Visual Deployment Pipeline

Bijlage

M.R. de Meza, 10043659

Info Support
Kruisboog 42
3905 TG Veenendaal
Tel. +31(0)318 - 55 20 20
Fax +31(0)318 - 55 23 55

De Haagse Hogeschool
Johanna Westerdijkplein 75
2521 EN Den Haag
Tel. +31(0)704 - 45 88 88
Fax +31(0)704 - 45 88 25

Eerste Examinator
G.A. Mijnarends

Tweede examinator
J.J. van der Hoek

Opdrachtgever
P. Borgeld

Technisch begeleider
F. Sedney

Proces begeleider
L. van de Hoef

Datum
03-06-2016

Inleiding

Dit document bevat de alle bijlagen van de opdracht Visual Deployment Pipeline. Deze opdracht is uitgevoerd bij Info Support door Marc de Meza, student aan De Haagse Hogeschool.

Bijlage I Plan van Aanpak

Bijlage II Analyse mogelijkheden ophalen gegevens uit MRM

Bijlage III Functioneel ontwerp

Bijlage IV Technisch ontwerp

Bijlage V Handleiding

Bijlage VI Afstudeerplan

Bijlage I

Plan van aanpak

Plan Van Aanpak

Visual deployment pipeline

Plan van aanpak

Titel	Plan van aanpak
Project/Onderwerp	Visual deployment pipeline
Versie	1.9
Status	Concept
Datum	02-02-2016
Bestand	Plan van aanpak 2016 Marc de Meza.docx
Bedrijf	Info Support

Info Support B.V. Kruisboog 42, 3905 TG, Veenendaal, Tel. +31 (0) 318 55 20 20 - Fax +31 (0) 318 55 23 55
Kenniscentrum De Smalle Zijde 39, 3903 LM Veenendaal, Tel. +31 (0) 318 50 11 19 - Fax +31 (0) 318 51 83 59

Historie

Versie	Status	Datum	Auteur	Verandering
1.0	Concept	02-02-2016	M.R. de Meza	Creatie
1.1	Concept	05-02-2016	M.R. de Meza	Feedback Felix Sedney 04-02-2016
1.2	Concept	09-02-2016	M.R. de Meza	Feedback Paul Borgeld 08-02-2016
1.3	Concept	10-02-2016	M.R. de Meza	Feedback Felix Sedney 10-02-2016
1.4	Concept	15-02-2016	M.R. de Meza	Feedback Paul Borgeld gesprek 10-02-2016
1.5	Concept	15-02-2016	M.R. de Meza	Feedback Laila van de Hoef 15-02-2016
1.6	Concept	16-02-2016	M.R. de Meza	Feedback Felix Sedney 16-02-2016
1.7	Concept	17-02-2016	M.R. de Meza	Feedback Paul Borgeld 17-02-2016
1.8	Concept	10-03-2016	M.R. de Meza	Planning en tooling aanpassen
1.9	Definitief	21-03-2016	M.R. de Meza	Definitief maken van document met toestemming van Felix Sedney

Distributie

Versie	Status	Datum	Aan
1.0	Concept	04-02-2016	Felix Sedney, Paul Borgeld en Laila van de Hoef
1.1	Concept	08-02-2016	Felix Sedney, Paul Borgeld en Laila van de Hoef
1.4	Concept	15-02-2016	Felix Sedney, Paul Borgeld en Laila van de Hoef
1.8	Concept	10-03-2016	Felix Sedney, Paul Borgeld en Laila van de Hoef

Referenties

Code	Bron
IN01	http://rodymiddelkoop.blogspot.nl/2010/05/how-to-not-manage-one-person-software.html
IN02	Beroepstaken_Informatica_1.1.pdf

Management samenvatting

Voor Info Support moet er een analyse worden gedaan naar de mogelijkheden om op basis van Microsoft Release Management een dashboard te realiseren. In dit dashboard moeten overzichten van software releases grafisch worden weergegeven.

Tijdens de opdracht zal er met Scrum als ontwikkelmethodiek gewerkt worden. Om voortgang te bewaken zullen er diverse documenten opgeleverd worden gedurende het proces. Deze documenten zijn het functioneel- en technisch ontwerp. Eventueel worden er ook test documenten opgeleverd.

