medewerker op een afdeling geweest is. Dit wordt geregistreerd als kloktijden. Er zijn 4 soorten terminals, namelijk:

- In, hier wordt geregistreerd als de medewerker de winkel binnenkomt
- Shop, functioneel hetzelfde als In
- Department, de medewerker registreert zich hier als hij begint te werken op de afdeling
- Out, het uitchecken voor vertrek.

Pauze wordt als een gewone afdeling gezien.

De gewerkte uren worden als volgt berekent:

- De tijd tussen het inchecken bij de eerste afdeling voor de dag, en het uitchecken bij out aan het eind van de dienst.

Als een gebruiker 2 of meer keer incheckt in dezelfde minuut bij dezelfde terminal wordt alleen 1 terminaltijd verwerkt. De rest wordt gezien als dubbel.
Als een gebruiker 2 of meer keer incheckt bij verschillende terminals binnen dezelfde minuut, worden beide terminals verwerkt.

Winkel realisatie:
Om de realisatie van de medewerkers in een afdeling te beheren wordt de volgende informatie per dag getoond:

- Totale kloktijden, opgesomd voor de afdeling/winkel
- Alle actieve medewerkers behorend tot een afdeling
- Medewerkers die ingehuurd zijn en ten minste 1 dienst op een afdeling gewerkt hebben
- Per dag:
    - Elke afdeling heeft de geinitialiseerde bron(kloktijden, rooster of gesloten)
    - Alle medewerker heeft een lijst met geregistreerde activiteiten
    - Alle medewerkers hebben een waarde wat aangeeft of alle uren volgens de business rules zijn gegaan, omtrent start –en eindtijd
    - Elke medewerker heeft een goedkeurende staat voor realisatie
    - Waardes voor of de dag een feestdag is, een notificatie word gegeven voor feestdagen
    - Per winkel de werktijden
- Per week:
    - De status per winkel
    - De status per afdeling
    - De laatst geinitialiseerde dag per afdeling
    - Of de winkel voor de laatste week de status “Finalized” heeft
    - Of er een functioneel bericht/error voor een medewerker behorend tot de afdeling is.

E.4 Uitbetaling

UC023:
De use case start als de winkelmanager aangeeft dat hij de betaalbare items en loonitems wilt exporteren.

1. Het systeem toont het scherm ‘Export data definition’
2. Het systeem bepaalt welke winkel, periode, definitie van specifieke arbeidskosten, uren, vergoedingen en/of social security dagen klaar zijn voor export, bezig zijn met exporteren of klaar zijn met exporteren
3. Als de export klaar is registreert het systeem de duur van de export
4. De gebruiker geeft aan dat het systeem de geselecteerde export definities met exporteren
5. Het systeem exporteert de specifieke arbeidskosten, uren, vergoedingen en/of social security dagen voor de gespecificeerde winkel voor de geselecteerde periode naar een extern systeem en maakt deze beschikbaar om te downloaden.
6. Het systeem zet de status van de export naar in progress

Corrigeren medewerkers verlofbalans:
Om het verlofbalans te corrigeren per verloftype van een medewerker in een week, kan de gebruiker het verlofbalans verhogen, verlagen of uitbetalen voor die week. Alleen verloftypes met het type "(TAK_LeaveRightType) BuildUp" en "Compensation" kunnen gecorrigeerd worden.

De correcties worden opgeslagen op de eerste werkdag van de week en worden behouden. Het is ook mogelijk om correcties aan het verlofbalans te verwijderen.

E.5 Overige functionaliteiten

Naast de functionaliteiten uit de hoofdmodules, beschreven in hoofdstuk 5.1 t/m 5.6, zijn er een aantal functionaliteiten die in meerdere modules gebruikt worden. deze staan hieronder per categorie beschreven.

E.5.1 Calculation components

E.5.1.1 Historie

PersistWaitDayHistory:
Als er een nieuwe wachtdag voor een medewerker is wordt dit ingevoerd in de wachtdagen historie. De historische wachtdag data bestaat uit:

- Jaar
- Het ziekterapport(geïdentificeerd door begin en einddatum)
- De hoeveelheid wachtdagen toegekend per type voor een medewerker in deze week voor het ziekterapport

Voor lonen gespecificeerd in de instelling: "SYS026 Remuneration type Wait day employer", de instelling "SYS027 Remuneration type Wait day not to pay" en de instelling "SYS028 Remuneration type Wait day reduce on leave" wordt een "NrOfWaitDaysEmployer", "NrOfWaitDaysNotToPay" of "NrOfWaitdaysReduceOnLeave" toegevoegd aan de wachtdagen historie. Als de reden van de huidige ziekte van het type potentiele wachtdag is, dan wordt het ziekterapport toegevoegd aan de wachtdagen historie.

Ophalen historisch gewerkte uren:
De historisch gewerkte uren en toegestane overuren voor een medewerker worden opgehaald van afgeronde betalingen of afgeronde roosters als de betaling nog niet afgerond is.

Gewerkte uren: uren worden geteld als gewerkt als zij geregistreerd zijn op een activiteit waarvan "CountInContractHours" op true is gezet, met als uitzondering geregistreerde uren op de activiteiten ziekte, verlof en feestdagen. Geregistreerde uren op de activiteit "PausePayment" worden alleen geteld worden als de instelling "LAT054: PausePaymentAreWorkedHours" ook op true is gezet.

Ophalen wachtdagen historie:
Deze feature haalt de wachtdagen historie informatie op welke gebruikt kan worden in de wachtdagen feature. De input van het ophalen is: de medewerker en de kalenderweek.

Het resultaat bestaat uit: de totale hoeveelheid wachtdagen toegekend per type voor een medewerker tot deze week per ziekte rapport in het huidige kalenderjaar, de totale hoeveelheid ziekmeldingen in het kalenderjaar waarvoor de wachtdag is gegeven en de einddatum van de vorige ziekte van het type potentiele wachtdag.

E.5.1.2 Pauze profielen

Pauze profiel:

Per mederwerker, per dag worden de pauzes bepaald worden per geregistreerde dienst. Voor elke pauze wordt het volgende teruggegeven:

- "ProfilePauseRelativeStartTime"
- "ProfilePauseDuration"
- "ProfilePaid"

De pauzes per medewerker per dag zijn afhankelijk van:

- Het contracttype van de medewerker
- Het pauze profiel die op die dag in de winkel gold
- De dag waarop de dienst is geregistreerd
- De starttijd en duur van de geregistreerde dienst

De pauzes worden bepaald in de "PauseProfile" wanneer er aan beide condities in de volgende volgorde is voldaan:

- De laatste "StartTimeShift" voor de gelijk aan de starttijd van de geregistreerde dienst
- De hoogste "ProfileDurationShift" kleiner of gelijk aan de duur van de geregistreerde shift.

Als zowel de start als de duur van de pauze 0 is, wordt de pauze genegeerd.

E.5.1.3 Geregistreerde uren

Boundary policy:

Deze feature is optioneel in de realisatie en optioneel in de uitbetaling.
Alle geklokte uren per medewerker per dag zullen verwerkt worden als gerealiseerde uren nadat deze policy eroverheen is gegaan. De ingeklokte tijden en de uitgeklokte tijden zullen in dat geval aangepast worden aan de hand van deze policy.
Bijvoorbeeld: als een medewerker 8.55 begint met werken en 10.12 weggaat, en in de policy staat dat er per 15 min wordt afgerond, dan zullen de werktijden zijn: 9 tot 10.

Pause policy:

Gebaseerd op geregistreerde uren en een instelling om de pauze profiel tijden te gebruiken, worden de geregistreerde pauzes verwijderd en nieuwe pauzes toegevoegd aan de dienst.

Als de totale hoeveelheid pauzes per dag gelijk is aan de totale toegepaste pauze profiel tijd, dan wordt er geen actie ondernomen. Als een pauze profiel tijd zodanig is verspreid dat het niet aan het einde van een dienst past, dan wordt deze verdeeld tussen meerdere diensten.

Validate business rules:

een dag is volgens business rules wanneer het verschil tussen het begin en de eindtijd, en de totale duur van de pauze van de geregistreerde activiteiten binnen een gespecificeerde magin van het begin en de eindtijd is. Margins kunnen gespecificeerd worden in de settings.

E.5.1.4 Settings

Vinden van een organisatie instelling:
Het is mogelijk om een enkele waarde voor een organisatie instelling op te halen, door de organisatie en datum op te geven.

Als er gezocht wordt naar de waarde van een benoemde instelling moet het altijd opgehaald worden van de laagste organisatielevel waarop het is gedefinieerd. Terwijl er rekening gehouden wordt met de validiteit.

Vinden van instellingen:
Het is mogelijk om een enkele waarde van een benoemde instelling op te halen door de volgende selectie criteria op te geven bij zoeken.

De waarde kan gedefinieerd worden op 5 levels:

1. Medewerker
2. Organisatie
3. Contracttype
4. Arbeidsovereenkomst
5. Omgeving

Als er gezocht wordt op de waarde van een benoemde instelling, moet het altijd van de onderste level gehaald worden, waarop het benoemd is(medewerker als laagst, omgeving als hoogst).

E.5.2 WHA

Check geregistreerde uren tegen toegestane werktijden:
Deze regel controleert de werktijden van een medewerker. Afhankelijk van de configuratie, zal deze regel een waarschuwing of een error geven als de medewerker buiten de toegestane werktijden heeft gewerkt.

Check geregistreerde uren tegen maximaal toegestane werkdagen per week:
Deze regel kijkt of de geregistreerde activiteiten van een medewerker de maximaal toegestane werkdagen overschrijd. Afhankelijk van de configuratie, zal deze regel een waarschuwing of een error geven als de meer dagen heeft gewerkt dan is toegestaan.

Check geregistreerde uren tegen maximaal toegestane werkuren per dag:
Deze regel kijkt of de gewerkte uren van de medewerker de maximaal toegestane uren voor een dag overschrijd. Afhankelijk van de configuratie, zal deze regel een waarschuwing of een error geven als de medewerker meer uren heeft gewerkt dan hij mag werken op een dag.

Check geregistreerde uren tegen maximaal toegestane werkuren per week:
Deze regel kijkt of de gewerkte uren van de medewerker de maximaal toegestane uren voor een week overschrijd. Afhankelijk van de configuratie, zal deze regel een waarschuwing of een error geven als de medewerker meer uren heeft gewerkt dan hij mag werken in een week.

Check geregistreerde uren tegen minimum hoeveelheid rusturen die nodig zijn tussen 2 werkdagen:
Deze regel kijkt of de geregistreerde uren van de medewerker voldoen aan de minimale hoeveelheid rust die tussen 2 werkdagen moet zitten. Afhankelijk van de configuratie, zal

deze regel een waarschuwing of een error geven als de medewerker minder uren rust heeft dan hij moet hebben.

Check of minimale ononderbroken rustperiode binnen een week behaald is:
Deze regel kijkt of de minimale vereiste ononderbroken rustperiode binnen een week behaald is. . Afhankelijk van de configuratie, zal deze regel een waarschuwing of een error geven als de medewerker minder uren ononderbroken rust heeft gehad dan verplicht is.

E.5.3 ManagementInterface

Adapter betaaldata:
Deze feature beschrijft een adapter voor de management interface van betaaldata. De adapter transformeert de data van R&R naar het externe systeem. in de export worden medewerkers die hebben gewerkt in acht genomen. Medewerkers die in andere winkels gewerkt hebben worden daar gezet.
Adapter roosterdata:
Deze feature beschrijft een adapter voor de management interface van roosterdata. De adapter transformeert de data van R&R naar het externe systeem.
In de export worden alleen medewerkers die gewerkt hebben in de winkel geëxporteerd. Medewerkers die in andere winkels gewerkt hebben worden in die winkel gezet. Dit geld voor winkels die in een pool zitten. Eigen medewerkers die hebben gewerkt voor een winkel buiten de pool worden wel meegenomen in deze export

Handler betaaldata:
De R&R management interface exporteert data naar externe systemen. De trigger voor de export is of een afgeronde betalingsweek of een her-export van R&R. de periode type voor de betalingsdata is altijd een week.

Handler roosterdata:
De R&R management interface exporteert roosterdata naar externe systemen. De trigger is of een afgeronde roosterweek of een her-export van R&R.

E.5.4 Organisation

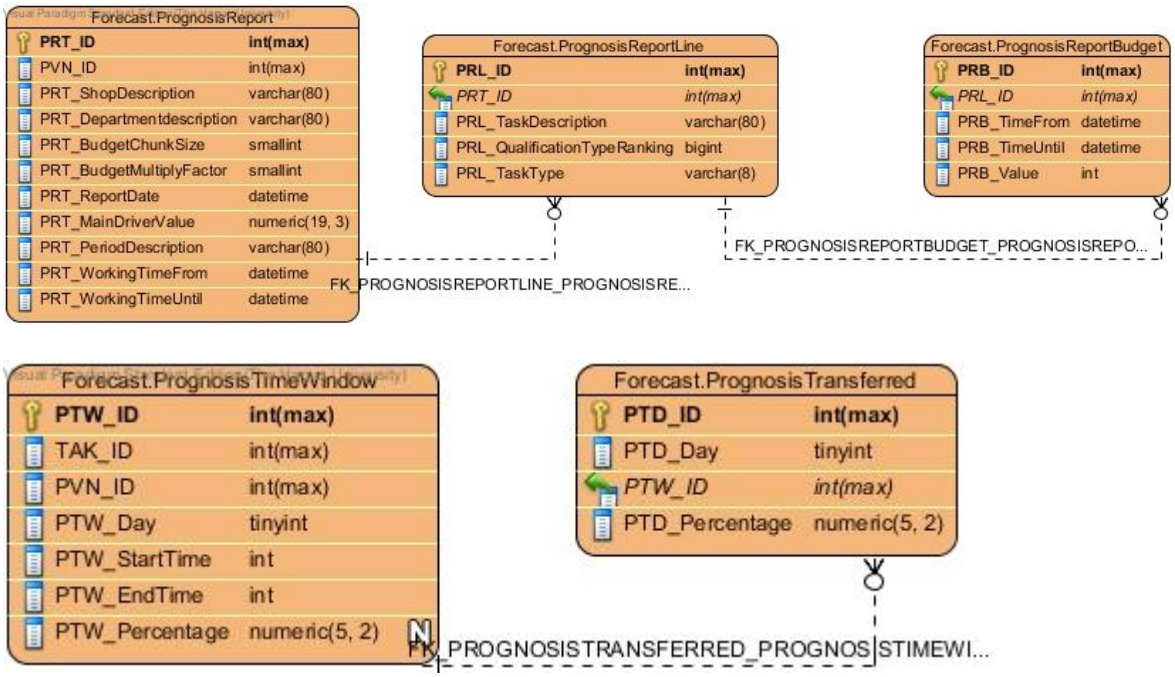
Verstuur communicatie bericht:
Deze feature levert de controles om een bericht te sturen. Voor een manager is het mogelijk om een email naar alle medewerkers te sturen, die momenteel en in de toekomst nog in dienst zijn. Deze medewerkers moeten als standaardafdeling de geselecteerde afdeling hebben.

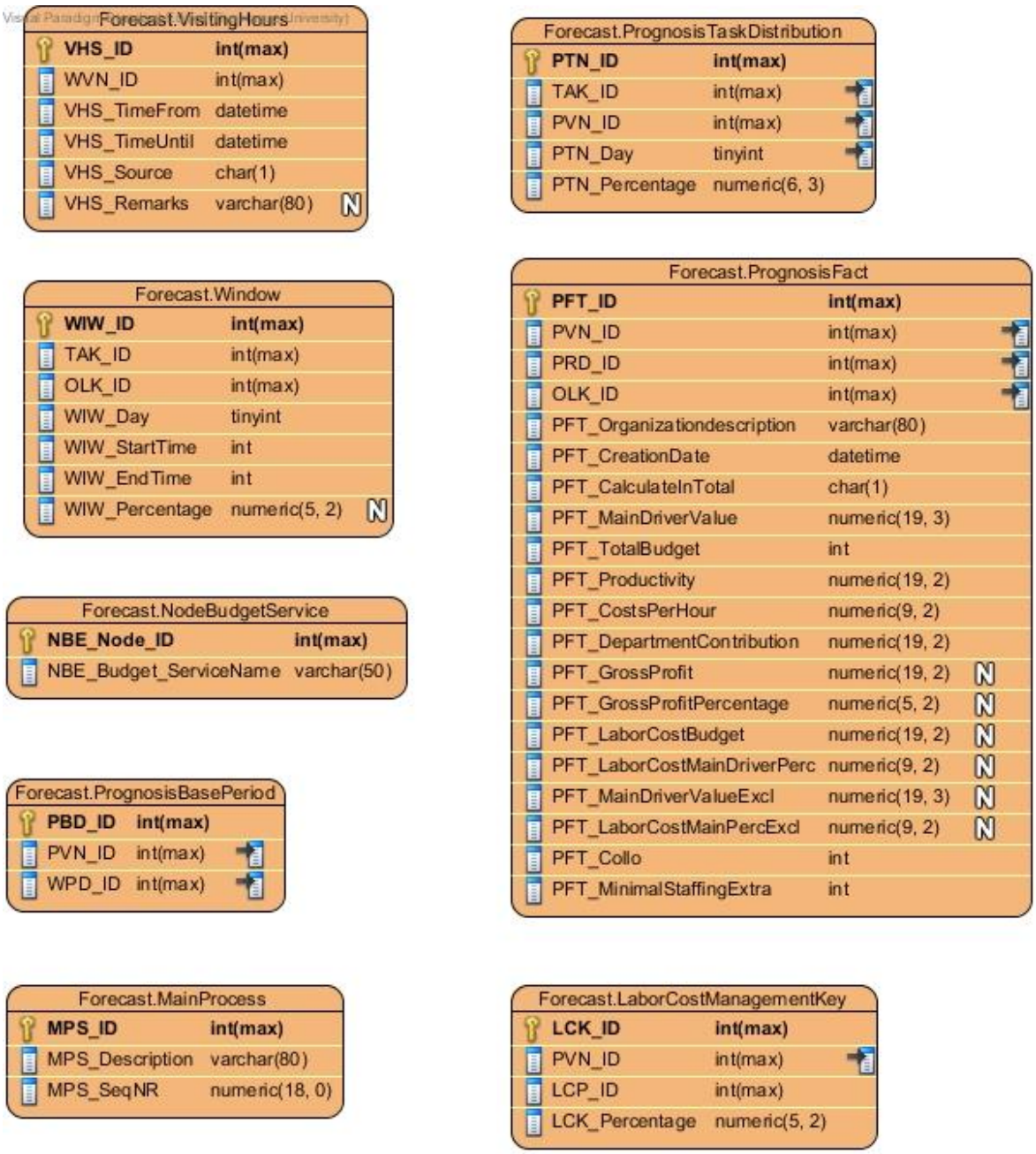
Bijlage F: Technische diagrammen

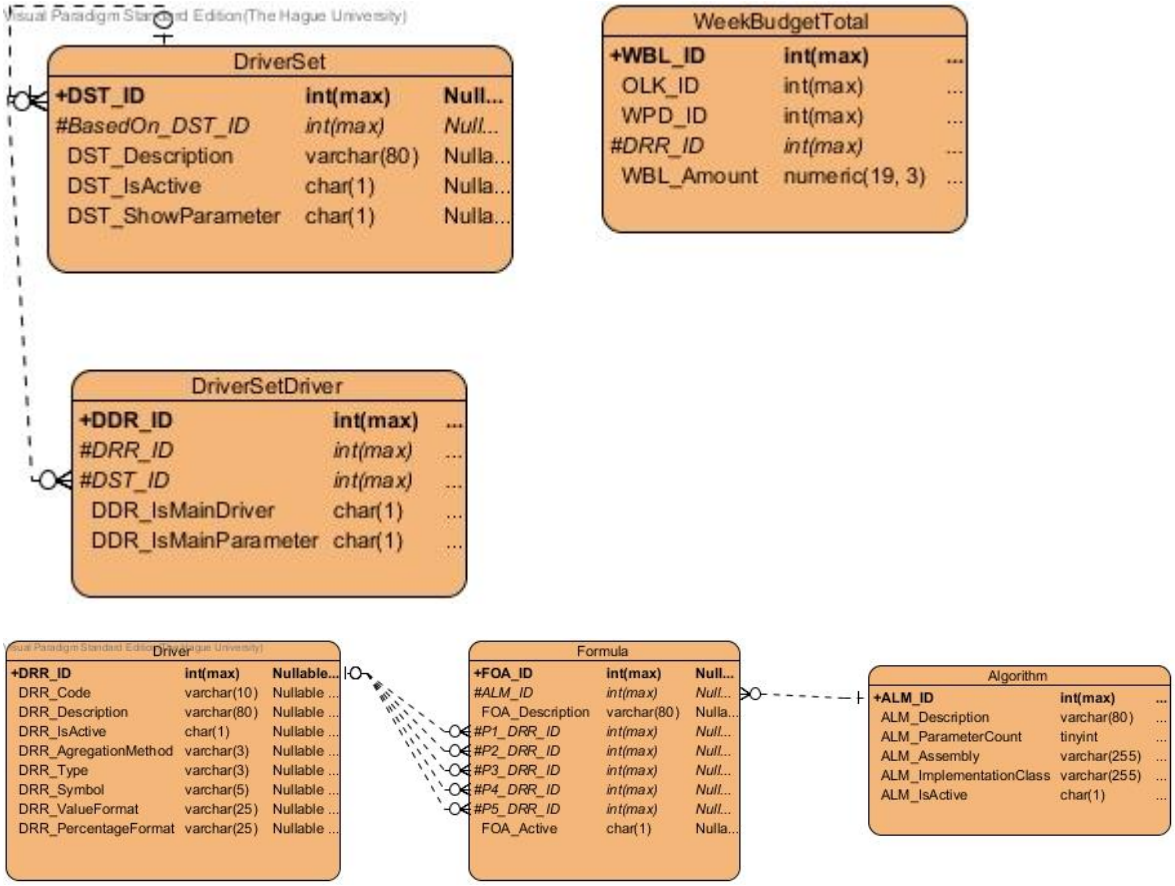
Er zijn ook een aantal diagrammen reverse engineered uit de database. Hierbij zijn in een database, waar ik toegang toe heb gekregen door de opdrachtgever, de tabellen eerst als scripts gegenereerd op basis van hun prefix (realisation, schedule enz.). Dit is gedaan met behulp van de generate script functionaliteit uit SQL-Server. De scripts zijn opgeslagen als ANSI-tekst in een .sql file (1 file per prefix). Vervolgens zijn de scripts in Visual Paradigm omgezet naar ERD-diagrammen. Het enige wat ik aan de diagrammen heb gewijzigd is de layout, en af en toe het diagram opgedeeld in kleine sub diagrammen. Deze diagrammen zijn te vinden in bijlage F: Technische diagrammen.

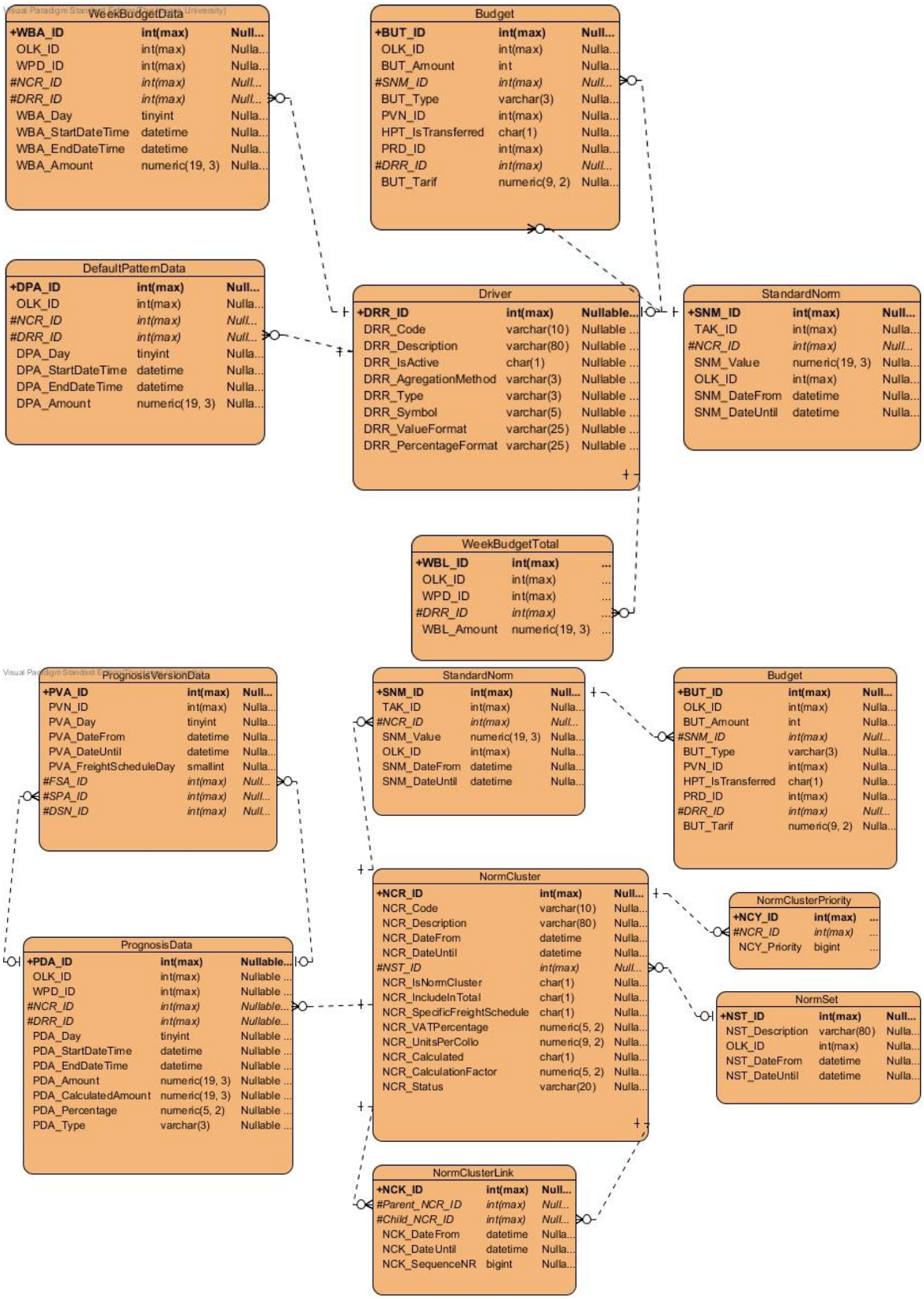
F.1 Prognose

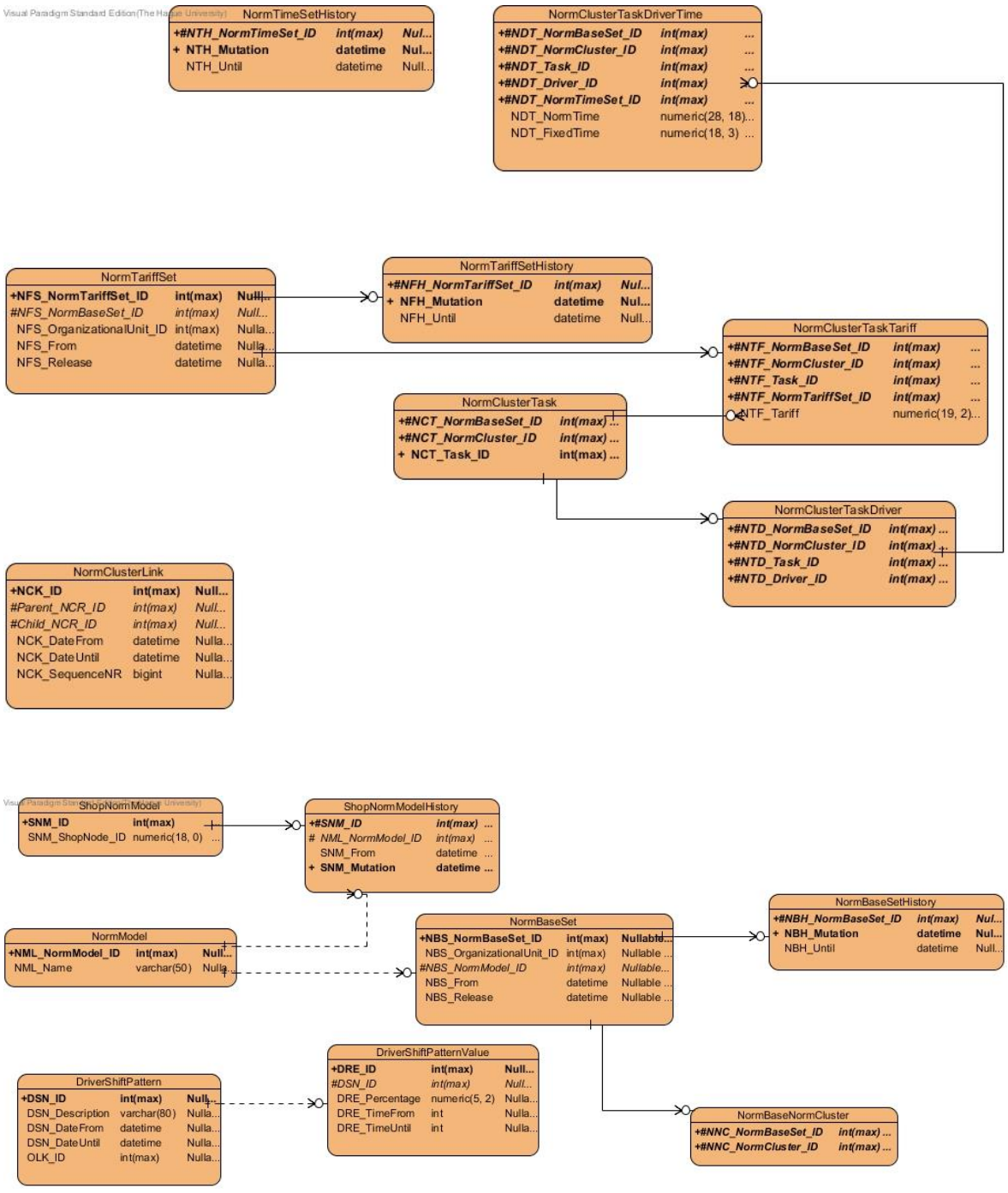
De tabellen uit de prognose module hebben als prefix: "Forecast."




Forecast.VisitingHours
VHS_ID int(max)
WVN_ID int(max)
VHS_TimeFrom datetime
VHS_TimeUntil datetime
VHS_Source char(1)
VHS_Remarks varchar(80) N
Forecast.PrognosisTaskDistribution
PTN_ID int(max)
TAK_ID int(max)
PVN_ID int(max)
PTN_Day tinyint
PTN_Percentage numeric(6, 3)
Forecast.Window
WIW_ID int(max)
TAK_ID int(max)
OLK_ID int(max)
WIW_Day tinyint
WIW_StartTime int
WIW_EndTime int
WIW_Percentage numeric(5, 2) N
Forecast.PrognosisFact
PFT_ID int(max)
PVN_ID int(max)
PRD_ID int(max)
OLK_ID int(max)
PFT_Organizationdescription varchar(80)
PFT_CreationDate datetime
PFT_CalculateInTotal char(1)
PFT_MainDriverValue numeric(19, 3)
PFT_TotalBudget int
PFT_Productivity numeric(19, 2)
PFT_CostsPerHour numeric(9, 2)
PFT_DepartmentContribution numeric(19, 2)
PFT_GrossProfit numeric(19, 2) N
PFT_GrossProfitPercentage numeric(5, 2) N
PFT_LaborCostBudget numeric(19, 2) N
PFT_LaborCostMainDriverPerc numeric(9, 2) N
PFT_MainDriverValueExcl numeric(19, 3) N
PFT_LaborCostMainPercExcl numeric(9, 2) N
PFT_Collo int
PFT_MinimalStaffingExtra int
Forecast.NodeBudgetService
NBE_Node_ID int(max)
NBE_Budget_ServiceName varchar(50)
Forecast.PrognosisBasePeriod
PBD_ID int(max)
PVN_ID int(max)
WPD_ID int(max)
Forecast.MainProcess
MPS_ID int(max)
MPS_Description varchar(80)
MPS_SeqNR numeric(18, 0)
Forecast.LaborCostManagementKey
LCK_ID int(max)
PVN_ID int(max)
LCP_ID int(max)
LCK_Percentage numeric(5, 2)

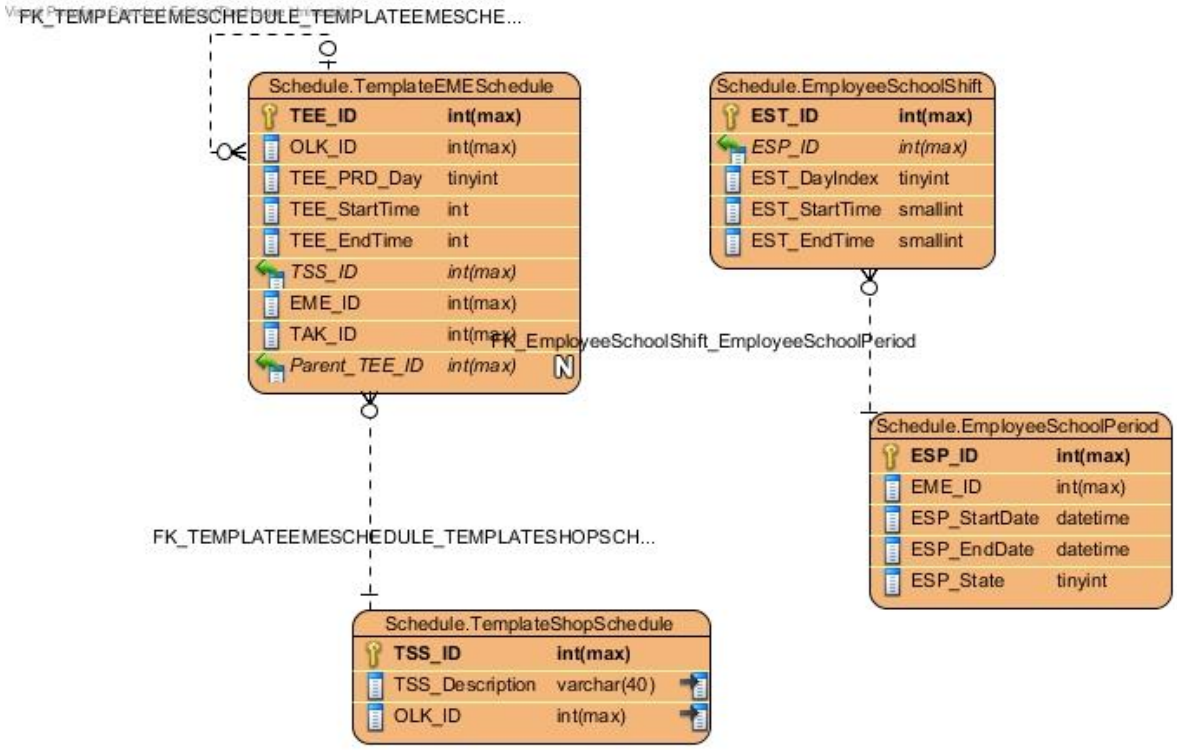

DriverSet
+DST_ID int(max) Null...
#BasedOn_DST_ID int(max) Null...
DST_Description varchar(80) Nulla...
DST_IsActive char(1) Nulla...
DST_ShowParameter char(1) Nulla...
WeekBudgetTotal
+WBL_ID int(max) ...
OLK_ID int(max) ...
WPD_ID int(max) ...
#DRR_ID int(max) ...
WBL_Amount numeric(19, 3) ...
DriverSetDriver
+DDR_ID int(max) ...
#DRR_ID int(max) ...
#DST_ID int(max) ...
DDR_IsMainDriver char(1) ...
DDR_IsMainParameter char(1) ...
Driver
+DRR_ID int(max) Nullable...
DRR_Code varchar(10) Nullable ...
DRR_Description varchar(80) Nullable ...
DRR_IsActive char(1) Nullable ...
DRR_AgregationMethod varchar(3) Nullable ...
DRR_Type varchar(3) Nullable ...
DRR_Symbol varchar(5) Nullable ...
DRR_ValueFormat varchar(25) Nullable ...
DRR_PercentageFormat varchar(25) Nullable ...
Formula
+FOA_ID int(max) Null...
#ALM_ID int(max) Null...
FOA_Description varchar(80) Nulla...
#P1_DRR_ID int(max) Null...
#P2_DRR_ID int(max) Null...
#P3_DRR_ID int(max) Null...
#P4_DRR_ID int(max) Null...
#P5_DRR_ID int(max) Null...
FOA_Active char(1) Nulla...
Algorithm
+ALM_ID int(max) ...
ALM_Description varchar(80) ...
ALM_ParameterCount tinyint ...
ALM_Assembly varchar(255) ...
ALM_ImplementationClass varchar(255) ...
ALM_IsActive char(1) ...

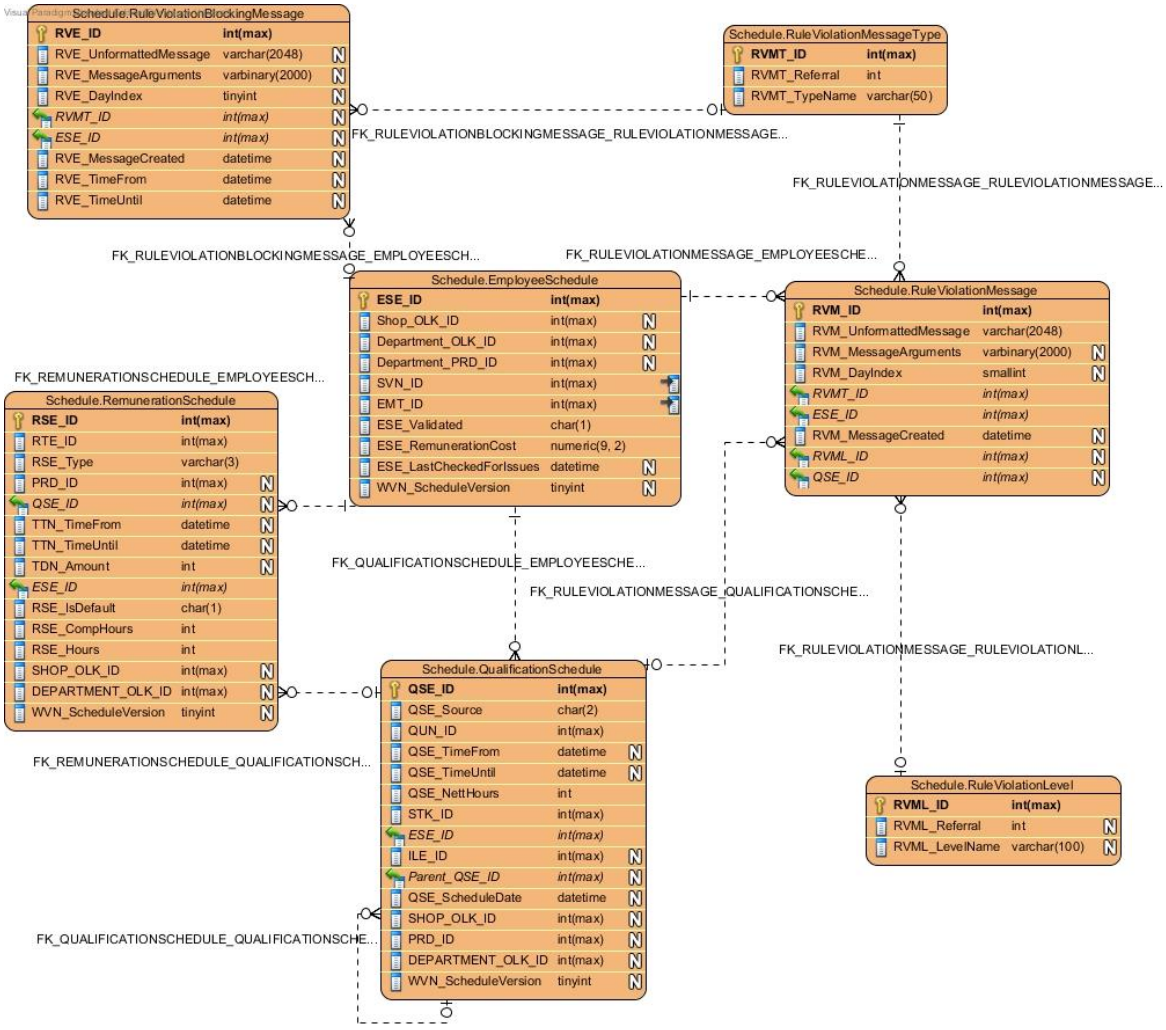

WeekBudgetData
+WBA_ID int(max) Null...
OLK_ID int(max) Nulla...
WPD_ID int(max) Nulla...
#NCR_ID int(max) Null...
#DRR_ID int(max) Null...
WBA_Day tinyint Nulla...
WBA_StartDateTime datetime Nulla...
WBA_EndDateTime datetime Nulla...
WBA_Amount numeric(19, 3) Nulla...
Budget
+BUT_ID int(max) Null...
OLK_ID int(max) Nulla...
BUT_Amount int Nulla...
#SNM_ID int(max) Null...
BUT_Type varchar(3) Nulla...
PVN_ID int(max) Nulla...
HPT_IsTransferred char(1) Nulla...
PRD_ID int(max) Nulla...
#DRR_ID int(max) Null...
BUT_Tarif numeric(9, 2) Nulla...
DefaultPatternData
+DPA_ID int(max) Null...
OLK_ID int(max) Nulla...
#NCR_ID int(max) Null...
#DRR_ID int(max) Null...
DPA_Day tinyint Nulla...
DPA_StartDateTime datetime Nulla...
DPA_EndDateTime datetime Nulla...
DPA_Amount numeric(19, 3) Nulla...
Driver
+DRR_ID int(max) Nullable...
DRR_Code varchar(10) Nullable ...
DRR_Description varchar(80) Nullable ...
DRR_IsActive char(1) Nullable ...
DRR_AgregationMethod varchar(3) Nullable ...
DRR_Type varchar(3) Nullable ...
DRR_Symbol varchar(5) Nullable ...
DRR_ValueFormat varchar(25) Nullable ...
DRR_PercentageFormat varchar(25) Nullable ...
StandardNorm
+SNM_ID int(max) Null...
TAK_ID int(max) Nulla...
#NCR_ID int(max) Null...
SNM_Value numeric(19, 3) Nulla...
OLK_ID int(max) Nulla...
SNM_DateFrom datetime Nulla...
SNM_DateUntil datetime Nulla...
WeekBudgetTotal
+WBL_ID int(max) ...
OLK_ID int(max) ...
WPD_ID int(max) ...
#DRR_ID int(max) ...
WBL_Amount numeric(19, 3) ...
PrognosisVersionData
+PVA_ID int(max) Null...
PVN_ID int(max) Nulla...
PVA_Day tinyint Nulla...
PVA_DateFrom datetime Nulla...
PVA_DateUntil datetime Nulla...
PVA_FreightScheduleDay smallint Nulla...
#FSA_ID int(max) Null...
#SPA_ID int(max) Null...
#DSN_ID int(max) Null...
StandardNorm
+SNM_ID int(max) Null...
TAK_ID int(max) Nulla...
#NCR_ID int(max) Null...
SNM_Value numeric(19, 3) Nulla...
OLK_ID int(max) Nulla...
SNM_DateFrom datetime Nulla...
SNM_DateUntil datetime Nulla...
Budget
+BUT_ID int(max) Null...
OLK_ID int(max) Nulla...
BUT_Amount int Nulla...
#SNM_ID int(max) Null...
BUT_Type varchar(3) Nulla...
PVN_ID int(max) Nulla...
HPT_IsTransferred char(1) Nulla...
PRD_ID int(max) Nulla...
#DRR_ID int(max) Null...
BUT_Tarif numeric(9, 2) Nulla...
NormCluster
+NCR_ID int(max) Null...
NCR_Code varchar(10) Nulla...
NCR_Description varchar(80) Nulla...
NCR_DateFrom datetime Nulla...
NCR_DateUntil datetime Nulla...
#NST_ID int(max) Null...
NCR_IsNormCluster char(1) Nulla...
NCR_IncludeInTotal char(1) Nulla...
NCR_SpecificFreightSchedule char(1) Nulla...
NCR_VATPercentage numeric(5, 2) Nulla...
NCR_UnitsPerCollo numeric(9, 2) Nulla...
NCR_Calculated char(1) Nulla...
NCR_CalculationFactor numeric(5, 2) Nulla...
NCR_Status varchar(20) Nulla...
NormClusterPriority
+NCY_ID int(max) ...
#NCR_ID int(max) ...
NCY_Priority bigint ...
NormSet
+NST_ID int(max) Null...
NST_Description varchar(80) Nulla...
OLK_ID int(max) Nulla...
NST_DateFrom datetime Nulla...
NST_DateUntil datetime Nulla...
PrognosisData
+PDA_ID int(max) Nullable...
OLK_ID int(max) Nullable ...
WPD_ID int(max) Nullable ...
#NCR_ID int(max) Nullable...
#DRR_ID int(max) Nullable...
PDA_Day tinyint Nullable ...
PDA_StartDateTime datetime Nullable ...
PDA_EndDateTime datetime Nullable ...
PDA_Amount numeric(19, 3) Nullable ...
PDA_CalculatedAmount numeric(19, 3) Nullable ...
PDA_Percentage numeric(5, 2) Nullable ...
PDA_Type varchar(3) Nullable ...
NormClusterLink
+NCK_ID int(max) Null...
#Parent_NCR_ID int(max) Null...
#Child_NCR_ID int(max) Null...
NCK_DateFrom datetime Nulla...
NCK_DateUntil datetime Nulla...
NCK_SequenceNR bigint Nulla...



F.2 Rooster

De tabellen uit de rooster module hebben als prefix: “Schedule.”

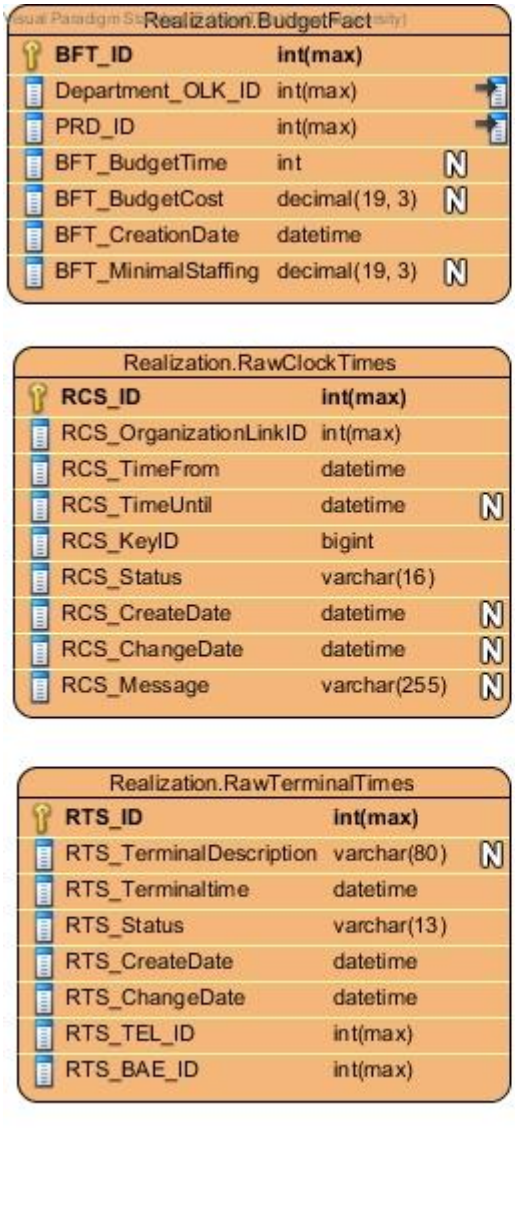

FK_TEMPLATEEMESCHEDULE_TEMPLATEEMESCHE...
Schedule.TemplateEMESchedule
TEE_ID int(max)
OLK_ID int(max)
TEE_PRD_Day tinyint
TEE_StartTime int
TEE_EndTime int
TSS_ID int(max)
EME_ID int(max)
TAK_ID int(max)
Parent_TEE_ID int(max)
N
Schedule.EmployeeSchoolShift
EST_ID int(max)
ESP_ID int(max)
EST_DayIndex tinyint
EST_StartTime smallint
EST_EndTime smallint
FK_EmployeeSchoolShift_EmployeeSchoolPeriod
Schedule.EmployeeSchoolPeriod
ESP_ID int(max)
EME_ID int(max)
ESP_StartDate datetime
ESP_EndDate datetime
ESP_State tinyint
FK_TEMPLATEEMESCHEDULE_TEMPLATESHOPSCH...
Schedule.TemplateShopSchedule
TSS_ID int(max)
TSS_Description varchar(40)
OLK_ID int(max)



F.3 Realisatie

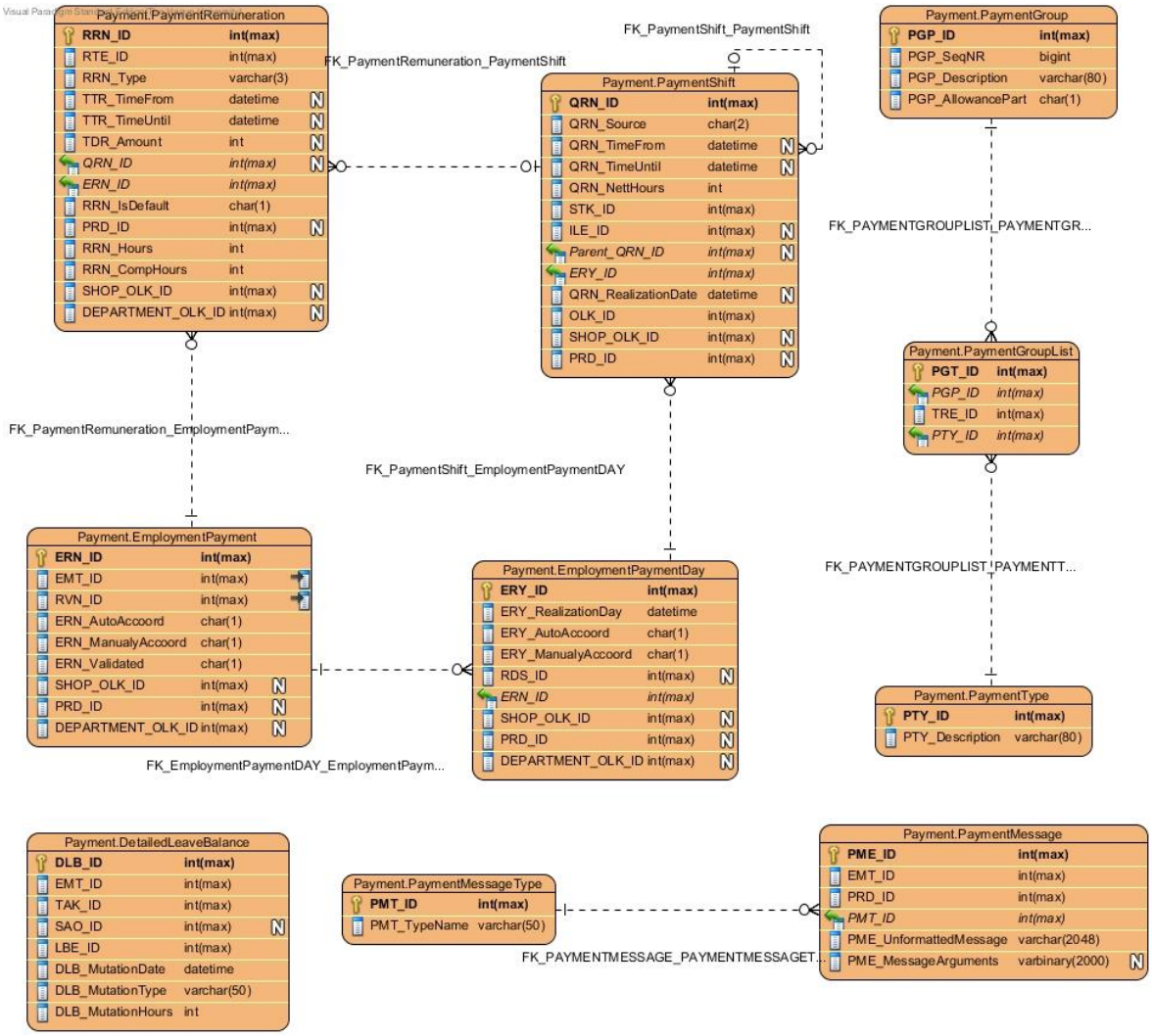
De tabellen uit de realisatie module hebben als prefix: “Realization.”


Visual Paradigm Standard Edition(The Hague University)
Realization.RealizationRemuneration
RRN_ID int(max)
RTE_ID int(max)
RRN_Type varchar(3)
TTR_TimeFrom datetime N
TTR_TimeUntil datetime N
TDR_Amount int N
QRN_ID int(max) N
ERN_ID int(max)
RRN_IsDefault char(1)
PRD_ID int(max) N
RRN_Hours int
RRN_CompHours int
SHOP_OLK_ID int(max) N
DEPARTMENT_OLK_ID int(max) N
FK_RealizationShift_RealizationShift
FK_RealizationRemuneration_RealizationShift
Realization.RealizationShift
QRN_ID int(max)
QRN_Source char(2)
QRN_TimeFrom datetime N
QRN_TimeUntil datetime N
QRN_NettHours int
STK_ID int(max)
ILE_ID int(max) N
Parent_QRN_ID int(max) N
ERY_ID int(max)
QRN_RealizationDate datetime N
OLK_ID int(max)
SHOP_OLK_ID int(max) N
PRD_ID int(max) N
FK_RealizationRemuneration_EmploymentRealizati...
Realization.EmploymentRealization
ERN_ID int(max)
EMT_ID int(max)
RVN_ID int(max)
ERN_AutoAccoord char(1)
ERN_ManualyAccoord char(1)
ERN_Validated char(1)
SHOP_OLK_ID int(max) N
PRD_ID int(max) N
DEPARTMENT_OLK_ID int(max) N
FK_RealizationShift_EmploymentRealizationDAY
Realization.EmploymentRealizationDay
ERY_ID int(max)
ERY_RealizationDay datetime
ERY_AutoAccoord char(1)
ERY_ManualyAccoord char(1)
RDS_ID int(max)
ERN_ID int(max)
SHOP_OLK_ID int(max) N
PRD_ID int(max) N
DEPARTMENT_OLK_ID int(max) N
FK_EmploymentRealizationDAY_EmploymentRealizat...
Realization.RealizationDayStatus
RDS_ID int(max)
RDS_RealizationDate datetime
WPD_ID int(max)
RDS_Source varchar(13)
RDS_Status varchar(13)
FK_EmploymentRealizationDAY_REALIZATIONDAYSTAT...
Realization.RealizationMessageType
RMT_ID int(max)
RMT_TypeName varchar(50)
FK_REALIZATIONMESSAGE_REALIZATIONMESSAGET...
Realization.RealizationMessage
RME_ID int(max)
EMT_ID int(max)
PRD_ID int(max)
RMT_ID int(max)
RME_UnformattedMessage varchar(2048)
RME_MessageArguments varbinary(2000) N



F.4 Uitbetaling

De tabellen uit de uitbetaling module hebben als prefix: “Payment.”


Payment.PaymentRemuneration
RRN_ID int(max)
RTE_ID int(max)
RRN_Type varchar(3)
TTR_TimeFrom datetime N
TTR_TimeUntil datetime N
TDR_Amount int N
QRN_ID int(max) N
ERN_ID int(max)
RRN_IsDefault char(1)
PRD_ID int(max) N
RRN_Hours int
RRN_CompHours int
SHOP_OLK_ID int(max) N
DEPARTMENT_OLK_ID int(max) N
FK_PaymentRemuneration_PaymentShift
FK_PaymentShift_PaymentShift
Payment.PaymentShift
QRN_ID int(max)
QRN_Source char(2)
QRN_TimeFrom datetime N
QRN_TimeUntil datetime N
QRN_NettHours int
STK_ID int(max)
ILE_ID int(max) N
Parent_QRN_ID int(max) N
ERY_ID int(max)
QRN_RealizationDate datetime N
OLK_ID int(max)
SHOP_OLK_ID int(max) N
PRD_ID int(max) N
Payment.PaymentGroup
PGP_ID int(max)
PGP_SeqNR bigint
PGP_Description varchar(80)
PGP_AllowancePart char(1)
FK_PAYMENTGROUPLIST_PAYMENTGR...
Payment.PaymentGroupList
PGT_ID int(max)
PGP_ID int(max)
TRE_ID int(max)
PTY_ID int(max)
FK_PaymentRemuneration_EmploymentPaym...
FK_PaymentShift_EmploymentPaymentDAY
FK_PAYMENTGROUPLIST_PAYMENTT...
Payment.EmploymentPayment
ERN_ID int(max)
EMT_ID int(max)
RVN_ID int(max)
ERN_AutoAccoord char(1)
ERN_ManualyAccoord char(1)
ERN_Validated char(1)
SHOP_OLK_ID int(max) N
PRD_ID int(max) N
DEPARTMENT_OLK_ID int(max) N
Payment.EmploymentPaymentDay
ERY_ID int(max)
ERY_RealizationDay datetime
ERY_AutoAccoord char(1)
ERY_ManualyAccoord char(1)
RDS_ID int(max) N
ERN_ID int(max)
SHOP_OLK_ID int(max) N
PRD_ID int(max) N
DEPARTMENT_OLK_ID int(max) N
Payment.PaymentType
PTY_ID int(max)
PTY_Description varchar(80)
FK_EmploymentPaymentDAY_EmploymentPaym...
Payment.DetailedLeaveBalance
DLB_ID int(max)
EMT_ID int(max)
TAK_ID int(max)
SAO_ID int(max) N
LBE_ID int(max)
DLB_MutationDate datetime
DLB_MutationType varchar(50)
DLB_MutationHours int
Payment.PaymentMessageType
PMT_ID int(max)
PMT_TypeName varchar(50)
FK_PAYMENTMESSAGE_PAYMENTMESSAGET...
Payment.PaymentMessage
PME_ID int(max)
EMT_ID int(max)
PRD_ID int(max)
PMT_ID int(max)
PME_UnformattedMessage varchar(2048)
PME_MessageArguments varbinary(2000) N

Bijlage G: Flowcharts

In deze bijlage zijn de flowcharts, behorende bij de kernfunctionaliteiten uit de 4 grote modules, te vinden. Deze flows zijn enkel ter verduidelijking van wat er gebeurd, en bevatten niet alle details(en voldoen niet altijd aan alle UML regels). De flowcharts zijn te vinden in bijlage G: Flowcharts.

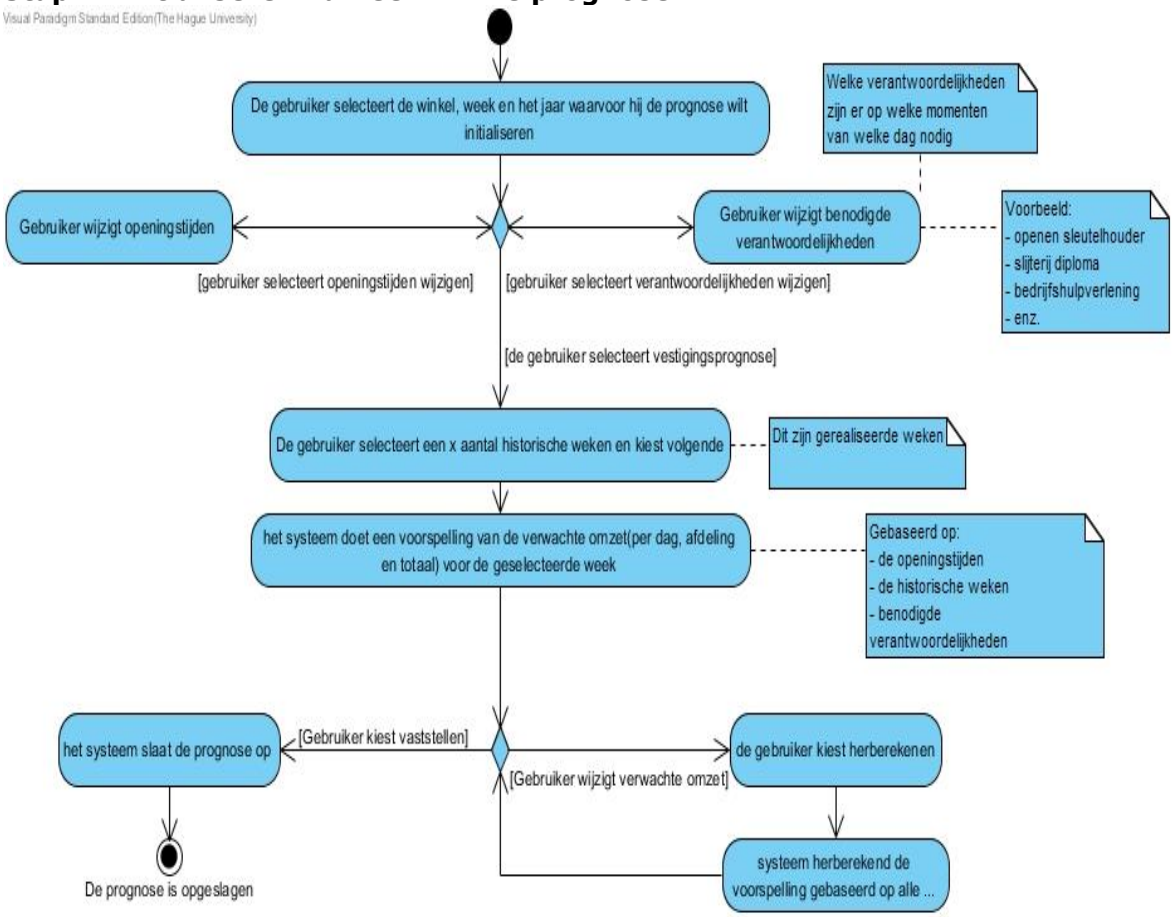
G.1 Prognose

Binnen de prognose module zijn er 3 stappen om een winkelprognose vast te stellen. Deze stappen zijn:

1. Het initialiseren van een winkelprognose
2. Het beheren/vaststellen van de afdelingsprognose
3. Het vaststellen van de winkelprognose(vaststellen alle afdelingsprognoses)

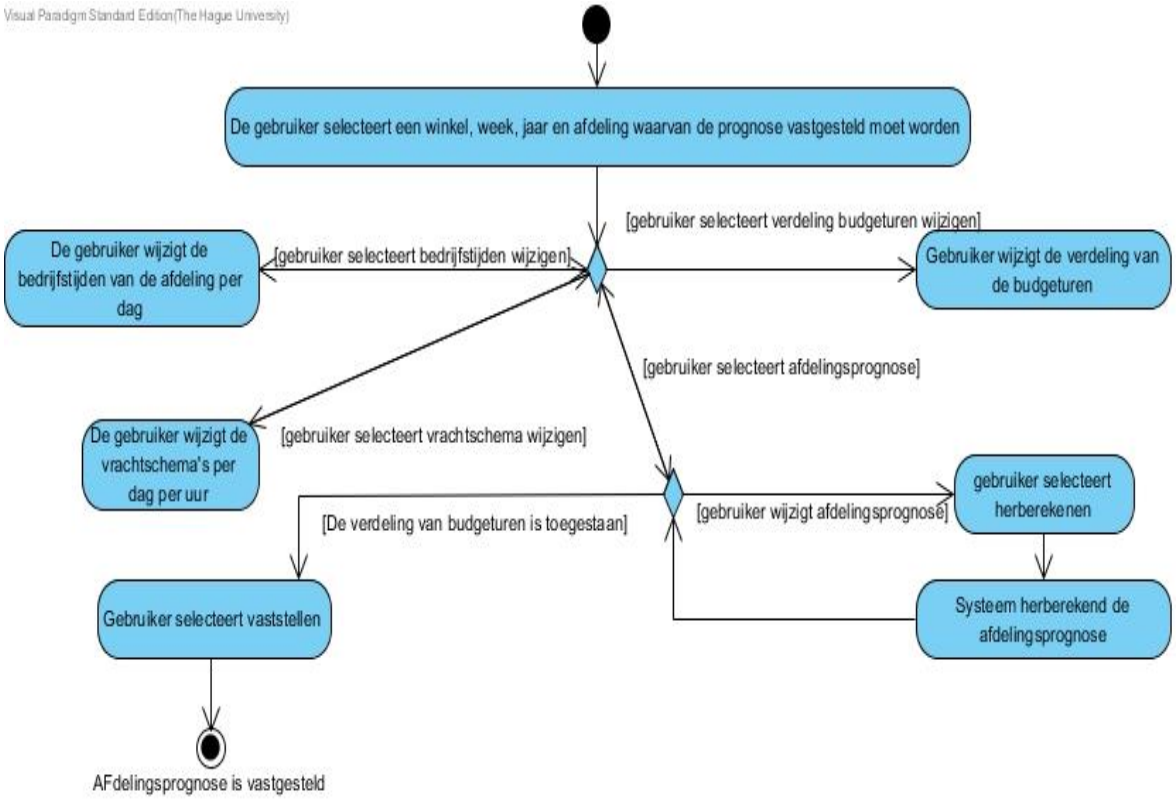
De flow per stap ziet er als volgt uit:

Stap 1: initialiseren van een winkelprognose



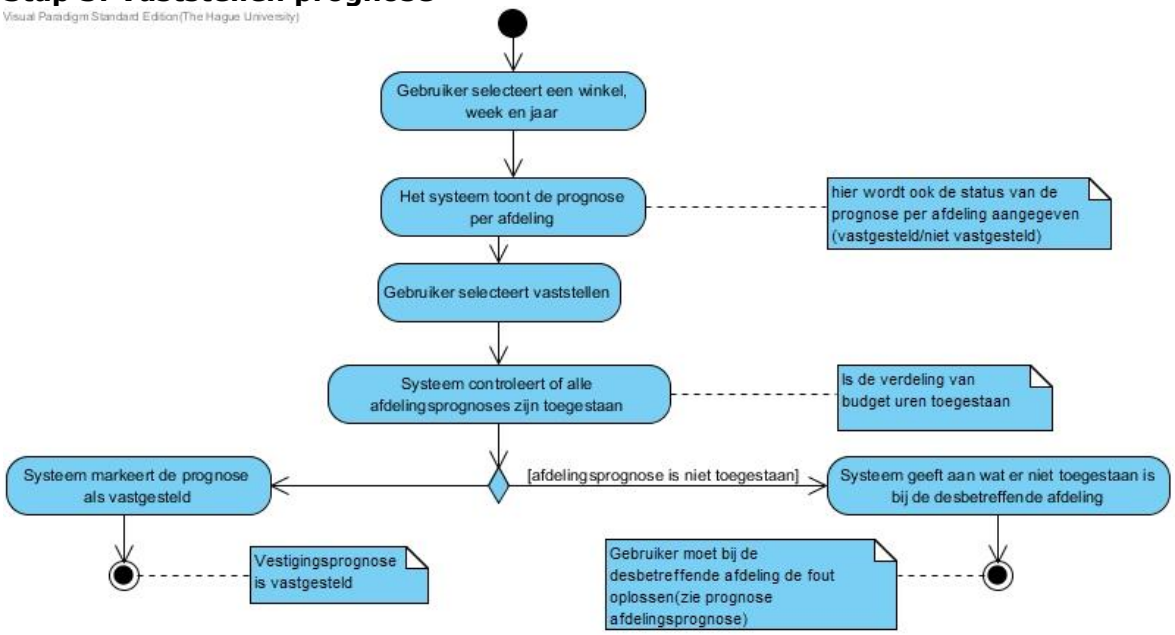
Bij het eerste beslispunt heeft de gebruiker de mogelijkheid om verschillende onderdelen uit te voeren per tab. De flow kan echter alleen verder vanuit het selecteren van historische weken.

Stap 2: beheren/vaststellen van de afdelingsprognose



Ook hier geld, dat bij het 1e beslispunt de gebruiker verschillende punten kan wijzigen, vanuit verschillende tabjes. De afdelingsprognose kan alleen vastgesteld worden vanuit afdelingsprognose.

Stap 3: Vaststellen prognose



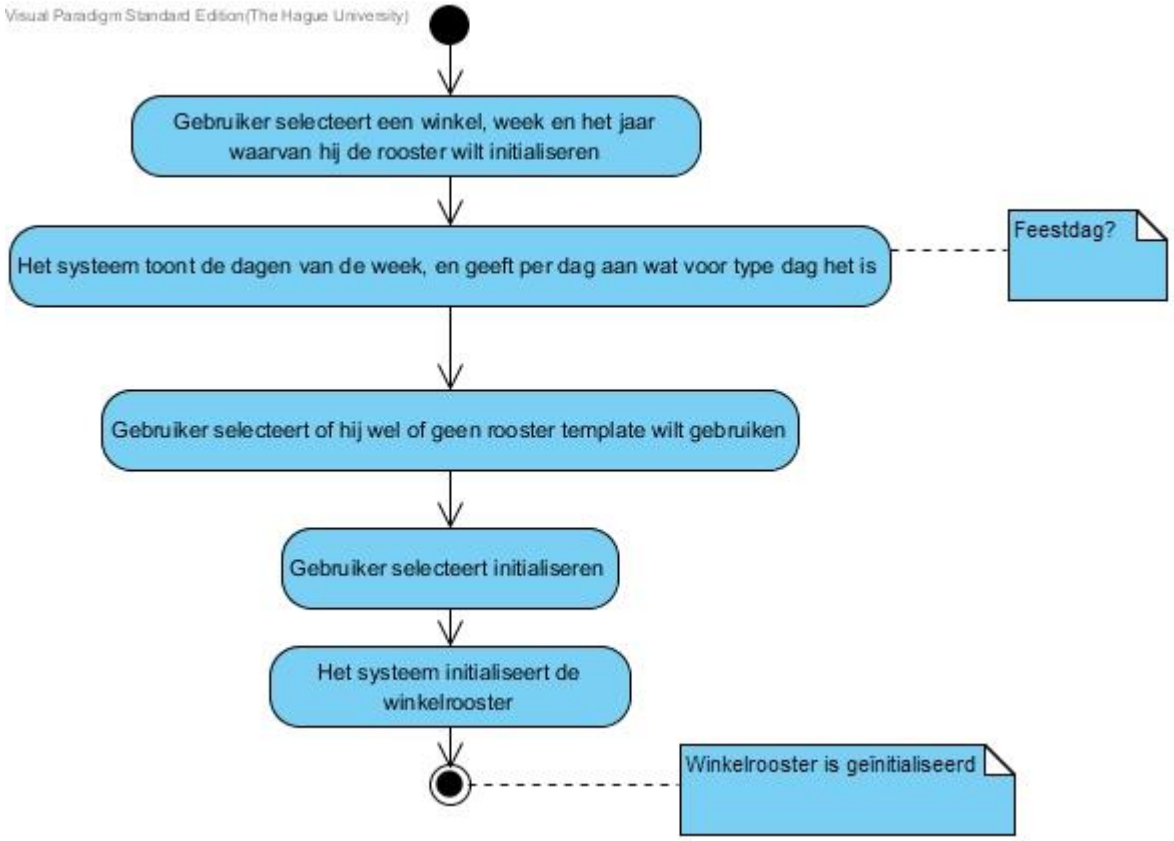
Het vaststellen van de winkelprognose is hetzelfde als alle afdelingsprognoses in 1 keer vaststellen. Het is echter niet mogelijk om wijzigingen per afdeling te maken vanuit hier.

G.2 Rooster

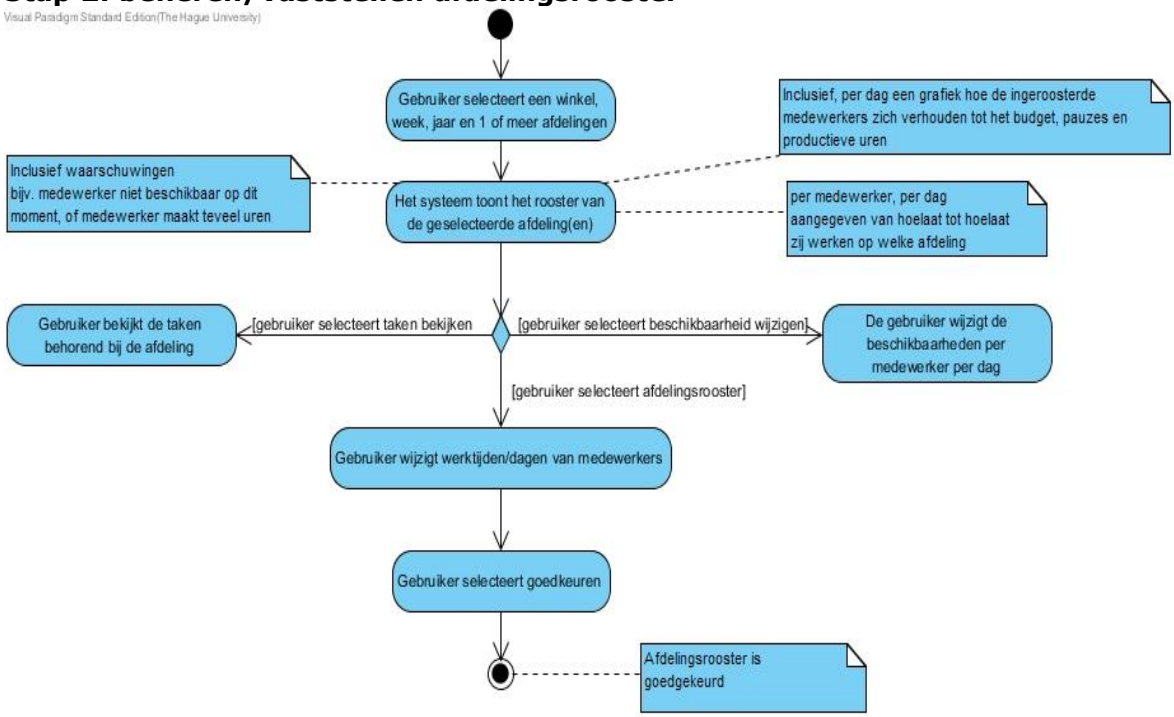
Binnen de roostermodule zijn er 3 stappen voor het vaststellen van een rooster. Daarnaast zijn er nog de mogelijkheden om standaard roosters(voor de afdelingen en de vestigingen) en standaard beschikbaarheden in te stellen. De 3 stappen voor het vaststellen van een rooster zijn:

1. Initialiseren van de winkelrooster
2. Het beheren/vaststellen van een afdelingsrooster
3. Het vaststellen van de winkelrooster

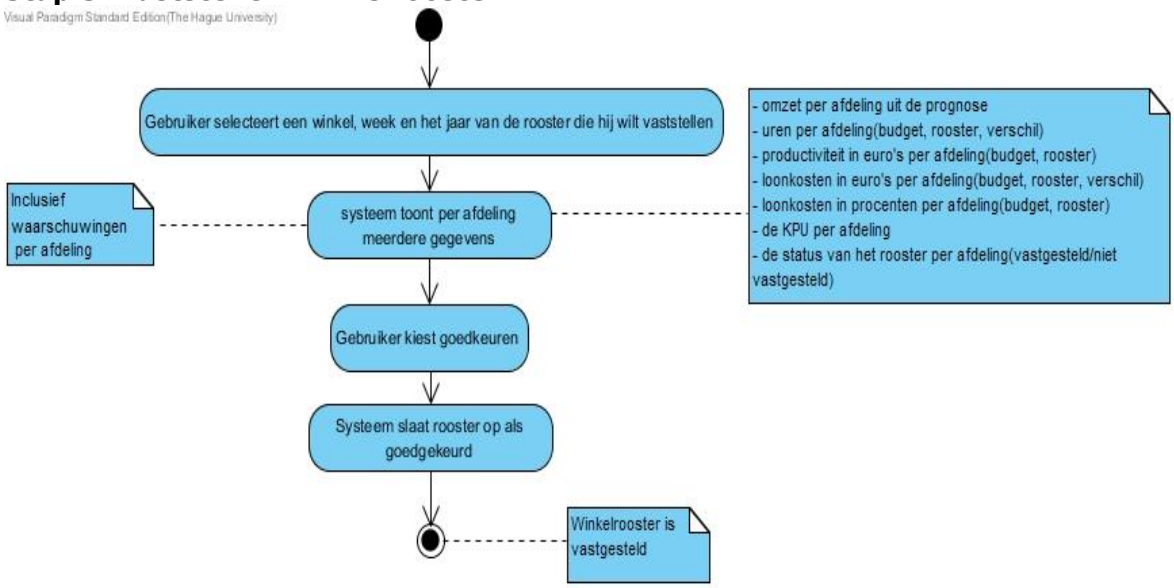
Stap 1: initialiseren winkelrooster


Visual Paradigm Standard Edition(The Hague University)
Gebruiker selecteert een winkel, week en het jaar waarvan hij de rooster wilt initialiseren
Het systeem toont de dagen van de week, en geeft per dag aan wat voor type dag het is
Feestdag?
Gebruiker selecteert of hij wel of geen rooster template wilt gebruiken
Gebruiker selecteert initialiseren
Het systeem initialiseert de winkelrooster
Winkelrooster is geïnitialiseerd

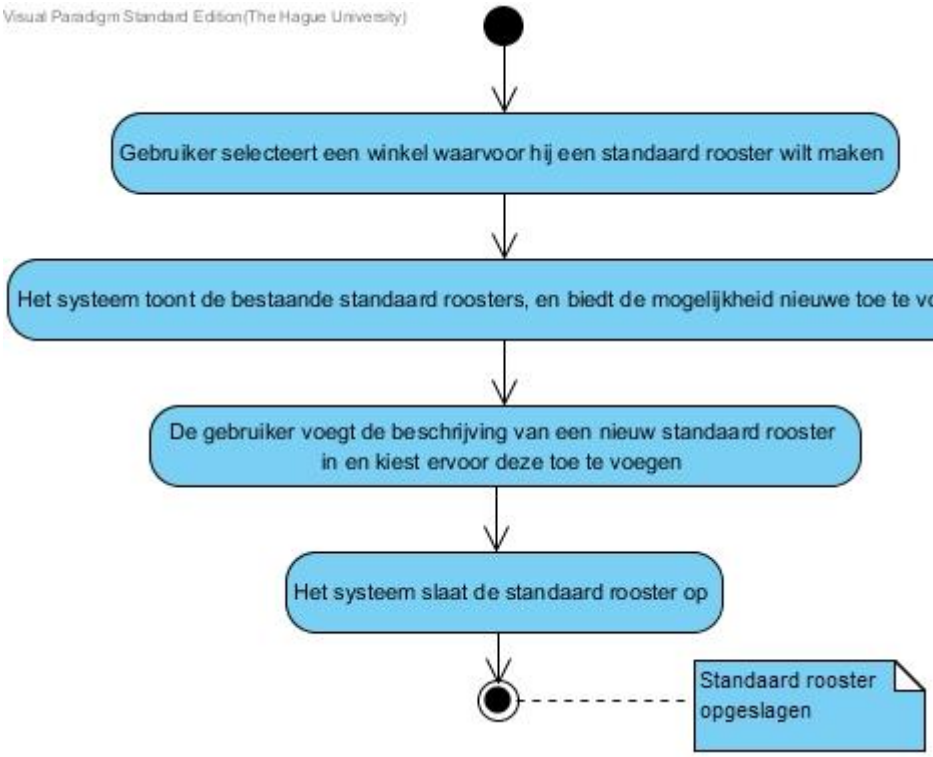
Stap 2: beheren/vaststellen afdelingsrooster



Stap 3: vaststellen winkelrooster

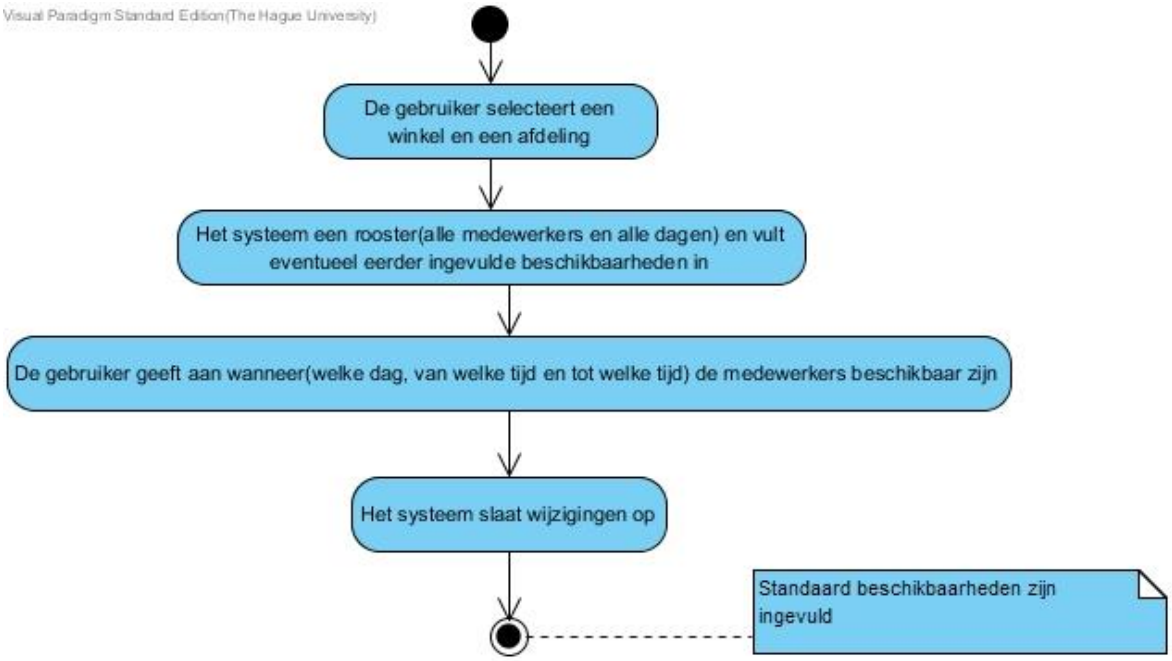


Beheren standaard winkelrooster



Beheren standaard afdelingsrooster

Beheren standaard beschikbaarheden



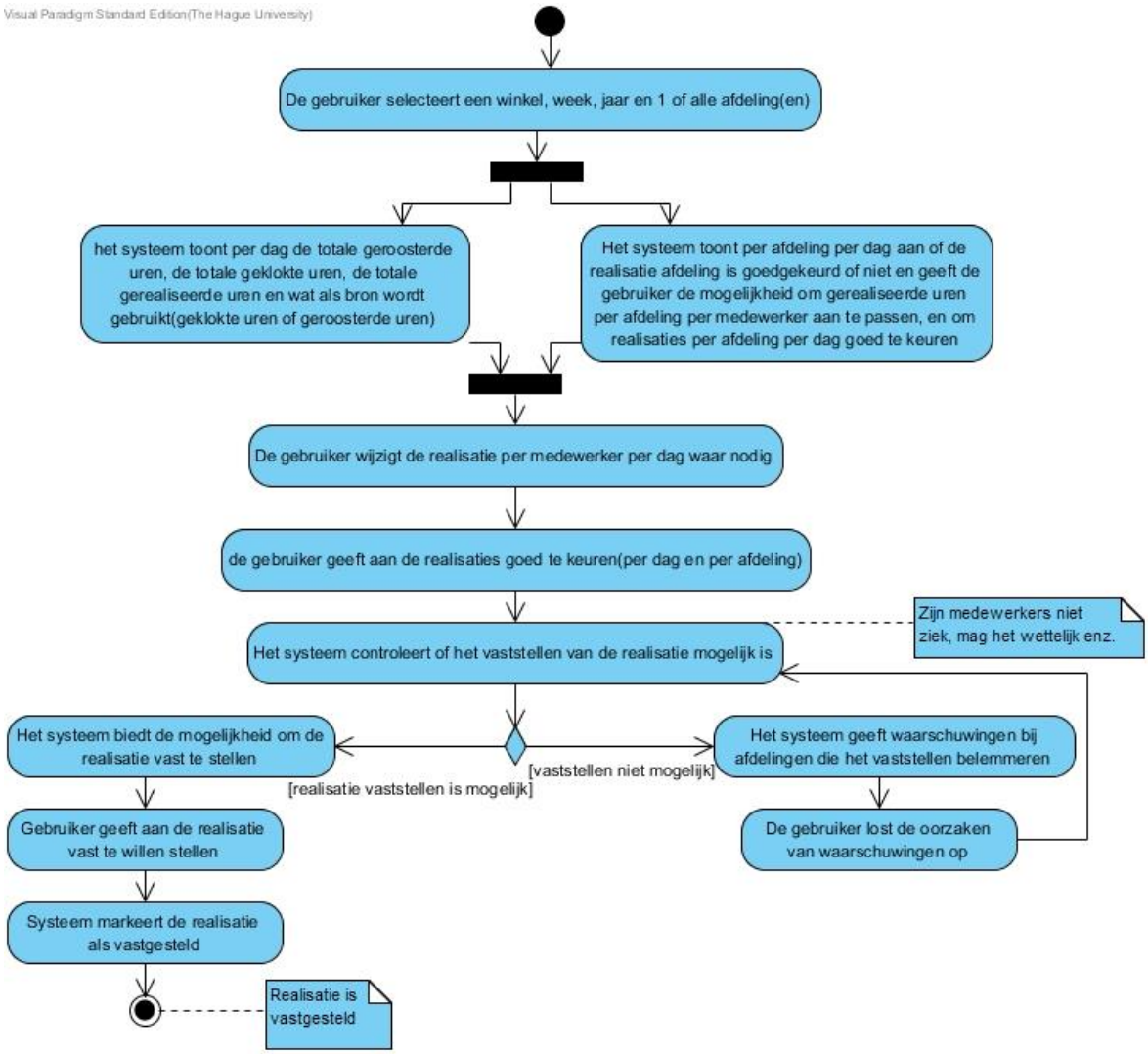
G.3 Realisatie

Er zijn 2 handelingen in de realisatie module. Dit zijn:

1. Invullen van de uren realisatie per week
2. Invullen van de gerealiseerde omzet/klanten

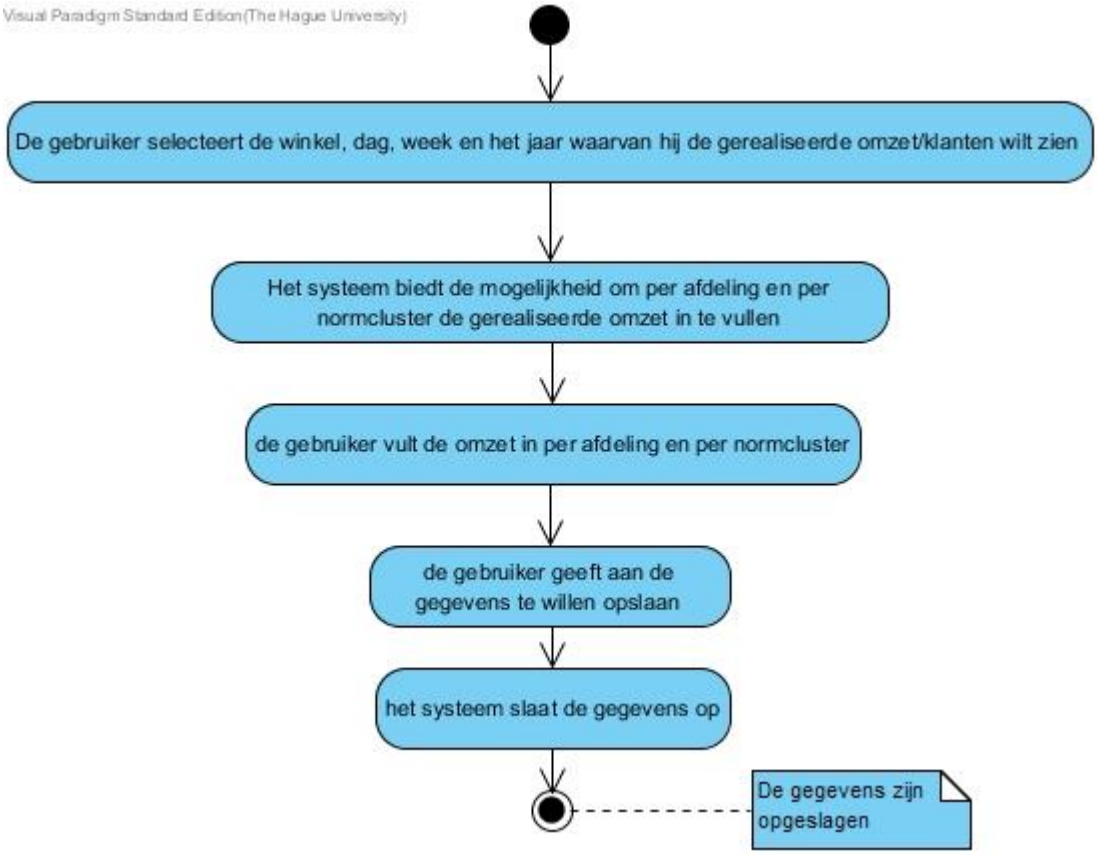
De flow van de 2 gaan als volgt:

Handeling 1: invullen gerealiseerde uren per week:



let op: de 2 activiteiten onder de horizontale balk zijn 2 delen op hetzelfde scherm. Vanwege de grote zijn ze gescheiden.

Handeling 2: gerealiseerde omzet/klanten



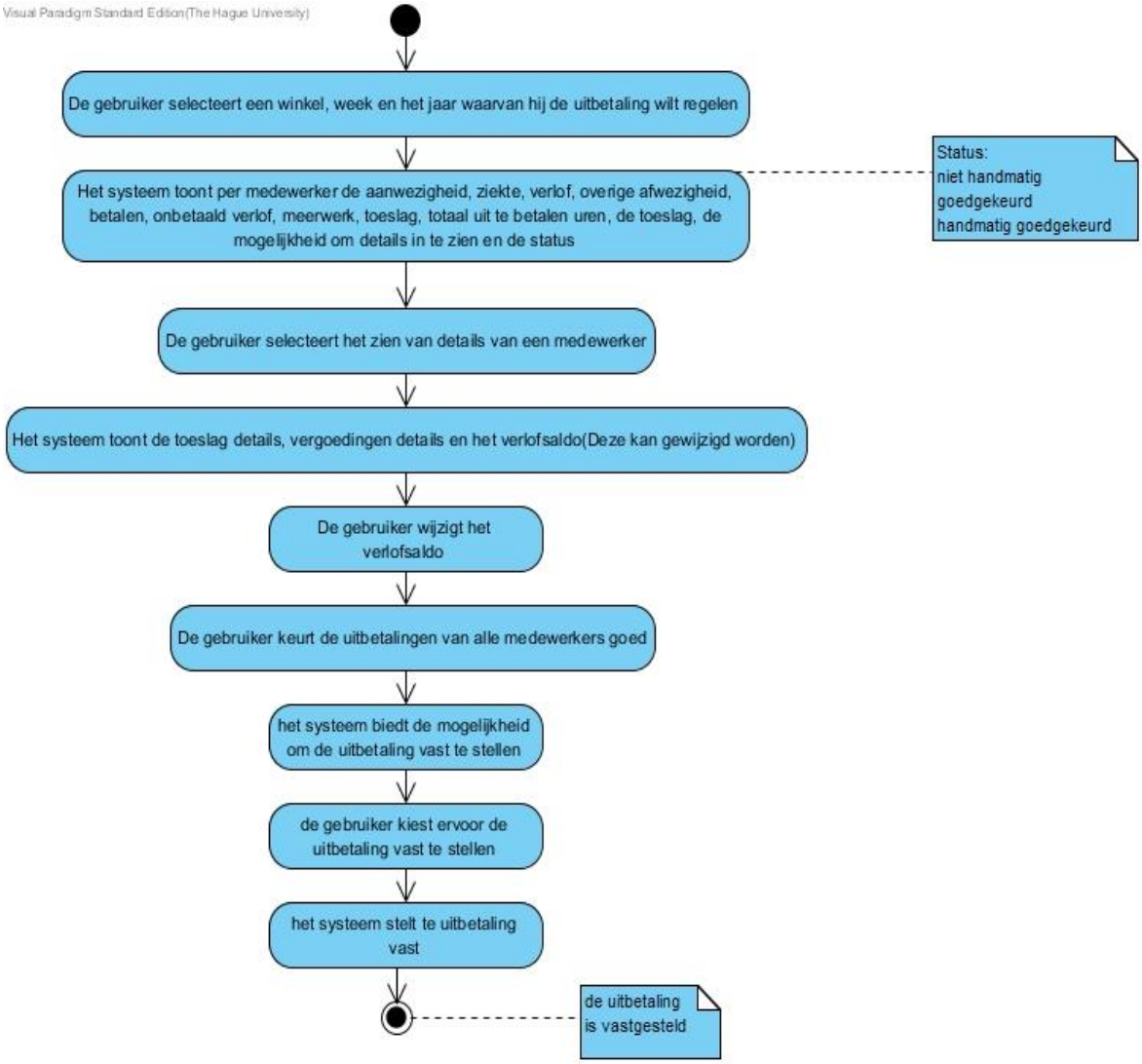
G.4 Uitbetaling

Binnen de uitbetalingsmodule zijn er 2 handelingen, namelijk:

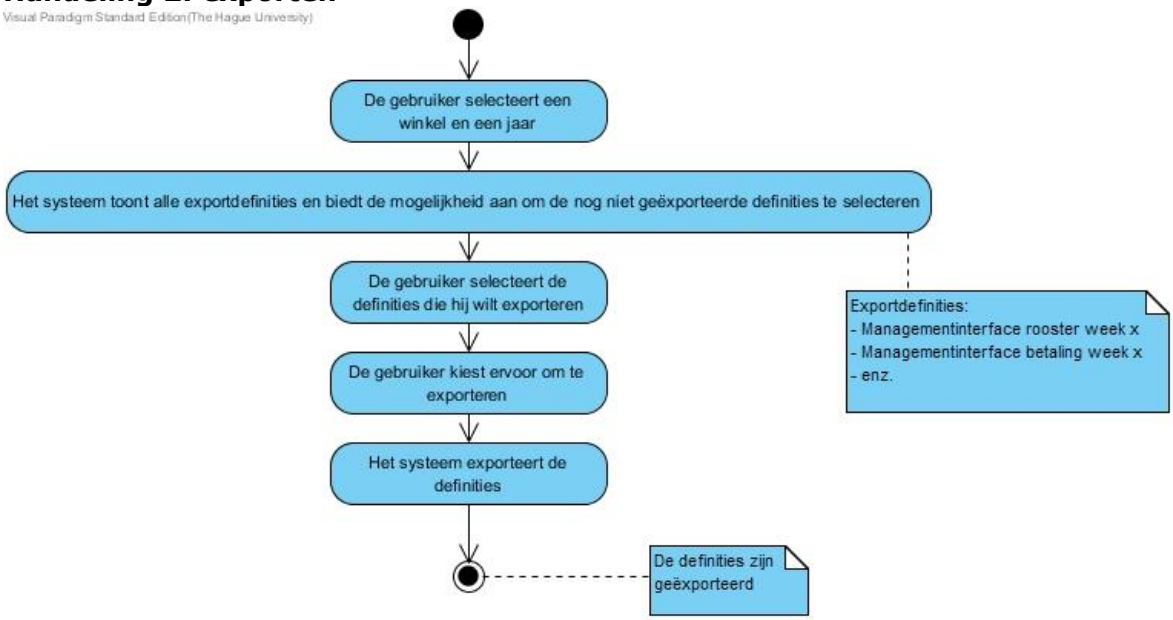
1. Beheren van betalingen
2. Exporteren van exportdefinities

De flow per handeling is als volgt:

Handeling 1: betalingen beheren


Visual Paradigm Standard Edition(The Hague University)
De gebruiker selecteert een winkel, week en het jaar waarvan hij de uitbetaling wilt regelen
Het systeem toont per medewerker de aanwezigheid, ziekte, verlof, overige afwezigheid, betalen, onbetaald verlof, meerwerk, toeslag, totaal uit te betalen uren, de toeslag, de mogelijkheid om details in te zien en de status
Status:
niet handmatig goedgekeurd
handmatig goedgekeurd
De gebruiker selecteert het zien van details van een medewerker
Het systeem toont de toeslag details, vergoedingen details en het verlofsaldo(Deze kan gewijzigd worden)
De gebruiker wijzigt het verlofsaldo
De gebruiker keurt de uitbetalingen van alle medewerkers goed
het systeem biedt de mogelijkheid om de uitbetaling vast te stellen
de gebruiker kiest ervoor de uitbetaling vast te stellen
het systeem stelt te uitbetaling vast
de uitbetaling is vastgesteld

Handeling 2: exporten



Bijlage C

Pakketselectie

Xavyr Rademaker

Management samenvatting

Dit document bevat de uitvoering en de resultaten van de pakketselectie. De reden dat er een pakketselectie is uitgevoerd is, dat de Vries WFM voordelen kan behalen uit het overstappen naar document databases voor hun applicatie R&R-web. Het is echter nog niet duidelijk welke document databases het beste aansluit op hun eisen en wensen.