Inhoudsopgave

1 Opdracht

1.1 Opdrachtgever en opdrachtnemer

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

Info Support

Adres	Kruisboog 42
Postcode	3965 TG Veenendaal
Telefoon	+31 318 552020
Faxnummer	+31 318 552355

Contact persoon

Naam	Marc de Meza
Email	marc.demeza@infosupport.com
Telefoon	+31 651 123848
School	De Haagse Hogeschool

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

Info Support

Adres	Kruisboog 42
Postcode	3965 TG Veenendaal
Telefoon	+31 318 552020
Faxnummer	+31 318 552355

Contact Persoon

Naam	Paul Borgeld
Email	paul.borgeld@infosupport.com

Haagse Hogeschool

Adres	Johanna Westerdijkplein 75
Postcode	2521 EN Den Haag
Postbus	1336
Telefoon	+31 704 458888
Faxnummer	+31 704 458825

Eerste Examinator

Naam	G.A. Mijnarends
Email	g.a.mijnarends.hhs.nl
Telefoon	+31 686 805976

Tweede Examinator

Naam	J.J. van der Hoek
Email	j.j.vanderhoek.hhs.nl
Telefoon	+31 686 805945

1.2 Opdrachtdefinitie

Aanleiding
Sinds Team Foundation Server 2013 heeft Microsoft het mogelijk gemaakt om geautomatiseerd applicaties te kunnen installeren door gebruik te maken van Microsoft Release Management. In Microsoft Release Management is het mogelijk om een deploymentstraat te definiëren. Vervolgens kan per omgeving beschreven worden hoe de software op die omgevingen geïnstalleerd moet worden. Tijdens en na de installatie kan er gekeken worden hoe de installatie is verlopen en welke stappen zijn uitgevoerd.

Probleemstelling
Microsoft Release Management geeft momenteel geen mogelijkheid om in één oogopslag te zien welke versies van applicaties op een bepaalde omgeving zijn geïnstalleerd. Per applicatie is het ook niet duidelijk op welke omgevingen deze geïnstalleerd is.

Momenteel is er een eenvoudig dashboard gemaakt door Info Support waarin een deel van deze informatie wordt weergegeven. Dit dashboard geeft nog lang niet alle gewenste informatie weer. Als er meer informatie nodig is moet de release manager in Microsoft Release Management hiernaar gaan zoeken.

Doelstelling
Er moet een analyse uitgevoerd worden naar de mogelijkheden om op basis van gegevens uit Microsoft Release Management een dashboard te realiseren die een deployment pipeline visualiseert. Verder moet er ook een applicatie worden geschreven waarin deze visualisatie getoond wordt. In de visualisatie van de deployment pipeline moet er per applicatie te zien zijn op welke omgevingen het is geïnstalleerd. Verder moet er in dit overzicht ook gegevens zoals versie nummers en eventueel fouten tijdens installatie getoond worden. De gegevens moeten niet alleen per applicatie getoond worden, maar ook per omgeving.

De opdrachtnemer moet van De Haagse Hogeschool ook een afstudeerverslag schrijven waarin het proces van de opdracht duidelijk beschreven is. Verder moeten beslissingen die tijdens het project genomen worden ook duidelijk beschreven worden.

Resultaat
Als de opdracht is uitgevoerd zal het bedrijf beschikken over een dashboard dat op verschillende schermgrootte en toestellen duidelijk wordt weergegeven. Dit dashboard zal in de vorm van een webapplicatie overal binnen het domein te benaderen zijn. In het dashboard zullen de diverse overzichten weergegeven worden. De informatie van de deployment pipeline, waar normaal gesproken op verschillende locaties in MRM naar gezocht moet worden, zal in één overzicht weergegeven worden.

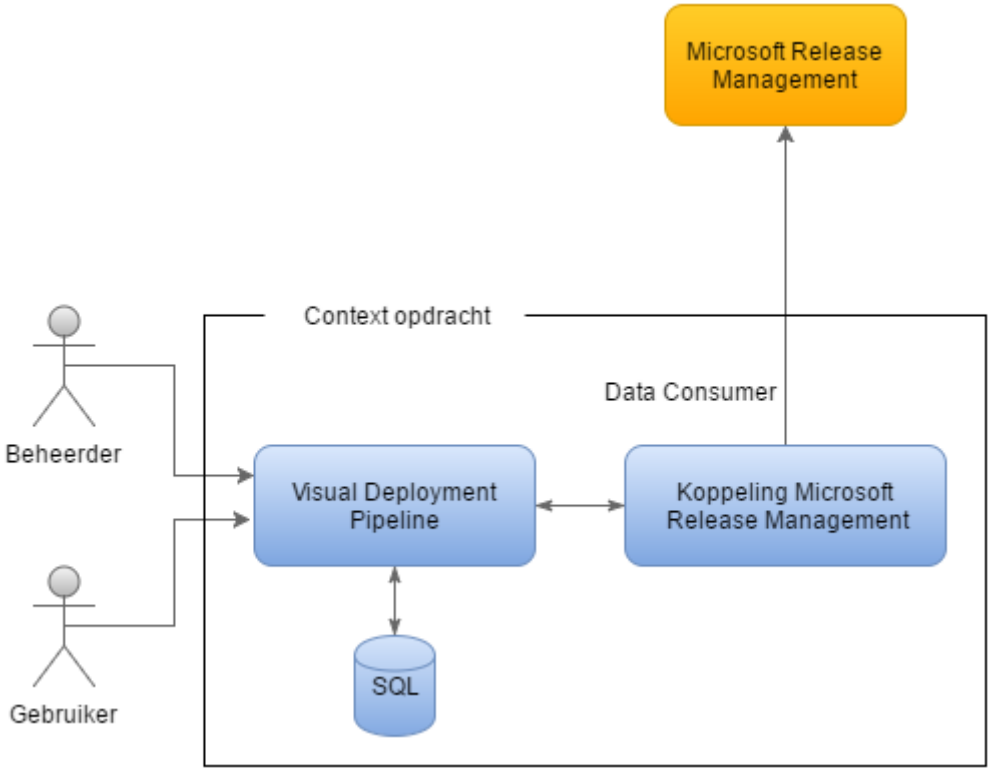
Effect
De opdracht zal ervoor zorgen dat het bedrijf tijd bespaart bij het vinden van informatie over releases. Ten slotte komt er ook meer inzicht in het deployment proces. Dit zal ervoor zorgen dat fouten eerder worden opgespoord.