Om de juiste database voor hun in kaart te brengen, heb ik een lijst van 35 document databases opgesteld. Vervolgens ben ik meerdere malen met de opdrachtgever gaan zitten op te kijken wat zijn eisen/wensen aan de document database zijn.

Per eis is er gekeken welke databases aan de eis voldoen. Op het moment dat een database niet aan een eis voldeed, viel deze af. Aan het eind waren er nog 4 databases over. Om van deze 4 databases tot 1 uiteindelijke database te komen is er een puntensysteem geïntroduceerd.

Er zijn aan deze databases weer eisen gesteld. Deze eisen hebben een weging gekregen. Vervolgens is er per eis gekeken hoe een database punten kan scoren. De totale score per eis per database was de weging * de punten.

Nadat de laatste set eisen was verwerkt bleek dat zowel RavenDB als OrientDB 12 van de 12 punten gescoord hadden. Om hier uiteindelijk 1 database uit te kiezen is met de opdrachtgever gezeten en is er besproken welke het beste past in de situatie van de Vries WFM.

Uit dit gesprek is gekomen dat RavenDB de document database is die het beste past bij R&R-web en de Vries WFM, omdat de leercurve hiervan kleiner is dan die van RavenDB, en er minder handelingen nodig zijn om CRUD acties uit te voeren dan bij OrientDB.

Inhoudsopgave

1. Inleiding

In dit hoofdstuk wordt besproken wat de aanleiding van deze pakketselectie is, welk probleem er opgelost moet worden en wat de doelstelling van de pakketselectie is.

1.1 Aanleiding

Uit een voorgaand onderzoek is gebleken dat de Vries WFM in theorie voordelen kan behalen uit het overstappen naar document databases. Vanwege het grote aanbod aan document databases, welke allemaal net wat anders zijn geïmplementeerd, is het van belang dat de Vries WFM overstapt naar een database die voldoet aan hun eisen en wensen.

1.2 Probleemstelling

Het probleem is dat de Vries WFM niet weet welke document database het beste bij hun eisen, wensen en applicatie past.

1.3 Doelstelling

Het doel van de pakketselectie is het achterhalen welke document database het beste bij de Vries WFM past.

2. Initiële lijst en initiële criteria

Tijdens het maken van het plan van aanpak en het voorgaande onderzoek kwamen al een aantal criteria van de opdrachtgever aan de nieuwe database naar boven. Deze criteria zijn in dit hoofdstuk te vinden. Verder is er ook een lijst met document databases als uitgangspunt genomen. Deze lijst is gebaseerd op de lijst met document databases van wikipedia(https://en.wikipedia.org/wiki/Document-oriented_database) en van www.nosql-database.org onder het kopje 'document store'.

2.1 Criteria

Hieronder zijn de tot nu toe bekende criteria te vinden. Per criteria wordt ook beschreven waar deze vandaan komt, en wanneer deze genoemd is.

Criteria	Source	Datum	Meegenomen in deze lijst?
De database moet op Windows server 2012 R2 draaien.	Opdrachtgever → PVA	3-2-2016/16-2-2016	Ja
Er moet documentatie beschikbaar zijn van de nieuwe database	Opdrachtgever → eerste onderzoek	16-2-2016	Nee
De nieuwe database moet te monitoren zijn	Opdrachtgever → eerste onderzoek	16-2-2016	Nee
De nieuwe database moet sneller zijn dan de huidige database	Opdrachtgever → eerste onderzoek	16-2-2016	Nee
Het moet mogelijk zijn om op performance te sturen(indexen enz.)	Opdrachtgever → eerste onderzoek	16-2-2016	Nee
Het moet mogelijk zijn back-ups te maken van de database	Opdrachtgever → eerste onderzoek	16-2-2016	Nee
Het inladen van een back-up mag maximaal 3 uur duren	Opdrachtgever → eerste onderzoek	16-2-2016	Nee
De kosten voor de database moeten niet te hoog zijn	Opdrachtgever → eerste onderzoek	16-2-2016	Nee
Klanten mogen geen data van elkaar zien (momenteel hebben zij eigen databases)	Opdrachtgever → eerste onderzoek	16-2-2016	Nee
Er moeten rechten en rollen toegekend kunnen worden aan gebruikers van de database	Opdrachtgever → eerste onderzoek	16-2-2016	Nee

Het is wenselijk dat transacties of volledig, of helemaal niet aankomen in de database	Opdrachtgever → eerste onderzoek	16-2-2016	Nee
Het moet mogelijk zijn om vanuit C#/.Net te communiceren met de database	Opdrachtnemer	01-03-2016	Ja
Er moeten stored procedures gemaakt kunnen worden in de database	Opdrachtgever → eerste onderzoek	16-2-2016	Nee

2.2 Initiële lijst

Er is een totale lijst met document databases opgesteld. Deze lijst komt van www.nosql-database.org onder het kopje 'document store' en https://en.wikipedia.org/wiki/Document-oriented_database onder het kopje 'implementations'. Vervolgens wordt per criteria gekeken of de database voldoet aan de criteria, en hoe ik aan deze informatie kom.

Totale lijst:

DBMS	Naam uitgever/ bedrijf	Source	Site
1 BaseX	BaseX team	Wikipedia	http://basex.org/
2 Cloudant	Cloudant Inc.	Wikipedia/ nosql	https://cloudant.com
3 Clusterpoint	Clusterpoint Ltd.	Wikipedia/ nosql	https://www.clusterpoint.com/
4 Couchbase server	Couchbase, Inc.	Wikipedia/ Nosql	http://www.couchbase.com/
5 CouchDB	Apache Software Foundation	Wikipedia/ Nosql	https://couchdb.apache.org/
6 CrateIO	CRATE Technology GmbH	Wikipedia	https://crate.io/
7 DocumentDB	Microsoft	Wikipedia	https://azure.microsoft.com/en-us/services/documentdb/
8 Elasticsearch	Shay Banon	Wikipedia/ nosql	https://www.elastic.co/
9 eXist	eXist	Wikipedia	http://exist-db.org/exist/apps/homepage/index.html
10 HyperDex	hyperdex.org	Wikipedia	http://hyperdex.org/
11 Informix	IBM	Wikipedia	http://www-01.ibm.com/software/data/informix/
12 Jackrabbit	Apache Foundation	Wikipedia	http://jackrabbit.apache.org/jcr/index.html
13 MarkLogic	MarkLogic Corporation	Wikipedia/ nosql	http://developer.marklogic.com/

14 MongoDB	MongoDB Inc.	Wikipedia/ nosql	**https://www.mongodb.org/**
15 OrientDB	Orient Technologies	Wikipedia/ nosqsql	http://orientdb.com/
16 Qizx	Qualcomm	Wikipedia	https://www.qualcomm.com/qizx
17 RavenDB	Hibernating Rhinos	Wikipedia/ nosql	http://ravendb.net/
~~18 RethinkDB~~	~~Beta sinds 10-2-2016 https://download.rethinkdb.com/windows/~~	~~Wikipedia/ nosql~~	~~http://www.rethinkdb.com/~~
19 Sedna	sedna.org/	Wikipedia	http://sedna.org/
20 SimpleDB	Amazon	Wikipedia	https://aws.amazon.com/simpledb/
21 ArangoDB	ArangoDB GmbH	Nosql	https://www.arangodb.com/
22 SequoiaDB	SequoiaDB	Nosql	http://www.sequoiadb.com/
23 NeDB	-	Nosql	https://github.com/louischatriot/nedb
24 Terrastore	-	Nosql	https://code.google.com/archive/p/terrastore/
25 RaptorDB			http://www.codeproject.com/Articles/375413/RaptorDB-the-Document-Store
26 AmisaDB	Amisalabs	Nosql	http://www.amisalabs.com/
27 JasDB	Obera software	Nosql	http://www.oberasoftware.com/jasdb/
28 djondb	Djondb	Nosql	http://djondb.com/
29 EJDB	Softmotions Ltd	Nosql	http://ejdb.org/
30 densoDB	-	Nosql	https://densodb.codeplex.com/
31 SDB	-	Nosql	http://pagenotes.com/wordpress/2011/12/08/sdb/ https://github.com/radare/sdb
32 NOSQL embedded db	-	Nosql	https://github.com/petersirka/nosql
~~33 ThruDB~~	~~Niet meer ontwikkeld~~		~~https://code.google.com/archive/p/thrudb/~~
34 iBoxDB	-	NoSQL	http://www.iboxdb.com/
35 BergDB	-	NoSQL	http://bergdb.com/

Totaal: 35

3. Verwerking eerste set criteria

Aan de volledige lijst, werden eerst een aantal basis eisen gesteld. Deze eisen waren voorafgaand aan het onderzoek al bekend. De onderstaande tabel toont aan welke criteria er op dit punt waren, en welke verwerkt zijn in dit hoofdstuk(als eerst dus).

3.1 Criteria

Er zijn op dit punt geen nieuwe criteria toegevoegd. Er is enkel geselecteerd wat de basis eisen aan de database waren. In de rechterkolom is te zien welke eisen op dit punt zijn meegenomen.

Criteria	Source	Datum	Meegenomen in deze lijst?
De database moet op Windows server 2012 R2 draaien.	Opdrachtgever → PVA	3-2-2016/16-2-2016	Ja
Er moet documentatie beschikbaar zijn van de nieuwe database	Opdrachtgever → eerste onderzoek	16-2-2016	Nee
De nieuwe database moet te monitoren zijn	Opdrachtgever → eerste onderzoek	16-2-2016	Nee
De nieuwe database moet sneller zijn dan de huidige database	Opdrachtgever → eerste onderzoek	16-2-2016	Nee
Het moet mogelijk zijn om op performance te sturen(indexen enz.)	Opdrachtgever → eerste onderzoek	16-2-2016	Nee
Het moet mogelijk zijn back-ups te maken van de database	Opdrachtgever → eerste onderzoek	16-2-2016	Nee
Het inladen van een back-up mag maximaal 3 uur duren	Opdrachtgever → eerste onderzoek	16-2-2016	Nee
De kosten voor de database moeten niet te hoog zijn	Opdrachtgever → eerste onderzoek	16-2-2016	Nee
Klanten mogen geen data van elkaar zien (momenteel hebben zij eigen databases)	Opdrachtgever → eerste onderzoek	16-2-2016	Nee
Er moeten rechten en rollen toegekend kunnen worden aan gebruikers van de database	Opdrachtgever → eerste onderzoek	16-2-2016	Nee

Het is wenselijk dat transacties of volledig, of helemaal niet aankomen in de database	Opdrachtgever → eerste onderzoek	16-2-2016	Nee
Het moet mogelijk zijn om vanuit C#/.Net te communiceren met de database	Opdrachtnemer	01-03-2016	Ja
Er moeten stored procedures gemaakt kunnen worden in de database	Opdrachtgever → eerste onderzoek	16-2-2016	Nee

3.2 Uitvoering per criteria

3.2.1 De database moet op Windows server 2012 R2 draaien.

Om antwoord te vinden op deze criteria, zijn de sites van de databases bekeken. Op deze sites is gekeken naar de platformen die zij supporten en/of installatiehandleidingen.

DBMS	Windows server 2012 R2	Source
1 BaseX	Ja (Java)	http://docs.basex.org/wiki/Startup
~~Cloudant~~	~~Nee~~	~~http://www-01.ibm.com/support/knowledgecenter/SSTPQH_1.0.0/com.ibm.cloudant.local.install.doc/topics/hardware_software_requirements.html?lang=en~~
~~Clusterpoint~~	~~Nee (DBaas)~~	~~https://www.clusterpoint.com/~~
2 Couchbase server	Ja	http://developer.couchbase.com/documentation/server/4.1/getting-started/installing.html#installing
3 CouchDB	Ja	http://docs.couchdb.org/en/1.6.1/install/windows.html
4 CrateIO	Ja (Java)	https://crate.io/docs/getting-started/local/windows/
~~DocumentDB~~	~~Nee (DBaas)~~	~~https://azure.microsoft.com/en-us/documentation/articles/documentdb-introduction/~~
5 Elasticsearch	Ja (Java)	https://www.elastic.co/support/matrix
6 eXist	Ja (Java)	http://exist-db.org/exist/apps/doc/quickstart.xml
~~HyperDex~~	~~Nee~~	~~http://hyperdex.org/download/~~
7 Informix	Ja	http://www-01.ibm.com/support/docview.wss?uid=swg27013343
8 Jackrabbit	Ja (Java)	http://jackrabbit.apache.org/jcr/standalone-server.html
9 MarkLogic	Ja	http://developer.marklogic.com/products/marklogic-server/requirements-8.0
10 MongoDB	Ja	https://docs.mongodb.org/manual/installation/
11 OrientDB	Ja (Java)	http://orientdb.com/docs/last/Windows-Service.html
12 Qizx	Ja (Java)	https://www.qualcomm.com/qizx
13 RavenDB	Ja (.net)	http://ravendb.net/docs/article-page/3.0/csharp/start/getting-started

~~14 RethinkDB~~	~~Ja~~	~~**http://www.rethinkdb.com/docs/install/**~~
15 Sedna	Ja	http://sedna.org/install.html#bininst_win
~~SimpleDB~~	~~Nee (DBaas)~~	~~https://aws.amazon.com/simpledb/details/~~
16 ArangoDB	Ja	https://docs.arangodb.com/Installing/Windows.html
~~SequoiaDB~~	~~Nee~~	~~http://www.sequoiadb.com/en/index.php?m=Download&a=index~~
17 NeDB	Ja (nodejs)	https://github.com/louischatriot/nedb
18 Terrastore	Ja(Java)	https://code.google.com/archive/p/terrastore/wikis/Getting_Started.wiki
19 RaptorDB	Ja(.net)	http://www.codeproject.com/Articles/375413/RaptorDB-the-Document-Store
~~AmisaDB~~	~~Nee (DBaas)~~	~~http://www.amisalabs.com/docs~~
20 JasDB	Ja	http://www.oberasoftware.com/jasdb/
21 djonDB	Ja	http://djondb.com/blog/docs/
22 EJDB	Ja	http://ejdb.org/doc/install/windows.html
23 densodb	Ja	https://densodb.codeplex.com/
24 SDB	Ja(PHP, via lamp enz)	http://nosql-database.org/
25 NoSQL embedded db	Ja(nodeJS)	https://github.com/petersirka/nosql
26 iBoxDB	Ja?	http://www.iboxdb.com/
27 BergDB	Ja?	http://bergdb.com/

Totaal: 27

3.2.2 Het moet mogelijk zijn om vanuit C#/.Net te communiceren met de database

Om antwoord te vinden op deze criteria, zijn de sites van de databases bekeken. Op deze sites is gekeken naar de aanwezigheid van drivers, de manier van communiceren met de database vanuit een applicatie of de supported languages.

DBMS	Communiceren C#/.Net	Manier van communiceren	Source
BaseX	Ja	C# language binding	http://docs.basex.org/wiki/Clients
Couchbase server	Ja	.NET SDK	http://developer.couchbase.com/documentation/server/4.1/sdks/dotnet-2.2/dotnet-intro.html
CouchDB	Ja	http / community libraries	http://docs.couchdb.org/en/1.6.1/intro/api.html https://github.com/danielwertheim/mycouch
CrateIO	Ja	HTTP	https://crate.io/docs/getting-started/clients/rest/

Elasticsearch	Ja	Elasticsearch.Net	**https://www.elastic.co/guide/en/elasticsearch/client/net-api/current/_elasticsearch_net.html**
eXist	Ja	http	http://exist-db.org/exist/apps/doc/devguide_rest.xml
Informix	Ja	http	http://www.ibm.com/support/knowledgecenter/SSGU8G_12.1.0/com.ibm.json.doc/ids_json_051.htm?lang=en
~~Jackrabbit~~	~~Nee~~	~~Java~~	~~http://jackrabbit.apache.org/jcr/jcr-api.html~~
MarkLogic	Ja	C# library XCC	http://developer.marklogic.com/labs/mldotnet https://docs.marklogic.com/dotnet/xcc/index.html
MongoDB	Ja	C# Driver	https://docs.mongodb.org/getting-started/csharp/client/
OrientDB	Ja	http (C# driver under development)	http://orientdb.com/docs/last/OrientDB-REST.html
~~Qizx~~	~~Nee~~	~~-~~	~~https://www.qualcomm.com/documents/qizx-xquery-developer-guide~~ ~~h1.3~~
RavenDB	Ja		http://ravendb.net/docs/article-page/3.0/csharp/start/getting-started
~~RethinkDB~~	~~Ja~~	~~C# driver (Community supported)~~	~~http://www.rethinkdb.com/docs/install-drivers/~~
~~Sedna~~	~~Nee~~	~~-~~	~~http://sedna.org/progguide/ProgGuide.html~~
ArangoDB	Ja	C# library	http://arangoclient.net/
~~NeDB~~	~~Nee~~	~~-~~	~~https://github.com/louischatriot/nedb~~
Terrastore	Ja	http	https://code.google.com/archive/p/terrastore/wikis/HTTP_Client_API.wiki
RaptorDB	Ja	-	http://www.codeproject.com/Articles/375413/RaptorDB-the-Document-Store
JasDB	Ja	http	https://github.com/oberasoftware/jasdb/wiki/REST-service
djonDB	Ja	C# library	http://djondb.com/

Ejdb	Ja	.net/C# language binding	http://ejdb.org/doc/bindings/index.html
Densodb	Ja	C# library	https://densodb.codeplex.com/
~~SDB~~	~~Nee~~	~~-~~	~~http://nosql-database.org/~~
~~NoSQL embedded db~~	~~Nee~~	~~-~~	~~https://github.com/petersirka/nosql~~
iBoxDB	Ja	-	http://www.iboxdb.com/
BergDB	Ja	-	http://bergdb.com/

Totaal: 21

3.3 Resultaat/overgebleven lijst

De initiële lijst bestaat uit:

1. BaseX
2. Couchbase Server
3. CouchDB
4. CrateIO
5. Elasticsearch
6. eXist
7. Informix
8. MarkLogic
9. MongoDB
10. OrientDB
11. RavenDB
12. ~~RethinkDB~~
13. ArangoDB
14. Terrastore
15. RaptorDB
16. JasDB
17. djonDB
18. Ejdb
19. Densodb
20. iBoxDB
21. BergDB

4. Verwerking tweede set criteria

Na het bekijken van de criteria 'De database moet op Windows server 2012 R2 draaien' en 'Het moet mogelijk zijn om vanuit C#/.Net te communiceren met de database' ten opzichte van de lijst met document databases, is er een lijst van 21 databases opgesteld. Aan deze lijst zijn er weer eisen gesteld. In dit hoofdstuk komt naar boven welke eisen er op dit punt nog over zijn(of erbij zijn gekomen) en welke verwerkt zijn in de tweede set criteria.

4.1 Criteria

In de vorige stap zijn er bepaalde criteria uit de al bestaande lijst met criteria gehaald. De criteria die al wel bekend waren, maar niet zijn verwerkt in de eerste set zijn meegenomen. De opdrachtgever zal uit deze lijst kunnen kiezen welke criteria in de tweede set verwerkt zal worden, en hij krijgt de mogelijkheid nieuwe criteria toe te voegen, die eventueel ook verwerkt kunnen worden op dit punt(of meegenomen worden naar een eventueel later punt).

4.1.1 Meegenomen criteria

De volgende criteria waren al wel bekend bij het maken van de initiële lijst, maar zijn op dat moment nog niet als doorslaggevend voor die desbetreffende lijst gezien. In de rechterkolom is weer te zien welke criteria zijn opgenomen in de tweede set.

Criteria	Source	Datum	Meegenomen in deze lijst?
Er moet documentatie beschikbaar zijn van de nieuwe database	Opdrachtgever → eerste onderzoek	16-2-2016	Ja
De nieuwe database moet te monitoren zijn	Opdrachtgever → eerste onderzoek	16-2-2016	Nee
De nieuwe database moet sneller zijn dan de huidige database	Opdrachtgever → eerste onderzoek	16-2-2016	Nee
Het moet mogelijk zijn om op performance te sturen(indexen enz.)	Opdrachtgever → eerste onderzoek	16-2-2016	Nee
Het moet mogelijk zijn back-ups te maken van de database	Opdrachtgever → eerste onderzoek	16-2-2016	Ja
Het inladen van een back-up mag maximaal 3 uur duren	Opdrachtgever → eerste onderzoek	16-2-2016	Nee
De kosten voor de database moeten niet te hoog zijn	Opdrachtgever → eerste onderzoek	16-2-2016	Nee

Klanten mogen geen data van elkaar zien (momenteel hebben zij eigen databases)	Opdrachtgever → eerste onderzoek	16-2-2016	Ja
Er moeten rechten en rollen toegekend kunnen worden aan gebruikers van de database	Opdrachtgever → eerste onderzoek	16-2-2016	Ja
Het is wenselijk dat transacties of volledig, of helemaal niet aankomen in de database	Opdrachtgever → eerste onderzoek	16-2-2016	Ja
Er moeten stored procedures gemaakt kunnen worden in de database	Opdrachtgever → eerste onderzoek	16-2-2016	Nee

4.1.2 Nieuwe criteria

De volgende criteria zijn toegevoegd naar eindleiding van het gesprek op 2-3-2016:

Criteria	Source	Datum	Meegenomen in deze lijst?
Load balancing. Deze criteria moet nog verder gespecificeerd worden, maar is niet voor deze fase.	Opdrachtgever → gesprek nieuwe criteria	2-3-2016	Nee

4.2 Uitvoering per criteria

Hieronder wordt per criteria beschreven hoe deze is aangepakt, en wat de resultaten ervan zijn. De gebruikte links en eventuele beschrijvingen zijn te vinden in de bijlage. De exacte bijlage wordt per criteria genoemd.

4.2.1 Er moet documentatie beschikbaar zijn van de nieuwe database

Per database is er gekeken of er documentatie beschikbaar was omtrent de volgende onderdelen:

- CRUD acties
- Performance optimalisatie → wat kan er gedaan worden als de database langzaam wordt
- Hoe wordt de structuur van de database aangepast?(toevoegen nieuwe collections enz.)
- Installatiehandleiding

De bovenstaande punten zijn bepaald door de opdrachtgever en moeten terug te vinden zijn in de documentatie van de database. Hiervoor is alleen gekeken naar documentatie op de site van de database, of een link die op hun eigen site/GitHub

vermeld staat. De resultaten zijn te vinden in de tabel op de volgende pagina. De links van waar de resultaten in de tabel terug te vinden zijn is te vinden in bijlage B: Links beschikbare documentatie.

DBMS	CRUD	Performance	Aanpassen db structuur	Installatie	Opmerkingen
1 BaseX	Ja	Ja	Ja	Ja	
2 Couchbase Server	Ja	Ja	Ja	Ja	
3 CouchDB	Ja	Ja	Ja	Ja	
~~CrateIO~~	~~Nee~~	~~Nee~~	~~Nee~~	~~Ja~~	
4 Elasticsearch	Ja	Ja	Ja	Ja	
5 eXist	Ja	Ja	Ja	Ja	
~~6 Informix~~	~~Ja~~	~~Ja~~	~~Ja~~	~~Ja~~	~~De documentatie is lastig te vinden: http://www.ibm.com/support/knowledgecenter/SSGU8G_12.1.0/com.ibm.welcome.doc/welcome.htm?lang=en~~ ~~afgevallen vanwege onduidelijke/moeilijk te vinden documentatie~~
7 MarkLogic	Ja	Ja	Ja	Ja	
8 MongoDB	Ja	Ja	Ja	Ja	
9 OrientDB	Ja	Ja	Ja	Ja	
10 RavenDB	Ja	Ja	Ja	Ja	
~~RethinkDB~~					~~Beta sinds 10-2-2016~~
11 ArangoDB	Ja	Ja	Ja	Ja	
~~Terrastore~~	~~Ja~~	~~Nee~~	~~Ja~~	~~Ja~~	
~~RaptorDB~~	~~Ja~~	~~Nee~~	~~Nee~~	~~Ja~~	
~~JasDB~~	~~Ja~~	~~Nee~~	~~Nee~~	~~Ja~~	
~~djonDB~~	~~Ja~~	~~Nee~~	~~?~~	~~Ja~~	
~~Ejdb~~	~~Ja~~	~~Nee~~	~~Nee~~	~~Ja~~	
~~Densodb~~	~~Nee~~	~~Nee~~	~~Nee~~	~~Nee~~	
~~iBoxDB~~	~~Ja~~	~~Nee~~	~~Nee~~	~~Nee~~	
~~BergDB~~	~~Ja~~	~~Nee~~	~~Ja~~	~~Ja~~	

Totaal: 10

4.2.2 Klanten mogen geen data van elkaar zien (momenteel hebben zij eigen databases)

Bij deze criteria is gekeken of het mogelijk is meerdere instanties van de database(of meerdere databases binnen de database) op 1 server te installeren. Hiervoor is de documentatie bekeken. In de documentatie is gezocht naar 2 dingen, namelijk:

1. Op welke wijze wordt de opslag van databases, collections en documents ingericht?
    a) Is er sprake van: een DBMS, daarin DB's, daarin collections en daarin documents?
    b) Is er sprake van: een DBMS, daarin collections en daarin documents?
        - Zo ja, kan er binnen een collection nog een onderscheid in type documents gemaakt worden?
2. Op het moment dat er sprake is van situatie 1b, waarbij er geen onderscheid gemaakt kan worden in type documents, moet er gekeken worden of er meerdere instanties van de dbms geïnstalleerd kunnen worden op 1 server.
    a. Als er wel onderscheid in type documents gemaakt kan worden, moet er gekeken worden of de structuur niet gewoon hetzelfde is als in situatie 1a.

Op het moment dat situatie 1a van toepassing is, betekent dit dat enkele klant zijn eigen database kan krijgen, zoals dit momenteel gaat. Op het moment dat 1b van toepassing is, is het afhankelijk van het resultaat op situatie 2. De bevindingen over de manier van opslag zijn te vinden in bijlage C: Opslagwijze databases en collections.

DBMS	**Opslag wijze**	**Opmerkingen**
BaseX	DBMS → collection → sub-collection → documents	Sub-collection zijn vergelijkbaar met namespaces
Couchbase Server	DBMS → buckets → buckets/items → items	
CouchDB	DBMS → collections → documents → views	
Elasticsearch	DBMS →indexen → types → documents	
eXist	DBMS → collection → sub-collection → documents	Sub-collection zijn vergelijkbaar met namespaces
MarkLogic	DBMS → DB's → collections → subcollections → documents	
MongoDB	DBMS →DB's → Collections → Documents	

OrientDB	DBMS → DB's → Classes → Clusters → Documents	
RavenDB	DBMS →DB's → Collections → Documents	
ArangoDB	DBMS →DB's → Collections → Documents	

Totaal: 10

4.2.3 Het moet mogelijk zijn back-ups te maken van de database

Voor de mogelijkheid om back-ups te maken, is de documentatie per database langsgelopen. Hier is gekeken naar wat de back-up mogelijkheden zijn. Mochten er beperkte back-up mogelijkheden zijn bij een database(dus wel kunnen back-uppen, maar er zijn restricties op, bijvoorbeeld de database mag niet live staan tijdens maken van een back-up) is dit met de opdrachtgever besproken. Hij heeft hierbij aangegeven of hij dit acceptabel vindt of niet. De manier van back-uppen en de link naar waar het gevonden is, zijn te vinden in bijlage D: Back-up mogelijkheden.

Bij het maken van een back-up mag de database tijdelijk offline zijn. Dit mag echter geen uren duren. Verder moet het mogelijk zijn de back-ups in te laden op andere omgevingen/servers(wel Windows server 2012 R2) en moet het mogelijk zijn de back-ups te automatiseren(als het via de command line gaat kan het ook zelf geautomatiseerd worden).

DBMS	Back-up	Opmerkingen
BaseX	Ja	Niet bekend of de DB online of offline moet zijn op moment van back-up/restore, ook niet bekend of automatisch
eXist	Ja	DB staat online maar is niet beschikbaar, transacties worden in de wacht gezet tot back-up gemaakt is Bestanden toegevoegd na de back-up blijven bestaan bij de restore tenzij anders is aangegeven
MarkLogic	Ja	DB wel beschikbaar gedurende back-up, maar wijzigingen na de start worden niet meegenomen in de back-up Back-ups kunnen ingepland worden

MongoDB	Ja	Niet bekend of de DB online of offline moet zijn op moment van back-up/restore. Kleine back-ups en restores mogelijk via de command line, grotere via de file system.
OrientDB	Ja	DB wordt gelocked voor writes gedurende de back-up. Automatische back-ups mogelijk door middel van plugin
RavenDB	Ja	De DB blijft online terwijl back-up wordt gemaakt. Als de database niet embed in de applicatie is, kunnen er automatische back-ups gemaakt worden.
ArangoDB	Ja	Niet bekend of de DB online of offline moet zijn op moment van back-up/restore. Back-ups worden gemaakt met een commando in de console. Het herstellen van back-ups ook. Via de task manager is het mogelijk om back-ups te automatiseren.
Couchbase Server	Ja	Via de command of zelf kopiëren. Database kan online blijven staan. 2 soorten incrementele back-ups mogelijk.
~~CouchDB~~	~~Nee~~	~~Geen documentatie~~
Elasticsearch	Ja	Het is mogelijk om back-ups te maken. Het is niet bekend of de db online of offline moet zijn gedurende deze handeling.

Totaal: 9

4.2.4 Er moeten rechten en rollen toegekend kunnen worden aan gebruikers van de database

Om te achterhalen of het mogelijk is om rechten en rollen toe te kennen aan gebruikers, is de documentatie van de databases geraadpleegd. Waar onduidelijkheden waren is de opdrachtgever gevraagd naar zijn mening.

DBMS	Rollen/rechten toekennen	Opmerkingen
BaseX	Ja	Het aanmaken van gebruikers en toekennen van rollen aan deze gebruikers is mogelijk. Dit kan via de console
eXist	Ja	Het geven van rechten aan gebruikers is mogelijk op basis van de manier hoe Unix het doet(read, write en execute) op document en collection niveau.
MarkLogic		Het aanmaken van gebruikers en rollen is mogelijk. Dit kan uitgevoerd worden door de rollen admin en security.
MongoDB	Ja	Het maken van rechten en rollen is mogelijk, om dit te doen moet er eerst een admin rol aangemaakt worden.
OrientDB	Ja	Biedt de mogelijkheid users en rollen te maken op zowel database als server(DBMS) niveau.
RavenDB	Ja	Authenticatie met Windows authentication of OAuth. Autorisatie per user/groep of via de bundle. Dit kan overigens alleen in de commerciële variant van RavenDB.
~~ArangoDB~~	~~Nee~~	~~Alleen volledig toegang of geen toegang.~~
~~Couchbase Server~~	~~Nee~~	~~Er is sprake van 1 full admin en 1 read only admin. De eerste kan alles doen, de 2e kan alles lezen maar niks aanpassen. Verder is het mogelijk per bucket een wachtwoord op te geven.~~
~~Elasticsearch~~	~~Nee~~	~~Elasticsearch ondersteunt op geen manier security.~~

Totaal: 6

4.2.5 Transacties moeten of volledig, of helemaal niet aankomen in de database

Om te kijken of de databases hieraan voldoen, zal de documentatie geraadpleegd worden. Binnen de documentatie zal gekeken worden naar transactie support. De gevonden links zijn te vinden in bijlage F: Transactie support.

DBMS	Transactie support	Opmerkingen
BaseX	Ja	Volledige ACID support
~~eXist~~	~~Nee~~	~~Geen documentatie over beschikbaar~~
MarkLogic	Ja	Volledige ACID support
~~MongoDB~~	~~Nee~~	~~Atomair op documentniveau, gegroepeerde operaties kunnen geïsoleerd worden.~~
OrientDB	Ja	Volledig ACID support
RavenDB	Ja	Volledig ACID support bij operaties op documenten/batches van operaties op meerdere documenten in 1 transactie. BASE als er indexen bij de transactie betrokken zijn.

Totaal: 4

4.3 Resultaat/overgebleven lijst

- BaseX
- MarkLogic
- OrientDB
- RavenDB

Vanwege de omvang van de lijst, is besloten deze lijst als shortlist te nemen.

5. Verwerking shortlist

In dit hoofdstuk wordt weer geselecteerd welke criteria behandeld zullen worden, en hoe deze aangepakt gaan worden. Vervolgens wordt er per criterium gekeken hoe de databases hierop scoren.

5.1 Criteria

Om van de shortlist tot een uiteindelijke database te komen, zal er anders gewerkt worden dan voorheen. Er zullen namelijk nog steeds criteria opgehaald worden, en er wordt nog steeds bepaald welke criteria gebruikt gaan worden om van shortlist naar resultaat te komen. Het is echter zo dat er per criteria, die gebruikt gaat worden, een weging wordt meegegeven. Deze weging is een indicatie van hoe belangrijk de criteria is, maar zal ook gebruikt worden in de resultaten.

Naast een weging, wordt er ook per criteria gekeken hoeveel er punten uitgedeeld kunnen worden aan de database bij dat criterium en op basis waarvan deze punten worden uitgedeeld. Dit wordt ook vooraf gedaan, meteen nadat de weging is bepaald.

Als een criterium is losgelaten op alle databases uit de shortlist, worden de punten verdeeld over de databases(afhankelijk van de resultaten). Vervolgens worden de punten * de weging van het criterium gedaan, om een totaalscore per database per criterium te krijgen.

Als alle criteria behandeld zijn, is er dus een totaalscore te berekenen. Op basis van deze totaalscore is er 1 database over(degene met de hoogste score).

5.1.1 Meegenomen criteria

In de onderstaande tabel is te zien welke criteria zijn meegenomen uit eerdere momenten. In de tabel is ook te zien welk van deze criteria gebruikt zal worden van de shortlist naar de einddatabase te komen. Verder is er bij deze criteria een weging gegeven. De criteria die meegenomen zijn, zijn bepaald door de opdrachtgever. Ook de wegingen hieraan zijn gegeven door de opdrachtgever.

Criteria	Source	Datum	Meegenomen	Weging
De nieuwe database moet te monitoren zijn	Opdrachtgever → eerste onderzoek	16-2-2016	Nee	-
De nieuwe database moet sneller zijn dan de huidige database	Opdrachtgever → eerste onderzoek	16-2-2016	Ja	1
Het moet mogelijk zijn om op performance te sturen(indexen enz.)	Opdrachtgever → eerste onderzoek	16-2-2016	Ja	3
Het inladen van een back-up mag maximaal 3 uur duren	Opdrachtgever → eerste onderzoek	16-2-2016	Nee	-
De kosten voor de database moeten niet te hoog zijn	Opdrachtgever → eerste onderzoek	16-2-2016	Nee	-
Er moeten stored procedures gemaakt kunnen worden in de database	Opdrachtgever → eerste onderzoek	16-2-2016	Nee	-
Load balancing → het moet mogelijk zijn de DBMS op meerdere servers te installeren	Opdrachtgever → gesprek nieuwe criteria	2-3-2016	Ja	2

5.1.2 Nieuwe criteria

Er zijn geen nieuwe criteria bij gekomen.

5.2 Te verdienen punten per criteria

Per criteria is besloten hoe het aangepakt gaat worden, en hoe er punten verdient kunnen worden per database. Dit is in overleg met de opdrachtgever gedaan.

Het moet mogelijk zijn om op performance te sturen(indexen enz.):
Om aan deze eis te voldoen is er afgesproken dat er gekeken mag worden of er documentatie beschikbaar is, of het moet aangetoond worden door middel van het uitvoeren/testen van de eis.

Punten	Eis
+1	Data moet terug te vinden zijn op basis van andere attributen dan het ID zonder dat alle documents langsgelopen worden (vergelijkbaar met secundaire indexen)

Met deze eis wordt het volgende bedoeld:

Data moet terug te vinden zijn op basis van andere attributen dan het ID zonder dat alle documents langsgelopen worden:
Op het moment dat de opdrachtgever zoekt op bijvoorbeeld de naam van een persoon, moet de database niet alle personen langsgaan om de juiste personen vinden(door bijvoorbeeld een index te plaatsen op naam kan dit verholpen worden).

Load balancing. Het moet mogelijk zijn de dbms op verschillende machines te installeren
Om aan deze eis te voldoen is er afgesproken dat er gekeken mag worden of er documentatie beschikbaar is, of het moet aangetoond worden door middel van het uitvoeren/testen van de eis.

Punten	Eis
+1	Als de DBMS op verschillende machines geïnstalleerd kan worden
+1	Als 1 DB/collection(gegevens van 1 klant) verplaatst kan worden van de DBMS uit de ene machine naar de DBMS op de andere machine
+1	Als replication, op 2 of meer machines dezelfde database draaien van een DB/collection, mogelijk is op meerdere servers

Hieronder een korte toelichting bij de eisen:

De DBMS moet op verschillende machines geïnstalleerd kunnen worden:
Momenteel heeft de Vries 3 instanties van SQL-server geïnstalleerd. Deze instanties staan niet op dezelfde machines(verschillende servers/virtualisaties). De opdrachtgever wilt dat dit ook mogelijk is met de nieuwe DBMS(het kunnen installeren op verschillende servers/virtualisaties).

Een DB/Collection kunnen verplaatsen naar een DBMS op een andere machine:
Op het moment dat er een server down gaat, of overbelast wordt, moeten databases/collections binnen de DBMS verplaatst worden naar andere machines. Het verplaatsen van de databases/collections moet mogelijk zijn.

Replication van een database/collection moet mogelijk zijn op meerdere servers:
Hiermee wordt het volgende bedoeld:
Database A draait op server A. Op het zelfde moment draait een replica(kopie) van database A op server B. Beide databases kunnen bevraagd worden door applicaties en beide bevatten (voor het grootste deel van de tijd) dezelfde data.

De nieuwe database moet sneller zijn dan de huidige database
Om deze eis te testen is er besloten dat er 4 databases geïnstalleerd worden, 1 SQL-server database en 3 document databases(de 3 overgebleven databases). De opdrachtgever zal aangeven welke tabellen uit de huidige database bevraagd/geüpdatet moeten worden om als vergelijking te dienen. Deze tabellen zullen worden nagebouwd(in de juiste vorm van document databases) en er zal een realistische hoeveelheid data in ingeladen worden.

Vervolgens zullen er query's(de opdrachtgever geeft aan wat er bevraagd moet worden) opgesteld worden en uitgevoerd worden op de verschillende databases. dit wordt gedaan via de GUI/de command line en waar nodig via een console applicatie. Wel moet caching uit staan en moeten bepaalde resultaten uit te lezen zijn(hoeveel data is er gelezen/hoeveel tijd heeft het gekost). De query's zullen zowel read en write operaties zijn.

Tot slot worden er een aantal 'bulk' inserts gedaan(grote hoeveelheden in 1x) om te kijken hoe de verschillende databases dat doen ten opzichte van SQL-server.

Punten	Eis
+1	Sneller of even snel als SQL-Server(Select query)
+1	Sneller of even snel als SQL-Server(Insert query)
+1	Het invoeren van grote hoeveelheden documents in 1x is sneller dan SQL-Server

5.3 Uitvoering per criteria

Hieronder is per criteria het volgende te zien:
- Hoe de databases hebben gescoord
- Een samenvatting van de aanpak/uitvoering

De volledige aanpak/uitvoering is te vinden in bijlage G: Sturen op performance, Bijlage H: Draaien op meerdere machines en Bijlage I: Performance vs. SQL-Server.

5.3.1 Het moet mogelijk zijn om op performance te sturen

DBMS	Documenten ophalen zonder dat alle documenten gelezen worden	Opmerkingen	Totaal punten*
BaseX	Nee	Het is niet mogelijk de indexen op elementen bij te sturen	0
~~MarkLogic~~	~~Nee~~	~~Na 2 dagen proberen niet aan de praat gekregen~~ ~~http://docs.marklogic.com/guide/rest-dev/service#id_20421~~	
Orient DB	Ja	Er kan gebruik gemaakt worden van de sql command 'where' en er kunnen indexen toegevoegd worden	3
Raven DB	Ja	Er kan alleen gezocht worden op indexen, het is dus altijd optimaal en er kan gefilterd worden	3

*aantal ja's * 3

Om te achterhalen of documenten op te halen zijn zonder dat alle documenten gelezen worden, zijn de databases gevuld met documenten/gegevens. Vervolgens zijn er op de verschillende databases gekeken hoe zij data ophalen, en of/hoe dit geoptimaliseerd kan worden.

BaseX:
In BaseX worden er automatisch indexen op elementen toegevoegd. Er kan gezocht worden op alle aanwezig elementen, en op de waardes van deze elementen(bijvoorbeeld personen waar '<name>' element de waarde 'Xavyr' bevat.

Verder is het ook mogelijk, door middel van XQuery, om deze query's efficiënt te laten uitvoeren, mits je de structuur van het document weet. Dit kan in het voorbeeld van personen die Xavyr heten als volgt gaan: for $person in db.open('naam_database')/person where $person/name='Xavyr' return $person.

Hoewel deze query efficiënt is(het kijkt alleen naar hoofdelement name binnen element person), worden nog steeds alle hoofdelementen nagelopen. Hierdoor voldoet BaseX niet aan de eis.

OrientDB:
In OrientDB is er eerst een query uitgevoerd, waarin gezocht werd naar 1 document binnen een collection op basis van andere attributen dan het ID. Na het uitvoeren van de query is er gekeken hoeveel documenten er gelezen zijn om tot het resultaat te komen. Het resultaat was alle documenten in de collection.

Vervolgens is er een index toegevoegd op de attribuut waarop gezocht wordt. Nadat de index is toegevoegd is de query nogmaals uitgevoerd en is er weer gekeken naar de query stats. Ditmaal werd er maar 1 document gelezen, en het resultaat was 1 document.
Het is in OrientDB dus mogelijk om documenten op te halen, zonder dat alle documenten gelezen worden.

RavenDB:
Bij RavenDB kunnen query's alleen uitgevoerd worden op attributen welke een index hebben. Hierdoor zijn de resultaten altijd optimaal. Dit is getest door te proberen een query uit te voeren zonder index, hierop gaf RavenDB een error. Vervolgens is er een index toegevoegd, en is dezelfde query uitgevoerd. Ditmaal lukte de query wel.

Tot slot ben ik erachter gekomen dat RavenDB wel de mogelijkheid biedt om dynamisch indexen te maken op basis van de query. Wat het doet is, het kijkt naar wat er gezocht wordt en maakt een index aan op basis van die attributen.

5.3.2 het moet mogelijk zijn de DBMS op meerdere machines te installeren

DBMS	Verschillende machines installeren	Verplaatsen van DB naar andere machine	Replication	Opmerkingen	Totaal punten*
BaseX	Ja	Ja	Nee		4
OrientDB	Ja	Ja	Ja		6
RavenDB	Ja	Ja	Ja		6

*aantal ja's * 2

Om deze eisen te achterhalen zijn de document database geïnstalleerd op mijn laptop en op mijn werkstation. Beide computers hebben Windows 10. Dit was bij alle 3 de databases mogelijk, waardoor iedereen de punten voor eerste eis verdiende.

Vervolgens is er gekeken of ik databases van de DBMS op mijn laptop kon verplaatsen naar de DBMS op mijn werkstation. Dit ging per DBMS als volgt:

BaseX:

Ik heb database gebruikt die in de eis over performance sturen ook is gebruikt. Van deze database heb ik een back-up gemaakt. BaseX zet de back-up in een .zip bestand in de data folder van BaseX. Vervolgens is dit .zip bestand van mijn laptop naar de data folder van BaseX op mijn desktop verplaatst.

Nadat de back-up verplaats was heb ik de back-up restored op mijn desktop. Het uitvoeren van de restore command zorgde ervoor dat de database werd aangemaakt op mijn desktop, met dezelfde inhoud als op mijn laptop.

OrientDB:
In OrientDB zijn dezelfde stappen ondernomen als bij BaseX. Er is een export gemaakt op mijn laptop. Deze export is verplaatst naar mijn desktop, en hier weer geïmporteerd. Na het importeren is er gekeken of de database op mijn desktop dezelfde data bevat als op mijn laptop.

RavenDB:
Om te achterhalen of databases van de ene naar de andere machine te verplaatsen is in RavenDB zijn dezelfde stappen ondernomen als bij BaseX.

Replication:
Om te achterhalen of replication mogelijk is in de verschillende DBMS-en, is er gekeken naar beschikbare documentatie. Hierin is gezocht naar replication en hoe dit opgezet moet worden. Zowel RavenDB als OrientDB hadden hier documentatie en uitleg over beschikbaar.

Bij BaseX was dat niet het geval. Er is een pdf (met sheets die gebruikt zijn in een presentatie) uit 2013 te vinden waarin distribution en replication behandeld worden, maar er is niks over te vinden in de documentatie van BaseX.

5.3.3 De nieuwe database moet sneller zijn dan de huidige database

DBMS	Select	Insert	bulk	Opmerkingen	Totaal punten*
BaseX	ja	Ja	Ja		3
OrientDB	Ja	Ja	Ja		3
RavenDB	Ja	Ja	Ja		3

*aantal ja's * 1

Om de performance van de document databases vs. SQL-Server te meten, is er eerst een SQL-Server database opgezet. Deze database bevatte dezelfde tabellen, met dezelfde data als een deel van de R&R-web applicatie.

De database bevatte de volgende tabellen en hoeveelheid rows per tabel:

Tabel	Aantal rows
Schedule.EmployeeSchedule	356533
Schedule.QualificationSchedule	537094
Reference.Period	2831
Organization.WorkPeriod	18099
Organization.WorkVersion	84390
Organization.OrganizationLink	171
Organization.Organization	40
Organization.Employment	2963
Organization.Employee	2371
Organization.Qualification	7981
Reference.Task	87

Nadat de database stond zijn de select, insert en bulk stored procedures uitgevoerd.

SQL-Server:
De query's/stored procedures voor de SQL-Server database werden geleverd door de opdrachtgever. Deze query's werden aangepast naar de database die ik beschikbaar had(delen die invloed hadden op tabellen die ik niet had zijn uit de stored procedures gehaald). Vervolgens zijn de stored procedures uitgevoerd. De 3 handelingen hadden de volgende resultaten:

- Select: 2 seconden
- Insert: 26 seconden
- Bulk: 37 seconden

Document databases:
Vervolgens is deze data per tabel uitgelezen, en opgeslagen in de 3 document databases. Binnen de document databases is er nog 1 collection bij gekomen, namelijk Afdelingsrooster. Deze heeft de volgende structuur:

```
Afdelingsrooster:{
        Employments:[{
                employmentID: "", → dit is een reference naar het volledige document
                isHired: "",
                qualificationSchedule: [{
                        department_olk_id: "",
                        qse_id : "", → dit is een reference naar het volledige document
                        datefrom: "",
                        parent_qse_ID: "",
                        timefrom: "",
                        timeuntil: "",
                        scheduledDate: "",
                        netthours: "",
                        type: "",
                        task: {
                                stk_id: "", → dit is een reference naar het volledige document
                                 CountInContractHours: "",
                                 STK_ScheduleManual: ""
                        }
                }]
        }],
```

```
        afdelingID: “”,
        winkelID: “”,
        versie: “”,
        period: {
        year: “”,
        week: “”,
        startdate: “”,
        enddate: “”,
        period_ID: “”,
        }
}
```

De select query’s zijn uitgevoerd in de GUI’s/Studio’s van de document databases.

Aangezien niet alle document databases stored procedures konden bevatten, en het idee van document databases is dat de functionaliteit op de applicatie zit, heb ik besloten een console applicatie te schrijven die de insert en de bulk uitvoerde.

Bij de start van de insert/bulk werd de starttijd tijd opgenomen, en aan het eind werd de eindtijd opgenomen. Hiertussen is het verschil in tijd uitgeprint in de console.

BaseX:
BaseX leverde de volgende resultaten:
- Select: 159.87 ms
- Insert: 0.7426649 seconden
- Bulk: 1.1972720 seconden

OrientDB:
OrientDB behaalde de volgende tijden bij het uitvoeren van de query’s:
- Select: 0.599 seconden
- Insert: 5.75 seconden
- Bulk: 7.35 seconden

RavenDB:
RavenDB behaalde de volgende tijden:
- Select: 194 ms
- Insert: 1.12 seconden
- Bulk: 1.84 seconden

Alle 3 de document databases waren dus sneller dan de SQL-Server database.

5.4 Resultaat

De volgende resultaten zijn behaald door de verschillende databases:

DBMS → Eis ↓	BaseX	RavenDB	OrientDB
Documenten ophalen zonder dat alle documenten gelezen worden	0	3	3
Losse documenten tegelijk invoeren			
Installeren op verschillende machines	2	2	2
Verplaatsen van DB naar andere machines	2	2	2
Replication mogelijkheden	0	2	2
Sneller dan SQL-server(Select)	1	1	1
Sneller dan SQL-server(Insert)	1	1	1
Sneller dan SQL-server(Bulk)	1	1	1
Totaal	**5**	**12**	**12**

Er is een gelijkspel tussen RavenDB en OrientDB uit de eisen gekomen.

Dit resultaat is met de opdrachtgever besproken. Hem is gevraagd wat nu te doen. Hij wilde het volgende weten: Welke database is het makkelijkst te gebruiken door een developer die er nog nooit mee gewerkt heeft. Vervolgens is per database het volgende uitgelegd:

5.4.1 RavenDB

RavenDB is het eenvoudigst te leren, omdat de driver zo vanzelf sprekend is. De code voor CRUD acties is doorgenomen met de opdrachtgever en er is uitgelegd hoe indexen werkt. Verder is er uitgelegd dat RavenDB uit zichzelf geen joins ondersteunt. Om documents te joinen, moeten er meerdere query's uitgevoerd worden.

5.4.2 OrientDB

OrientDB is iets moeilijker te leren, maar kan ook meer dan RavenDB. Zo heeft OrientDB bijvoorbeeld een variant op JOINS ingebouwd(LINK) welke sneller werkt dan JOINS. Het is echter zo, dat de officiele driver van OrientDB niet goed werkt/gedocumenteerd is en ook niet klaar voor productie is(staat op de volgende site: https://github.com/orientechnologies/OrientDB-NET.binary).

De rest API is wel goed gedocumenteerd, maar vereist meer handelingen van de ontwikkelaar om CRUD acties uit te voeren. De objecten moeten namelijk ge(de)serialiseerd worden van en naar JSON voor het verzenden. Bij het serializeren moet er rekening gehouden worden met de attributen: @class en @type, welke meegestuurd moeten worden.

Ook moet er een request worden opgesteld voor elke CRUD handeling, en moet de response uitgelezen worden.

5.4.3 Eindresultaat

Op basis van de informatie over de 2 databases, en een korte walkthrough door de CRUD handelingen, heeft de opdrachtgever aangegeven dat RavenDB het best klinkt voor hem. Gezien de eerder genoemde punten deelde ik deze mening met hem. Vandaar dat RavenDB gekozen is als de database voor R&R-web.

6. Conclusie

Aan de hand van alle uitgevoerde tests, de afgewogen criteria en het laatste gesprek met de opdrachtgever kan er worden geconcludeerd dat RavenDB het beste past bij de eisen en wensen van de Vries WFM, voor de R&R-web applicatie.

Bijlage A: Glossary

DBMS: een database management systeem. Hieronder wordt de implementatie van document databases bedoelt(MongoDB, BaseX, CouchDB enz.).

DB: een database. Hiermee wordt bedoeld: een database binnen het DBMS. Bij sommige DBMS-en is een database gelijk aan een collection. In dit geval wordt er vermeld dat deze als hetzelfde gezien worden, en wordt de term collection verder gebruikt.

Collection: Een collection is een verzameling van documents. In relationele databases zou dit gezien worden als een tabel.

Document: Een document is een enkel bestand met gegevens. Dit is vergelijkbaar met een row in relationele databases.

System collections: Collections die data over de structuur(indexen enz.) van collections binnen de database bevat.

Non-system collections: Collections waar de documents in staan.

Namespace: een aantal DBMS-en die maar 1 database, maar meerdere collections binnen het geheel ondersteunen maken gebruik van namespaces. Dit is in principe vergelijkbaar met een collection.

Bijlage B: Links beschikbare documentatie

Deze bijlage geeft voor de eis: 'Er moet documentatie beschikbaar zijn van de nieuwe database' aan waar de documentatie gevonden is per database.

B.1 BaseX

CRUD acties
Voor het uitvoeren van CRUD acties vanuit C# zijn de volgende voorbeelden beschikbaar:
https://github.com/BaseXdb/basex/tree/master/basex-api/src/main/c%23

voor de query mogelijkheden met de REST API is het volgende beschikbaar:
http://docs.basex.org/wiki/REST

Performance optimalisatie → wat kan er gedaan worden als de database langzaam wordt
BaseX maakt automatisch indexen aan voor data waarop hij vindt dat relevant is. Daarnaast is het ook mogelijk om zelf indexen aan te maken. Dit is te vinden op:
http://docs.basex.org/wiki/Indexes
Hoe wordt de structuur van de database aangepast?(toevoegen nieuwe collections enz.)
Het aanmaken van nieuwe databases gaat via de GUI op de server. De documentatie hierover is te vinden op:
http://docs.basex.org/wiki/Databases

Verder zijn er ook nog administratieve acties, te vinden op:
http://docs.basex.org/wiki/Commands#Administration

Installatiehandleiding
De installatiehandleiding is te vinden op:
http://docs.basex.org/wiki/Startup

B.2 Couchbase Server

CRUD acties
Crud acties vanuit C# zijn de vinden op:
http://developer.couchbase.com/documentation/server/4.1/sdks/dotnet-2.2/documents.html

CRUD acties vanuit de console is hier te vinden:
http://developer.couchbase.com/documentation/server/4.1/developer-guide/data-access-overview.html

Performance optimalisatie → wat kan er gedaan worden als de database langzaam wordt
Couchbase heeft verschillende manieren van indexeren, namelijk: mapreduce views, spatial views en global secundaire indexen. Meer hierover is te vinden op:
http://developer.couchbase.com/documentation/server/4.1/indexes/indexing-overview.html

Verder is het volgende beschikbaar:
http://docs.couchbase.com/admin/admin/Concepts/concept-intro.html

Hoe wordt de structuur van de database aangepast?(toevoegen nieuwe collections enz.)
Het aanmaken van buckets in couchbase wordt gedaan door de administrator. Documentatie hierover is te vinden op:
http://developer.couchbase.com/documentation/server/4.1/clustersetup/create-bucket.html

Installatiehandleiding
Het snel installeren is te vinden op:
http://developer.couchbase.com/documentation/server/4.1/getting-started/installing.html

De gedetailleerde installatiehandleiding is te vinden op:
http://developer.couchbase.com/documentation/server/4.1/install/init-setup.html#topic12527

B.3 CouchDB

CRUD acties
Het uitvoeren van CRUD acties(de requests) is te vinden op:
http://docs.couchdb.org/en/1.6.1/intro/api.html

Informatie over de community C# library is te vinden op:
https://github.com/danielwertheim/mycouch

Performance optimalisatie → wat kan er gedaan worden als de database langzaam wordt
Informatie over het verbeteren van de performance is hier te vinden:
http://docs.couchdb.org/en/1.6.1/maintenance/performance.html

Hoe wordt de structuur van de database aangepast?(toevoegen nieuwe collections enz.)
Het maken van de database via curl wordt hier beschreven:
http://docs.couchdb.org/en/1.6.1/intro/api.html#databases

Installatiehandleiding
De installatie op windows wordt beschreven op:
http://docs.couchdb.org/en/1.6.1/install/windows.html

B.4 CrateIO

CRUD acties
Er is alleen 1 voorbeeld te vinden van het uitvoeren van een query via de rest api. Dit voorbeeld is helaas niet te gebruiken in C#.

Performance optimalisatie → wat kan er gedaan worden als de database langzaam wordt
Er is geen documentatie omtrent performance optimalisatie te vinden, alleen dat het te configureren is.
Hoe wordt de structuur van de database aangepast?(toevoegen nieuwe collections enz.)

-

Installatiehandleiding
Te vinden op:
https://crate.io/docs/getting-started/local/windows/

B.5 Elasticsearch

CRUD acties
Het uitvoeren van query's in .Net is te vinden op:
http://nest.azurewebsites.net/nest/core/get.html

het aanmaken/wijzigen van een document vanuit .Net is te vinden op:
http://nest.azurewebsites.net/nest/core/

http://nest.azurewebsites.net/nest/core/update.html

Delete is te vinden op:
http://nest.azurewebsites.net/nest/core/delete.html

Verder is er voor de console nog documentatie beschikbaar op:
https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html

Performance optimalisatie → wat kan er gedaan worden als de database langzaam wordt
Er zijn een aantal tips beschikbaar omtrent performance bij het gebruik van indexes. Deze zijn te vinden op:
https://www.elastic.co/guide/en/elasticsearch/guide/current/indexing-performance.html

Hoe wordt de structuur van de database aangepast?(toevoegen nieuwe collections enz.)
Het aanmaken en wijzigen van de database structuur wordt gedaan vanuit de code(vanuit de query's).
https://www.elastic.co/guide/en/elasticsearch/guide/current/_indexing_employee_documents.html

Installatiehandleiding
Een van de installatiehandleidingen is te vinden op:
https://www.elastic.co/guide/en/elasticsearch/reference/current/_installation.html

B.6 eXist

CRUD acties
Alle requests voor CRUD acties worden beschreven op:
http://exist-db.org/exist/apps/doc/devguide_rest.xml#D2.2.3.14

Performance optimalisatie → wat kan er gedaan worden als de database langzaam wordt
De documentatie omtrent performance tuning is te vinden op:

http://exist-db.org/exist/apps/doc/tuning.xml

Hoe wordt de structuur van de database aangepast?(toevoegen nieuwe collections enz.)
Er staat beschreven hoe een nieuwe schema aangemaakt kan worden via het dashboard, en hoe files hiernaar geüpload kunnen worden. Dit is terug te vinden op:
http://exist-db.org/exist/apps/doc/uploading-files.xml

Installatiehandleiding
De installatiehandleiding is te vinden op:
http://exist-db.org/exist/apps/doc/quickstart.xml

Om eXist als een service te draaien is de handleiding te vinden op:
http://exist-db.org/exist/apps/doc/advanced-installation.xml

B.7 Informix

CRUD acties
Informatie over het uitvoeren van CRUD acties in C# is te vinden op:
http://www.ibm.com/support/knowledgecenter/SSGU8G_12.1.0/com.ibm.json.doc/ids_json_014.htm?lang=en

het query'en gaat via de drivers van MongoDB.

Performance optimalisatie → wat kan er gedaan worden als de database langzaam wordt
In het volgende kopje wordt gekeken, hoe performance gemeten en beheert kan worden in Informix:
http://www.ibm.com/support/knowledgecenter/SSGU8G_12.1.0/com.ibm.perf.doc/ids_prf_037.htm?lang=en

Hoe wordt de structuur van de database aangepast?(toevoegen nieuwe collections enz.)
Het aanpassen van de structuur van de database kan gewoon via http requests. De requests staan hier vermeld:
http://www.ibm.com/support/knowledgecenter/SSGU8G_12.1.0/com.ibm.json.doc/ids_json_051.htm?lang=en

Dit kan ook via de drivers beschreven bij CRUD acties.

Ook de crud acties via http zijn hier te vinden.
Installatiehandleiding
Voor het installeren zijn een aantal links beschikbaar, namelijk:
Voorbereiden van de installatie:
http://www.ibm.com/support/knowledgecenter/SSGU8G_12.1.0/com.ibm.inst.doc/ids_inst_040.htm?lang=en

Uitvoeren van de installatie op een interactieve manier:
http://www.ibm.com/support/knowledgecenter/SSGU8G_12.1.0/com.ibm.inst.doc/ids_inst_012.htm?lang=en

Uitvoeren van de installatie op een niet interactieve manier:

http://www.ibm.com/support/knowledgecenter/SSGU8G_12.1.0/com.ibm.inst.doc/ids_inst_015.htm?lang=en

De niet interactieve manier is alleen mogelijk op het moment dat er al eens eerder een installatie is geweest op een interactieve manier, en daarbij een installatie template is gemaakt. Het maken van dit template staat op:
http://www.ibm.com/support/knowledgecenter/SSGU8G_12.1.0/com.ibm.inst.doc/ids_inst_011.htm?lang=en

B.8 MarkLogic

CRUD acties
Voor de rest API is de volgende documentatie beschikbaar:
https://docs.marklogic.com/REST/client/management

Voor het werken met de C# library(XCC) is weinig tot geen documentatie beschikbaar.
Enkel:
https://docs.marklogic.com/dotnet/xcc/index.html

en:
https://docs.marklogic.com/guide/xcc/concepts#id_55196

Hier komt echter alleen insert en read naar voren, de rest van de commando's worden niet behandeld.

Performance optimalisatie → wat kan er gedaan worden als de database langzaam wordt
Er worden meerdere manieren voor het tunen van performance benoemd op:
https://docs.marklogic.com/guide/performance
Hoe wordt de structuur van de database aangepast?(toevoegen nieuwe collections enz.)
Het beheren van schema's in de MarkLogic is te vinden op:
http://docs.marklogic.com/guide/admin/schemas
Het beheren van de gehele database staat op:
http://docs.marklogic.com/guide/admin/databases
Installatiehandleiding
Het installeren van MarkLogic op een Windows pc is te vinden op:
http://docs.marklogic.com/guide/installation/procedures#id_28962

B.9 MongoDB

CRUD acties
Er is documentatie beschibaar voor het uitvoeren van CRUD acties in MongoDB. Deze is te vinden op: https://docs.mongodb.org/getting-started/csharp/client/ en
http://mongodb.github.io/mongo-csharp-driver/2.2/reference/driver/crud/

Verder zijn CRUD acties via de console(in javascript) te vinden op:
https://docs.mongodb.org/manual/core/crud-introduction/

Performance optimalisatie → wat kan er gedaan worden als de database langzaam wordt

Omtrent performance tuning is het volgende beschikbaar:
https://www.mongodb.com/presentations/mongosv-2012/mongodb-performance-tuning
en
https://docs.mongodb.org/manual/tutorial/optimize-query-performance-with-indexes-and-projections/

Verder is het mogelijk om indexen aan te maken (https://docs.mongodb.org/manual/core/indexes-introduction/) en sharding (https://docs.mongodb.org/manual/core/sharding-introduction/)

Hoe wordt de structuur van de database aangepast?(toevoegen nieuwe collections enz.)
De structuur van: sub databases en documents wordt aangemaakt vanuit de applicatie. Ook de bestaande databases komen hiervandaan. Op het moment dat er een sub database of collection wordt opgevraagd welke niet bestaat wordt deze aangemaakt.
https://docs.mongodb.org/getting-started/csharp/insert/

https://docs.mongodb.org/manual/mongo/

Installatiehandleiding
Ook hier is documentatie over beschikbaar, namelijk op:
https://docs.mongodb.org/manual/installation/

B.10 OrientDB

CRUD acties
Voor het uitvoeren van CRUD acties via de C# driver kan gekeken worden op:
https://github.com/yojimbo87/OrientDB-NET.binary/wiki#fluent-api-for-sql-queries

Aangezien er op github staat dat de driver nog ontwikkeld wordt, is het verstandig om ook naar de http mogelijkheden te kijken. Deze zijn hier te vinden:
http://orientdb.com/docs/last/OrientDB-REST.html

Performance optimalisatie → wat kan er gedaan worden als de database langzaam wordt
Tips over het optimaliseren van de database is te vinden op:
http://orientdb.com/docs/last/Performance-Tuning.html

Hoe wordt de structuur van de database aangepast?(toevoegen nieuwe collections enz.)
Het maken/verwijderen van een databases in OrientDB kan via de console en via de GUI. Via de console gaat het als volgt:
http://orientdb.com/docs/last/Server-Management-Java.html

Installatiehandleiding
Voor de installatie is het volgende beschikbaar:
http://orientdb.com/docs/last/Tutorial-Installation.html

Het installeren als een service op Windows staat hier:
http://orientdb.com/docs/last/Windows-Service.html

Voor het runnen van de server:

http://orientdb.com/docs/last/Tutorial-Run-the-server.html

Voor het runnen van de GUI:
http://orientdb.com/docs/last/Tutorial-Run-the-studio.html

B.11 RavenDB

CRUD acties
De Update/Create actie:
http://ravendb.net/docs/article-page/3.0/csharp/client-api/commands/documents/put

Read:
http://ravendb.net/docs/article-page/3.0/csharp/client-api/commands/documents/get

Read met LINQ(incl. gebruikmakend van indexen):
http://ravendb.net/docs/article-page/3.0/csharp/indexes/querying/basics
Delete:
http://ravendb.net/docs/article-page/3.0/csharp/client-api/commands/documents/delete

Delete/update multiple:
http://ravendb.net/docs/article-page/3.0/csharp/client-api/commands/documents/how-to/delete-or-update-documents-using-index

Performance optimalisatie → wat kan er gedaan worden als de database langzaam wordt
Tussen de configuratie opties staan een paar onderdelen waar rekening mee gehouden kan worden omtrent performance:
http://ravendb.net/docs/article-page/3.0/csharp/server/configuration/configuration-options

Naast de informatie omtrent indexes(http://ravendb.net/docs/article-page/3.0/csharp/indexes/what-are-indexes) wordt er ook aangegeven hoe LoadStartsWith methode voor performancewinst kan zorgen:
http://ravendb.net/docs/article-page/3.0/csharp/indexes/querying/paging

Hoe wordt de structuur van de database aangepast?(toevoegen nieuwe collections enz.)
De volledige structuur van de database wordt bepaald vanuit de applicatie.
Het maken van databases en het beheren hiervan kan via Studio. Meer informatie over Studio is te vinden op:
http://ravendb.net/docs/article-page/3.0/csharp/studio/accessing-studio
Installatiehandleiding
De installatie met de installer is te vinden op:
http://ravendb.net/docs/article-page/3.0/csharp/server/installation/using-installer

~~B.12 RethinkDB – Navragen, Windows nog in beta~~

~~**CRUD acties**~~

~~**Performance optimalisatie → wat kan er gedaan worden als de database langzaam wordt**~~

~~**Hoe wordt de structuur van de database aangepast?(toevoegen nieuwe collections enz.)**~~

~~**Installatiehandleiding**~~
~~Te vinden op:~~
~~http://ravendb.net/docs/article-page/3.0/csharp/server/installation/using-installer~~

~~of voor de embedded versie(RavenDB embed in de applicatie):~~
~~http://ravendb.net/docs/article-page/3.0/csharp/server/installation/embedded~~

B.13 ArangoDB

CRUD acties
Het uitvoeren van CRUD acties via http is te vinden op:
https://docs.arangodb.com/HttpDocument/WorkingWithDocuments.html

Verder zijn de console(js) commando's te vinden op:
https://docs.arangodb.com/Documents/DatabaseMethods.html

De C# library heeft dezelfde methodes als de js commando's. Het gebruik van een aantal van ze is te vinden op:
http://arangoclient.net/Document/DatabaseSetting

Performance optimalisatie → wat kan er gedaan worden als de database langzaam wordt
Om de performance te verbeteren is het onder andere mogelijk om indexen te gebruiken, te vinden op:
https://docs.arangodb.com/IndexHandling/IndexBasics.html
Verder is er een tool voor het meten van performance factoren:
https://docs.arangodb.com/Advanced/Arangob.html

Hoe wordt de structuur van de database aangepast?(toevoegen nieuwe collections enz.)
Het beheren van databases in ArangoDB staat gedocumenteerd op:
https://docs.arangodb.com/Databases/WorkingWith.html

Het maken van collections staat beschreven op:
https://docs.arangodb.com/Collections/index.html

Het is ook mogelijk nieuwe databases te maken via de http api:
https://docs.arangodb.com/HttpDatabase/DatabaseManagement.html

Installatiehandleiding
De windows installer van Arangodb is te vinden op:
https://docs.arangodb.com/Installing/Windows.html

B.14 Terrastore

CRUD acties
De http requests voor CRUD acties zijn te vinden op:
https://code.google.com/archive/p/terrastore/wikis/HTTP_Client_API.wiki

Performance optimalisatie → wat kan er gedaan worden als de database langzaam wordt
-

Hoe wordt de structuur van de database aangepast?(toevoegen nieuwe collections enz.)
Collections worden gemaakt op het moment dat er een document in geplaatst wordt:
https://code.google.com/archive/p/terrastore/wikis/HTTP_Client_API.wiki

Installatiehandleiding
De installatiehandleiding van Terrastore staat op:
https://code.google.com/archive/p/terrastore/wikis/Getting_Started.wiki

B.15 RaptorDB

CRUD acties
Create en Read worden behandelt in:
http://www.codeproject.com/Articles/375413/RaptorDB-the-Document-Store#sampleapp

update en delete worden niet besproken
Performance optimalisatie → wat kan er gedaan worden als de database langzaam wordt
-
Hoe wordt de structuur van de database aangepast?(toevoegen nieuwe collections enz.)
-
Installatiehandleiding
Er is een beknopte installatiehandleiding te vinden op deze site:
http://www.codeproject.com/Articles/375413/RaptorDB-the-Document-Store#sampleapp

B.16 JasDB

CRUD acties
De CRUD acties via http staan beschreven op:
https://github.com/oberasoftware/jasdb/wiki/REST-service

Performance optimalisatie → wat kan er gedaan worden als de database langzaam wordt
-

Hoe wordt de structuur van de database aangepast?(toevoegen nieuwe collections enz.)
-

Installatiehandleiding
De installatiehandleiding is te vinden op:
https://github.com/oberasoftware/jasdb/wiki/Installing-and-configuring-JasDB

B.17 djonDB

CRUD acties
De CRUD acties staan beschreven op:
http://djondb.com/blog/docs/

De voorbeelden in C# staan op:
http://djondb.com/blog/docs/

Performance optimalisatie → wat kan er gedaan worden als de database langzaam wordt
-
Hoe wordt de structuur van de database aangepast?(toevoegen nieuwe collections enz.)
De linux instructies staan op:
http://djondb.com/blog/docs/

Hier staan echter geen instructies voor windows.

Installatiehandleiding
De installatie wordt gedaan vanuit een wizard, het vervolgens opzetten van de database is te vinden op:
http://djondb.com/blog/docs/

B.18 Ejdb

CRUD acties
Alle javascript functies(incl. CRUD acties) staan beschreven op:
http://ejdb.org/doc/ql/ql.html#querying

voor de C# language binder is er enkel een voorbeeld van CREATE en READ te vinden op:
https://github.com/Softmotions/ejdb-csharp

Performance optimalisatie → wat kan er gedaan worden als de database langzaam wordt
-

Hoe wordt de structuur van de database aangepast?(toevoegen nieuwe collections enz.)
-

Installatiehandleiding
De build en installatiehandleidingen zijn te vinden op:
http://ejdb.org/doc/install/building.html#building-windows

Het moet overigens wel in Linux gebuild worden, of er moet een pre-built versie gebruikt worden van:
http://softmotions.com/ejdb/archives/

voor de commandline interface:
http://ejdb.org/doc/cli.html

B.19 Densodb

CRUD acties
-

Performance optimalisatie → wat kan er gedaan worden als de database langzaam wordt
-

Hoe wordt de structuur van de database aangepast?(toevoegen nieuwe collections enz.)
-

Installatiehandleiding
-

B.20 iBoxDB

CRUD acties
Er staat niks expliciet vermeld hoe de CRUD acties gaan, het is wel deels af te leiden uit het voorbeeld te vinden op:
http://www.iboxdb.com/

onder het kopje 'Examples' .

Performance optimalisatie → wat kan er gedaan worden als de database langzaam wordt
-

Hoe wordt de structuur van de database aangepast?(toevoegen nieuwe collections enz.)
-

Installatiehandleiding
-

B.21 BergDB

CRUD acties
Er is een voorbeeld van de CREATE, READ en UPDATE te vinden op:
http://bergdb.com/2015.4/docs/get-started.html

De gedetailleerde docs zijn te vinden op:
http://bergdb.com/2015.4/docs/get-started.html

let op dat dit voor de Java implementatie is, volgens de site werkt de C# implementatie op dezelfde manier.

Performance optimalisatie → wat kan er gedaan worden als de database langzaam wordt
-

Hoe wordt de structuur van de database aangepast?(toevoegen nieuwe collections enz.)
Alles wordt gedaan vanuit de applicatie, zie:
http://bergdb.com/2015.4/javadoc/index.html
(DatabaseBuilder class)

Collecties worden gemaakt op basis van de classes.

Installatiehandleiding
De installatiehandleiding is te vinden op:
http://bergdb.com/2015.4/docs/get-started.html

onder het kopje: 'C# version'.

Bijlage C: Opslagwijze databases en collections

C.1 BaseX

De structuur binnen baseX is als volgt:

- DBMS
    a) Collections*
        - Sub-collections**
            - Documents

* BaseX beschouwt hun databases als lichtgewicht, en vergelijkt deze met collections.
**Het is wel mogelijk om documents te groeperen binnen een collection, vergelijkbaar met wat normaal een collection is.

http://docs.basex.org/wiki/Graphical_User_Interface

http://docs.basex.org/wiki/Databases#Access_Resources

C.2 Couchbase Server

De structuur binnen Couchbase Server is als volgt:

- DBMS
    a) Buckets
        - Buckets/items(als de items een 'type' hebben)
            - Items (Documents/key-value)

Binnen Couchbase server is er sprake van buckets en items. Items kunnen key-value pairs en documents zijn. Een bucket kan gebruikt worden als zowel een database, als een collection. Ik heb echter geen documentatie gevonden waarin stond dat het mogelijk is een bucket in een bucket te hebben. Binnen een bucket worden items opgeslagen. Een item kan zijn een document of een key-value pair. De documents kunnen los toegevoegd worden, of gegroepeerd worden met een attribuut als 'type'.

De keuze tussen los toevoegen en het attribuut type is afhankelijk van hoe variabel de bucket is. Is de bucket als database geïmplementeerd, dan zal een document een attribuut nodig hebben om te filteren. Wordt een bucket als collection ingezet, dan bevindt er binnen deze bucket alleen gegroepeerde data, wat betekent dat een attribuut om te groeperen overbodig is.

http://developer.couchbase.com/documentation/server/4.1/data-modeling/physical-data-modeling.html

C.3 CouchDB

De structuur binnen CouchDB is als volgt:

- DBMS
    a) Collections
        - Documents
            - Views

In CouchDB is een database gelijk aan een collection. In deze collections worden documenten opgeslagen. Het verschil is wel, dat collections/databases data bevatten die volledig onafhankelijk van elkaar zijn(vergelijkbaar met schema's in relationele databases). Het is in CouchDB een conventie, dat documenten die niks met elkaar te maken hebben(een werknemer en openingstijden van een winkel) in dezelfde database worden opgeslagen,

maar (conventie is dat zij) een attribuut bijhouden waaraan zij de herkennen zijn(bijvoorbeeld: 'type').

Het ophalen van documents werkt ook anders in CouchDB. Er wordt vanuit gegaan dat er views gemaakt worden. Deze views worden gebruikt om bijbehorende data te groeperen zoals ze worden opgehaald. Als de documents een type hebben gekregen, kan in de view bijvoorbeeld gecontroleerd worden op dit attribuut. Een view zou er dan bijvoorbeeld als volgt uitzien:

```
Function(doc){
        if(doc.type == 'het type wat je wilt ophalen'){
                //doe eventueel filtering enz.
                emit(returnKey, returnValue);
        }
}
```

De emit kan ook binnen een loop staan, als er meerdere dingen teruggegeven dienen te worden.

Uitleg over werken met CouchDB:
http://guide.couchdb.org/draft/documents.html

Een overview van CouchDB:
http://docs.couchdb.org/en/1.6.1/intro/overview.html

Uitleg over views:
http://docs.couchdb.org/en/1.6.1/couchapp/views/intro.html

C.4 Elasticsearch

De structuur binnen Elasticsearch is als volgt:

- DBMS
    a) Index(soort collection)
        - Type(gezamelijke types binnen een collection(user data, blog data enz.))
            - Documents

De index is een soort collection, er worden documents in opgeslagen. Het is ook mogelijk om meerdere indexen te hebben binnen het DBMS. Een voorbeeld van een index kan zijn medewerkers of winkelgegevens. Let er wel op dat indexen een bepaalde overhead met zich mee brengen op het gebied van schijfruimte, geheugen gebruik en het gebruik van bestandsbeschrijvingen. Een grote index kan dus wenselijker zijn dan veel kleine indexen.

De documents worden binnen een index verdeeld onder types. Een type is vergelijkbaar met een klasse in de applicatie. Als er bijvoorbeeld gekozen wordt om de gehele applicatie in 1 index te plaatsen, dan zouden medewerker en winkelgegevens types zijn. Alle documents binnen een type worden geacht dezelfde attributen te hebben.

De keuze tussen de 2 ligt aan een aantal factoren, namelijk:

- Als er een parent en een child is: gebruik 2 type in 1 index
- Als de documenten dezelfde structuur hebben: zet ze in een type, zo niet zet ze in verschillende indexen

- Als je veel documenten hebt per type, kunnen deze ook in een indexen geplaatst worden
    a) De overhead van indexen wordt dan teniet gedaan
- In andere situaties kan er gekozen worden voor verschillende types binnen 1 index

Uitleg type vs. index:
https://www.elastic.co/blog/index-vs-type

Uitleg basis principes (type, indexen en documents):
https://www.elastic.co/guide/en/elasticsearch/reference/current/_basic_concepts.html

C.5 eXist

De structuur binnen eXist is als volgt:

- DB
    a) Collections
        - Namespaces
            - Documents

Het is binnen eXist niet mogelijk om meerdere databases te runnen, er is hier in ieder geval geen documentatie over gevonden. Alle gevonden documentatie, gaat uit van een enkele database.(Documentatie te vinden op:

http://exist-db.org/exist/apps/doc/).

C.6 Informix

De structuur binnen Informix is hetzelfde als die van MongoDB, alleen gebruiken zij andere termen. Deze termen zijn als volgt:

- DBMS
    a) Databases
        - Table (gelijk aan collection)
            - Record (gelijk aan document)
                - Column (gelijk aan field, attribuut binnen document)

http://www.ibm.com/support/knowledgecenter/SSGU8G_12.1.0/com.ibm.json.doc/ids_json_037.htm?lang=en

C.7 MarkLogic

De structuur binnen MarkLogic is als volgt:

- DBMS
    a) Databases
        - Forests(Collections)
            - Stands (sub-collections)
                - Documents

Het gebruik van meerdere stands binnen een collection is niet verplicht, maar het wordt wel aangeraden omdat het ervoor zorgt dat de database de data efficiënter kan verwerken en het verbeterd de concurrency.

Dit is te vinden op:
http://docs.marklogic.com/guide/concepts/data-management#id_85565
en:

https://docs.marklogic.com/guide/admin/databases#id_60599

C.8 MongoDB

Binnen MongoDB is er sprake van:

- Het DBMS
    a) Databases
        - Collections
            - Documents

Dit is af te leiden uit de uitleg op:
https://docs.mongodb.org/manual/mongo/

C.9 OrientDB

Binnen OrientDB is er sprake van:

- DBMS
    a) Databases
        - Classes(vergelijkbaar met collections/classes uit OO-programming
            - Clusters(een class kan verdeeld worden over meerdere clusters
                - Documents

De structuur is gevonden op:
http://orientdb.com/docs/last/Concepts.html

Hier is ook een uitleg over de classes en clusters terug te vinden.

C.10 RavenDB

De structuur binnen RavenDB is als volgt:

- DBMS
    a) Databases
        - Collections
            - Documents

Collections worden in ravenDB automatisch aangemaakt op basis van het Entiteit dat opgeslagen wordt(de class):
https://ravendb.net/docs/article-page/3.0/csharp/client-api/faq/what-is-a-collection

De aanwezigheid van meerdere databases met daarin collections komt hier naar boven:
http://ravendb.net/docs/article-page/3.0/csharp/studio/accessing-studio

C.11 ArangoDB

De structuur binnen ArangoDB is als volgt:

- DBMS
    a) Databases
        - Collections
            - Documents

Af te leiden uit de glossary:
https://docs.arangodb.com/Glossary/index.html

Bijlage D: Back-up mogelijkheden

D.1 BaseX

In BaseX kan er zowel een back-up gemaakt worden vanuit de GUI als vanuit de console.

Vanuit de GUI kan het maken van de back-up via: Database → Manage. Het inladen van de back-up gaat ook via Database → Manage. Via de console gaat het met de commands CREATE BACKUP [naam van de database] en RESTORE [naam van het back-up bestand].

Het back-up systeem van BaseX maakt een back-up van de gehele database, en verwijderd bij het terugzetten alle documents die niet tot de back-up behoren.

Als er een back-up op maandag 18.00 uur is uitgevoerd, er op maandag 19.00 uur een document wordt toegevoegd en op maandag 20.00 uur de backup wordt hersteld, is het bestand van 19.00 uur verwijderd.
http://docs.basex.org/wiki/Backups

D.2 eXist

In eXist zijn er 2 mogelijkheden om een backup te maken van de database, via de client en via de server.

Via de client kan de client bepalen hoe het proces van de backup verloopt. Dit kan via de Java admin client of via een command line. De server wordt niet gelocked gedurende het maken van de back-up. Dit heeft als gevolg, dat gebruikers data kunnen aanpassen terwijl een back-up gemaakt wordt. Dit kan betekenen dat afhankelijkheden tussen verschillende resources niet juist worden meegenomen in de backup. Deze manier van back-ups wordt daarom niet aangeraden.

Via de server worden back-ups meestal getriggerd door de job scheduler van eXist-db, maar ze kunnen ook via de web interface getriggerd worden. Bij een back-up vanaf de server, wordt de database geswitched naar een beschermde service voordat de back-up start. eXist-db zal wachten tot alle transacties die nog verwerkt moeten worden, verwerkt zijn voordat hij naar beschermde mode gaat. Vervolgens wordt de back-up gemaakt. Alle requests van clients die binnen komen terwijl de back-up bezig is, worden in een queue gezet totdat de back-up klaar is. Het wordt dan ook aangeraden om deze manier van back-ups maken te gebruiken.

Het herstellen van de data uit de back-up kan via de Java client, en via de command line. Wat belangrijk is om te weten bij het herstellen is, dat alleen data die in de back-up staat aangepast wordt. Data wat na het maken van de back-up is toegevoegd wordt niet verwijderd of aangepast. Een voorbeeld: Als er op maandag 18.00 uur een back-up is gemaakt, om 18.30 uur een bestand is toegevoegd, om 19.00 een bestand wat ook in de back-up voorkwam is gewijzigd en om 20.00 uur de back-up wordt hersteld, zal de herstelde database het toegevoegde bestand van 18.30 uur bevatten, en de oude versie van het bestand dat om 19.00 uur gewijzigd was.

Het is overigens wel mogelijk om de gehele database terug te zetten naar de back-up(dus in het vorige voorbeeld: alles terug naar de staat van 18.00 uur), maar dit moet expliciet aangegeven worden bij het herstellen.

Het is via de server ook mogelijk om incrementele back-ups te maken.
http://exist-db.org/exist/apps/doc/backup.xml?q=back-up&field=all&id=D2.2.9

D.3 Informix

Het is binnen Informix op 2 manieren mogelijk om back-ups te maken, on-bar en ontape. De verschillen tussen beide zijn te vinden op:
http://www.ibm.com/support/knowledgecenter/SSGU8G_12.1.0/com.ibm.bar.doc/ids_bar_177.htm?lang=en

http://www.ibm.com/support/knowledgecenter/SSGU8G_12.1.0/com.ibm.bar.doc/ids_bar_170.htm?lang=en

http://www.ibm.com/support/knowledgecenter/SSGU8G_12.1.0/com.ibm.bar.doc/ids_bar_169.htm?lang=en

D.4 MarkLogic

Binnen MarkLogic worden back-ups en restore acties als transacties gezien. De back-ups/restores garanderen dus de consistentie van de data. De database wordt echter niet gelocked op het moment dat de back-up/restore bezig is. In plaats daarvan, worden alle wijzigingen die gemaakt worden gedurende de back-up/restore gewoon niet meegenomen in de back-up/restore. Administratieve taken(drop, delete, clear enz.) kunnen niet uitgevoerd worden gedurende een back-up/restore.

Back-ups kunnen handmatig uitgevoerd worden, maar kunnen ook ingepland worden(dagelijks, wekelijks, eenmalig enz.). Verder is het ook mogelijk om incrementele back-ups te maken.
http://docs.marklogic.com/guide/admin/backup_restore

D.5 MongoDB

MongoDB ondersteunt 4 manieren voor het maken van back-ups, waarvan er 2 altijd mogelijk zijn, deze zullen hier behandeld worden.

Het is binnen mongodb mogelijk om een back-up te maken met behulp van utilities uit mongodb. Deze utilities zijn:

- mongodump
- mongorestore

Deze utilities zijn te gebruiken vanuit de command line van mongodb(mongod) en deze worden gebruikt(aangeraden) voor het back-uppen en restoren van kleine hoeveelheden data, bijvoorbeeld een collection. Voor volledige back-ups adviseert MongoDB om gebruik te maken van het file system.

Voor grotere back-ups, bijvoorbeeld van de hele database/alle databases, wordt aangeraden om gebruik te maken van file system snapshots.

https://docs.mongodb.org/manual/tutorial/backup-and-restore-tools/

https://docs.mongodb.org/manual/tutorial/backup-with-filesystem-snapshots/

https://docs.mongodb.org/manual/core/backups/

D.6 OrientDB

In Orientdb wordt de database gelocked op het moment dat er een back-up gemaakt wordt, deze lock geld alleen voor write operaties. De back-ups worden gemaakt met de BACKUP DATABASE commando, en deze back-ups worden hersteld met de RESTORE DATABASE commando.

Naast de commando's is het ook mogelijk om automatisch back-ups te laten maken door middel van een server plugin.
http://orientdb.com/docs/last/Backup-and-Restore.html

http://orientdb.com/docs/last/Automatic-Backup.html

D.7 RavenDB

In RavenDB kunnen databases op 2 manieren geback-upt worden. Beide manieren kunnen uitgevoerd worden terwijl de database online staat en requests verwerkt. De manieren zijn:

- Door gebruik te maken van een bestaande (enerprise) back-up solution.
    a) RavenDB support VSS back-ups, de back-up solution moet alleen geconfigureerd worden om een back-up van RavenDB's data directory te maken.
- Door het back-up en restore systeem van RavenDB te gebruiken.
    a) Gedurende het begin van het maken van de back-up blijft de database online en kan hij reageren op read en write requests.

RavenDB vertrouwt op OS services om data opslag en back-ups te beheren. Deze services zijn altijd forward compatible, maar niet altijd backward compatible.

Als de database embed in de applicatie is, kan de back-up gemaakt worden door de methode DocumentDatabase.Mainenance.StartBackup() aan te roepen. Als er gebruik wordt gemaakt van de client/server mode, kan Raven.Backup.exe gebruikt worden om back-ups te maken/plannen. Er mag overigens maar 1 back-up operatie tegelijk runnen.

Het herstellen van een 'SYSTEM' database is een offline operatie, en kan dus alleen uitgevoerd worden als de instantie van RavenDB runt. Het herstellen van 'non-SYSTEM' databases is daarentegen een online operatie, wat inhoudt dat de server waarop de herstel wordt uitgevoerd moet runnen.

Het is ook niet mogelijk om een bestaande database directory over te schrijven bij het herstellen van de back-up. Op het moment dat er een bestaande directory overschreven moet worden, dient de database offline gehaald te worden, de directory verwijderd te worden en de herstel daarna uitgevoerd te worden.
https://ravendb.net/docs/article-page/3.0/csharp/server/administration/backup-and-restore

D.8 ArangoDB

In AranangoDB is het mogelijk om een dump van de database te maken met behulp van de commando arangodump --output-directory "naam van de directory". Met deze commando worden alle 'non-system' collections van de default database gedumpt. De dump zal niet lukken op het moment dat de directory al bestaat. Dit wordt gedaan om te voorkomen dat er

perongeluk een dump overschreven wordt. Als je toch wilt dat de eerdere dump overschreven wordt, dien je dit aan te geven door --overwrite true toe te voegen aan de commando.

Om specifieke databases, (system)collecties of endpoints te dumpen, zijn er toevoegingen aan het commando. Deze zijn te vinden op:
https://docs.arangodb.com/2.1/Arangodump/README.html

Het herstellen van de dump directory werkt op bijna dezelfde manier als het dumpen van de data. De grootste verschillen zijn dat het commando arangorestore --input directory "naam van directory" is. Meer informatie hierover op:
https://docs.arangodb.com/2.1/Arangorestore/README.html

Het is in ArangoDB mogelijk om taken in te plannen via de task manager. Met behulp van deze task manager is het dus ook mogelijk om het back-up commando te automatiseren.
https://docs.arangodb.com/ModuleTasks/index.html

D.9 Couchbase Server

Het maken van back-ups in Couchbase Server is op 2 manieren te doen, namelijk:

- Via de cbbackup commando
- Door het kopiëren van de bestanden

Het wordt aangeraden om de cbbackup commando te gebruiken, daar deze een back-up van alles kan maken, inclusief configuratie van de database.
http://developer.couchbase.com/documentation/server/4.1/backup-restore/backup-cbbackup.html

het herstellen van de back-up gaat via de command cbrestore. Hierbij hoeft de configuratie van de nieuwe database niet hetzelfde te zijn als die van de back-up.
http://developer.couchbase.com/documentation/server/4.1/backup-restore/restore-cbrestore.html

verder is het mogelijk om 2 soorten incrementele back-ups te maken. Dit zijn:

- Differentiële incrementele back-ups
    a) Deze bevatten alle data die anders is ten opzichte van de vorige back-up.
        - Worden sneller gemaakt, laden wel langzamer in daar alle differentiële back-ups na een full back-up 1 voor 1 ingeladen moeten worden.
- Cumulatieve incrementele back-ups
    a) Deze bevatten alle data die gewijzigd zijn sinds de laatste full back-up.
        - Worden langzamer gemaakt omdat ze meer data bevatten, maar laden sneller in aangezien er maar 1 nodig is na de laatste full back-up.

Deze 2 manieren van incrementeel back-ups maken kunnen ook gecombineerd worden.
http://developer.couchbase.com/documentation/server/4.1/backup-restore/incremental-backup.html

D.10 CouchDB

-
Geen documentatie

D.11 Elasticsearch

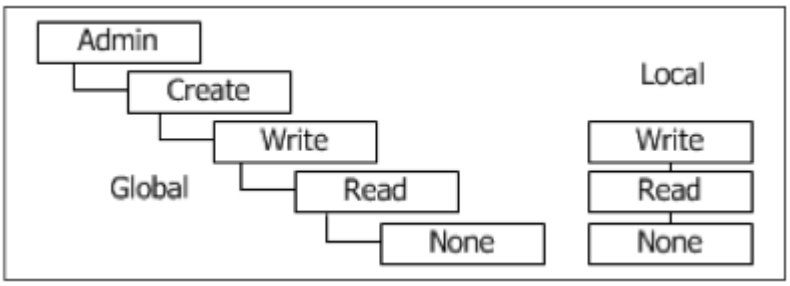
Het is in elasticsearch mogelijk om snapshots te maken van 1 of meerdere indexen(databases/collections). Het is ook mogelijk deze snapshots op andere clusters weer in te laden door de repository van snapshots toe te voegen aan de cluster.

https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-snapshots.html

Bijlage E: Rechten en rollen

E.1 BaseX

Het is in BaseX mogelijk om gebruikers aan te maken, en deze rechten te geven. De hiërarchie in deze rechten ziet er als volgt uit:



Een recht wat bovenin staat kan alles wat onder hem zit ook. Op het moment dat er een gebruiker aangemaakt wordt, heeft deze geen rechten, deze moeten handmatig toegekend worden. Het geven van permissies is overigens alleen relevant als er gebruik gemaakt wordt van de client/server mode. Alle andere modes(GUI en standalone) worden uitgevoerd met administrator recht.

http://docs.basex.org/wiki/User_Management

E.2 eXist

Het is binnen eXist mogelijk om gebruikers te maken, en deze read, write of execute rechten te geven op resources/collections. Dit is gebaseerd op de Unix manier van rechten. De rechten kunnen gegeven worden op:

- document niveau
    a) openen, toevoegen, overschrijven enz.
- document kopieer/beweeg niveau
    a) het kopiëren/verplaatsen van documents
- collection niveau
    a) toevoegen/verwijderen van collections, kopieren/bewegen van collections

http://exist-db.org/exist/apps/doc/security.xml

E.3 MarkLogic

Binnen MarkLogic kunnen users en roles gedefinieerd worden. Per document kan aangegeven welke rollen welke rechten op die document hebben. De rechten kunnen zijn:

- Read
- Insert
- Update
- Execute

Naast de rollen die de gebruiker aanmaakt, zijn er 2 voor gedefinieerde rollen, namelijk: admin en security. De admin kan alles in het systeem, inclusief het benaderen van de admin interface(de interface waar de dbms beheert kan worden, databases/collections aangemaakt kunnen worden, back-uppen en herstellen van databases enzovoorts).