1.4 Afbakening

De focus van de opdracht is:

1. Analyse naar mogelijkheden om gegevens uit Microsoft Release Management te halen om een dashboard te realiseren
2. Realiseren van het dashboard
3. Koppeling met Microsoft Release Management realiseren

In figuur 1 is een context diagram van de opdracht te zien. Hierin is te zien dat Microsoft Release Management buiten de context staat. Dit betekent dat het dashboard data vanuit Microsoft Release Management gaat verzamelen, opslaan en weergeven.



Figuur 1

Alles buiten de rechthoek in het context diagram valt buiten de scope van de opdracht. Dit betekent dat er geen wijzigingen zullen plaats vinden aan componenten buiten de context. Een opsomming van de grenzen van de opdracht zijn:

1. Het dashboard gaat alleen data weergeven van Microsoft Release Management en geen data in Microsoft Release Management aanpassen. (Alleen een consumer en geen provider)
2. Om informatie te zien van het dashboard hoeft de gebruiker zich niet te authentiseren. De informatie is voor iedereen binnen het domein toegankelijk.

1.5 Afhankelijkheden

Voor het succesvol realiseren van de opdracht zullen de opdrachtgever en technische begeleider het volgende beschikbaar moeten stellen:

Technische aspecten

1. Werkplek voor de opdrachtnemer met minimaal een werkende computer
2. Beschikking over een testomgeving met reële testdata van Microsoft Release Management
3. Op de testomgeving moet er een SQL-database aanwezig zijn
4. Op de testomgeving moet een webserver aanwezig zijn
5. Een ontwikkelomgeving met Visual Studio
6. Een ontwikkelomgeving met SQL-server
7. Team Foundaition Server/GIT

Organisatorische aspecten

1. Beschikbaarheid van de opdrachtgever
2. Beschikbaarheid van de technische begeleider
3. Trainingen voor de onderwerpen die aan bod komen bij het project:
    a. HTML5/CSS3
    b. Test Driven Development
    c. Team Foundation Server/GIT
    d. AngularJS
    e. Spec Flow
4. Beschikbaarheid van een medewerker die ervaring heeft met Microsoft Release Management

1.6 Kwaliteitseisen

Aan het dashboard zijn er een aantal kwaliteitseisen gesteld. Deze eisen zullen later gebruikt worden om acceptatiecriteria op te stellen voor het testen.

1. Bij het ophalen van nieuwe data uit Microsoft Release Management mag er geen merkbare performance verlies zijn in Microsoft Release Management
2. Op alle schermen van het dashboards moet de informatie actueel zijn
3. Het dashboard mag geen performance verlies ondervinden bij inladen van nieuwe gegevens

Vanuit school worden er aan dit project een aantal kwaliteitseisen gesteld. Deze eisen worden gemeten door het behalen van vooraf afgesproken beroepstaken. De beroepstaken moeten op een bepaald niveau gehaald worden. Per beroepstaak is aangegeven op welke niveau De Haagse Hogeschool verwacht dat de opdrachtnemer de beroepstaak behaald.

Uitvoeren analyse door definitie van requirements niveau 3
De requirements van de opdracht moeten geanalyseerd en gedefinieerd worden. De opdracht betreft één opdrachtgever, maar meerdere stakeholders. Gedurende het project zullen er ook wijzigingen in de requirements plaats vinden. De opdrachtnemer zal hiermee moeten omgaan.