De securityrol kan alles in kader van security uitvoeren, zoals het maken van users, rollen enzovoorts.

http://docs.marklogic.com/guide/admin/security

http://docs.marklogic.com/guide/security/role

E.4 MongoDB

Binnen MongoDB is het ook mogelijk om users en rollen aan te maken. Per rol is het aan te geven welke privileges zij hebben. Een rol heeft de rechten alleen binnen de database waar de rol is aangemaakt.

Om users en rollen te maken, moet er eerst een admin user aangemaakt worden. Dit kan gedaan worden door MongoDB eerst te starten zonder authenticatie controle.

https://docs.mongodb.org/manual/security/

https://docs.mongodb.org/manual/administration/security-checklist/

E.5 OrientDB

Binnen OrientDB is het mogelijk om users en roles te maken. Deze users en roles worden gemaakt op database niveau. Een rol bevat een mode en rechten. Een mode kunnen dingen zijn als'Deny all but' of 'Allow all but', terwijl een recht bijvoorbeeld het volgende kan zijn 'Database = read' of 'Database = all'.

http://orientdb.com/docs/last/Database-Security.html

Naast security op database niveau, is het ook mogelijk users op server(DBMS) niveau te maken. Op server niveau kunnen de gebruikers 'resources'(= rechten) krijgen. Deze rechten kunnen bijvoorbeeld zijn: het maken of droppen van databases, zien van alle beschikbare databases op de server enz.

http://orientdb.com/docs/last/Server-Security.html

E.6 RavenDB

Het is in RavenDB wordt authenticatie gedaan door middel van OAuth, of met behulp van Windows authentication. Dit kan zelf besloten worden.
http://ravendb.net/docs/article-page/3.0/csharp/client-api/how-to/work-with-authentication

De autorisatie kan ingesteld worden per gebruiker/groep gebruikers. Dit is echter alleen mogelijk in de commerciële variant van RavenDB, niet in de open source versie.
http://ravendb.net/docs/article-page/3.0/csharp/server/configuration/authentication-and-authorization

Verder is het via de bundle: authorization mogelijk om heel specifiek rechten te maken, en toe te kennen aan rollen.
http://ravendb.net/docs/article-page/3.0/Csharp/server/bundles/authorization

E.7 ArangoDB

Momenteel alleen of volledig toegang of geen toegang. Het is nog niet mogelijk om alleen recht op bepaalde handelingen/collecties te hebben.
https://docs.arangodb.com/ConfigureArango/Authentication.html

E.8 Couchbase Server

In couchbase server is er sprake van 1 administrator op DBMS niveau, en eventueel 1 read-only administrator op ditzelfde niveau. De administrator kan alle couchbase server functionaliteiten uitvoeren, terwijl de read-only admin alles kan lezen, maar niks kan aanpassen.

Verder is het mogelijk om per bucket een wachtwoord op te geven.
http://developer.couchbase.com/documentation/server/4.1/security/security-intro.html

E.9 Elasticsearch

Elasticsearch heeft geen features voor authenticatie of autorisatie.
https://www.elastic.co/blog/found-elasticsearch-in-production#security

https://www.elastic.co/blog/found-elasticsearch-as-nosql

Bijlage F: Transactie support

F.1 BaseX

BaseX biedt in de client/server architectuur volledige ACID support op transacties.
http://docs.basex.org/wiki/Transaction_Management

http://basex.org/products/

F.2 eXist

er is geen documentatie beschikbaar over de mate waarop eXist omgaat met transacties

F.3 MarkLogic

MarkLogic support de volledige set van ACID op transacties.

http://docs.marklogic.com/guide/concepts/overview#id_29193

F.4 MongoDB

Niet ACID, elk operatie op 1 document is atomair. Als er een operatie op meerdere documents gedaan wordt, wordt dit gezien als meerdere atomaire operaties. Het is wel mogelijk een operatie op meerdere documents te isoleren door middel van het $isolated commando.

Het is echter niet zo dat het alles gaat door of niks gaat door is. Als error zich voordoet, (als $isolated gebruikt wordt bij meerdere documents in 1 transactie) betekent dit dat 1 van de punten gefaald heeft, niet allemaal.

https://docs.mongodb.org/manual/core/write-operations-atomicity/

https://docs.mongodb.org/manual/tutorial/perform-two-phase-commits/

https://docs.mongodb.org/manual/faq/fundamentals/#does-mongodb-support-transactions

F.5 OrientDB

OrientDB support ACID op transacties.

http://orientdb.com/docs/last/Transactions.html

F.6 RavenDB

Operaties op documenten en batches van operaties op een set documenten zijn ACID. Als er indexen bij betrokken zijn is het BASE(de indexen zijn BASE).

http://ravendb.net/docs/article-page/3.0/Csharp/client-api/faq/transaction-support

Bijlage G: Sturen op performance

G.1 BaseX

Secundaire indexen:
BaseX laat niet zien hoeveel documents er gelezen zijn om tot het resultaat te komen. Verder is het zo, dat BaseX automatisch indexen aanmaakt op element niveau.

Het is echter niet mogelijk om verdere indexen te plaatsen(op element niveau), of deze indexen te optimaliseren.

Het volgende is gedaan om te query'en:

Inhoud database:
De volgende inhoud had de database op het moment van query'en:

```
<person>
  <name>Xavyr</name>
  <hobby>Niks</hobby>
</person>
<person>
  <name>Marc</name>
  <hobby>Iets</hobby>
</person>
```

Hierop is de volgende query uitgevoerd:

```
for $Person in db:open('xml_test_indexes')//person where $Person//name='Xavyr' return $Person
```

Dit leverde de volgende output op:

```
<person>
  <name>Xavyr</name>
  <hobby>Niks</hobby>
</person>
```

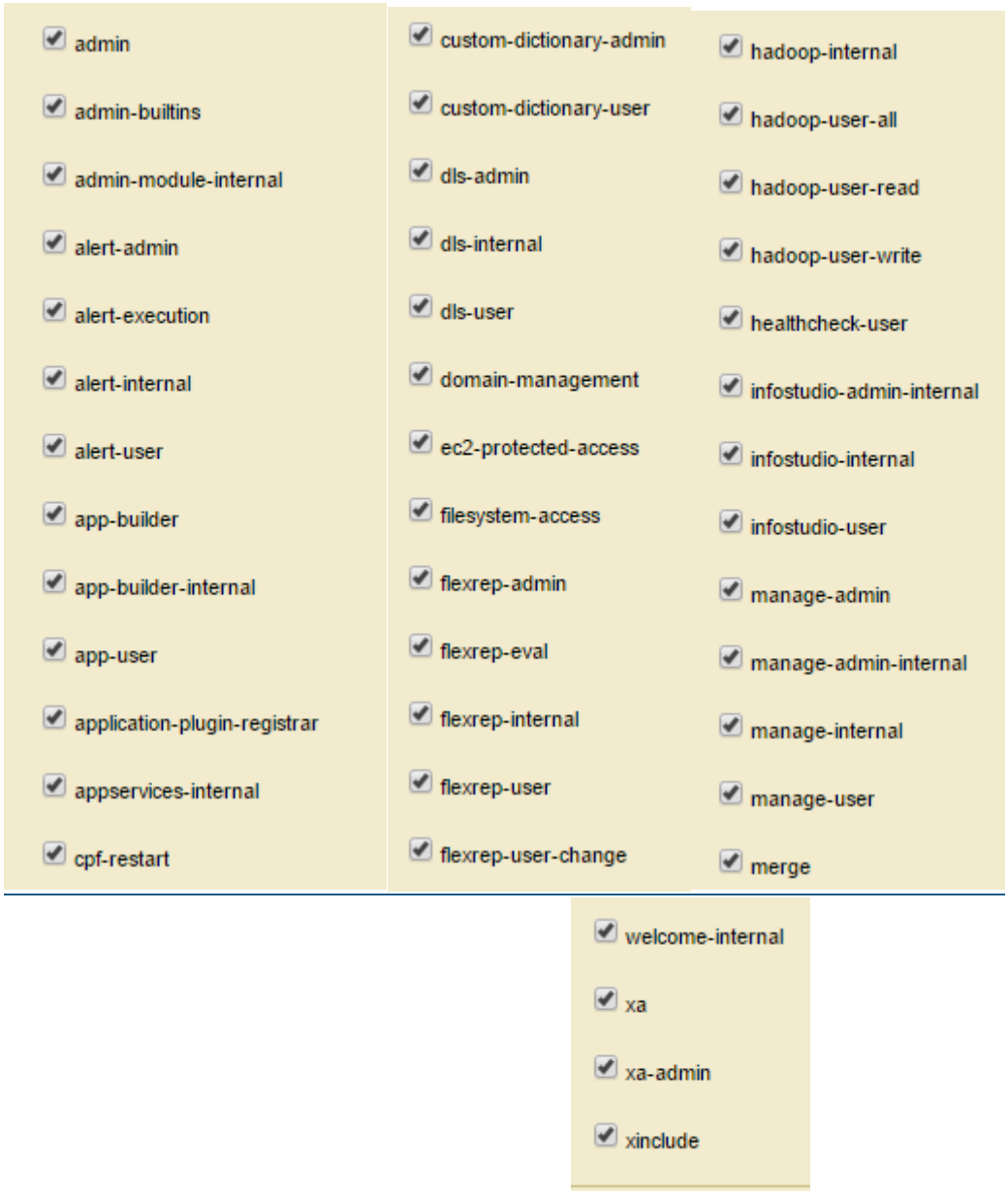
G.2 MarkLogic

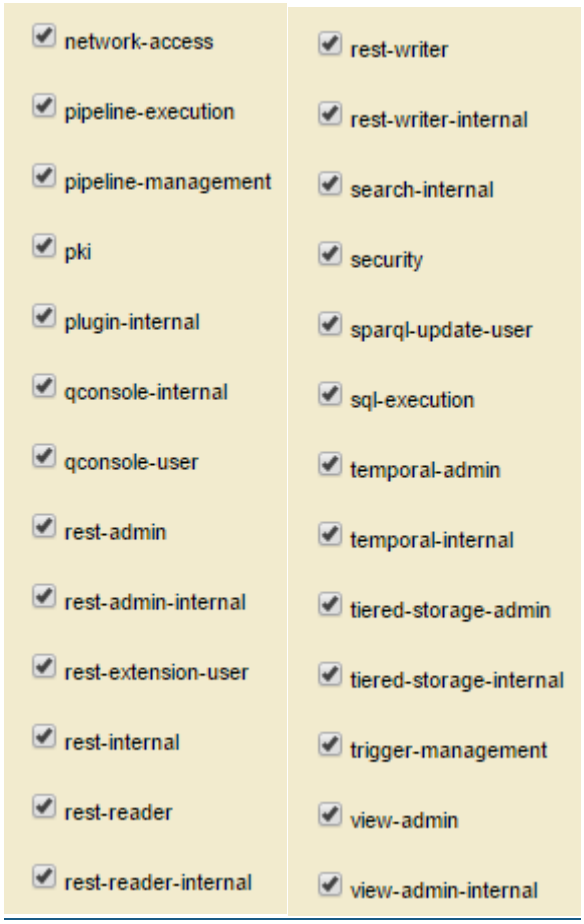
Er is besloten niet verder te werken met MarkLogic, omdat ik het niet voor elkaar kreeg deze database werkend te krijgen. Ik heb het volgende geprobeerd om met de database te communiceren:

Ik wilde met de database communiceren met behulp van de rest api die MarkLogic aanbied. Hiervoor is het voorbeeld op http://docs.marklogic.com/guide/rest-dev/service#id_20421 gevolgd(stap 1). Om de database "rest-example" te maken heb ik curl geïnstalleerd en heb ik de request zoals in stap 1 te zien is overgetypt en uitgevoerd. Het enige wat ik anders heb gedaan is de username en password aangepast(namelijk de username en password van mijn admin account). De volgende request is uitgevoerd:

```
curl --anyauth --user xavyr:password -i -X POST \
     -d'{"rest-api":{"name":"rest-example"}}' \
     -H "Content-type: application/json" \
```

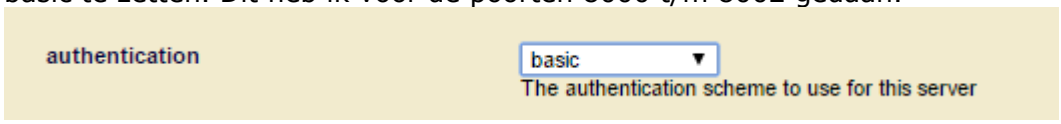
Toekennen alle rechten aan admin:
De eerste poging retourneerde de http status 401: Unauthorized en een status 400: Bad request. Vervolgens heb ik de admin expliciet alle rechten toegekend, en heb ik zijn default rollen op alle rollen die te maken hebben met rest-services gezet. De rechten zijn in de onderstaande afbeeldingen te vinden.


admin
admin-builtins
admin-module-internal
alert-admin
alert-execution
alert-internal
alert-user
app-builder
app-builder-internal
app-user
application-plugin-registrar
appservices-internal
cpf-restart
custom-dictionary-admin
custom-dictionary-user
dls-admin
dls-internal
dls-user
domain-management
ec2-protected-access
filesystem-access
flexrep-admin
flexrep-eval
flexrep-internal
flexrep-user
flexrep-user-change
hadoop-internal
hadoop-user-all
hadoop-user-read
hadoop-user-write
healthcheck-user
infostudio-admin-internal
infostudio-internal
infostudio-user
manage-admin
manage-admin-internal
manage-internal
manage-user
merge
welcome-internal
xa
xa-admin
xinclude
role name (capability)
[Keep]
admin (execute)
[add] rest-admin execute
[add] rest-writer read
[add] rest-reader read
[add] read
more permissions



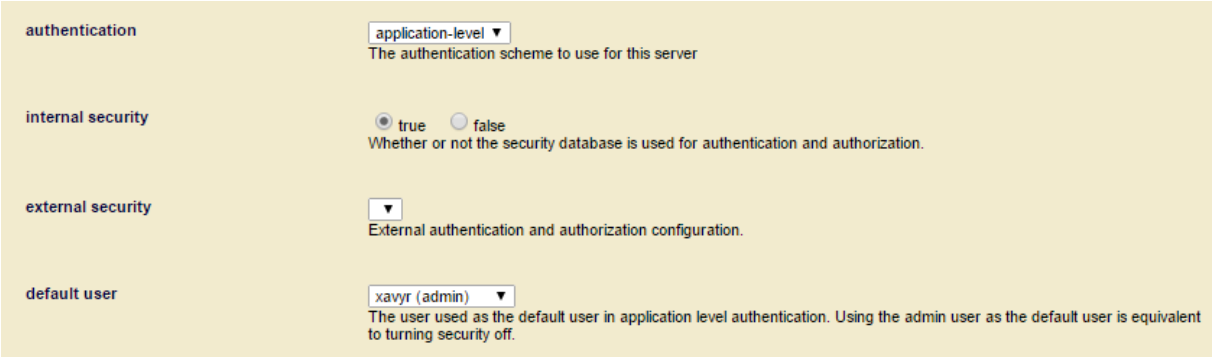
Na het toekennen van deze rechten, heb ik weer geprobeerd de request uit te voeren. Nog steeds kreeg ik als response de statussen 401 en 400.

Security op basic authentication:
Na enig tijd googelen las ik dat het kan werken door de authentication manier van digest op basic te zetten. Dit heb ik voor de poorten 8000 t/m 8002 gedaan.



Weer heb ik geprobeerd de request uit te voeren, maar weer hetzelfde resultaat.

Disabelen security op alle poorten:
Vervolgens heb ik geprobeerd de security op de poorten 8000 t/m 8002 uit te schakelen. Dit is gedaan door de volgende instellingen te hanteren voor poorten 8000 t/m 8002.



Maken nieuwe group:
Hierna heb ik geprobeerd een nieuwe 'group' aan te maken op poort 8003. Ook hier heb ik alle vormen van security uitgeschakeld, zoals in de bovenstaande stappen. Vervolgens heb ik geprobeerd de request op deze poort af te vuren. Weer met hetzelfde resultaat.

Postman:
Om uit te sluiten dat ik de request verkeerd deed, heb ik de applicatie postman geïnstalleerd. Via postman heb ik dezelfde request op alle poorten uitgeprobeerd. Ook hier kreeg ik als response status 401 en 400.

.NET request:
Ook heb ik geprobeerd de request te sturen vanuit .NET, met behulp van HttpWebRequest klasse. Ook dit gaf status 401 als response.

Mede afstudeerder:
Tot slot heb ik gevraagd of een mede afstudeerder het wilde proberen, om te kijken of het hem wel zou lukken. Ook hij heeft alle door mij ondernomen manieren uitgeprobeerd, zonder het gewenste resultaat te behalen.

Zonder credentials:
Omdat de security op de server volledig was uitgeschakeld, heb ik geprobeerd de request te sturen zonder credentials(het idee erachter was, dat MarkLogic misschien alsnog authenticatie toepaste omdat er credentials werden opgestuurd). De request zag er als volgt uit:

```
1. curl -i -X POST \
2.      -d'{"rest-api":{"name":"rest-example"}}' \
3.      -H "Content-type: application/json" \
```

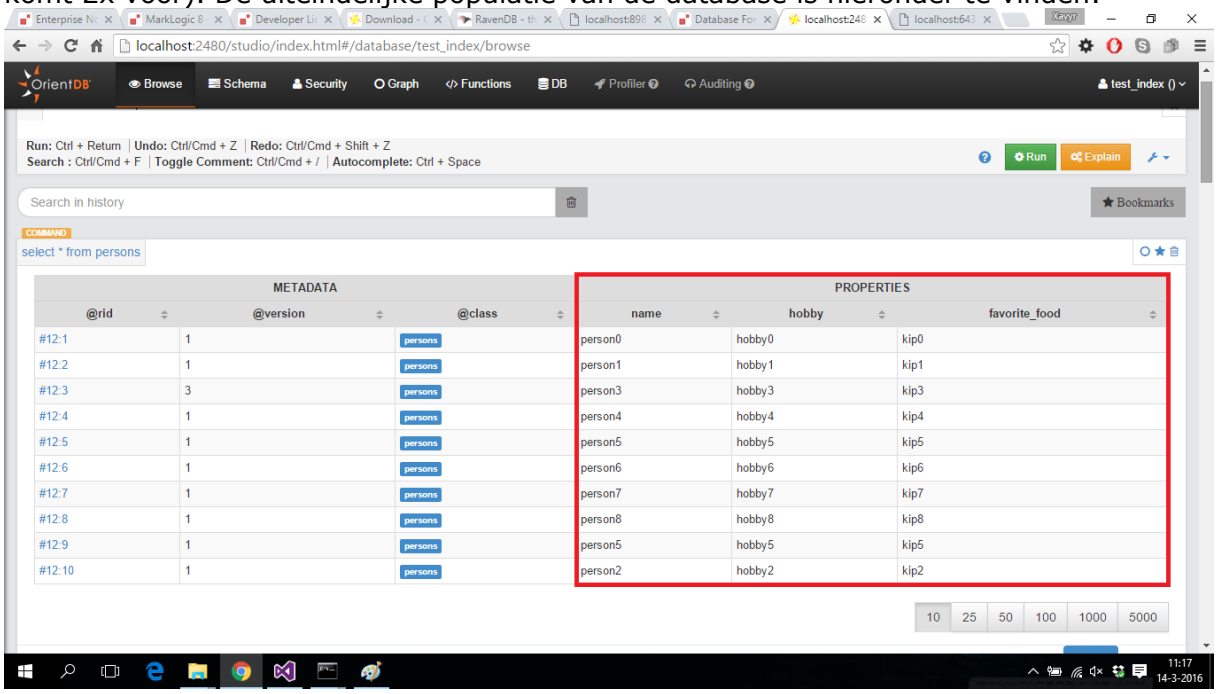
Dit resulteerde nog steeds in de responses 401 en 400.

G.3 OrientDB

Secundaire indexen:

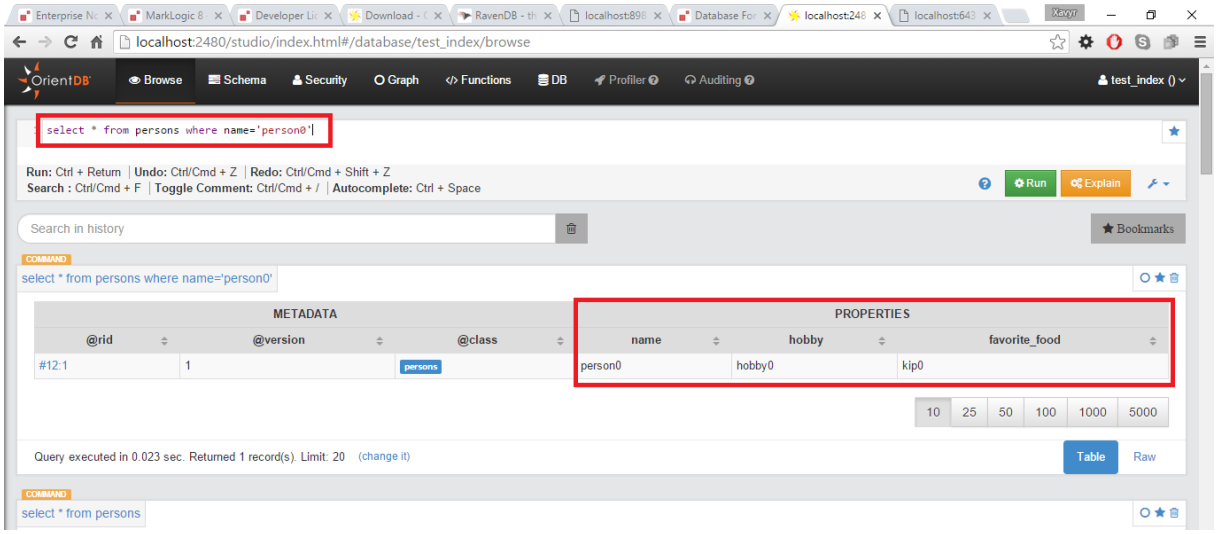
In OrientDB is een database aangemaakt, daarin zijn een paar documents toegevoegd, en zijn er query's en indexen op uitgeprobeerd. Deze uitvoering en resultaten hiervan zijn hieronder te vinden:

Toevoegen documents:
Er zijn 10 documents toegevoegd met allemaal de volgende attributen: name, hobby, favorite_food. De waardes hiervan waren: 'person*', 'hobby*' en 'kip*' (*=0 t/m 8, en 5 komt 2x voor). De uiteindelijke populatie van de database is hieronder te vinden:

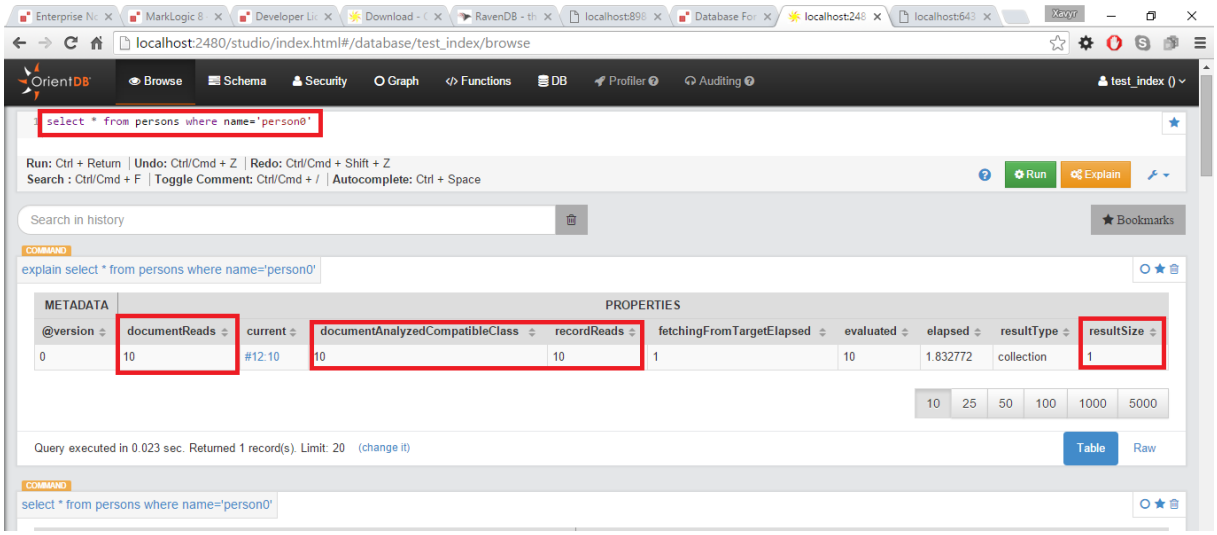


Uitvoeren select query's zonder index:
Volgens zijn er select query's uitgevoerd, waarbij gezocht werd op de naam(name). Hierbij is er gezocht op een unieke naam in de lijst, namelijk 'person0' en een naam die vaker voorkomt, 'person5'. Bij de resultaten is gekeken hoeveel records de database heeft gelezen om tot het resultaat te komen.

De resultaten van het ophalen van person0 zag er, zoals verwacht, als volgt uit:

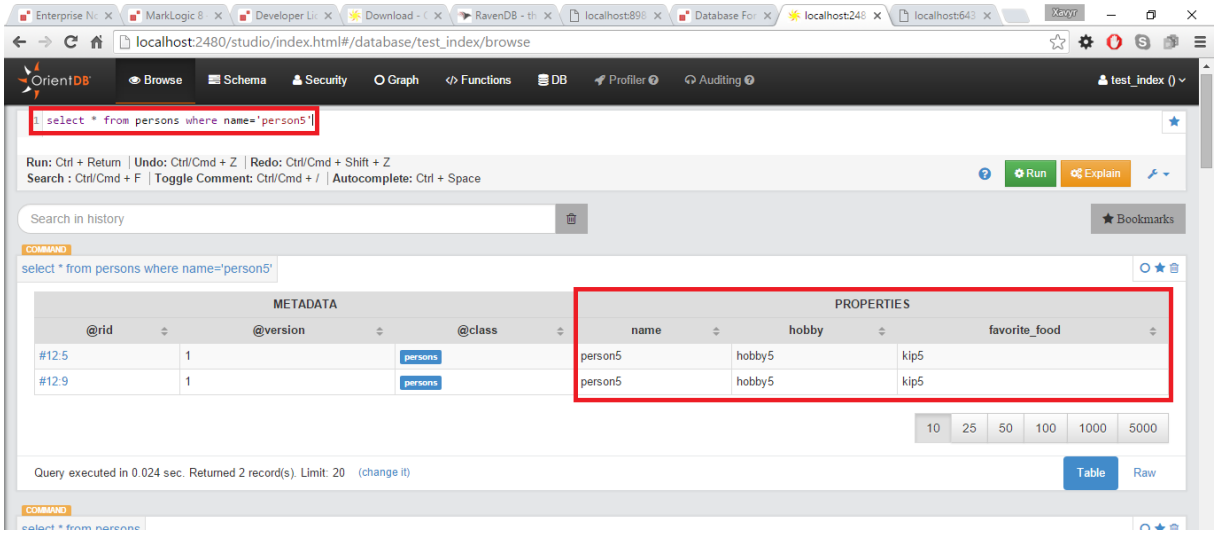


De analyse die hierbij hoorde was:

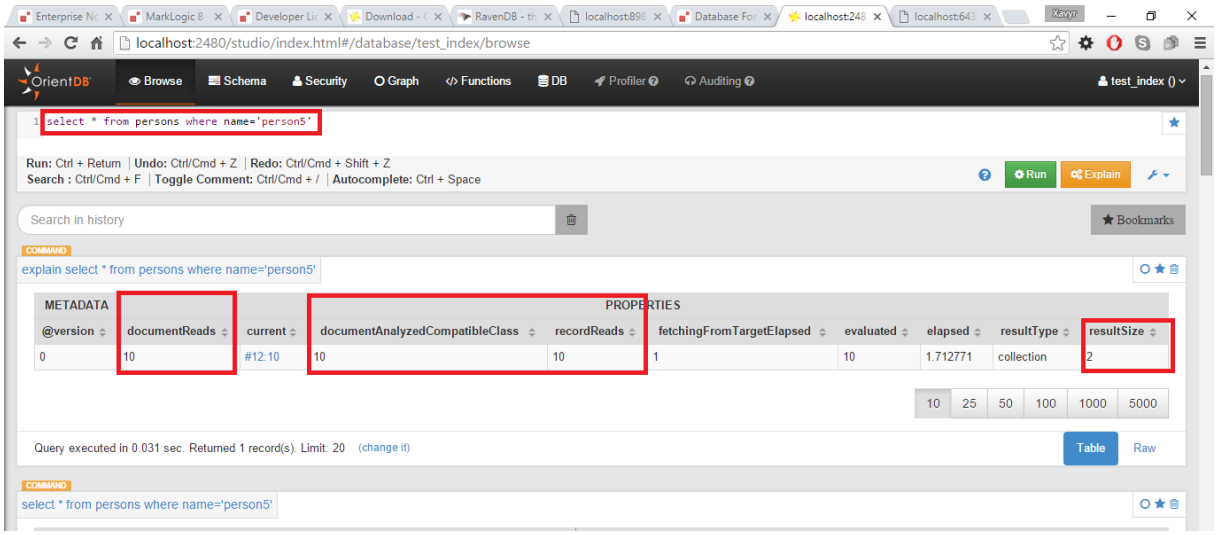


Zoals op de met rood omkaderde onderdelen van de afbeelding te zien, heeft de database 10 documents gelezen, 10 documents geanalyseerd en 10 records gelezen, waarbij het resultaat 1 document was.

Voor het ophalen van person5(meerdere records) zagen de resultaten er als volgt uit:

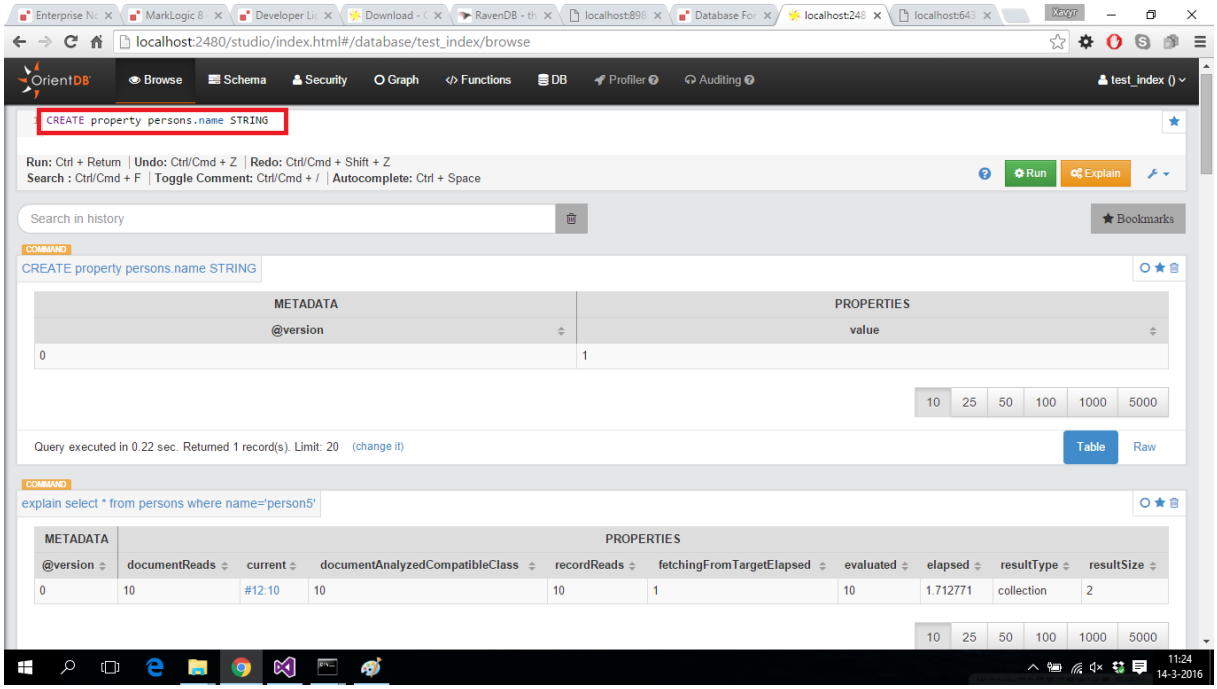


Met de analyse:

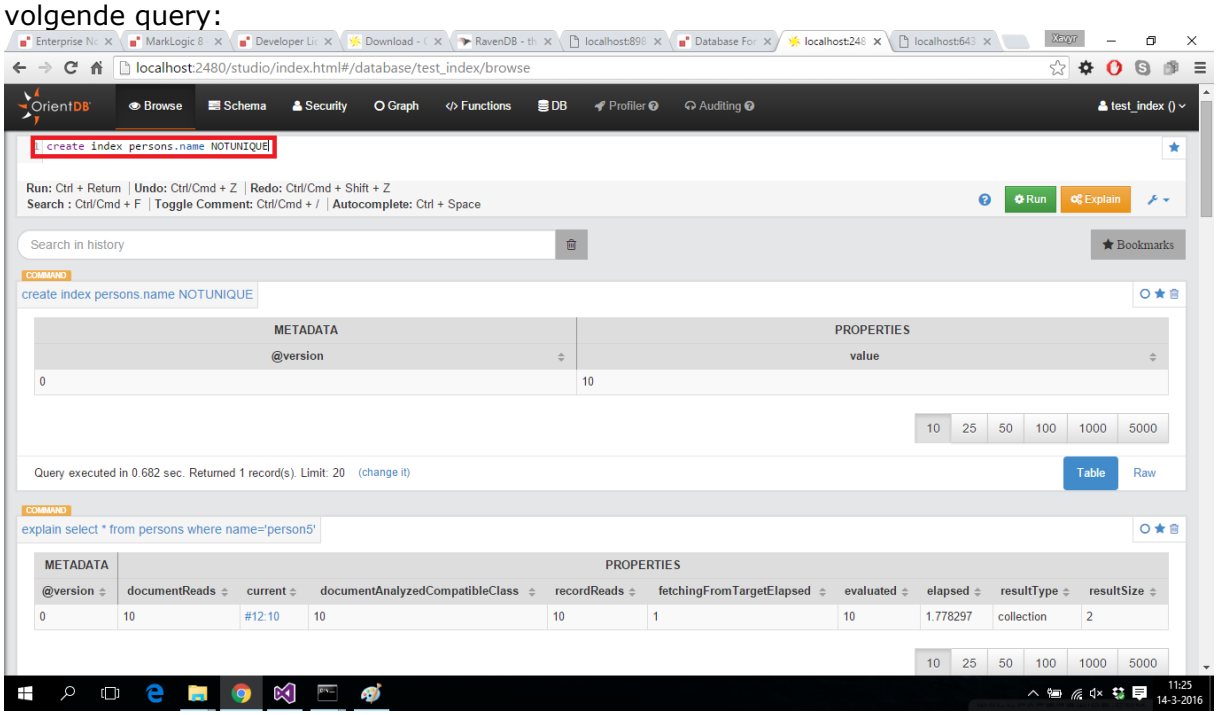


Ook hier zijn er 10 documents gelezen en geanalyseerd. De result bestond uit 2 documents.

Toevoegen property:
Om een index op een attribuut toe te voegen, moet de class waarop je de index wilt een property krijgen. Het toevoegen van de property name op de class persons ging als volgt:

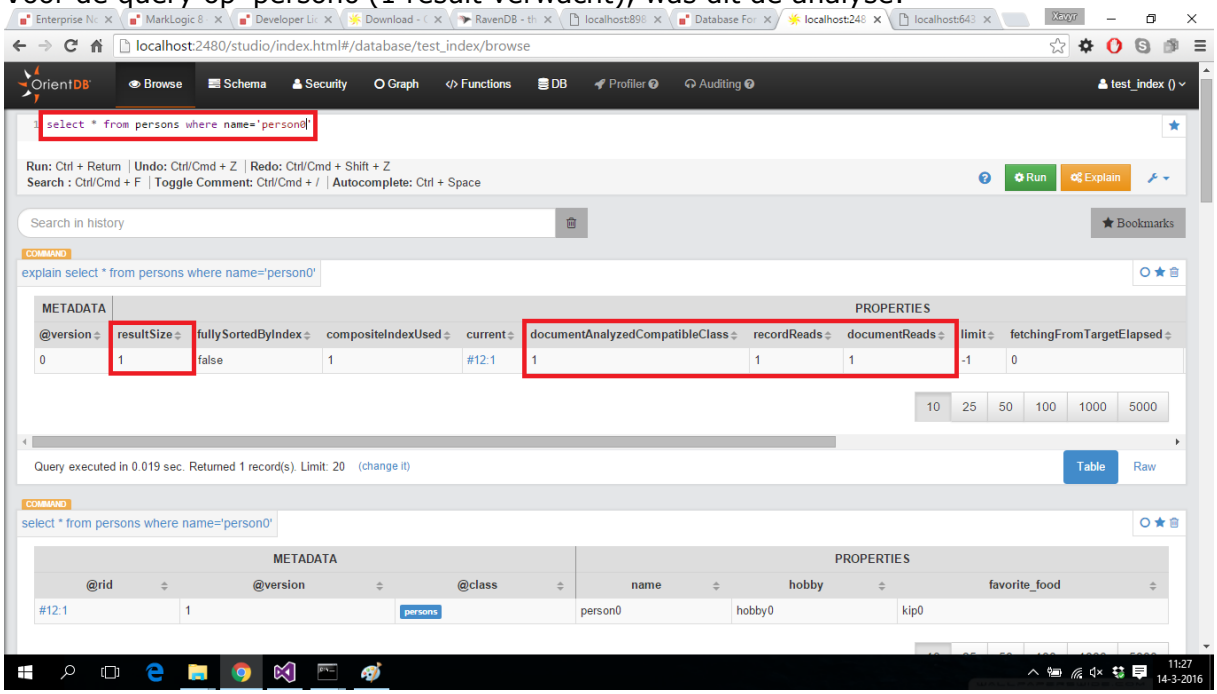


Toevoegen index:
Nadat de property was aangemaakt, is er een index op deze property gezet, met de volgende query:



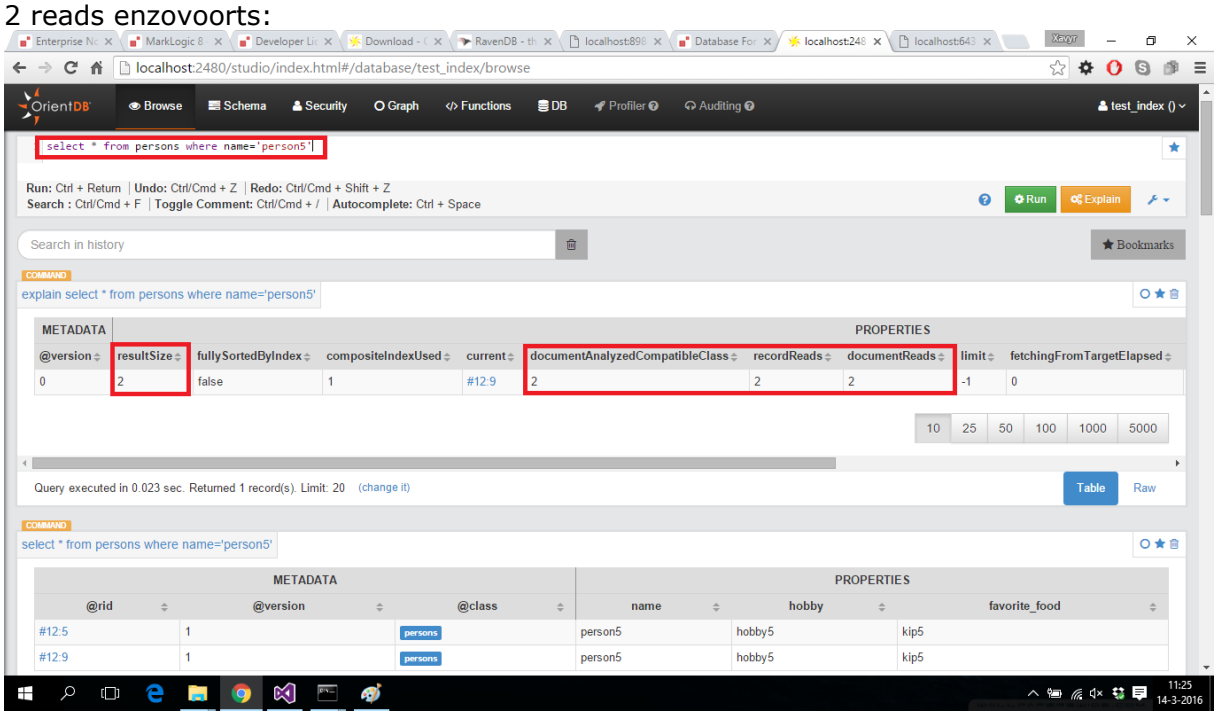
Uitvoeren select query's met index:
Vervolgens zijn de eerder uitgevoerde query's opnieuw uitgevoerd. De resultaten waren hetzelfde, maar de analyses verschilde. Deze zagen er als volgt uit:

Voor de query op ‘person0’(1 result verwacht), was dit de analyse:

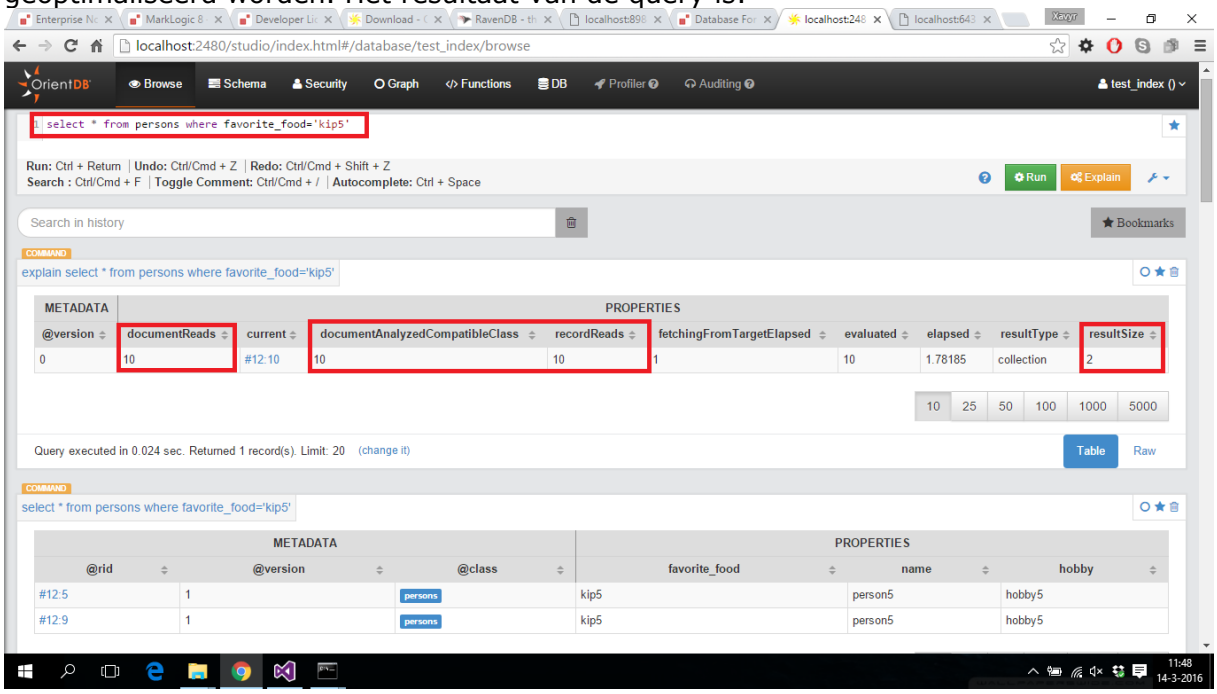


Zoals te zien was het resultaat bij deze query inderdaad weer 1, alleen is de hoeveelheid documenten gelezen, records gelezen en geanalyseerde documents van 10 teruggebracht naar 1.

Voor de query op ‘person5’(2 resultaten), gebeurde er hetzelfde, maar dan met 2 resultaten, 2 reads enzovoorts:



Er is voor de zekerheid nog een query uitgevoerd, waarbij gezocht werd op de favorite_food attribuut. Deze is uitgevoerd om aan te tonen dat alleen de attributen met index geoptimaliseerd worden. Het resultaat van de query is:



Zoals te zien worden er hier nog gewoon 10 documents gelezen op het moment dat er maar 2 terugkomen.

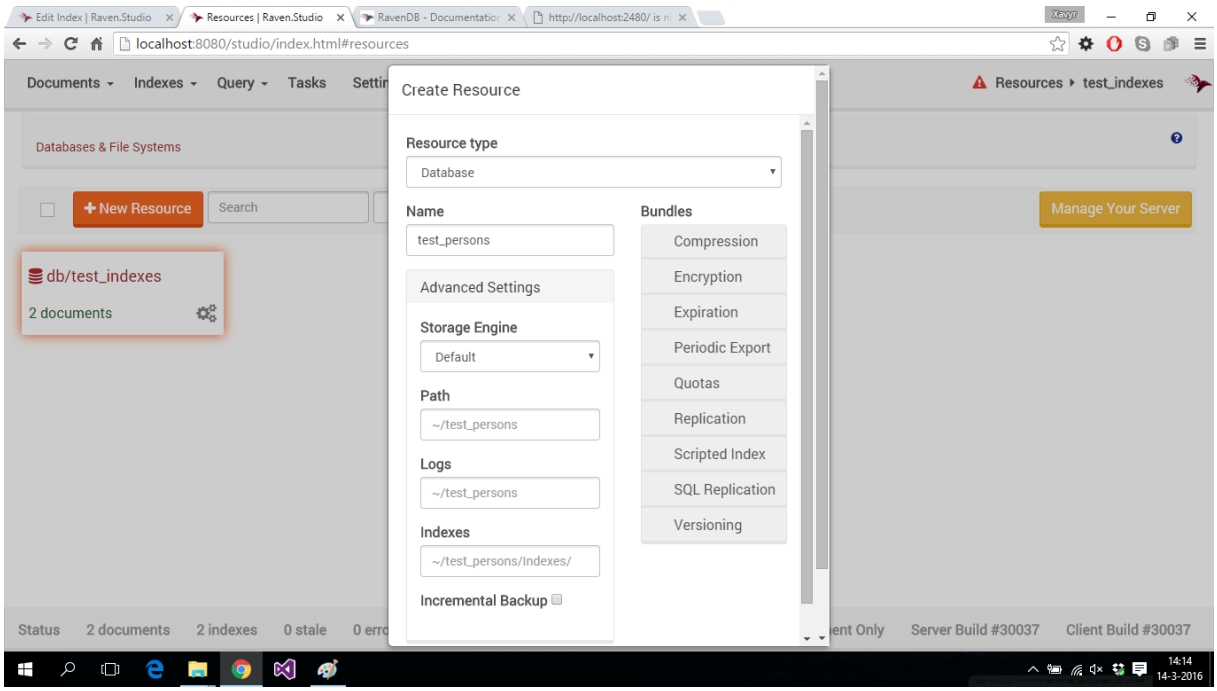
G.4 RavenDB

Secundaire indexen:
Het is in RavenDB alleen mogelijk te query-en op indexen. Omdat het zonder indexen niet mogelijk is documenten uit een collection te filteren, is er geen sprake van optimalisatie. Ook toont RavenDB niet zien hoeveel reads hij heeft uitgevoerd om tot het uiteindelijke resultaat te komen, omdat hij altijd de minimale hoeveelheid reads doet(als er 1 document als result is, doet hij maar 1 read).

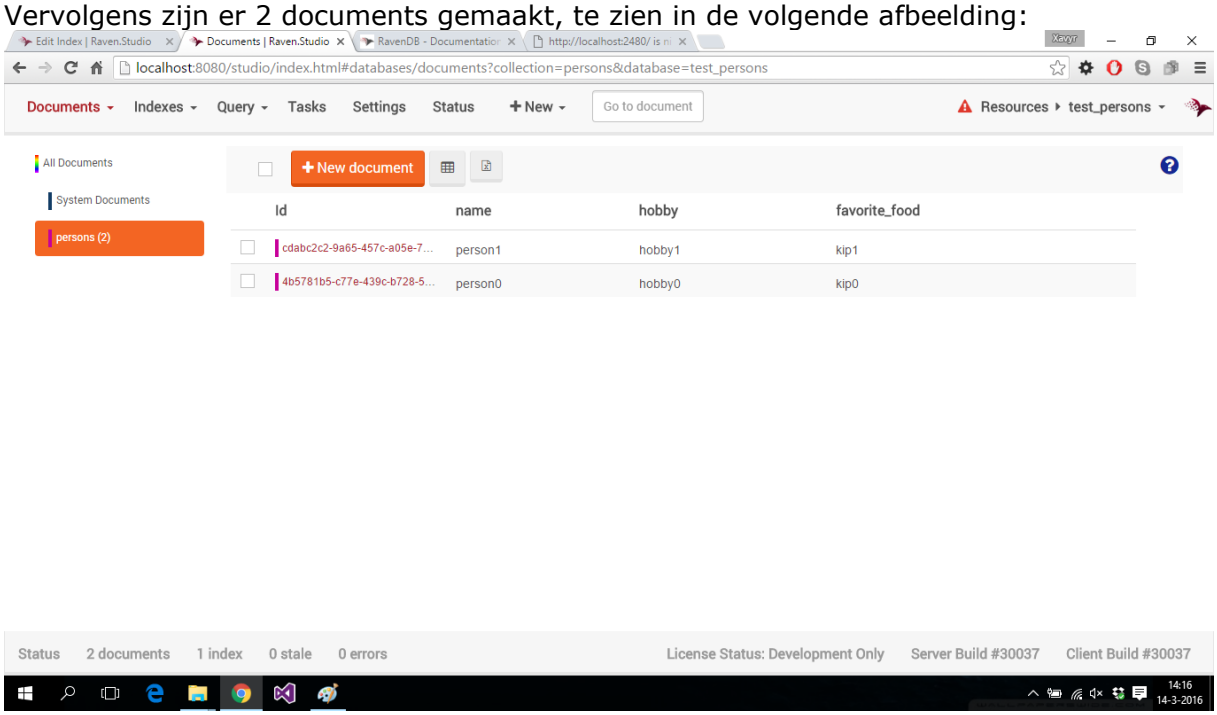
Op het moment dat er een query wordt uitgevoerd, waarin wordt gezocht op een veld wat geen index heeft, maakt RavenDB automatisch een index op dit veld aan.

"Indexes are server-side functions that define using which fields (and what values) document can be searched on and are the only way to satisfy queries in RavenDB"
http://ravendb.net/docs/article-page/3.0/csharp/indexes/what-are-indexes

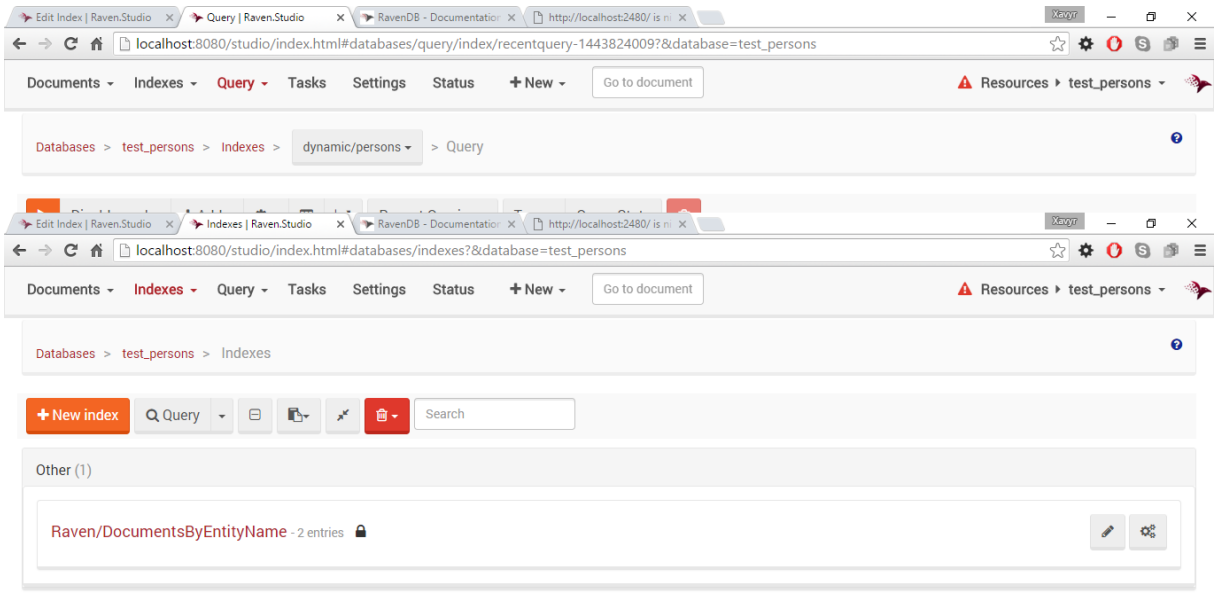
Het volgende is uitgeprobeerd om dit te bevestigen:
Maken database:
Allereerst is er een database gemaakt, genaamd: test_persons.



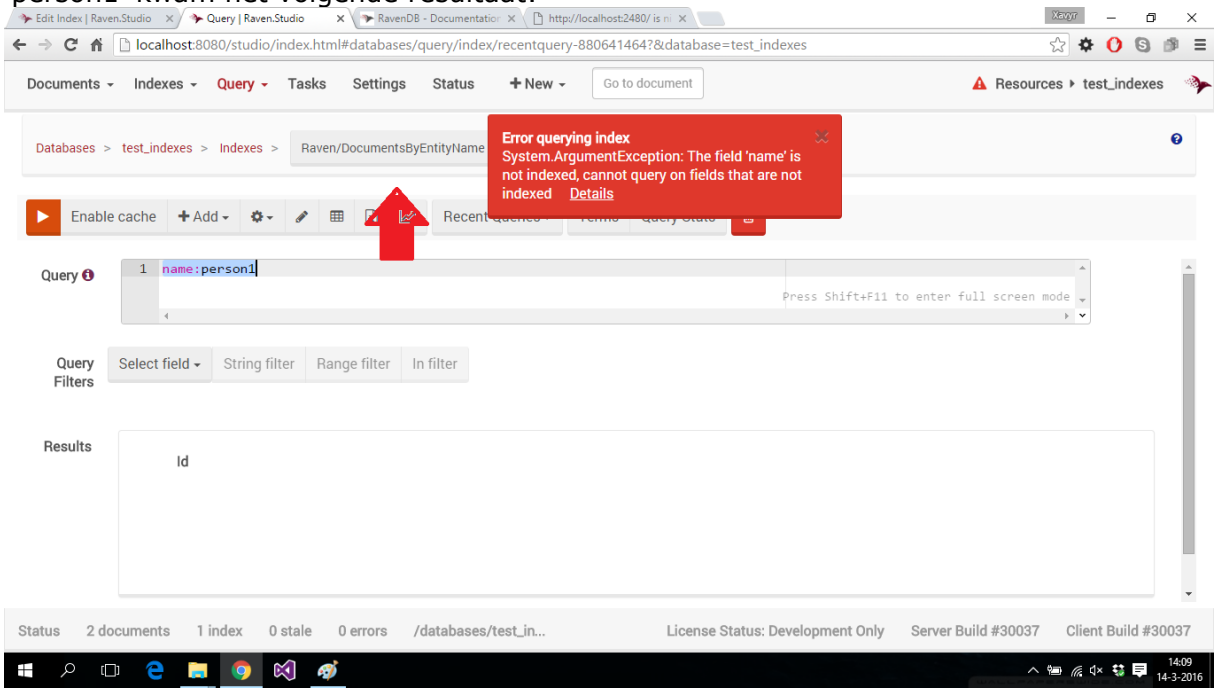
Vervolgens zijn er 2 documents gemaakt, te zien in de volgende afbeelding:



Uitvoeren select query's met default index:
Voorafgaand aan de uitvoering van de eerste query, is er gekeken naar de huidige indexen. Dit was alleen de default index:



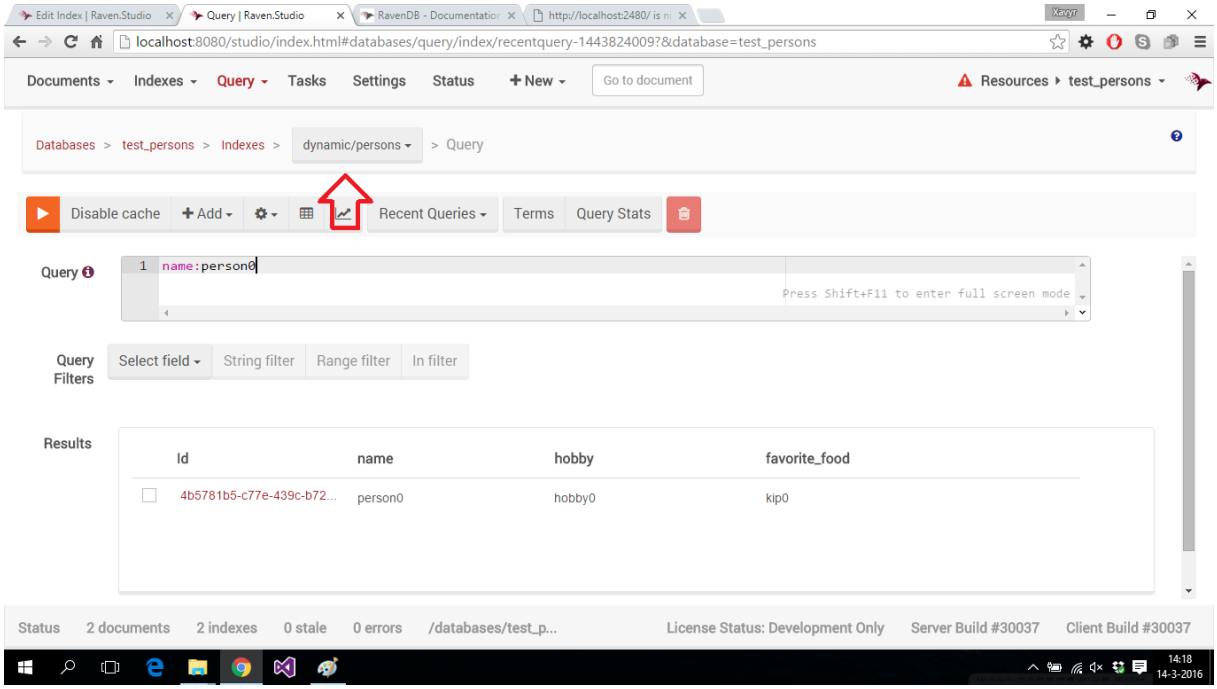
Welke het mogelijk maakt te filteren op entiteit.
Toen er met behulp van deze index geprobeerd werd te query'en op personen met als naam 'person1' kwam het volgende resultaat:



Zoals te zien op de afbeelding is het niet mogelijk, met deze index te zoeken op een persoon zijn naam.

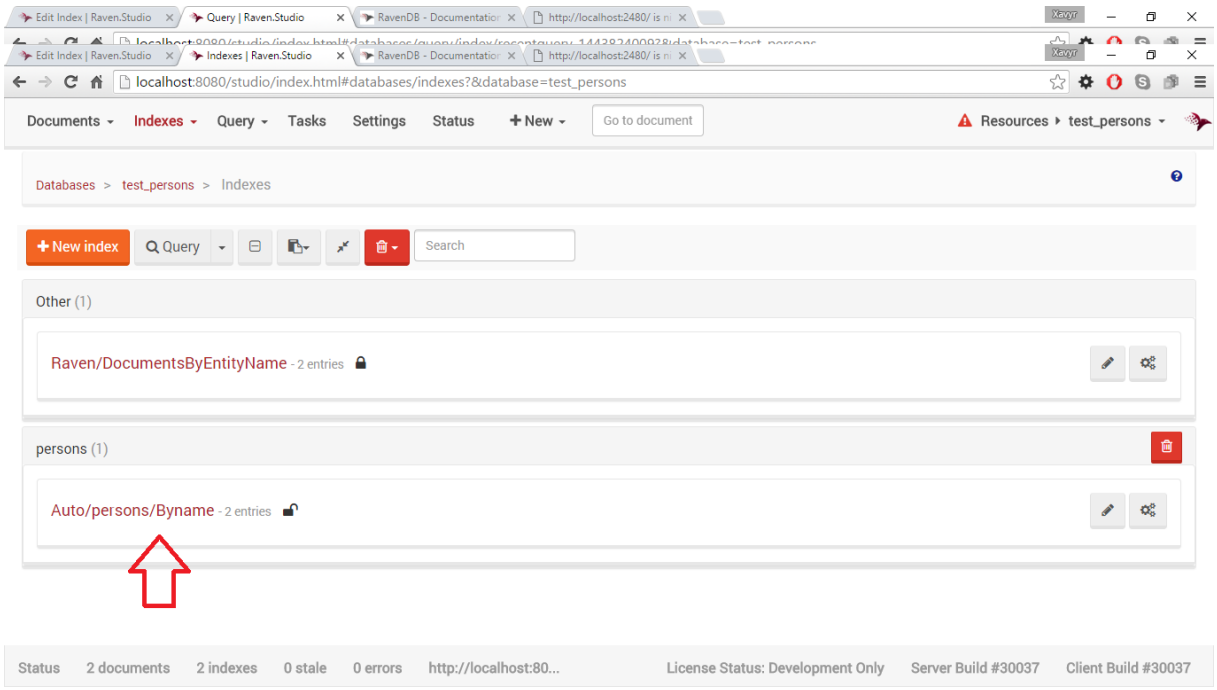
Uitvoeren query op collection persons:

Vervolgens is dezelfde query uitgevoerd, maar dan direct op de collection(dit wordt dynamisch/dynamic op de collection genoemd). Hier kwamen wel de verwachte resultaten uit:



Door deze query uit te voeren met als gebruikte index dynamic/person, zeg ik eigenlijk tegen RavenDB dat hij dynamisch indexen moet maken, op alle attributen waar ik op filter. Omdat RavenDB zelf de index maakt zodra de query wordt uitgevoerd, is het op deze manier wel mogelijk te query'en op attributen.

Na het uitvoeren van deze query, is ook te zien dat er een index is aangemaakt:


Edit Index | Raven.Studio
Query | Raven.Studio
RavenDB - Documentation
http://localhost:2480/ is ni
Indexes | Raven.Studio
localhost:8080/studio/index.html#databases/indexes?&database=test_persons
Documents
Indexes
Query
Tasks
Settings
Status
New
Go to document
Resources
test_persons
Databases > test_persons > Indexes
New index
Query
Search
Other (1)
Raven/DocumentsByEntityName - 2 entries
persons (1)
Auto/persons/Byname - 2 entries
Status
2 documents
2 indexes
0 stale
0 errors
http://localhost:80...
License Status: Development Only
Server Build #30037
Client Build #30037

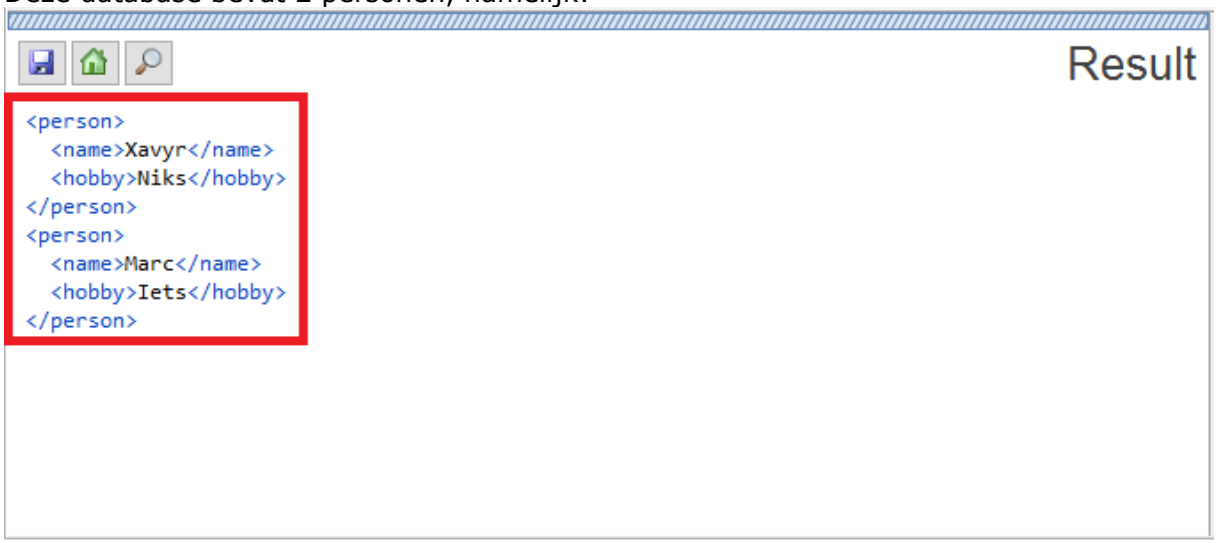
Bijlage H: Draaien op meerdere machines

H.1 BaseX

Verschillende machines installeren:
Het is mogelijk BaseX op verschillende machines te installeren. Dit heb ik getest door BaseX zowel op mijn laptop als op mijn werkstation te installeren. Beide computers draaien op Windows 10.

Verplaatsen van DB naar andere machine:
Het verplaatsen van de database van de ene naar de andere machine is mogelijk. Dit is mogelijk door gebruik te maken van back-up en restore. Om dit te testen is een database aangemaakt op mijn laptop genaamd: xml_test_indexes.

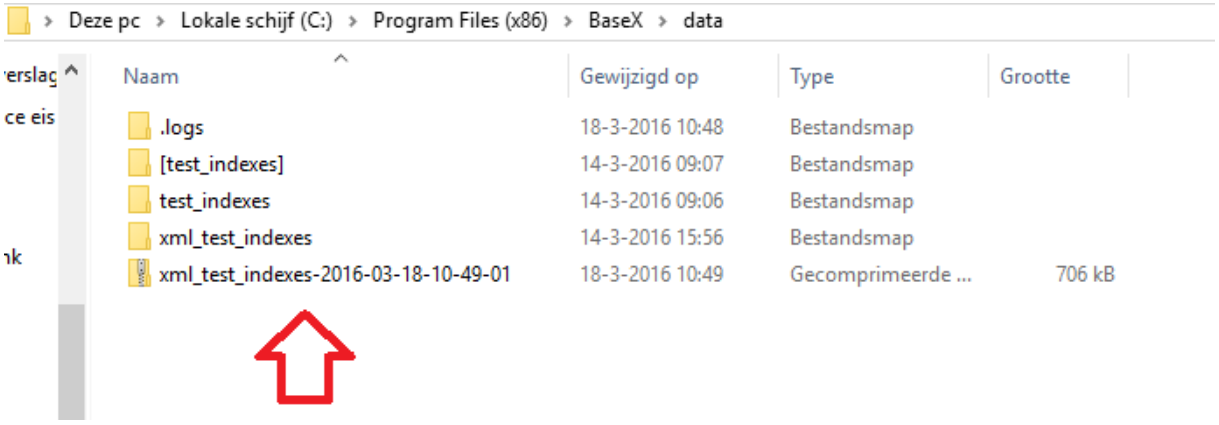
Inhoud database op laptop:
Deze database bevat 2 personen, namelijk:



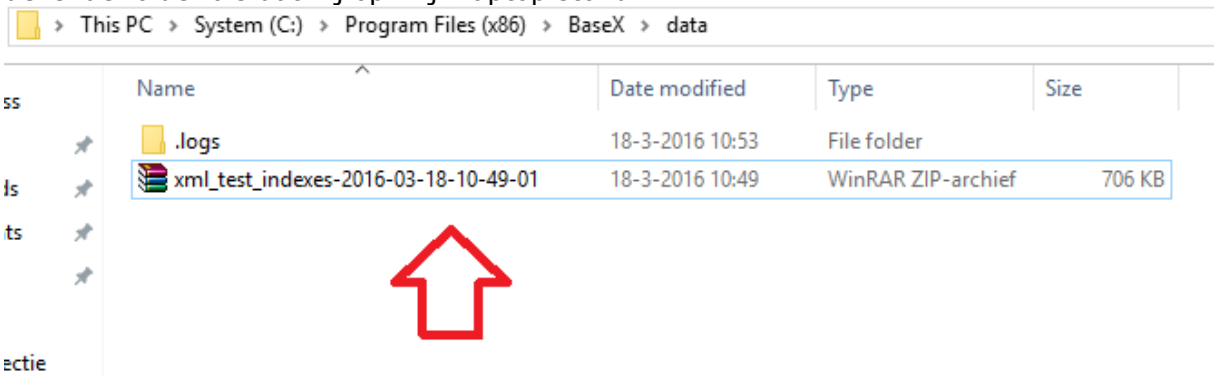
Back-up maken op laptop:
Via de BaseX client, komt mee met de installatie, is hier een back-up van gemaakt met behulp van de volgende command:

```
BaseX 8.4.1 [Client]
Try 'help' to get more information.
> create backup xml_test_indexes
```

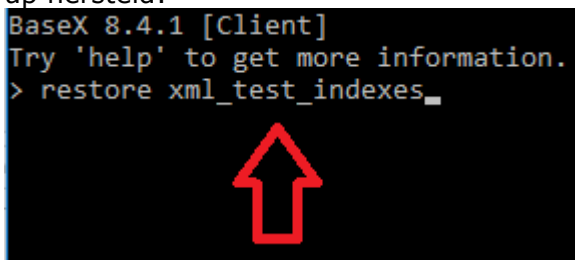
Wat de volgende zip bestand maakte in de map: installatiefolderBaseX/data.



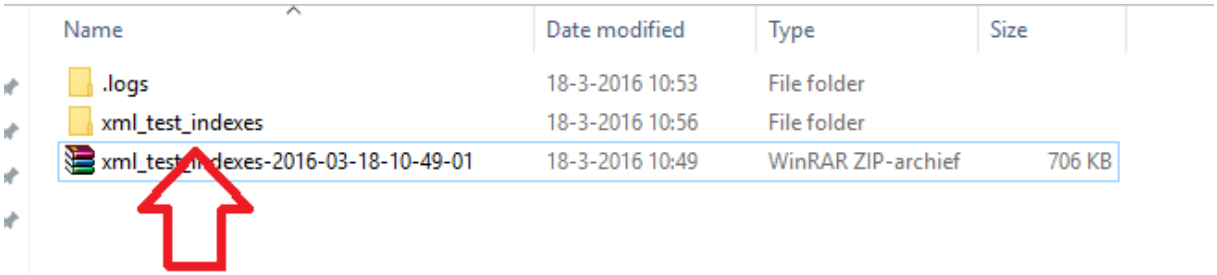
Verplaatsen back-up:
Vervolgens is dit zip bestand, via usb, verplaatst naar mijn desktop. Hier is hij geplaatst in dezelfde folder als dat hij op mijn laptop stond.



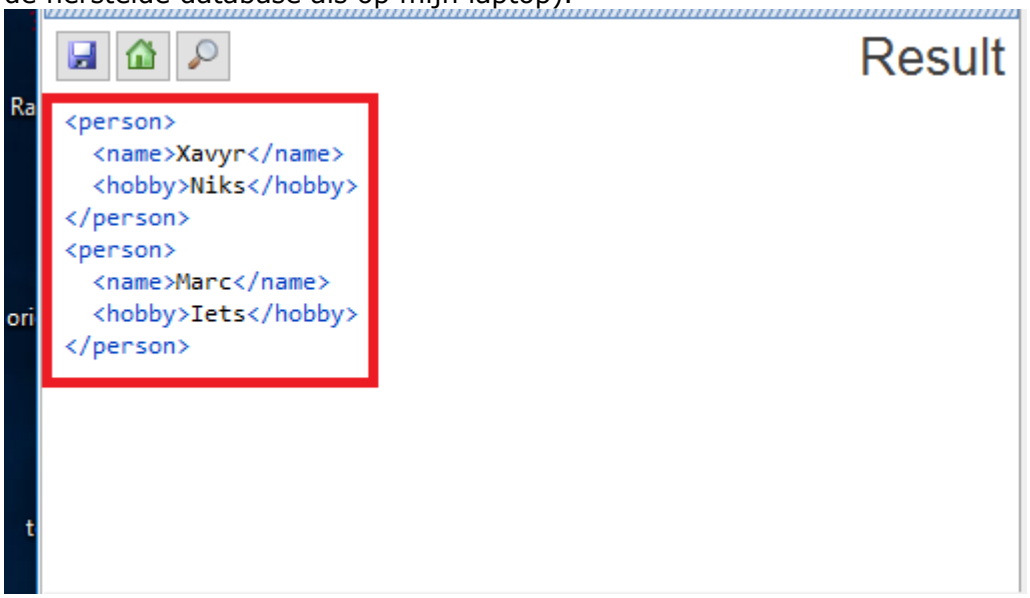
Herstellen back-up:
Door de volgende command in de BaseX client uit te voeren(op de desktop) heb ik de back-up hersteld:



Resultaat:
Na het herstellen van de database is te zien dat er een folder is gemaakt voor de database:



Via de GUI is naar de inhoud van deze database gekeken(bevinden dezelfde personen zich in de herstelde database als op mijn laptop).



Replication:
De enige informatie die er beschikbaar is over replication in BaseX is http://files.basex.org/publications/xmlprague2013/2013/Dirk-Kirsten-Distributed-XQuery.pdf . Dit is een pdf (bevat de sheets van een presentatie) uit 2013, waarin besproken wordt hoe BaseX replication in de toekomst zal gaan aanpakken. In de documentatie van BaseX is hier echter niks over terug te vinden.

H.2 OrientDB

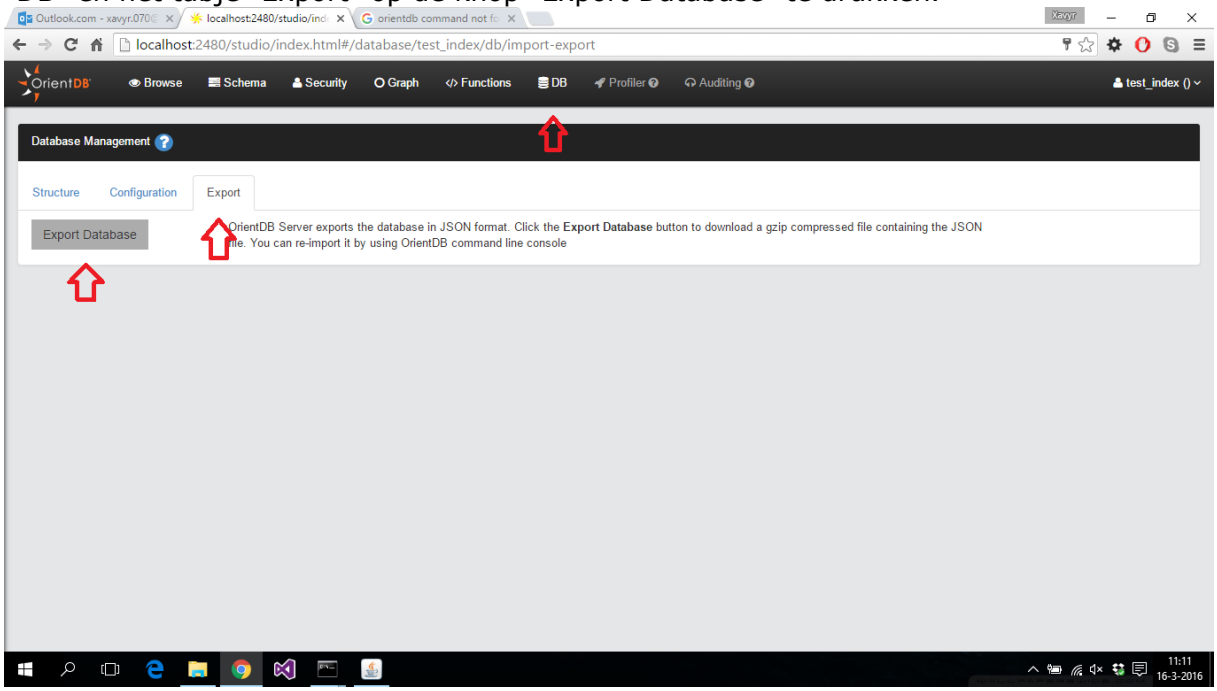
Verschillende machines installeren:
Het is mogelijk OrientDB op verschillende machines te installeren. Dit heb ik getest door OrientDB op mijn laptop te draaien, en op mijn werkstation. Beide computers draaien op Windows 10 en hebben Java geïnstalleerd(nodig om OrientDB te runnen).

Verplaatsen van DB naar andere machine:
Het verplaatsen van een DB naar een andere machine is mogelijk door middel van:

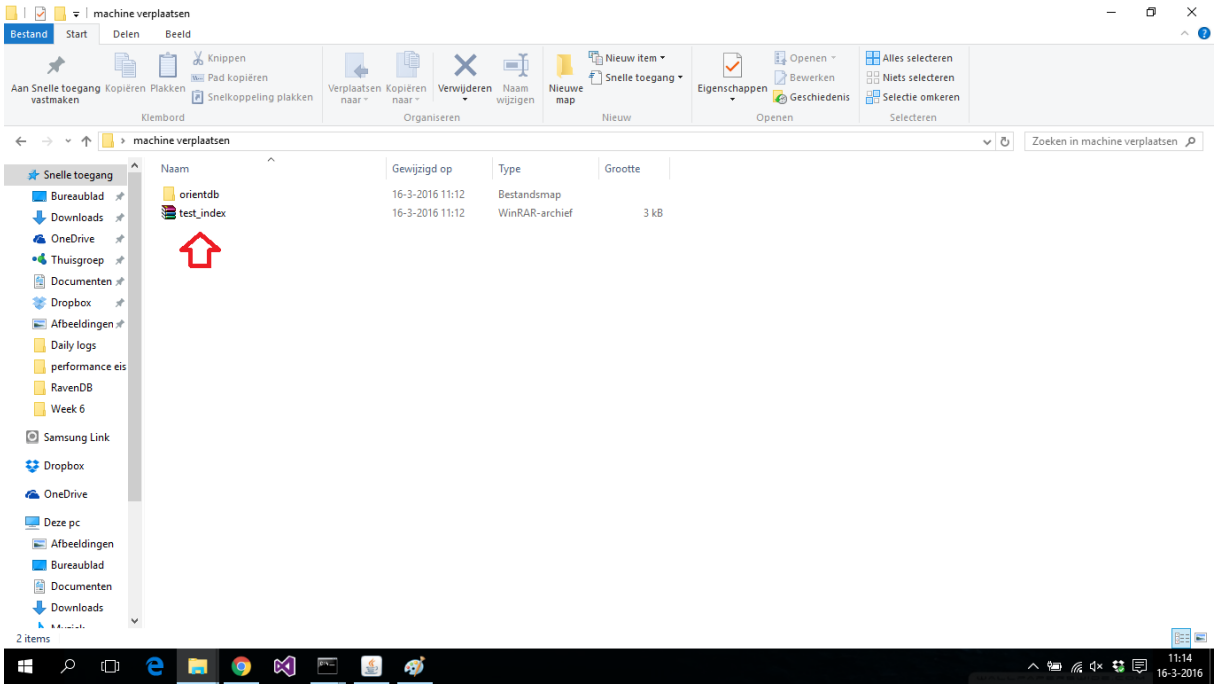
- Import en export
- Back-up en recovery

Voor dit voorbeeld is gebruik gemaakt van import en export. De volgende stappen zijn ondernomen:
Exporteren database:
Voor de test is de database uit de eis “Het moet mogelijk zijn om op performance te sturen” gebruikt, omdat deze al data bevatte. Verder is het exporteren gedaan op mijn laptop, aangezien de te gebruiken database zich daar bevond.

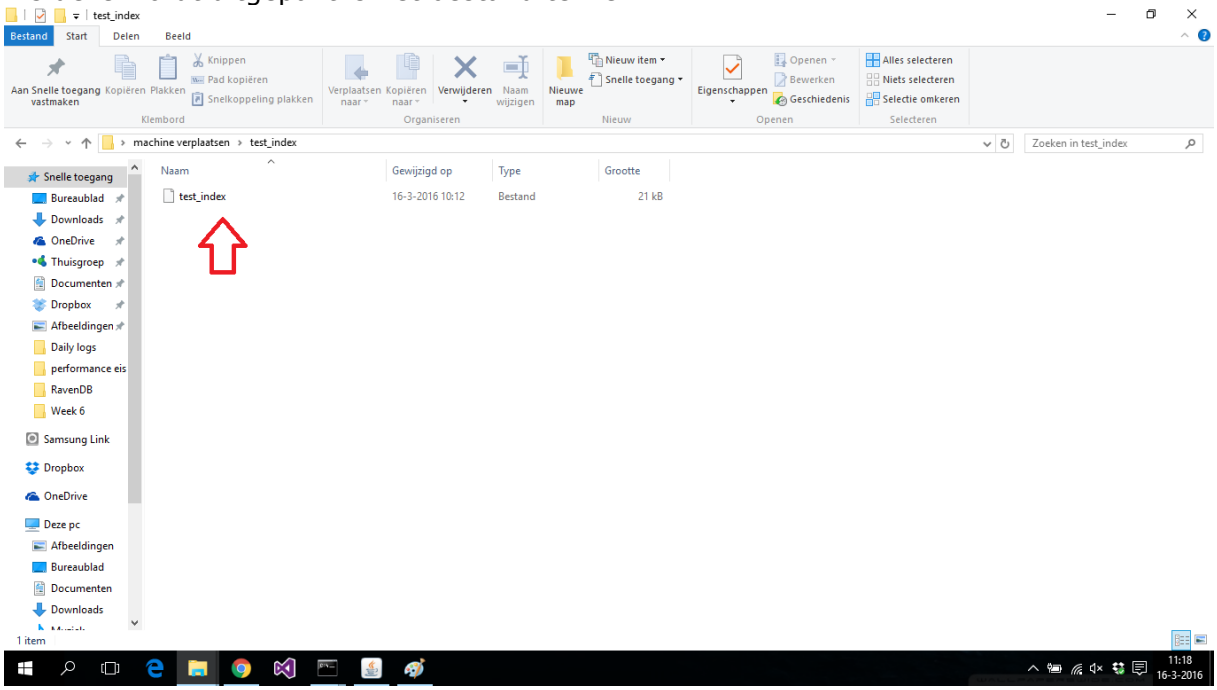
Van deze database is via de Studio een export gemaakt. Dit is gedaan door onder het kopje “DB” en het tabje “Export” op de knop “Export Database” te drukken.



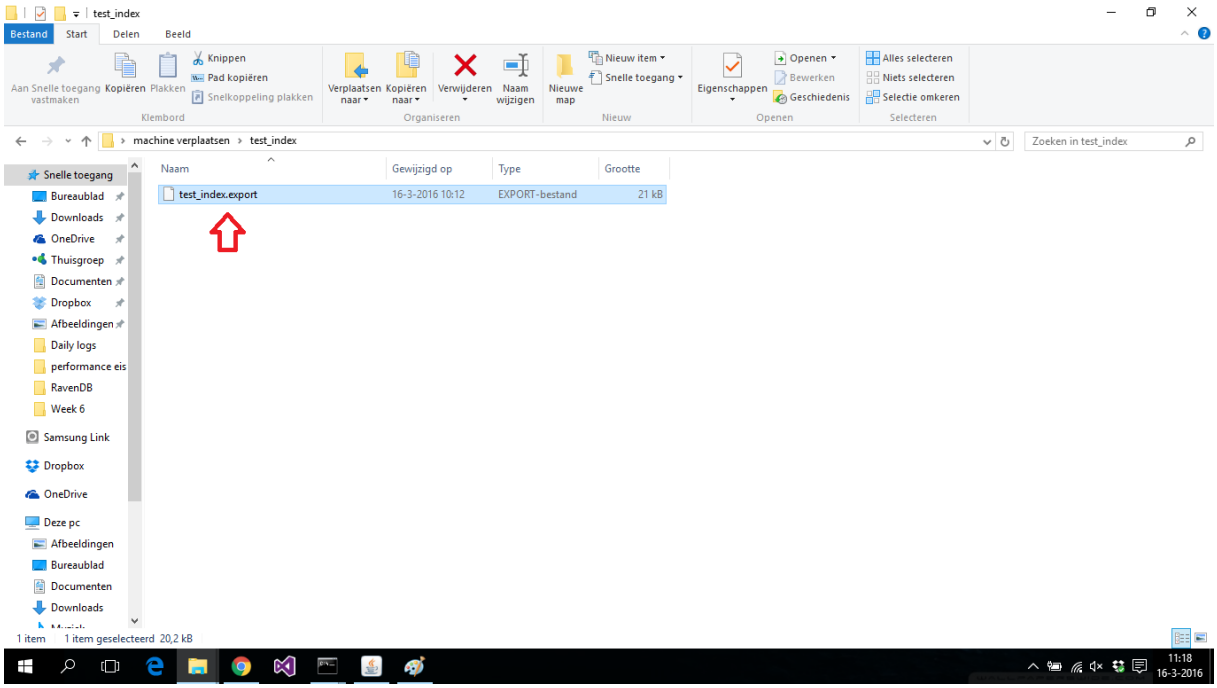
Deze maakt een JSON bestand aan in het formaat .export en formatteert deze naar een .tar.gz bestand.



Als deze wordt uitgepakt is het bestand te zien:

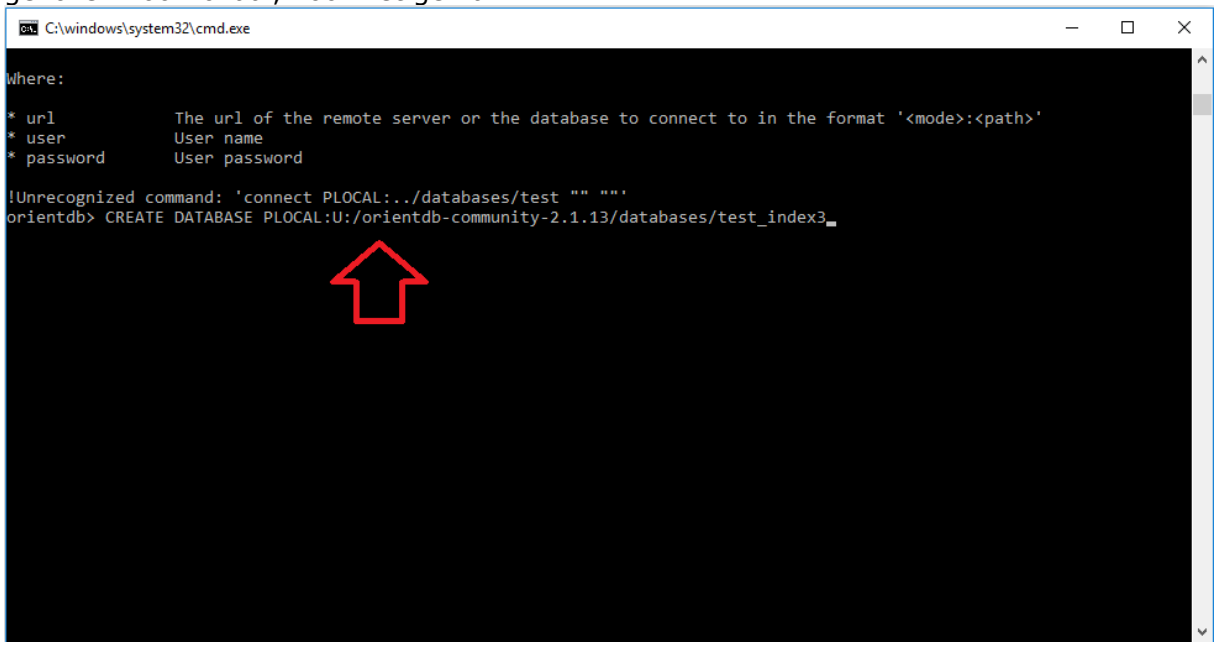


Het nadeel van exporteren via de Studio, is dat hij het exportbestand zonder extensie opslaat. Dit moet er zelf bij toegevoegd worden(door .export achter de naam te zetten):

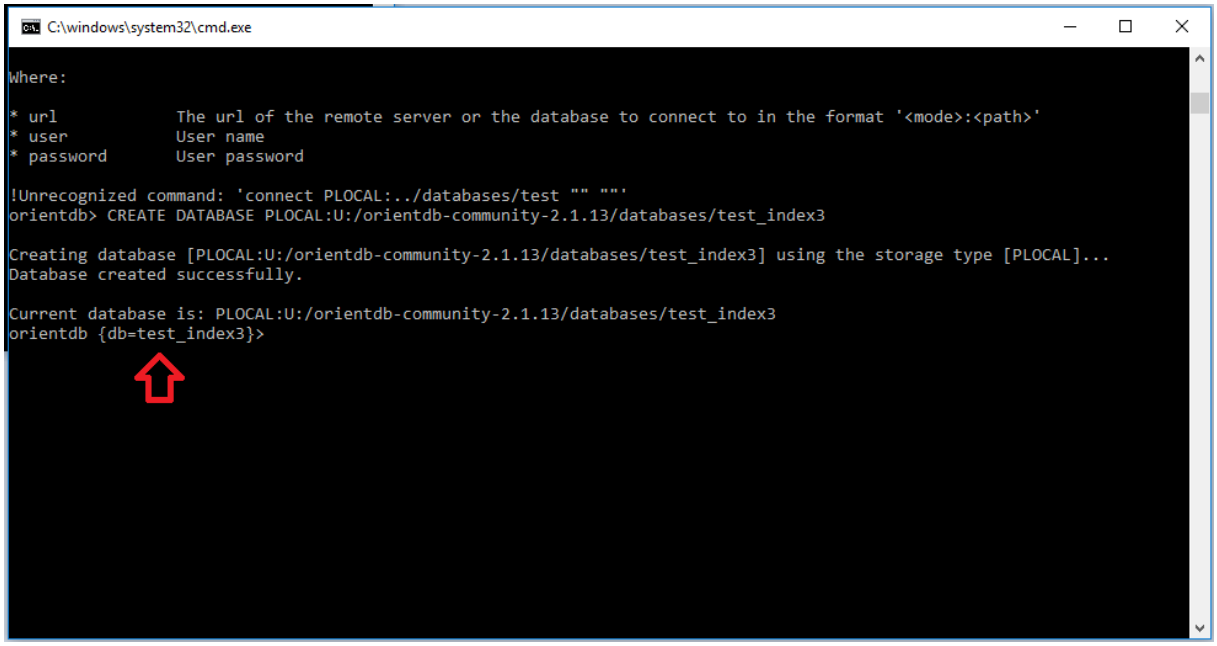


Vervolgens is het bestand op een USB stick gezet, en verplaatst naar mijn desktop. Hier is het bestand van de USB naar een map op de desktop zelf verplaatst.

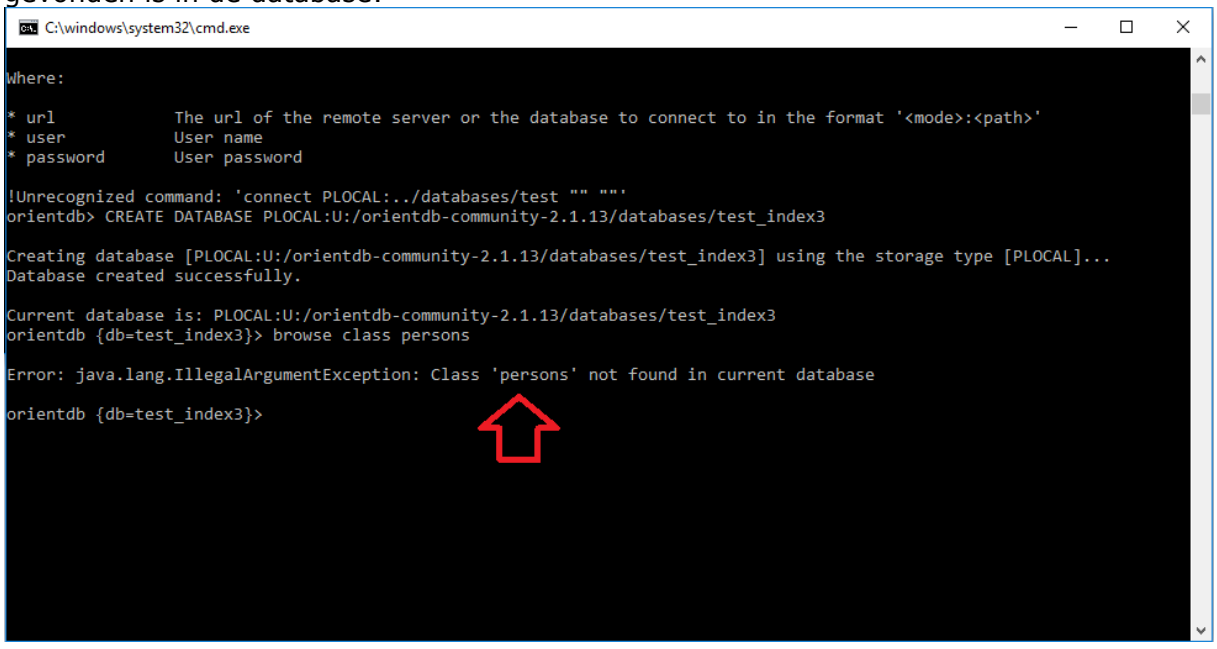
Aanmaken DB op desktop:
Om de data te kunnen importeren moest er eerst een nieuwe database aangemaakt worden. Het importeren kan ook door met een bestaande database te connecten, maar aangezien ik er nog geen had op mijn desktop heb ik besloten een nieuwe database aan te maken. Verder is het via de console mogelijk een database lokaal(PLocal) of remote aan te maken. Er is nu gekozen voor lokaal, voor het gemak.



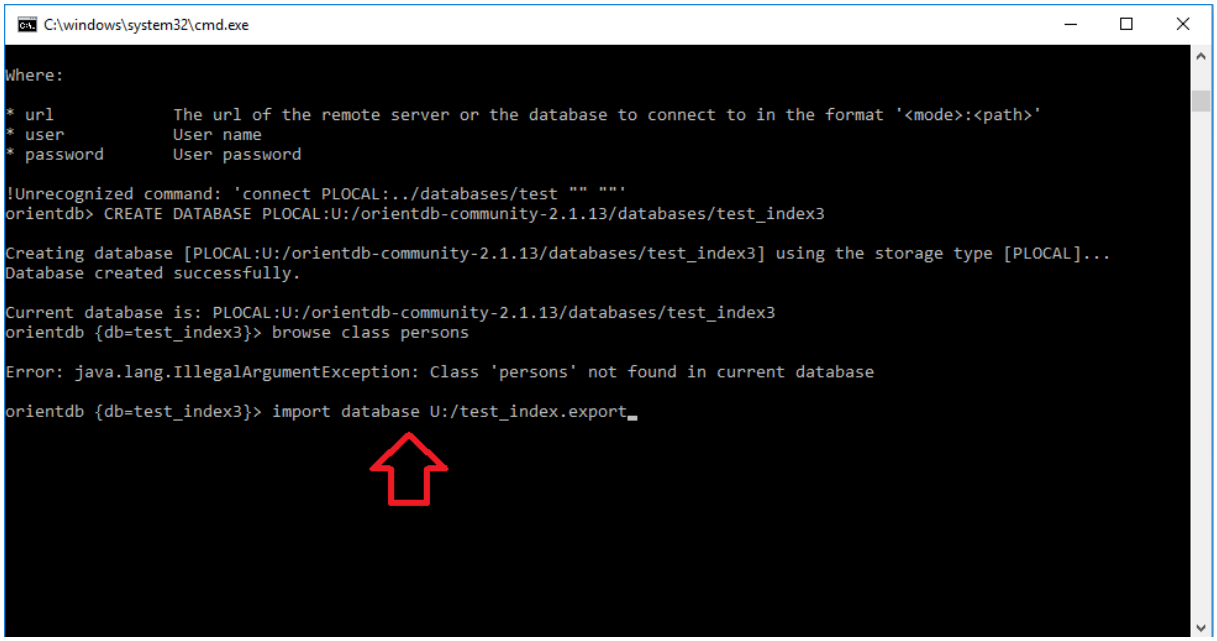
Na het uitvoeren van deze commando is te zien dat ik ben verbonden met deze database:



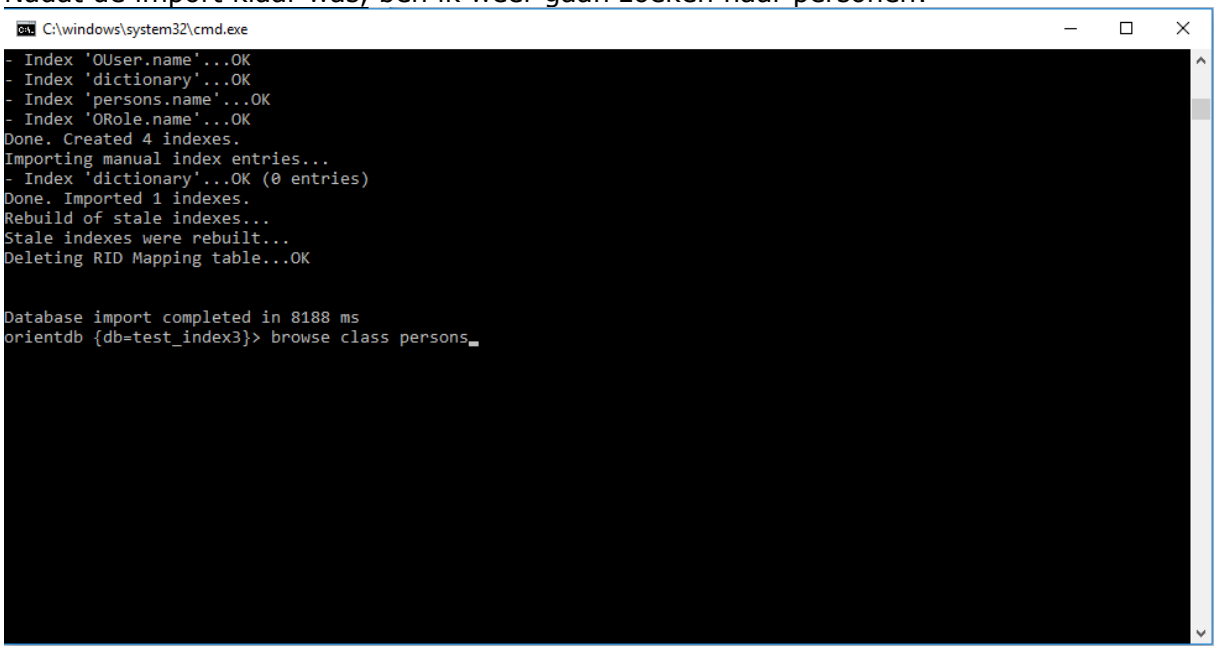
Zoeken naar personen voor import:
Vervolgens heb ik(om aan te tonen dat de database geen personen bevat) gezocht naar personen in deze database. Dit leverde een exception op, omdat de class persons niet gevonden is in de database:



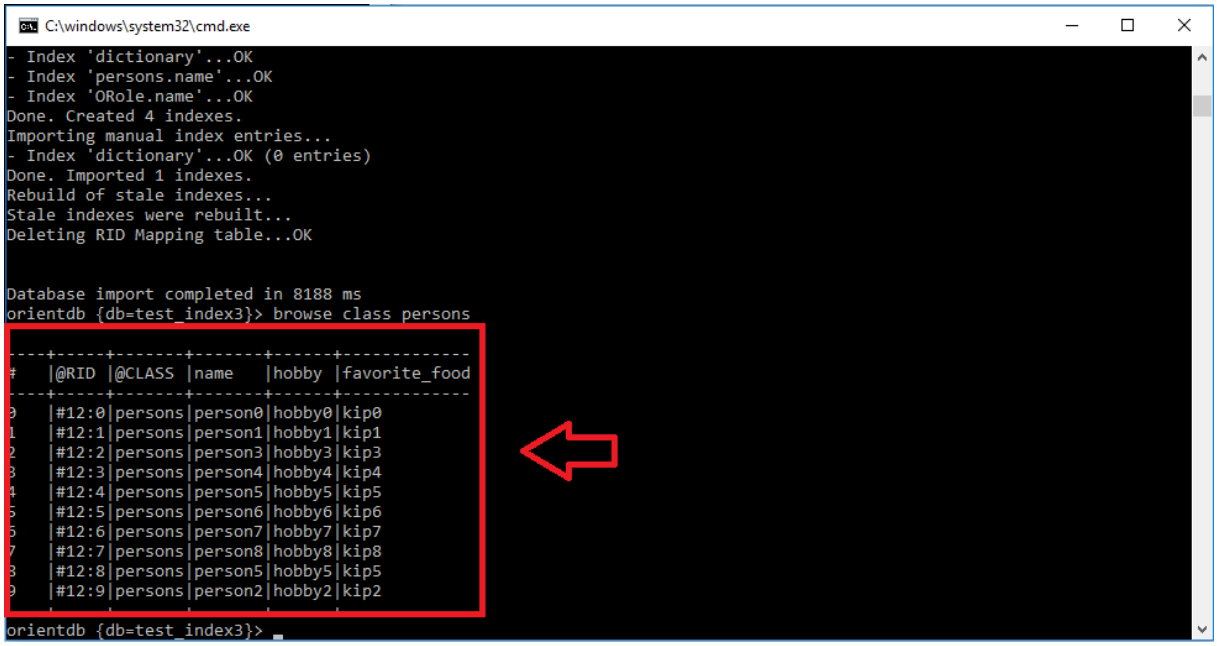
Importeren database:
Vervolgens is het export bestand van mijn laptop geïmporteerd in de database op mijn desktop.



Zoeken naar personen na import:
Nadat de import klaar was, ben ik weer gaan zoeken naar personen:



Wat deze keer wel resultaten gaf, namelijk de 10 personen uit de database:



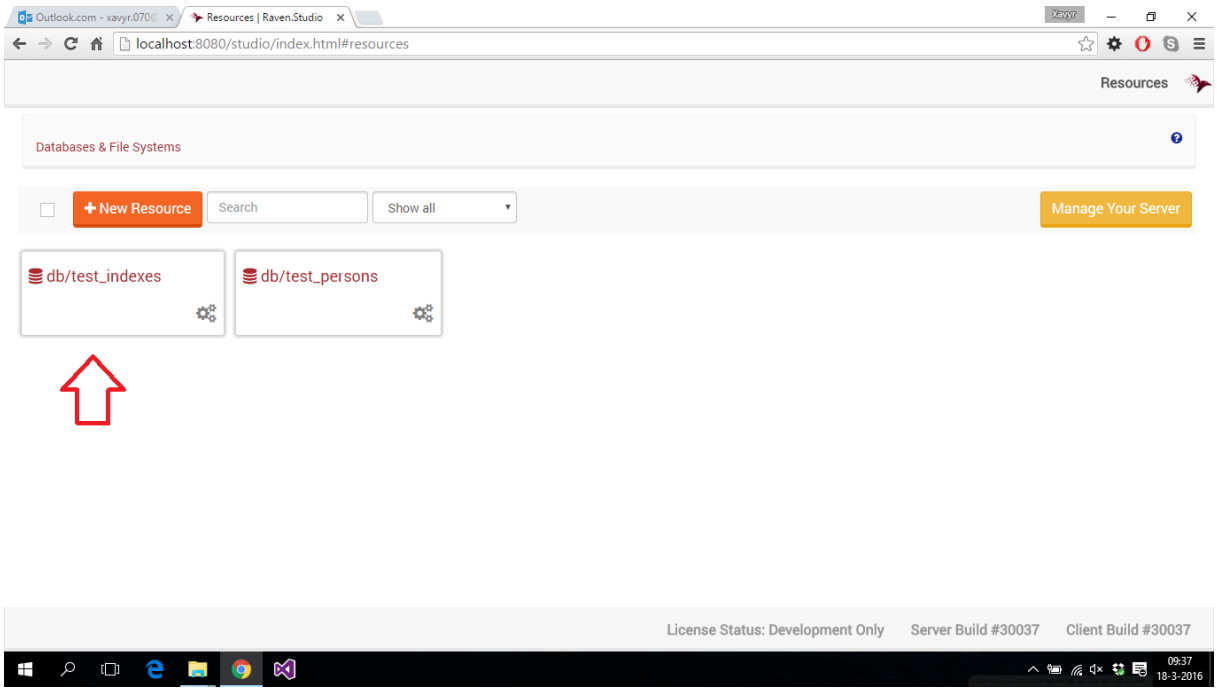
Replication:
Het is in OrientDB mogelijk om replication in te zetten. Dit is gevonden in hun documentatie: https://orientdb.com/docs/last/Replication.html

H.3 RavenDB

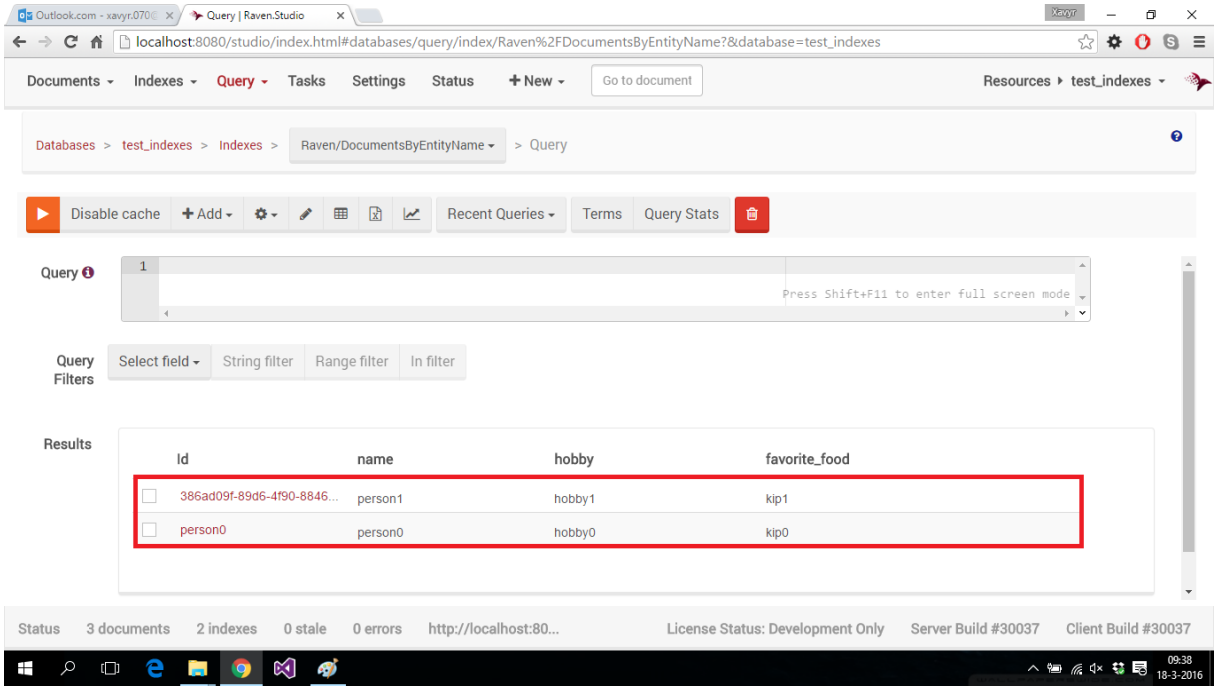
Verschillende machines installeren:
Het is mogelijk RavenDB op verschillende machines te installeren. Dit heb ik getest door RavenDB zowel op mijn laptop als op mijn werkstation te installeren. Beide computers draaien op Windows 10.

Verplaatsen van DB naar andere machine:
Het is ook mogelijk om een database te verplaatsen van de ene naar de andere machine. Ook hiervoor heb ik gebruik gemaakt van back-up en restore. Allereerst heb ik een back-up gemaakt van een database op mijn laptop, genaamd: test_indexes. De volgende stappen zijn ondernomen om de data uit deze database te verplaatsen:

Inhoud database op laptop:
Allereerst is er een database gemaakt, en gevuld met 2 documents. Deze database is te zien op de onderstaande afbeelding:



De documents die zich bevonden in deze database waren:



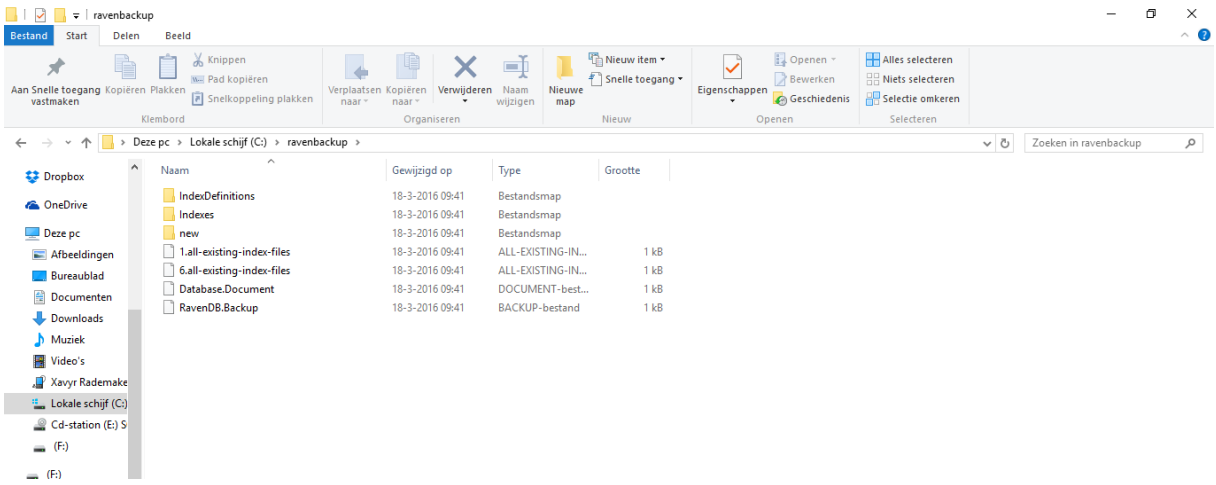
Maken back-up:
Vervolgens heb ik een back-up gemaakt, vanuit de back-up utility, welke zich bevind in een zip bestand te vinden op de site van RavenDB. Dit zip bestand is te downloaden op het moment dat de installer ook gedownload wordt.

De back-up utility bevind zich in het zip bestand onder: /Backup/Raven.Backup.exe. Dit is gewoon een command line utilty die het maken van backups versimpeld.

In deze utility moeten 2 stappen ondernomen worden, namelijk: het invullen van de url naar de database die een back-up moet krijgen, en het invullen van de uri naar de locatie waar de back-up opgeslagen moet worden(bijvoorbeeld C:\ravenbackup). Let er wel op dat de uri moet verwijzen naar een lege folder.

```
Enter RavenDB server URL:
http://localhost:8080/databases/test_indexes
Enter backup destination:
C:\ravenbackup
```

Als dit is uitgevoerd ziet de back-up folder er als volgt uit:



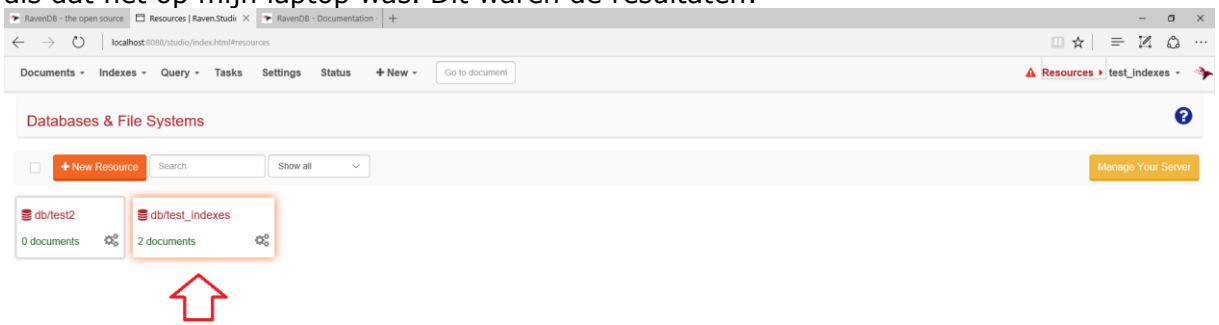
Herstellen database:
Vervolgens is deze folder, via usb, verplaats naar mijn desktop. Hier is de database weer ingeladen via de command-line(Start→cmd).

Dit is gedaan door te navigeren naar de folder(binnen de zip bestand) /Server. Vervolgens is de volgende command uitgevoerd:

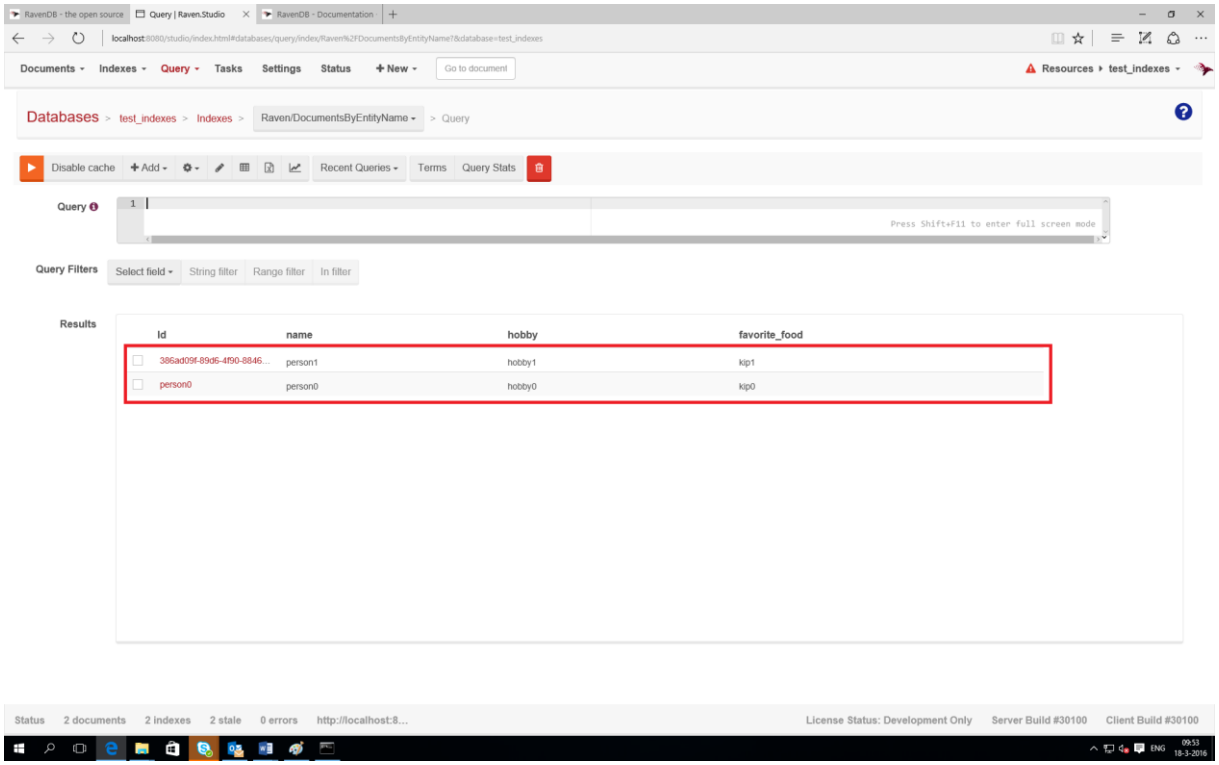
```
U:\RavenDB-Build-30100\Server>Raven.Server.exe --restore-source="C:\ravenbackup" --restore-database http://localhost:8080/databases/test_indexes
Started restore operation from C:\ravenbackup on http://localhost:8080/databases/test_indexes server.
Completed restore operation from C:\ravenbackup on http://localhost:8080/databases/test_indexes server.
```

Hierbij is de restore-source: de uri naar de folder waar de backup zich in bevind, en de restore-database de url naar de te herstellen database(in dit geval heeft de database geen url omdat deze nog niet bestaat, maar met deze commando wordt de database aangemaakt en de data erin geladen).

Resultaten herstel:
Na het herstel is via de studio gekeken of de database precies hetzelfde op de desktop was als dat het op mijn laptop was. Dit waren de resultaten:



De database is aangemaakt.



De data in de database is hetzelfde als op mijn laptop.

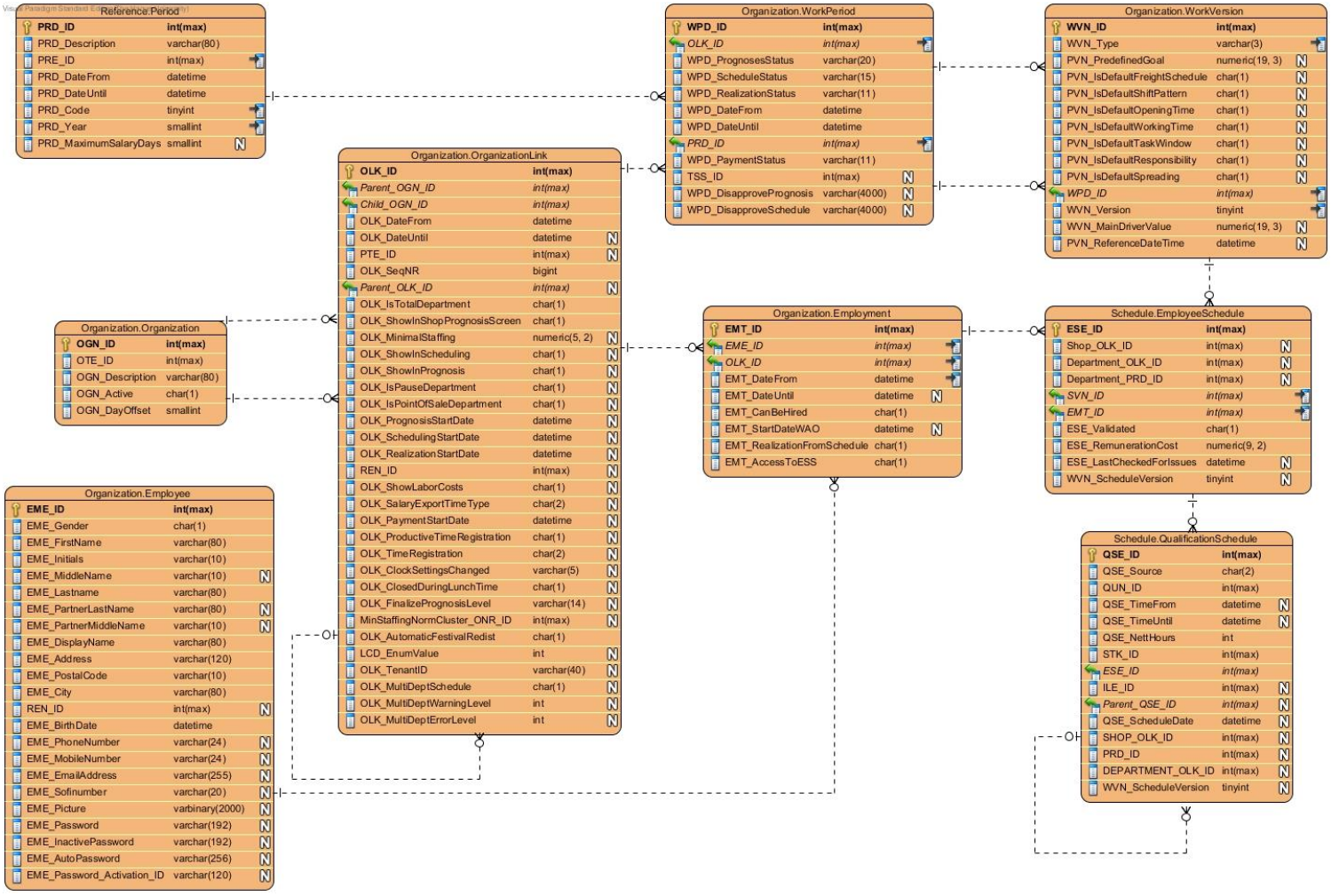
Replication:
Het is in ravenDB mogelijk om replica's op meerdere servers te zetten. Dit is gevonden in hun documentatie op:
https://ravendb.net/docs/article-page/3.0/csharp/server/scaling-out/replication/how-replication-works

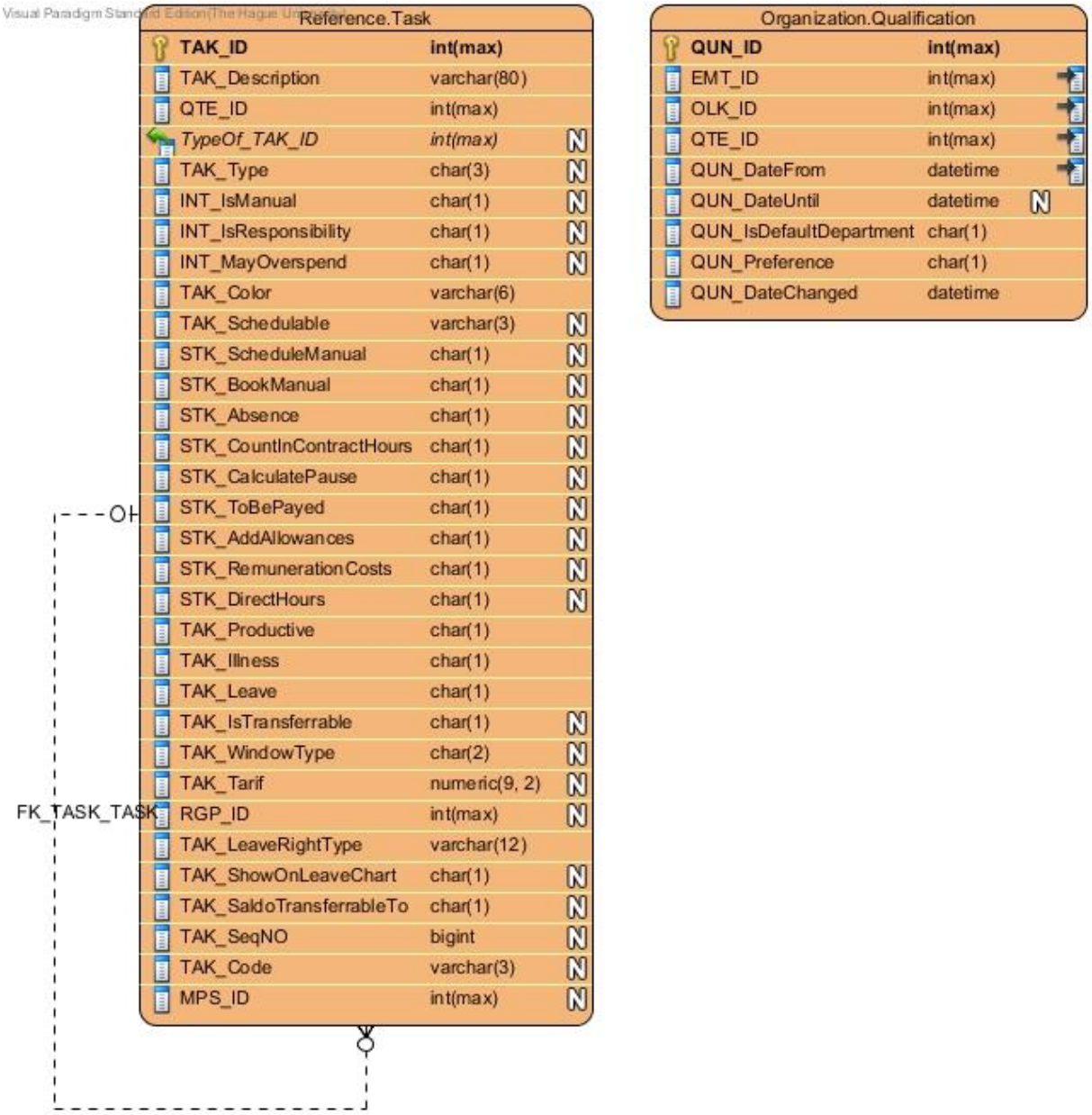
Bijlage I: Performance vs. SQL-Server

Om te achterhalen of de document databases sneller zijn dan de huidige database, is er besloten de volgende tabellen uit de database te query'en:

- Schedule.EmployeeSchedule
- Schedule.QualificationSchedule
- Organization.WorkVersion
- Organization.WorkPeriod
- Reference.Period
- Organization.OrganizationLink
- Organization.Organization
- Organization.Employment
- Organization.Employee
- Organization.Qualification(later toegevoegd aan de hand van de query)
- Reference.Task(later toegevoegd aan de hand van de query)
-

De relaties tussen deze tabellen is de vinden in de volgende diagrammen:





Voor de laatste 2 tabellen(Task en Qualification) geld dat zij de volgende relaties hebben met de tabellen uit de 1e afbeelding:

- Task
  - Qualificationschedule heeft een 1 op meer relatie met deze tabel(1 task kan bij meerdere qualificationschedules horen)
- Qualification
  - Wordt verbonden met EmployeeSchedule op basis van Qualification.OLK_ID en EmployeeSchedule.Department_OLK_ID

Wat wordt er getest:

Er zullen 3 query's uitgevoerd worden op deze tabellen/documents, 1 select, 1 insert en 1 insert met grotere hoeveelheden data(bulk).

Voor de select moet het volgende worden opgehaald: van alle medewerkers die op 1 afdeling werken, alle diensten op het rooster op basis van de jaar, week, winkel en afdeling.

Voor de insert moet het volgende worden toegevoegd: alle diensten van 1 medewerker binnen een versie van het rooster van 1 afdeling, van 1 winkel in een bepaalde periode.

Voor de insert van grotere hoeveelheden data moet het volgende worden toegevoegd: alle diensten van alle medewerkers binnen een versie van het rooster van een winkel.

Per handeling kan er 1 punt verdient worden als de database sneller is dan de SQL-server database.

Structuur document databases:

Om de structuur van de document databases te bepalen is het volgende document als handvat gebruikt: https://s3.amazonaws.com/info-mongodb-com/RDBMStoMongoDBMigration.pdf.

Verder is er gekeken naar de te gebruiken select query, en welke data er precies opgevraagd wordt, en welke data gebruikt wordt om te filteren. Van de te gebruiken data is de vraag gesteld: hoe vaak wijzigt deze data. De data die veel wijzigt is als reference opgenomen, de data die (bijna) niet wijzigt is als embedded document meegenomen.

Aangezien er alleen delen uit de verschillende tabellen nodig zijn(enkele kolommen worden maar opgehaald en er wordt gefilterd op enkele kolommen) is er gekozen om naar alle aparte documents een reference te bewaren.

Voor de document databases worden van alle bovenstaande tabellen(zie diagram) collections/namespaces gemaakt. Er wordt echter 1 document bij gemaakt, namelijk: afdelingsrooster.

De structuur van de documents is als volgt bepaald:

De structuur van alle documents, gebaseerd zijn op de bovenstaande tabellen zal hetzelfde zijn als de tabellen, maar dan in JSON/XML formaat. Op het gebied van embedding/referencing wordt het volgende gedaan:

Voor de huidige situatie wordt er gekeken naar de te gebruiken query, en wordt er gekeken: welke kolommen worden er gebruikt om te joinen, welke kolommen worden er gebruikt om te filteren, welke kolommen worden er opgehaald en hoevaak wijzigt deze data.

De query heeft voornamelijk raakvlak met de toegevoegde collection genaamd “afdelingsrooster”. Deze collection zal alle informatie bevatten die nodig is voor het ophalen van alle diensten van alle medewerkers van een afdeling. De documents uit deze collection zullen er als volgt uitzien:

```
Afdelingsrooster:{
        Employments:[{
                employmentID: “”, → dit is een reference naar het volledige document
                isHired: “”,
```

```
            qualificationSchedule: [{
                department_olk_id: “”,
                qse_id : “”, → dit is een reference naar het volledige document
                datefrom: “”,
                parent_qse_ID: “”,
                timefrom: “”,
                timeuntil: “”,
                scheduledDate: “”,
                netthours: “”,
                type: “”,
                task: {
                    stk_id: “”, → dit is een reference naar het volledige document
                     CountInContractHours: “”,
                     STK_ScheduleManual: “”
                }
            }]
        }],
        afdelingID: “”,
        winkelID: “”,
        versie: “”,
        period: {
        year: “”,
        week: “”,
        startdate: “”,
        enddate: “”,
        period_ID: “”,
        }
}
```

De reden dat er voor deze structuur is gekozen is, dat het in de applicatie ook zo gebruikt kan worden. Er wordt namelijk alleen data bij elkaar opgeslagen wat niet (vaak) wijzigt, en alleen de data die opgehaald dient te worden.

Op deze manier hoeven niet alle andere collections langsgegaan te worden als deze data opgehaald dient te worden. Aangezien de data uit andere collections welke, welke hier embed is, niet (vaak) wijzigt hoeven de ontwikkelaars zich niet druk te maken dat 1 update op meerdere documents uitgevoerd moet worden. Er worden wel references naar de overige documents bijgehouden voor het geval dat het nodig blijkt te zijn deze data te combineren.

Hoeveelheid data:

De SQL-server database bevatte op het moment van testen de volgende hoeveelheid rows:

Tabel	Aantal rows
Schedule.EmployeeSchedule	356533
Schedule.QualificationSchedule	537094
Reference.Period	2831
Organization.WorkPeriod	18099
Organization.WorkVersion	84390
Organization.OrganizationLink	171
Organization.Organization	40
Organization.Employment	2963
Organization.Employee	2371
Organization.Qualification	7981
Reference.Task	87

Deze data is overgezet naar de document databases, wat inhoud dat de document databases dezelfde collections(de tabellen) met dezelfde hoeveelheid documents(rows) had. Verder was er 1 collection toegevoegd genaamd "DepartmentSchedule" waarvan de structuur eerder beschreven is. Deze collection bevatte 18322 documents.

I.1 SQL-Server

Om vergelijkingsmateriaal te verkrijgen(hoelang doet de huidige database erover) is de opdrachtgever gevraagd naar de query's/stored procedures die zij momenteel voor deze handelingen gebruiken. Deze query's/stored procedures en de resultaten die hieruit naar voren kwamen staan hieronder beschreven.

Aanpak:
Hieronder wordt per handeling beschreven wat er uitgevoerd gaat worden en hoe dit uitgevoerd gaat worden.

Select
De volgende stored procedure is uitgevoerd op de SQL-server database om de juiste data op te halen. De stored procedure is overigens geleverd door de opdrachtgever.

```
CREATE PROCEDURE [Schedule].[csp_GetShopSchedule8]
AS
BEGIN

SET NOCOUNT ON;

BEGIN TRY

DECLARE @p_DepartmentOrganizationLinkIds
TABLE(ID BIGINT);

DECLARE @l_StartDate DATE;

DECLARE @l_EndDate DATE;

SELECT        @l_StartDate = prd.[PRD_DateFrom],
@l_EndDate = prd.[PRD_DateUntil]
FROM          [Reference].[Period] prd
```

```
WHERE         prd.[PRD_ID] = 2882;

INSERT INTO @p_DepartmentOrganizationLinkIds
SELECT DISTINCT
ese.Department_OLK_ID ID
FROM Schedule.EmployeeSchedule ese
WHERE ese.Department_OLK_ID = 11543;

-- Get the employeeschedules
DECLARE @l_employeeSchedules
TABLE(EmploymentId BIGINT,
DepartmentOrganizationLinkId BIGINT,
Id BIGINT,
IsHired BIT, IsHiredFromPool BIT);

INSERT INTO @l_employeeSchedules
SELECT DISTINCT
ese.EMT_ID EmploymentId,
ese.Department_OLK_ID DepartmentOrganizationLinkId,
ese.ESE_ID Id,
IIF(qun.[QUN_ID] IS NULL, 1, 0) IsHired,
IIF(wpd.[OLK_ID] = 11521, 0, 1) IsHiredFromPool
FROM @p_DepartmentOrganizationLinkIds olkDep
INNER JOIN [Schedule].[EmployeeSchedule] initialEse
ON initialEse.[Shop_OLK_ID] = 11521
AND initialEse.[Department_OLK_ID] = olkDep.[ID]
AND initialEse.[Department_PRD_ID] = 2882
AND initialEse.[WVN_ScheduleVersion] = 1
INNER JOIN [Organization].[Employment] emt
ON emt.[EMT_ID] = initialEse.[EMT_ID]
AND Schedule.[fn_D_OverlapsInclusive](emt.[EMT_DateFrom],
emt.[EMT_DateUntil], @l_StartDate, @l_EndDate) = 1
INNER JOIN [Schedule].[EmployeeSchedule] ese
ON  ese.[EMT_ID] = initialEse.[EMT_ID]
AND ese.[Department_PRD_ID] = 2882
INNER JOIN [Organization].[WorkPeriod] wpd
ON  wpd.[OLK_ID] = ese.[Shop_OLK_ID]
AND wpd.[PRD_ID] = 2882
LEFT OUTER JOIN [Organization].[Qualification] qun
ON  qun.[OLK_ID] = ese.[Department_OLK_ID]
AND qun.[QUN_IsDefaultDepartment] = 'Y'
AND qun.[EMT_ID] = ese.[EMT_ID]
AND Schedule.[fn_D_OverlapsInclusive](qun.[QUN_DateFrom],
qun.[QUN_DateUntil], @l_StartDate, @l_EndDate) = 1
LEFT OUTER JOIN [Schedule].[QualificationSchedule] qse
ON  qse.[ESE_ID] = ese.[ESE_ID]
AND qse.[Shop_OLK_ID] = ese.[Shop_OLK_ID]
AND qse.[PRD_ID] = ese.[Department_PRD_ID]
AND qse.[Department_OLK_ID] = ese.[Department_OLK_ID]
AND qse.[WVN_ScheduleVersion] = ese.[WVN_ScheduleVersion]
WHERE
       ((wpd.[OLK_ID] = 11521 AND ese.WVN_ScheduleVersion = 1)
OR (wpd.[OLK_ID] <> 11521 AND 0 = 1
AND ese.WVN_ScheduleVersion = 0)
OR (wpd.[OLK_ID] <> 11521
AND 0 = 0 AND wpd.[WPD_ScheduleStatus] = 'FINAL'
```

```
AND ese.WVN_ScheduleVersion = 1)
OR (wpd.[OLK_ID] <> 11521 AND 0 = 0
AND wpd.[WPD_ScheduleStatus] <> 'FINAL'
AND ese.WVN_ScheduleVersion = 0))
AND (qse.[QSE_ID] IS NOT NULL OR qun.[QUN_ID] IS NOT NULL);

-- Get scheduled activities for all active employments
SELECT ese.[EmploymentId] AS [EmploymentId]
,qse.[Department_OLK_ID] AS [DepartmentOrganizationLinkIdWorkedFor]
,CAST(qse.[QSE_TimeFrom] AS Date) AS [Date]
,qse.[QSE_ID] AS [ActivityId]
,qse.[Parent_QSE_ID] AS [ParentActivityId]
,CAST(qse.[QSE_TimeFrom] AS Time) AS [StartTime]
,CAST(qse.[QSE_TimeUntil] AS Time) AS [EndTime]
,qse.[STK_ID] AS [ActivityTypeId]
,ese.[IsHired] AS [HiredIn]
FROM @l_EmployeeSchedules ese
INNER JOIN [Schedule].[QualificationSchedule] qse
ON  ese.[ID] = qse.[ESE_ID]
AND qse.[QSE_TimeFrom] IS NOT NULL
INNER JOIN [Reference].[Task] tak
ON tak.TAK_ID = qse.STK_ID
AND (tak.STK_CountInContractHours = 'Y' OR qse.Parent_QSE_ID IS NOT NULL OR
tak.STK_ScheduleManual = 'Y');

-- Get absence activities for all active employments

SELECT ese.[EmploymentId] AS [EmploymentId],
qse.[Department_OLK_ID] AS [DepartmentOrganizationLinkId],
 CAST(qse.[QSE_ScheduleDate] AS Date) AS [Date],
 qse.[STK_ID] AS [ActivityTypeId],
qse.[QSE_NettHours] AS [Duration]
FROM @l_EmployeeSchedules ese
INNER JOIN [Schedule].[QualificationSchedule] qse
ON  ese.[ID] = qse.[ESE_ID]
AND qse.[QSE_TimeFrom] IS NULL;
END TRY

       BEGIN CATCH

THROW;
       END CATCH
END
```

Insert

Voor de insert is de volgende stored procedure gebruikt:

```
CREATE PROCEDURE [Schedule].[csp_PersistLaborAgreementEmploymentData]
AS
BEGIN
       SET NOCOUNT ON;
       BEGIN TRY

              DECLARE @p_AbsenceActivities
              TABLE(ID BIGINT, EmploymentID BIGINT, DepartmentOrganizationLinkID BIGINT,
Source char(2), Date DateTime, StartTime DateTime, EndTime DateTime, ActivityTypeId
bigint, ParentActivityId BIGINT, Duration int, AbsenceReferenceId BIGINT);
```

```
            INSERT INTO @p_AbsenceActivities
            SELECT DISTINCT
            qse.QSE_ID,
            ese.EMT_ID EmploymentID,
            ese.Department_OLK_ID DepartmentOrganizationLinkID,
            qse.QSE_Source,
            qse.QSE_TimeFrom,
            qse.QSE_TimeFrom,
            qse.QSE_TimeUntil,
            qse.STK_ID,
            qse.Parent_QSE_ID,
            qse.QSE_NettHours,
            qse.ILE_ID
            FROM Schedule.EmployeeSchedule ese
            JOIN Schedule.QualificationSchedule qse
            ON ese.ESE_ID = qse.ESE_ID
            WHERE ese.Department_OLK_ID = 11605
            AND ese.Shop_OLK_ID = 11441
            AND qse.PRD_ID = 2793
            AND ese.WVN_ScheduleVersion = 1
            AND ese.ESE_ID = 1569669;

            DECLARE @p_Activities
            TABLE(ID BIGINT, EmploymentID BIGINT, DepartmentOrganizationLinkID BIGINT,
Source char(2), Date DateTime, StartTime DateTime, EndTime DateTime, ActivityTypeId
bigint, ParentActivityId BIGINT, Duration int);
            INSERT INTO @p_Activities
            SELECT DISTINCT
            qse.QSE_ID,
            ese.EMT_ID EmploymentID,
            ese.Department_OLK_ID DepartmentOrganizationLinkID,
            qse.QSE_Source,
            qse.QSE_TimeFrom,
            qse.QSE_TimeFrom,
            qse.QSE_TimeUntil,
            qse.STK_ID,
            qse.Parent_QSE_ID,
            qse.QSE_NettHours
            FROM Schedule.EmployeeSchedule ese
            JOIN Schedule.QualificationSchedule qse
            ON ese.ESE_ID = qse.ESE_ID
            WHERE ese.Department_OLK_ID = 11605
            AND ese.Shop_OLK_ID = 11441
            AND qse.PRD_ID = 2793
            AND ese.WVN_ScheduleVersion = 1
            AND ese.ESE_ID = 1569669;

            DECLARE @p_EmploymentsAndDepartments
            TABLE(EmploymentID BIGINT, DepartmentOrganizationLinkID BIGINT, ESE_ID
BIGINT);
            INSERT INTO @p_EmploymentsAndDepartments
            SELECT DISTINCT
            ese.EMT_ID EmploymentID,
            ese.Department_OLK_ID DepartmentOrganizationLinkID,
            ese.ESE_ID
            FROM Schedule.EmployeeSchedule ese
```

```
            WHERE ese.Department_OLK_ID = 11605
            AND ese.Shop_OLK_ID = 11441
            AND ese.Department_PRD_ID = 2793
            AND ese.WVN_ScheduleVersion = 1
            AND ese.ESE_ID = 1569669;

            -- Get period info
            DECLARE @l_PeriodId bigint = 2882;
            DECLARE @l_StartDate DATE;
            DECLARE @l_EndDate DATE;
            SELECT        @l_StartDate = prd.[PRD_DateFrom],
                          @l_EndDate = prd.[PRD_DateUntil]
            FROM          [Reference].[Period] prd
            WHERE         prd.[PRD_ID] = @l_PeriodId;

            -- Create and fill a table that stores the latest schedule version for each
new employment/department combination
            DECLARE @l_NewEmploymentsDepartmentsScheduleversions AS TABLE
([EmploymentId] bigint, [DepartmentOrganizationLinkId] bigint, [ScheduleVersion] tinyint,
ESE_ID BIGINT);
            INSERT INTO @l_NewEmploymentsDepartmentsScheduleversions
            SELECT        employmentsAndDepartments.[EmploymentId],
employmentsAndDepartments.[DepartmentOrganizationLinkId], IIF (wpd.[WPD_ScheduleStatus] =
'FINAL', 1, 0 ), employmentsAndDepartments.ESE_ID
            FROM          @p_EmploymentsAndDepartments employmentsAndDepartments
            INNER JOIN [Organization].[OrganizationLink] olk
                  ON olk.[OLK_ID] =
employmentsAndDepartments.[DepartmentOrganizationLinkId]
            INNER JOIN    [Organization].[WorkPeriod] wpd
                  ON wpd.[OLK_ID] = olk.[Parent_OLK_ID]
                  AND wpd.[PRD_ID] = @l_PeriodId;

            -- Create and fill a table that stores the latest schedule version for each
employment/department combination
            DECLARE @l_OldEmploymentsDepartmentsScheduleversions AS TABLE
([EmploymentId] bigint, [DepartmentOrganizationLinkId] bigint, [ScheduleVersion] tinyint);
            INSERT INTO @l_OldEmploymentsDepartmentsScheduleversions
            SELECT Employments.[EmploymentId], ese.[Department_OLK_ID],
ese.[WVN_ScheduleVersion]
            FROM (SELECT DISTINCT employmentId from @p_EmploymentsAndDepartments) as
Employments
            INNER JOIN    [Schedule].[EmployeeSchedule] ese
                  ON ese.[EMT_ID] = Employments.[EmploymentId]
                  AND ese.[Department_PRD_ID] = @l_PeriodId
            INNER JOIN    [Organization].[WorkPeriod] wpd
                  ON wpd.[OLK_ID] = ese.[Shop_OLK_ID]
                  AND wpd.[PRD_ID] = @l_PeriodId
            WHERE ese.WVN_ScheduleVersion = IIF (wpd.[WPD_ScheduleStatus] = 'FINAL', 1,
0 );


            -- Delete all remunerations for the provided employments in this week


            DECLARE @l_QualificationScheduleDeletes TABLE(ID BIGINT);
```

```
		-- Delete all activities/pauses for the provided employments in this week
		INSERT INTO @l_QualificationScheduleDeletes
		SELECT qse.QSE_ID
		FROM		@l_OldEmploymentsDepartmentsScheduleversions
employmentsDepartmentsScheduleversions
		INNER JOIN	[Schedule].[EmployeeSchedule] ese
			ON ese.[Department_PRD_ID] = @l_PeriodId
			AND ese.[Department_OLK_ID] =
employmentsDepartmentsScheduleversions.[DepartmentOrganizationLinkId]
			AND ese.[WVN_ScheduleVersion] =
employmentsDepartmentsScheduleversions.[ScheduleVersion]
			AND ese.[EMT_ID] =
employmentsDepartmentsScheduleversions.[EmploymentId]
		INNER JOIN	[Schedule].[QualificationSchedule] qse
			ON qse.[Shop_OLK_ID] = ese.[Shop_OLK_ID]
			AND qse.[PRD_ID] = ese.[Department_PRD_ID]
			AND qse.[Department_OLK_ID] = ese.[Department_OLK_ID]
			AND qse.[WVN_ScheduleVersion] = ese.[WVN_ScheduleVersion]
			AND qse.[ESE_ID] = ese.[ESE_ID];


		DECLARE @l_ScheduleDeletes TABLE(ID BIGINT);

		-- Delete all 'empty' schedules in this week for the provided employments
		INSERT INTO @l_ScheduleDeletes
		SELECT ese.[ESE_ID]
		FROM [Schedule].[EmployeeSchedule] ese
		INNER JOIN @l_OldEmploymentsDepartmentsScheduleversions oeds
			ON oeds.EmploymentId = ese.EMT_ID
			AND oeds.DepartmentOrganizationLinkId = ese.Department_OLK_ID
			AND oeds.ScheduleVersion = ese.WVN_ScheduleVersion
		WHERE ese.[Department_PRD_ID] = @l_PeriodId
		AND NOT EXISTS (
			SELECT 0
			FROM @l_NewEmploymentsDepartmentsScheduleversions neds
			WHERE neds.EmploymentId = ese.EMT_ID
				AND neds.DepartmentOrganizationLinkId =
ese.Department_OLK_ID);


		DELETE [Schedule].[QualificationSchedule]
		FROM [Schedule].[QualificationSchedule] qse
		INNER JOIN @l_QualificationScheduleDeletes ids
			ON qse.[QSE_ID] = ids.[ID];

		DELETE [Schedule].[EmployeeSchedule]
		FROM [Schedule].[EmployeeSchedule] ese
		INNER JOIN @l_ScheduleDeletes ids
			ON ese.[ESE_ID] = ids.[ID];

		-- Ensure that schedules exist
		MERGE [Schedule].[EmployeeSchedule] AS target
		USING
		(
```

```
                        SELECT          employmentsDepartmentsScheduleversions.[EmploymentId],

        employmentsDepartmentsScheduleversions.[DepartmentOrganizationLinkId],

        employmentsDepartmentsScheduleversions.[ScheduleVersion],
                                        employmentsDepartmentsScheduleversions.ESE_ID,
                                        wvn.[WVN_ID] AS [WorkVersionId],
                                        olk.[Parent_OLK_ID] AS [ShopOrganizationLinkId]
                        FROM            @l_NewEmploymentsDepartmentsScheduleversions
employmentsDepartmentsScheduleversions
                        INNER JOIN      [Organization].[OrganizationLink] olk
                                ON olk.[OLK_ID] =
employmentsDepartmentsScheduleversions.[DepartmentOrganizationLinkId]
                        INNER JOIN      [Organization].[WorkPeriod] wpd
                                ON wpd.[OLK_ID] = olk.[OLK_ID]
                                AND wpd.[PRD_ID] = @l_PeriodId
                        INNER JOIN      [Organization].[WorkVersion] wvn
                                ON wvn.[WVN_Type] = 'SVN'
                                AND wvn.[WPD_ID] = wpd.[WPD_ID]
                                AND wvn.[WVN_Version] =
employmentsDepartmentsScheduleversions.[ScheduleVersion]
                ) AS source
                ON
                (
                        target.[Shop_OLK_ID] = source.[ShopOrganizationLinkId]
                        AND target.[Department_OLK_ID] =
source.[DepartmentOrganizationLinkId]
                        AND target.ESE_ID = source.ESE_ID
                        AND target.[Department_PRD_ID] = @l_PeriodId
                        AND target.[WVN_ScheduleVersion] =source.[ScheduleVersion]
                        AND target.[EMT_ID] = source.[EmploymentId]
                        AND target.[SVN_ID] = source.[WorkVersionId]
                )
                WHEN NOT MATCHED THEN
                        INSERT(Shop_OLK_ID, Department_OLK_ID, Department_PRD_ID, SVN_ID,
EMT_ID, ESE_Validated, ESE_RemunerationCost, ESE_LastCheckedForIssues,
WVN_ScheduleVersion)
                        VALUES(source.[ShopOrganizationLinkId],
source.[DepartmentOrganizationLinkId], @l_PeriodId, source.[WorkVersionId],
source.[EmploymentId], 'Y', 0.00, GetDate(), source.[ScheduleVersion]);

                -- Create a mapping table from the provided ID's to the actual ID's
                DECLARE @l_ActivityIdMapping AS TABLE ([Incoming] bigint, [Generated]
bigint);

                -- Insert the new activities
                MERGE INTO      [Schedule].[QualificationSchedule]
                USING
                (
                        SELECT activities.[Id],
                                        activities.[Source],
                                        activities.[Date],
                                        activities.[StartTime],
                                        activities.[EndTime],
                                        activities.[Duration],
                                        activities.[ActivityTypeId],
```

```
                                activities.[DepartmentOrganizationLinkId],
                                ese.[ESE_ID] as [EmployeeScheduleId],
                                qun.[QUN_ID] as [QualificationId],
                                ese.[Shop_OLK_ID] as [ShopOrganizationLinkId],
                                employmentsDepartmentsScheduleversions.[ScheduleVersion]
                FROM   @p_Activities activities
                INNER JOIN    @l_NewEmploymentsDepartmentsScheduleversions
employmentsDepartmentsScheduleversions
                        ON activities.[EmploymentId] =
employmentsDepartmentsScheduleversions.[EmploymentId]
                        AND activities.[DepartmentOrganizationLinkId] =
employmentsDepartmentsScheduleversions.[DepartmentOrganizationLinkId]
                INNER JOIN    [Schedule].[EmployeeSchedule] ese
                        ON ese.[Department_PRD_ID] = @l_PeriodId
                        AND ese.[Department_OLK_ID] =
employmentsDepartmentsScheduleversions.[DepartmentOrganizationLinkId]
                        AND ese.[WVN_ScheduleVersion] =
employmentsDepartmentsScheduleversions.[ScheduleVersion]
                        AND ese.[EMT_ID] =
employmentsDepartmentsScheduleversions.[EmploymentId]
                INNER JOIN [Organization].[Qualification] qun
                        ON qun.[OLK_ID] = ese.[Department_OLK_ID]
                        AND qun.[EMT_ID] = ese.[EMT_ID]
                        AND Schedule.[fn_D_OverlapsInclusive](@l_StartDate,
@l_EndDate, qun.[QUN_DateFrom], qun.[QUN_DateUntil]) = 1
                WHERE  activities.[ParentActivityId] IS NULL
        ) AS source ON 1 = 0
        WHEN NOT MATCHED THEN
                INSERT ([QSE_Source], [QUN_ID], [QSE_TimeFrom], [QSE_TimeUntil],
[QSE_NettHours], [STK_ID], [ESE_ID], [Shop_OLK_ID], [PRD_ID], [Department_OLK_ID],
[WVN_ScheduleVersion])
                VALUES
                (
                        source.[Source],
                        source.[QualificationId],
                        (CAST(source.[Date] AS DATETIME) + CAST(source.[StartTime] AS
DATETIME)),
                        (CAST(source.[Date] AS DATETIME) + CAST(source.[EndTime] AS
DATETIME)),
                        source.[Duration],
                        source.[ActivityTypeId],
                        source.[EmployeeScheduleId],
                        source.[ShopOrganizationLinkId],
                        @l_PeriodId,
                        source.[DepartmentOrganizationLinkId],
                        source.ScheduleVersion
                )
        OUTPUT source.[Id], inserted.[QSE_ID] INTO @l_ActivityIdMapping;

        -- Insert the new pauses
        MERGE INTO    [Schedule].[QualificationSchedule]
        USING
        (
                SELECT activities.[Id],
                        activities.[Source],
                        activities.[Date],
```

```
                                activities.[StartTime],
                                activities.[EndTime],
                                activities.[Duration],
                                activities.[ActivityTypeId],
                                activities.[DepartmentOrganizationLinkId],
                                idMapping.[Generated] AS [ParentActivityId],
                                ese.[ESE_ID] as [EmployeeScheduleId],
                                qun.[QUN_ID] as [QualificationId],
                                ese.[Shop_OLK_ID] as [ShopOrganizationLinkId],
                                employmentsDepartmentsScheduleversions.[ScheduleVersion]
                        FROM    @p_Activities activities
                        INNER JOIN    @l_NewEmploymentsDepartmentsScheduleversions
employmentsDepartmentsScheduleversions
                                ON activities.[EmploymentId] =
employmentsDepartmentsScheduleversions.[EmploymentId]
                                AND activities.[DepartmentOrganizationLinkId] =
employmentsDepartmentsScheduleversions.[DepartmentOrganizationLinkId]
                        INNER JOIN    [Schedule].[EmployeeSchedule] ese
                                ON ese.[Department_PRD_ID] = @l_PeriodId
                                AND ese.[Department_OLK_ID] =
activities.[DepartmentOrganizationLinkId]
                                AND ese.[WVN_ScheduleVersion] =
employmentsDepartmentsScheduleversions.[ScheduleVersion]
                                AND ese.[EMT_ID] = activities.[EmploymentId]
                        INNER JOIN    @l_ActivityIdMapping idMapping ON
activities.[ParentActivityId] = idMapping.[Incoming]
                        INNER JOIN [Organization].[Qualification] qun
                                ON qun.[OLK_ID] = ese.[Department_OLK_ID]
                                AND qun.[EMT_ID] = ese.[EMT_ID]
                                AND Schedule.[fn_D_OverlapsInclusive](@l_StartDate,
@l_EndDate, qun.[QUN_DateFrom], qun.[QUN_DateUntil]) = 1
                        WHERE  activities.[ParentActivityId] IS NOT NULL
                ) AS source ON 1 = 0
                WHEN NOT MATCHED THEN
                        INSERT ([QSE_Source], [QUN_ID], [QSE_TimeFrom], [QSE_TimeUntil],
[QSE_NettHours], [STK_ID], [ESE_ID], [Parent_QSE_ID], [Shop_OLK_ID], [PRD_ID],
[Department_OLK_ID], [WVN_ScheduleVersion])
                        VALUES
                        (
                                source.[Source],
                                source.[QualificationId],
                                (CAST(source.[Date] AS DATETIME) + CAST(source.[StartTime] AS
DATETIME)),
                                (CAST(source.[Date] AS DATETIME) + CAST(source.[EndTime] AS
DATETIME)),
                                source.[Duration],
                                source.[ActivityTypeId],
                                source.[EmployeeScheduleId],
                                source.[ParentActivityId],
                                source.[ShopOrganizationLinkId],
                                @l_PeriodId,
                                source.[DepartmentOrganizationLinkId],
                                source.ScheduleVersion
                        )
                OUTPUT source.[Id], inserted.[QSE_ID] INTO @l_ActivityIdMapping;
```

```
		-- Insert the new absence activities
		MERGE INTO	[Schedule].[QualificationSchedule]
		USING
		(
			SELECT activities.[Id],
				activities.[Source],
				activities.[Date],
				activities.[Duration],
				activities.[ActivityTypeId],
				activities.[AbsenceReferenceId],
				activities.[DepartmentOrganizationLinkId],
				ese.[ESE_ID] as [EmployeeScheduleId],
				qun.[QUN_ID] as [QualificationId],
				ese.[Shop_OLK_ID] as [ShopOrganizationLinkId],
				employmentsDepartmentsScheduleversions.[ScheduleVersion]
			FROM	@p_AbsenceActivities activities
			INNER JOIN	@l_NewEmploymentsDepartmentsScheduleversions
employmentsDepartmentsScheduleversions
				ON activities.[EmploymentId] =
employmentsDepartmentsScheduleversions.[EmploymentId]
				AND activities.[DepartmentOrganizationLinkId] =
employmentsDepartmentsScheduleversions.[DepartmentOrganizationLinkId]
			INNER JOIN	[Schedule].[EmployeeSchedule] ese
				ON ese.[Department_PRD_ID] = @l_PeriodId
				AND ese.[Department_OLK_ID] =
activities.[DepartmentOrganizationLinkId]
				AND ese.[WVN_ScheduleVersion] =
employmentsDepartmentsScheduleversions.[ScheduleVersion]
				AND ese.[EMT_ID] = activities.[EmploymentId]
			INNER JOIN [Organization].[Qualification] qun
				ON qun.[OLK_ID] = ese.[Department_OLK_ID]
				AND qun.[EMT_ID] = ese.[EMT_ID]
				AND Schedule.[fn_D_OverlapsInclusive](@l_StartDate,
@l_EndDate, qun.[QUN_DateFrom], qun.[QUN_DateUntil]) = 1
		) AS source ON 1 = 0
		WHEN NOT MATCHED THEN
			INSERT ([QSE_Source], [QUN_ID], [QSE_NettHours], [STK_ID], [ESE_ID],
			[ILE_ID], [QSE_ScheduleDate], [Shop_OLK_ID], [PRD_ID],
			[Department_OLK_ID], [WVN_ScheduleVersion])
			VALUES
			(
				source.[Source],
				source.[QualificationId],
				source.[Duration],
				source.[ActivityTypeId],
				source.[EmployeeScheduleId],
				source.[AbsenceReferenceId],
				source.[Date],
				source.[ShopOrganizationLinkId],
				@l_PeriodId,
				source.[DepartmentOrganizationLinkId],
				source.ScheduleVersion
			)
		OUTPUT source.[Id], inserted.[QSE_ID] INTO @l_ActivityIdMapping;
	END TRY
```

```
	BEGIN CATCH
		THROW;
	END CATCH
END
```

Bulk

De volgende stored procedure is uitgevoerd op de SQL-server database voor de bulk insert:

```
CREATE PROCEDURE [Schedule].[csp_PersistLaborAgreementEmploymentDataBulk]
AS
BEGIN
	SET NOCOUNT ON;
	BEGIN TRY

		DECLARE @p_AbsenceActivities
		TABLE(ID BIGINT, EmploymentID BIGINT, DepartmentOrganizationLinkID BIGINT,
Source char(2), Date DateTime, StartTime DateTime, EndTime DateTime, ActivityTypeId
bigint, ParentActivityId BIGINT, Duration int, AbsenceReferenceId BIGINT);
		INSERT INTO @p_AbsenceActivities
		SELECT DISTINCT
		qse.QSE_ID,
		ese.EMT_ID EmploymentID,
		ese.Department_OLK_ID DepartmentOrganizationLinkID,
		qse.QSE_Source,
		qse.QSE_TimeFrom,
		qse.QSE_TimeFrom,
		qse.QSE_TimeUntil,
		qse.STK_ID,
		qse.Parent_QSE_ID,
		qse.QSE_NettHours,
		qse.ILE_ID
		FROM Schedule.EmployeeSchedule ese
		JOIN Schedule.QualificationSchedule qse
		ON ese.ESE_ID = qse.ESE_ID
		WHERE ese.Shop_OLK_ID = 11441
		AND qse.PRD_ID = 2793
		AND ese.WVN_ScheduleVersion = 1;

		DECLARE @p_Activities
		TABLE(ID BIGINT, EmploymentID BIGINT, DepartmentOrganizationLinkID BIGINT,
Source char(2), Date DateTime, StartTime DateTime, EndTime DateTime, ActivityTypeId
bigint, ParentActivityId BIGINT, Duration int);
		INSERT INTO @p_Activities
		SELECT DISTINCT
		qse.QSE_ID,
		ese.EMT_ID EmploymentID,
		ese.Department_OLK_ID DepartmentOrganizationLinkID,
		qse.QSE_Source,
		qse.QSE_TimeFrom,
		qse.QSE_TimeFrom,
		qse.QSE_TimeUntil,
		qse.STK_ID,
		qse.Parent_QSE_ID,
		qse.QSE_NettHours
		FROM Schedule.EmployeeSchedule ese
		JOIN Schedule.QualificationSchedule qse
		ON ese.ESE_ID = qse.ESE_ID
```

```
            WHERE  ese.Shop_OLK_ID = 11441
            AND qse.PRD_ID = 2793
            AND ese.WVN_ScheduleVersion = 1;
            DECLARE @p_EmploymentsAndDepartments
            TABLE(EmploymentID BIGINT, DepartmentOrganizationLinkID BIGINT, ESE_ID
BIGINT);
            INSERT INTO @p_EmploymentsAndDepartments
            SELECT DISTINCT
            ese.EMT_ID EmploymentID,
            ese.Department_OLK_ID DepartmentOrganizationLinkID,
            ese.ESE_ID
            FROM Schedule.EmployeeSchedule ese
            WHERE ese.Shop_OLK_ID = 11441
            AND ese.Department_PRD_ID = 2793
            AND ese.WVN_ScheduleVersion = 1;

            -- Get period info
            DECLARE @l_PeriodId bigint = 2793;
            DECLARE @l_StartDate DATE;
            DECLARE @l_EndDate DATE;
            SELECT        @l_StartDate = prd.[PRD_DateFrom],
                          @l_EndDate = prd.[PRD_DateUntil]
            FROM          [Reference].[Period] prd
            WHERE         prd.[PRD_ID] = @l_PeriodId;

            -- Create and fill a table that stores the latest schedule version for each
new employment/department combination
            DECLARE @l_NewEmploymentsDepartmentsScheduleversions AS TABLE
([EmploymentId] bigint, [DepartmentOrganizationLinkId] bigint, [ScheduleVersion] tinyint,
ESE_ID BIGINT);
            INSERT INTO @l_NewEmploymentsDepartmentsScheduleversions
            SELECT        employmentsAndDepartments.[EmploymentId],
employmentsAndDepartments.[DepartmentOrganizationLinkId], IIF (wpd.[WPD_ScheduleStatus] =
'FINAL', 1, 0 ), employmentsAndDepartments.ESE_ID
            FROM          @p_EmploymentsAndDepartments employmentsAndDepartments
            INNER JOIN [Organization].[OrganizationLink] olk
                  ON olk.[OLK_ID] =
employmentsAndDepartments.[DepartmentOrganizationLinkId]
            INNER JOIN    [Organization].[WorkPeriod] wpd
                  ON wpd.[OLK_ID] = olk.[Parent_OLK_ID]
                  AND wpd.[PRD_ID] = @l_PeriodId;

            -- Create and fill a table that stores the latest schedule version for each
employment/department combination
            DECLARE @l_OldEmploymentsDepartmentsScheduleversions AS TABLE
([EmploymentId] bigint, [DepartmentOrganizationLinkId] bigint, [ScheduleVersion] tinyint);
            INSERT INTO @l_OldEmploymentsDepartmentsScheduleversions
            SELECT Employments.[EmploymentId], ese.[Department_OLK_ID],
ese.[WVN_ScheduleVersion]
            FROM (SELECT DISTINCT employmentId from @p_EmploymentsAndDepartments) as
Employments
            INNER JOIN    [Schedule].[EmployeeSchedule] ese
                  ON ese.[EMT_ID] = Employments.[EmploymentId]
                  AND ese.[Department_PRD_ID] = @l_PeriodId
            INNER JOIN    [Organization].[WorkPeriod] wpd
                  ON wpd.[OLK_ID] = ese.[Shop_OLK_ID]
```

```
                AND wpd.[PRD_ID] = @l_PeriodId
            WHERE ese.WVN_ScheduleVersion = IIF (wpd.[WPD_ScheduleStatus] = 'FINAL', 1,
0 );

            -- Delete all remunerations for the provided employments in this week

            DECLARE @l_QualificationScheduleDeletes TABLE(ID BIGINT);

            -- Delete all activities/pauses for the provided employments in this week
            INSERT INTO @l_QualificationScheduleDeletes
            SELECT qse.QSE_ID
            FROM          @l_OldEmploymentsDepartmentsScheduleversions
employmentsDepartmentsScheduleversions
            INNER JOIN    [Schedule].[EmployeeSchedule] ese
                  ON ese.[Department_PRD_ID] = @l_PeriodId
                  AND ese.[Department_OLK_ID] =
employmentsDepartmentsScheduleversions.[DepartmentOrganizationLinkId]
                  AND ese.[WVN_ScheduleVersion] =
employmentsDepartmentsScheduleversions.[ScheduleVersion]
                  AND ese.[EMT_ID] =
employmentsDepartmentsScheduleversions.[EmploymentId]
            INNER JOIN    [Schedule].[QualificationSchedule] qse
                  ON qse.[Shop_OLK_ID] = ese.[Shop_OLK_ID]
                  AND qse.[PRD_ID] = ese.[Department_PRD_ID]
                  AND qse.[Department_OLK_ID] = ese.[Department_OLK_ID]
                  AND qse.[WVN_ScheduleVersion] = ese.[WVN_ScheduleVersion]
                  AND qse.[ESE_ID] = ese.[ESE_ID];

            DECLARE @l_ScheduleDeletes TABLE(ID BIGINT);

            -- Delete all 'empty' schedules in this week for the provided employments
            INSERT INTO @l_ScheduleDeletes
            SELECT ese.[ESE_ID]
            FROM [Schedule].[EmployeeSchedule] ese
            INNER JOIN @l_OldEmploymentsDepartmentsScheduleversions oeds
                  ON oeds.EmploymentId = ese.EMT_ID
                  AND oeds.DepartmentOrganizationLinkId = ese.Department_OLK_ID
                  AND oeds.ScheduleVersion = ese.WVN_ScheduleVersion
            WHERE ese.[Department_PRD_ID] = @l_PeriodId
            AND NOT EXISTS (
                  SELECT 0
                  FROM @l_NewEmploymentsDepartmentsScheduleversions neds
                  WHERE neds.EmploymentId = ese.EMT_ID
                        AND neds.DepartmentOrganizationLinkId =
ese.Department_OLK_ID);

            DELETE [Schedule].[QualificationSchedule]
            FROM [Schedule].[QualificationSchedule] qse
            INNER JOIN @l_QualificationScheduleDeletes ids
                  ON qse.[QSE_ID] = ids.[ID];

            DELETE [Schedule].[EmployeeSchedule]
            FROM [Schedule].[EmployeeSchedule] ese
            INNER JOIN @l_ScheduleDeletes ids
                  ON ese.[ESE_ID] = ids.[ID];
```

```
		-- Ensure that schedules exist
		MERGE [Schedule].[EmployeeSchedule] AS target
		USING
		(
			SELECT		employmentsDepartmentsScheduleversions.[EmploymentId],

	employmentsDepartmentsScheduleversions.[DepartmentOrganizationLinkId],

	employmentsDepartmentsScheduleversions.[ScheduleVersion],
					employmentsDepartmentsScheduleversions.ESE_ID,
					wvn.[WVN_ID] AS [WorkVersionId],
					olk.[Parent_OLK_ID] AS [ShopOrganizationLinkId]
			FROM		@l_NewEmploymentsDepartmentsScheduleversions
employmentsDepartmentsScheduleversions
			INNER JOIN	[Organization].[OrganizationLink] olk
				ON olk.[OLK_ID] =
employmentsDepartmentsScheduleversions.[DepartmentOrganizationLinkId]
			INNER JOIN	[Organization].[WorkPeriod] wpd
				ON wpd.[OLK_ID] = olk.[OLK_ID]
				AND wpd.[PRD_ID] = @l_PeriodId
			INNER JOIN	[Organization].[WorkVersion] wvn
				ON wvn.[WVN_Type] = 'SVN'
				AND wvn.[WPD_ID] = wpd.[WPD_ID]
				AND wvn.[WVN_Version] =
employmentsDepartmentsScheduleversions.[ScheduleVersion]
		) AS source
		ON
		(
			target.[Shop_OLK_ID] = source.[ShopOrganizationLinkId]
			AND target.[Department_OLK_ID] =
source.[DepartmentOrganizationLinkId]
			AND target.ESE_ID = source.ESE_ID
			AND target.[Department_PRD_ID] = @l_PeriodId
			AND target.[WVN_ScheduleVersion] =source.[ScheduleVersion]
			AND target.[EMT_ID] = source.[EmploymentId]
			AND target.[SVN_ID] = source.[WorkVersionId]
		)
		WHEN NOT MATCHED THEN
			INSERT(Shop_OLK_ID, Department_OLK_ID, Department_PRD_ID, SVN_ID,
EMT_ID, ESE_Validated, ESE_RemunerationCost, ESE_LastCheckedForIssues,
WVN_ScheduleVersion)
			VALUES(source.[ShopOrganizationLinkId],
source.[DepartmentOrganizationLinkId], @l_PeriodId, source.[WorkVersionId],
source.[EmploymentId], 'Y', 0.00, GetDate(), source.[ScheduleVersion]);

		-- Create a mapping table from the provided ID's to the actual ID's
		DECLARE @l_ActivityIdMapping AS TABLE ([Incoming] bigint, [Generated]
bigint);

		-- Insert the new activities
		MERGE INTO	[Schedule].[QualificationSchedule]
		USING
		(
			SELECT activities.[Id],
				activities.[Source],
				activities.[Date],
```

```
                                    activities.[StartTime],
                                    activities.[EndTime],
                                    activities.[Duration],
                                    activities.[ActivityTypeId],
                                    activities.[DepartmentOrganizationLinkId],
                                    ese.[ESE_ID] as [EmployeeScheduleId],
                                    qun.[QUN_ID] as [QualificationId],
                                    ese.[Shop_OLK_ID] as [ShopOrganizationLinkId],
                             employmentsDepartmentsScheduleversions.[ScheduleVersion]
                      FROM   @p_Activities activities
                      INNER JOIN    @l_NewEmploymentsDepartmentsScheduleversions
employmentsDepartmentsScheduleversions
                             ON activities.[EmploymentId] =
employmentsDepartmentsScheduleversions.[EmploymentId]
                             AND activities.[DepartmentOrganizationLinkId] =
employmentsDepartmentsScheduleversions.[DepartmentOrganizationLinkId]
                      INNER JOIN    [Schedule].[EmployeeSchedule] ese
                             ON ese.[Department_PRD_ID] = @l_PeriodId
                             AND ese.[Department_OLK_ID] =
employmentsDepartmentsScheduleversions.[DepartmentOrganizationLinkId]
                             AND ese.[WVN_ScheduleVersion] =
employmentsDepartmentsScheduleversions.[ScheduleVersion]
                             AND ese.[EMT_ID] =
employmentsDepartmentsScheduleversions.[EmploymentId]
                      INNER JOIN [Organization].[Qualification] qun
                             ON qun.[OLK_ID] = ese.[Department_OLK_ID]
                             AND qun.[EMT_ID] = ese.[EMT_ID]
                             AND Schedule.[fn_D_OverlapsInclusive](@l_StartDate,
@l_EndDate, qun.[QUN_DateFrom], qun.[QUN_DateUntil]) = 1
                      WHERE  activities.[ParentActivityId] IS NULL
               ) AS source ON 1 = 0
               WHEN NOT MATCHED THEN
                      INSERT ([QSE_Source], [QUN_ID], [QSE_TimeFrom], [QSE_TimeUntil],
[QSE_NettHours], [STK_ID], [ESE_ID], [Shop_OLK_ID], [PRD_ID], [Department_OLK_ID],
[WVN_ScheduleVersion])
                      VALUES
                      (
                             source.[Source],
                             source.[QualificationId],
                             (CAST(source.[Date] AS DATETIME) + CAST(source.[StartTime] AS
DATETIME)),
                             (CAST(source.[Date] AS DATETIME) + CAST(source.[EndTime] AS
DATETIME)),
                             source.[Duration],
                             source.[ActivityTypeId],
                             source.[EmployeeScheduleId],
                             source.[ShopOrganizationLinkId],
                             @l_PeriodId,
                             source.[DepartmentOrganizationLinkId],
                             source.ScheduleVersion
                      )
               OUTPUT source.[Id], inserted.[QSE_ID] INTO @l_ActivityIdMapping;

               -- Insert the new pauses
               MERGE INTO    [Schedule].[QualificationSchedule]
               USING
```

```
		(
			SELECT activities.[Id],
					activities.[Source],
					activities.[Date],
					activities.[StartTime],
					activities.[EndTime],
					activities.[Duration],
					activities.[ActivityTypeId],
					activities.[DepartmentOrganizationLinkId],
					idMapping.[Generated] AS [ParentActivityId],
					ese.[ESE_ID] as [EmployeeScheduleId],
					qun.[QUN_ID] as [QualificationId],
					ese.[Shop_OLK_ID] as [ShopOrganizationLinkId],
				employmentsDepartmentsScheduleversions.[ScheduleVersion]
			FROM	@p_Activities activities
			INNER JOIN	@l_NewEmploymentsDepartmentsScheduleversions
employmentsDepartmentsScheduleversions
				ON activities.[EmploymentId] =
employmentsDepartmentsScheduleversions.[EmploymentId]
				AND activities.[DepartmentOrganizationLinkId] =
employmentsDepartmentsScheduleversions.[DepartmentOrganizationLinkId]
			INNER JOIN	[Schedule].[EmployeeSchedule] ese
				ON ese.[Department_PRD_ID] = @l_PeriodId
				AND ese.[Department_OLK_ID] =
activities.[DepartmentOrganizationLinkId]
				AND ese.[WVN_ScheduleVersion] =
employmentsDepartmentsScheduleversions.[ScheduleVersion]
				AND ese.[EMT_ID] = activities.[EmploymentId]
			INNER JOIN	@l_ActivityIdMapping idMapping ON
activities.[ParentActivityId] = idMapping.[Incoming]
			INNER JOIN [Organization].[Qualification] qun
				ON qun.[OLK_ID] = ese.[Department_OLK_ID]
				AND qun.[EMT_ID] = ese.[EMT_ID]
				AND Schedule.[fn_D_OverlapsInclusive](@l_StartDate,
@l_EndDate, qun.[QUN_DateFrom], qun.[QUN_DateUntil]) = 1
			WHERE  activities.[ParentActivityId] IS NOT NULL
		) AS source ON 1 = 0
		WHEN NOT MATCHED THEN
			INSERT ([QSE_Source], [QUN_ID], [QSE_TimeFrom], [QSE_TimeUntil],
[QSE_NettHours], [STK_ID], [ESE_ID], [Parent_QSE_ID], [Shop_OLK_ID], [PRD_ID],
[Department_OLK_ID], [WVN_ScheduleVersion])
			VALUES
			(
				source.[Source],
				source.[QualificationId],
				(CAST(source.[Date] AS DATETIME) + CAST(source.[StartTime] AS
DATETIME)),
				(CAST(source.[Date] AS DATETIME) + CAST(source.[EndTime] AS
DATETIME)),
				source.[Duration],
				source.[ActivityTypeId],
				source.[EmployeeScheduleId],
				source.[ParentActivityId],
				source.[ShopOrganizationLinkId],
				@l_PeriodId,
				source.[DepartmentOrganizationLinkId],
```

```
                        source.ScheduleVersion
                )
        OUTPUT source.[Id], inserted.[QSE_ID] INTO @l_ActivityIdMapping;

        -- Insert the new absence activities
        MERGE INTO    [Schedule].[QualificationSchedule]
        USING
        (
                SELECT activities.[Id],
                        activities.[Source],
                        activities.[Date],
                        activities.[Duration],
                        activities.[ActivityTypeId],
                        activities.[AbsenceReferenceId],
                        activities.[DepartmentOrganizationLinkId],
                        ese.[ESE_ID] as [EmployeeScheduleId],
                        qun.[QUN_ID] as [QualificationId],
                        ese.[Shop_OLK_ID] as [ShopOrganizationLinkId],
                        employmentsDepartmentsScheduleversions.[ScheduleVersion]
                FROM   @p_AbsenceActivities activities
                INNER JOIN    @l_NewEmploymentsDepartmentsScheduleversions
employmentsDepartmentsScheduleversions
                        ON activities.[EmploymentId] =
employmentsDepartmentsScheduleversions.[EmploymentId]
                        AND activities.[DepartmentOrganizationLinkId] =
employmentsDepartmentsScheduleversions.[DepartmentOrganizationLinkId]
                INNER JOIN    [Schedule].[EmployeeSchedule] ese
                        ON ese.[Department_PRD_ID] = @l_PeriodId
                        AND ese.[Department_OLK_ID] =
activities.[DepartmentOrganizationLinkId]
                        AND ese.[WVN_ScheduleVersion] =
employmentsDepartmentsScheduleversions.[ScheduleVersion]
                        AND ese.[EMT_ID] = activities.[EmploymentId]
                INNER JOIN [Organization].[Qualification] qun
                        ON qun.[OLK_ID] = ese.[Department_OLK_ID]
                        AND qun.[EMT_ID] = ese.[EMT_ID]
                        AND Schedule.[fn_D_OverlapsInclusive](@l_StartDate,
@l_EndDate, qun.[QUN_DateFrom], qun.[QUN_DateUntil]) = 1
        ) AS source ON 1 = 0
        WHEN NOT MATCHED THEN
                INSERT ([QSE_Source], [QUN_ID], [QSE_NettHours], [STK_ID], [ESE_ID],
[ILE_ID], [QSE_ScheduleDate], [Shop_OLK_ID], [PRD_ID], [Department_OLK_ID],
[WVN_ScheduleVersion])
                VALUES
                (
                        source.[Source],
                        source.[QualificationId],
                        source.[Duration],
                        source.[ActivityTypeId],
                        source.[EmployeeScheduleId],
                        source.[AbsenceReferenceId],
                        source.[Date],
                        source.[ShopOrganizationLinkId],
                        @l_PeriodId,
                        source.[DepartmentOrganizationLinkId],
                        source.ScheduleVersion
```

```
                    )
            OUTPUT source.[Id], inserted.[QSE_ID] INTO @l_ActivityIdMapping;
    END TRY
    BEGIN CATCH
            THROW;
    END CATCH
END
```

Uitvoering:

Select
De volgende resultaten kwamen van het uitvoeren van de select statement:

```
14  exec Schedule.csp_GetShopSchedule8
```

100 %

Results | Messages

	EmploymentId	DepartmentOrganizationLinkIdWorkedFor	Date	ActivityId	ParentActivityId	StartTime	EndTime	ActivityTypeId	HiredIn
1	121340	11543	2013-04-23	6754467	NULL	08:30:00.0000000	17:00:00.0000000	1	0
2	121340	11543	2013-04-24	6754523	NULL	08:00:00.0000000	17:00:00.0000000	1	0
3	121340	11543	2013-04-25	6754546	NULL	08:30:00.0000000	17:00:00.0000000	1	0
4	121340	11543	2013-04-28	6754568	NULL	10:00:00.0000000	19:00:00.0000000	1	0
5	121340	11543	2013-04-23	6754310	6754467	08:31:00.0000000	09:30:00.0000000	6	0
6	121340	11543	2013-04-24	6754311	6754523	08:01:00.0000000	09:00:00.0000000	6	0
7	121340	11543	2013-04-25	6754312	6754546	08:31:00.0000000	09:30:00.0000000	6	0
8	121340	11543	2013-04-28	6754313	6754568	10:01:00.0000000	11:00:00.0000000	6	0

	EmploymentId	DepartmentOrganizationLinkId	Date	ActivityTypeId	Duration

Query executed successfully. | XAVYR (11.0 RTM) | XAVYR\xavyr (58) | RR-performancetest | 00:00:02 | 8 rows

De stored procedure is dus uitgevoerd in 2 seconden.

Insert
De volgende resultaten kwamen van het uitvoeren van de insert procedure:

```
15  exec Schedule.csp_PersistLaborAgreementEmploymentData
```

100 %

Messages

```
Command(s) completed successfully.
```

100 %

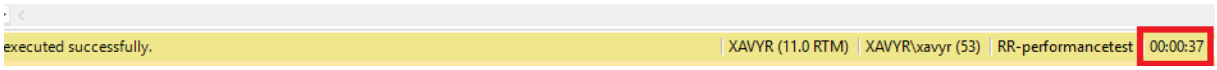
Query executed successfully. | XAVYR (11.0 RTM) | XAVYR\xavyr (58) | RR-performancetest | 00:00:26 | 0 rows

De stored procedure deed er dus 26 seconden over om af te ronden.

Bulk
De uitvoering van de stored procedure leverde het volgende resultaat op:

```
exec Schedule.csp_PersistLaborAgreementEmploymentDataBulk
```

```
ges
nd(s) completed successfully.
```



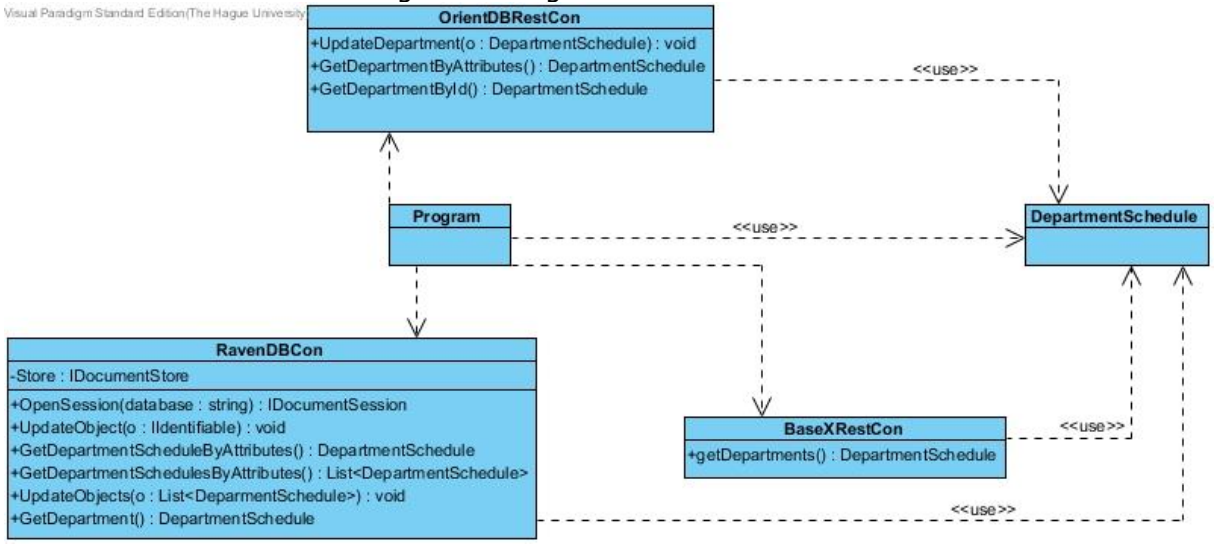
SQL-server deed er dus 37 seconden over om af te ronden.

I.2 Document databases

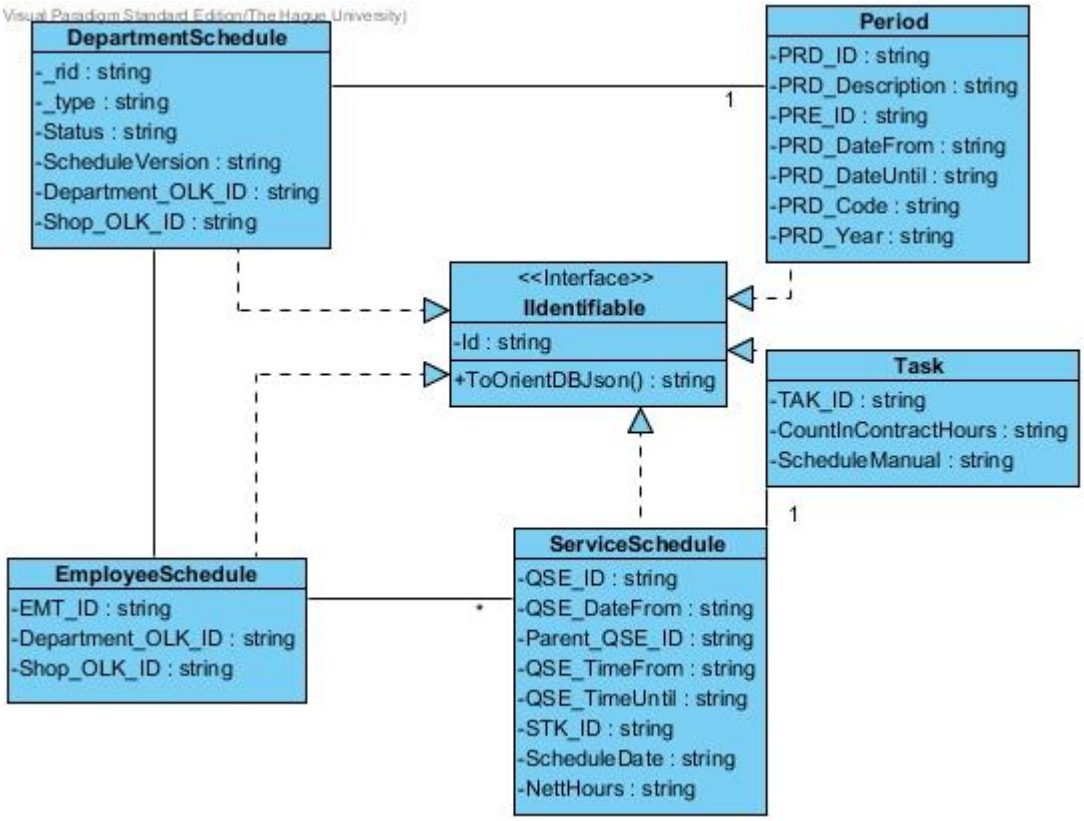
Voor de document databases zijn andere query's gebruikt. De reden dat er andere query's zijn gebruikt is, dat deze databases op een andere manier werken. Daar de afdelingsroosters als losse documents zijn genomen, hoeven de hele stored procedures niet uitgevoerd te worden.

Daarnaast is er voor de insert en de bulk handelingen een console applicatie geschreven. De reden dat er een console applicatie is geschreven is, dat RavenDB geen GUI heeft om insert/update query's uit te voeren. Voor deze database zou er dus een applicatie geschreven moeten worden. Om 1 lijn te trekken is er besloten voor alle databases dezelfde applicatie te gebruiken.

De applicatie is geschreven in C#, en de gebruikte code wordt per database gegeven. De structuur van de database zag er als volgt uit:



De models (en de nieuwe collection in de document database) zien er als volgt uit:



I.2.1 BaseX

Hieronder wordt beschreven hoe de tests zijn uitgevoerd in BaseX en wat de resultaten hiervan waren.

Aanpak:
De query's op BaseX heb ik als volgt aangepakt:
Select
Alle tabellen van de SQL-server database en de nieuwe collection zijn naar XML files weggeschreven. Deze zijn ingeladen in BaseX. Vervolgens is de DepartmentSchedule opgevraagd aan de hand van de volgende waardes:

- Shop_OLK_ID
- Department_OLK_ID
- ScheduleVersion
- Period.PRD_ID

De query is uitgevoerd in XQuery. Dit is gedaan via de GUI.