Voor het definiëren van de requirements zal er gebruik worden gemaakt van "Requirements as a Scenario". Dit is een techniek waar de opdrachtnemer nog geen ervaring in heeft en dit maakt de beroepstaak daarom complexer.

Ontwerpen systeemdeel niveau 3
Er zullen drie verschillende onderdelen van het systeem ontworpen worden. Deze delen zijn de database, de applicatie en een deel van de architectuur. Er wordt gebruik gemaakt van UML.

Bouwen applicatie niveau 4

De applicatie gaat een koppeling maken met Microsoft Release Management. Deze koppeling gaat gerealiseerd worden door de opdrachtnemer. Hoe dit in zijn werk zal gaan, moet nog onderzocht worden door de opdrachtnemer.

De complexiteit van het realiseren ligt bij het maken van het dashboard waar al de informatie wordt getoond. Het betreft meerdere statistieken van Microsoft Release Management.

Uitvoeren van en rapporteren over het testproces niveau 3
Voor het testen van de gerealiseerde functionaliteit zal er gebruik gemaakt worden van Test Driven Development. Om ervoor te zorgen dat de applicatie in de toekomst gemakkelijk uitgebreid kan worden zullen alle tests geautomatiseerd zijn. Hierdoor is het mogelijk om snel te kijken of oude functionaliteit nog werkt.

1.7 Uitgangspunten

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

1. Er zal worden geprogrammeerd in Visual Studio
2. Programmeren kan alleen gebeuren op locatie bij Info Support
3. Dashboard back-end wordt gemaakt in ASP.NET
4. Er zal HTML5/CSS3 worden gebruikt voor de dashboard webclient
5. De webclient wordt gerealiseerd in Angular 1
6. Er wordt gebruik gemaakt van TDD voor het testen van risicovolle en complexe onderdelen
7. De opdrachtgever zal bepalen wanneer een risicovolle en complexe onderdeel voldoende getest is
8. Voor acceptatie tests van de back-end wordt de tool Spec Flow gebruikt
9. De webclient wordt getest met Protractor
10. De opdrachtgever wil tijdens het project gemakkelijk wijzigingen in de requirements kunnen uitvoeren.
11. Er wordt met Scrum gewerkt
12. Er zal gebruik gemaakt worden van SQL Server
13. Er zal tijd gereserveerd worden om aan het afstudeerverslag te werken
14. Het Afstudeerverslag wordt ingeleverd op 3 juni 2016
15. De Afstudeer periode is 1 februari tot 3 juni 2016

1.8 Randvoorwaarden

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

1. Trainingen indien nodig
2. Feedback van de opdrachtgever
3. Feedback van de technische begeleider

2 Risicomanagement

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	Niet voldoende kennis over team foundation server en Microsoft Release Management om de opdracht te kunnen realiseren				
	Kennis over Microsoft Release Management is niet voldoende	Aanvragen meer trainingen over Microsoft Release Management als die bestaan of Vragen aan medewerker die kennis over het onderwerp heeft	P	Afstudeerder	Zodra de sprint waar de verbinding met Microsoft Release Management dreigt achter te lopen.
	Verbinding tussen Microsoft Release Management en het dashboard duren te lang	Vragen of iemand van infosupport hier al kennis over heeft vragen om hulp	P	Afstudeerder	Zodra de sprint waar de verbinding met Microsoft Release Management dreigt achter te lopen.
2	De koppeling tussen Microsoft Release Management en het dashboard kan niet gerealiseerd worden				
	Microsoft Release Management ondersteunt gewenste functionaliteiten in volgende versie	Nieuwe gesprekken met opdrachtgever om te kijken of we de gewenste functionaliteit anders kunnen realiseren	S	Afstudeerder	Zodra er een nieuwe Release Of patch uitkomt van Microsoft Release Management die deze functionaliteit bezit
	Verbinding tussen Microsoft Release Management en dashboard is niet mogelijk	Onderzoeken of er alternatieven zijn voor het ophalen van informatie uit Microsoft Release Management	S	Afstudeerder/ Technische begeleider	Wanneer er gewerkt wordt aan de Microsoft Release Management koppeling
	Verbinding tussen Microsoft Release Management en Dashboard zal in volgende versie eruit gehaald worden	Verder zoeken naar nieuwe manieren van verbinden met Microsoft Release Management of een andere manier bedenken van registratie van deze gegevens	S	Afstudeerder	Zodra er een nieuwe Release Of patch uitkomt van Microsoft Release Management die deze functionaliteit