De query zag er als volgt uit:
De volgende query is uitgevoerd in BaseX:

```
for $DepartmentSchedule
in db:open('BaseXFiles')/DepartmentSchedules/DepartmentSchedule
where $DepartmentSchedule/Shop_OLK_ID=11521 and
$DepartmentSchedule/Department_OLK_ID=11543 and
$DepartmentSchedule/ScheduleVersion=1 and
$DepartmentSchedule/Period/PRD_ID=2882
```

```
return $DepartmentSchedule
```

Insert
De insert statement van BaseX is via een console applicatie gedaan. De applicatie haalde 1 document op, verwijderde het document uit de database, en voegde het document weer toe. Hiervoor is de volgende code gebruikt:

In program:

```
DateTime start = DateTime.Now;
BaseXDBRestCon con = new BaseXDBRestCon();
DepartmentSchedule d = con.GetDepartmentByAttributes();
Console.WriteLine("Get success");
con.DeleteSingleDepartmentScheduleByAttribute();
Console.WriteLine("delete success");
con.SaveObject(d);
Console.WriteLine("Save success");
DateTime end = DateTime.Now;
TimeSpan diff = end - start;
Console.Write("totaltime: " + diff);
Console.ReadKey();
```

Het ophalen van een document:

```
public DepartmentSchedule GetDepartmentByAttributes()
{
    string xquery = "for $DepartmentSchedule in db:open('BaseXFiles')/DepartmentSchedules/DepartmentSchedule "+
        "where $DepartmentSchedule/Department_OLK_ID=11605 and $DepartmentSchedule/Shop_OLK_ID=11441 " +
        "and $DepartmentSchedule/ScheduleVersion=1 and $DepartmentSchedule/Period/PRD_ID=2793  " +
        "return $DepartmentSchedule";
    var httpWebRequest = (HttpWebRequest)WebRequest.Create("http://localhost:8984/rest?query=" + xquery);
    httpWebRequest.Method = WebRequestMethods.Http.Get;
    string _auth = string.Format("{0}:{1}", "admin", "admin");
    string _enc = Convert.ToBase64String(Encoding.UTF8.GetBytes(_auth));
    string cred = string.Format("{0} {1}", "Basic", _enc);
    httpWebRequest.Headers[HttpRequestHeader.Authorization] = cred;
    httpWebRequest.Headers[HttpRequestHeader.AcceptEncoding] = "gzip.deflate";
    httpWebRequest.Headers[HttpRequestHeader.CacheControl] = "no-cache";
    XmlDocument doc = new XmlDocument();
    var response = httpWebRequest.GetResponse();
    XmlSerializer serializer = new XmlSerializer(typeof(DepartmentSchedule));
    return (DepartmentSchedule) serializer.Deserialize(response.GetResponseStream());
}
```

Het verwijderen van een document:

```
public void DeleteSingleDepartmentScheduleByAttribute(DepartmentSchedule d)
{
    string xquery = "for $DepartmentSchedule in db:open('BaseXFiles')/DepartmentSchedules/DepartmentSchedule "+
        "where $DepartmentSchedule/Department_OLK_ID=" + d.Department_OLK_ID +
        " and $DepartmentSchedule/Shop_OLK_ID=" + d.Shop_OLK_ID +
        " and $DepartmentSchedule/ScheduleVersion=" + d.ScheduleVersion +
        " for $period in $DepartmentSchedule/Period/PRD_ID="+ d.Period.PRD_ID +
        " return delete node $DepartmentSchedule";
    var httpWebRequest = (HttpWebRequest)WebRequest.Create("http://localhost:8984/rest?query=" + xquery);
    httpWebRequest.Method = WebRequestMethods.Http.Get;
    string _auth = string.Format("{0}:{1}", "admin", "admin");
    string _enc = Convert.ToBase64String(Encoding.UTF8.GetBytes(_auth));
    string cred = string.Format("{0} {1}", "Basic", _enc);
    httpWebRequest.Headers[HttpRequestHeader.Authorization] = cred;
    httpWebRequest.Headers[HttpRequestHeader.AcceptEncoding] = "gzip.deflate";
    httpWebRequest.Headers[HttpRequestHeader.CacheControl] = "no-cache";
    var response = httpWebRequest.GetResponse();
    response.Close();
}
```

Het opslaan van een document:

```
public void SaveObject(IIdentifiable o)
{
    var stringwriter = new System.IO.StringWriter();
    XmlSerializer serializer = new XmlSerializer(typeof(DepartmentSchedule));
    serializer.Serialize(stringwriter, o);
    string xml = stringwriter.ToString();
    xml = xml.Remove(0, 41);
    string xquery = "let $up := " + xml +
    " return "+
    "insert node $up as last into db:open('BaseXFiles')/DepartmentSchedules";
    var httpWebRequest = (HttpWebRequest)WebRequest.Create("http://localhost:8984/rest?query="+xquery);
    httpWebRequest.Method = WebRequestMethods.Http.Get;
    string _auth = string.Format("{0}:{1}", "admin", "admin");
    string _enc = Convert.ToBase64String(Encoding.UTF8.GetBytes(_auth));
    string cred = string.Format("{0} {1}", "Basic", _enc);
    httpWebRequest.Headers[HttpRequestHeader.Authorization] = cred;
    httpWebRequest.Headers[HttpRequestHeader.AcceptEncoding] = "gzip.deflate";
    httpWebRequest.Headers[HttpRequestHeader.CacheControl] = "no-cache";
    var response = httpWebRequest.GetResponse();
    response.Close();
}
```

Bulk

Voor de bulk is dezelfde applicatie gebruikt. Hiervoor zag de code er als volgt uit:

De main methode van program:

```
DateTime start = DateTime.Now;
BaseXDBRestCon con = new BaseXDBRestCon();
List<DepartmentSchedule> d = con.GetDepartmentsByAttributes();
Console.WriteLine("get success");
con.DeleteMultipleDepartmentSchedulesByAttribute(d);
Console.WriteLine("Delete success");
con.SaveMultipleDepartmentSchedules(d);
Console.WriteLine("Save success");
DateTime end = DateTime.Now;
TimeSpan diff = end - start;
Console.Write("totaltime: " + diff);
Console.ReadKey();
```

Het ophalen van alle departments was als volgt:

```
public List<DepartmentSchedule> GetDepartmentsByAttributes()
{
    string xquery = "let $result := for $DepartmentSchedule in db:open('BaseXFiles')/DepartmentSchedules/DepartmentSchedule " +
        "where $DepartmentSchedule/Shop_OLK_ID=11442 and $DepartmentSchedule/ScheduleVersion=1 " +
        "and $DepartmentSchedule/Period/PRD_ID=2794 " +
        "return $DepartmentSchedule " +
        "return <DepartmentSchedules> {$result} </DepartmentSchedules>";
    var httpWebRequest = (HttpWebRequest)WebRequest.Create("http://localhost:8984/rest?query=" + xquery);
    httpWebRequest.Method = WebRequestMethods.Http.Get;
    string _auth = string.Format("{0}:{1}", "admin", "admin");
    string _enc = Convert.ToBase64String(Encoding.UTF8.GetBytes(_auth));
    string cred = string.Format("{0} {1}", "Basic", _enc);
    httpWebRequest.Headers[HttpRequestHeader.Authorization] = cred;
    httpWebRequest.Headers[HttpRequestHeader.AcceptEncoding] = "gzip.deflate";
    httpWebRequest.Headers[HttpRequestHeader.CacheControl] = "no-cache";
    var response = httpWebRequest.GetResponse();
    OrientResult result = new OrientResult();
    XmlSerializer serializer = new XmlSerializer(typeof(OrientResult));
    result = (OrientResult)serializer.Deserialize(response.GetResponseStream());
    return result.result;
```

Het opslaan van alle departments ging als volgt:

```
public void SaveMultipleDepartmentSchedules(List<DepartmentSchedule> depts)
{
    foreach (DepartmentSchedule dept in depts)
    {
        this.SaveObject(dept);
    }
}
```

De SaveObject methode is hetzelfde als bij de insert query.

Het verwijderen van alle departments ging als volgt:

```
public void DeleteMultipleDepartmentSchedulesByAttribute(List<DepartmentSchedule> depts)
{
    foreach (DepartmentSchedule d in depts)
    {
        this.DeleteSingleDepartmentScheduleByAttribute(d);
    }
}
```

Ook de DeleteSingleDepartmentScheduleByAttribute methode was hetzelfde als bij de insert.

Uitvoering:

Select

Het uitvoeren van de select query resulteerde in de volgende tijden:

Result:
- Hit(s): 1 Item
- Updated: 0 Items
- Printed: 1994 Bytes
- Read Locking: local [BaseXFiles]
- Write Locking: none

Timing:
- Parsing: 0.34 ms
- Compiling: 0.4 ms
- Evaluating: 158.93 ms
- Printing: 0.2 ms
- Total Time: 159.87 ms

BaseX is met 159.87ms dus sneller met het ophalen van de afdelingsroosters dan SQL-Server.

Insert
Het uitvoeren van de insert leverde de volgende resultaten op:

```
Get success
delete success
Save success
totaltime: 00:00:00.7426649
```

Ook hiermee is BaseX sneller dan SQL-server.

Bulk
Het uitvoeren van de bulk leverde de volgende resultaten op:

```
get success
Delete success
Save success
totaltime: 00:00:01.1972720
```

Ook hiermee is BaseX sneller dan SQL-server.

I.2.2 OrientDB

Hieronder wordt beschreven hoe de tests zijn uitgevoerd in OrientDB en wat de resultaten hiervan waren.

Aanpak:
Select
Alle tabellen van de SQL-server database en de nieuwe collection zijn overgezet naar JSON en vervolgens weggeschreven naar OrientDB. Vervolgens is de DepartmentSchedule opgevraagd aan de hand van de volgende waardes:

- Shop_OLK_ID
- Department_OLK_ID
- ScheduleVersion
- Period.PRD_ID

Dit is gedaan via de Studio met SQL als query taal.

De volgende query is gebruikt:

```
1 SELECT * FROM DepartmentSchedule WHERE Department_OLK_ID="11662" AND ScheduleVersion="0" AND Shop_OLK_ID="11661" AND Period.PRD_ID="2818"
```

Insert
De insert statement van OrientDB is via een console applicatie gedaan. Reden dat het op deze manier is gedaan, is dat RavenDB geen studio heeft waarin je update statements kan doen. Om de vergelijking van alle document databases gelijk te houden heb ik besloten ze allemaal met dezelfde applicatie te doen. De applicatie haalde 1 document op, op basis van de volgende waarde:

Attribuut	Waarde
Department_OLK_ID	11807
ScheduleVersion	0
Shop_OLK_ID	11781
Period.PRD_ID	3042

Vervolgens sloeg de applicatie alle qualifications(diensten) van de eerste medewerker uit het afdelingsrooster op in een variabele. Nadat deze diensten waren opgeslagen, verwijderde hij alle diensten van de medewerker en sloeg de afdelingsrooster weer op. Tot slot haalde hij de afdelingsrooster weer op, en voegde hij alle diensten toe.

De code die voor het ophalen van de afdelingsrooster was als volgt:

```
public DepartmentSchedule GetDepartmentByAttributes(){
    var httpWebRequest = (HttpWebRequest)WebRequest.Create("http://localhost:2480/query/rrweb_perf/sql/select from DepartmentSchedule where
    httpWebRequest.Method = WebRequestMethods.Http.Get;// "POST";
    httpWebRequest.ContentType = "application/json";
    string _auth = string.Format("{0}:{1}", "root", "naily1993");
    string _enc = Convert.ToBase64String(Encoding.UTF8.GetBytes(_auth));
    string cred = string.Format("{0} {1}", "Basic", _enc);
    httpWebRequest.Headers[HttpRequestHeader.Authorization] = cred;
    httpWebRequest.Headers[HttpRequestHeader.AcceptEncoding] = "gzip.deflate";
    httpWebRequest.Headers[HttpRequestHeader.CacheControl] = "no-cache";
    var response = httpWebRequest.GetResponse();
    var json = "";
    using (var r = new StreamReader(response.GetResponseStream()))
    {
        json = r.ReadToEnd();
    }
    json = json.Substring(11);
    json = json.Remove(json.Length - 2);
    json = json.Replace("@", "_");
    DepartmentSchedule d = new System.Web.Script.Serialization.JavaScriptSerializer().Deserialize<DepartmentSchedule>(json);
    return d;
}
```

De gebruikte query is voor een deel weggevallen, maar zag er als volgt uit: select from DepartmentSchedule where Period.PRD_ID=’ 3402’ and Department_OLK_ID=’11807’ and ScheduleVersion=’0’ and Shop_OLK_ID=’11781’.

De code voor het updaten van de afdelingsrooster was:

```
public void UpdateDepartment(DepartmentSchedule o)
{
    o._rid = o._rid.Remove(0, 1);
    var httpWebRequest = (HttpWebRequest)WebRequest.Create("http://localhost:2480/document/rrweb_perf/" + o._rid);
    httpWebRequest.Method = WebRequestMethods.Http.Put;// "POST";
    string _auth = string.Format("{0}:{1}", "root", "naily1993");
    string _enc = Convert.ToBase64String(Encoding.UTF8.GetBytes(_auth));
    string cred = string.Format("{0} {1}", "Basic", _enc);
    httpWebRequest.Headers[HttpRequestHeader.Authorization] = cred;
    httpWebRequest.Headers[HttpRequestHeader.AcceptEncoding] = "gzip.deflate";
    httpWebRequest.Headers[HttpRequestHeader.CacheControl] = "no-cache";
    using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))
    {
        string json = o.ToOrientDBJson();

        streamWriter.Write(json);
        streamWriter.Flush();
        streamWriter.Close();
    }
    httpWebRequest.ProtocolVersion = HttpVersion.Version10;
    var httpResponse = (HttpWebResponse)httpWebRequest.GetResponse();
    httpResponse.Close();
```

De code voor de program was:

```
DateTime start = DateTime.Now;
OrientDBRestCon ocon = new OrientDBRestCon();
DepartmentSchedule d = ocon.GetDepartmentByAttributes();
ServiceSchedule[] ss = d.Employments[0].Qualifications.ToArray();
int i = d.Employments.First().Qualifications.Count;
Console.WriteLine("first: "+ i );
d.Employments.First().Qualifications.Clear();
int y = d.Employments.First().Qualifications.Count;
Console.WriteLine("clear no-save " + y);
ocon.UpdateDepartment(d);
d = ocon.GetDepartmentByAttributes();
i = d.Employments.First().Qualifications.Count;
Console.WriteLine("cleared: " + i);
d.Employments.First().Qualifications.AddRange(ss);
ocon.UpdateDepartment(d);
d = ocon.GetDepartmentByAttributes();
Console.WriteLine("final: " + d.Employments[0].Qualifications.Count);
DateTime end = DateTime.Now;
TimeSpan diff = end - start;
Console.Write("totaltime: " + diff);
Console.ReadKey();
```

Bulk

Voor de bulk uitvoering is dezelfde applicatie gebruikt als voor de insert. De code van program zag er als volgt uit:

```
DateTime start = DateTime.Now;
OrientDBRestCon con = new OrientDBRestCon();
List<DepartmentSchedule> d = con.GetDepartmentsByAttributes();
List<TempDepartmentSchedule> temporaries = ClearShopDepartmentSchedules(d);
con.UpdateDepartments(d);
Console.WriteLine("Cleared all");
d = con.GetDepartmentsByAttributes();

FillShopDepartmentSchedules(temporaries, d);
con.UpdateDepartments(d);
Console.WriteLine("Filled all");

DateTime end = DateTime.Now;
TimeSpan diff = end - start;
Console.Write("totaltime: " + diff);
Console.ReadKey();
```

```
private static List<TempDepartmentSchedule> ClearShopDepartmentSchedules(List<DepartmentSchedule> schedules)
{
    List<TempDepartmentSchedule> returnList = new List<TempDepartmentSchedule>();
    foreach (DepartmentSchedule dept in schedules)
    {
        foreach(etl.NewModel.EmployeeSchedule emp in dept.Employments)
        {
            var temp = new TempDepartmentSchedule()
            {
                Department_OLK_ID = dept.Department_OLK_ID,
                EMT_ID = emp.EMT_ID,
                Qualifications = emp.Qualifications.ToArray()
            };

            returnList.Add(temp);
            emp.Qualifications.Clear();
        }
    }
    return returnList;
}
private static void FillShopDepartmentSchedules(List<TempDepartmentSchedule> temporaries, List<DepartmentSchedule> schedules)
{
    foreach (DepartmentSchedule dept in schedules)
    {
        foreach (TempDepartmentSchedule temp in temporaries)
        {
            if (temp.Department_OLK_ID == dept.Department_OLK_ID)
            {
                etl.NewModel.EmployeeSchedule emp = dept.Employments.Where(empl => empl.EMT_ID == temp.EMT_ID).First();
                emp.Qualifications.AddRange(temp.Qualifications);
            }
        }
    }
}
```

In de controller gebeurde het volgende:

```
public List<DepartmentSchedule> GetDepartmentsByAttributes()
{
    var httpWebRequest = (HttpWebRequest)WebRequest.Create("http://localhost:2480/query/rrweb_perf/sql/select from DepartmentSchedule
    httpWebRequest.Method = WebRequestMethods.Http.Get;
    httpWebRequest.ContentType = "application/json";
    string _auth = string.Format("{0}:{1}", "root", "naily1993");
    string _enc = Convert.ToBase64String(Encoding.UTF8.GetBytes(_auth));
    string cred = string.Format("{0} {1}", "Basic", _enc);
    httpWebRequest.Headers[HttpRequestHeader.Authorization] = cred;
    httpWebRequest.Headers[HttpRequestHeader.AcceptEncoding] = "gzip.deflate";
    httpWebRequest.Headers[HttpRequestHeader.CacheControl] = "no-cache";
    var response = httpWebRequest.GetResponse();
    var json = "";
    using (var r = new StreamReader(response.GetResponseStream()))
    {
        json = r.ReadToEnd();
    }
    json = json.Replace("@", "_");
    OrientResult result = new System.Web.Script.Serialization.JavaScriptSerializer().Deserialize<OrientResult>(json);
    return result.result;
}
```

De gebruikte query is voor een deel weggevallen, maar zag er als volgt uit: select from DepartmentSchedule where Period.PRD_ID=’ 3402’ and ScheduleVersion=’0’ and Shop_OLK_ID=’11781’.

```
public void UpdateDepartments(List<DepartmentSchedule> schedules)
{
    foreach (DepartmentSchedule dept in schedules)
    {
        this.UpdateDepartment(dept);
    }
}
```

Hierbij was UpdateDepartment de methode uit de insert statement.

Uitvoering:

Select

Dit resulteerde in de volgende tijd:

METADATA	PROPERTIES								
@version	resultSize	fullySortedByIndex	compositeIndexUsed	current	documentAnalyzedCompatibleClass	recordReads	documentReads	limit	fetchingFromTargetElapsed
0	1	false	1	#21:15483	2280	2280	2280	-1	591

10 25 50 100 1000 5000

Query executed in 0.599 sec. Returned 1 record(s). Limit: 20 (change it) Table Raw

OrientDB deed er 0,599 seconden over en is dus sneller dan SQL-Server.

Insert

De insert test leverde de volgende resultaten op:

```
first: 8
clear no-save 0
cleared 0
final: 8
totaltime: 00:00:05.7524260
```

Met 5,7524260 seconden is OrientDB dus sneller dan SQL-server.

Bulk

Dit resulteerde in het volgende:

```
Cleared all
Filled all
totaltime: 00:00:07.3543957
```

OrientDB deed er 7.3543957 seconden over om deze handeling uit te voeren en is dus sneller dan SQL-server.

I.2.3 RavenDB

Hieronder wordt beschreven hoe de tests zijn uitgevoerd in RavenDB en wat de resultaten hiervan waren.

Aanpak:

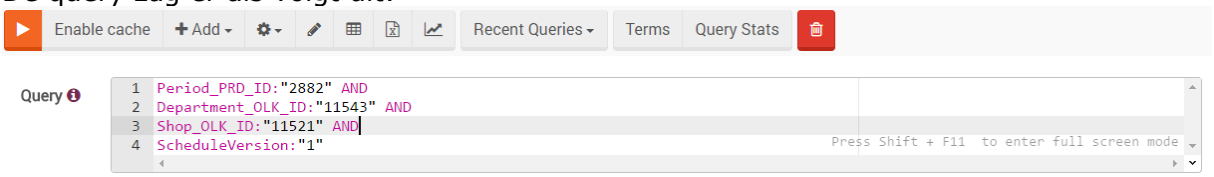
Select

Alle tabellen van de SQL-server database en de nieuwe collection zijn overgezet naar JSON en vervolgens weggeschreven naar OrientDB. Vervolgens is de DepartmentSchedule opgevraagd aan de hand van de volgende waardes:

- Shop_OLK_ID
- Department_OLK_ID
- ScheduleVersion
- Period.PRD_ID

Dit is gedaan via de Studio.

De query zag er als volgt uit:



Er zijn overigens indexen op alle betreffende attributen gezet, omdat er anders niet gefilterd kon worden.

Insert

De insert statement van RavenDB is via een console applicatie gedaan. Reden dat het op deze manier is gedaan, is dat RavenDB geen studio heeft waarin je update statements kan doen. De applicatie haalde 1 document op, op basis van de volgende waarde:

Attribuut	Waarde
Department_OLK_ID	11807
ScheduleVersion	0
Shop_OLK_ID	11781
Period.PRD_ID	3042

Vervolgens sloeg de applicatie alle qualifications(diensten) van de eerste medewerker uit het afdelingsrooster op in een variabele. Nadat deze diensten waren opgeslagen, verwijderde hij alle diensten van de medewerker en sloeg de afdelingsrooster weer op. Tot slot haalde hij de afdelingsrooster weer op, en voegde hij alle diensten toe.

De code die voor het ophalen en updaten van de afdelingsrooster was als volgt:

```
public void UpdateObject(IIdentifiable o)
{
    using (var session = OpenSession("RRweb_performance"))
    {
        session.Store(o, o.Id);
        session.SaveChanges();
    }
}

public DepartmentSchedule GetDepartmentScheduleByAttributes()
{
    using (Store.DatabaseCommands.DisableAllCaching())
    {
        using (var session = OpenSession("RRweb_performance"))
        {
            var d = session.Query<DepartmentSchedule>("GetDepartmentScheduleByPeriodDepartmentShopAndVersion");
            return d.Where(ds => ds.Department_OLK_ID == "11807" && ds.ScheduleVersion == "0"
                && ds.Shop_OLK_ID == "11781" && ds.Period.PRD_ID == "3042").Single();
        }
    }
}
```

De code uit de program was:

```
DateTime start = DateTime.Now;
RavenDBCon con = new RavenDBCon();
DepartmentSchedule d = con.GetDepartmentScheduleByAttributes();
ServiceSchedule[] ss = d.Employments[0].Qualifications.ToArray();
int i = d.Employments.First().Qualifications.Count;
Console.WriteLine("first: " + i);
d.Employments.First().Qualifications.Clear();
int y = d.Employments.First().Qualifications.Count;
Console.WriteLine("clear no-save " + y);
con.UpdateObject(d);
d = con.GetDepartmentScheduleByAttributes();
i = d.Employments.First().Qualifications.Count;
Console.WriteLine("cleared " + i);
d.Employments.First().Qualifications.AddRange(ss);
con.UpdateObject(d);
d = con.GetDepartmentScheduleByAttributes();
Console.WriteLine("final:" + d.Employments[0].Qualifications.Count());
DateTime end = DateTime.Now;
TimeSpan diff = end - start;
Console.Write("totaltime: " + diff);
Console.ReadKey();
```

Bulk

Voor de bulk heb ik dezelfde applicatie gebruikt als bij de insert. De code van de program class zag er als volgt uit:

```
DateTime start = DateTime.Now;
RavenDBCon con = new RavenDBCon();
List<DepartmentSchedule> d = con.GetDepartmentSchedulesByAttributes();
List<TempDepartmentSchedule> temporaries = ClearShopDepartmentSchedules(d);
con.UpdateDepartments(d);
Console.WriteLine("Cleared all");
d = con.GetDepartmentSchedulesByAttributes();

FillShopDepartmentSchedules(temporaries, d);
con.UpdateDepartments(d);
Console.WriteLine("Filled all");

DateTime end = DateTime.Now;
TimeSpan diff = end - start;
Console.Write("totaltime: " + diff);
Console.ReadKey();
```

Er worden 2 methodes aangeroepen in dit stuk, namelijk: ClearShopDepartmentSchedules en FillShopDepartmentSchedules. Deze zagen er als volgt uit:

```
private static List<TempDepartmentSchedule> ClearShopDepartmentSchedules(List<DepartmentSchedule> schedules)
{
    List<TempDepartmentSchedule> returnList = new List<TempDepartmentSchedule>();
    foreach (DepartmentSchedule dept in schedules)
    {
        foreach(etl.NewModel.EmployeeSchedule emp in dept.Employments)
        {
            var temp = new TempDepartmentSchedule()
            {
                Department_OLK_ID = dept.Department_OLK_ID,
                EMT_ID = emp.EMT_ID,
                Qualifications = emp.Qualifications.ToArray()
            };

            returnList.Add(temp);
            emp.Qualifications.Clear();
        }
    }
    return returnList;
}
```

```
private static void FillShopDepartmentSchedules(List<TempDepartmentSchedule> temporaries, List<DepartmentSchedule> schedules)
{
    foreach (DepartmentSchedule dept in schedules)
    {
        foreach (TempDepartmentSchedule temp in temporaries)
        {
            if (temp.Department_OLK_ID == dept.Department_OLK_ID)
            {
                etl.NewModel.EmployeeSchedule emp = dept.Employments.Where(empl => empl.EMT_ID == temp.EMT_ID).First();
                emp.Qualifications.AddRange(temp.Qualifications);
            }
        }
    }
}
```

Verder is de volgende Get methode gebruikt uit de databasecontroller:

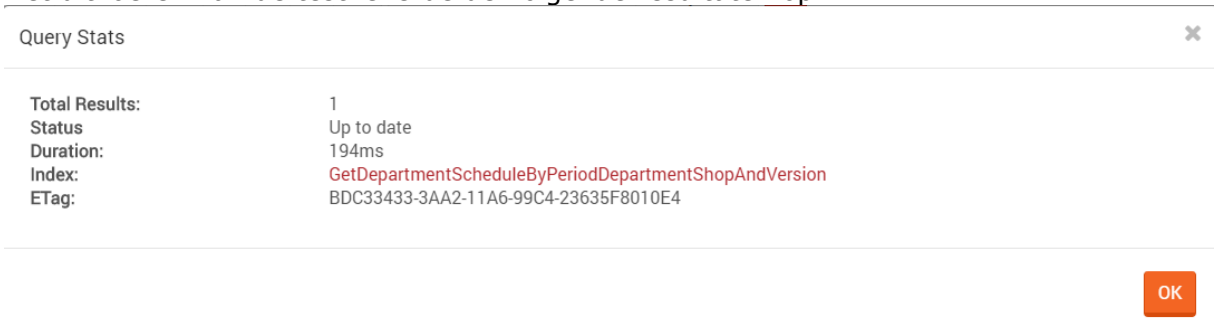
```
public List<DepartmentSchedule> GetDepartmentSchedulesByAttributes()
{
    using (Store.DatabaseCommands.DisableAllCaching())
    {
        using (var session = OpenSession("RRweb_performance"))
        {
            var d = session.Query<DepartmentSchedule>("GetDepartmentScheduleByPeriodDepartmentShopAndVersion");
            return d.Where(ds => ds.ScheduleVersion == "0"
                && ds.Shop_OLK_ID == "11781" && ds.Period.PRD_ID == "3042").ToList();
        }
    }
}
```

En de volgende Update methode:

```
public void UpdateDepartments(List<DepartmentSchedule> o)
{
    using (var session = OpenSession("RRweb_performance"))
    {
        foreach (DepartmentSchedule io in o)
        {
            session.Store(io, io.Id);
            session.SaveChanges();
        }
    }
}
```

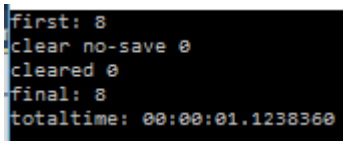
Uitvoering:

Select
Het uitvoeren van de test leverde de volgende resultaten op:



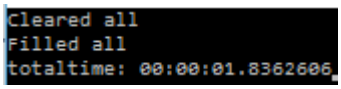
RavenDB is met 194 ms sneller dan SQL-Server, en heeft dus een punt verdient.

Insert
De resultaten die de insert test opleverde waren als volgt:



Het toevoegen van alle diensten van 1 medewerker op een afdeling heeft dus 1.1238360 seconden geduurd en is daarmee sneller dan SQL-server.

Bulk
De bulk uitvoering gaf de volgende resultaten:



Met 1.8362606 seconden is RavenDB ook hiermee sneller dan SQL-Server.

Bijlage D

Functioneel ontwerp

Xavyr Rademaker

Inhoudsopgave

1. Inleiding

Dit document bevat het functionele ontwerp van de proof of concept van R&R-web. De functionaliteiten worden per sprint behandeld en laten zien wat er nodig is om de functionaliteit te realiseren.

De functionaliteiten worden op verschillende manieren beschreven, namelijk:

- Mock ups
    - Tonen hoe het resultaat eruit moet komen te zien
- Klassendiagrammen
    - Welke entiteiten zijn er nodig om deze functionaliteit te realiseren?
    - Welke andere klassen zijn er nodig om deze functionaliteit te realiseren?
        - Boundary: hier communiceert de gebruiker mee
        - Control: deze regelt alles onderwater
- Sequence diagrammen
    - Hoe communiceren de boundary en de controls met elkaar om de entiteiten op te halen/te wijzigen

2. Sprint 1

Dit hoofdstuk bespreekt het functioneel ontwerp van sprint 1. Eerst worden de backlog items benoemd, vervolgens wordt per backlog item een conceptueel diagram getoond, met daarbij een conceptuele sequentie diagram. Deze diagrammen samen tonen aan hoe de backlog item gerealiseerd wordt.

De backlog items zijn in het Nederlands opgesteld, omdat het gesprek waar de backlog items naar voren zijn gekomen in het Nederlands werd gevoerd. De diagrammen zijn allemaal in het Engels opgesteld.

2.1 Sprint Backlog

De volgende backlog items zijn in deze sprint gerealiseerd:

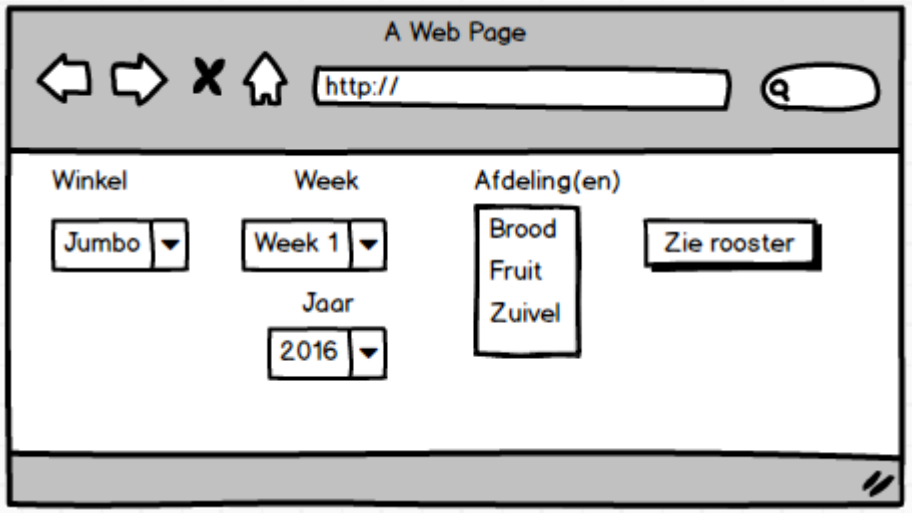
User story	Module	Story points
Als een afdelingsmanager wil ik diensten kunnen inzien die ik eerder ingevuld heb, zodat de medewerkers weten wanneer ze moeten werken en ik weet dat ik genoeg mensen heb die mijn werk kunnen doen.	Ro	8
Als een afdelingsmanager wil ik een afdelingsrooster kunnen maken zodat ik de benodigde mensen op de juiste tijd kan inplannen	Ro	6
Als een afdelingsmanager wil ik medewerkers van andere winkels kunnen inplannen op het moment dat ik zelf medewerkers te kort kom zodat ik genoeg medewerkers kan inplannen	Ro	10
Als afdelingsmanager wil ik geïnformeerd worden van bijzondere dagen zodat ik daar rekening mee kan houden bij het inplannen	Ro/P	4

2.2 Zien afdelingsrooster

Het kunnen zien van het rooster ging erom dat de afdelingsmanager kan zien welke medewerkers, op welk moment werkte. Hierbij moesten er eerst zoekcriteria opgegeven worden(winkel, week, jaar en afdeling(en). Vervolgens werd het rooster van de geselecteerde afdeling(en) van de geselecteerde winkel, binnen een periode(Week en jaar) getoond.

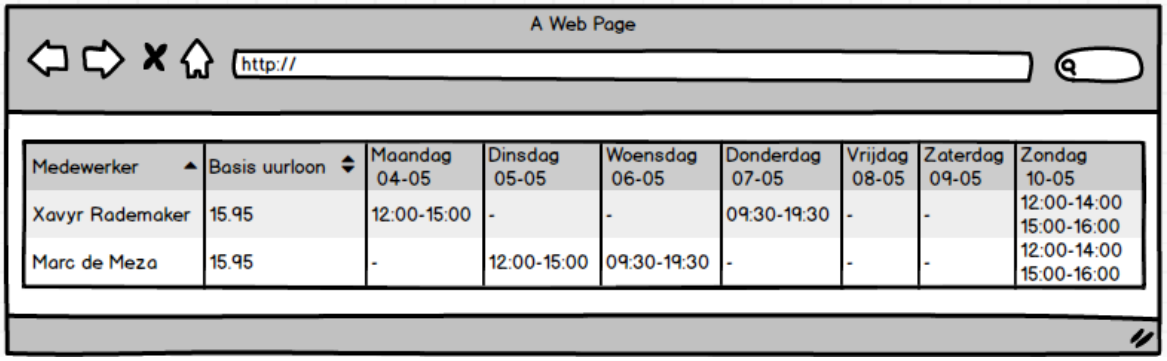
2.2.1 Mock up(s)

Het zoeken van het rooster gaat als volgt:



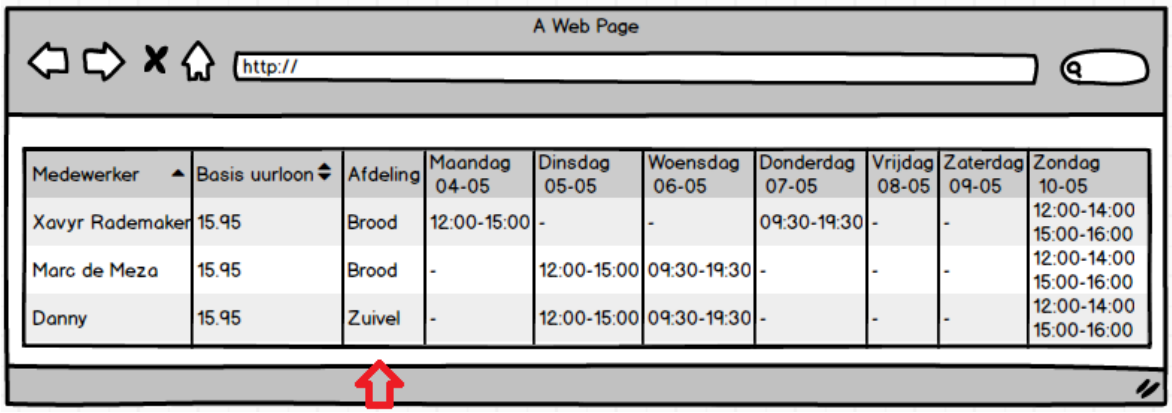
Let erop dat er meerdere afdelingen geselecteerd kunnen worden en dat de afdelingen moeten horen bij de geselecteerde winkel, en moeten bestaan binnen de geselecteerde periode.

Als alle criteria zijn ingevuld, en er wordt gekozen voor zie rooster dan krijgt de gebruiker het volgende beeld te zien.



Medewerker	Basis uurloon	Maandag 04-05	Dinsdag 05-05	Woensdag 06-05	Donderdag 07-05	Vrijdag 08-05	Zaterdag 09-05	Zondag 10-05
Xavyr Rademaker	15.95	12:00-15:00	-	-	09:30-19:30	-	-	12:00-14:00 15:00-16:00
Marc de Meza	15.95	-	12:00-15:00	09:30-19:30	-	-	-	12:00-14:00 15:00-16:00

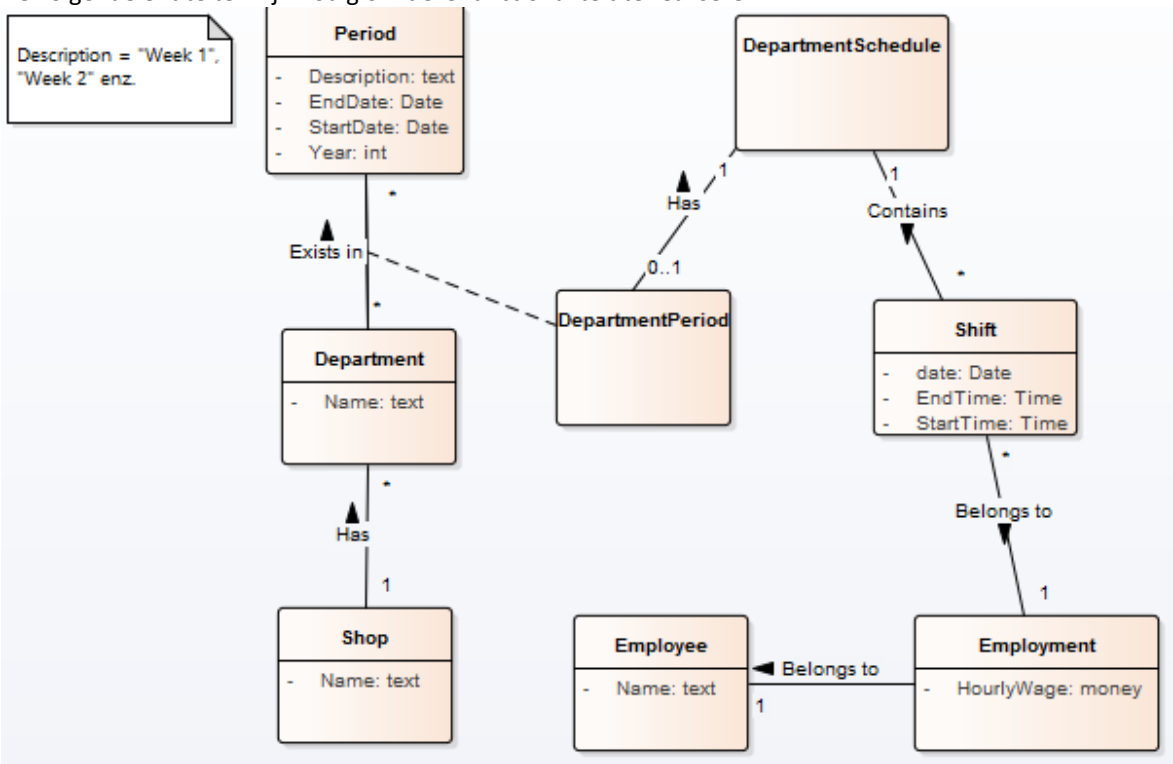
Op het moment dat er meerdere afdelingen geselecteerd zijn, komt er een extra kolom bij in de tabel. Deze kolom geeft aan over welke afdeling het gaat. Dit is te zien in de volgende afbeelding.



Medewerker	Basis uurloon	Afdeling	Maandag 04-05	Dinsdag 05-05	Woensdag 06-05	Donderdag 07-05	Vrijdag 08-05	Zaterdag 09-05	Zondag 10-05
Xavyr Rademaker	15.95	Brood	12:00-15:00	-	-	09:30-19:30	-	-	12:00-14:00 15:00-16:00
Marc de Meza	15.95	Brood	-	12:00-15:00	09:30-19:30	-	-	-	12:00-14:00 15:00-16:00
Danny	15.95	Zuivel	-	12:00-15:00	09:30-19:30	-	-	-	12:00-14:00 15:00-16:00

2.2.2 Entiteiten

De volgende entiteiten zijn nodig om deze functionaliteit te realiseren.



Period: Een periode is de combinatie van week(description) en jaar. Deze is nodig om te achterhalen over welke periode een afdelingsrooster gaat, en om te weten in welke periode welke afdelingen bestaan. Verder bevat de periode ook de start en eind week van de periode(als het een week betreft dan dus de datum van Maandag en de datum van Zondag).

Department: Een afdeling van een winkel. Dit kan bijvoorbeeld zijn: Brood, Zuivel enz.

Shop: Een filiaal van een winkel. Dit kan bijvoorbeeld zijn Jumbo Leyweg, Jumbo de Stede enz.

Departmentperiod: De combinatie van afdeling en periode. Op het moment dat deze combinatie bestaat, betekent het dat die afdeling bestaat in deze periode.

DepartmentSchedule: Het rooster van een afdeling van een winkel in een bepaalde periode(week 1 2016 bijvoorbeeld). Deze bevat van alle ingeroosterde (dienstverbanden van) medewerkers hun diensten.

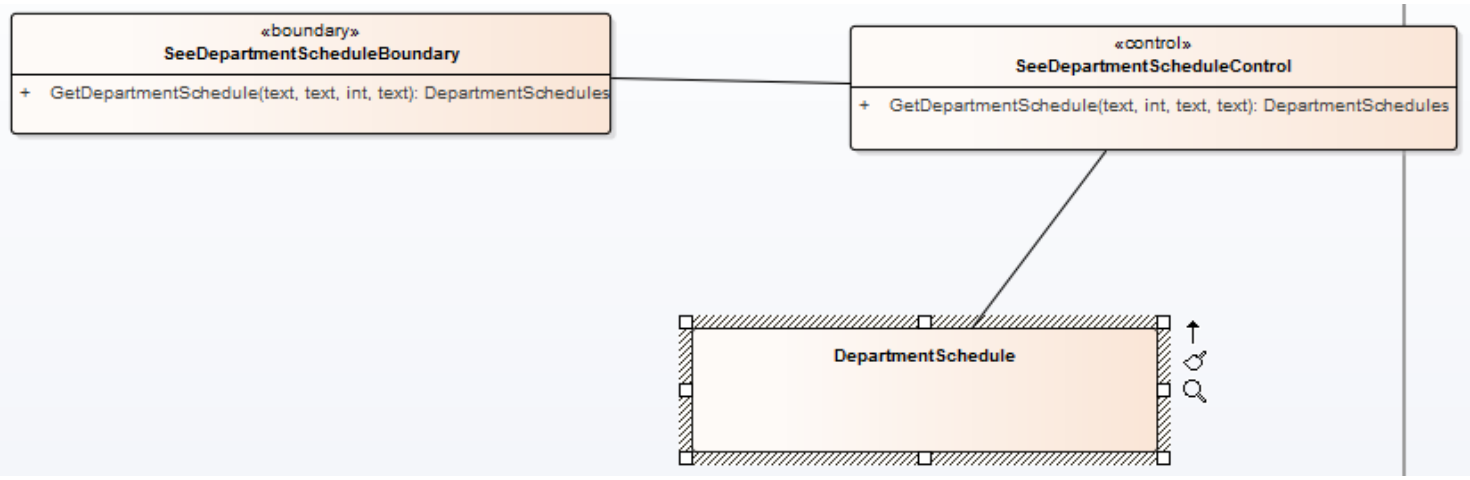
Shift: Een dienst van (een dienstverband van) een medewerker op een afdelingsrooster.

Employment: Een dienstverband van een medewerker(zijn o.a. zijn contractgegevens).

Employee: Een medewerker, bijvoorbeeld “Xavyr Rademaker”.

2.2.3 Control en boundary

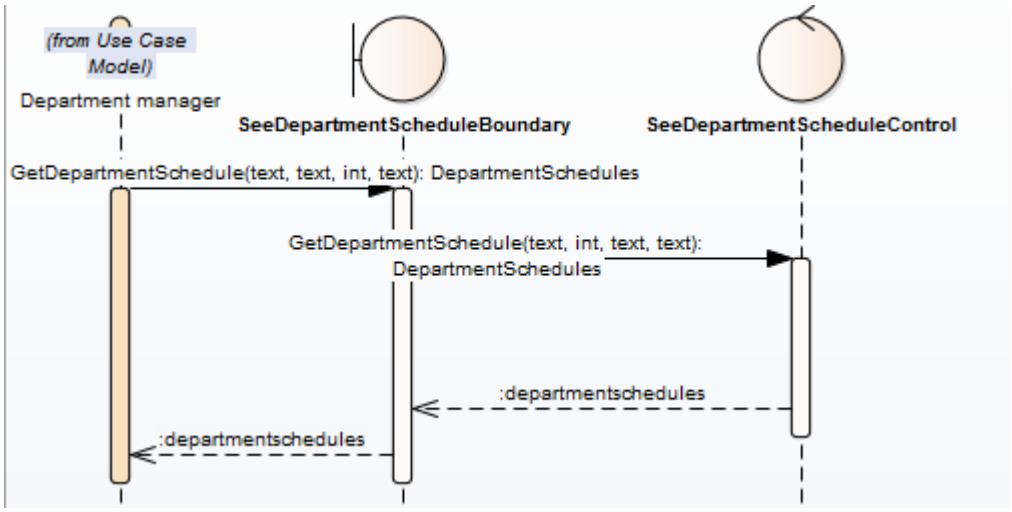
Om deze functionaliteit te realiseren is een boundary voldoende. Deze zit ziet er als volgt uit:



De boundary stuur de vraag door naar de control. De control haalt de daadwerkelijke afdelingsroosters op, op basis van de opgevraagde criteria.

2.2.4 Sequentie diagram

De in 2.2.3 beschreven flow is te zien in het volgende diagram:



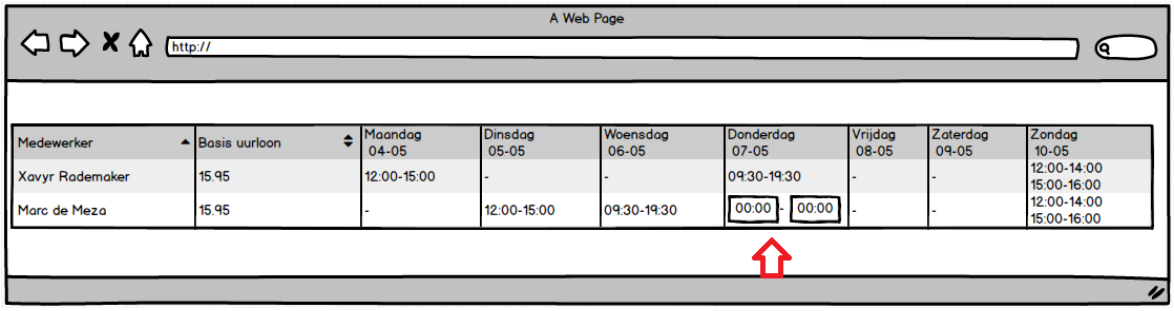
Let erop dat alle entiteiten meekomen vanuit het afdelingsrooster.

2.3 Maken/wijzigen diensten

Het maken/wijzigen van diensten is simpelweg het aanpassen/toevoegen van een dienst van een (dienstverband van een) medewerker op een afdelingsrooster. Hiervoor moest het scherm uit 2.2 uitgebreid worden met een input mogelijkheid.

2.3.1 Mock up(s)

Het scherm gaat er als volgt uitzien:



Zoals in de afbeelding te zien, kunnen diensten in de tabel aangepast worden. Hierbij kunnen de start en eindtijd aangepast worden. Het toevoegen van een nieuwe dienst ziet er hetzelfde uit.

2.3.2 Constraints

Het is belangrijk dat de starttijd van een dienst kleiner moet zijn dan de eindtijd van een dienst. Anders mag deze niet opgeslagen worden.

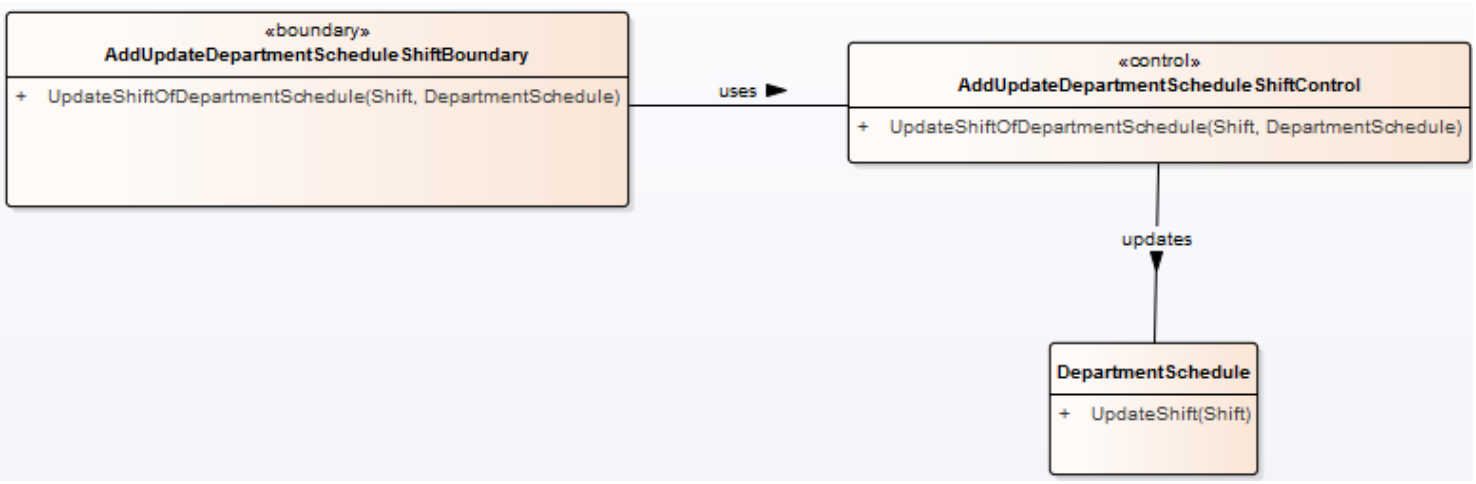
2.3.3 Entiteiten

Om deze functionaliteit te realiseren zijn geen nieuwe entiteiten nodig. Het gaat om dezelfde entiteiten als in 2.2: zien afdelingsrooster.

2.3.4 Control en boundary

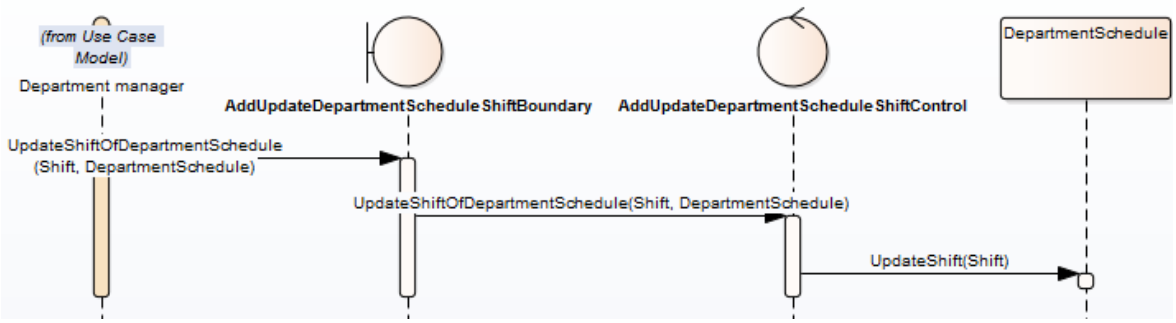
Daar de boundary alleen communiceert met de gebruiker, en er hier sprake is van een handeling in het systeem moet er voor deze backlog item wel een control gemaakt worden. De control voert de handelingen op het afdelingsrooster uit(het wijzigen/toevoegen van een dienst).

Dit ziet er als volgt uit:



2.3.5 Sequentie diagram(men)

De flow ziet er als volgt uit:



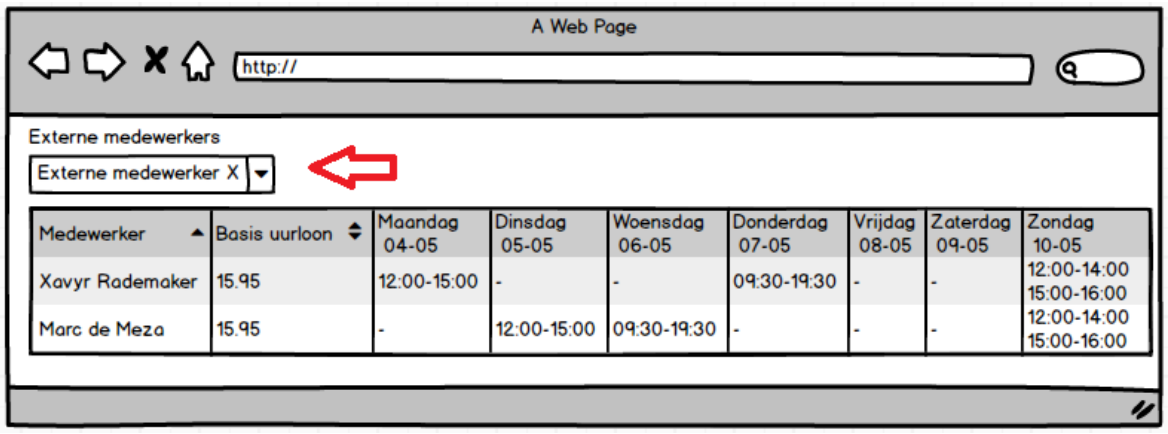
2.4 Medewerkers inhuren

Het inhuren van een medewerker houdt in dat er een medewerker van een andere afdeling/winkel ingehuurd wordt om een dienst uit te voeren in het geopende afdelingsrooster. Een voorbeeld hierbij is:
Medewerker A werkt bij Jumbo Leyweg afdeling brood.
Medewerker B werkt bij Jumbo Leyweg afdeling zuivel.

De afdelingsmanager van afdeling brood roostert Medewerker B in op zijn afdeling.

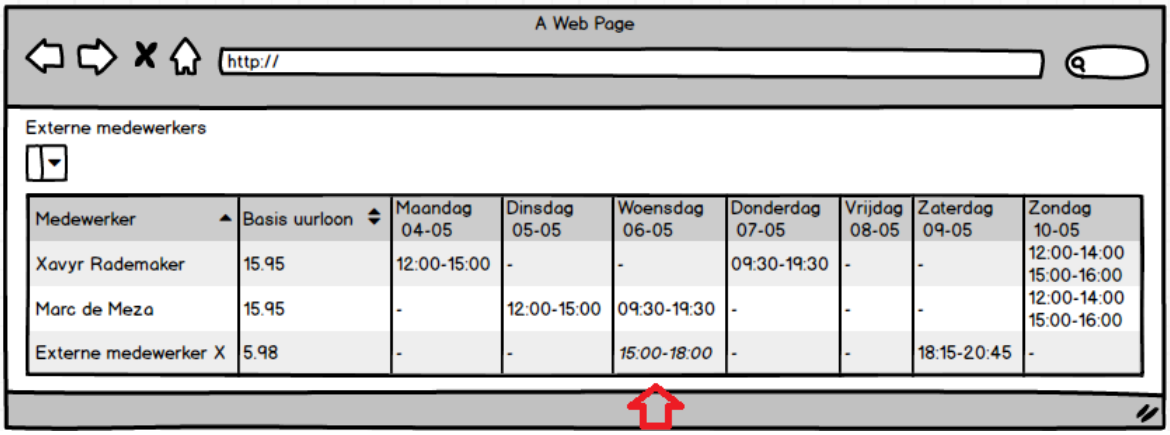
2.4.1 Mock up(s)

Dit zal er als volgt uitzien:



Medewerker	Basis uurloon	Maandag 04-05	Dinsdag 05-05	Woensdag 06-05	Donderdag 07-05	Vrijdag 08-05	Zaterdag 09-05	Zondag 10-05
Xavyr Rademaker	15.95	12:00-15:00	-	-	09:30-19:30	-	-	12:00-14:00 15:00-16:00
Marc de Meza	15.95	-	12:00-15:00	09:30-19:30	-	-	-	12:00-14:00 15:00-16:00

Op het moment dat een medewerker geselecteerd wordt verschijnt deze in het rooster. Hierbij worden alle diensten die die medewerker op andere afdelingen heeft ook meegenomen. Een dienst die een medewerker bij een andere afdeling heeft wordt schuingedrukt weergeven. Dit is te zien in de onderstaande afbeelding.



Medewerker	Basis uurloon	Maandag 04-05	Dinsdag 05-05	Woensdag 06-05	Donderdag 07-05	Vrijdag 08-05	Zaterdag 09-05	Zondag 10-05
Xavyr Rademaker	15.95	12:00-15:00	-	-	09:30-19:30	-	-	12:00-14:00 15:00-16:00
Marc de Meza	15.95	-	12:00-15:00	09:30-19:30	-	-	-	12:00-14:00 15:00-16:00
Externe medewerker X	5.98	-	-	*15:00-18:00*	-	-	18:15-20:45	-

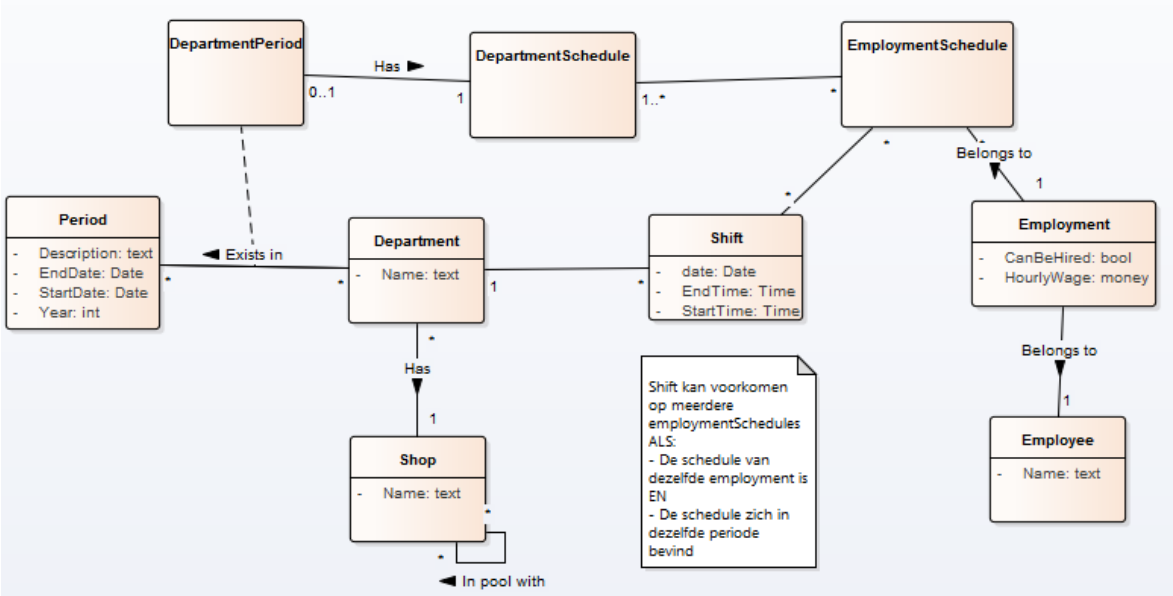
2.4.2 Constraints

Er zijn een aantal beperkingen op het inhuren van medewerkers, namelijk:

- De medewerker moet gehuurd kunnen worden
- De medewerker moet werken bij een andere afdeling van dezelfde filiaal als de filiaal waarvan het afdelingsrooster open is OF
- De medewerker moet werken bij een filiaal wat in dezelfde “Pool” zit als het filiaal waarvan het rooster open is.

2.4.3 Entiteiten

Hoewel er voor deze functionaliteit geen nieuwe entiteiten nodig zijn, is het wel nodig een aantal relaties tussen entiteiten aan te passen/toe te voegen. De entiteiten zullen er als volgt uitzien:



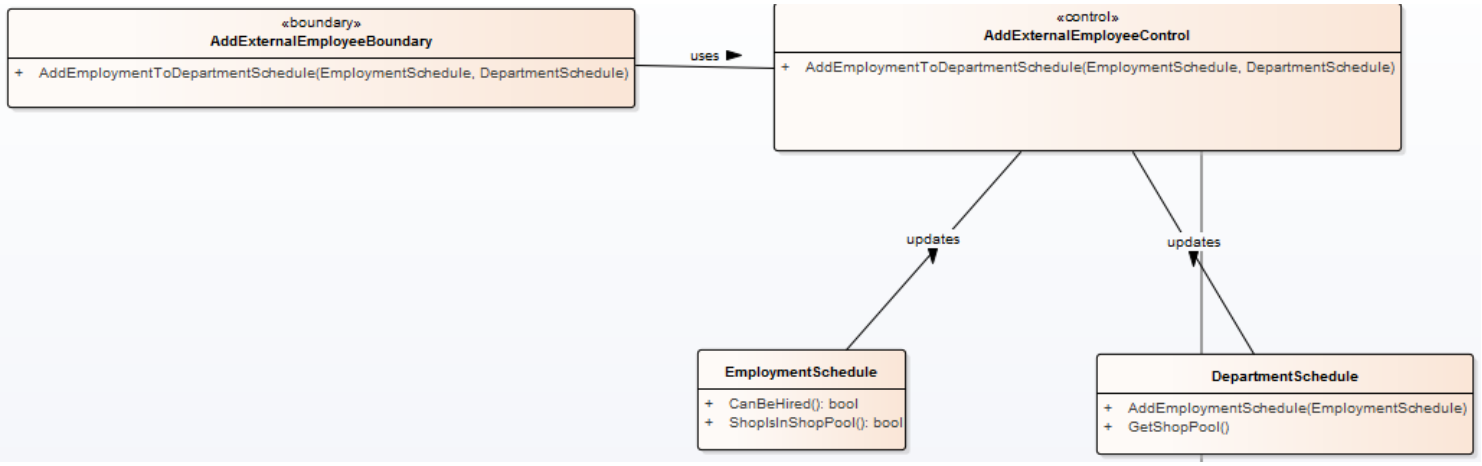
De volgende punten zijn veranderd ten opzichte van 2.2: Zien afdelingsrooster:

- Een employment heeft een attribuut: CanBeHired erbij gekregen. Deze is nodig om aan te geven of een (dienstverband van een) medewerker ingehuurd kan worden.
- Shop heeft een relatie met zichzelf, deze is nodig om aan te geven welke filialen met elkaar in een pool zitten.
- De combinatie van een (dienstverband van een) medewerker en shifts kunnen op meerdere afdelingsroosters voorkomen(als een medewerker wordt ingehuurd, worden dezelfde diensten op zowel de originele als de ingehuurde rooster getoond). Deze combinatie wordt **EmploymentSchedule** genoemd. (het rooster van een medewerker)
- Een medewerker kan ook ingehuurd worden zonder dat hij diensten heeft, dat betekent dat employmentschedule bijhoudt bij welke dienstverband hij hoort, maar geen diensten bevat.
- Een shift weet bij welke afdeling de dienst is. Dit is nodig om te weten of de dienst schuingedrukt moet worden of niet.

Let erop dat je dus als het ware een EmploymentSchedule toevoegd aan het afdelingsrooster.

2.4.4 Control en boundary

Voor het toevoegen van een (dienstverband van een) medewerker aan een afdelingsrooster zijn zowel een control als een boundary nodig. De boundary om met de gebruiker te communiceren, en een control om de handelingen op het rooster uit te voeren(toevoegen aan het rooster). Dit ziet er als volgt uit:



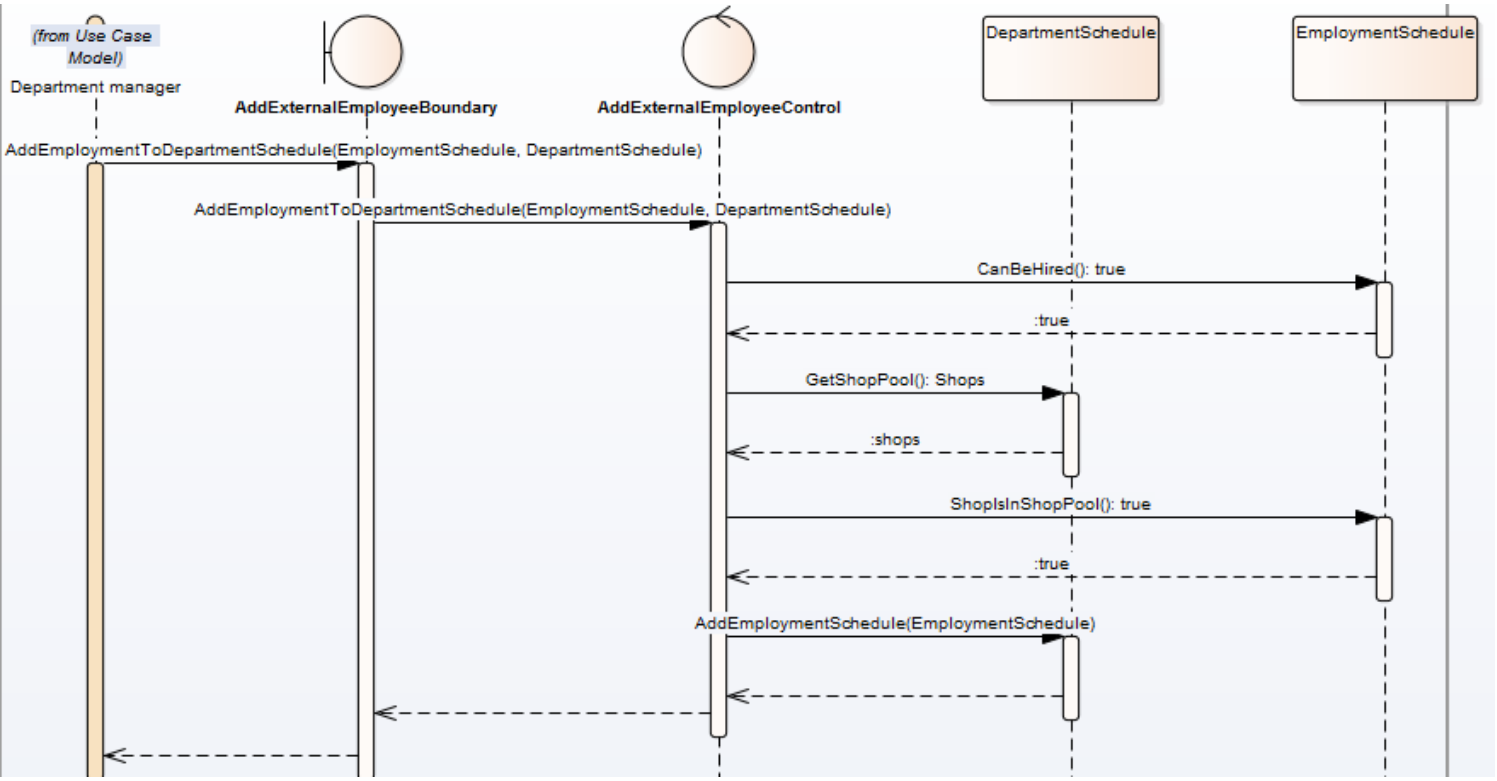
2.4.5 Sequentie diagram(men)

Hierbij horen 3 flows, namelijk:

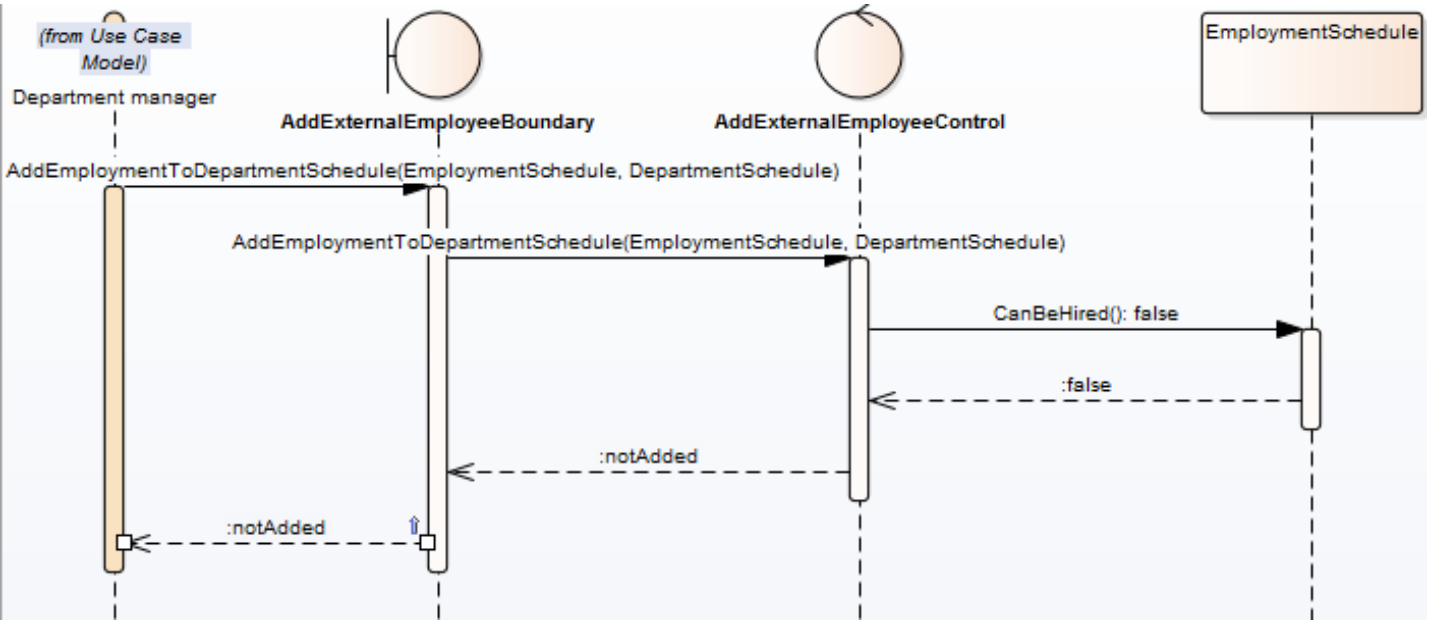
- Alles voldoet aan de constraints en de (dienstverband van de) medewerker mag worden toegevoegd aan het rooster
- De medewerker mag niet ingehuurd worden(CanBeHired is false), dus er wordt niks toegevoegd aan het rooster
- De medewerker werkt voor een winkel die niet in de pool zit, dus er wordt niks toegevoegd aan het rooster.

Deze flows gaan als volgt:

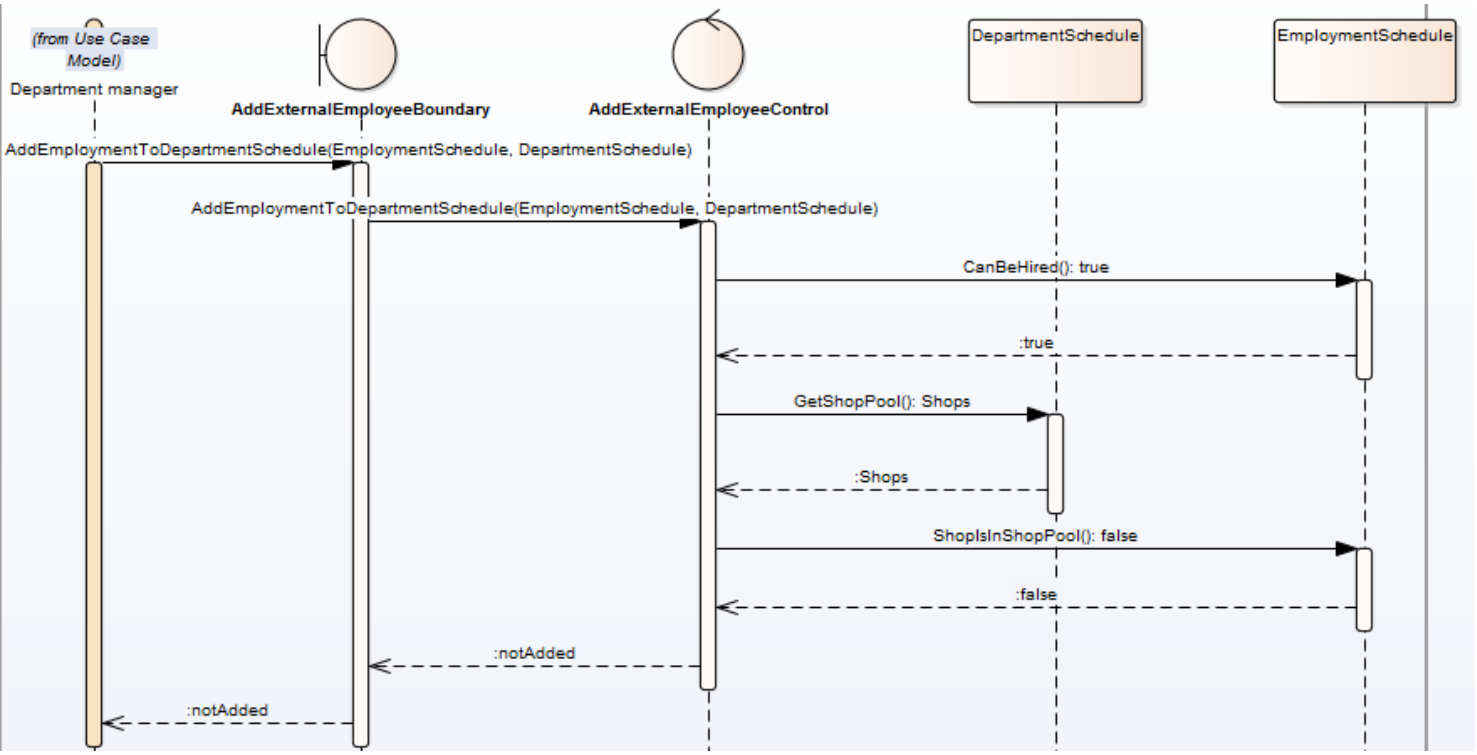
Alles voldoet aan de constraints en de (dienstverband van de) medewerker mag worden toegevoegd aan het rooster



De medewerker mag niet ingehuurd worden(CanBeHired is false), dus er wordt niks toegevoegd aan het rooster



De medewerker werkt voor een winkel die niet in de pool zit, dus er wordt niks toegevoegd aan het rooster.



2.5 Zien bijzondere dagen

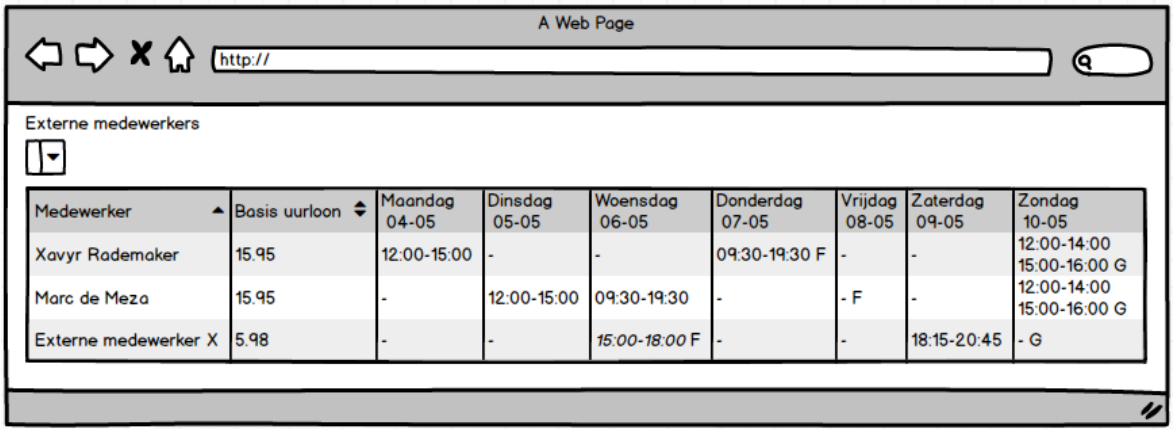
Het zien van bijzondere dagen houdt het volgende in:

- De gebruiker moet kunnen zien of er feestdagen in de week zijn
- De gebruiker moet kunnen zien of de afdeling gesloten is op dagen in de week

De openingstijden van een afdeling worden per week(Periode) bepaald. Het kan dus zijn dat een afdeling in week 1 op maandag open is van 10:00-18:00, en in week 2 op maandag gesloten is.

2.5.1 Mock up(s)

Dit zal er als volgt uit gaat zien:

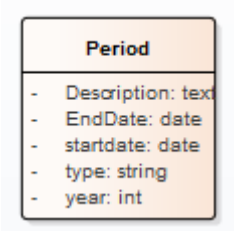


Hierbij zijn de F's: Feestdag
En de G's: gesloten

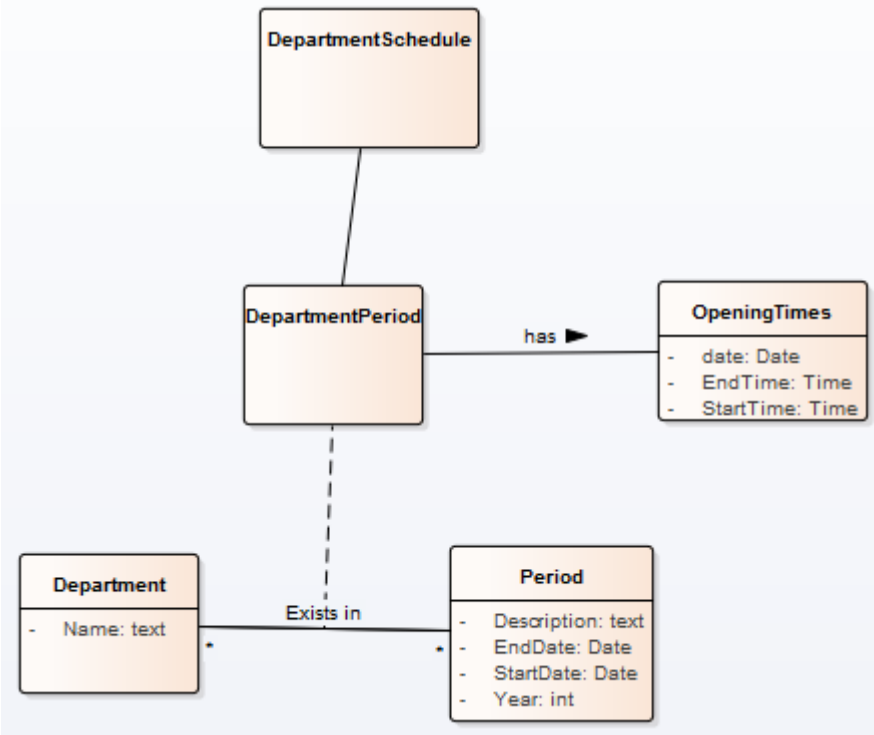
2.5.2 Entiteiten

Voor deze functionaliteit moesten er een aantal entiteiten toegevoegd/gewijzigd worden. De volgende diagrammen laten deze wijzigingen zien:

Zien feestdagen:
Voor het zien van feestdagen is alleen de entiteit period aangepast. Deze heeft een type erbij gekregen.



Zien gesloten dagen:
Aangezien de openingstijden van een afdeling per periode bepaald worden, zijn de openingstijden gekoppeld aan een afdelingsperiode(DepartmentPeriod).



2.5.3 Constraints

De type van Period kan zijn: Holiday of Week.

- Als het type week is, is description: Week x (x = weeknummer)
- als het type Holiday is, is description: Naam feestdag

2.5.4 Sequentie diagram(men)

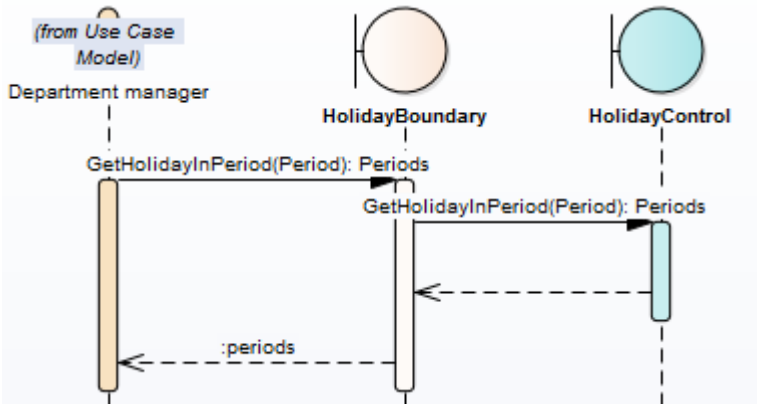
Het tonen van deze dagen verloopt in 2 stappen, namelijk:

1. Het ophalen van het rooster, hier worden de openingstijden gelijk meegenomen vanuit de departmentperiod die gekoppeld is aan het afdelingsrooster
2. Het ophalen van alle feestdagen binnen de periode uit de departmentperiod van het afdelingsrooster.

Dit ziet er als volgt uit:

1 ophalen van rooster + openingstijden
Dit gebeurd op dezelfde manier als het ophalen van het rooster(het is letterlijk dezelfde functionaliteit) alleen komt er wat extra's mee.

2 ophalen feestdagen
voor het ophalen van de feestdagen is alleen de periode nodig, welke is meegekomen vanuit de departmentperiod. Dit gaat als volgt:



Let er hierbij op dat:

- Er alleen periodes opgehaald worden die:
    - waarvan het type Holiday is EN
        - Een startdatum hebben binnen de start –en einddatum van de opgegeven periode OF
        - Een einddatum hebben binnen de start –en einddatum van de opgegeven periode OF
        - Een start –en einddatum hebben binnen de start –en einddatum van de opgegeven periode

3. Sprint 2

Dit hoofdstuk bespreekt het functioneel ontwerp van sprint 2. Eerst worden de backlog items benoemd, vervolgens wordt per backlog item een conceptueel diagram getoond, met daarbij een conceptuele sequentie diagram. Deze diagrammen samen tonen aan hoe de backlog item gerealiseerd wordt.

3.1 Sprint Backlog

De volgende backlog items zijn in deze sprint gerealiseerd:

User story	Module	Story points
Als afdelingsmanager wil ik diensten kunnen verwijderen zodat ik optimaal kan roosteren	Ro	4
Als afdelingsmanager wil ik kunnen zien of dat ik genoeg mensen heb ingepland op een specifiek tijdstip op basis van prognose zodat ik nog mensen kan inhuren als ik tekort kom/minder mensen kan plannen als ik er genoeg heb	Ro/P	15
Als gebruiker wil ik kunnen inloggen en de juiste rechten krijgen zodat ik mijn functie kan vervullen.	Auth	2
Als afdelingsmanager wil ik een akkoord kunnen geven op het rooster zodat de winkelmanager weet dat ik klaar ben met roosteren	Ro	1

3.2 Verwijderen dienst(en)

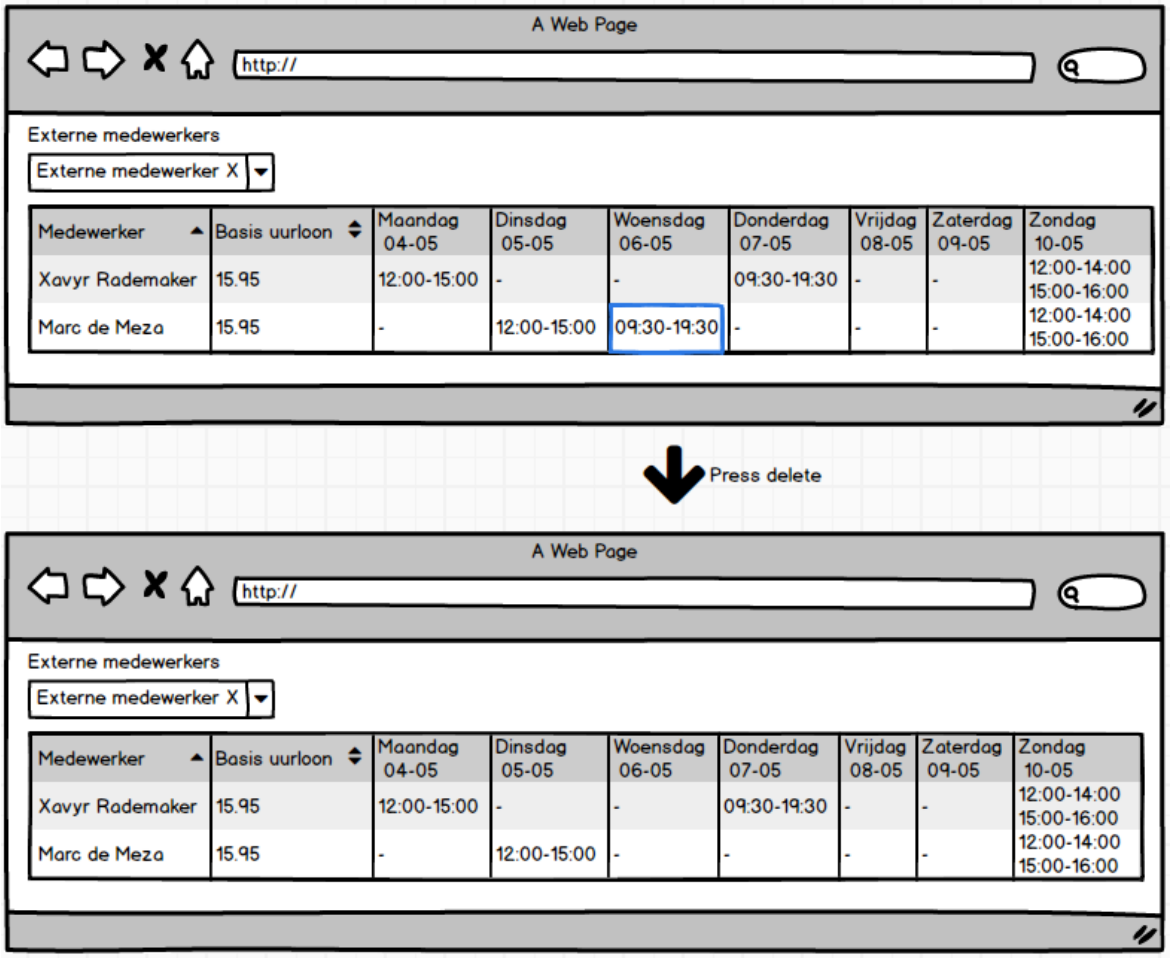
Voor het verwijderen van diensten zijn 2 mogelijkheden:

1. Er wordt 1 enkele dienst van 1 medewerker verwijderd uit het rooster(en alle roosters waar deze dienst in voorkomt)
2. Alle diensten van 1 medewerker op 1 dag worden verwijderd uit het rooster(en alle roosters waar deze dienst in voorkomt)

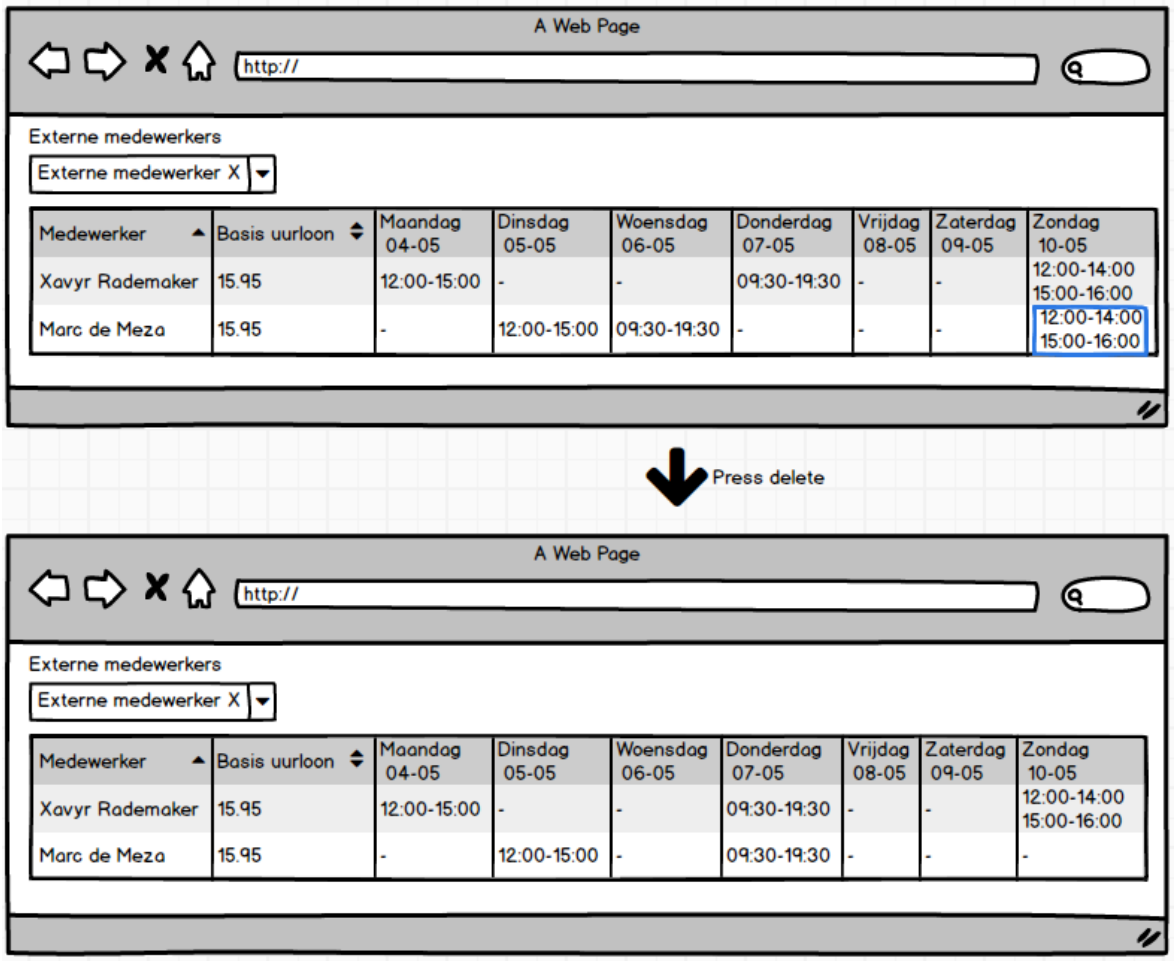
3.2.1 Mock up(s)

Deze 2 mogelijkheden zien er als volgt uit:

Verwijderen 1 enkele dienst:



Verwijderen alle diensten van 1 medewerker op 1 dag:

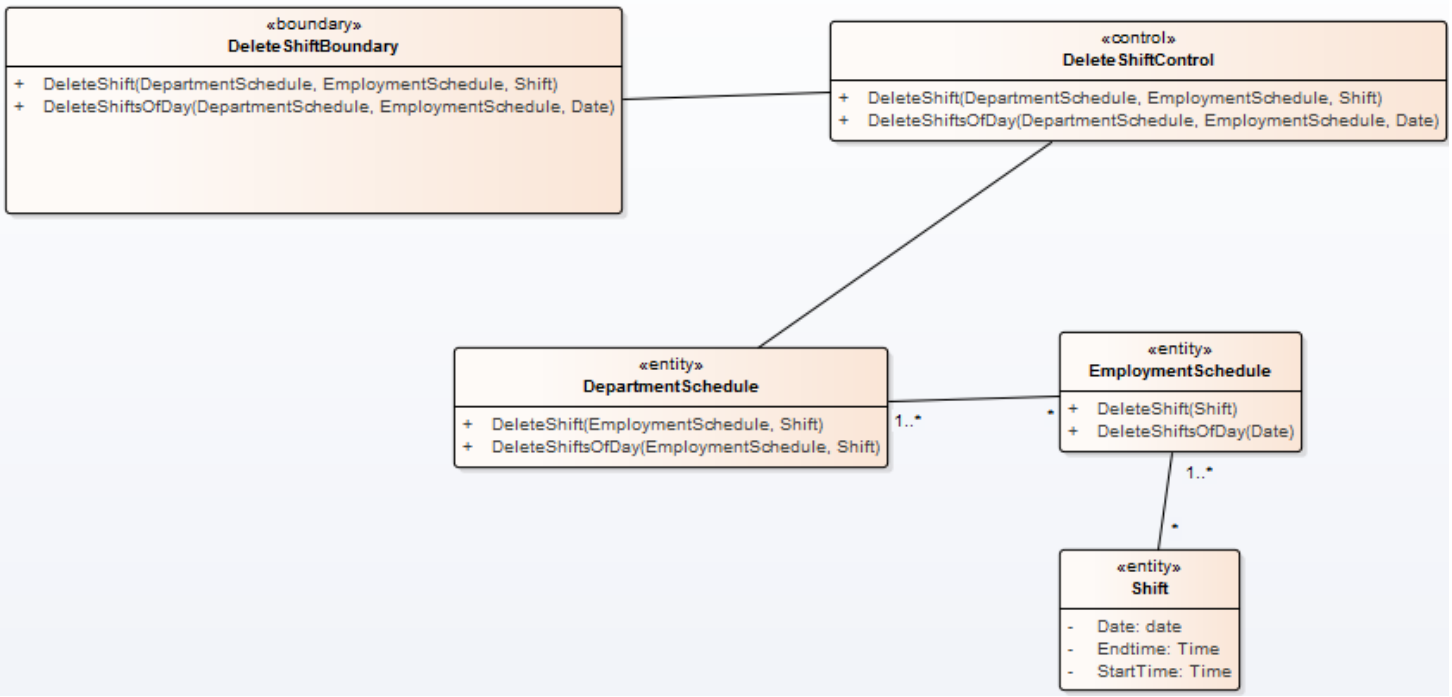


3.2.2 Entiteiten

Hiervoor zijn geen nieuwe entiteiten nodig, aangezien het gaat over diensten die ook in sprint 1 zijn gebruikt.