					breekt
3	Dashboard geeft de informatie uit Microsoft Release Management niet weer zoals gewenst				
	Opdrachtnemer heeft geen duidelijk beeld van de implementatie die de opdrachtgever in gedachte heeft	Het maken van wireframes voordat het realiseren van het dashboard begint	P	Afstudeerder	Zodra het realiseren van functionaliteit begint.
	Opdrachtnemer heeft niet voldoende kennis in HTML en CSS.	Vragen of er HTML EN CSS trainingen zijn of eventueel een medewerker die de opdrachtnemer wil ondersteunen	S	Afstudeerder	Zodra de opdrachtgever aangeeft dat hij niet tevreden is met het ontwerp
4	Opdrachtgever is niet vaak genoeg beschikbaar voor sprintreviews. Waardoor het project vertraging oploopt				
	Opdrachtgever heeft geen tijd voor sprint reviews	Opdrachtgever aangeven dat het project gevaar kan lopen en doorgeven aan procesbegeleider	P	Afstudeerder	Als de opdrachtnemer merkt dat het niet mogelijk is om een afspraak te maken met de opdrachtgever
	Opdrachtgever is onvoldoende op kantoor	Proberen af te spreken op locatie waar de opdrachtgever wel aanwezig is of opdrachtgever aangeven dat het project gevaar kan lopen en doorgeven aan proces begeleider	P	Afstudeerder	Als de opdrachtnemer merkt dat de opdrachtgever nooit aanwezig is
	Opdrachtgever komt niet naar vergaderingen	Opdrachtgever aangeven dat het project gevaar kan lopen en doorgeven aan proces begeleider	P	Afstudeerder	Als de opdrachtgever 1 keer niet aanwezig is op een gesprek
5	Applicatie heeft te veel bugs en kan niet in productie genomen worden				
	Tijdens het realiseren is er teveel functionaliteit geëist door de opdrachtgever	Prioriteiten stellen met de opdrachtgever wat eerst gerealiseerd zal worden	S	Afstudeerder	Wanneer er teveel taken voor een iteratie zijn ingepland
	De opdrachtnemer heeft geen aandacht besteed aan het testen van de software	Er voor zorgen dat er tijdens een iteratie genoeg tijd wordt besteed aan testen	P	Afstudeerder	Bij het einde van elke iteratie kijken of alle nieuwe functionaliteit is getest

	De opdrachtnemer heeft niet genoeg testen uitgevoerd	Tijdens de volgende iteratie testen schrijven voor de afgelopen iteratie	S	Afstudeerder/ Opdrachtgever	Bij een sprintreview kijken of alle tests zijn uitgevoerd en geschreven

P/S = Preventief of Schade beperkend

3 Aanpak

3.1 Ontwikkelmethodiek

Aan het begin van dit project had de opdrachtgever een aantal eisen en wensen ten aanzien van het proces. Deze eisen en wensen waren:

1. Overzicht tijdens het gehele project
2. Snel kunnen ingrijpen als iets mis gaat
3. Goed kunnen omgaan met wijzigende requirements
4. Korte tussentijdse opleveringen
5. Tijdens het project prioriteren van requirements
6. "Requirements as a Scenario"

Aan de hand van deze eisen en wensen was het duidelijk dat er niet met een waterval methodiek gewerkt kon worden. Hiermee is het niet mogelijk om tussentijdse opleveringen te doen en ook is het niet mogelijk om snel in te grijpen. Met een iteratieve ontwikkelmethode kan er wel voldaan worden aan deze eisen en wensen.

De opdrachtgever had verder ook een aantal voorkeuren gegeven:

1. Sprints van twee weken
2. Keuze tussen Scrum en Kanban

Info Support voert opdrachten voornamelijk uit met Scrum. Verder heeft de opdrachtnemer geen ervaring met Kanban en wel met Scrum.

Met Scrum is het gemakkelijk om overzicht te houden van het project, omdat er per sprint een contactmoment is met de opdrachtgever (Sprint review). Hier kan de opdrachtgever gelijk ingrijpen als het nodig is.

Bij het plannen van een sprint kan de opdrachtgever ook gelijk aangeven welke backlogs de hoogste prioriteit hebben. Hier kan hij gelijk requirements wijzigen als dat nodig is.

Als laatste kan de opdrachtgever ook "Requirements as a Scenario" opstellen. Dit worden de user stories.

Scrum past goed bij de eisen en wensen van de opdrachtgever en voldoen ook aan zijn voorkeuren. Daarom is er gekozen om tijdens dit project met Scrum te werken.