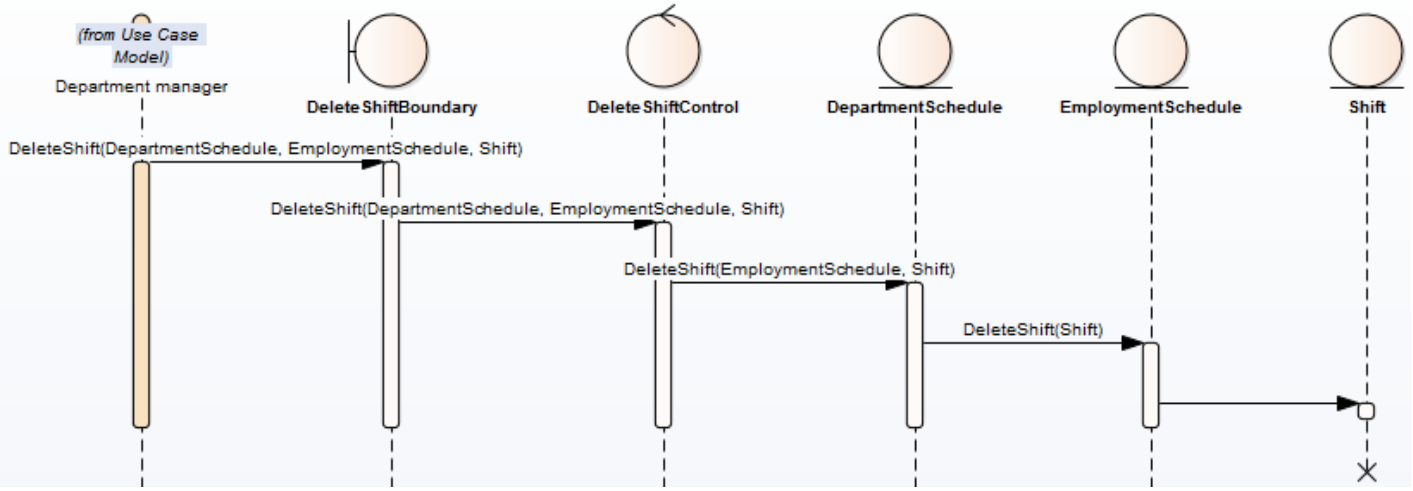
3.2.3 Controller en boundary

Voor het verwijderen van diensten zijn wel een nieuwe control en boundary gebruikt. Deze zagen er als volgt uit:

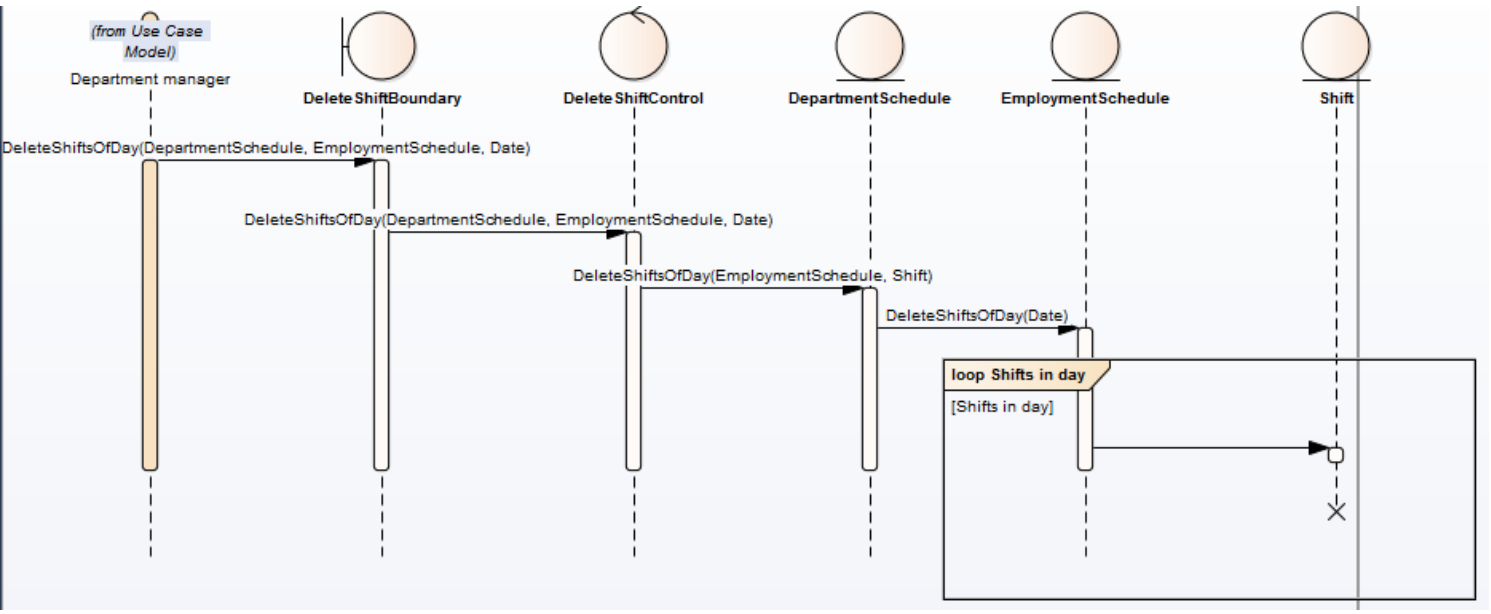


3.2.4 Sequentie diagram(men)

Het verwijderen van 1 enkele dienst van een medewerker gaat als volgt:



Het verwijderen van alle diensten van 1 dag van 1 medewerker gaat als volgt:



3.3 Zien prognose data

Het zien van de prognose data houdt het volgende in: de gebruiker wilt per dag, per uur in een grafiek zien hoeveel productieve uren hij moet inroosteren. Een voorbeeld hiervan is:

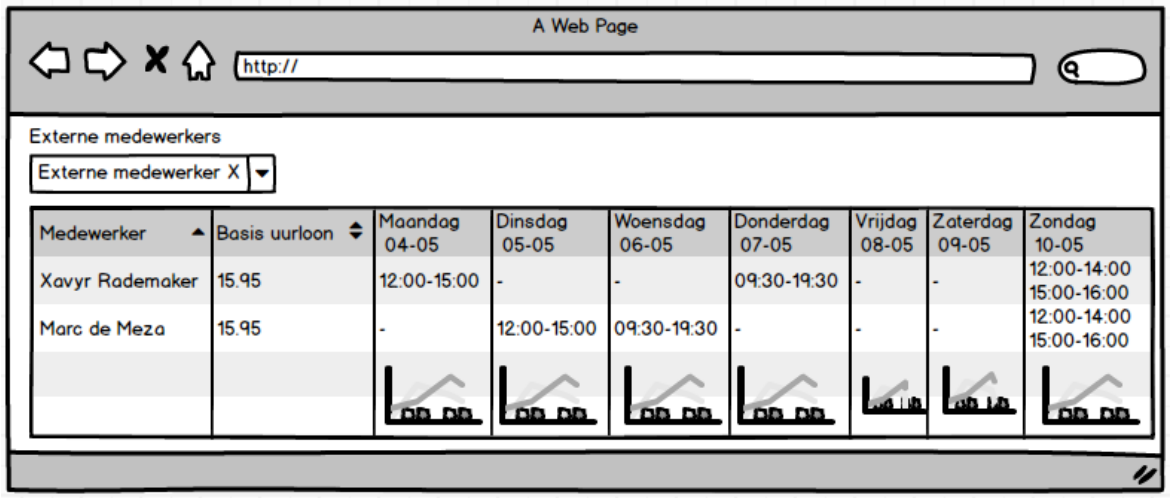
Maandag 03-04 van 06:00 tot 07:00 zijn 3:00 productieve uren nodig. 1 productief uur is 1 medewerker die 1 uur werkt. Oftewel 3:00 productieve uren van 06:00 tot 07:00 houdt in 3 medewerkers die van 06:00 tot 07:00 uur werken.

Naast het zien van hoeveel productieve uren hij moet inroosteren, moet in de grafiek ook te zien zijn hoeveel productieve uren hij momenteel heeft ingeroosterd. Ook dit moet per uur te zien zijn.

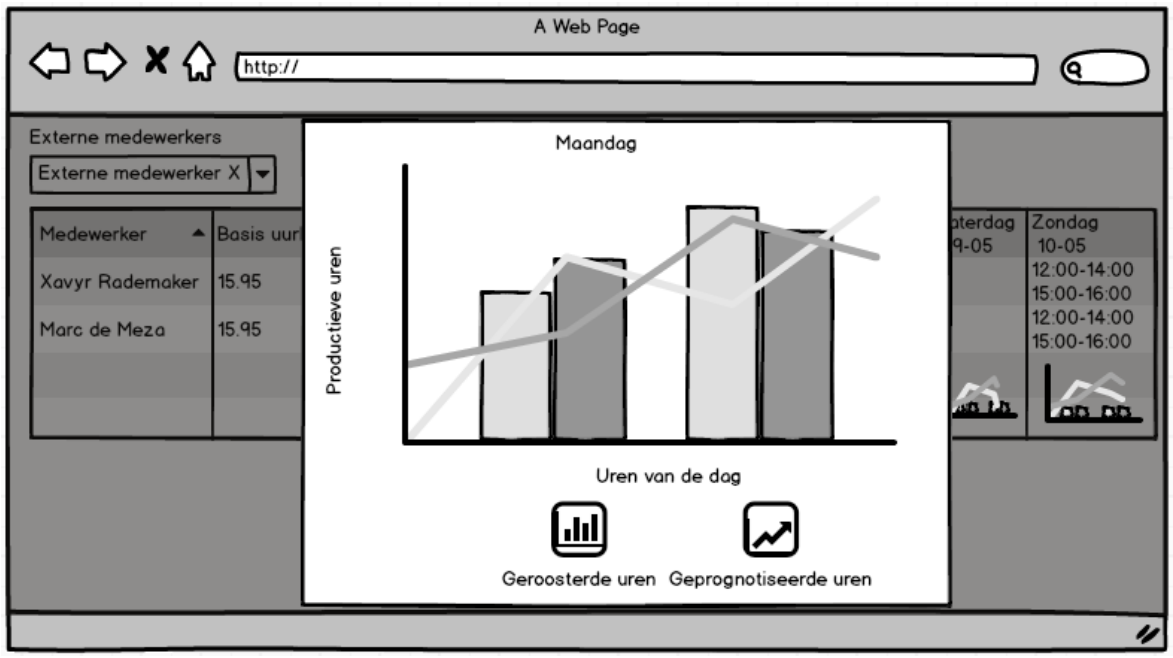
De informatie over hoeveel de gebruiker per uur zou moeten inroosteren is output van de prognose module. Er hoort altijd maximaal 1 rooster bij 1 prognose(en andersom). Deze rooster/prognose horen namelijk bij dezelfde afdeling in dezelfde periode(dezelfde departmentperiod).

3.3.1 Mock up(s)

Het zien van de grafieken in het rooster ziet er als volgt uit:

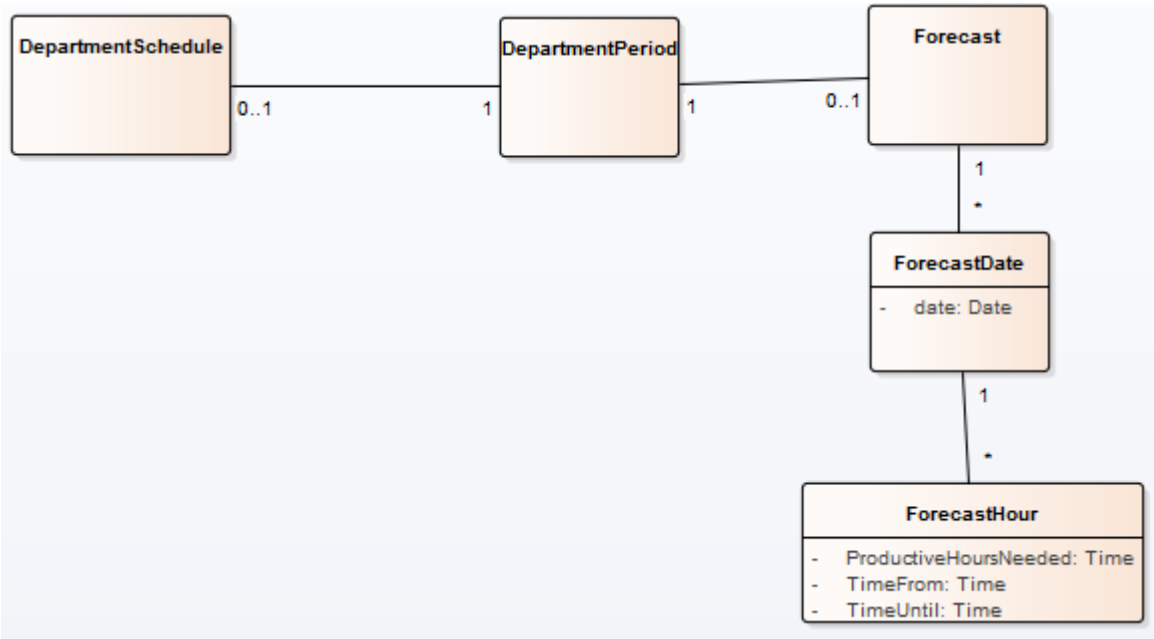


Als er vervolgens op een grafiek geklikt wordt, wordt deze in het groot weergeven. Dit is te zien op de onderstaande afbeelding.



3.3.2 Entiteiten

Om deze informatie te kunnen weergeven waren een aantal nieuwe entiteiten/relaties tussen entiteiten nodig. Deze waren als volgt:



Een korte uitleg per nieuwe entiteit:

- **Forecast**: De volledige afdelingsprognose, dit wilt zeggen alles wat er is voorspeld voor een bepaalde afdeling in een bepaalde periode
- **ForecastDate**: 1 datum binnen de voorspelling(bijvoorbeeld maandag 04-05)
- **ForecastHour**: 1 uur binnen de dag waarover een voorspelling gaat(bijvoorbeeld 12:00 t/m 13:00). Hierin zitten ook de voorspelde productieve uren.

3.3.3 Sequentie diagram(men)

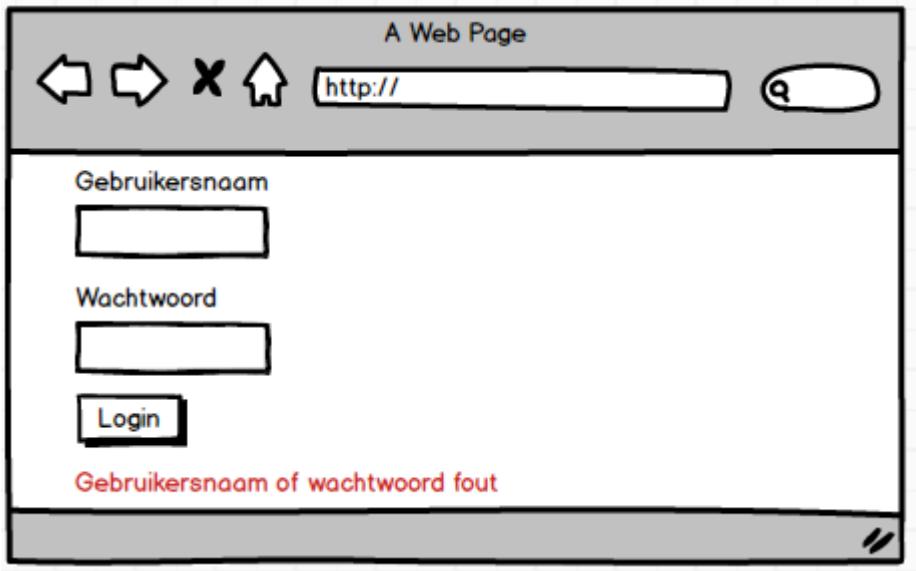
Het ophalen van de prognose, behorende bij een afdelingsrooster, gaat op dezelfde manier als het ophalen van een afdelingsrooster (zie paragraaf 2.2). De prognose gegevens komen via de departmentperiod mee met het afdelingsrooster.

3.4 Inloggen

Het inloggen gebeurd door middel van een gebruikernaam en een wachtwoord. Op het moment dat deze goed zijn, wordt de gebruiker doorgestuurd naar een pagina die past bij zijn rol(een afdelingsmanager zal doorgestuurd worden naar het zoeken van een afdelingsrooster).

3.4.1 Mock up(s)

Het inlogscherm ziet er als volgt uit:



De tekst onderin de afbeelding(gebruikersnaam of wachtwoord fout) wordt alleen getoond als de credentials niet kloppen. In dit geval wordt de gebruiker niet doorgestuurd, want hij is niet ingelogd.

3.4.2 Entiteiten

Voor het inloggen zijn 2 dingen nodig, namelijk:

- Een **account**, de gebruiker die wilt inloggen.
- **Rollen** van de account(winkelmanager, afdelingsmanager enz.)

Deze entiteiten zien er als volgt uit:

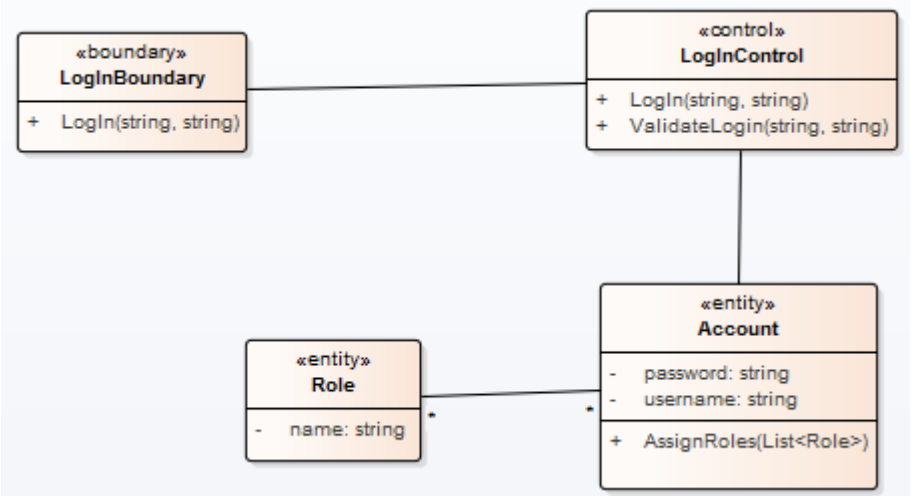
Een account kan meerdere rollen hebben(je kan winkelmanager en afdelingsmanager zijn bijvoorbeeld). Verder kan een rol ook bij meerdere accounts horen(een winkel kan meerdere afdelingsmanagers hebben).

3.4.3 Constraints

De rechten die een gebruiker in het systeem heeft, zijn gebaseerd op zijn rol. De rollen moeten dus goed gedefinieerd worden, maar ook de minimale rechten geven aan een rol(op deze manier zijn combinaties van rollen en rechten makkelijker te verdelen).

3.4.4 Controller en boundary

Voor het inloggen zijn een controller en een boundary nodig. De boundary communiceert met de gebruiker, en laat hem weten of het inloggen wel of niet gelukt is(doorsturen naar nieuwe pagina of melding geven dat het verkeerd is gegaan). De control doet de daadwerkelijke validaties(bestaat de gebruiker met de opgegeven credentials).



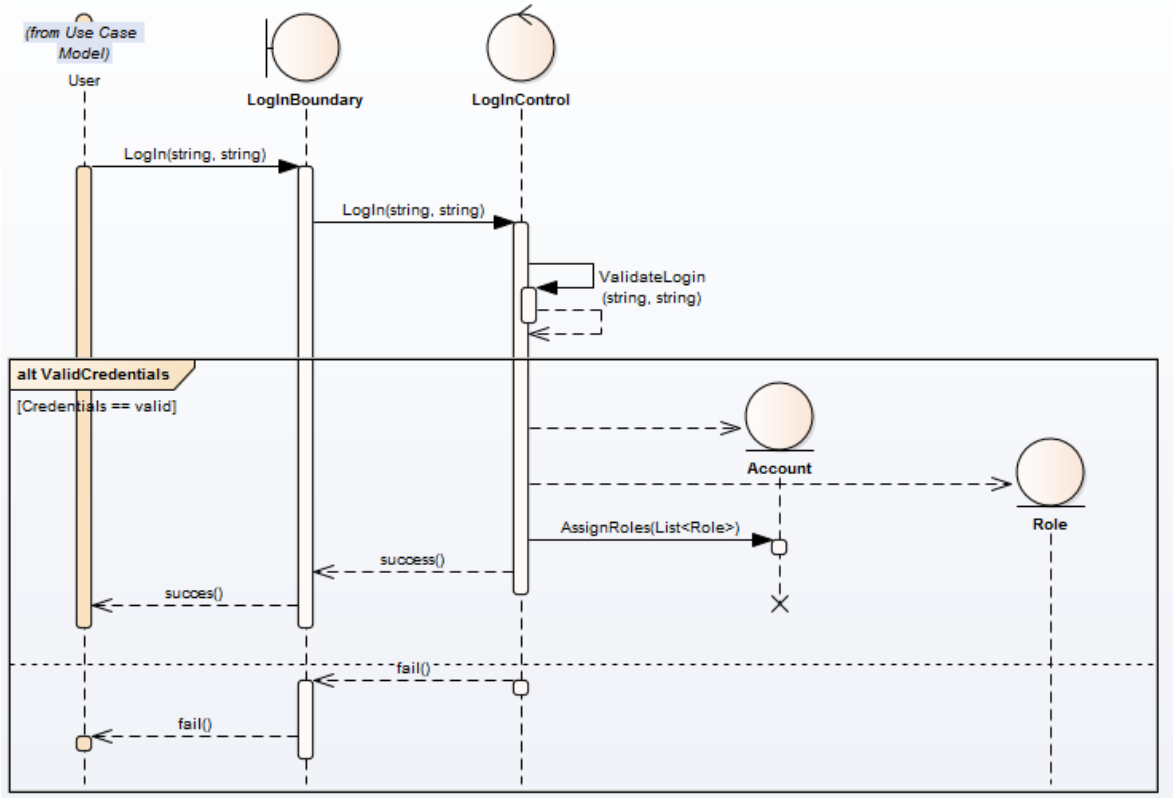
3.4.5 Sequentie diagram(men)

Voor het inloggen zijn er 2 mogelijke situaties, namelijk:

1. Het inloggen verloopt succesvol
2. De opgegeven credentials zijn niet geldig

Om deze situaties te realiseren is de volgende diagram gemaakt:

Hiermee konden de 2 situaties gerealiseerd worden op de volgende manier:

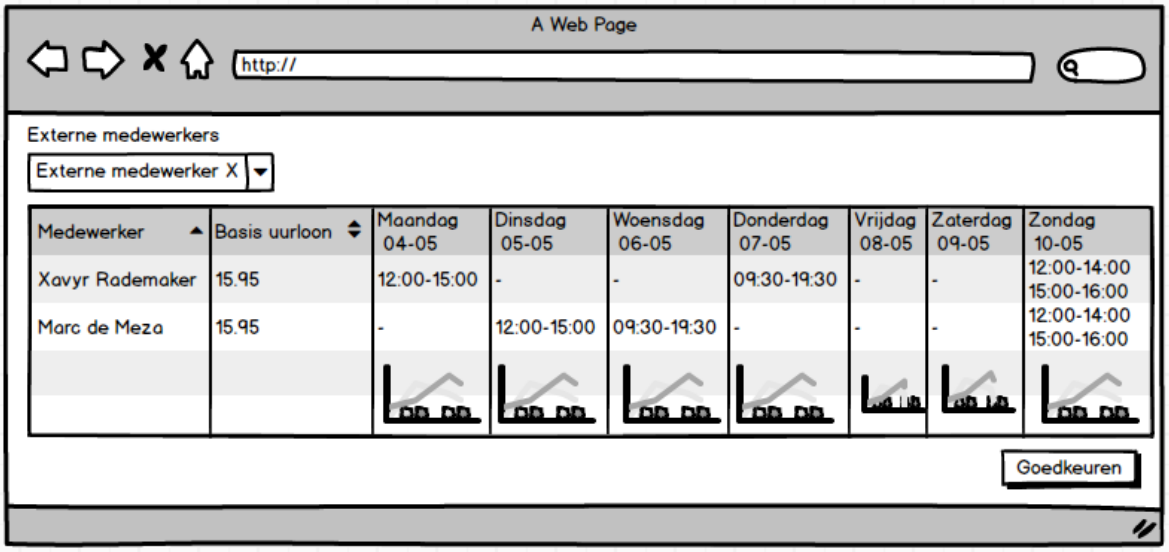


3.5 Akkoord geven op afdelingsrooster

Het akkoord geven op het afdelingsrooster gebeurd door de afdelingsmanager. Op het moment dat hij akkoord geeft, veranderd de status van een afdelingsrooster van Initialized naar Proposed.

3.5.1 Mock up(s)

Het goedkeuren van een afdelingsrooster is te zien in de onderstaande afbeelding:



De gebruiker kan het rooster goedkeuren door op de knop goedkeuren te drukken.

3.5.2 Constraints

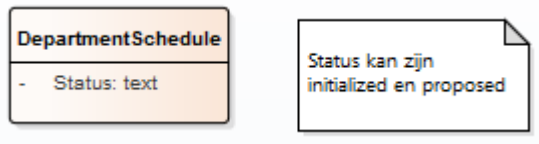
Op het moment dat een afdelingsrooster niet meer de status Initialized heeft is de goedkeuren knop disabled.

De status van een afdelingsrooster kan zijn:

- Proposed(als een afdelingsmanager het heeft goedgekeurd)
- Initialized(zonder enig goedkeuring)

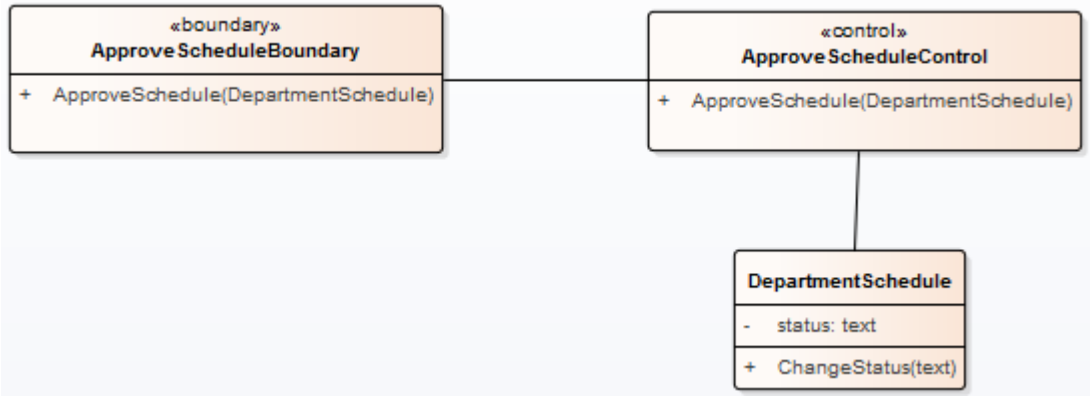
3.5.3 Entiteiten

Voor deze functionaliteit zijn geen nieuwe entiteiten nodig. De entiteit DepartmentSchedule moet alleen uitgebreid worden met een status. Dit ziet er als volgt uit:



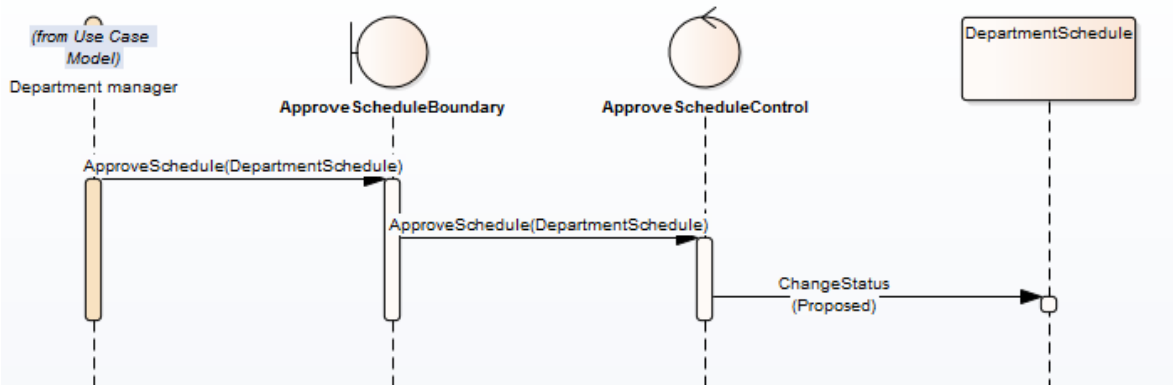
3.5.4 Controller en boundary

Om het rooster goed te keuren zijn de volgende control en boundaries nodig:



3.5.5 Sequentie diagram(men)

De onderstaande diagram laat zien hoe de functionaliteit in ze werking gaat:



4. Sprint 3

Dit hoofdstuk bespreekt het functioneel ontwerp van sprint 3. Eerst worden de backlog items benoemd, vervolgens wordt per backlog item een conceptueel diagram getoond, met daarbij een conceptuele sequentie diagram. Deze diagrammen samen tonen aan hoe de backlog item gerealiseerd wordt.

4.1 Sprint Backlog

De volgende backlog items zijn in deze sprint gerealiseerd:

User story	**Module**	**Story points**
Als winkelmanager wil alle afdelingsroosters van de winkel kunnen vaststellen zodat de staat van het rooster terug kan zien zoals ik er een akkoord op heb gegeven. afronden	R	2
als afdelingsmanager wil ik kunnen zien hoeveel tijd dat mijn medewerkers krijgen om een taak uit te voeren zodat ik niet teveel of te weinig mensen inplan en of dat ik de goede personen inplan	R/p	6
Als afdelingsmanager wil ik kunnen zien of een medewerker beschikbaar is zodat ik geen niet beschikbare mensen inrooster	R	10
Als afdelingsmanager wil ik een afdelingsrooster kunnen initialiseren zodat ik het roosterproces kan starten	R	6

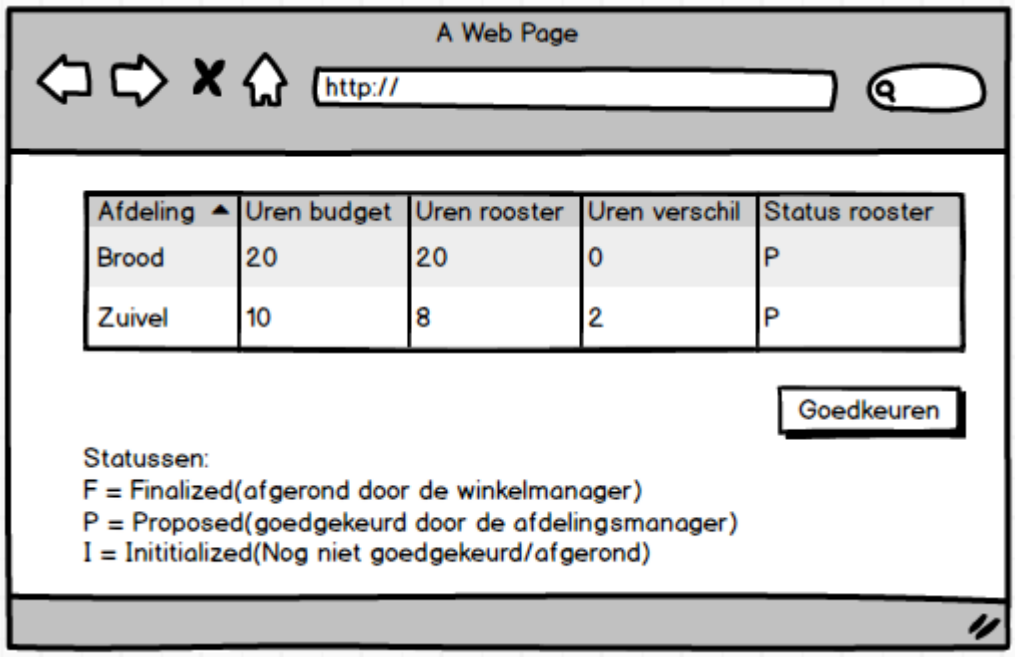
4.2 Vaststellen winkelrooster

Het vaststellen van het winkelrooster door de winkelmanager houdt het volgende in:

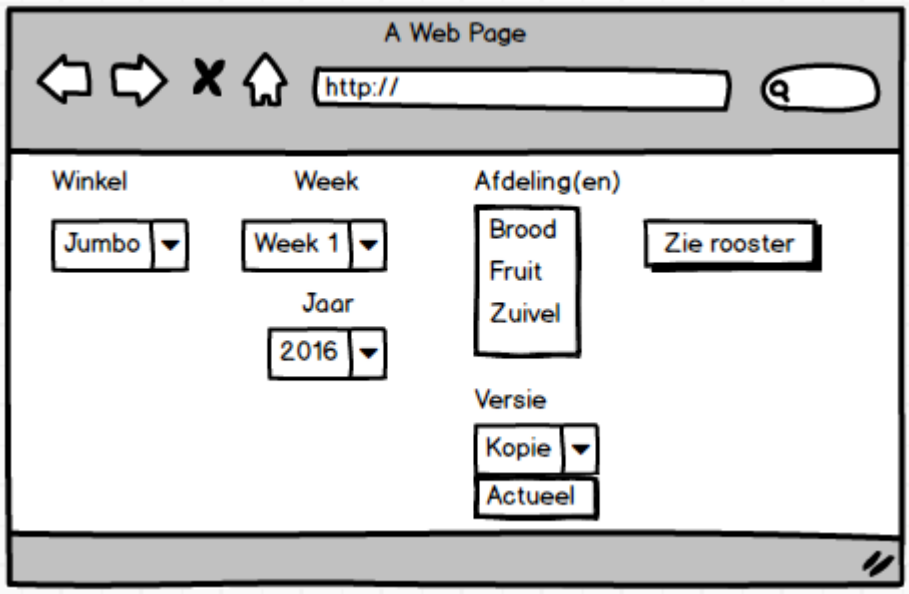
- De winkelmanager ziet van alle afdelingsroosters hoeveel budget uren(de uren uit 3.3: zien prognose data) er waren per afdelingsrooster per periode(week) en hoeveel uren er daadwerkelijk zijn ingeroosterd op de afdelingsroosters
- De winkelmanager kiest ervoor om het winkelrooster vast te stellen
- De status van de afdelingsroosters van de winkel veranderen naar Finalized
- Er wordt van elke afdelingsrooster van de winkel een kopie gemaakt, welke niet meer aangepast kan worden
    - De kopie krijgt versie 0
    - Het origineel krijgt versie 1, deze kan nog wel aangepast worden.

4.2.1 Mock up(s)

Het goedkeuren van het rooster ziet er als volgt uit:



Om beide versies(versie 1 en versie 0) terug te kunnen zien moet het zoeken op afdelingsroosters aangepast worden. Hier moet namelijk ook gekozen kunnen worden welke versie je wilt zien. Dit scherm gaat er als volgt uitzien:



4.2.2 Constraints

De selectbox moet alleen aan te passen zijn op het moment dat de status van de afdelingsrooster van de geselecteerde afdeling in de geselecteerde periode “Finalized” is. Anders is er maar 1 versie, en kan er dus niet gekozen worden.

Op het moment dat er een medewerker wordt ingehuurd van een andere afdeling/winkel moeten alleen de diensten van de laatste versie van het rooster worden overgenomen. Op het moment dat er een versie 1 bestaat van het rooster, betekent dit dat de diensten van de medewerker op versie 1 overgenomen moeten worden. Op het moment dat er geen versie 1 bestaat, moeten de diensten van versie 0 worden overgenomen.

4.2.3 Entiteiten

Voor het afronden van het winkelrooster zijn geen nieuwe entiteiten nodig. De constraint uit paragraaf 3.5.2 (over de status van een afdelingsrooster) moet alleen nog aangepast worden. Hierover meer in de volgende paragraaf.

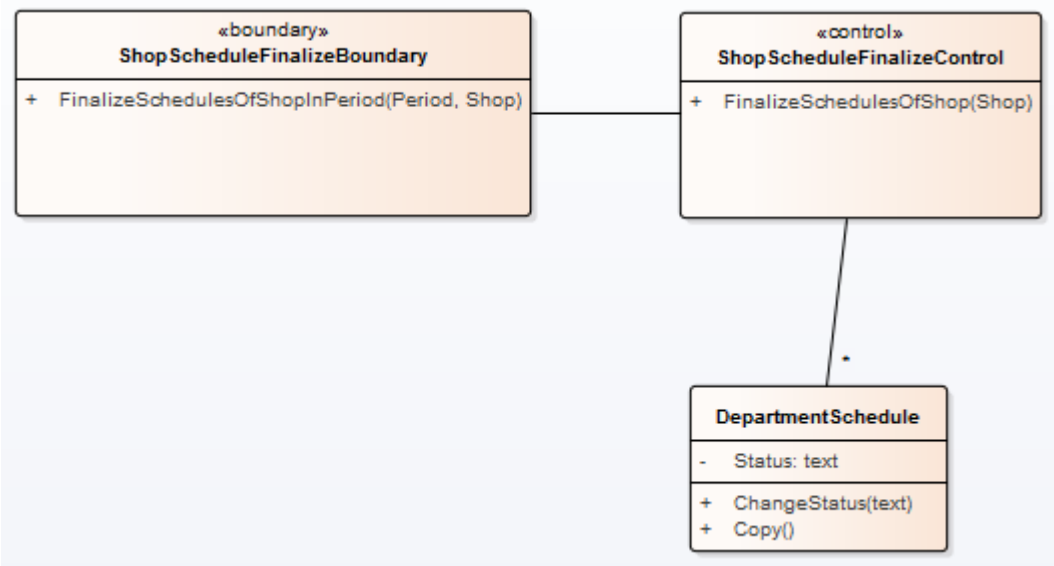
4.2.4 Constraints

Zoals in de vorige paragraaf beschreven moet de constraint op de status van een afdelingsrooster bijgewerkt worden. Deze constraint moet veranderen naar:
De status van een afdelingsrooster kan zijn:

- Proposed (als de afdelingsmanager het heeft goedgekeurd)
- Initialized(zonder enige goedkeuring)
- Finalized(als de winkelmanager het heeft goedgekeurd)

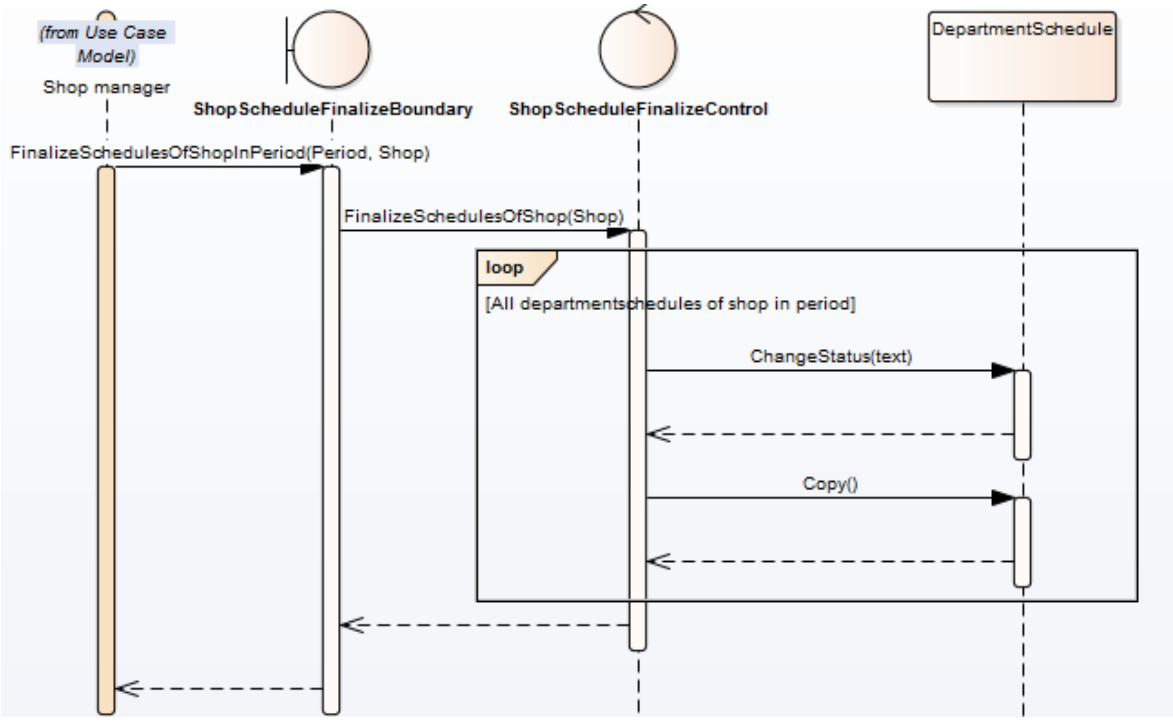
4.2.5 Controller en boundary

Om deze functionaliteit de realiseren is het volgende nodig:



4.2.6 Sequentie diagram(men)

De flow van het afronden verloopt als volgt:

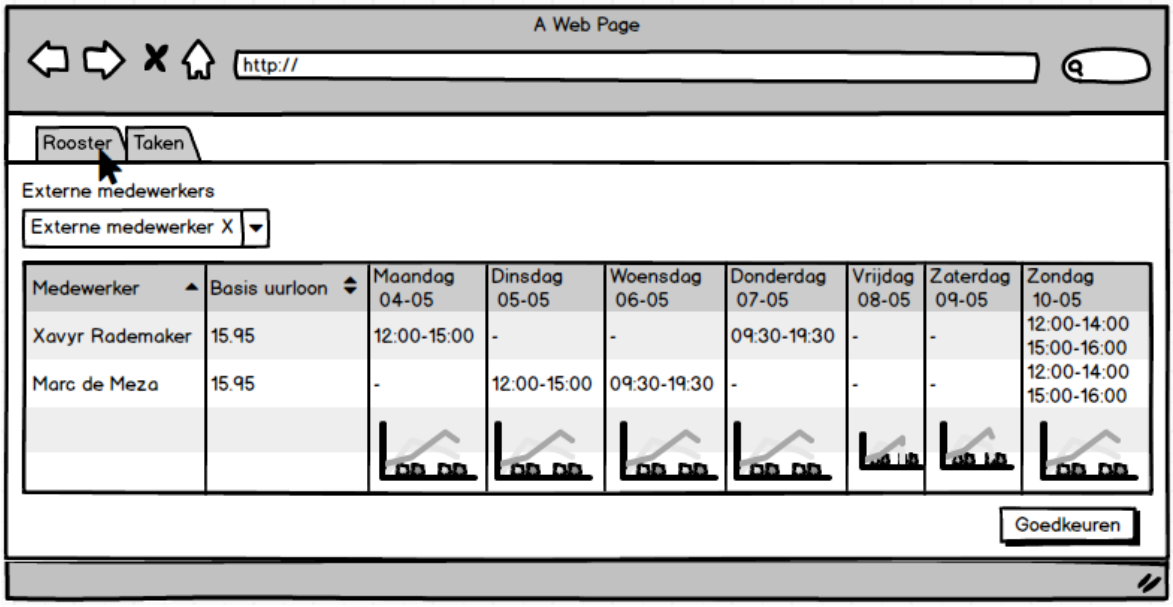


4.3 Zien budget taken

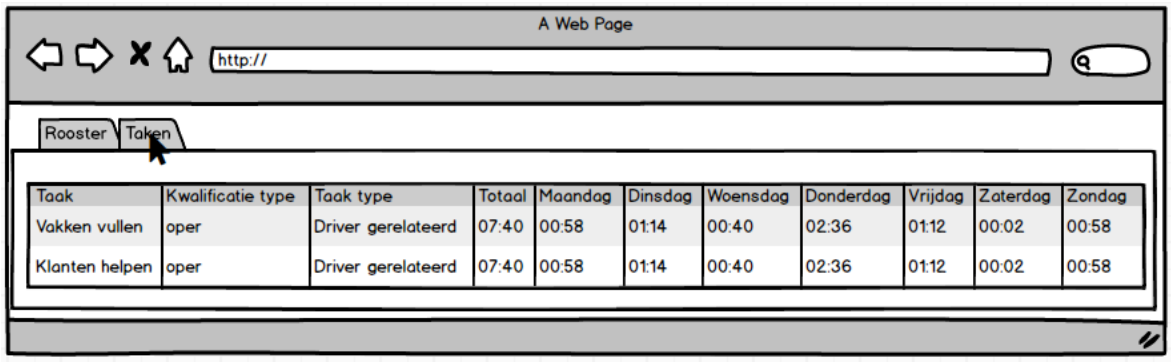
Het aantal uur wat een afdeling per dag kan besteden aan een taak(budget) is output van de prognose module. Deze moet alleen gezien worden.

4.3.1 Mock up(s)

Het zien van de budget voor de taken gaat als volgt. Op hetzelfde scherm als waar het rooster zich bevind, is er een tabblad. Deze is te zien op de volgende afbeelding:

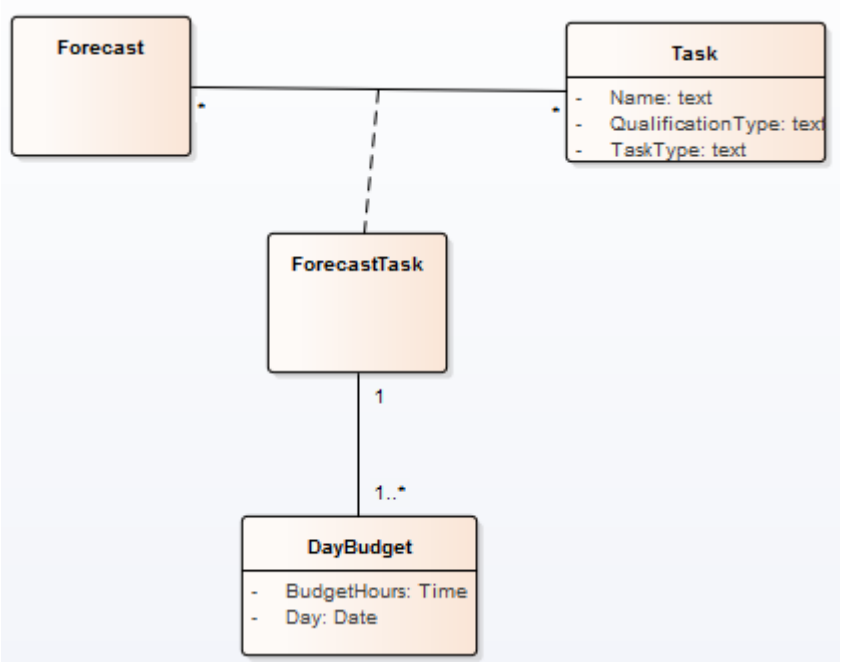


Op het moment dat de gebruiker op het tabje “taken” klikt, wordt per taak, per dag aangegeven hoeveel uur eraan bestaat kan worden. Dit ziet er als volgt uit:



4.3.2 Entiteiten

De volgende entiteiten zijn nodig om deze functionaliteit te realiseren.



Hier zijn een aantal nieuwe entiteiten te vinden, namelijk:

- **Task**: een taak welke gedaan moet worden op de afdeling(vakken vullen, klanten helpen, schoonmaken enz.)
- **ForecastTask**: de combinatie van taak en forecast(prognose) deze combinatie moet altijd minimaal 1 budget hebben
- **DayBudget**: het budget(in uren) wat besteed kan worden aan een bepaalde taak. Bijvoorbeeld: maandag 04-05, vakken vullen 3:00 uur.

4.3.3 Controller en boundary

Zoals in paragraaf 3.3 vermeld, kan de prognose data in het rooster opgehaald worden aan de hand van de departmentperiod. Daar deze data weer aan de prognose verbonden is, kan ook deze data op dezelfde manier opgehaald worden.

4.3.4 Sequentie diagram(men)

N.v.t. zie uitleg vorige paragraaf.

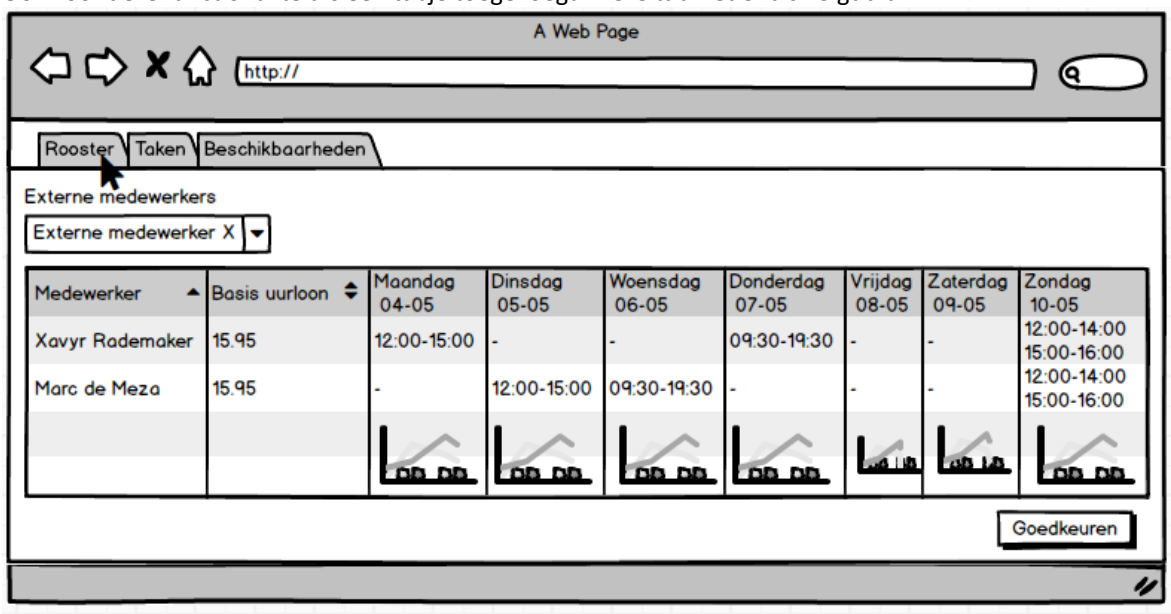
4.4 Beheren beschikbaarheid

Het beheren van de beschikbaarheid van een medewerker houdt het volgende in:

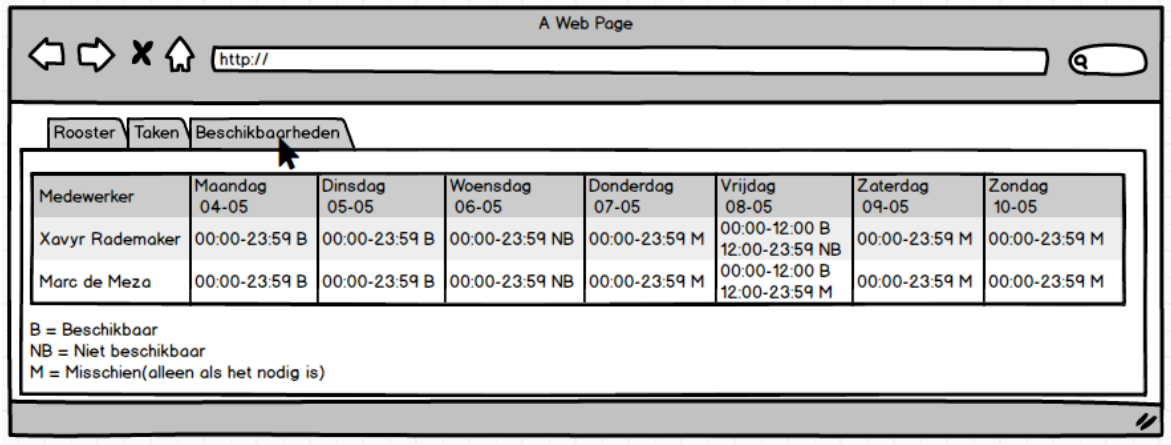
- Per dag kan er per medewerker aangegeven worden van hoelaat tot hoelaat zij:
    - Beschikbaar zijn
    - Niet beschikbaar zijn
    - Liever niet beschikbaar zijn, maar ingeroosterd kunnen worden als het moet

4.4.1 Mock up(s)

Ook voor deze functionaliteit is een tabje toegevoegd. Deze tab ziet er als volgt uit:

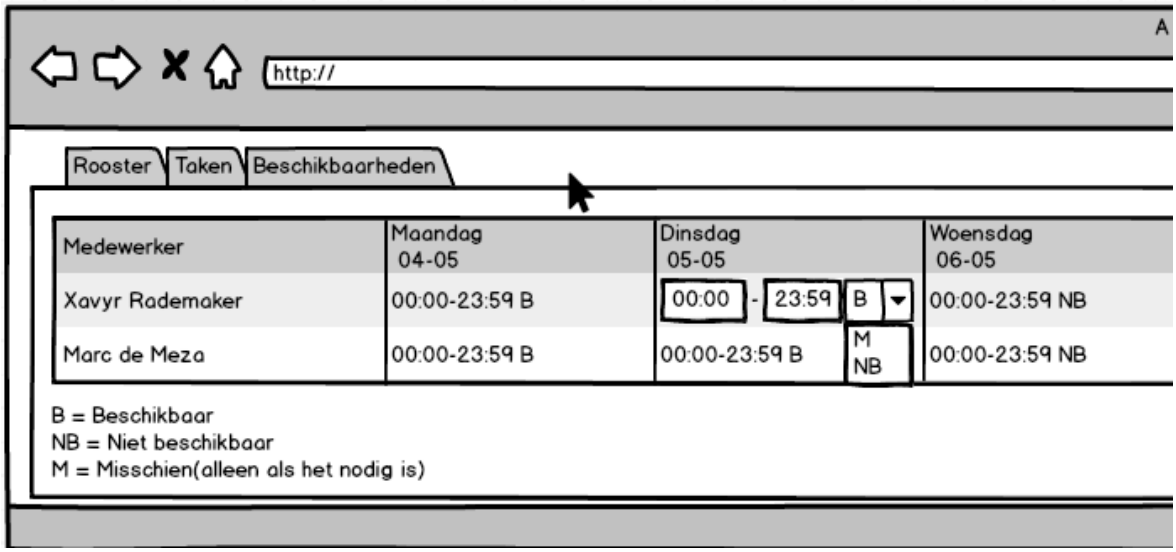


Als er op de tab beschikbaarheid geklikt wordt ziet de gebruiker het volgende:

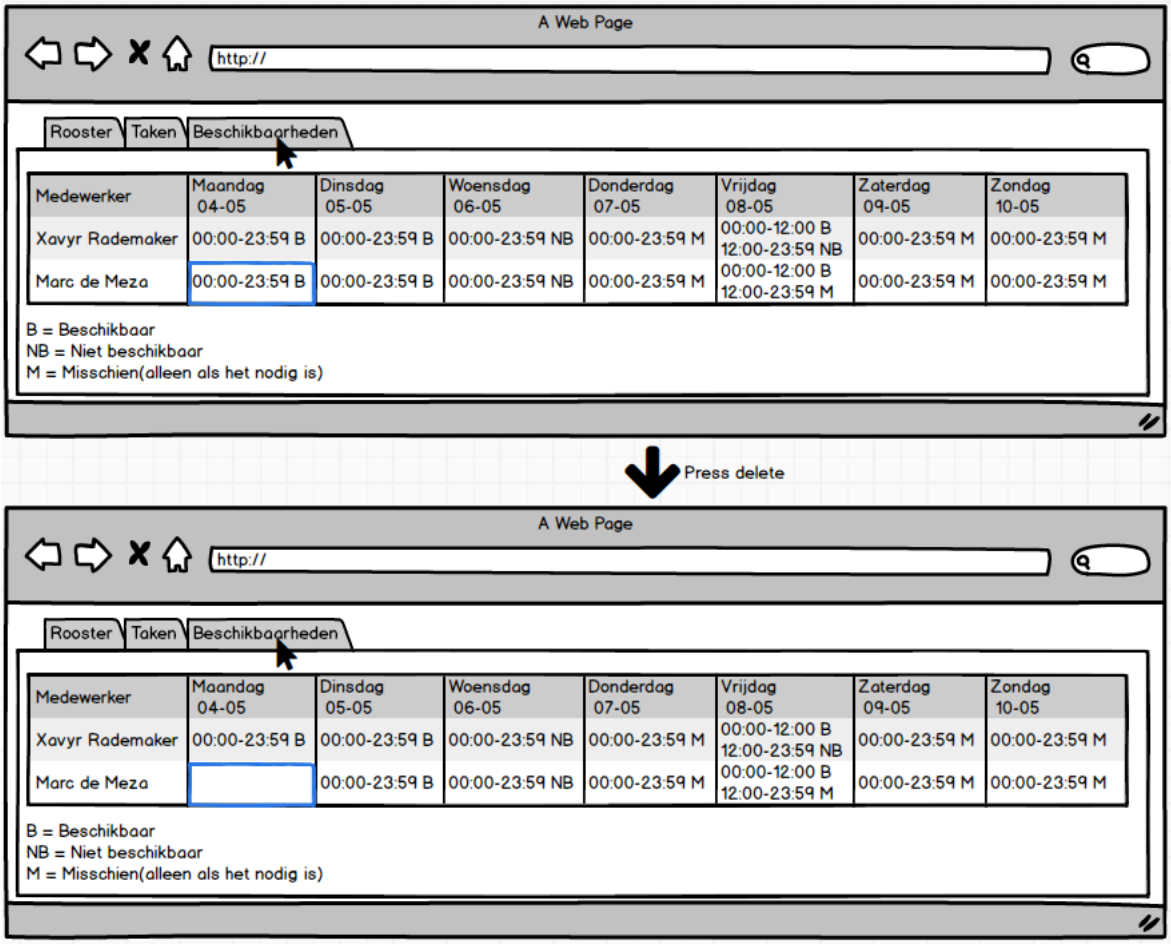


Naast het zien, kan de gebruiker ook beschikbaarheden toevoegen, wijzigen en verwijderen. Deze handelingen zien er vrijwel hetzelfde uit als het toevoegen/wijzigen/verwijderen van een dienst. Namelijk:

Toevoegen/wijzigen



Verwijderen:



4.4.2 Entiteiten

Voor het beheren van beschikbaarheden zijn de volgende entiteiten toegevoegd:

Deze entiteiten houden het volgende in:

- **Availability**: de beschikbaarheid van een medewerker. Dit is per dag, van een bepaalde tijd tot een bepaalde tijd. Verder kan met type aangegeven worden of hij in de aangegeven tijd wel, niet of liever niet beschikbaar is.

4.4.3 Constraints

Het type van de availability mag alleen 1 van het volgende zijn:

- Beschikbaar(medewerker kan ingezet worden)
- Niet beschikbaar(medewerker kan niet ingezet worden)
- Liever niet beschikbaar(de medewerker kan ingezet worden, maar liever niet als het niet nodig is)

4.4.4 Controller en boundary

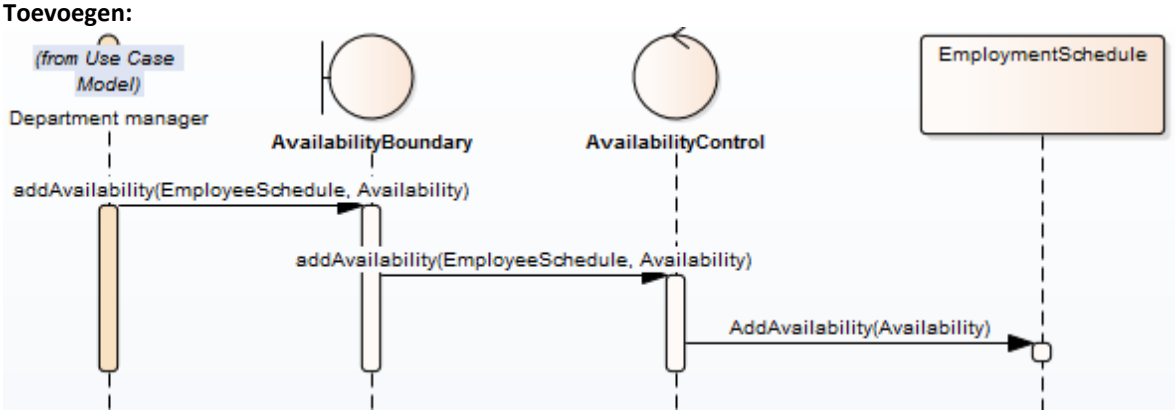
Voor het zien van de beschikbaarheden zijn geen control en boundary meer nodig, de gegevens komen mee vanuit de employmentschedule in het afdelingsrooster(departmentschedule).

Voor het toevoegen/wijzigen/verwijderen van beschikbaarheden zijn wel controls en boundaries nodig. Deze zijn hieronder te zien:

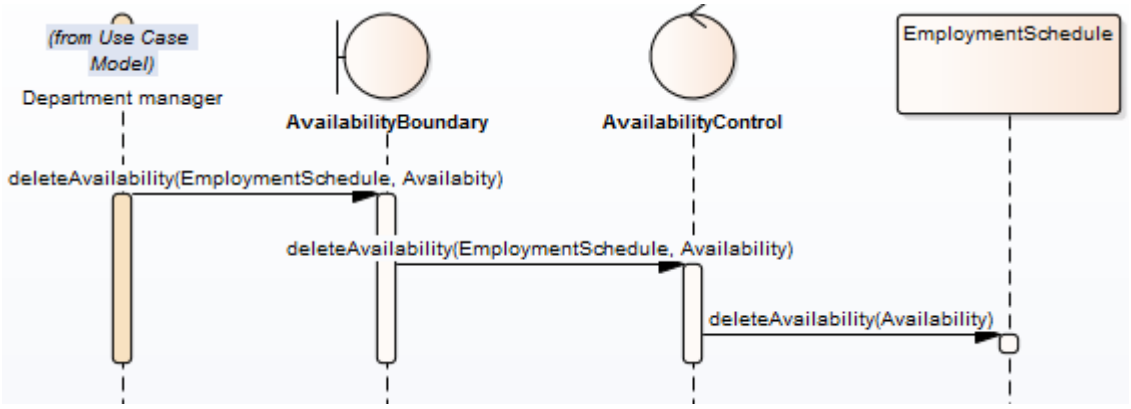
4.4.5 Sequentie diagram(men)

Per subfunctie is hieronder een sequentie diagram te vinden:

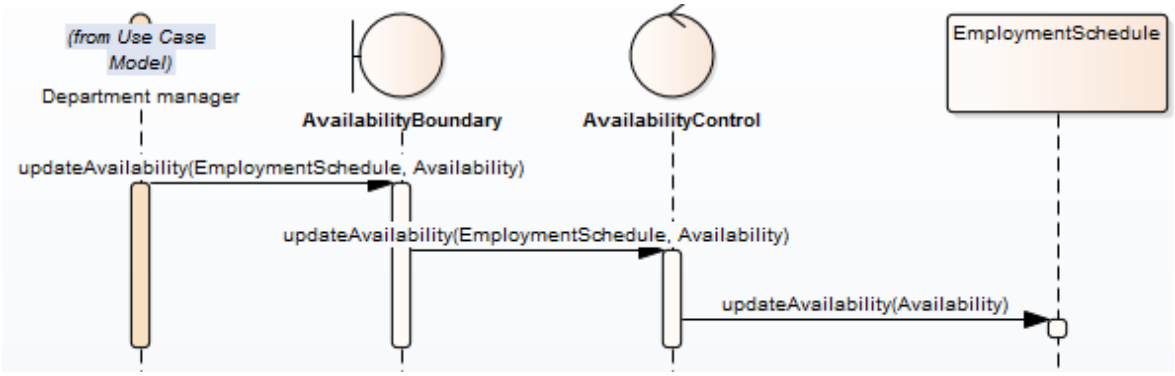
Toevoegen:



Verwijderen:



Wijzigen:



4.5 Initialiseren afdelingsrooster

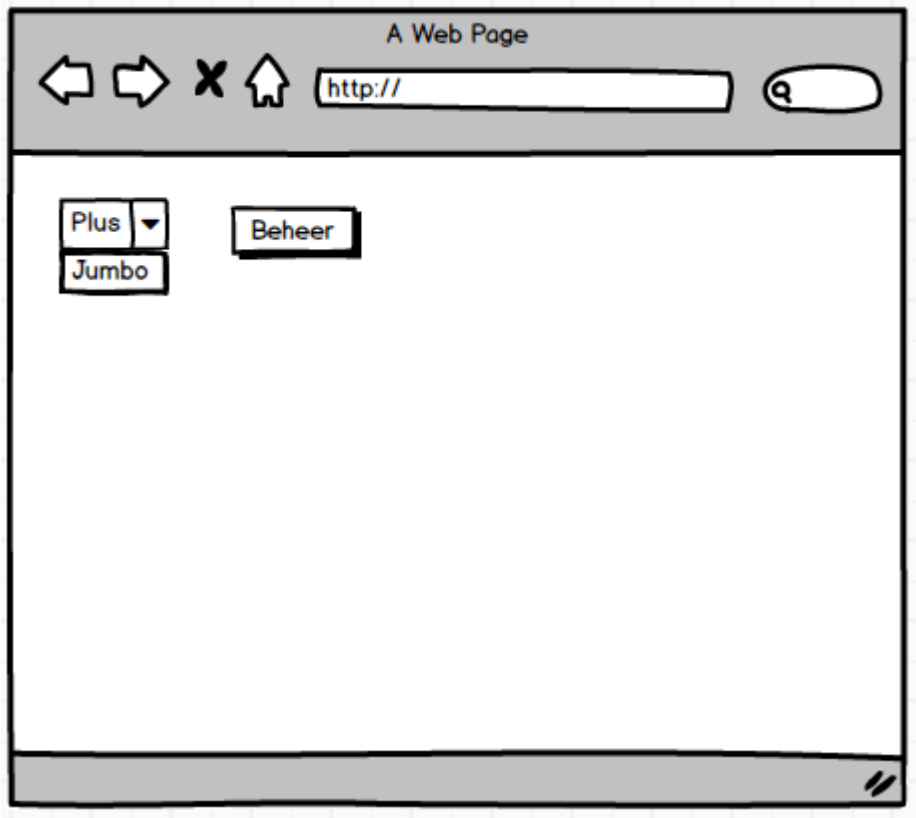
Het initialiseren van afdelingsroosters bestond uit 3 onderdelen, namelijk:

- Het aanmaken/beheren van winkelrooster templates
  - Deze hebben een label en zijn een verzameling van afdelingsrooster templates
- Het aanmaken/beheren van afdelingsrooster templates
  - Dit zijn templates van roosters, welke diensten van medewerkers bevatten(onafhankelijk van de datum van een dienst).
    - Bijvoorbeeld: Xavyr Rademaker, Maandag, 12:00-13:00
- Het initialiseren van een afdelingsrooster met behulp van de templates
  - Op het moment dat er een afdelingsrooster wordt geïnitialiseerd, worden alle diensten van de geselecteerde template overgezet naar de nieuwe gemaakte afdelingsroosters.

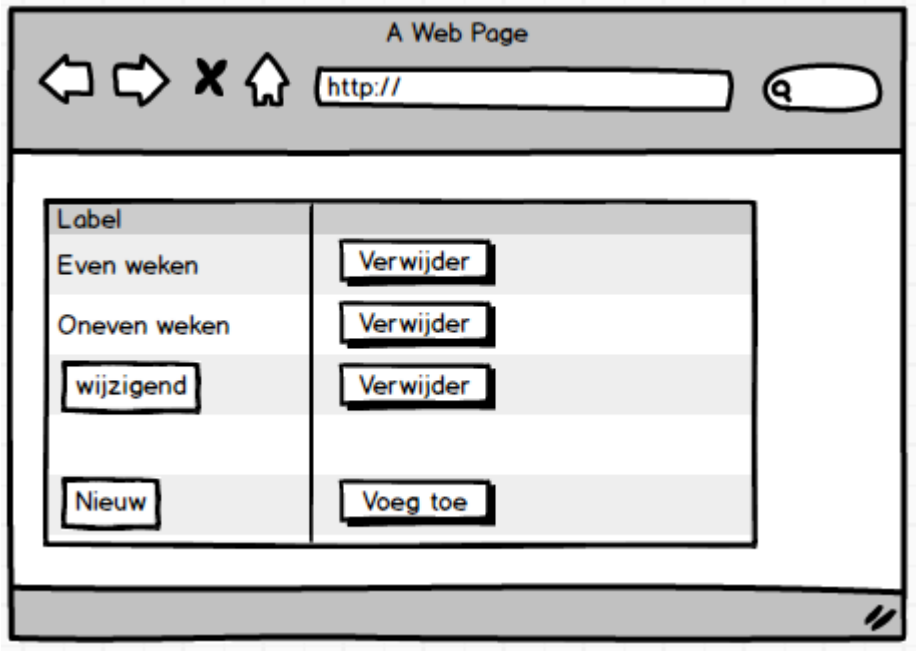
4.5.1 Mock up(s)

Winkelrooster:
Het aanmaken/beheren van de winkelrooster gaat er als volgt uitzien:

Allereerst wordt er gekozen van welke winkel de templates beheert moeten worden. Dit gaat als volgt:



In de dropdown wordt een winkel geselecteerd. Vervolgens wordt er op beheer gedrukt. Het drukken op beheer, lijdt naar het volgende scherm.



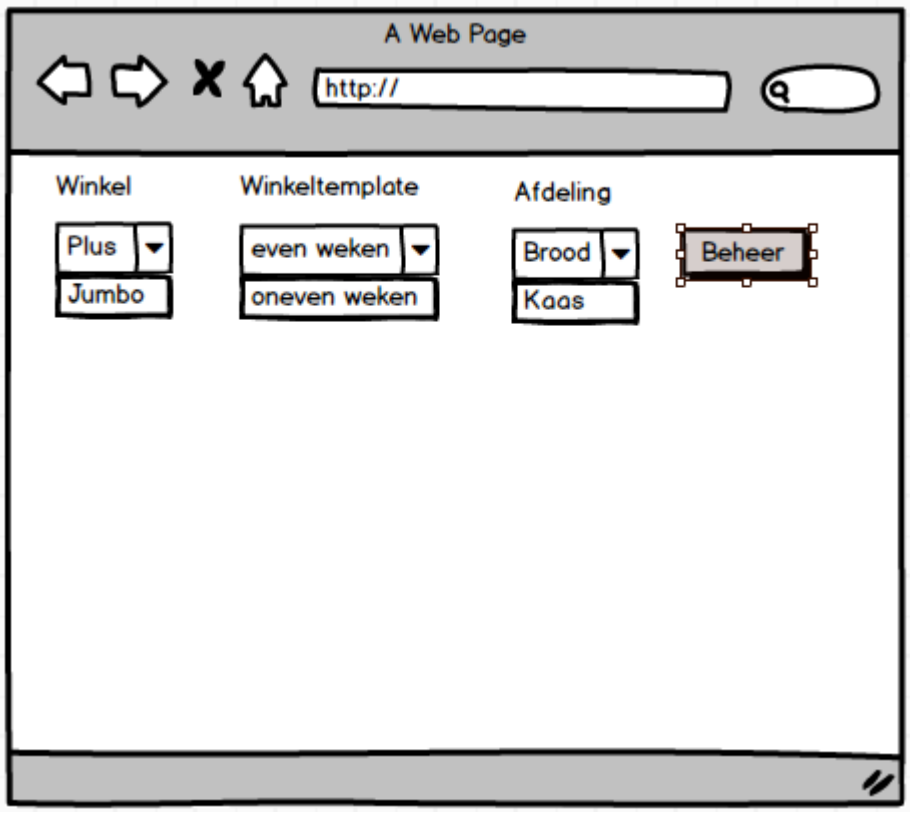
In deze tabel staan alle labels van de winkel templates. Deze kunnen gewijzigd worden(zoals te zien bij label “wijzigend”), verwijderd worden(door de buttons) en er kunnen nieuwe templates toegevoegd worden(de onderste row).

Afdelingsrooster:
Het aanmaken/beheren van afdelingsroosters gaat er als volgt uitzien:

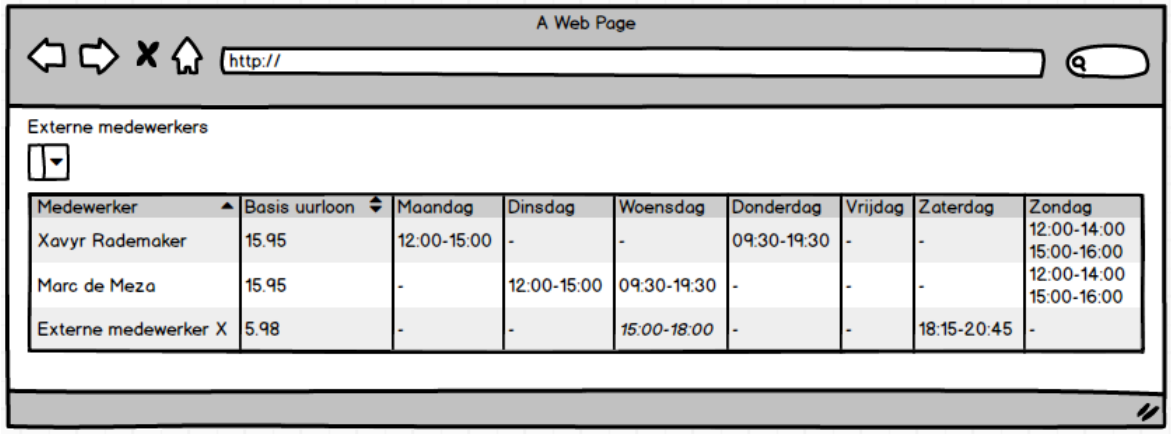
Eerst wordt er een afdelingstemplate gekozen. Dit gebeurd op basis van:

- Winkel
- Winkel template van de winkel
- Afdeling van de winkel

Deze selectie gaat als volgt:



Als er op beheer wordt gedrukt is het volgende scherm te zien:



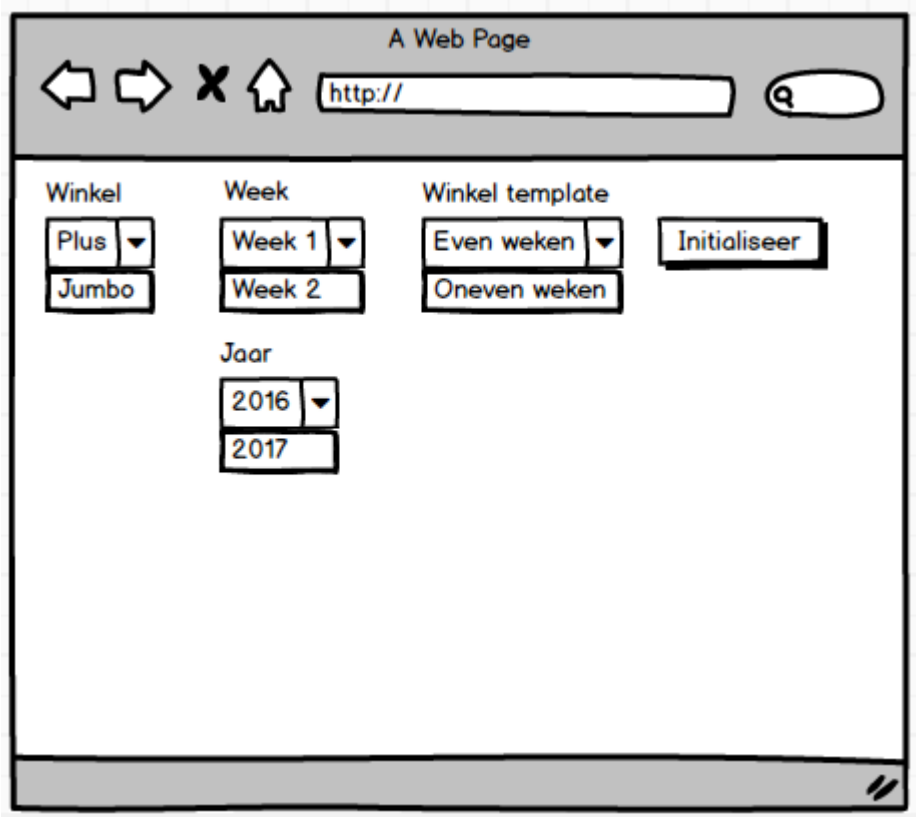
Medewerker	Basis uurloon	Maandag	Dinsdag	Woensdag	Donderdag	Vrijdag	Zaterdag	Zondag
Xavyr Rademaker	15.95	12:00-15:00	-	-	09:30-19:30	-	-	12:00-14:00 15:00-16:00
Marc de Meza	15.95	-	12:00-15:00	09:30-19:30	-	-	-	12:00-14:00 15:00-16:00
Externe medewerker X	5.98	-	-	*15:00-18:00*	-	-	18:15-20:45	-

Dit scherm is in principe een versimpelde versie van het beheren van een afdelingsrooster. De verschillen zijn:

- Beschikbaarheden en prognose data zijn hier niet zichtbaar
- Het is niet voor een specifieke week in het jaar, er zijn dus geen exacte dagen te zien (3 mei enz.).

Alles wat te maken heeft met diensten in het afdelingsrooster beheren kan, kan hier verder ook(diensten toevoegen, wijzigen, verwijderen en inhuren van externe medewerkers).

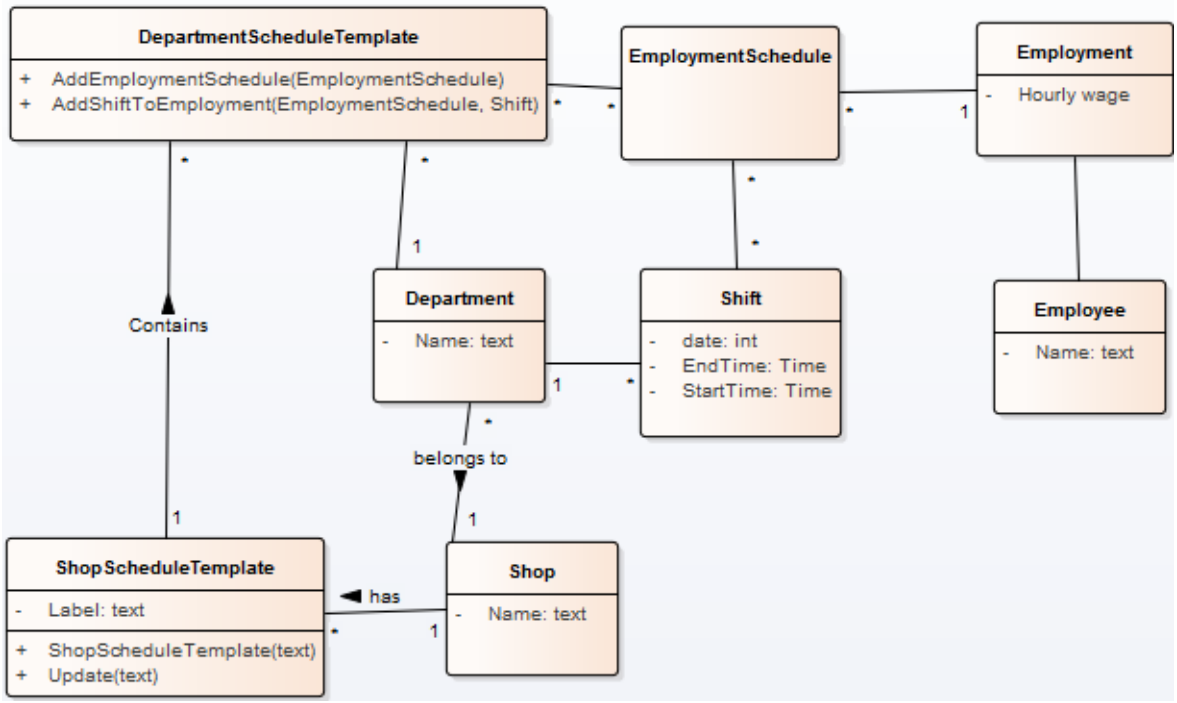
Initialiseren:
Het initialiseren/beheren van afdelingsroosters gaat er als volgt uitzien:



Er wordt een winkel, periode en template gekozen. Vervolgens klikt de gebruiker op initialiseer. Na het drukken op de knop worden de afdelingsroosters voor de geselecteerde periode aangemaakt. Ook worden alle diensten die te vinden zijn in de template toegekend aan de afdelingsroosters.

4.5.2 Entiteiten

Om deze functionaliteit te realiseren zijn de volgende entiteiten nodig:

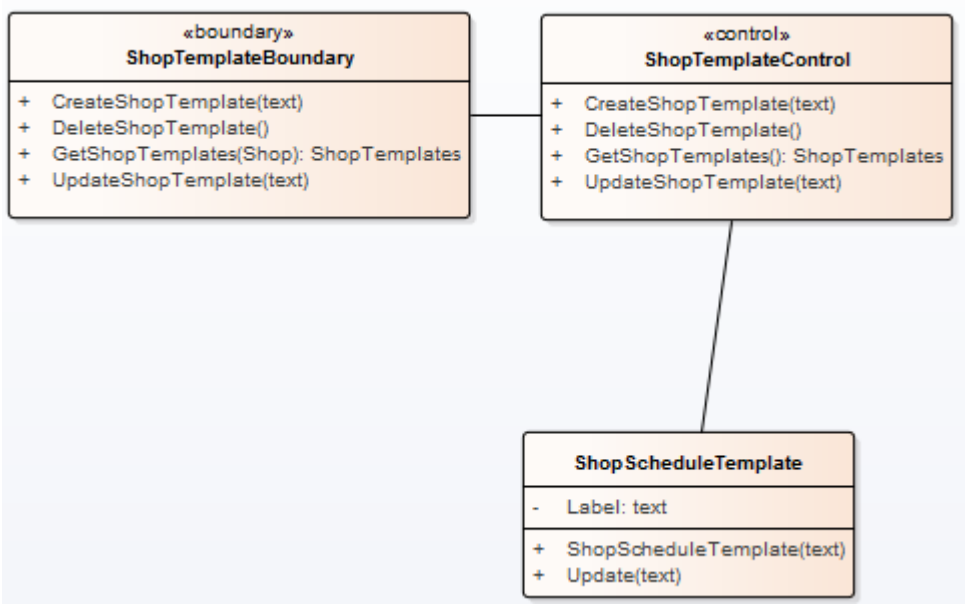


De nieuw toegevoegde entiteiten houden het volgende in:

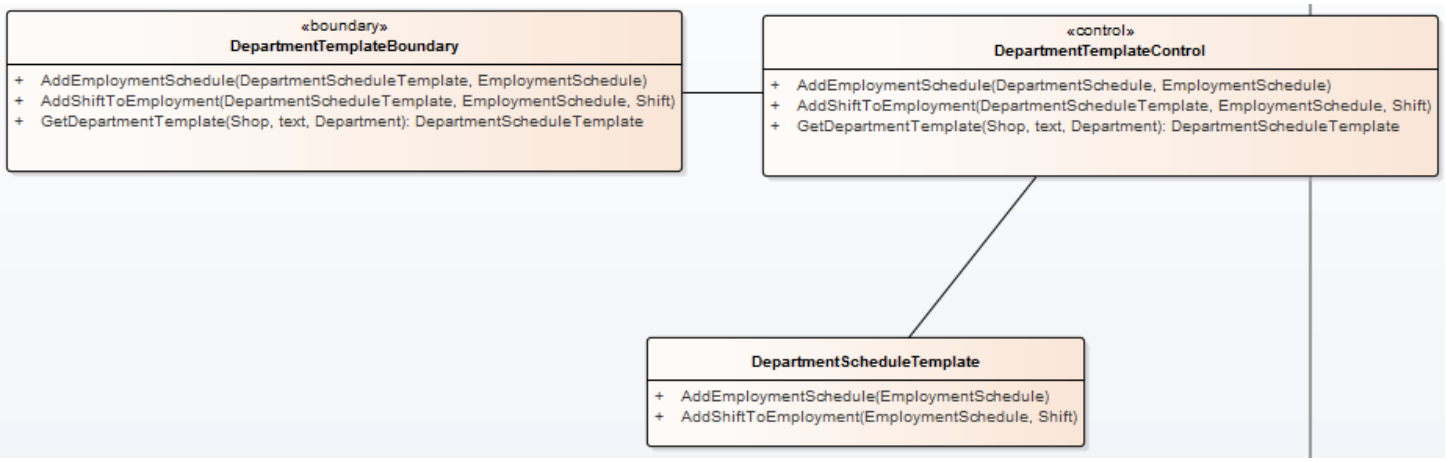
- **ShopScheduleTemplate**: de template van een winkelrooster. Deze bevat een label en de mogelijkheid om te update.
- **DepartmentScheduleTemplate:** de template van een afdelingsrooster. Deze hoort bij 1 winkelrooster template, en bevat mogelijkheden om externe medewerkers en diensten toe te voegen. Verder houdt hij de medewerkersroosters bij, die er hetzelfde uitzien als de medewerkersroosters van een afdelingsrooster(zie vorige paragrafen/hoofdstukken).

4.5.3 Controller en boundary

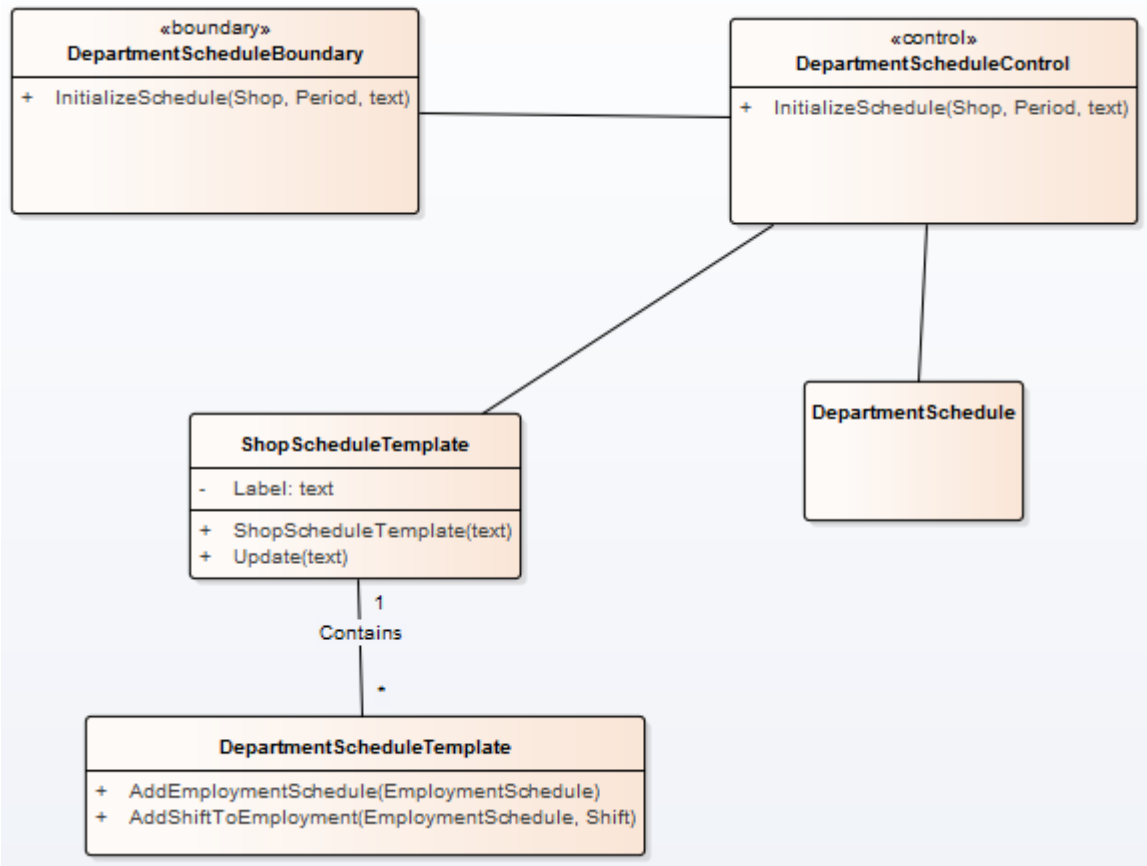
Beheren winkeltemplate:
Voor het beheren van de winkeltemplates zijn de volgende klasses nodig:



Beheren afdelingstemplate:
Voor het beheren van de afdelingstemplate zijn de volgende klasses nodig:



Initialiseren rooster:
Voor het initialiseren van het rooster zijn de volgende klasses nodig:

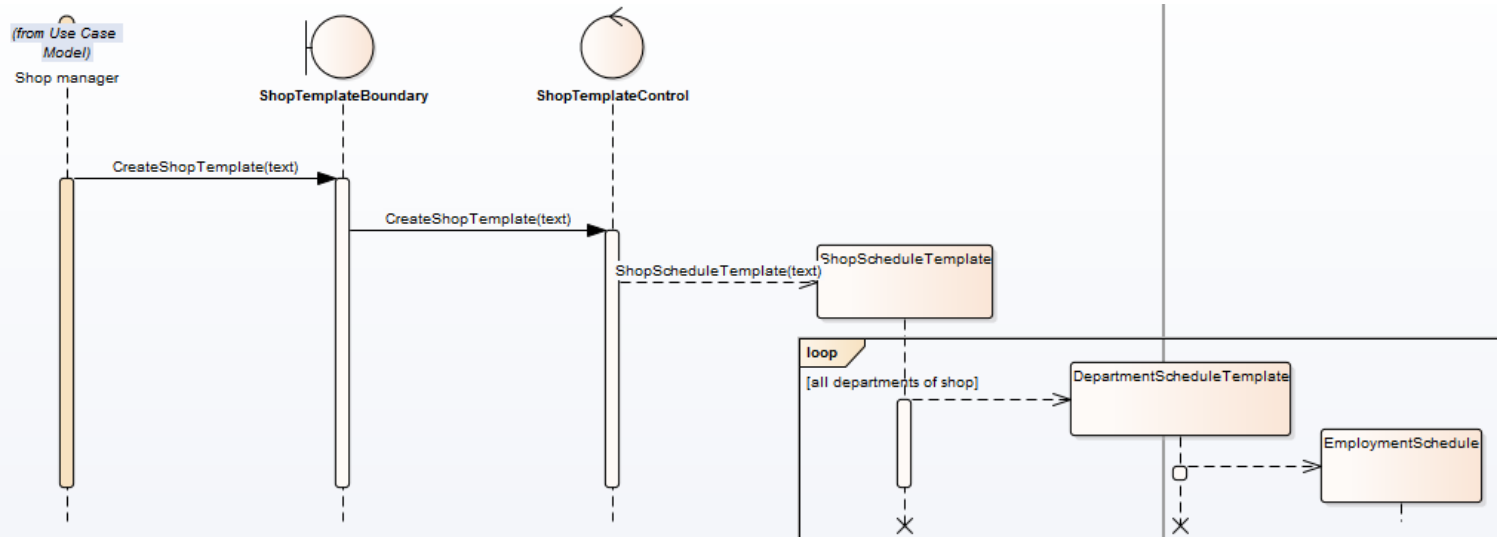


4.5.4 Sequentie diagram(men)

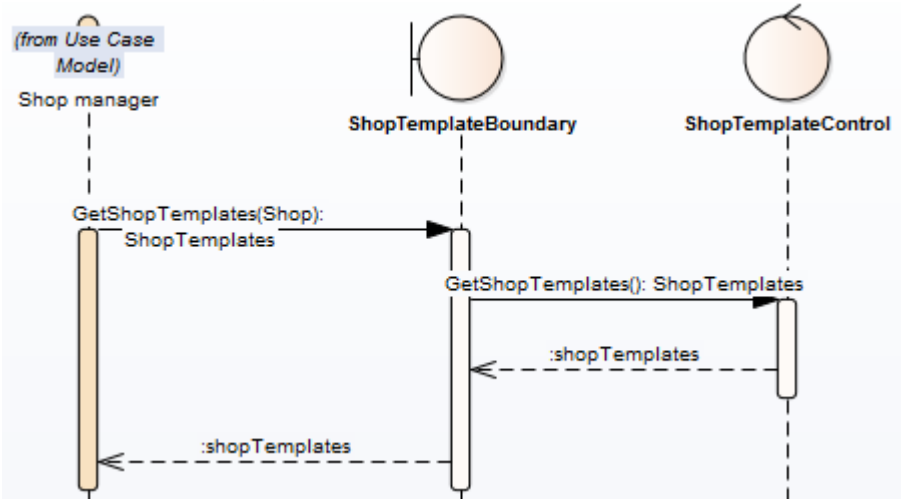
De klasses uit de vorige paragraaf werken op de volgende manier samen om de functionaliteiten te realiseren:

Beheren winkeltemplate:

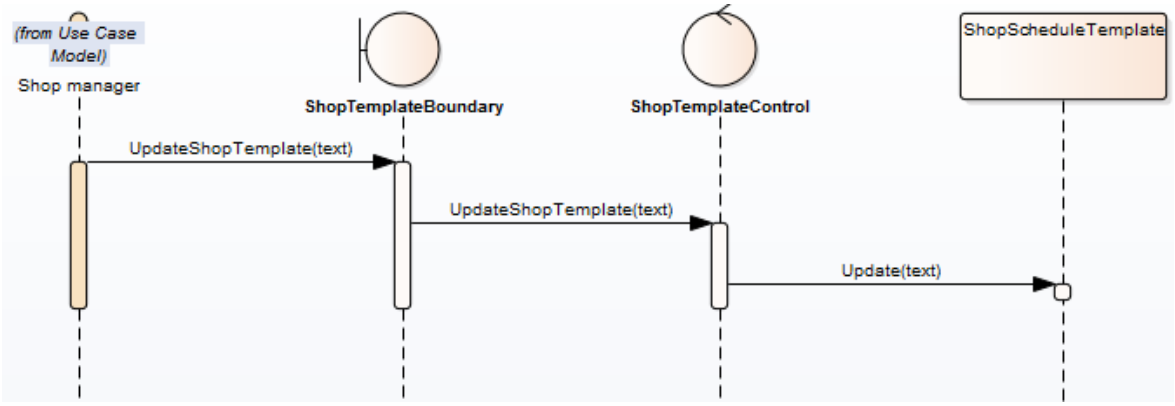
Maken winkelrooster template:



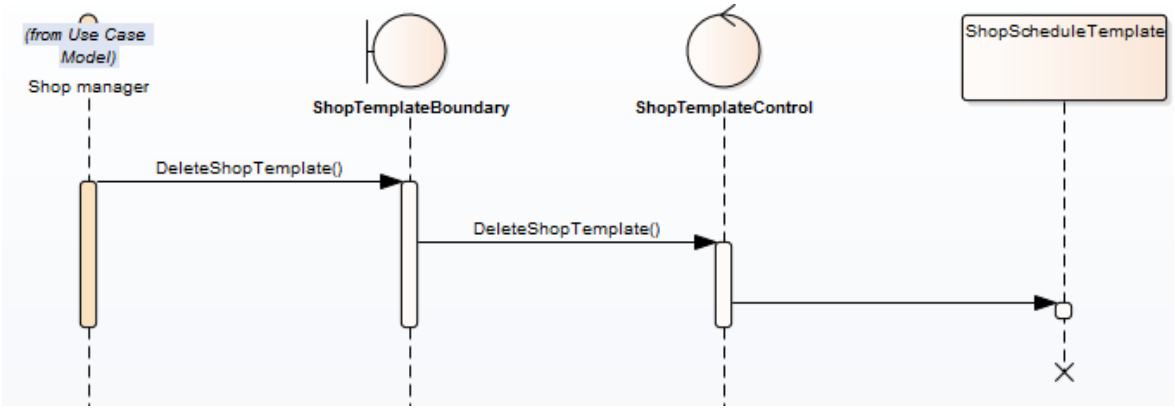
Zien winkelrooster template:



Updaten winkelrooster template:

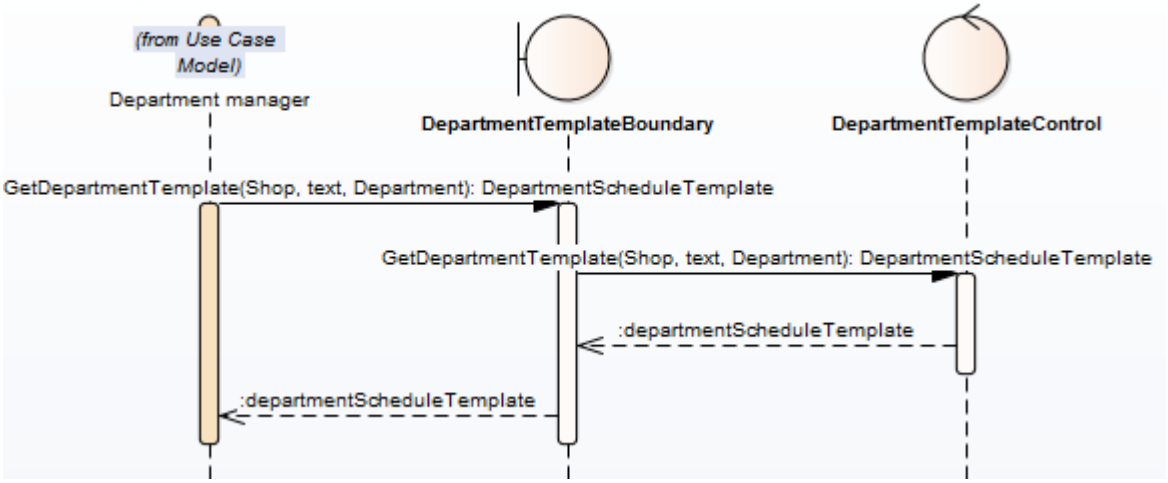


Verwijderen winkelrooster template:

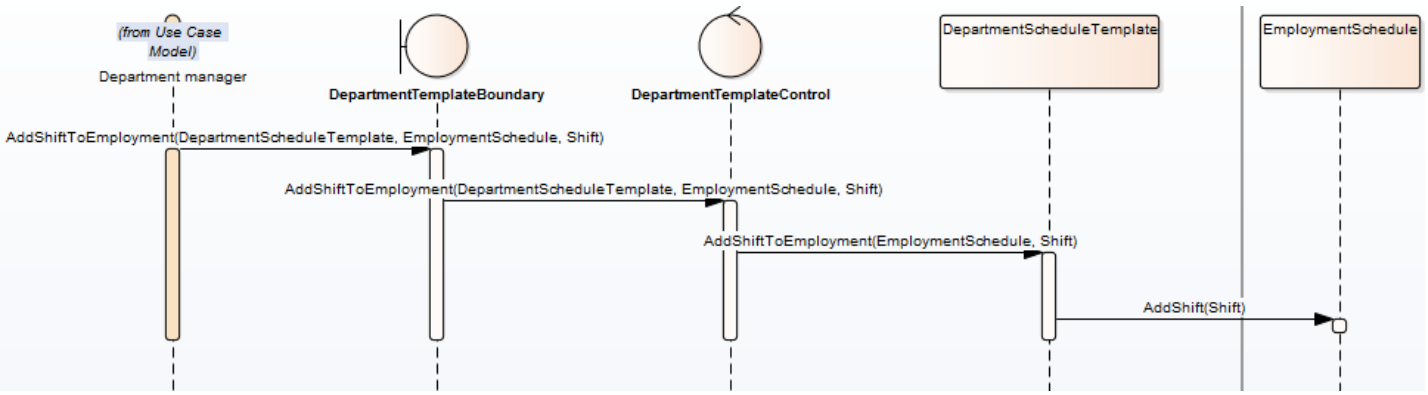


Beheren afdelingstemplates:

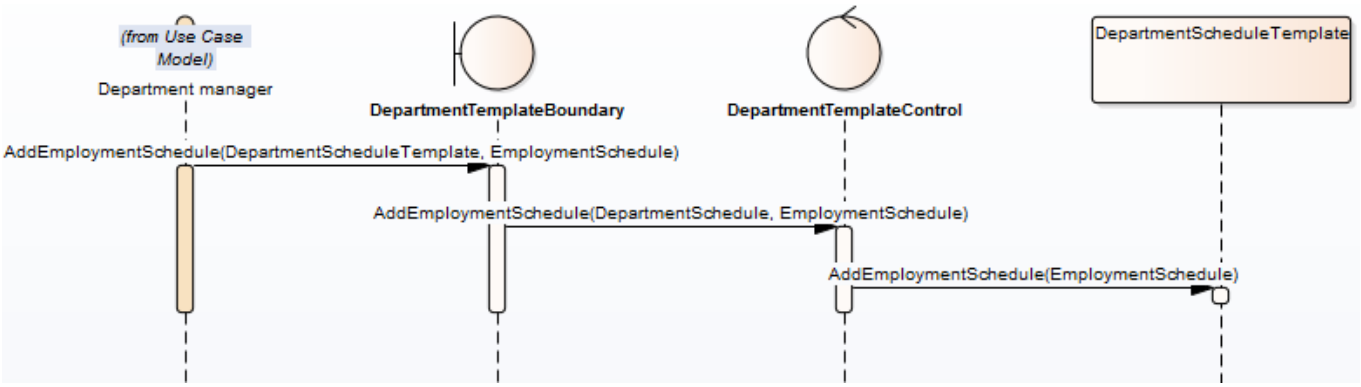
Zien afdelingsrooster:



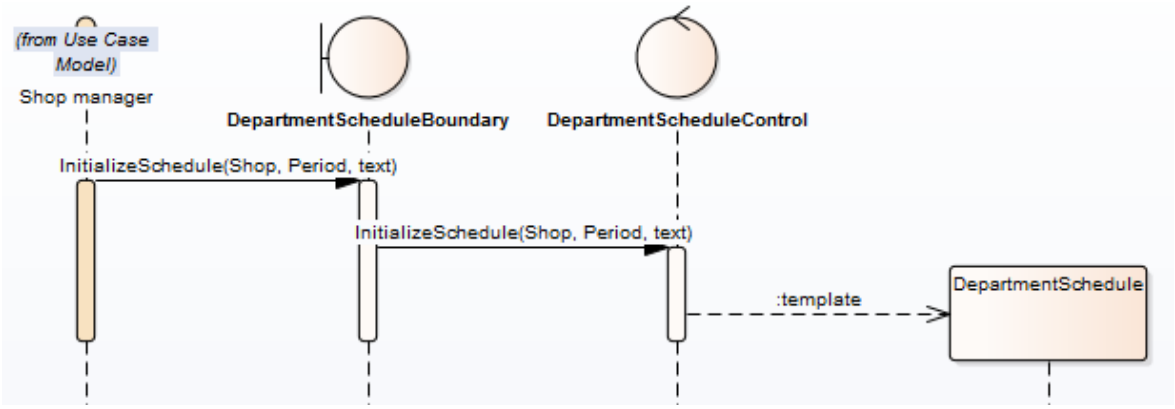
Updaten afdelingsrooster – toevoegen diensten:



Updaten afdelingsrooster – toevoegen externe medewerker:



Initialiseren rooster:



Bijlage E

Technisch ontwerp

Xavyr Rademaker

Inhoudsopgave

1. Inleiding

In dit document worden alle ontwerp beslissingen, welke zijn genomen gedurende het ontwikkelen van de proof of concept van R&R-web, toegelicht. De beslissingen worden per sprint behandeld, en tonen verschillende delen van de proof of concept.

De beslissingen worden weergeven door middel van sequentie –en klassendiagrammen. De beslissingen kunnen gaan over een van de volgende onderwerpen:

- Klassendiagrammen
    - Welke klasses/interfaces zijn er en welke methodes/properties hebben zij
    - Welke associaties hebben de klasses/interfaces met elkaar
- Sequence diagrammen
    - Hoe communiceren de klasses/objecten met elkaar
- Database diagrammen
    - Welke entiteiten worden opgeslagen, en welke relaties hebben deze entiteiten met elkaar?
    - Hoe zijn deze relaties geïmplementeerd in RavenDB(wat is embed, wat is reference)
- Toelichtingen op code/query's
    - Uitleg over wat bepaalde stukken code doen/wat de query's doen

2. Sprint 1

Dit hoofdstuk toont de beslissingen die genomen zijn in de eerste sprint. Hierbij moet gedacht worden aan:

- Klassendiagrammen
- Sequence diagrammen
- Database diagrammen
- Toelichtingen op code en de diagrammen

2.1 Backlogs

De volgende backlogs zijn in deze sprint behandeld:

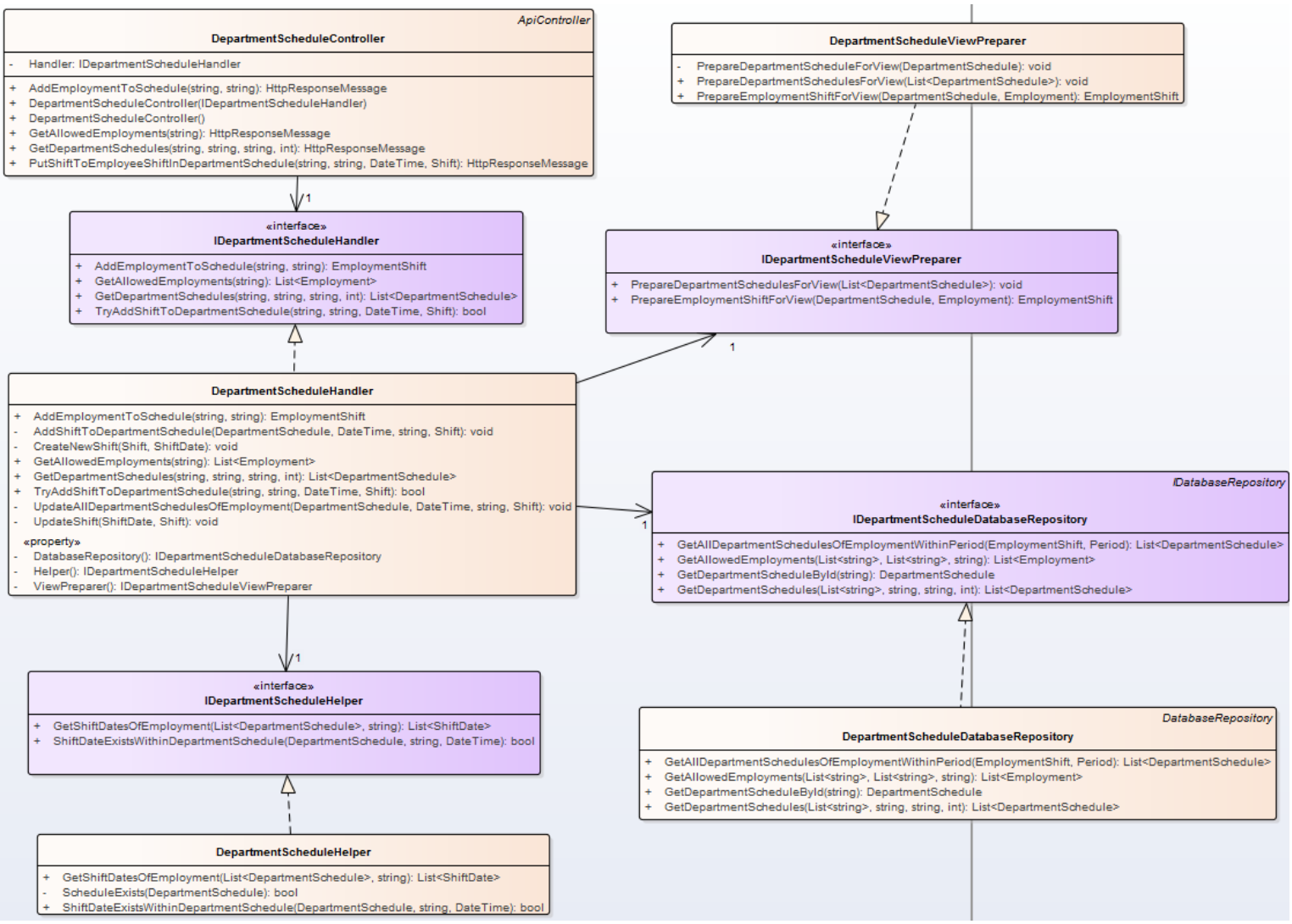
User story	Module	Story points
Als een afdelingsmanager wil ik diensten kunnen inzien die ik eerder ingevuld heb, zodat de medewerkers weten wanneer ze moeten werken en ik weet dat ik genoeg mensen heb die mijn werk kunnen doen.	Ro	8
Als een afdelingsmanager wil ik een afdelingsrooster kunnen maken zodat ik de benodigde mensen op de juiste tijd kan inplannen	Ro	6
Als een afdelingsmanager wil ik medewerkers van andere winkels kunnen inplannen op het moment dat ik zelf medewerkers te kort kom zodat ik genoeg medewerkers kan inplannen	Ro	10
Als afdelingsmanager wil ik geïnformeerd worden van bijzondere dagen zodat ik daar rekening mee kan houden bij het inplannen	Ro/P	4

2.2 Design

De volgende design diagrammen/beslissingen zijn er gemaakt in sprint 1.

2.2.1 Klassendiagrammen

De volgende design klassendiagrammen zijn er gemaakt in deze sprint:

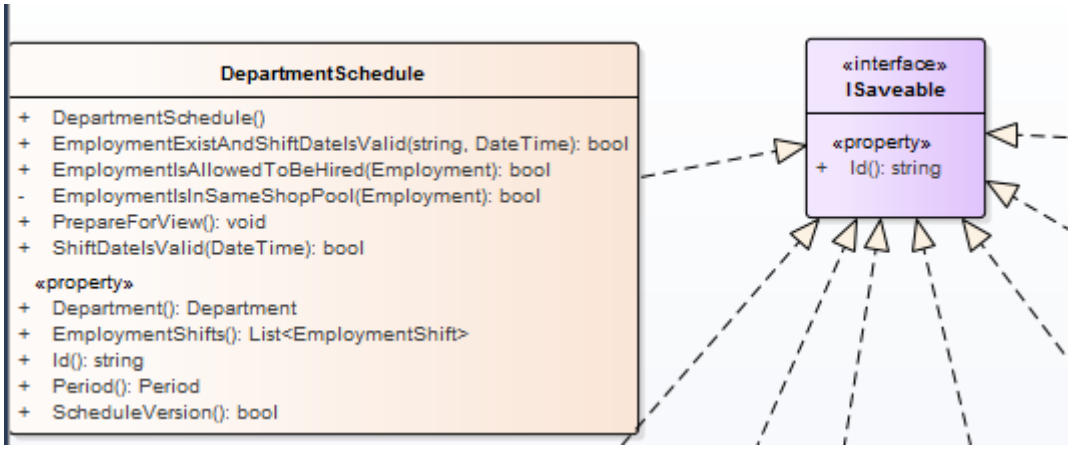


Een korte uitleg per klasse:

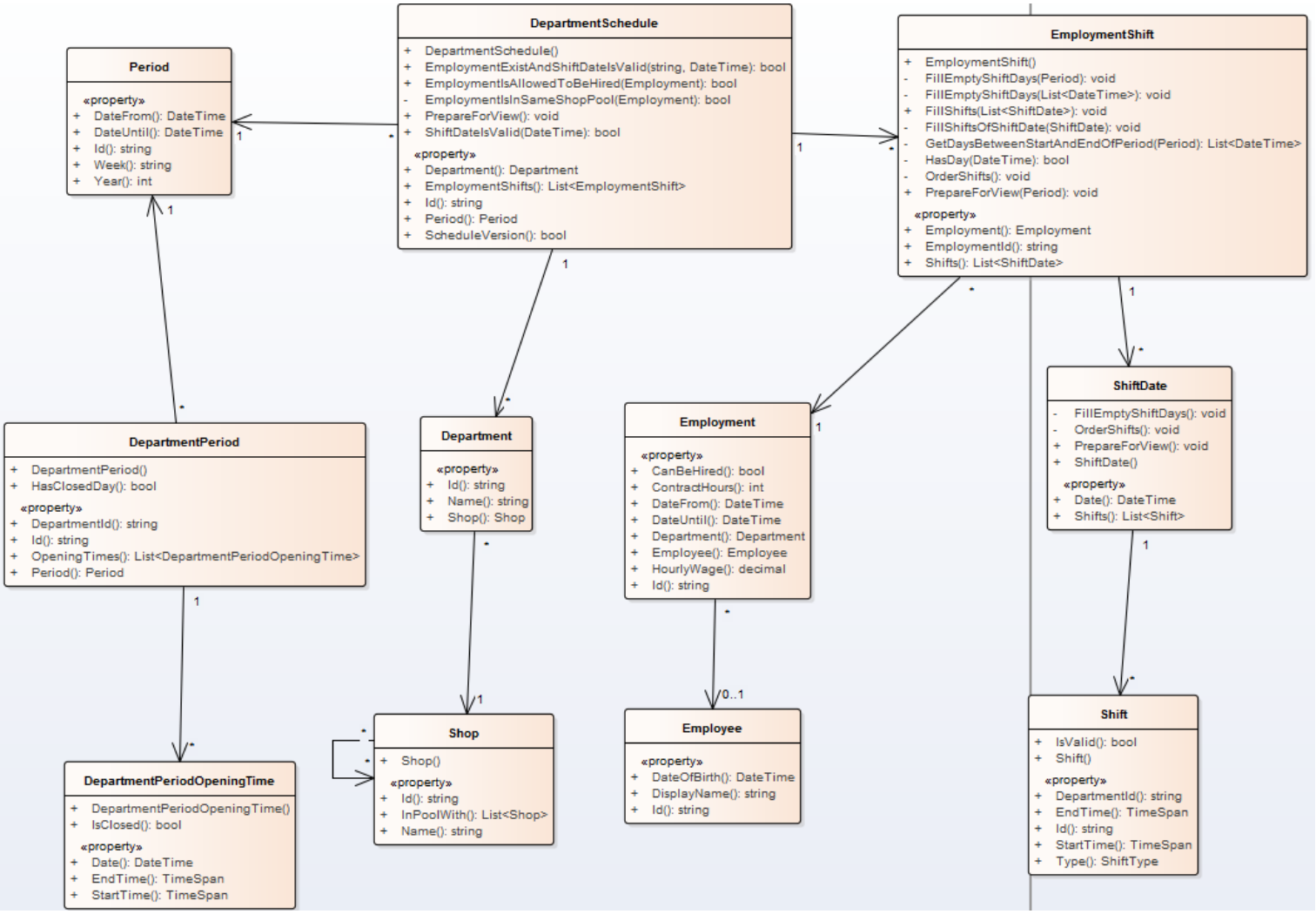
- DepartmentScheduleController is de WebApi Controller. Deze is verantwoordelijk voor het ontvangen van requests van de client, en het geven van responses op deze requests
- DepartmentScheduleHandler is verantwoordelijk voor het aansturen van alle andere klasses. De controller praat met de handler, en de handler stuurt als het ware de rest aan.
- DepartmentScheduleViewPreparer zorgt ervoor dat klassen die te maken hebben met departmentschedule zich klaarmaken voor de view. Dit kan inhouden het aanvullen van shifts, of het wijzigen van bepaalde properties.
- De DepartmentScheduleDatabaseRepository is verantwoordelijk voor de communicatie met de database
- De DepartmentScheduleHelper bevat enkel helper methodes, zoals het ophalen van id's uit een lijst.

Alle models die een eigen collection hebben in de database realiseren van de interface ISaveable. Deze interface bevat een property genaamd Id, van het type string. De reden dat hiervoor gekozen is, is dat klasses die worden opgeslagen in RavenDB een Id (unieke identifier) toegekend krijgen.

Dit Id ziet er als volgt uit: “{collectionname}/{uniekgetal}”. De {collectionname} is standaard: {naam van de klasse} + “s”. Om op dit Id te kunnen query’en, moet de property id dus wel aanwezig zijn. Dit interface dwingt dat af.



De models hadden onderling de volgende relaties:

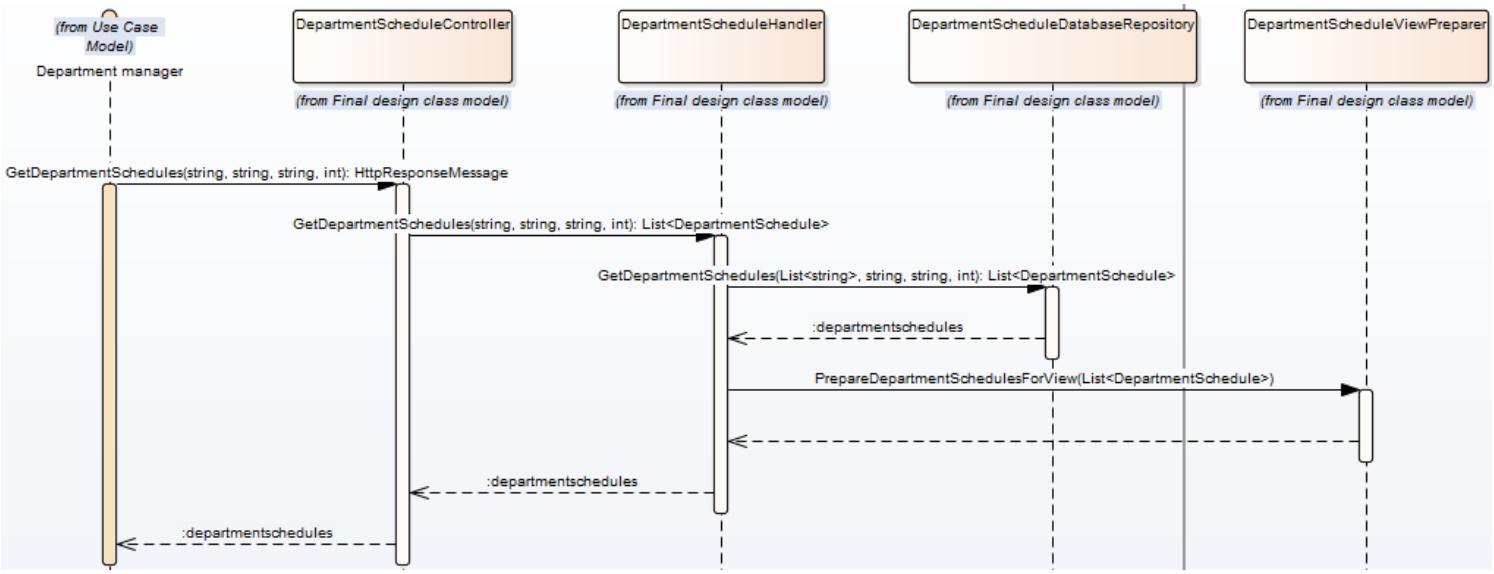


2.2.2 Sequentie diagrammen

De volgende sequentie diagrammen horen bij de backlogs:

Als een afdelingsmanager wil ik diensten kunnen inzien die ik eerder ingevuld heb, zodat de medewerkers weten wanneer ze moeten werken en ik weet dat ik genoeg mensen heb die mijn werk kunnen doen:

Voor deze functionaliteit moeten de afdelingsroosters uit de database gehaald worden. Als deze zijn opgehaald moeten deze voorbereid worden om te tonen op de client(in dit geval het sorteren van lijsten enz.). Tot slot worden de roosters naar de client gestuurd.

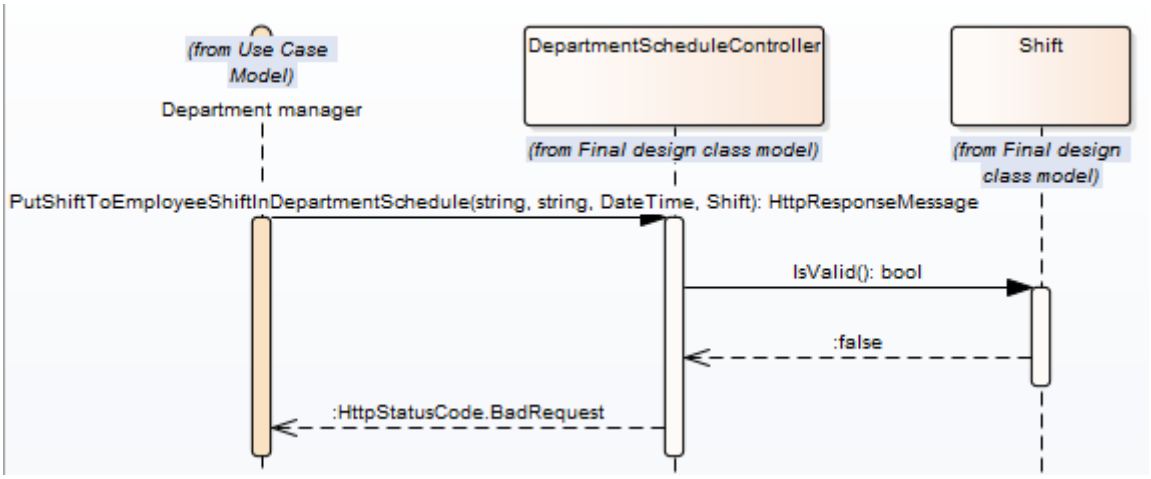


Als een afdelingsmanager wil ik een afdelingsrooster kunnen maken zodat ik de benodigde mensen op de juiste tijd kan inplannen:

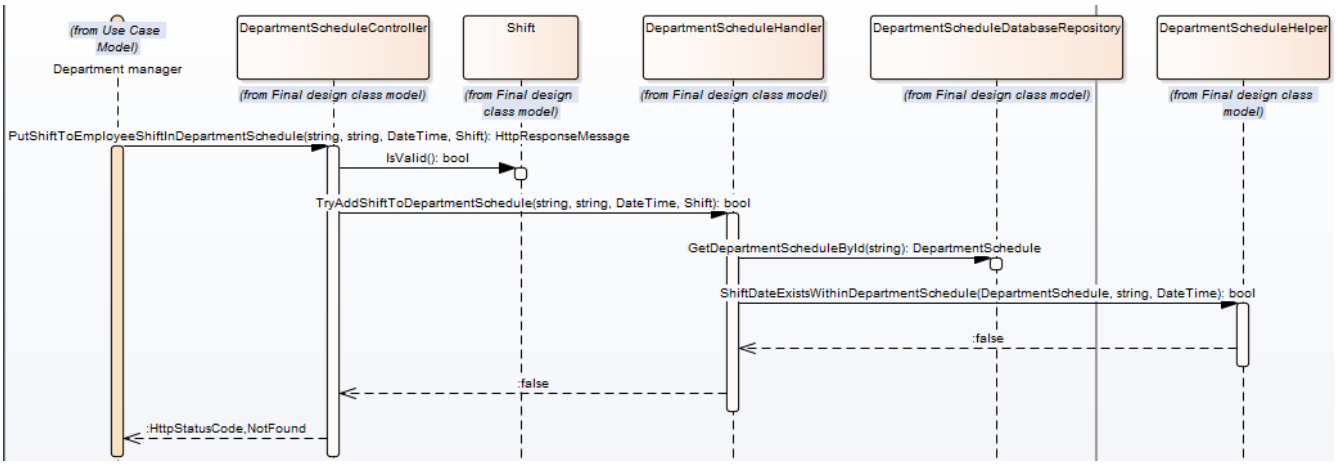
Er zijn bij deze functionaliteit 3 mogelijke flows, namelijk:

- De shift is niet valid(starttijd is na de eindtijd)
- De medewerker/het rooster/de dag binnen het rooster is niet gevonden binnen het rooster(kan alleen als data wordt gemanipuleerd onderweg naar de server)
- Alles gaat goed

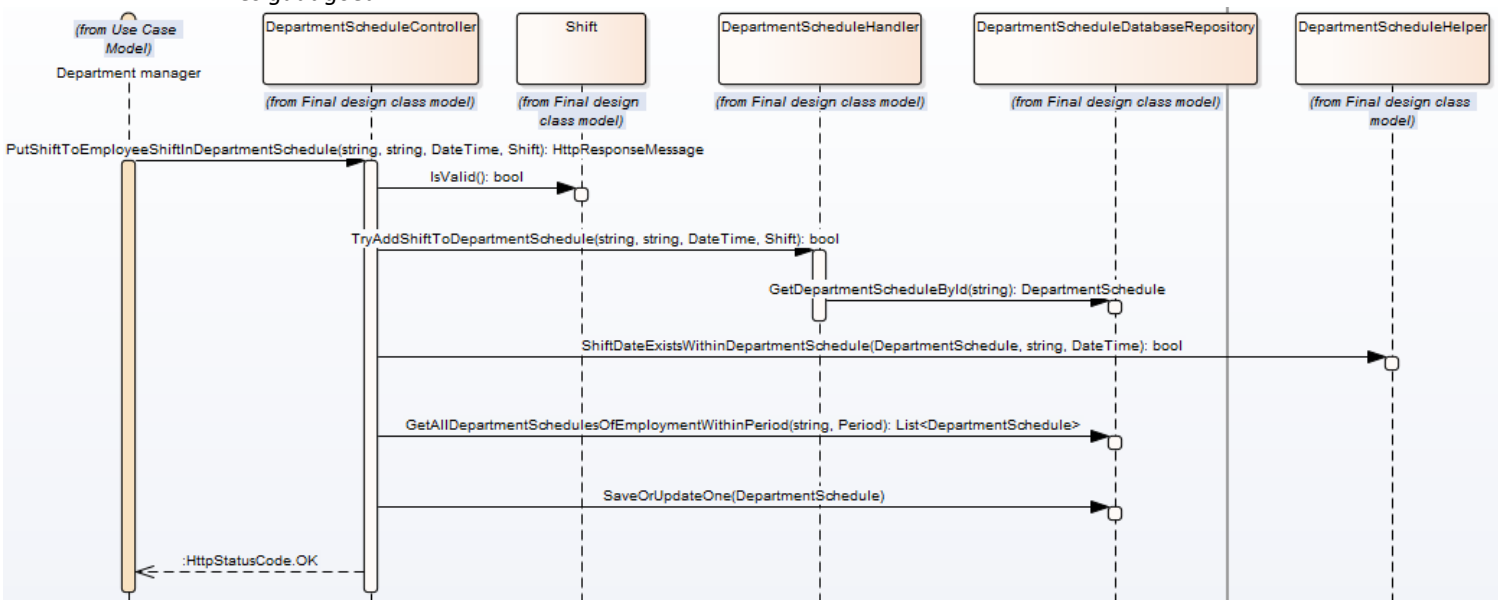
De volgende diagrammen horen bij deze flows:
Shift niet valide:



Medewerker niet gevonden binnen het rooster:



Alles gaat goed:

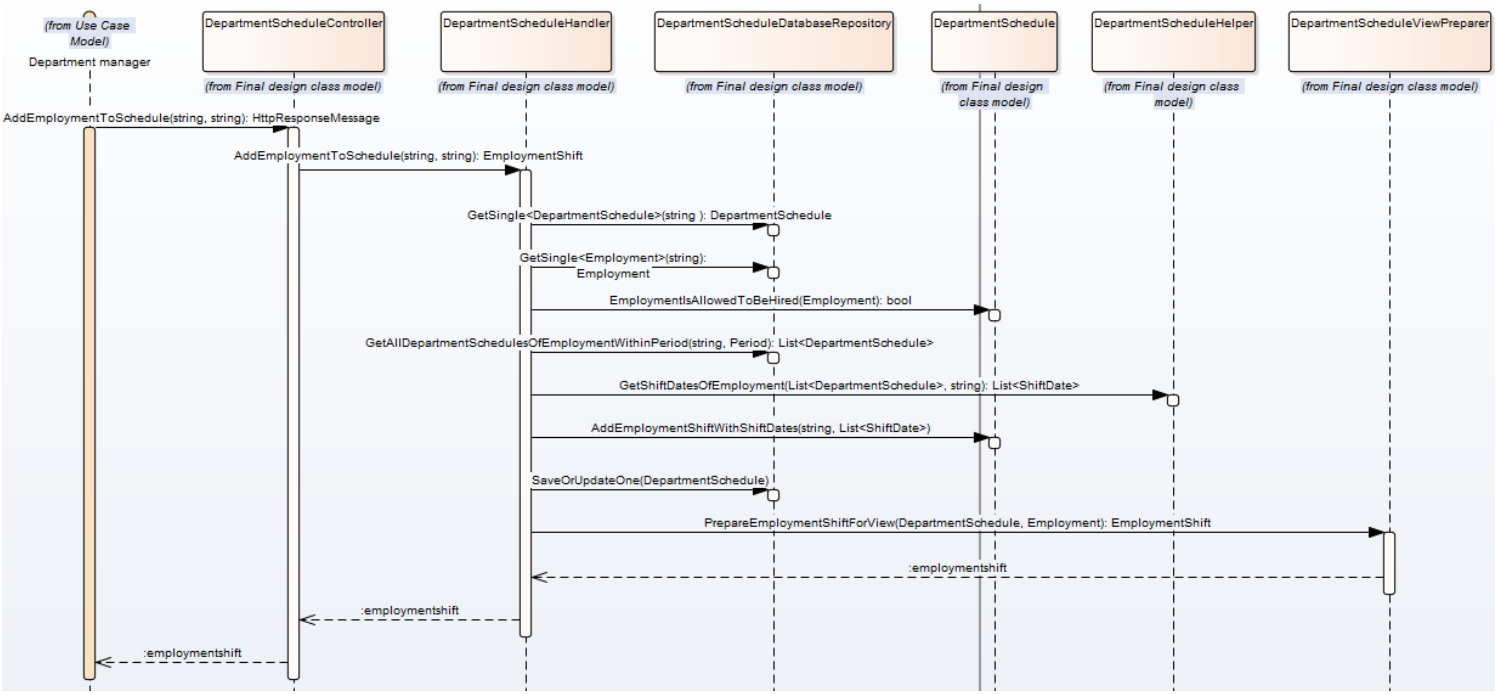


Als een afdelingsmanager wil ik medewerkers van andere winkels kunnen inplannen op het moment dat ik zelf medewerkers te kort kom zodat ik genoeg medewerkers kan inplannen:

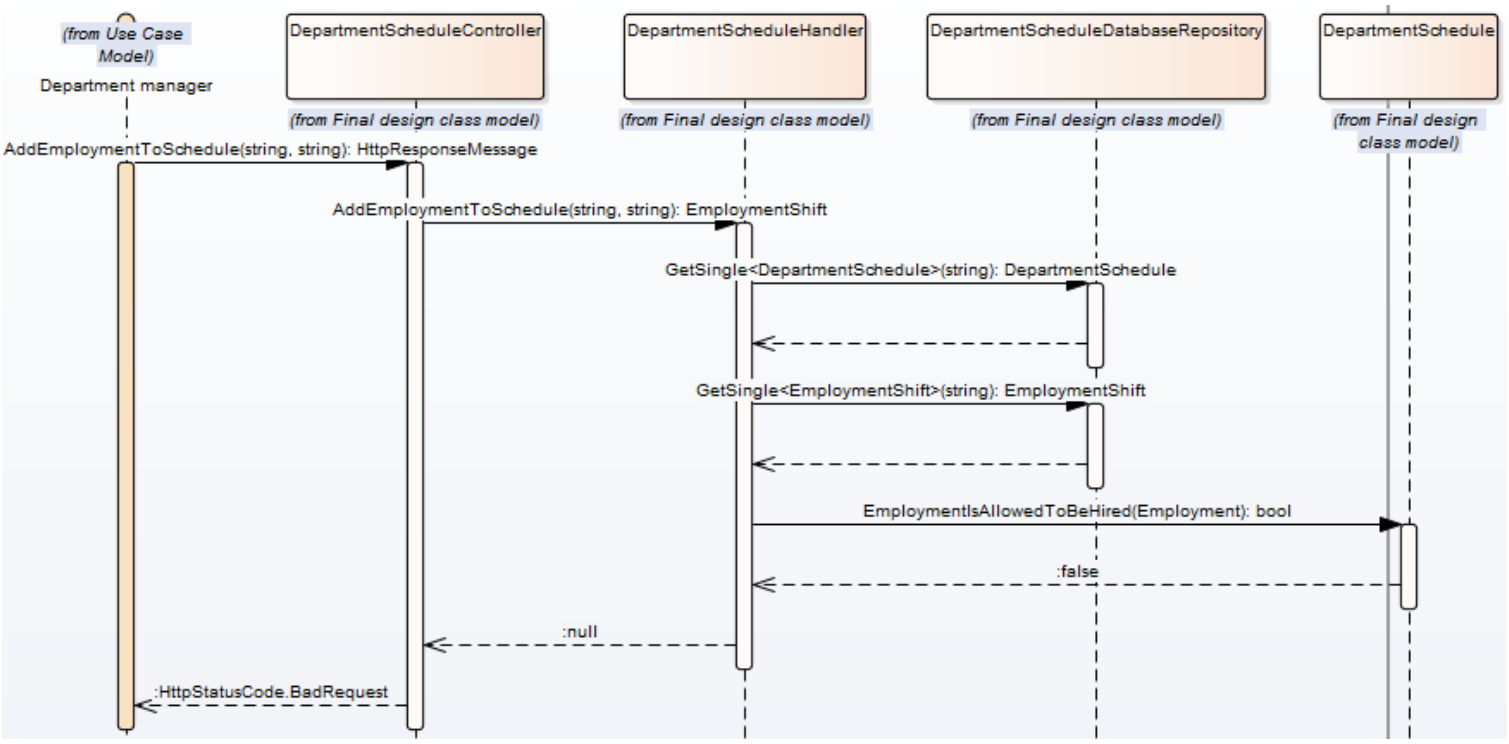
Voor deze backlog zijn er 2 flows, namelijk:

- Er wordt een medewerker toegevoegd aan het rooster
- De medewerker is niet toegestaan in het rooster(de data is onderweg naar de server gemanipuleerd).
-

Toegestaan:



Niet toegestaan:



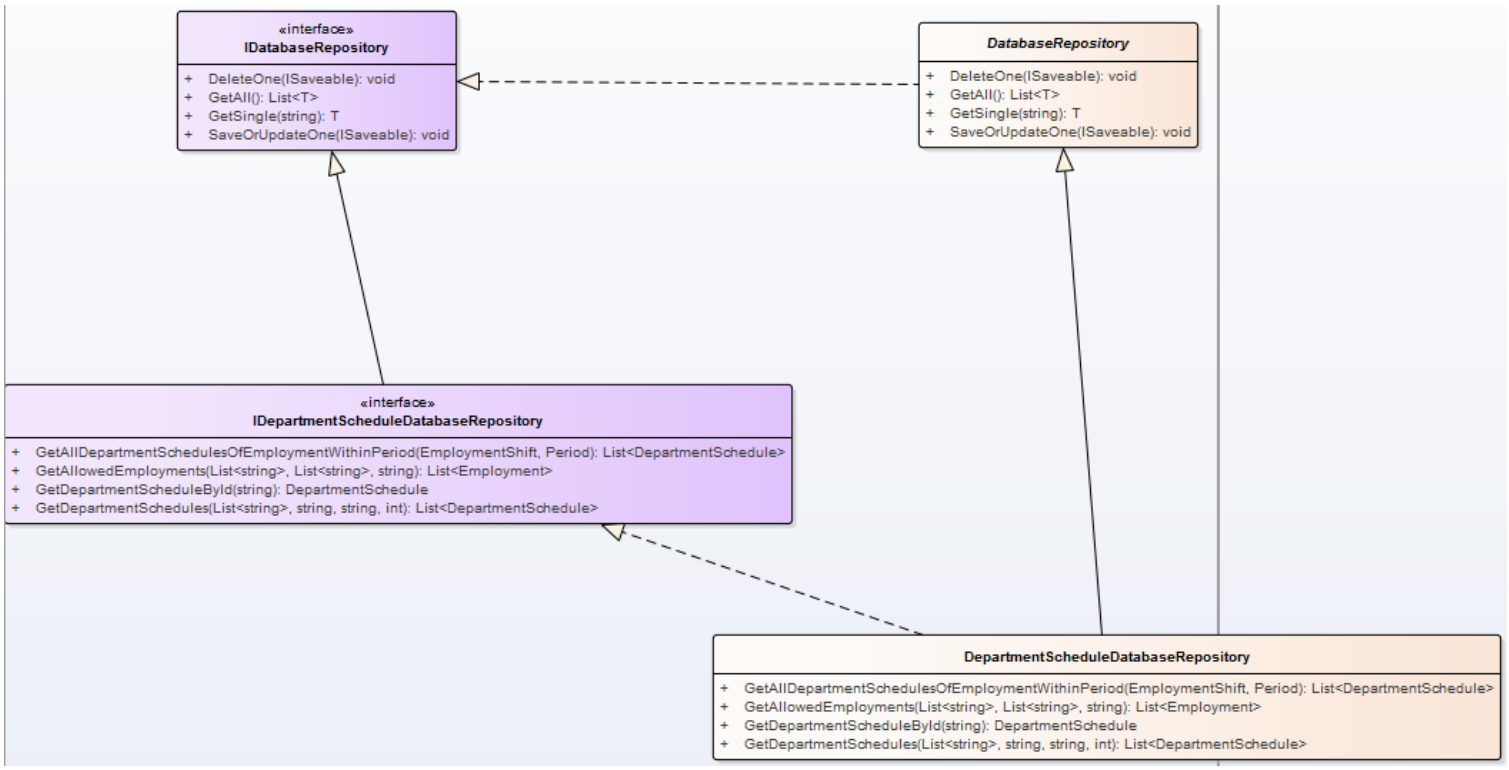
Als afdelingsmanager wil ik geïnformeerd worden van bijzondere dagen zodat ik daar rekening mee kan houden bij het inplannen:
De informatie over openingsdagen/tijden wordt meegenomen bij het ophalen van het afdelingsrooster(zie 1e backlog). Voor de feestdagen wordt een aparte request gedaan, en wordt op de client gekeken welke dagen als feestdagen gemarkeerd moeten worden.

2.2.3 Beslissingen

In deze sprint zijn de grootste design beslissingen gevallen in de datalaag en de web api controllers. De webapi controllers zijn al toegelicht in hoofdstuk 2.2.1, en zullen dus niet nogmaals toegelicht worden.

2.2.3.1 Datalaag

Voor de databaserepositories is er voor de volgende structuur gekozen:



Er is een interface welke de generieke methodes bevat(ophalen van alles, saven van alles enz.). Daarnaast is er nog een interface, welke specifieke methodes bevat(ophalen van afdelingsroosters met includes bijvoorbeeld, deze worden in paragraaf 2.4.1 besproken). Per WebApi controller is er 1 specifieke interface welke overerft van de generieke interface.

De realisaties van deze interfaces zijn eigenlijk precies hetzelfde. Er is 1 generieke repository, welke van de generieke interface realiseert. Deze generieke repository is wel abstract. Per specifieke interface is er ook 1 specifieke repository die de interface realiseert. De specifieke repository's erven weer van de generieke repository over, zodat zij de generieke methodes kunnen gebruiken.

2.3 Database

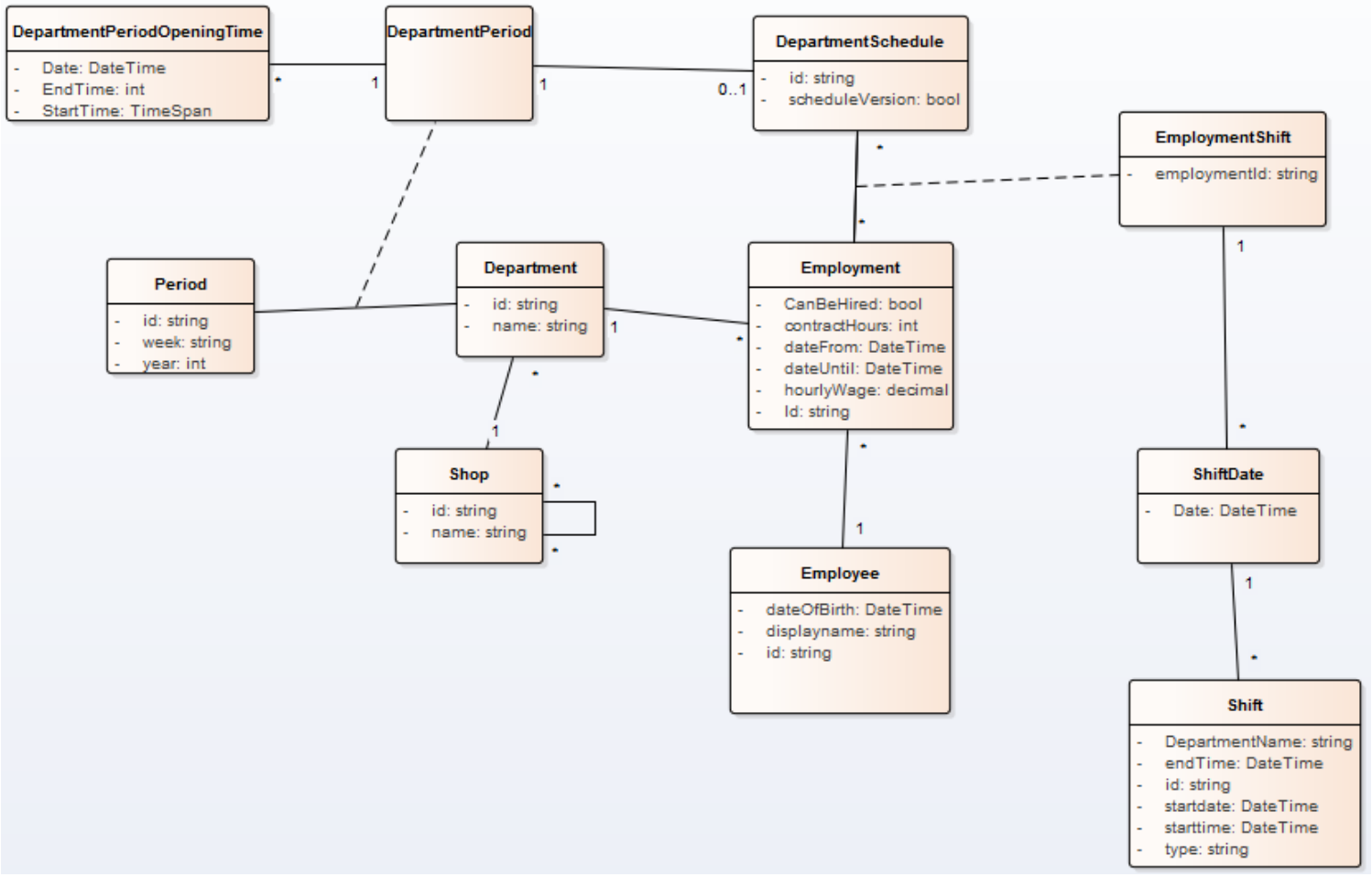
In ravenDB zijn er 3 manieren om relaties tussen objecten op te lossen, namelijk:

- Embedden
  - Het volledige document(object) in het andere document(object) opslaan
    - Oftewel Json in Json
- References
  - Het Id van een document bijhouden en aan de hand van het Id de documenten samen ophalen
    - Soort foreign key maar dan zonder constraints
- Denormalized reference
  - Een combinatie van de bovenste 2:
    - Je embed alleen de gegevens die je nodig hebt in het document en bewaard daarbij een reference naar het volledige object waar de gegevens vandaan komen

Zoals in het onderzoek gemeld, moet er bij het ontwerpen van je database een beslissing gemaakt worden voor welk van de 3 mogelijkheden je gaat.

2.3.1 Diagram

De database zag er bijna hetzelfde uit als de models, namelijk:

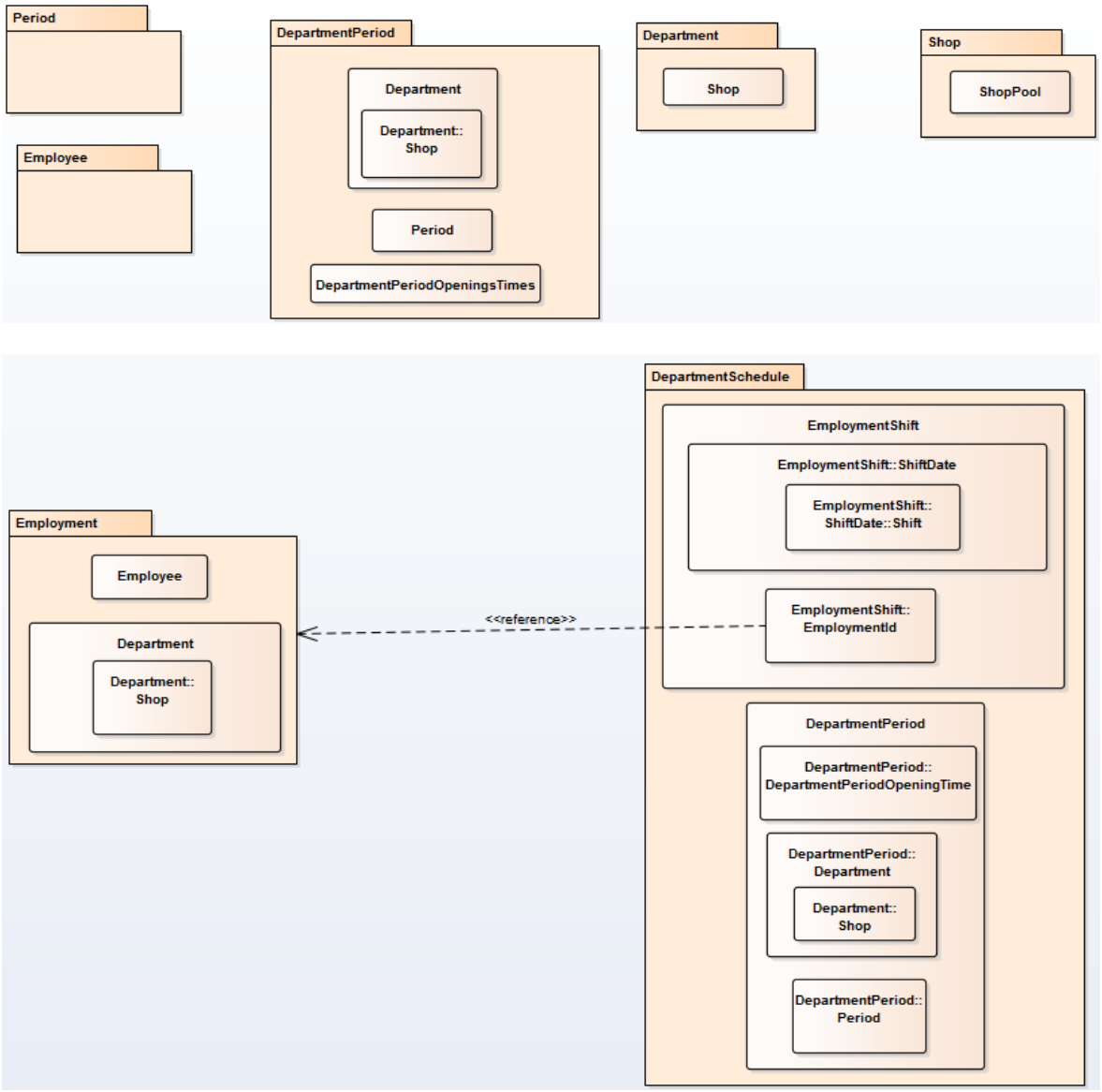


Bij het implementeren van deze structuur zijn een aantal beslissingen genomen.

2.3.2 Implementatie

In de volgende diagrammen is te zien hoe de entiteiten uit de vorige paragraaf zijn omgezet naar collections. De diagram moet als volgt gelezen worden:

- Alle packages zijn collections
- Alle klasses binnen een package zijn embedded documents
- Als er een dependancy lijn gebruikt wordt betekent het dat dit een reference is(staat ook aangegeven met <<reference>>).



Zoals te zien op de diagrammen is voor bijna elke relatie gekozen voor embedden. De reden dat hiervoor gekozen is, is dat veel van deze data vrij statische data is. De gegevens van de medewerker die worden bijgehouden(geboortedatum en naam) zijn niet echt gegevens die vaak wijzigen. Voor afdeling en winkel geld hetzelfde.

Hoewel de openingstijden vaker kunnen wijzigen, zal dit ook niet dagelijk gebeuren en gebeurd het op maximaal 2 plaatsen(in het rooster en in de collection departmentschedule).

Wat wellicht interessant is, is dat er voor de relatie van employmentshift naar employment wel voor een reference is gekozen. De reden dat hiervoor gekozen is, is dat de contractgegevens van de medewerker in employment worden bijgehouden. Daar een contract vaker kan wijzigen(ik heb begrepen dat het systeem sommige van deze gegevens wekelijks/maandelijks berekend). Is het niet verstandig om dit te embedden.

Hoewel wekelijks/maandelijks misschien niet als heel vaak klinkt is het zo, dat een medewerker per jaar op minimaal 52 roosters voorkomt. Meer als het een medewerker is die mag worden verhuurd. Als er maandelijks iets zou wijzigen, zou dit dus na 1 jaar op minimaal 53 plaatsen(52 roosters en eigen collection) gewijzigd moeten worden. Dit is niet praktisch.

2.4 Toelichtingen op code/query's

Om vanuit C# query's uit te voeren op een RavenDB database kan gebruik gemaakt worden van 2 methodes uit het session object, namelijk:

- Load
  - Wordt gebruikt om documents op te halen aan de hand van hun id
- Query
  - Wordt gebruikt om documents op te halen aan de hand van andere properties

De query methode verwacht de naam van een index als parameter. Op het moment dat je geen indexnaam opgeeft, zal hij zelf een index aanmaken bij het uitvoeren van de query. Bij het maken van de index hoort ook het indexeren van alle documents uit de collection.

Op het moment dat je geen indexnaam meegeeft en je hebt de query al eerder uitgevoerd zal RavenDB niet nogmaals dezelfde index aanmaken, hij gebruikt dan gewoon de eerder aangemaakte index.

Als de query methode gebruikt wordt, worden de query's in LINQ geschreven. In het volgende voorbeeld is te zien hoe het query'en op periodes van het type WEEK eruit zien:

```
3 references | 1/1 passing | Xavyr Rademaker, 10 days ago | 1 author, 1 change
public List<Period> GetNormalPeriods()
{
    using (var session = DatabaseConnection.OpenSession("RRWeb_POC"))
    {
        return session.Query<Period>()
            .Where(holiday => holiday.Type == Enums.PeriodType.WEEK).ToList();
    }
}
```

2.4.1 Includes

Om de references naar employment op te halen bij het tonen van een afdelingsrooster kunnen includes gebruikt worden. De include methode haalt eigenlijk alle verwezen objecten op samen met de oorspronkelijke query. Dit zorgt ervoor dat er maar 1 query naar de database nodig is, om alle data op te halen.

De employments moeten op de client(C#) nog wel toegevoegd worden aan het juiste afdelingsrooster. In de onderstaande 2 stukken code is te zien hoe de includes gebruikt worden.

```
8 references | 0/6 passing | Xavyr Rademaker, 1 day ago | 1 author, 4 changes
public DepartmentSchedule GetDepartmentScheduleById(string id)
{
    using (var session = DatabaseConnection.OpenSession("RRWeb_POC"))
    {
        DepartmentSchedule returnSchedule = session
        .Include<DepartmentSchedule>(deptSchedule => deptSchedule.EmploymentShifts.Select(empshift => empshift.EmploymentId))
        .Load(id);

            foreach (EmploymentShift shift in returnSchedule.EmploymentShifts)
            {
                shift.Employment = session.Load<Employment>(shift.EmploymentId);
            }
        int i = session.Advanced.NumberOfRequests;

        return returnSchedule;
    }
}
```

In de bovenstaande afbeelding wordt de include gebruikt op de Load methode(load wordt gebruikt om 1 document op te halen aan de hand van een id). De include methode verwacht overigens 1 of een list van strings in het volgende formaat: “{{collectionnaam}}/{{idnummer}}. Dit is ook het standaard formaat waarop ravenDB id’s toekent aan documenten.

In de onderstaande afbeelding is een ander voorbeeld van includes te zien.

```
public List<DepartmentSchedule> GetDepartmentSchedules(List<string> departmentNames, string shopName, string week, int year)
{
    using(var session = DatabaseConnection.OpenSession("RRWeb_POC"))
    {
        List<DepartmentSchedule> departmentschedules = session
            .Query<DepartmentSchedule>("DepartmentSchedules/ByPeriod_YearAndPeriod_DescriptionAndDepartmentNameAndShopName")
            .Customize(schedule => schedule.Include<DepartmentSchedule>(deptSchedule => deptSchedule.EmploymentShifts.Select(empShift => empShift.EmploymentId)))
            .Where(departmentSchedule => departmentSchedule.DepartmentPeriod.Department.Name.In(departmentNames)
                && (departmentSchedule.DepartmentPeriod.Department.Shop.Name == shopName)
                && (departmentSchedule.DepartmentPeriod.Period.Description == week)
                && (departmentSchedule.DepartmentPeriod.Period.Year == year))
            .ToList();

        foreach (var returnSchedule in departmentschedules) {
            foreach (EmploymentShift shift in returnSchedule.EmploymentShifts)
            {
                shift.Employment = session.Load<Employment>(shift.EmploymentId);
            }
        }

        return departmentschedules;
    }
}
```

Op het moment dat je een document wilt ophalen aan de hand van iets anders dan zijn Id, wordt de query methode gebruikt. Query ondersteund echter niet direct de methode .Include(). Vandaar dat eerst .Customize moet worden aangeroepen, en binnen customize moet de include methode weer aangeroepen worden. Het doet functioneel overigens nog steeds hetzelfde, het haalt alle opgevraagde data naar de client(naar het session object om precies te zijn).

2.4.2 JsonIgnore

Als je in een model properties hebt staan, die je niet in de database wilt opslaan kan gebruik gemaakt worden van het JsonIgnore attribuut. Deze ziet er als volgt uit:

```
[JsonIgnore]
4 references | Xavyr Rademaker, 12 days ago | 1 author, 2 changes
public Employment Employment { get; set; }
```

Let er bij dit attribuut wel op dat je de JsonIgnore gebruikt uit de namespace:

`Raven.Imports.Newtonsoft.Json` (dus: `using Raven.Imports.Newtonsoft.Json;`)

Op het moment dat je de JsonIgnore attribuut uit namespace `Newtonsoft.Json` gebruikt bestaat er een kans dat de webapi controller deze property ook zal negeren, daar deze dezelfde serializer gebruikt.

2.4.3 In()

Naast includes is er nog een LINQ methode die gebruikt kan worden bij het query'en. Deze methode is de In() methode. Deze methode kan gebruikt worden als je wilt weten of een string voorkomt in een lijst met strings. In het volgende voorbeeld wordt bijvoorbeeld gezocht op medewerkers waarvan de winkelID voorkomt in een list van strings genaamd shopIdPool(zie rood omringd stuk)

```
return session.Query<Employment>("Employments/ByCanBeHiredAndShopIdPool")
    .Where(employment => employment.CanBeHired == true
    && (employment.Department.Shop.Id.In(shopIdPool) || employment.Department.Shop.Id == shopFromId)
    && !employment.Id.In(employmentShifts)).ToList();
```

2.4.4 Shift validatie

In het onderzoek is naar voren gekomen dat constraints en functionaliteiten uit de database naar de applicatie verplaatst moeten worden. Een voorbeeld hiervan is de shift(een dienst). De starttijd van een toegevoegde/gewijzigde shift moet voor de eindtijd van deze shift zijn.

Allereerst heb ik validatie op de front-end geplaatst(angularjs controleert of de tijden een tijdformaat hebben en of de starttijd voor de eindtijd is). Op het moment dat deze data niet valide is, wordt er geen request naar de server gedaan om de wijzigingen op te slaan.

Op het moment dat deze data wel valide is wordt er een request naar de server gedaan. In deze request worden de tijden weer gevalideerd. Op server niveau ziet dat er als volgt uit:

```
1 reference | Xavyr Rademaker, 12 days ago | 1 author, 2 changes
public bool IsValid()
{
    return this.StartTime < this.EndTime;
}
```

Deze methode zit overigens in de shift klasse zelf.

3. Sprint 2

3.1 Backlogs

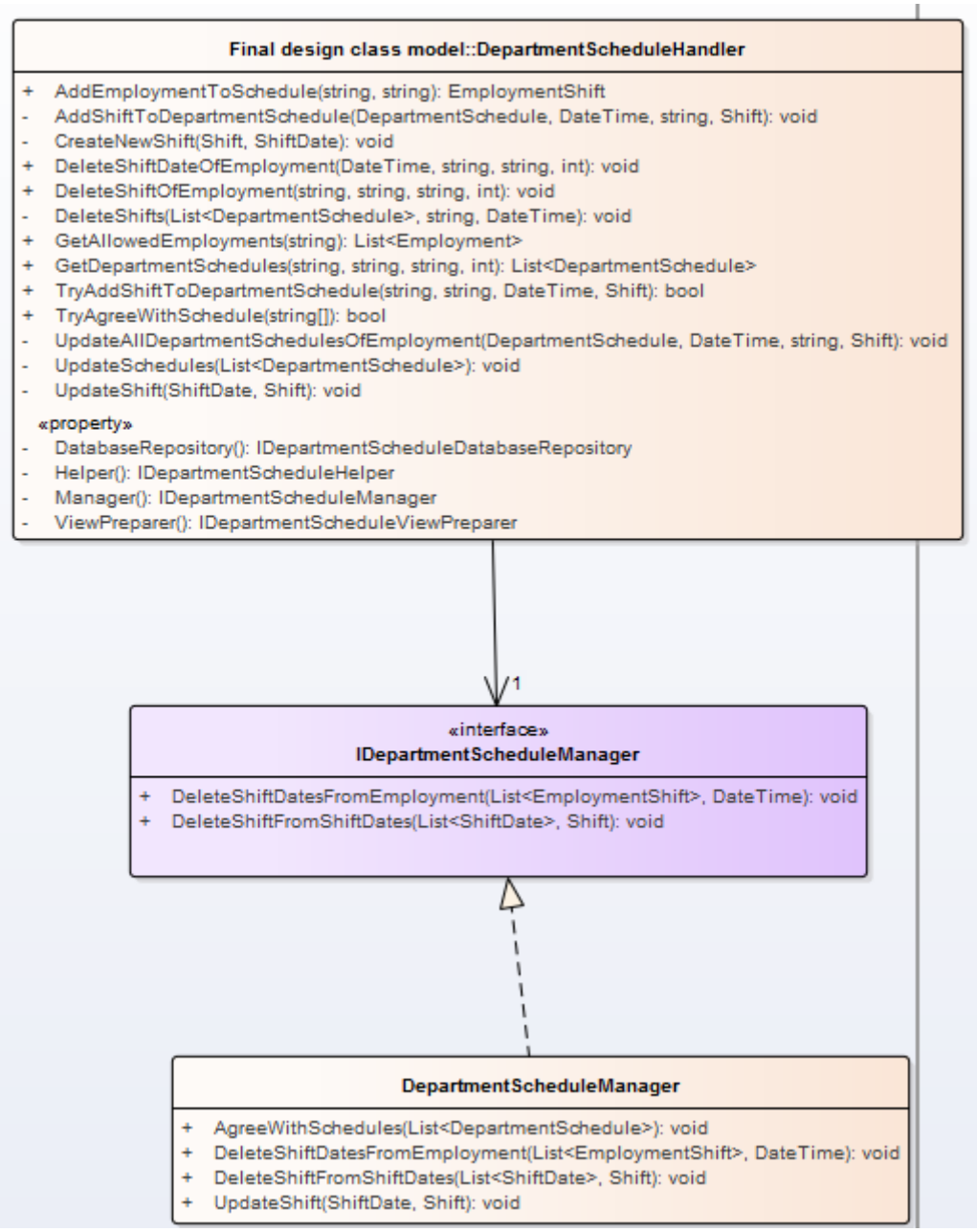
User story	Module	Story points
Als afdelingsmanager wil ik diensten kunnen verwijderen zodat ik optimaal kan roosteren	Ro	4
Als afdelingsmanager wil ik kunnen zien of dat ik genoeg mensen heb ingepland op een specifiek tijdstip op basis van prognose zodat ik nog mensen kan inhuren als ik tekort kom/minder mensen kan plannen als ik er genoeg heb	Ro/P	15
Als gebruiker wil ik kunnen inloggen en de juiste rechten krijgen zodat ik mijn functie kan vervullen.	Auth	2
Als afdelingsmanager wil ik een akkoord kunnen geven op het rooster zodat de winkelmanager weet dat ik klaar ben met roosteren	Ro	1

3.2 Design

3.2.1 Klassendiagrammen

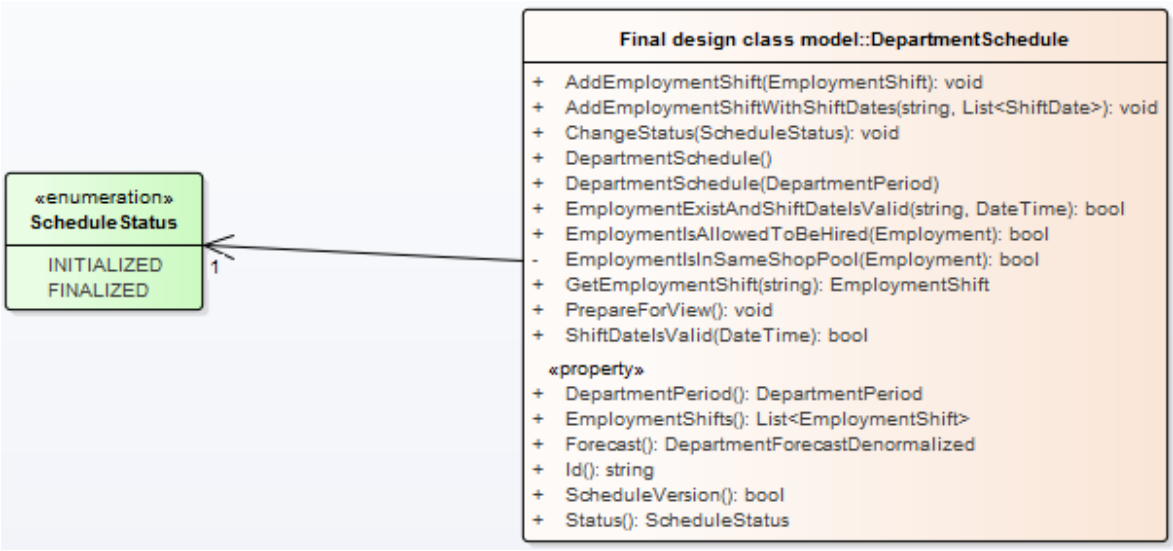
De volgende wijzigingen/aanvullingen zijn er gedurende deze sprint op het klassendiagram gekomen:

Handler nieuwe relatie:
De handler heeft een nieuwe associatie, naar een manager klasse. Deze is te zien op de volgende diagram:



De verantwoordelijkheid van de manager is in-memory CRUD acties(voornamelijk U en D) acties uitvoeren op een afdelingsrooster.

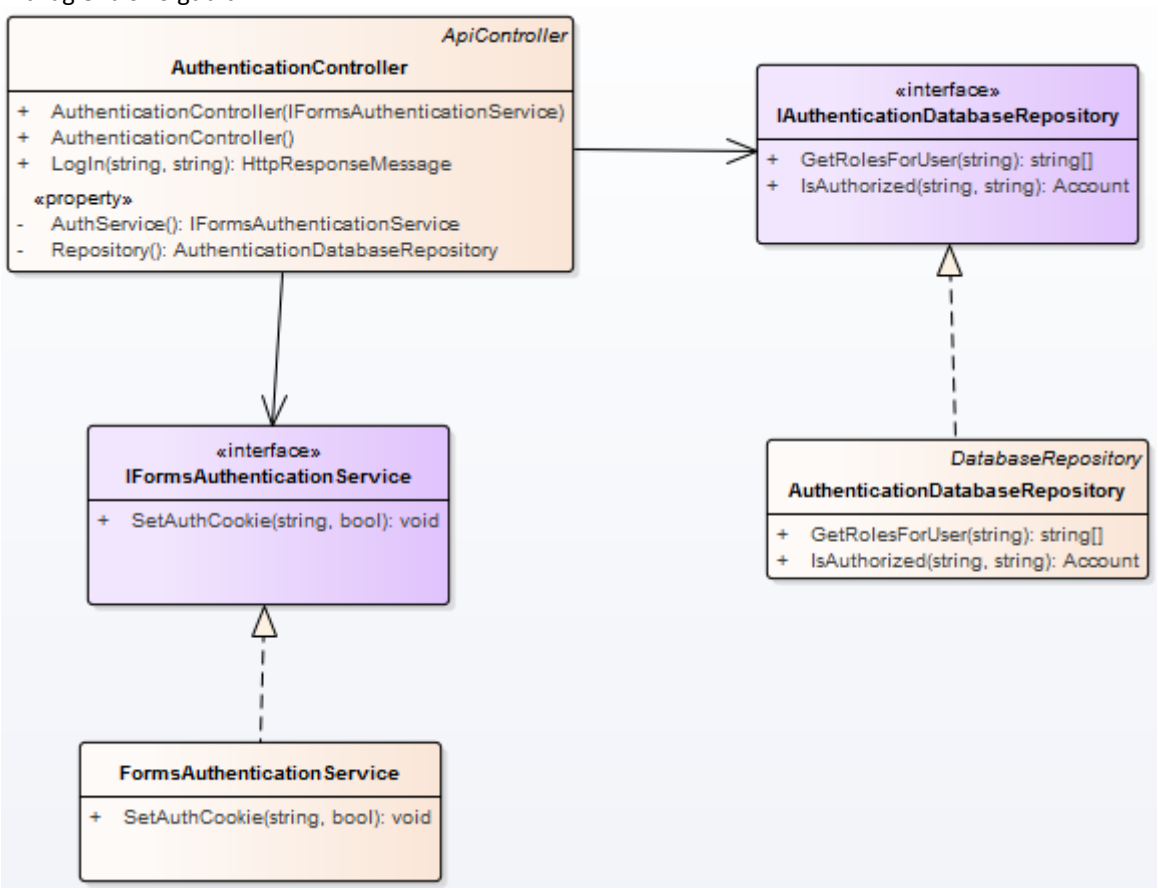
Nieuwe attributen/associatie departmentSchedule:



Om akkoord te kunnen gaan met het rooster, heeft het rooster een status gekregen. Deze status is een enum. De reden dat er voor een enum is gekozen, is om af te dwingen dat er alleen een select aantal waardes is toegestaan.

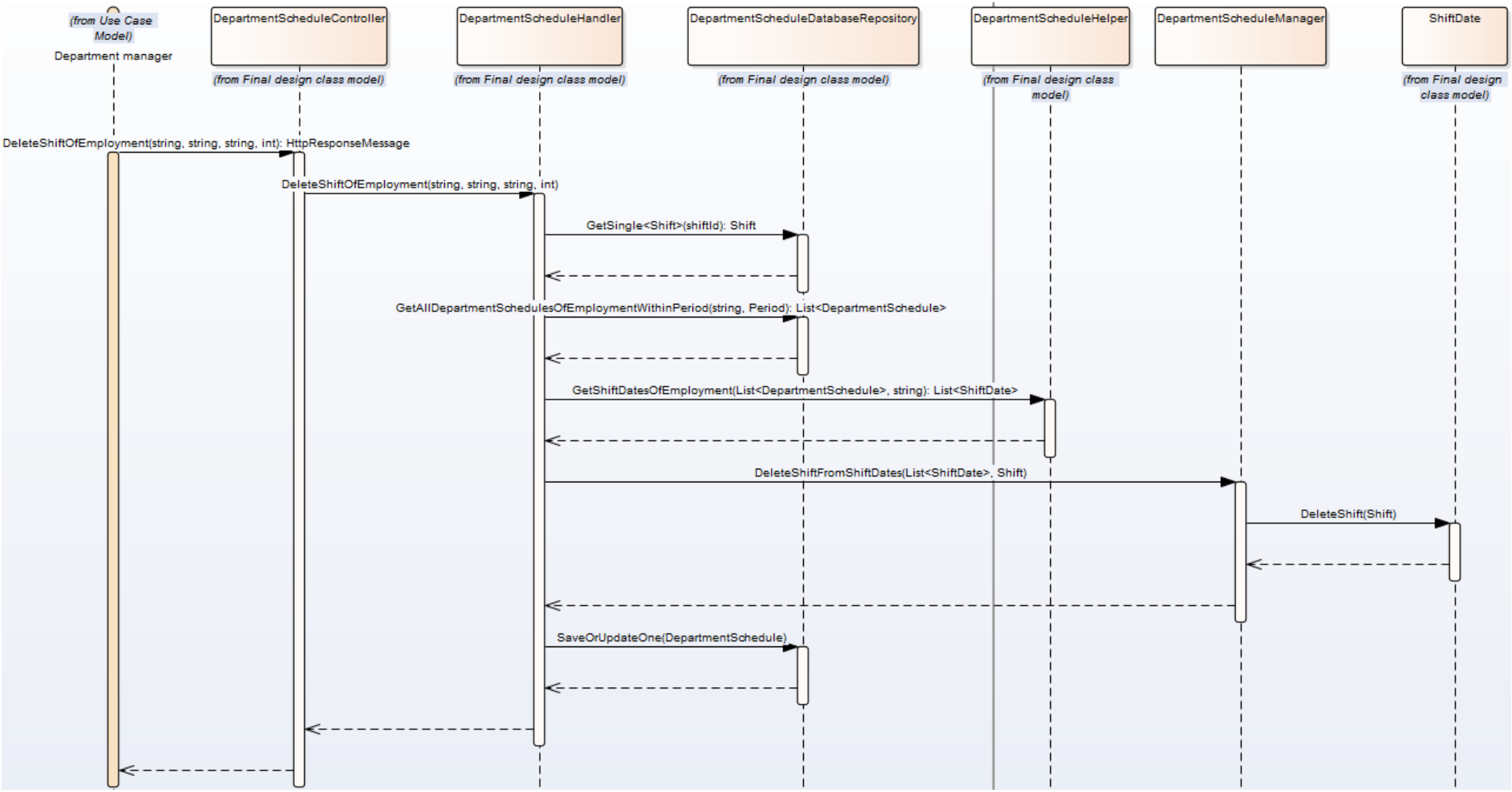
Login account en rollen:
Voor het inloggen is een authorisatie database repository, een authorisatie controller en een authorisatie service gemaakt. De service is overigens alleen gemaakt omdat ik anders de statische methode FormsAuthentication.SetAuthCookie() niet kon mocken.

Dit zag er als volgt uit:



3.2.2 Sequentiediagrammen

Als afdelingsmanager wil ik diensten kunnen verwijderen zodat ik optimaal kan roosteren:


(from Use Case Model)
Department manager
DepartmentScheduleController
(from Final design class model)
DepartmentScheduleHandler
(from Final design class model)
DepartmentScheduleDatabaseRepository
(from Final design class model)
DepartmentScheduleHelper
(from Final design class model)
DepartmentScheduleManager
ShiftDate
(from Final design class model)
DeleteShiftOfEmployment(string, string, string, int): HttpResponseMessage
DeleteShiftOfEmployment(string, string, string, int)
GetSingle<Shift>(shiftId): Shift
GetAllDepartmentSchedulesOfEmploymentWithinPeriod(string, Period): List<DepartmentSchedule>
GetShiftDatesOfEmployment(List<DepartmentSchedule>, string): List<ShiftDate>
DeleteShiftFromShiftDates(List<ShiftDate>, Shift)
DeleteShift(Shift)
SaveOrUpdateOne(DepartmentSchedule)

Als afdelingsmanager wil ik kunnen zien of dat ik genoeg mensen heb ingepland op een specifiek tijdstip op basis van prognose zodat ik nog mensen kan inhuren als ik tekort kom/minder mensen kan plannen als ik er genoeg heb:

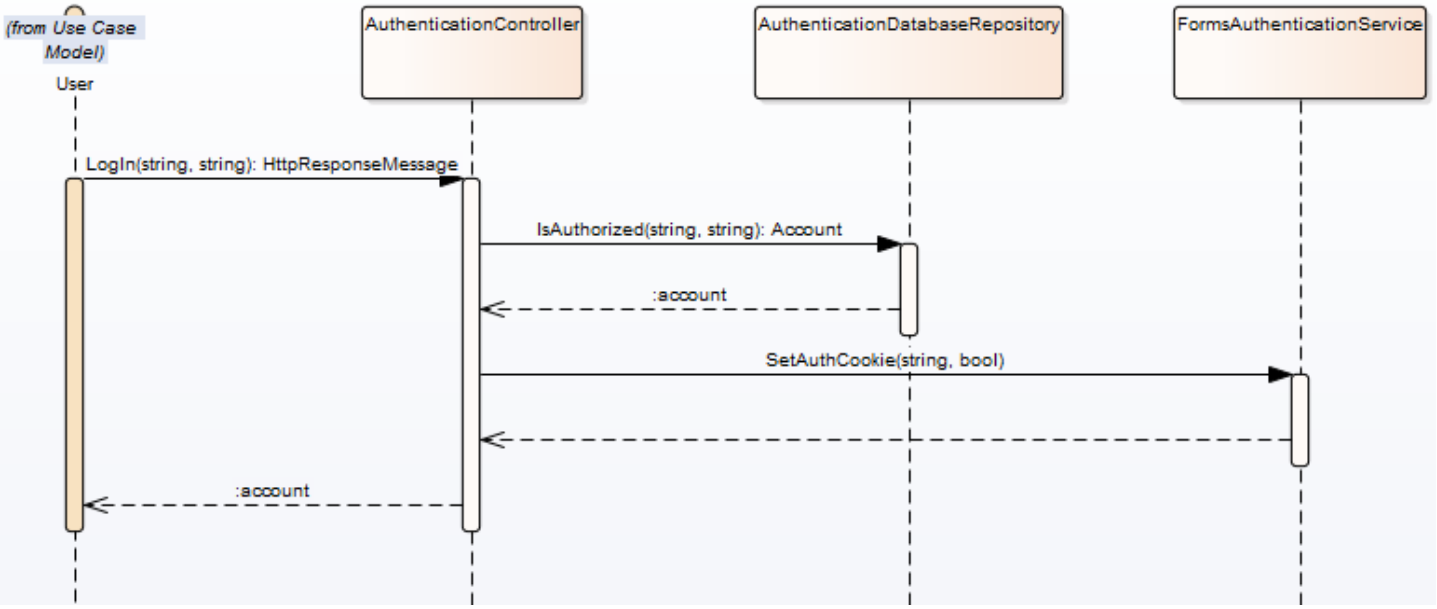
Vanwege de manier van opslaan kwam de prognose data gelijk mee bij het ophalen van het rooster(zie 1e backlog sprint 1).

Als gebruiker wil ik kunnen inloggen en de juiste rechten krijgen zodat ik mijn functie kan vervullen:

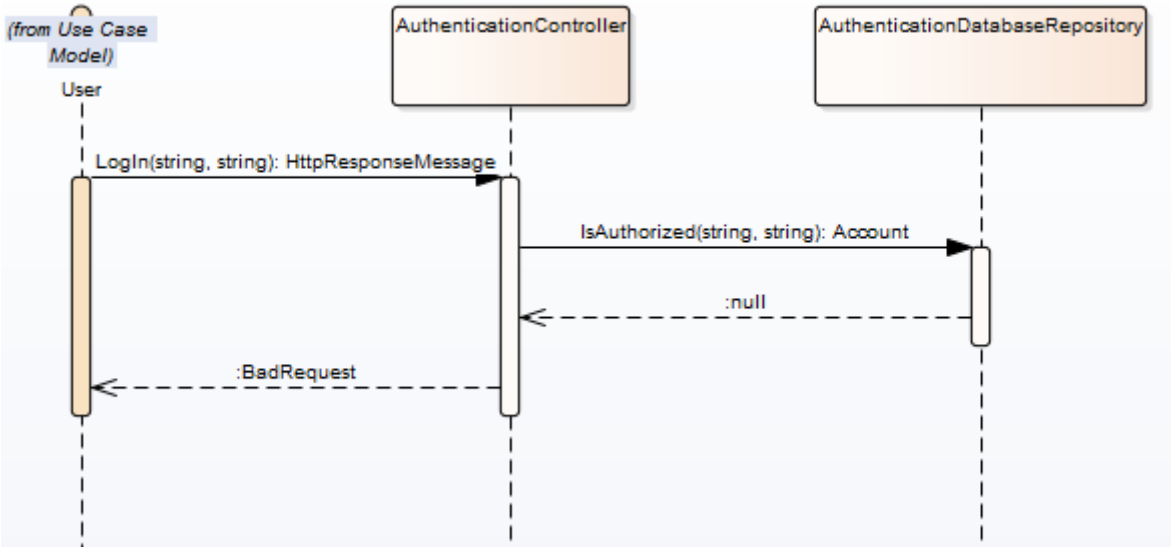
Er zijn 2 flows bij deze backlog:

- Login succesvol
- Login niet succesvol

Succes:



Geen succes:

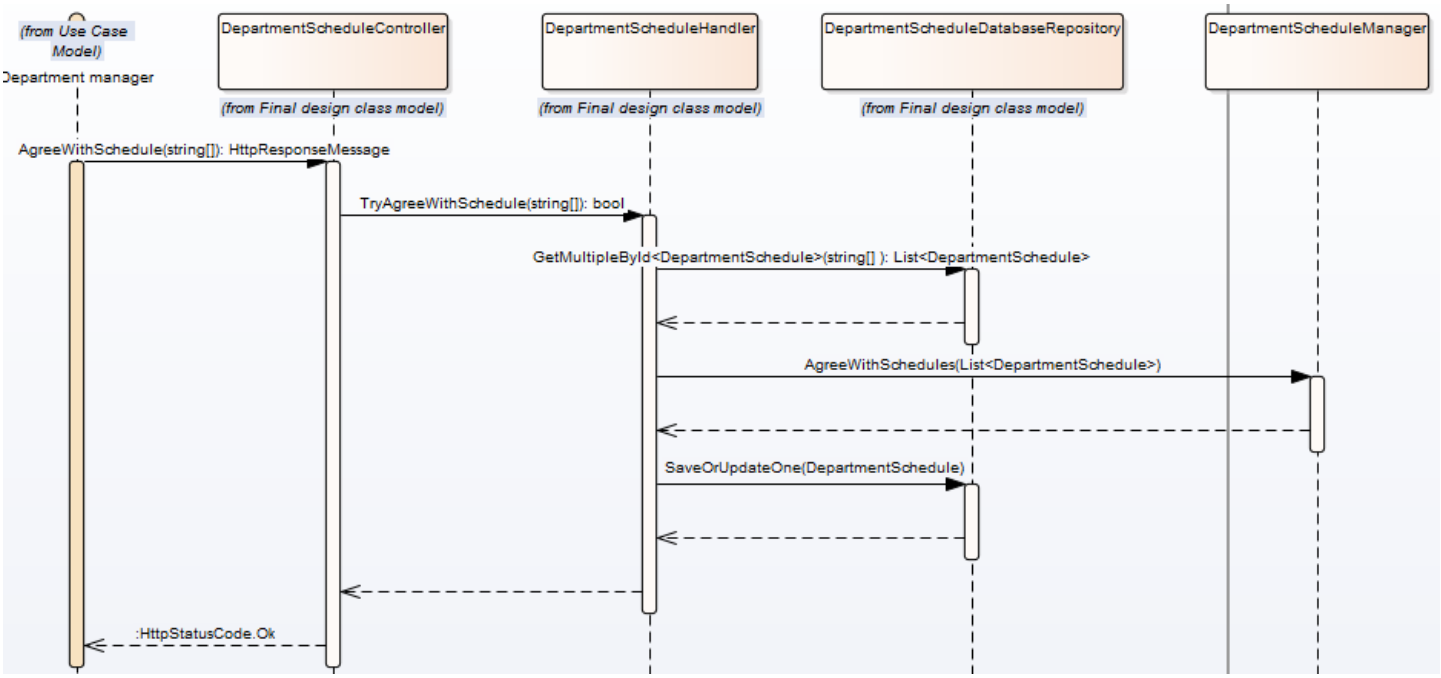


Als afdelingsmanager wil ik een akkoord kunnen geven op het rooster zodat de winkelmanager weet dat ik klaar ben met roosteren:

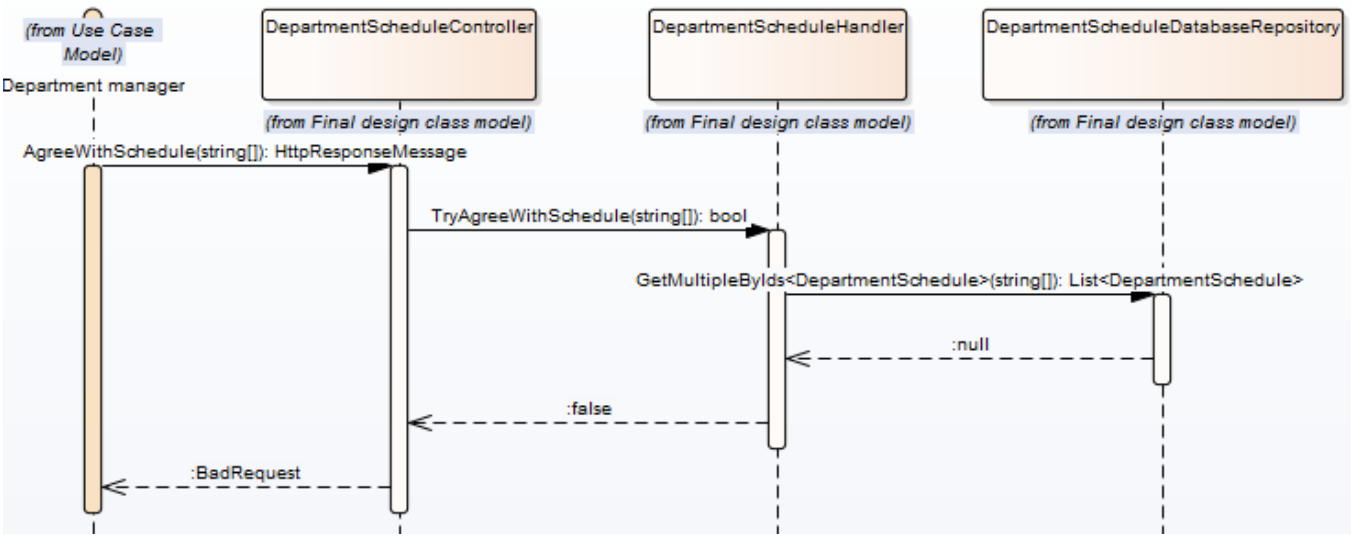
Er zijn ook hier 2 flows mogelijk:

- Succesflow
- Badrequest → als de afdelingsroosters niet bestaan(het id komt niet voor)

Succes:



Id komt niet voor:



3.2.3 Beslissingen

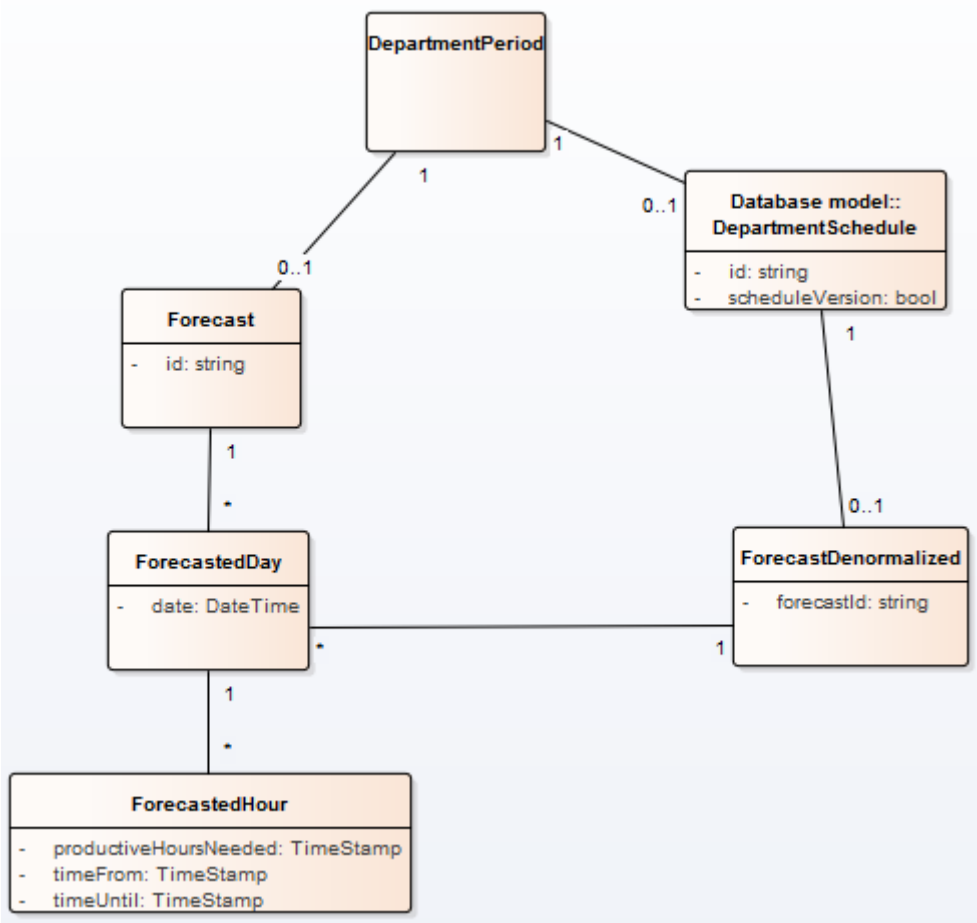
De genomen beslissingen gedurende deze sprint zijn te zien in: hoofdstuk 3.3.1 en hoofdstuk 3.4.

3.3 Database

3.3.1 Diagram

Het database diagram uit sprint 1 is uitgebreid in sprint 2. In deze paragraaf worden alleen de uitbreidingen getoond.

3.3.1.1 Tonen prognose data



Wat hier staat is het volgende:

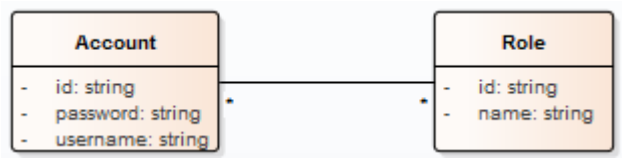
- Er is een collection genaamd forecast, deze houdt per dag, per uur bij hoeveel productieve uren er nodig zijn.
- De forecast heeft een relatie met departmentperiod, omdat een prognose hoort bij een afdeling in een periode.
- De departmentschedule hoort ook bij een departmentperiod

- De departmentschedule houdt een deel van de prognose bij, namelijk alleen de dagen, uren en productieve uren. Dit deel wordt opgeslagen in forecastDenormalized(een gedenormaliseerde vorm van forecast welke alleen de benodigde data bevat).
- De forecastDenormalized houdt geen departmentperiod bij, aangezien deze al bijgehouden wordt in de departmentschedule
- De forecastDenormalized houdt een referentie bij naar het oorspronkelijke forecast document.

Voor de functionaliteit in mijn POC had ik ook de prognose direct aan de afdelingsrooster kunnen koppelen, aangezien de prognose alleen data bevat die nodig is in het rooster(op de departmentperiod na).

De reden dat ik voor deze structuur gekozen heb, is dat in de echte applicatie de prognose veel meer data bevat. Een deel van deze data is niet nodig in het rooster of de vervolg modules. Bij het kiezen voor deze structuur is dus gedacht aan de volledige applicatie.

3.3.1.2 Inloggen



Voor het inloggen was zowel een Account als rollen nodig. Het is namelijk zo dat je bepaalde functionaliteiten alleen kunt uitvoeren als je een bepaalde rol hebt. Een account kan overigens meerdere rollen hebben.

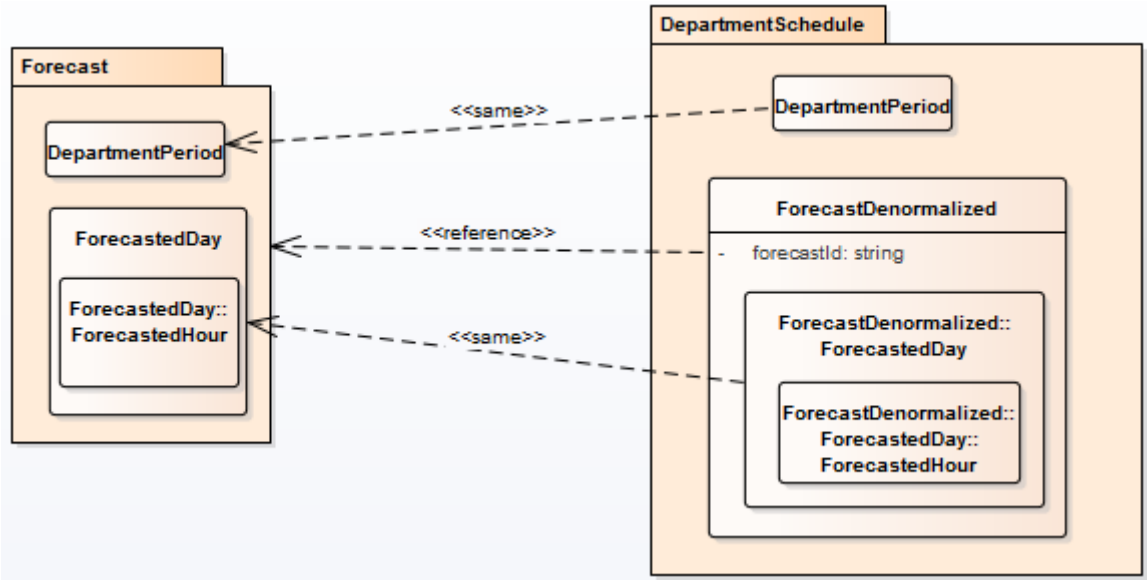
3.3.1.3 Akkoord gaan rooster

Voor het akkoord gaan met het rooster is enkel een status toegevoegd aan het rooster. Deze status is standaard Initialized, en zodra die is goedgekeurd wordt deze op Proposed gezet. Om af te dwingen dat deze status geen andere waardes bevat, is er op applicatie niveau een enum gebruikt. Dit is besproken in hoofdstuk 3.2.1.

3.3.2 Implementatie

3.3.2.1 Tonen prognose data

Voor het tonen van de prognose data in het afdelingsrooster is gebruik gemaakt van de manier denormalized reference(dit staat uitgelegd in hoofdstuk 2.3). De implementatie ziet er als volgt uit:

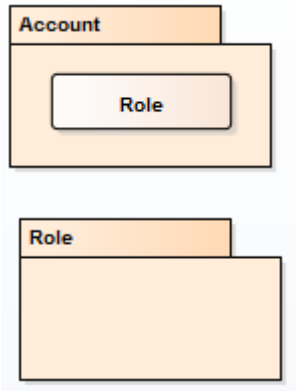


Zoals eerder uitgelegd wordt er gebruik gemaakt van denormalized references. De 2 departmentperiods die worden bijgehouden bevatten dezelfde data, dit is data die niet wijzigt. De 2 forecastedDays bevatten ook dezelfde data, ook deze wijzigt bijna niet. Als de forecastedDays wijzigen, gebeurd dit ook op maximaal 2 plaatsen.(prognose en rooster).

Verder is de forecastId uit ForecastDenormalized een reference naar de bijbehorende forecast, voor het geval dat je ooit toch het hele document nodig hebt.

Met de dependancie lijn <<same>> wordt bedoeld dat het om hetzelfde document gaat, wat op 2 plaatsen wordt embed.

3.3.2.2 Inloggen



Zoals in de afbeelding te zien, worden rollen embed in accounts.

3.4 Toelichtingen op code

De volgende stukken code hebben wellicht een toelichting nodig.

3.4.1 Formsauthentication

Voor de login functionaliteit is gebruik gemaakt van formsauthentication. Dit is authenticatie door middel van cookies. Hoewel dit niet geheel past bij een REST API(daar bijvoorbeeld mobile apps geen cookies gebruiken), heb ik hier toch voor gekozen, omdat het inloggen niet zo belangrijk is voor mijn proof of concept. Het is niet iets wat ik wil, of zou gaan aantonen namelijk. De reden dat het er toch inzit is dat rechten en rollen gebruikt worden bij bepaalde functionaliteiten uit de proof of concept.

Het plaatsen van een cookie gebeurd met de volgende regel code:

```
3 references | Xavyr Rademaker, 2 days ago | 1 author, 1 change
public void SetAuthCookie(string username, bool createPersistentCookie)
{
    FormsAuthentication.SetAuthCookie(username, createPersistentCookie);
}
```

3.4.2 Charts prognose data

Voor het tonen van de charts met prognose data is gebruik gemaakt van een open-source angular plugin genaamd Angular Charts (http://www.fusioncharts.com/angularjs-charts/#/demos/ex3). De reden dat ik hiervoor gekozen heb is vrij simpel, ik moest grafieken tonen en deze plugin doet dat voor me.