3.1.1 Aangepaste scrum

Het is niet mogelijk om Scrum correct uit te voeren in een project waar er alleen één persoon in de ontwikkelteam zit. Scrum is op de volgende wijze aangepast:

Rollen

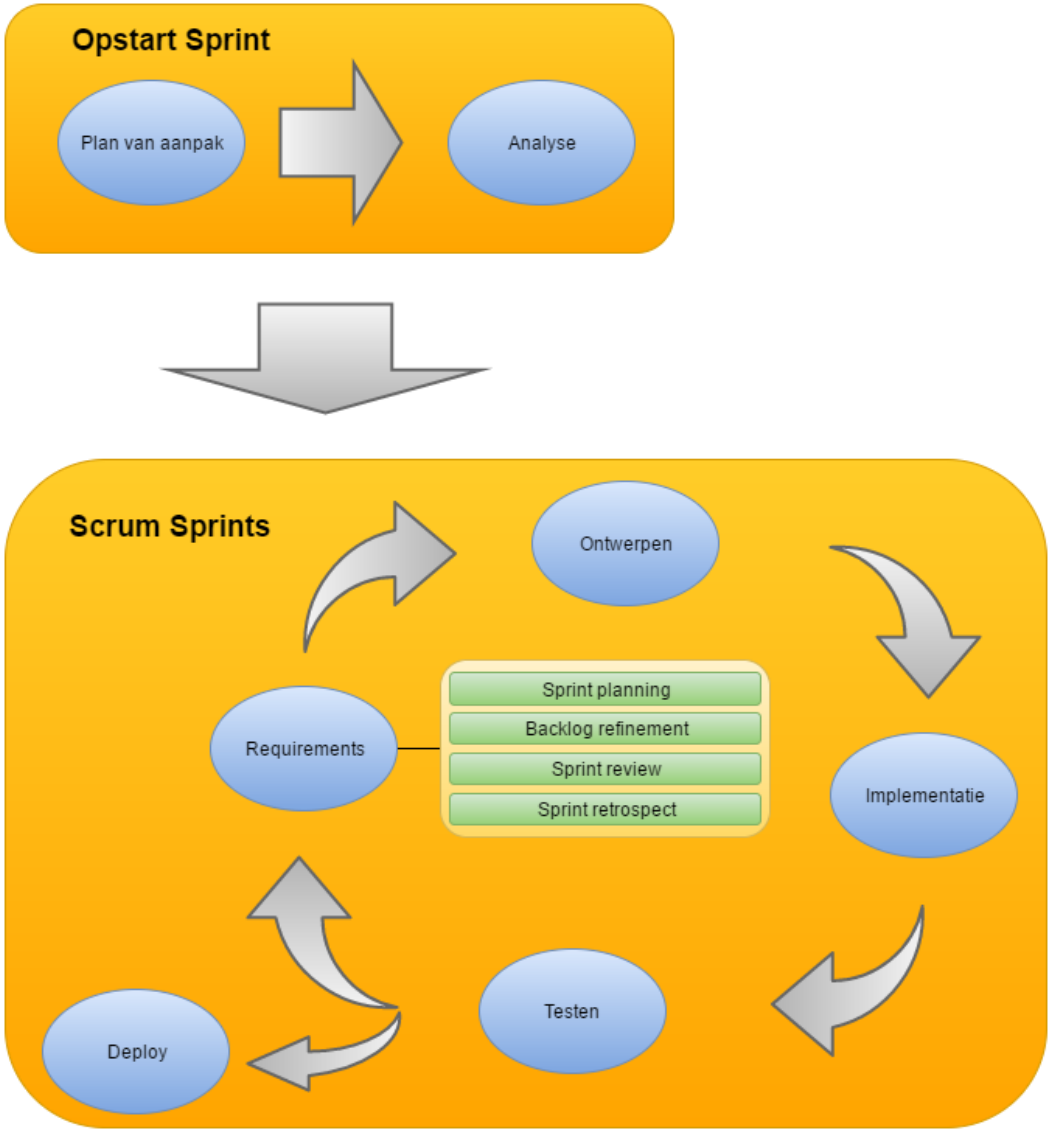
Rol	Wie
Product owner	Paul Borgeld (Opdrachtgever)
Ontwikkelteam	Marc de Meza (Opdrachtnemer)
Scrummaster	Marc de Meza (Opdrachtnemer)

Het scrumteam bestaat alleen uit Paul Borgeld en Marc de Meza

Daily stand ups
Er zal geen daily stand up gehouden worden door de opdrachtnemer. Alleen een daily stand up houden kan ervoor zorgen dat de opdrachtnemer een vertrouwen in de opdracht krijgt terwijl de opdracht achter loopt. Dit gebeurt omdat de opdrachtnemer positief kan denken over de voortgang, maar in werkelijkheid achter kan lopen. De voortgang van de opdracht zal in TFS bijgehouden worden.

3.2 *Fasering*

Aan het begin van het project zal er sprake zijn van een sprint 0, oftewel een "opstart sprint". In deze "opstart sprint" zullen verschillende documenten worden opgeleverd zoals een plan van aanpak en de analyse naar de mogelijkheden om gegevens uit Microsoft Release Management te halen. De rest van de sprints zullen bestaan uit de volgende fasen: Requirements → Ontwerpen → Implementatie → Test & verificatie. In figuur 2 is dit weergegeven.



Figuur 2

3.3 Doelstelling en op te leveren producten

3.3.1 Opstart sprint

Plan van aanpak
In de "opstart sprint" moet een plan van aanpak voor de opdracht gemaakt worden. In dit document komen organisatorische aspecten van het project aan de orde. Aan het einde van deze fase wordt het plan van aanpak opgeleverd.

Analyse
Om de opdracht te kunnen realiseren moet er een analyse uitgevoerd worden. In deze analyse moeten de mogelijkheden om data uit Microsoft Release Management te halen onderzocht worden. Het resultaat kan vervolgens gebruikt worden bij het realiseren van het dashboard.

Met dit document kan de opdrachtgever zien welke mogelijkheden er bestaan voor het ophalen van gegevens uit Microsoft Release Management. De motivatie van de gekozen mogelijkheid zal ook beargumenteerd zijn, zodat de opdrachtgever terug kan zien waarom het gekozen is.

3.3.2 Scrumfase

Requirements
Tijdens de requirements fase worden nieuwe requirements opgehaald voor het project. Na de requirements fase zal het functioneel ontwerp worden opgeleverd met daarin de "Requirements as a Scenario".

Ontwerpen
De nieuwe functionaliteiten voor de huidige sprint worden ontworpen. Dit wordt gedaan met UML. Na de ontwerp fase worden de beslissingen van het project opgeleverd in een technisch ontwerp.

Implementatie
De nieuwe functionaliteiten worden gerealiseerd door het "ontwikkelteam".

Test & verificatie
Tijdens het realiseren van de nieuwe functionaliteiten zal er gewerkt worden met Test Driven Development. Hiermee zullen de unit tests gemaakt worden.

Voor de acceptatie tests wordt SpecFlow gebruikt. Met SpecFlow worden de requirements overgezet naar Specifications by Example. Met deze specifications worden er automatisch tests gegenereerd door SpecFlow.

Alle tests worden uitgevoerd om te kijken of de geïmplementeerde functionaliteit is gerealiseerd volgens de requirements.

Deployment
Nadat de nieuwe functionaliteit is gerealiseerd en getest zal er een kleine demonstratie zijn tijdens de sprint review. Dit zal plaats vinden op de omgeving van de opdrachtnemer zelf.

Naarmate het project vordert zal de opdrachtgever mogelijk de applicatie in gebruik willen nemen. In dat geval zal de deployment plaats vinden op de productie omgeving.

3.3.3 Overige fasen

Er zijn nog een aantal taken die tijdens de sprint ook moeten gebeuren. Deze taken zijn onder andere het beoordelen van de sprint en het plannen ervan.

Back log refinement
Back log items die niet specifiek zijn worden samen met de product owner specifiek gemaakt. Dit vindt plaats aan het begin van een sprint.

Sprint planning
Tijdens de sprint planning wordt samen met de product owner besproken welke backlogs gerealiseerd gaan worden in de sprint. De product owner geeft ook gelijk de prioriteit weer voor de gekozen backlogs.

Sprint review
Het product dat in de sprint is afgerond wordt gepresenteerd aan de product owner. Tijdens de sprint review kan de opdrachtgever feedback geven op het product. Dit gebeurt aan het einde van een sprint.