Bij de keuze voor deze plugin zijn nog een aantal andere plugins overwogen, maar die boden niet de chart aan die ik nodig had(staaf –en lijndiagram gecombineerd).

Hoewel de library open source is, maakt het gebruik van de FusionChart core js library. Deze library is niet open-source en kost geld om te gebruiken. Zelf heb ik de trial versie gebruikt voor mijn POC. Deze trial versie heeft een watermark, linksonderin de diagrammen.(De watermark is niet te zien in de POC, omdat ik het uit de js bestanden heb verwijderd).

4. Sprint 3

4.1 Backlogs

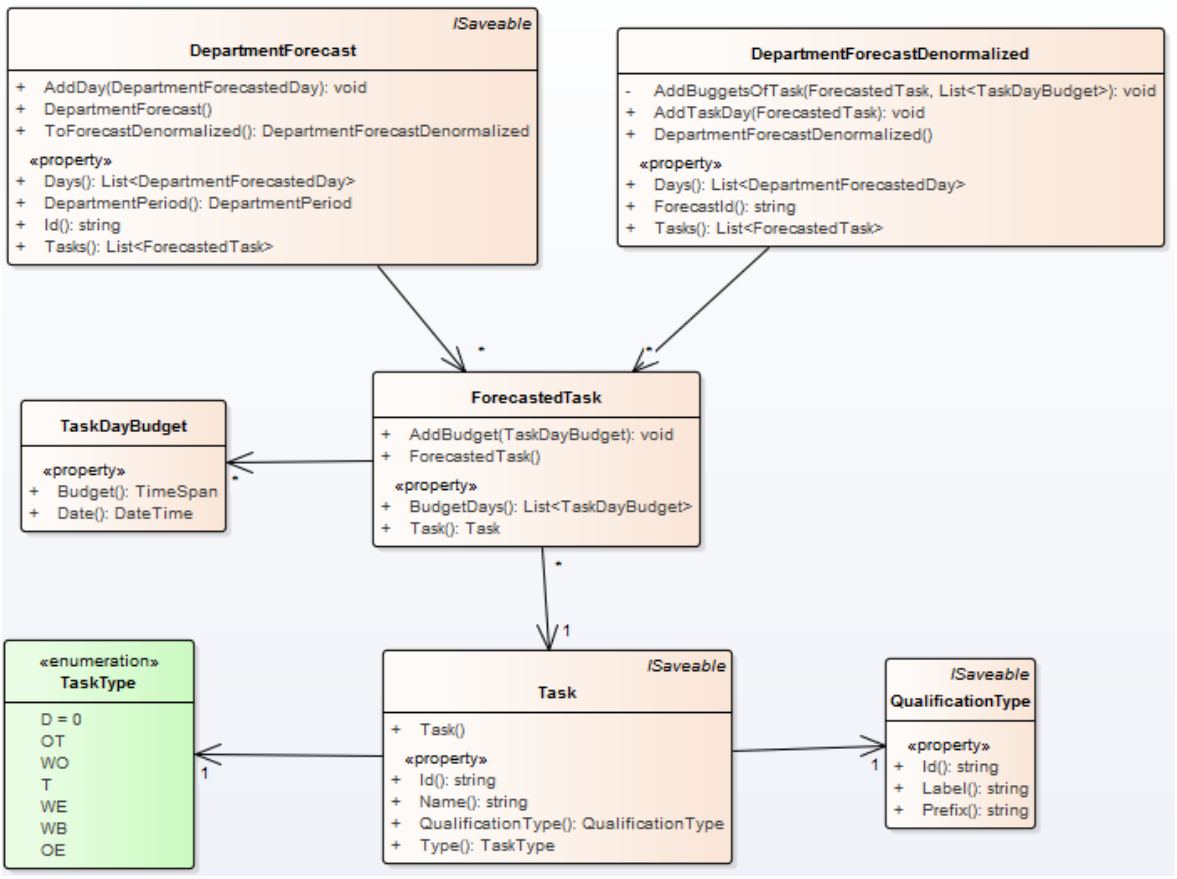
De volgende backlog items zijn in deze sprint gerealiseerd:

User story	Module	Story points
Als winkelmanager wil alle afdelingsroosters van de winkel kunnen vaststellen zodat de staat van het rooster terug kan zien zoals ik er een akkoord op heb gegeven. afronden	R	2
als afdelingsmanager wil ik kunnen zien hoeveel tijd dat mijn medewerkers krijgen om een taak uit te voeren zodat ik niet teveel of te weinig mensen inplan en of dat ik de goede personen inplan	R/p	6
Als afdelingsmanager wil ik kunnen zien of een medewerker beschikbaar is zodat ik geen niet beschikbare mensen inrooster	R	10
Als afdelingsmanager wil ik een afdelingsrooster kunnen initialiseren zodat ik het roosterproces kan starten	R	6

4.2 Design

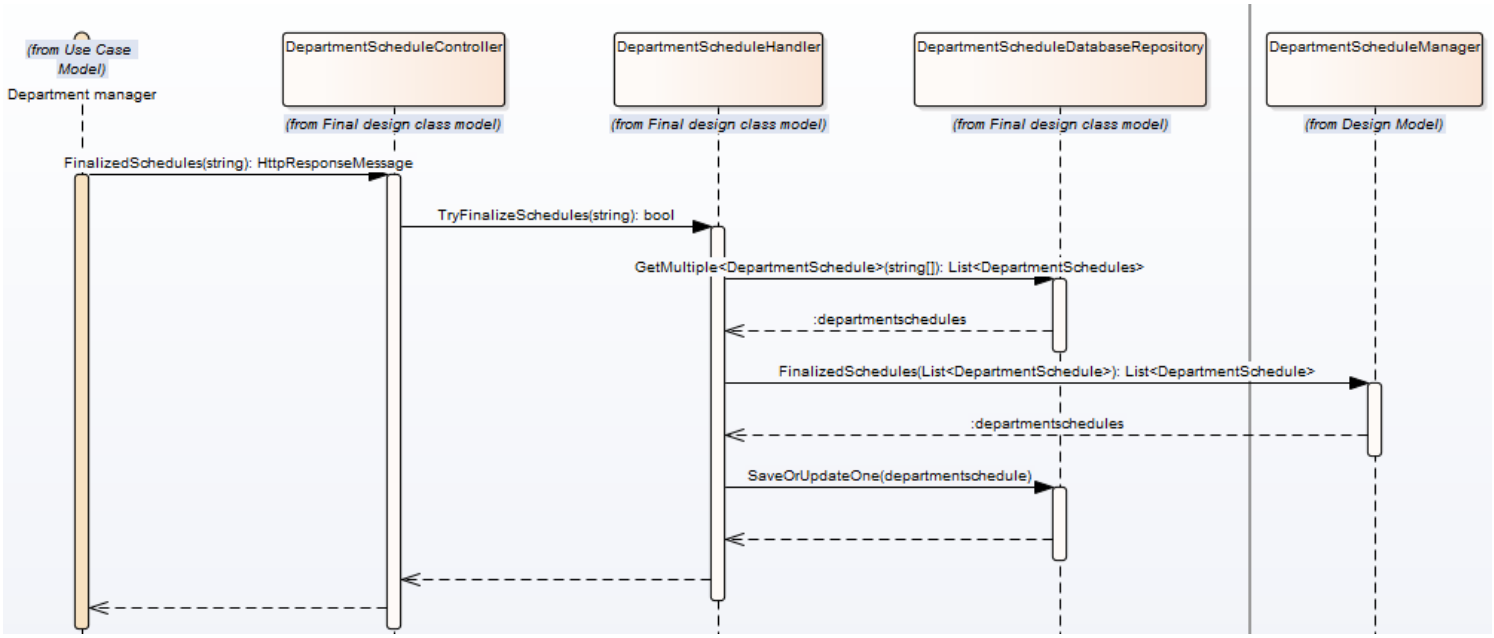
4.2.1 Klassendiagrammen

Zien taak budget:



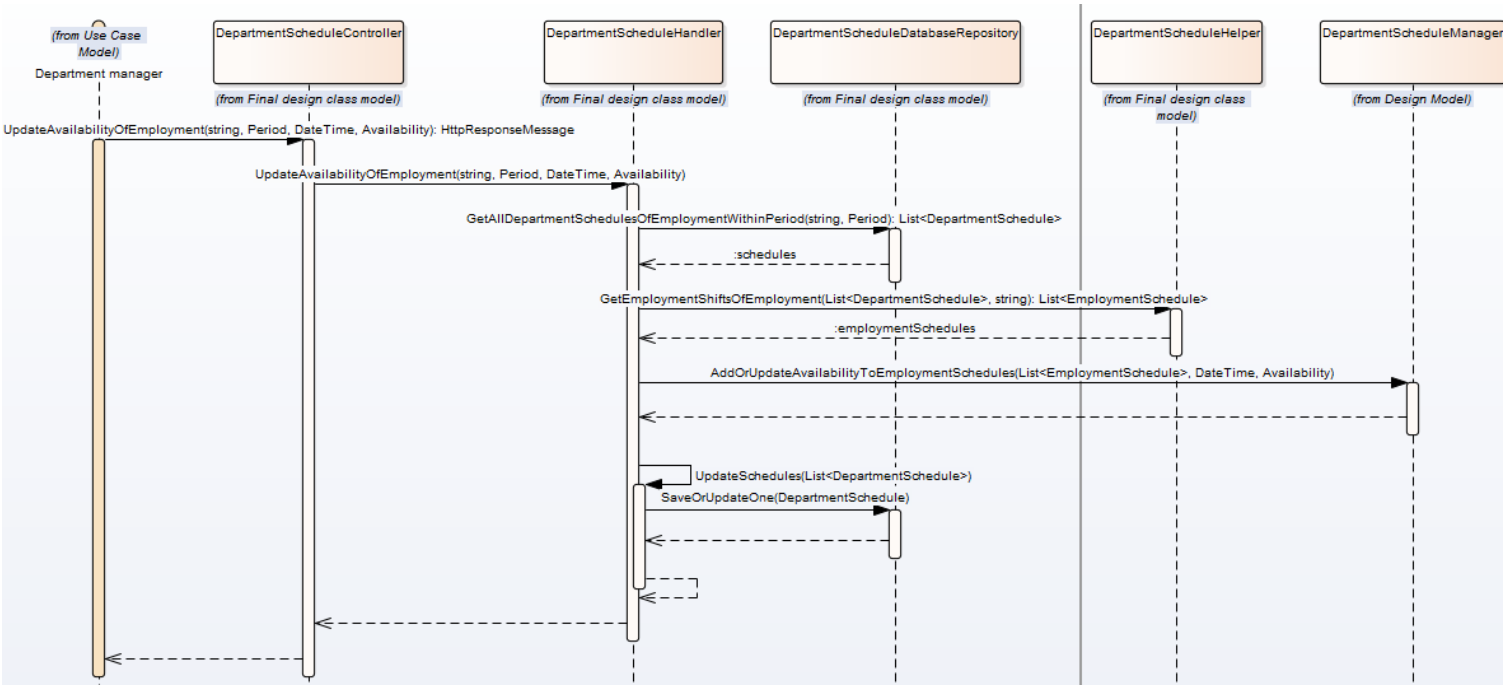
4.2.2 Sequentie diagrammen

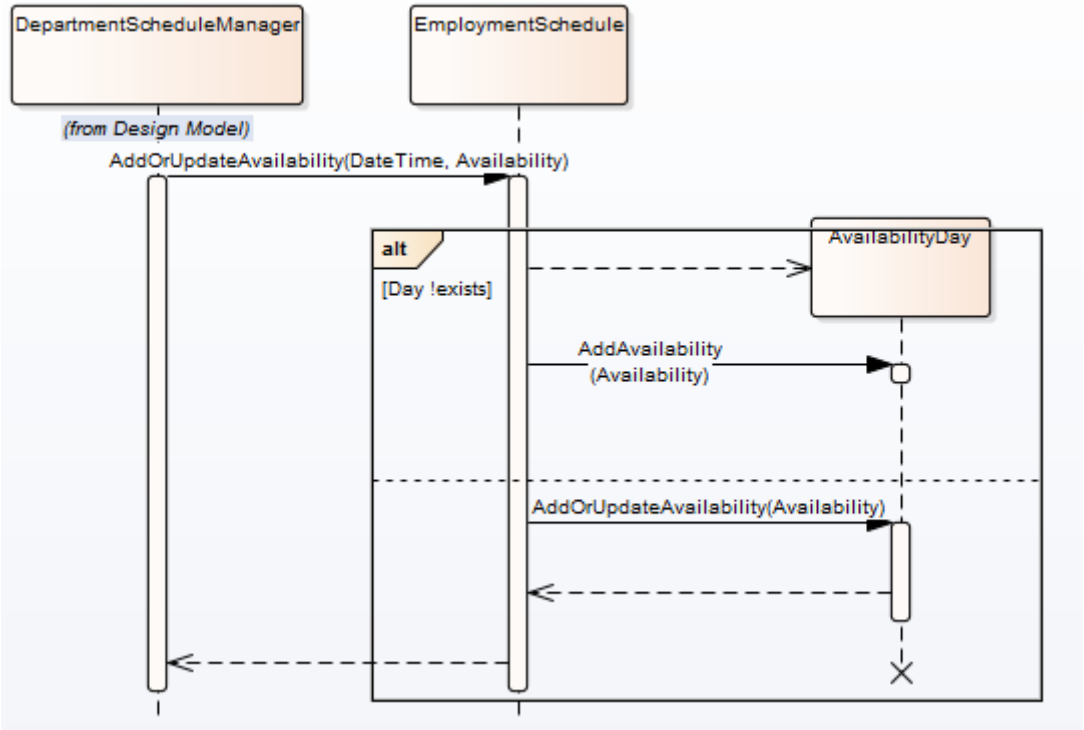
Vaststellen roosters:



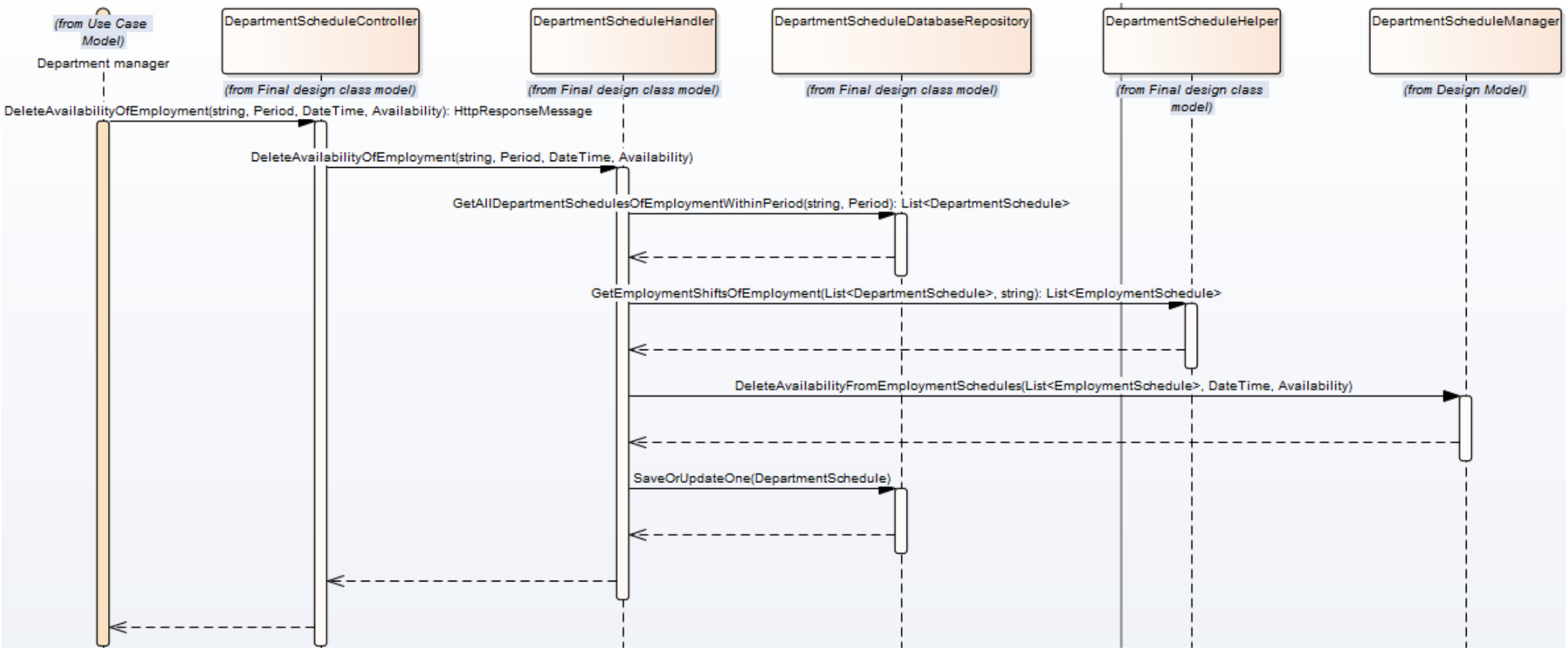
Beheren beschikbaarheid:

Toevoegen/wijzigen beschikbaarheid:





Verwijderen beschikbaarheid:

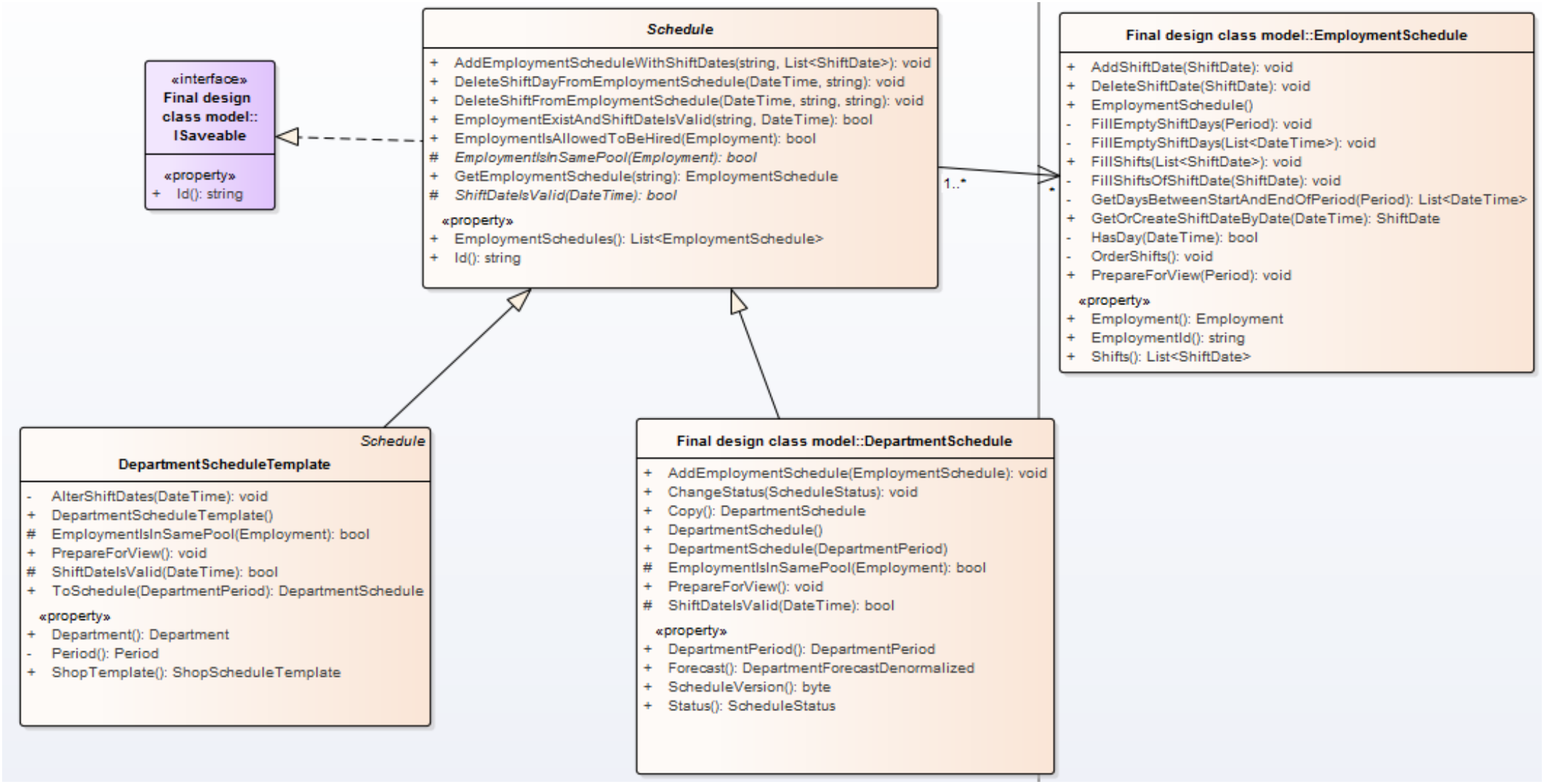

(from Use Case Model)
Department manager
DepartmentScheduleController
(from Final design class model)
DepartmentScheduleHandler
(from Final design class model)
DepartmentScheduleDatabaseRepository
(from Final design class model)
DepartmentScheduleHelper
(from Final design class model)
DepartmentScheduleManager
(from Design Model)
DeleteAvailabilityOfEmployment(string, Period, DateTime, Availability): HttpResponseMessage
DeleteAvailabilityOfEmployment(string, Period, DateTime, Availability)
GetAllDepartmentSchedulesOfEmploymentWithinPeriod(string, Period): List<DepartmentSchedule>
GetEmploymentShiftsOfEmployment(List<DepartmentSchedule>, string): List<EmploymentSchedule>
DeleteAvailabilityFromEmploymentSchedules(List<EmploymentSchedule>, DateTime, Availability)
SaveOrUpdateOne(DepartmentSchedule)

4.2.3 Beslissingen

Er is een superklasse gemaakt genaamd Schedule. Deze houdt de relatie naar employmentschedule bij in een property en realized van ISaveable.

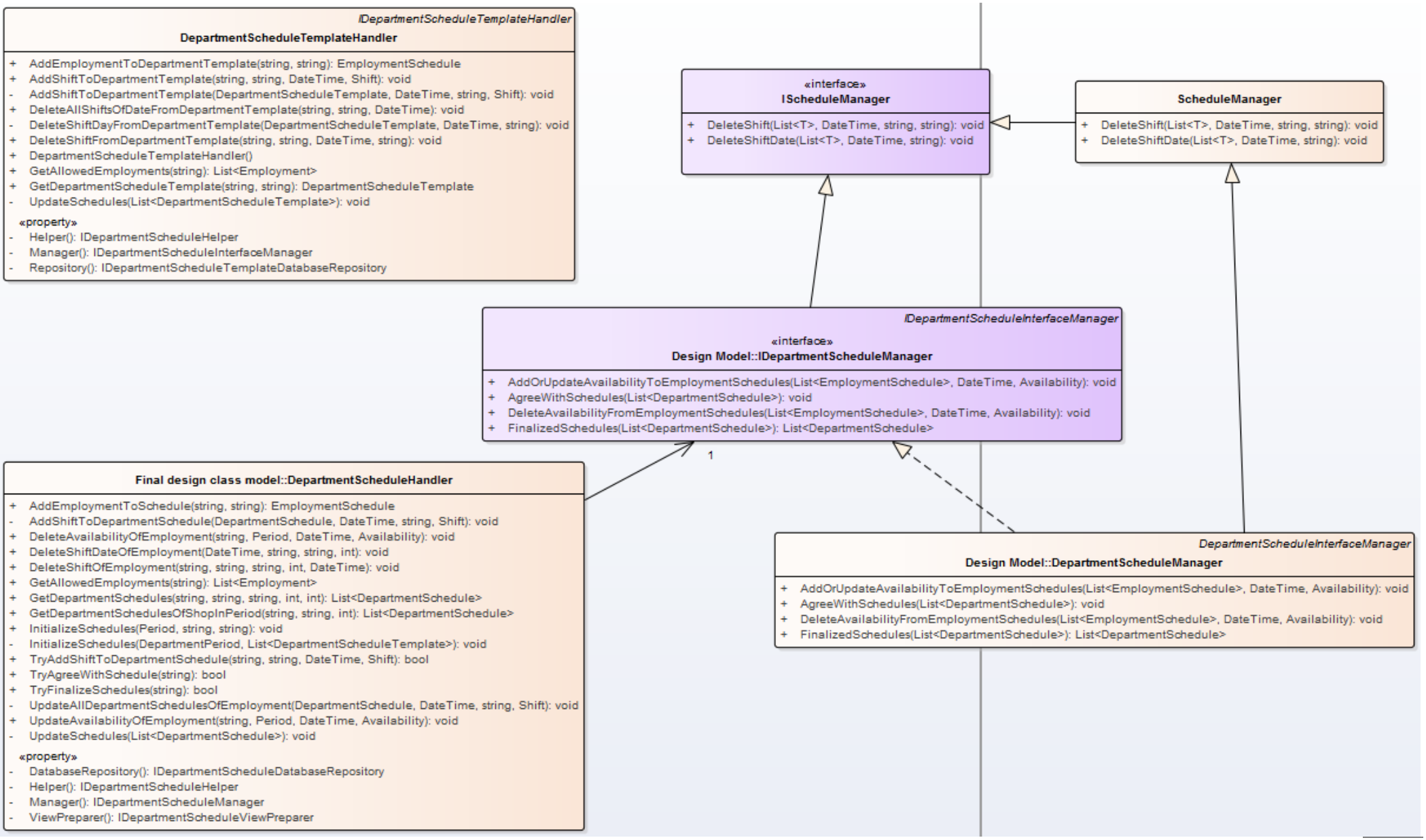
Verder bevat deze superklasse alle methodes die te maken hebben met het beheren van de employmentschedules. De superklasse is overigens abstract.

Dit ziet er als volgt uit:


«interface»
Final design class model::
ISaveable
«property»
+ Id(): string
Schedule
+ AddEmploymentScheduleWithShiftDates(string, List<ShiftDate>): void
+ DeleteShiftDayFromEmploymentSchedule(DateTime, string): void
+ DeleteShiftFromEmploymentSchedule(DateTime, string, string): void
+ EmploymentExistAndShiftDateIsValid(string, DateTime): bool
+ EmploymentIsAllowedToBeHired(Employment): bool
# EmploymentIsInSamePool(Employment): bool
+ GetEmploymentSchedule(string): EmploymentSchedule
# ShiftDateIsValid(DateTime): bool
«property»
+ EmploymentSchedules(): List<EmploymentSchedule>
+ Id(): string
1..*
*
Final design class model::EmploymentSchedule
+ AddShiftDate(ShiftDate): void
+ DeleteShiftDate(ShiftDate): void
+ EmploymentSchedule()
- FillEmptyShiftDays(Period): void
- FillEmptyShiftDays(List<DateTime>): void
+ FillShifts(List<ShiftDate>): void
- FillShiftsOfShiftDate(ShiftDate): void
- GetDaysBetweenStartAndEndOfPeriod(Period): List<DateTime>
+ GetOrCreateShiftDateByDate(DateTime): ShiftDate
- HasDay(DateTime): bool
- OrderShifts(): void
+ PrepareForView(Period): void
«property»
+ Employment(): Employment
+ EmploymentId(): string
+ Shifts(): List<ShiftDate>
Schedule
DepartmentScheduleTemplate
- AlterShiftDates(DateTime): void
+ DepartmentScheduleTemplate()
# EmploymentIsInSamePool(Employment): bool
+ PrepareForView(): void
# ShiftDateIsValid(DateTime): bool
+ ToSchedule(DepartmentPeriod): DepartmentSchedule
«property»
+ Department(): Department
- Period(): Period
+ ShopTemplate(): ShopScheduleTemplate
Final design class model::DepartmentSchedule
+ AddEmploymentSchedule(EmploymentSchedule): void
+ ChangeStatus(ScheduleStatus): void
+ Copy(): DepartmentSchedule
+ DepartmentSchedule()
+ DepartmentSchedule(DepartmentPeriod)
# EmploymentIsInSamePool(Employment): bool
+ PrepareForView(): void
# ShiftDateIsValid(DateTime): bool
«property»
+ DepartmentPeriod(): DepartmentPeriod
+ Forecast(): DepartmentForecastDenormalized
+ ScheduleVersion(): byte
+ Status(): ScheduleStatus

Naast de superklasse, is er ook een wijziging aangebracht op de managers. Er is een super manager welke een aantal methodes bevat die door beide realisaties hetzelfde worden uitgevoerd. Verder is er voor de departmentschedule klasse een sub manager gemaakt(de template had niet meer methodes nodig van de manager).

Op zowel de super als subklassen zijn interfaces geplaats, deze werken op dezelfde manier als de database repositories. Dit ziet er als volgt uit:

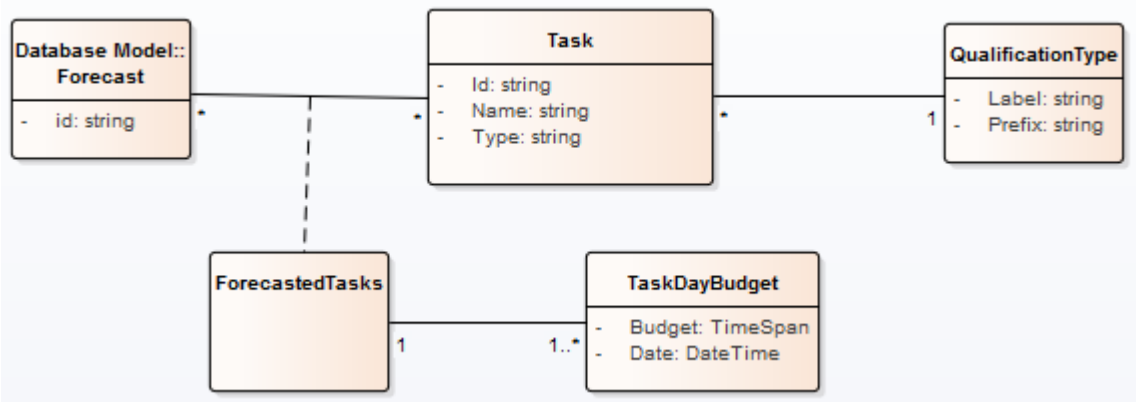

IDepartmentScheduleTemplateHandler
DepartmentScheduleTemplateHandler
+ AddEmploymentToDepartmentTemplate(string, string): EmploymentSchedule
+ AddShiftToDepartmentTemplate(string, string, DateTime, Shift): void
- AddShiftToDepartmentTemplate(DepartmentScheduleTemplate, DateTime, string, Shift): void
+ DeleteAllShiftsOfDateFromDepartmentTemplate(string, string, DateTime): void
- DeleteShiftDayFromDepartmentTemplate(DepartmentScheduleTemplate, DateTime, string): void
+ DeleteShiftFromDepartmentTemplate(string, string, DateTime, string): void
+ DepartmentScheduleTemplateHandler()
+ GetAllowedEmployments(string): List<Employment>
+ GetDepartmentScheduleTemplate(string, string): DepartmentScheduleTemplate
- UpdateSchedules(List<DepartmentScheduleTemplate>): void
«property»
- Helper(): IDepartmentScheduleHelper
- Manager(): IDepartmentScheduleInterfaceManager
- Repository(): IDepartmentScheduleTemplateDatabaseRepository
«interface»
IScheduleManager
+ DeleteShift(List<T>, DateTime, string, string): void
+ DeleteShiftDate(List<T>, DateTime, string): void
ScheduleManager
+ DeleteShift(List<T>, DateTime, string, string): void
+ DeleteShiftDate(List<T>, DateTime, string): void
IDepartmentScheduleInterfaceManager
«interface»
Design Model::IDepartmentScheduleManager
+ AddOrUpdateAvailabilityToEmploymentSchedules(List<EmploymentSchedule>, DateTime, Availability): void
+ AgreeWithSchedules(List<DepartmentSchedule>): void
+ DeleteAvailabilityFromEmploymentSchedules(List<EmploymentSchedule>, DateTime, Availability): void
+ FinalizedSchedules(List<DepartmentSchedule>): List<DepartmentSchedule>
1
Final design class model::DepartmentScheduleHandler
+ AddEmploymentToSchedule(string, string): EmploymentSchedule
- AddShiftToDepartmentSchedule(DepartmentSchedule, DateTime, string, Shift): void
+ DeleteAvailabilityOfEmployment(string, Period, DateTime, Availability): void
+ DeleteShiftDateOfEmployment(DateTime, string, string, int): void
+ DeleteShiftOfEmployment(string, string, string, int, DateTime): void
+ GetAllowedEmployments(string): List<Employment>
+ GetDepartmentSchedules(string, string, string, int, int): List<DepartmentSchedule>
+ GetDepartmentSchedulesOfShopInPeriod(string, string, int): List<DepartmentSchedule>
+ InitializeSchedules(Period, string, string): void
- InitializeSchedules(DepartmentPeriod, List<DepartmentScheduleTemplate>): void
+ TryAddShiftToDepartmentSchedule(string, string, DateTime, Shift): bool
+ TryAgreeWithSchedule(string): bool
+ TryFinalizeSchedules(string): bool
- UpdateAllDepartmentSchedulesOfEmployment(DepartmentSchedule, DateTime, string, Shift): void
+ UpdateAvailabilityOfEmployment(string, Period, DateTime, Availability): void
- UpdateSchedules(List<DepartmentSchedule>): void
«property»
- DatabaseRepository(): IDepartmentScheduleDatabaseRepository
- Helper(): IDepartmentScheduleHelper
- Manager(): IDepartmentScheduleManager
- ViewPreparer(): IDepartmentScheduleViewPreparer
DepartmentScheduleInterfaceManager
Design Model::DepartmentScheduleManager
+ AddOrUpdateAvailabilityToEmploymentSchedules(List<EmploymentSchedule>, DateTime, Availability): void
+ AgreeWithSchedules(List<DepartmentSchedule>): void
+ DeleteAvailabilityFromEmploymentSchedules(List<EmploymentSchedule>, DateTime, Availability): void
+ FinalizedSchedules(List<DepartmentSchedule>): List<DepartmentSchedule>

4.3 Database beslissingen

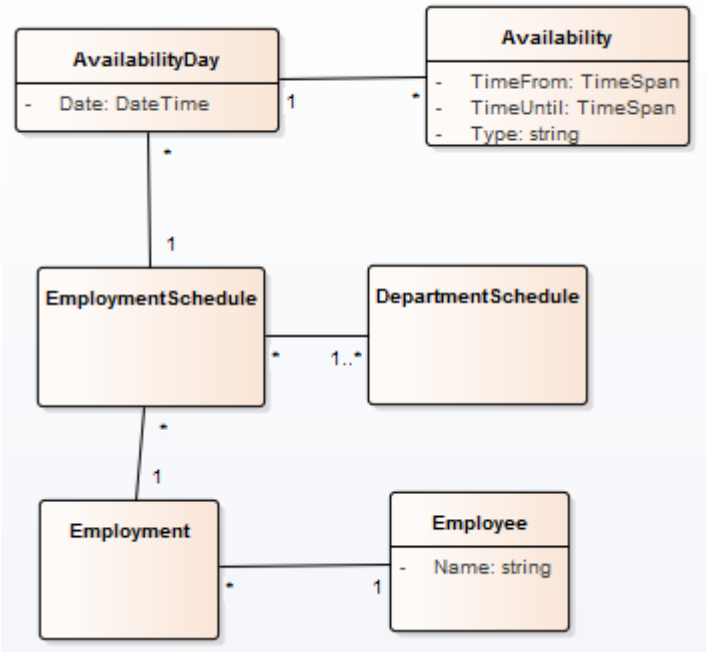
De database is als volgt uitgebreid:

4.3.1 Diagrammen

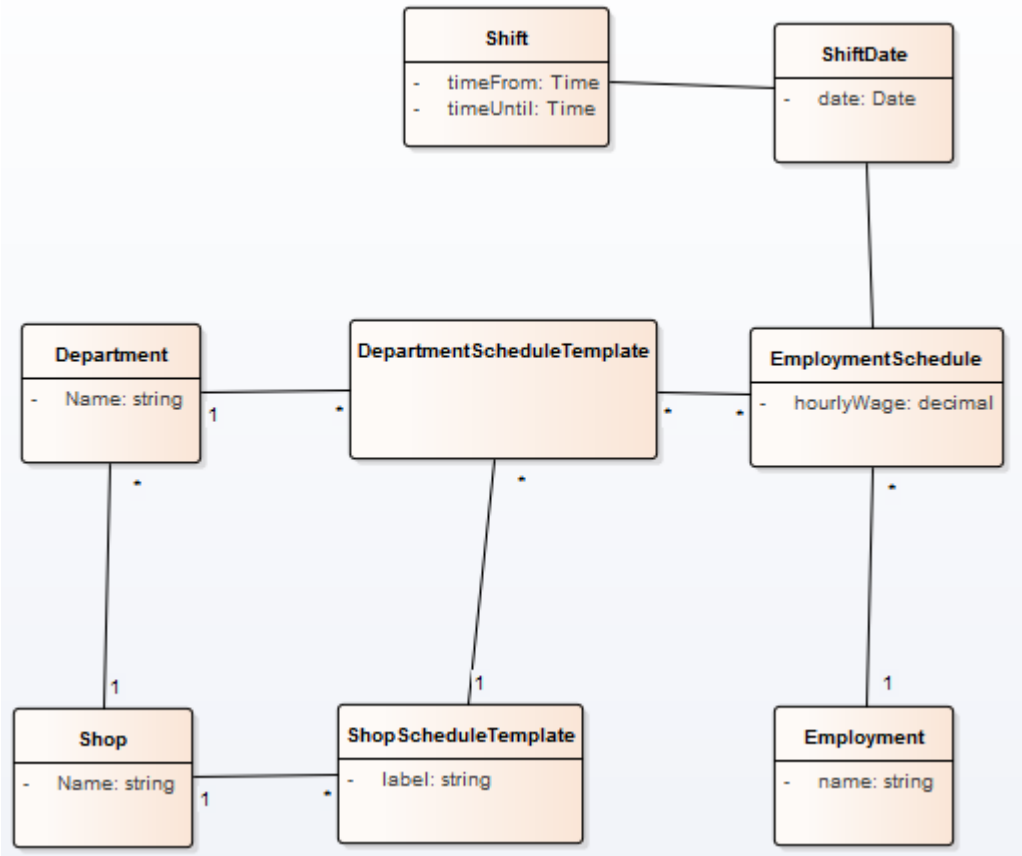
Zien taak budget:



Beheren beschikbaarheid:

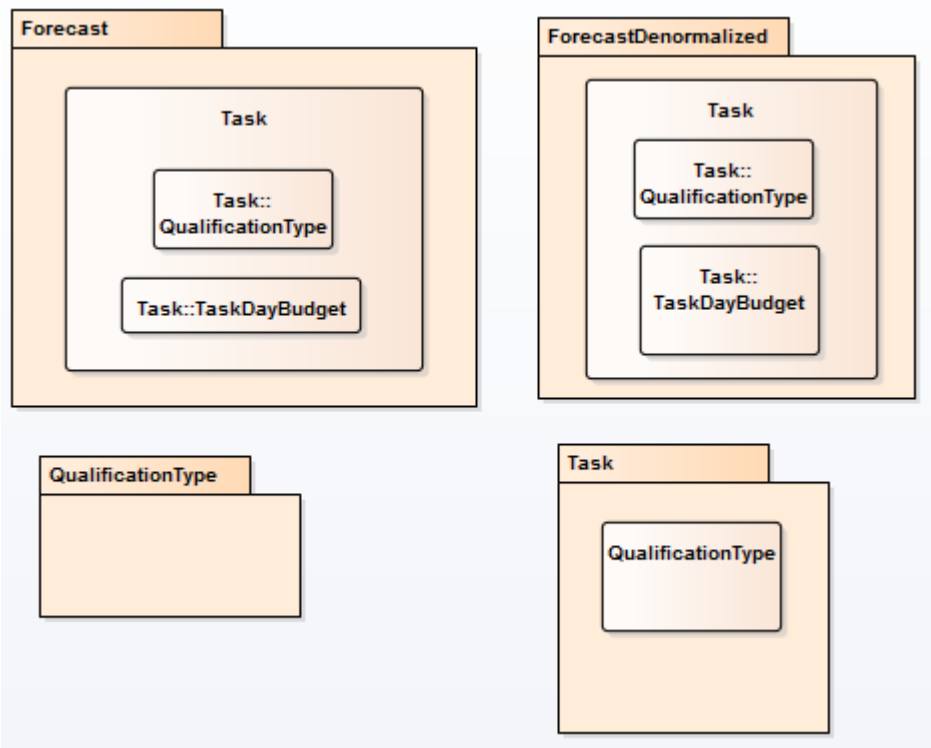


Initialiseren rooster:

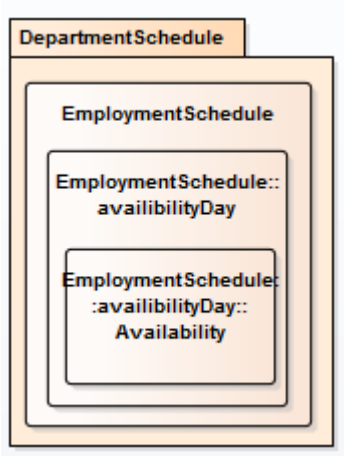


4.3.2 Implementatie

Zien taak budget:



Beheren beschikbaarheid:



Initialiseren rooster:

De departmentscheduletemplate embed de employeeschedules op dezelfde manier als departmentschedule dat doet(zie hoofdstuk 2). Verder embed hij zowel department als shopscheduletemplate.

Schopschedule template embed weer de shop.

4.3.3 Groote document departmentschedule

Zoals te zien in deze en de afgelopen sprints, is bevat het afdelingsrooster momenteel vrij veel data. In een forum (https://groups.google.com/forum/#!topic/ravendb/MvXvs6lNxWs) las ik dat Oren Eini(ookwel bekend als Ayende Rahien, Founder van Hibernating Rhino's het bedrijf achter RavenDB) aanraad documenten maximaal een paar honderd kilobyte groot te houden.

Ik heb een rooster gemaakt, met daarin een realistische hoeveelheid data(13 medewerkers, 56 diensten, 14 taken inclusief een budget voor elke dag voor elke taak en per medewerker 7 beschikbaarheden). Vervolgens heb ik de JSON van dit document in een .txt document gezet om te kijken hoe groot het document is. Dit kwam uit op 114 KB.

Dit betekent dat het document binnen de marge van een aantal honderd KB valt. Mocht het echter zo zijn dat er in de daadwerkelijke applicatie meer informatie bij komt kijken dan momenteel getoond wordt, dan zullen een aantal relaties naar references omgezet moeten worden om ruimte te besparen.

Een aantal van deze relaties kunnen zijn de budget van de taken en beschikbaarheden. Reden dat dit een reference kan zijn, is dat dit op een ander tabblad getoond wordt. Dit kan dus async opgehaald worden terwijl het rooster al getoond is. Op deze manier is de data er vlak na de rooster data, en merkt de gebruiker het niet omdat het zich in een ander tabblad bevind.

Een andere mogelijkheid is heel de denormalized reference van prognose ombouwen naar een volledige reference. Dit betekent echter wel dat de grafieken ook iets later getoond worden.

4.4 Toelichtingen op code

De methodes in de IScheduleManager hebben de volgende signature:

```
void DeleteShift<T>(List<T> schedules, DateTime date, string employmentId, string shiftId)
where T : Schedule;
```

Deze code houdt het volgende in:
De lijst die wordt meegegeven als parameter is een Generic lijst(oftewel het is van het type dat wordt aangegeven in de eerste T). echter is het zo, dat de where na de methode afdwingt dat T van het type Schedule is.

De reden dat er niet gewoon gekozen is voor een parameter van het type List<Schedule> is dat er dan gecast, of gekopieerd moet worden bij het aanroepen van de methode(List<Schedule> kan niet als parameter voor List<Schedule> gegeven worden, deze moet eerst gecast worden naar List<Schedule> OF moet gekopieerd worden met de ToList<Schedule>() methode).

Het probleem wat het kopiëren gaf, was dat de meegegeven lijst na deze handeling opgeslagen wordt(hij wordt als reference meegegeven). Dit kan niet op het moment dat er een kopie van de lijst wordt meegegeven.

Op deze manier kan de lijst meegegeven worden, zonder te hoeven casten of reference.

5. Sprint 4

In deze sprint is niks nieuwe ontwikkelt. Na het ontdekken van het belangrijke verschil tussen de Load en Query methode, is er alleen een klein stukje code gewijzigd. In dit hoofdstuk wordt deze wijziging toegelicht.

5.1 Load vs. Query

Gedurende deze sprint ben ik erachter gekomen dat er een belangrijk verschil is tussen de Load en de Query methode. Het is namelijk zo, dat de Load methode de meest recente(consistentie) versie van een document/list garandeert. Dit komt omdat load het ophaalt aan de hand van het Id, en dus geen index gebruikt om de documenten op te halen.

De query methode daarentegen maakt wel gebruik van indexen. Hierdoor kan RavenDB niet garanderen dat het de meest recente versie van een document/list wordt opgehaald. Het kan namelijk zo zijn dat een document net geupdate is, en dus nog bezig is met indexeren op het moment dat een query wordt uitgevoerd. Omdat het document nog bezig is met indexeren, wordt de nieuwste versie hiervan niet meegenomen in de query. Dit kan resulteren in of een oudere versie van het document(als het oude document ook aan de zoekcriteria voldoet) of het document wordt niet meegenomen in de resultaten(omdat de oude versie niet aan de zoekcriteria voldoet, maar de nieuwe versie wel).

Voor de applicatie betekent dit, dat er gekeken moet worden waar de meest recente versie altijd van belang is, en waar dit niet het geval is. Een voorbeeld van een query waar de meest recente versie niet van belang hoeft te zijn, is het ophalen van alle periodes. De reden dat consistentie hier minder van belang is, is dat het puur referentie data is wat niet wijzigt. Op het moment dat de periodes van 2017 worden toegevoegd, hoeven deze niet meteen in het systeem te staan.

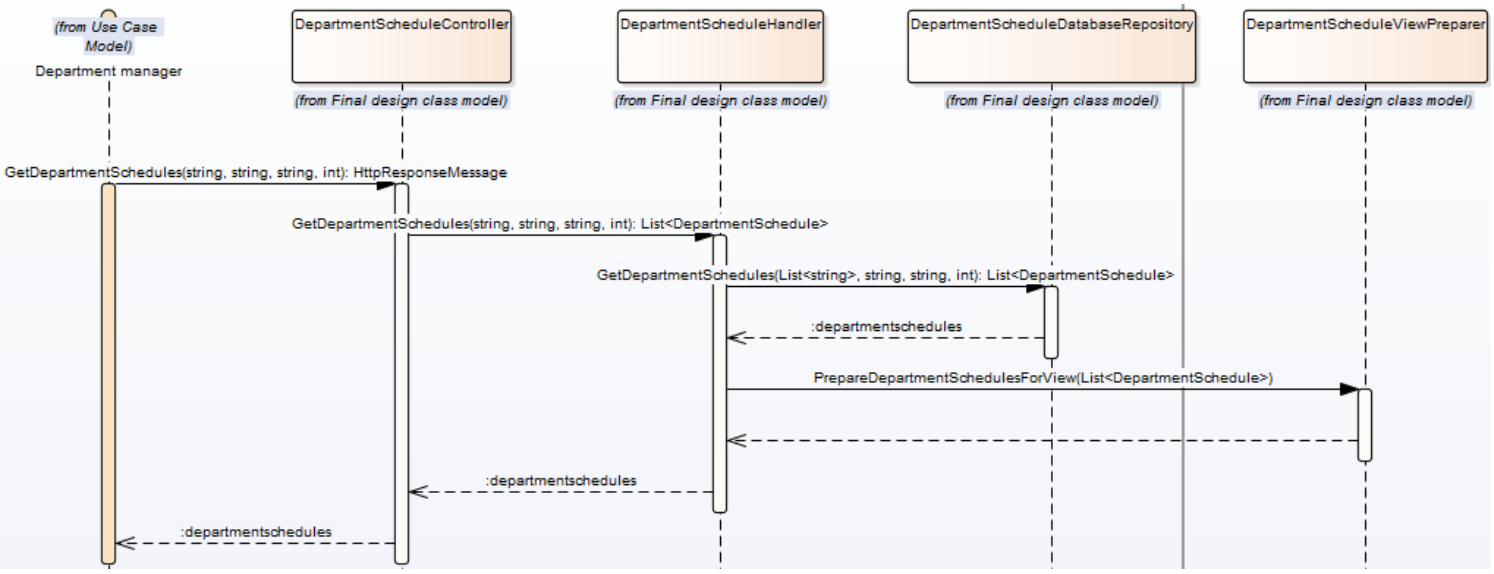
Een punt waar consistentie waarschijnlijk wel van belang is, is het rooster. Omdat je altijd de meest recente versie van het rooster wil ophalen om deze te kunnen bekijken/wijzigen, is het hier belangrijker dat je ook de meest recente versie ophaalt(en dus niet een versie waar net toegevoegde diensten nog niet op staan). Hier zal dus de Load methode gebruikt moeten worden.

Load kan gebruikt worden om 1 object te returne, maar ook om een list te returne. Dit is afhankelijk van de parameter die je meegeeft. Als je 1 string meegeeft als parameter, krijg je 1 object terug, als je een string [] meegeeft krijg je een list terug(eigenlijk een IEnumerable).

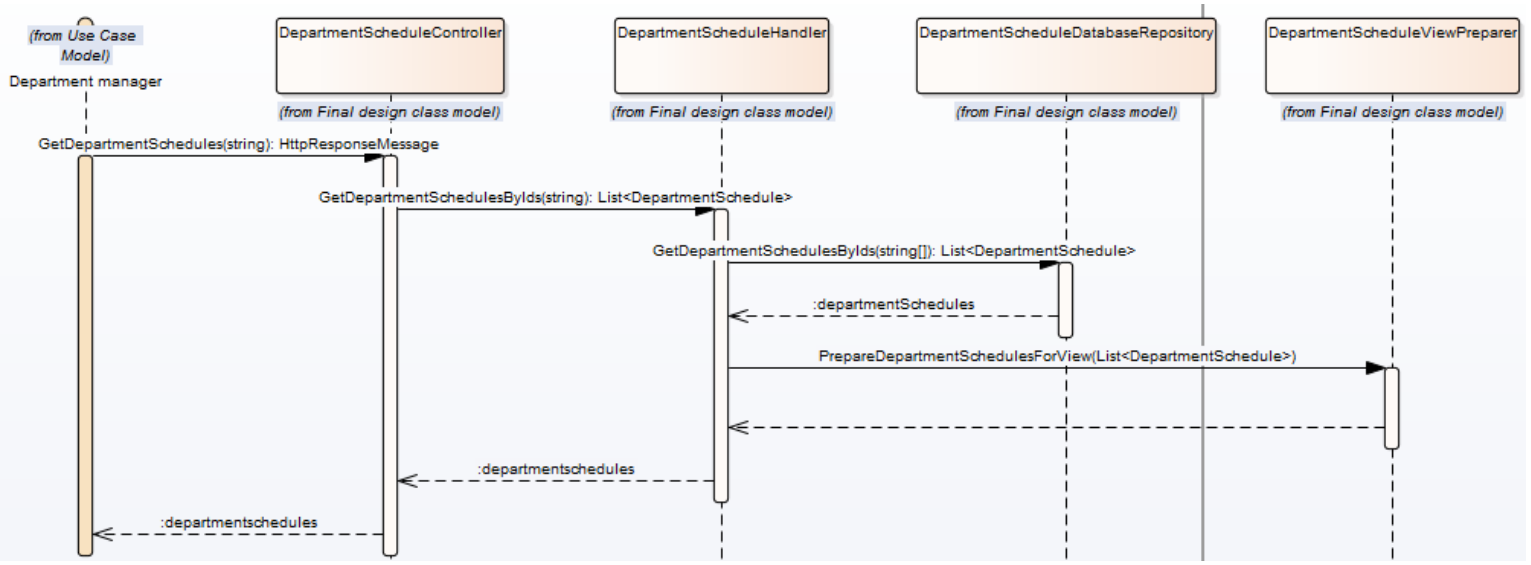
In de onderstaande diagrammen is te zien hoe ik in de proof of concept de flow van het ophalen van roosters heb aangepast. Hoewel dezelfde klasses gebruikt worden, zijn de het methodes die anders zijn.

5.1.1 Diagrammen:

Oude situatie:



Nieuwe situatie:



5.1.2 Query's

De gebruikte methodes om de roosters op te halen is dus ook veranderd. Voorheen was deze methode als volgt:

```
public List<DepartmentSchedule> GetDepartmentSchedules(List<string> departmentNames, string shopName, string week, int year)
public List<DepartmentSchedule> GetDepartmentSchedulesByIds(string[] ids)
{
	using (var session = DatabaseConnection.OpenSession("RRWeb_POC"))
	{
		List<DepartmentSchedule> departmentschedules = session
		.Include<DepartmentSchedule>(deptSchedule => deptSchedule.EmploymentSchedules.Select(empShift => empShift.EmploymentId))
		.Load<DepartmentSchedule>(ids)
		.ToList();

		foreach (var returnSchedule in departmentschedules)
		{
			foreach (EmploymentSchedule shift in returnSchedule.EmploymentSchedules)
			{
				shift.Employment = session.Load<Employment>(shift.EmploymentId);
			}
		}

		return departmentschedules;
	}
}
```

De nieuwe query ziet er als volgt uit:

Bijlage F

Conclusies en aanbevelingen

Xavyr Rademaker

Inhoudsopgave

1. Inleiding

Naar aanleiding van het project wat ik heb uitgevoerd omtrent document databases vs. relationele databases voor R&R-web, ben ik tot een aantal conclusies, aandachtspunten en adviezen gekomen. In dit document worden deze conclusies, aandachtspunten en adviezen uitgelegd en beargumenteerd. Er wordt advies gegeven over het overstappen naar de document database genaamd RavenDB.

Allereerst wordt uitgelegd hoe er gewerkt wordt met RavenDB. Hierbij wordt verteld hoe standaard CRUD acties in RavenDB gaan, en hoe ik het werken met deze database heb ervaren. Bij mijn ervaring vertel ik zowel wat ik prettig vond aan het werken met deze database, en wat ik er minder prettig aan vond. Dit wordt gedaan vanuit een ontwikkel perspectief(dus van het communiceren en werken met de database vanuit code) en vanuit een beheers perspectief(het werken met de database vanuit de GUI).

Nadat mijn mening omtrent RavenDB is uitgelegd, kom ik tot een conclusie. In de conclusie komt naar voren bij welke modules ik denk dat het praktisch is om over te stappen naar document databases aan de hand van mijn onderzoek en proof of concept. Ook wordt er in de conclusie besproken wat mijn advies is over het overstappen naar de document database RavenDB.

Tot slot doe ik nog een aantal aanbevelingen/adviezen. Deze zijn gebaseerd op de conclusie en mijn bevindingen gedurende het project.

2. Werken met RavenDB

In dit hoofdstuk bespreek ik hoe er gewerkt wordt met RavenDB. Allereerst worden CRUD(Create, Read, Update, Delete) acties kort toegelicht. Vervolgens bespreek ik mijn ervaring met deze database. Onder mijn ervaring valt:

- Wat vond ik prettig als ontwikkelaar
- Wat vond ik minder prettig als ontwikkelaar
- Wat vond ik prettig als databasebeheerder
- Wat vond ik minder prettig als databasebeheerder

2.1 CRUD

De standaard CRUD acties werken vrij eenvoudig vanuit C# vanwege de aanwezigheid van een library. Hieronder wordt per handeling besproken hoe dit werkt, en hoe ik er gebruik van heb gemaakt. Alle methodes gebruikt voor de CRUD acties komen uit de IDocumentSession interface.

2.1.1 Create en update

Het maken en updaten van documents zijn beide mogelijk met dezelfde methode. Deze methode heet Store en verwacht minimaal 1 parameter, namelijk het object wat je wilt opslaan. Het is mogelijk om een 2e parameter mee te geven, namelijk een string met de Id van het object. Als deze wordt meegegeven gebruikt RavenDB het opgegeven Id om het document te identificeren. Als deze niet is meegegeven of null is genereert RavenDB zelf een id.

Op het moment dat de meegegeven Id al bestaat, dan betekent dit dat het document al bestaat. In plaats van het document opnieuw op te slaan, wordt het bestaande document gewoon geüpdatet.

Omdat de eerste parameter van het type object is, is het mogelijk deze methode generiek te maken voor alle klassen die in de database opgeslagen moeten worden. Door een interface boven deze klassen te plaatsen (welke afdwingt dat alle realiserende klassen een property Id hebben) en deze interface mee te geven aan de opslaan/update methode werkt de methode voor alle te persisteren klassen. De methode zal er dan als volgt uitzien:

```
public void SaveOrUpdateOne(ISaveable saveable)
{
	using (var session = DatabaseConnection.OpenSession("RRWeb_POC"))
	{
		session.Store(saveable, saveable.Id);
		session.SaveChanges();
	}
}
```

2.1.2 Delete

Het verwijderen van documents gaat ook op basis van de Id van documenten. Hierdoor is het ook hiervoor mogelijk om generieke methodes te maken. Dit kan met behulp van dezelfde interface als beschreven in paragraaf 2.1.2, of met behulp van een string Id als parameter.

Met de interface ziet dit er als volgt uit:

```
public void Delete(ISaveable saveable)
{
	using (var session = DatabaseConnection.OpenSession("RRWeb_POC"))
	{
		session.Delete(saveable.Id);
		session.SaveChanges();
	}
}
```

2.1.3 Read – Load vs. Query

Voor het ophalen van documenten zijn er 2 manieren mogelijk, namelijk door te zoeken op basis van Id('s) en door te zoeken op basis van attributen. Het verschil tussen deze 2 manieren is heel belangrijk om te begrijpen.

2.1.3.1 Zoeken op Id

Het zoeken op Id('s) werkt vrij simpel, en is op dezelfde manier als de delete methode generiek te maken. Hiervoor wordt de load methode aangeroepen. Wat belangrijk is bij het zoeken op Id, is dat ACID gegarandeerd wordt door RavenDB bij deze manier van zoeken. Dit komt omdat er geen indexen bij komen kijken. Een voorbeeld van het zoeken op Id is:

```
public T GetSingle<T>(string id)
{
	using (var session = DatabaseConnection.OpenSession("RRWeb_POC"))
	{
		return session.Load<T>(id);
	}
}
```

Het is overigens ook mogelijk te zoeken op meerdere Id's tegelijk. Het enige verschil is dan dat er een string[] meegegeven wordt als parameter en er een list terug wordt gestuurd.

2.1.3.2 Zoeken op attributen

Voor het zoeken op attributen gelden er echter een aantal andere dingen. Het is namelijk zo, dat bij het zoeken op attributen geen ACID maar BASE geld. Het is dus mogelijk om een oudere versie van een document terug te krijgen bij het query'en, of dat net toegevoegde documenten nog niet opgehaald worden met de query. Dit komt omdat deze documenten mogelijk nog geïndexeerd moeten worden, of bezig zijn met indexeren op het moment van zoeken. Voor het zoeken op attributen moet namelijk een index gemaakt zijn op de attributen waarop gezocht wordt.

Omdat niet alle klassen dezelfde attributen hebben, is het ook niet mogelijk om het zoeken op attributen generiek te maken.

Verder is het zo dat het zoeken op attributen gebeurd door middel van LINQ query's.

2.2 Mijn ervaring

In de afgelopen paar weken heb ik functionaliteiten uit de rooster module nagebouwd, gebruik makend van RavenDB. In deze paragraaf bespreek ik hoe dit mij is bevallen vanuit ontwikkel oogpunt en vanuit databasebeheer oogpunt.

2.2.1 Prettig:

Wat zeer prettig is aan het werken met RavenDB is de mogelijkheid om complexe query's op te kunnen stellen met LINQ. De reden dat dit prettig is, is omdat LINQ een vrij bekende syntax is om door lijsten heen te zoeken in C#. Het is echter niet zo dat dit alleen mogelijk is met RavenDB. Een aantal OR-Mappers zoals Entity Framework bieden deze mogelijkheid namelijk ook. Dit is dus niet specifiek een reden om over te stappen naar RavenDB.

Wat wel mogelijk een reden is om over te stappen, is het feit dat het in RavenDB, net als in andere document databases, mogelijk is de data op te slaan zoals het in de applicatie gebruikt wordt. Aangezien er binnen de applicatie op verschillende plaatsen de database genormaliseerd is, terwijl de applicatie het gedenormaliseerd gebruikt, denk ik dat hier een reden kan liggen om over te stappen naar document databases.

Een aansluitende reden om over te stappen naar RavenDB, is het feit dat het in RavenDB mogelijk is documenten en references(documenten die niet embed zijn, maar naar verwezen wordt) op te halen in dezelfde query. Hierdoor kan voorkomen worden dat data wat vaak wijzigt, constant op veel plaatsen gewijzigd moet worden. Dit is geen vanzelfsprekend fenomeen binnen document databases.

Een andere punt uit RavenDB wat niet vanzelfsprekend is binnen document databases, is de aanwezigheid van ACID. Door de mogelijkheid te bieden tot consistente data, kan de gebruiker ervoor kiezen waar consistentie belangrijk is, en waar het minder belangrijk is.

Een laatst punt wat ik als prettig ervaren heb, is de C# library die RavenDB aanbied. De library werkt zoals verwacht, en is (naar mijn mening) vrij eenvoudig te leren(zie voorbeelden in paragraaf 2.2). De bijbehorende documentatie is ook helder met voldoende voorbeelden en uitleg.

2.2.2 Minder prettig:

Er zijn 2 punten die ik als wat minder prettig heb ervaren bij het gebruik van RavenDB. De eerste geld in principe voor alle document databases, namelijk het wijzigen/bijwerken van documenten die op meerdere plaatsen opgeslagen/embed is. Op het moment dat een medewerker op 2 roosters voorkomt, moet deze dus ook op 2 roosters geüpdatet worden. Hier moet constant rekening mee gehouden worden gedurende het ontwikkelen van de applicatie.

Een ander punt wat ik als minder prettig heb ervaring is de eventual consistentie op het moment dat er gequeried wordt. Hoewel het prettig is dat RavenDB je de mogelijkheid biedt het op beide manieren te doen, is de meest voor de hand liggende oplossing meestal het query'en op attributen. Nu is het zo dat je hierbij niet altijd de meest recente documenten terugkrijgt, zoals eerder uitgelegd. Ik kwam er zelf echter later pas achter dat het bij het gebruik van Id's wel consistent is, en heb hier dus ook geen problemen meer mee ondervonden.

3. Conclusie

Na het uitvoeren van het onderzoek, de pakketselectie en het bouwen van de proof of concept ben ik tot een aantal conclusies gekomen. Deze worden in dit hoofdstuk besproken.

3.1 Welke modules kunnen over op document databases

Van de 4 grote modules uit R&R-web ben ik ervan overtuigd dat de volgende modules kunnen overstappen naar document databases:

- Rooster module
- Realisatie
- Prognose
- Uitbetaling

3.1.1 Rooster module

Wat zowel in het onderzoek, als in de proof of concept naar voren is gekomen, is het rooster eigenlijk niks meer is dan geaggregeerde data. Een groot deel van deze data, komt ook alleen voor op roosters(bijvoorbeeld diensten van een medewerker). Deze data kan prima in 1 document opgeslagen worden, zoals in het proof of concept te zien is.

Dat het in 1 document kan worden opgeslagen helpt ook bij functionaliteiten als het toevoegen/wijzigen van 1 dienst aan 1 medewerker binnen 1 periode. Op dit moment worden namelijk alle diensten van die medewerker binnen een periode uit de database gehaald, en wordt de nieuwe situatie opgeslagen, omdat het toevoegen/wijzigen van 1 dienst invloed kan hebben op de andere diensten van de medewerker binnen die periode(oftewel de rest van de diensten kunnen ook wijzigen op basis van de wijziging op de ene). Dit is niet nodig als het allemaal 1 document is, omdat er dan maar 1 document geüpdatet hoeft te worden, omdat alle diensten zich in dit document bevinden.

3.1.2 Realisatie

De realisatie is in principe de data uit de rooster module, met daarbij de daadwerkelijk gemaakte uren. Dit betekent dat er ook hier sprake is van overbodig genormaliseerde data(dezelfde data als het rooster). Hier kan ook prima 1 document van gemaakt worden.

3.1.3 Prognose

Voor de prognose geld, dat in ieder geval de output in een document database geplaatst kan worden. Hierbij moet enkel gedacht worden aan de output die gebruikt wordt in de volgende modules(bijvoorbeeld het budget per taak in een afdelingsrooster en de benodigde productieve uren per uur per dag).

De reden dat deze data in een document database geplaatst kan worden, is dat het gebruikt wordt in onder andere de rooster module. Dit is dus data die ook in het rooster embed kan worden.

3.1.4 Uitbetaling

De uitbetalingsmodule kan ook overgezet worden naar document databases. De reden dat ik dit zeg, is dat ook de roostermodule bestaat uit geaggregeerde data.

3.1.5 Afstappen van SQL-Server

Dat alle modules kunnen overstappen betekent echter niet dat er afscheid genomen kan worden van SQL-Server. Ik weet namelijk niet of er vergelijkbare tools als Power BI beschikbaar zijn voor RavenDB. Daar er ook bepaalde analyses worden uitgevoerd op basis van de data uit de applicatie, moet er gekeken worden of hier mogelijkheden voor zijn in RavenDB.

3.2 Wel of niet overstappen naar RavenDB

Zoals in hoofdstuk 2 te zien is RavenDB een prima database om mee te werken als ontwikkelaar, en voldoet aan alle eisen en wensen van de Vries WFM. Ik ben echter niet van mening dat er op dit moment overgestapt kan worden naar RavenDB. Er zijn namelijk nog een aantal punten waarnaar gekeken moet worden voordat er overgestapt kan worden. Deze punten worden behandeld in paragraaf 4.1.

4. Aanbevelingen/advies

Naar aanleiding van de conclusie en de proof of concept, raadt ik de medewerkers van de Vries WFM aan om nog een aantal punten te onderzoeken voordat zij wel of niet overstappen. Mochten zij besluiten over te stappen, heb ik ook nog een aantal adviezen/aandachtspunten. Deze adviezen/aandachtspunten gaan over zowel het werken met document databases en het werken met RavenDB.

4.1 Vervolg onderzoek

Zoals eerder gezegd raad ik de medewerkers van de Vries WFM aan om een vervolgonderzoek te doen op mogelijkheden van RavenDB. In dit onderzoek moet gekeken worden naar de stabiliteit en betrouwbaarheid van RavenDB op de productie omgeving. In dit onderzoek moet in ieder geval aan het volgende gedacht worden:

- De betrouwbaarheid en stabiliteit van RavenDB onder verschillende belastingen(load, piek, duur enz.)
- Hoe gaat RavenDB om met concurrency
- Hoe gaat RavenDB om met Sharding*
- Eventueel punten als monitoring en andere criteria uit de pakketselectie die niet zijn verwerkt.
    - Denk ook aan de aanwezigheid van tools om de data te analyseren(zoals Power BI gebruikt wordt in SQL-Server).

*Sharding is het verspreiden van 1 database(let op, niet 1 DBMS maar 1 database in een DBMS) over meerdere servers. Dit zorgt ervoor dat er minder krachtigere servers(maar wel meer servers) nodig zijn om dezelfde hoeveelheid data te verwerken. Ook zorgt dit ervoor dat je oneindig schaalbaar bent, omdat er geen limiet is aan het aantal servers dat gebruikt wordt.

Nadat de bovenstaande punten zijn onderzocht, kan de Vries WFM bepalen of zij willen overstappen naar RavenDB of niet.

4.2 Advies

Op het moment dat er besloten wordt om over te stappen naar een document database/RavenDB moet er rekening gehouden worden met een aantal dingen. Hieronder wordt aangegeven waar rekening mee gehouden moet worden bij het gebruik van document databases en bij het gebruik van RavenDB.

4.2.1 Werken met document databases

De volgende adviezen heb ik over het werken met document databases:

4.2.1.1 Embedded documents

Zoals ik al eerder heb aangegeven, moet er rekening gehouden worden met data wat op meerdere plaatsen staat opgeslagen(embedded documents bijvoorbeeld). Deze moet via de applicatie op alle plaatsen bijgewerkt/verwijderd worden op het moment dat er 1 wordt gewijzigd/verwijderd.

Hier moet ook rekening mee gehouden worden bij het beheren van de database. Om te voorkomen dat je handmatig data in 1 document aanpast, maar het vergeet in een andere document aan te passen, kunnen er een aantal dingen gedaan worden:

1. Een kleine databasebeheer applicatie schrijven voor update en delete handelingen
2. Updates en deletes uitvoeren via de live/testomgeving,
    - De database wijzigen vanuit de applicatie aangezien de applicatie het wel goed moet doen

4.2.1.2 Embedding vs. Reference

Er moet ook gekeken worden naar wat te embedden en wat te referencen.

- Wat te reference en wat te embedden?
    - Dienstverbanden (Employments) kunnen beter ge-referenced worden, omdat deze data(bijvoorbeeld verlof mutaties) bevatten die regelmatig wijzigen
    - Diensten op een rooster kunnen embed worden, omdat deze op een beperkt aantal roosters voorkomen

Voor output van de ene module in de andere module zijn 2 mogelijkheden:

1. Schrijf een tussenlaag welke controleert of er al een document is om in te embedden
    a. Als bijvoorbeeld de prognose voor het rooster wordt gemaakt → kijken of er een rooster bestaat om in te embedden
    b. Als er een rooster wordt gemaakt zonder prognose → kijken of er een prognose bestaat om in te embedden
2. De output in 1 document stoppen, en deze reference door de volgende module
    a. De output van prognose in prognose output document stoppen, als er een rooster wordt gemaakt kijken of er een output is voor deze periode, en deze vervolgens reference
        i. bij het maken van een prognose en het maken van een rooster moet gekeken worden of de ander bestaat, om de id te achterhalen

Het voordeel van de 1e mogelijkheid is, dat de volledige documents embed worden, dit betekent dat er maar 1 query nodig is om het geheel op te halen. Het betekent echter wel dat een wijziging in de prognose module ook een wijziging in de rooster module kan betekenen.

Het voordeel van de 2e mogelijkheid is, dat de modules onafhankelijk van elkaar zijn. Als de prognose wijzigt, hoeft de roostermodule hier niks van te weten. Dit omdat hij ernaar refereert, en dus niet zelf geüpdatet hoeft te worden.

4.2.1.3 Samenwerken met verschillende teams

Omdat de structuur van de database hetzelfde is als de structuur van de applicatie, is het belangrijk dat elke applicatie/component die met de database communiceert dezelfde namen gebruiken voor dezelfde entiteiten/attributen. Dit kan op een aantal manieren afgedwongen worden, namelijk:

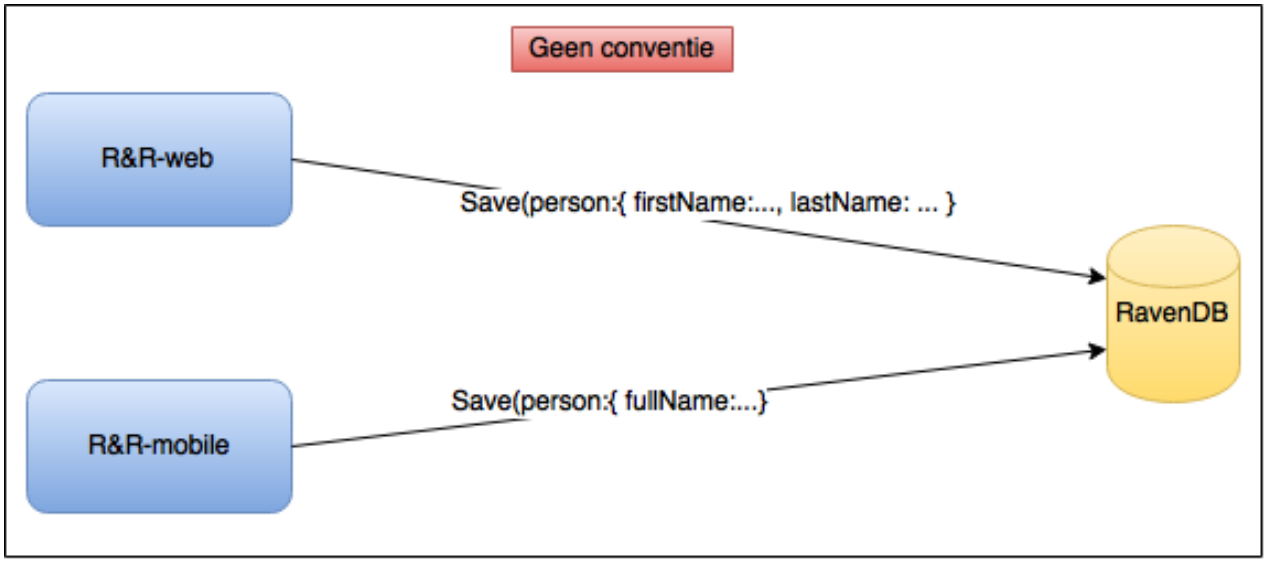
- Er worden tussen de verschillende teams/applicaties duidelijke naam conventies opgesteld. Oftewel wordt gezamenlijk een klassendiagram/ERD/EER gemaakt waarin precies staat hoe alles in de database moet heten(en dus ook hoe de models in applicatie eruit moeten zien)
- Er wordt 1 centrale applicatie gemaakt welke praat met de database, alle andere applicaties praten met deze applicatie d.m.v. bijvoorbeeld http.
- Er is 1 team wat bepaald hoe alles eruit gaat zien, en de rest kopieert dit.

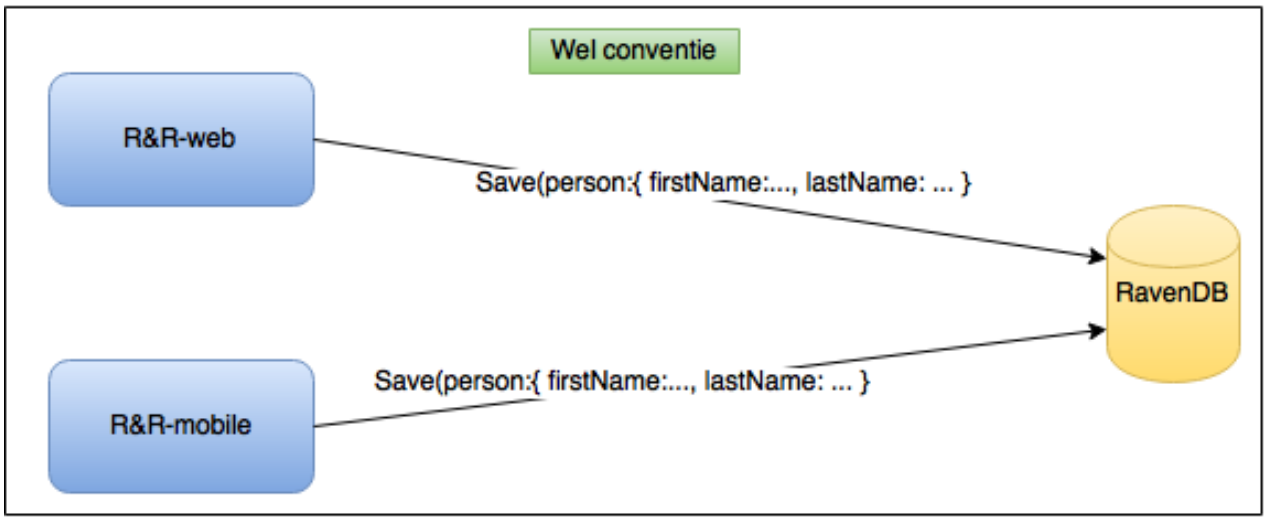
Hieronder wordt van de eerste 2 manieren een korte uitleg met voorbeelden gegeven:

Naam conventies:

Omdat je wilt voorkomen dat er verschillende collections komen die eigenlijk hetzelfde zijn, of dat de structuren van de documents binnen een collection afwijken per applicatie moeten er bepaalde conventies over de database gemaakt worden.

Deze conventies kunnen bijvoorbeeld gemaakt worden door de vertegenwoordigers van de team.
In de onderstaande afbeelding is te zien wat er mis kan gaan zonder conventies, en hoe het eruit ziet met conventies. Let erop dat in plaats van RavenDB elk andere document database ingevuld kan worden.

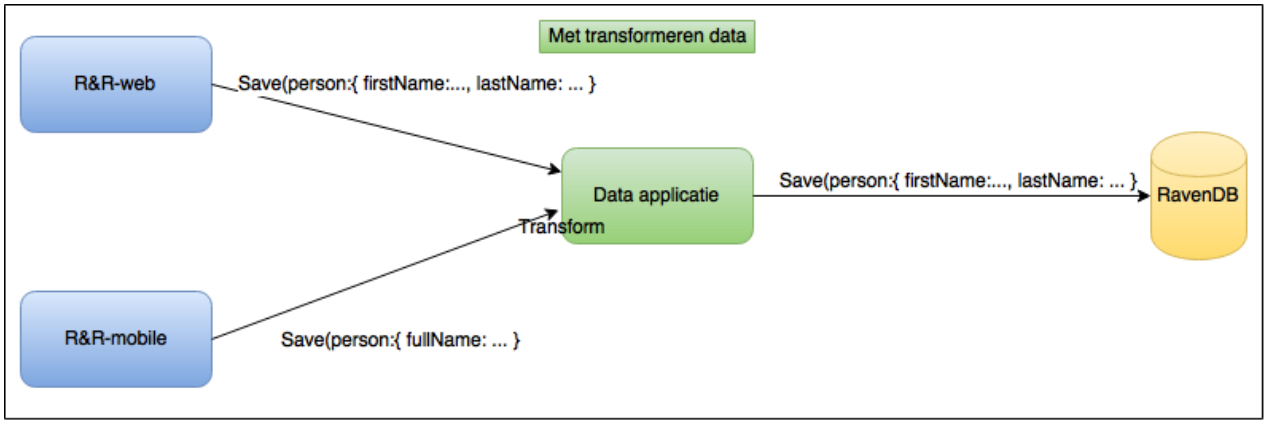


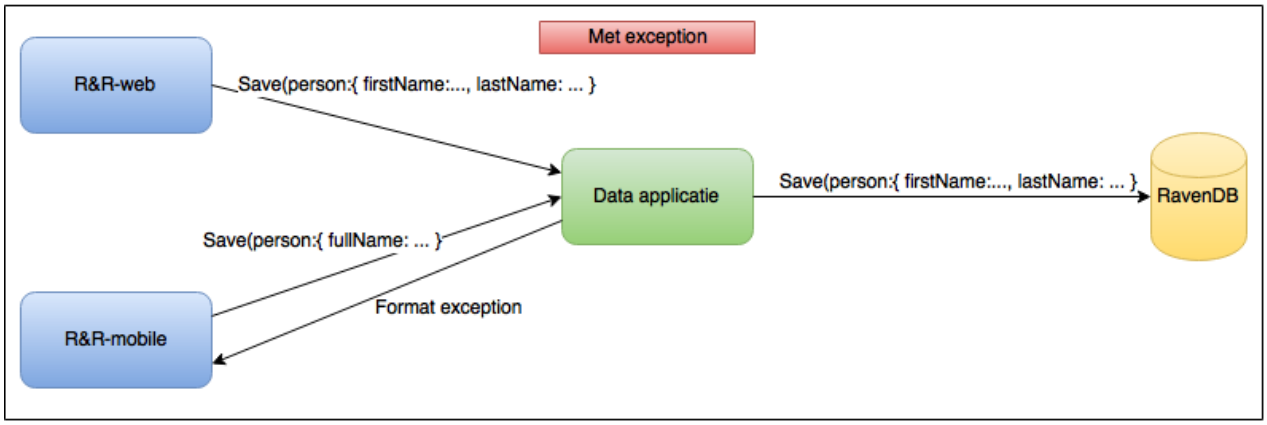


Centrale database applicatie:

Door een centrale applicatie te maken waar alle communicatie met de database zich bevind(de datalaag als een aparte applicatie) kan ook afgedwongen worden dat de database niet vervuild/rommelig wordt door verschillende structuren van documents. De centrale applicatie bepaald dan hoe de documents er in de database uit zullen zien, en valideert of binnenkomende requests aan dit formaat voldoen. Voldoet een binnekomende request niet aan dit formaat, dan kan de applicatie een exception gooien of de data transformeren naar het juiste formaat(afhankelijk van hoe het geïmplementeerd is). Verder kunnen ook constraints en/of validaties in deze applicatie plaatsvinden.

In de onderstaande afbeelding is te zien hoe dit eruit zou zijn met een centrale database applicatie(zowel met formateren als met een exception). Let erop dat in plaats van RavenDB elk andere document database ingevuld kan worden.





4.2.2 Werken met RavenDB

De volgende adviezen heb ik over het werken met RavenDB:

4.2.2.1 Load vs. Query

Zoals meerdere malen in dit document genoemd is het bij RavenDB belangrijk dat er gekeken wordt naar welke reads de meest recente versie van een object/de meest recente lijst van objecten nodig zijn. Voor deze reads dient de Load methode gebruikt te worden(zoeken op basis van Id). Voor alle andere reads kan de query methode gebruikt worden. Een voorbeeld om dit te doen in de rooster module is:

- Bij het tonen van de afdelingen die bestaan binnen een periode binnen een winkel worden de id's van afdelingsroosters behorend bij de afdelingen ook opgehaald.

- Haal met angularjs het id van de geselecteerde rooster op(allemaal op de front end)
    - Op basis van de geselecteerde afdeling
- Stuur deze naar de server
- Gebruik de load methode

Bij iets als selecteren van Periodes die van het type WEEK zijn is het minder van belang dat de volledigste lijst met periodes consistent is. Als iemand net alle periodes van 2017 toevoegt, is het niet erg als deze niet meteen geselecteerd kunnen worden.

4.2.2.2 Aanmaken van indexen

Het is in RavenDB mogelijk om indexen dynamisch te laten aanmaken, op basis van de zoek-criteria. Dit betekent dat op het moment dat een query wordt uitgevoerd, en de gezochte attributen zijn niet geïndexeerd, RavenDB zelf een index aanmaakt. Hoewel dit handig is als je niet zeker bent of er een index bestaat, of als je geen zin hebt om de index te schrijven, heeft dit ook een nadeel.

Het nadeel wat dynamische indexen met zich meebrengen is namelijk: op het moment dat de index wordt aangemaakt, moeten ook alle documenten langsgelopen worden om te kijken of zij geïndexeerd moeten worden. Op het moment dat er een grote database is, betekent dit dat het aanmaken van de index en indexeren van de documents tijd kan kosten. De query kan pas uitgevoerd worden op het moment dat de index klaar is met aanmaken/indexeren. Oftewel de eerste uitvoering van de query is traag, en kan zelf een exception gooien omdat de te gebruiken index te lang stale is.

Mijn advies is dan ook om op de productie omgeving alle indexen zelf toe te voegen, en hierop te query'en. Maak zo min mogelijk gebruik van dynamische indexen, om te voorkomen dat er zich onverwachts gedrag zal voordoen. Het maken van de indexen kan via de studio of via de applicatie. Aangezien meerdere klanten gebruik maken van dezelfde R&R-web applicatie, maar verschillende databases, adviseer ik dan ook de indexen in de applicatie te schrijven, en te laten deployen vanuit de applicatie. Zo weet je zeker dat alle databases waar de indexen hebben.

4.2.2.3 Includes

Op het moment dat je references gebruikt, en je hebt het gehele document(met de documents waarnaar gereferenced wordt) nodig, maak gebruik van de include methode. Dit voorkomt dat er meerdere query's naar de database nodig zijn.

Bijlage G: Afstudeerplan

[Onderwerp]

Xavyr Rademaker

Afstudeerplan

Informatie afstudeerder en gastbedrijf (*structuur niet wijzigen*)

Afstudeerblok: 2016-1.1 (start uiterlijk 8 februari 2016)
Startdatum uitvoering afstudeeropdracht: 1 februari 2016
Inleverdatum afstudeerdossier volgens jaarrooster: 3 juni 2016

Studentnummer: 11077581
Achternaam: dhr. Rademaker (*) *weghalen niet van toepassing*
Voorletters: X.T.
Roepnaam: Xavyr
Adres: Winkelstede 9
Postcode: 2543 BM
Woonplaats: Den Haag
Telefoonnummer: 0640166275
Mobiel nummer: 0640166275
Privé emailadres: xavyr.070@live.com

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

Naam studieloopbaanbegeleider: A. Wieman
Naam begeleidend examinator: G. Mijnarends
Naam tweede examinator: J.J. van der Hoek

Naam bedrijf: Info Support
Afdeling bedrijf:
Bezoekadres bedrijf: Kruisboog 42
Postcode bezoekadres: 3905 TG
Postbusnummer:
Postcode postbusnummer:
Plaats: Veenendaal
Telefoon bedrijf: +31 318 552020
Telefax bedrijf: +31 318 552355
Internetsite bedrijf: http://www.infosupport.com/

Achternaam opdrachtgever: dhr. Megens (*) *weghalen niet van toepassing*
Voorletters opdrachtgever: R.L.W.M.
Titulatuur opdrachtgever: ing.
Functie opdrachtgever: Ontwikkelaar/ontwerper
Doorkiesnummer opdrachtgever: n.v.t.
Email opdrachtgever: rick.megens@devrieswfm.com

Achternaam bedrijfsmentor: mw. Keurntjes (*) *weghalen niet van toepassing*
Voorletters bedrijfsmentor: M.
Titulatuur bedrijfsmentor:
Functie bedrijfsmentor: Afstudeercoördinator / Procesbegeleider
Doorkiesnummer bedrijfsmentor:
Email bedrijfsmentor: marieke.keurntjes@infosupport.com
NB: bedrijfsmentor mag dezelfde zijn als de opdrachtgever

Doorkiesnummer afstudeerder:
Functie afstudeerder (deeltijd/duaal):

Titel afstudeeropdracht:

Onderzoeken welke document-oriented database past bij R&R-web bij Info Support

Opdrachtomschrijving *(toelichtende tekst verwijderen)*

1. Bedrijf

Info Support is een bedrijf wat zich onder andere bezighoudt met het ontwikkelen van op maat gemaakte software voor bedrijven in verschillende branches. Deze branches zijn:

- Financiële dienstverlening
- Woningcorporaties
- Overheid
- Zorg en verzekering
- Decentralisaties
- Pensioenfondsen
- Industrie en energie
- Vervoer

Naast het ontwikkelen van software, houdt Info Support zich ook bezig met het beheren van software en het opleiden van zowel medewerkers als software ontwikkelaars van andere bedrijven.

Bij het ontwikkelen van de software maakt het bedrijf gebruik van de nieuwste technieken. Verder wordt deze software op projectbasis aangenomen, waarvan het beheer eventueel ook door het bedrijf gedaan wordt. Het bedrijf heeft verder ook een afdeling die zich bezighoudt met roosterapplicatie genaamd R&R web.

2. Probleemstelling

De huidige applicatie(R&R-web) maakt gebruik van een SQL-server database. Met de opkomst van nieuwe vormen van databases, waaronder document-oriented databases, vraagt de opdrachtgever zich af of deze nieuwe technieken gebruikt kunnen worden binnen de bestaande applicatie. Om deze keuze te kunnen maken, wil hij weten welke voor- en nadelen deze nieuwe vorm van databases heeft en hoe deze database passen bij het huidige systeem.

3. Doelstelling van de afstudeeropdracht

Het doel van de opdracht is het onderzoeken welke document-oriented database gebruikt kan worden bij de R&R applicatie, en de conclusie aantonen door middel van een proof of concept.

4. Resultaat

Op het moment dat de opdracht succesvol is afgesloten, is het bedrijf in staat om een afweging te maken om over te stappen op document oriented databases binnen (een gedeelte van) de huidige applicatie. Dit kunnen zij vervolgens invoeren.

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

Gedurende de afstudeeropdracht wordt er verwacht dat ik de voor –en nadelen van documenten-oriented databases ten opzichte van relationele databases in kaart breng. Vervolgens moet ik de huidige applicatie analyseren en, in combinatie met de resultaten uit voor –en nadelen, beslis in welke modules een document-oriented database gebruikt kan worden, en vooral ook welke implementatie van document-oriented databases het bedrijf zou moeten gebruiken. Als dit in kaart is gebracht, wordt er van mij verwacht dat ik de resultaten aantoon door middel van een proof of concept. De proof of concept zal gebaseerd worden op de functionaliteit van 1 of meerdere modules uit de huidige applicatie en geschreven worden in C#.

Om deze opdracht te kunnen realiseren zullen de volgende activiteiten uitgevoerd moeten worden:

Activiteit	**Duur(dagen)**	**Opmerkingen**
Schrijven plan van aanpak	2.5	Schrijven van PVA, valt samen met verkennen bedrijf.
Verkennen bedrijf	2.5	Het bedrijf en de betrokken stakeholders (in ieder geval de opdrachtgever) leren kennen.
Uitzoeken voor- en nadelen document oriented databases t.o.v. relationele databases	5	Selecteren bronnen, lezen bronnen en resultaten verwerken in rapport.
Onderzoeken welke document-oriented database het best bij de applicatie past	10	Uitvoeren pakketselectie. Schatting is voor eerste versie. Zal waarschijnlijk bij vervolgstappen ook nog overlap hebben.
Analyseren applicatie, kijken waar document oriented databases gebruikt kunnen worden	10	Tijd schatting, nog geen idee hoe groot applicatie is.
ontwerpen proof of concept	3	Ontwerpen applicatie proof of concept. Zal veel raakvlak hebben met ontwerpen database proof of concept.
Ontwerpen database proof of concept	7	Ontwerpen db incl. opslagstructuur bepalen
Bouwen proof of concept	20	Applicatie proof of concept & implementatie geselecteerde db
Testen proof of concept	10	Voor -en nadelen geselecteerde db naar voren laten komen in test t.o.v. SQL server (aantonen conclusie)
Opbouwen afstudeerdossier	15	

Bij het uitvoeren van de bovenstaande activiteiten worden de volgende technieken gebruikt:

- Agile, het bedrijf werkt volgens agile ontwikkelmethodieken, welke techniek er gebruikt wordt, wordt aan de start van de opdracht bepaald

- Document-oriented databases
- Pakketselectie
- C#, voor het bouwen van de proof of concept
- UML, voor het ontwerpen van de proof of concept
- UML/EER voor het ontwerpen van de database
- Unit tests om kwaliteit van de proof of concept vast te stellen
- Verschillende tests bij het selecteren van de database, en het aantonen van de resultaten omtrent keuze database

6. Op te leveren (tussen)producten

De volgende producten zullen gedurende mijn afstudeerperiode opgeleverd worden aan het bedrijf:

- Plan van aanpak
- Rapport voor- en nadelen document oriented databases t.o.v. relationele databases.
- Rapport welke document oriented database te gebruiken(onderzoeksrapport)
- Rapport bij welke modules kunnen document oriented databases gebruikt worden.
    - Inclusief advies of het verstandig is om over te stappen
- Ontwerp proof of concept
- Gebouwde proof of concept
- Testverslagen proof of concept

7. Te demonstreren competenties en wijze waarop

De volgende beroepstaken zal ik tijdens mijn afstudeer opdracht uitvoeren:

1.3 Selecteren van standaardsoftware, niveau 3: zelfstandig en lastig

Niveau 3 = *Het betreft het selecteren van één of meerdere applicatie(componenten) voor een simpel bedrijfsproces. Er is een 'longlist' beschikbaar of eenvoudig samen te stellen, maar de selectiecriteria zijn dubbelzinnig en spreken elkaar soms tegen.*

Gedurende de afstudeeropdracht zal er een pakketselectie uitgevoerd worden. Deze zal gaan over welke document-oriented database gebruikt zal moeten worden binnen de huidige applicatie. Hierbij moet rekening gehouden worden met de modules die geschikt zijn voor document-oriented databases en de functionaliteiten van deze modules. Ook moet er rekening gehouden worden met de criteria die aan de database gesteld worden.

2.2 Ontwerpen, bouwen en bevragen van een database, niveau 4: zelfstandig en complex

niveau 4 = *Het betreft het ontwerpen, bouwen en bevragen van een database die gebruikt gaat worden door vele verschillende groepen gebruikers. De beschikbaarheid van data moet gewaarborgd blijven en de performance is een kritische factor. Daarnaast wordt de kwaliteit van de data bewaakt d.m.v. in het DBMS beschikbare functionaliteiten.*

Gedurende de afstudeeropdracht zal er bij de proof of concept een database gebouwd moeten worden. Deze moet ontworpen, geïmplementeerd en bevraagd worden. Het bevragen zal gaan op verschillende manieren, aangezien de database wordt opgebouwd met verschillende structuren. Bij het bouwen en bevragen van de database moet rekening gehouden worden met performance en eventueel met concurrency.

3.3 Bouwen applicatie niveau 3: zelfstandig en lastig

Niveau 3 = *Het betreft het bouwen van een objectgeoriënteerde applicatie, waarbij geavanceerde*

concepten van de gebruikte programmeertaal aan de orde komen. Verder wordt rekening gehouden met toekomstige wijzigingen, testbaarheid en hergebruik. Het bouwen gebeurt in een ontwikkelomgeving.

Het te bouwen proof of concept zal OO geprogrammeerd worden. Verder wordt er gebruik gemaakt van geavanceerde concepten zoals lambda en LINQ.

3.5 Uitvoeren van en rapporteren over het testproces niveau 3: zelfstandig en lastig

Niveau 3 = *Bij het opstellen van het logisch testontwerp wordt gebruik gemaakt van een testontwerptechniek. Er is aandacht voor herhaalbaarheid van de testen. Het betreft hoofdzakelijk het testen van de functionaliteit. Testrapportage betreft het volledige systeem.*

De proof of concept zal uiteindelijk getest moeten worden. Hierbij moet de applicatie zelf getest worden om de kwaliteit van het proof of concept aan te tonen. Hierbij zullen er logische en fysieke testgevallen zijn, welke eventueel geautomatiseerd worden. Ook moet de database getest worden op de gestelde criteria om zo de voor en nadelen van de geselecteerde database aan te tonen. Deze tests zullen gebruikt worden om de conclusie uit de volgende rapporten hard te maken:

- voor –en nadelen document-oriented databases
- pakketselectie welke document-oriented database past bij de applicatie
- welke modules kunnen document-oriented databases gebruiken.