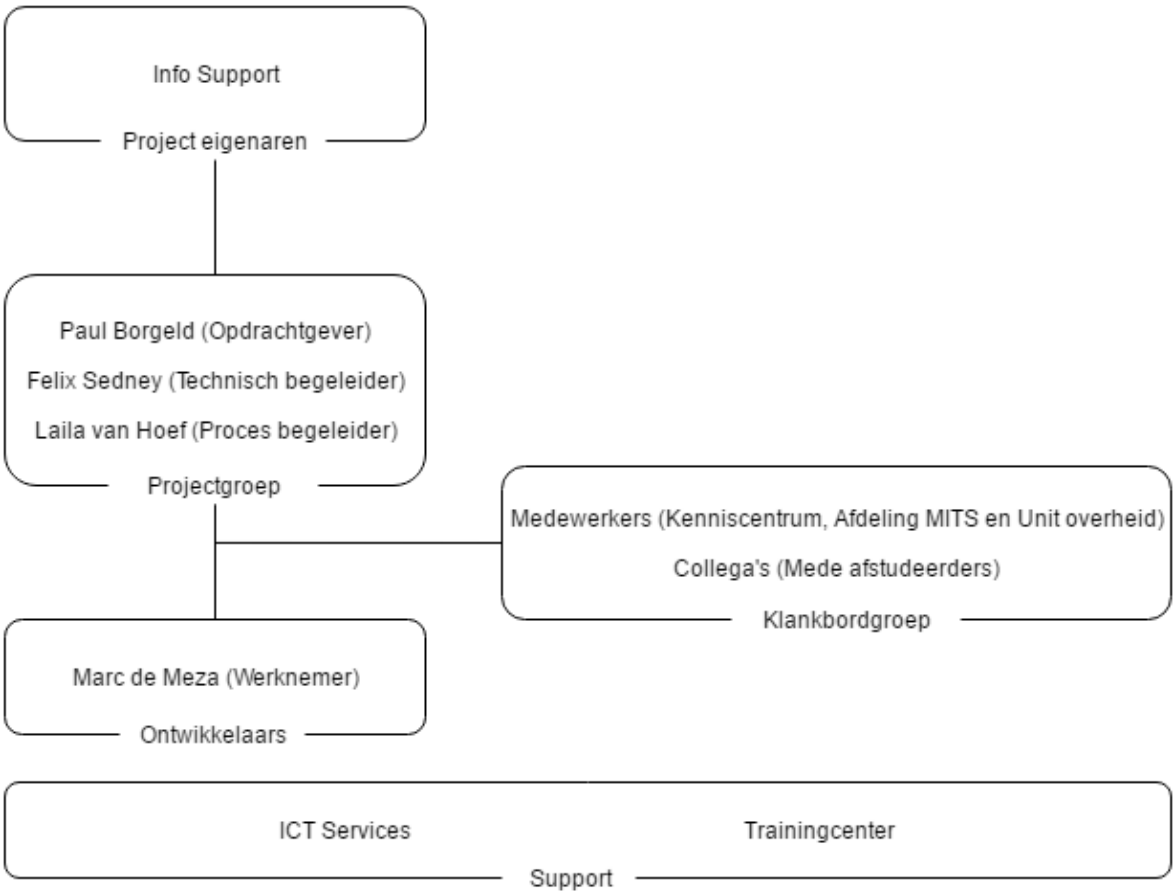
Sprint retrospect
Tijdens de sprint retrospect kan de product owner feedback geven over de sprint. Hier kan besproken worden wat goed of fout is gegaan. De retrospect wordt uitgevoerd aan het einde van een sprint.

4 Beheersaspecten

4.1 Organisatie

4.1.1 Normstelling

Bij dit project zijn er een aantal partijen betrokken. Deze partijen hebben elk een andere verantwoordelijkheid en belangen in het project. In figuur 3 is er een overzicht van alle rollen in het project en ook is er een beschrijving over hun bevoegdheden en verantwoordelijkheden.



Figuur 3

Project eigenaar
De project eigenaar was in dit geval Info Support. Info Support kon op elk moment het project stop zetten.

Projectgroep
De projectgroep bestaat uit alle begeleiders van de opdrachtnemer. Deze begeleiders kunnen benaderd worden door de opdrachtnemer voor eventuele vragen over het proces of technische vragen.

De opdrachtgever bepaalt welke richting het project op gaat. Hij neemt beslissingen ten aanzien van de prioriteiten en bepaalt wat er gerealiseerd gaat worden.

Klankbordgroep
De klankbordgroep heeft geen directe verantwoordelijkheden of bevoegdheden binnen het project. Deze groep ondersteunt de opdrachtnemer waar nodig of helpt de opdrachtnemer als er vraagstukken besproken moeten worden.

Ontwikkelaars
De opdrachtnemer is de enige ontwikkelaar in dit project. Hij is verantwoordelijk voor het ontwerpen, bouwen en testen van de applicatie. Verder moet de opdrachtnemer ook de voortgang communiceren naar de stuurgroep (De Haagse Hogeschool) en de projectgroep.

Support
ICT services en het kenniscentrum faciliteren en leiden de opdrachtnemer op indien nodig.

4.1.2 Voortgangscontrole

De voortgang van het project zal door middel van gesprekken met de projectgroep gebeuren. Tweewekelijks zal er een gesprek zijn met de procesbegeleider om de voortgang van het gehele project in kaart te brengen. Met de opdrachtgever zal er minimaal één keer per twee weken een gesprek zijn over de voortgang. Dit wordt gedaan in een sprint review. Met de technische begeleider zal er aan het begin veel contact zijn, maar tegen het einde van het project alleen wanneer nodig.

Voor het communiceren naar de stuurgroep zal de opdrachtnemer zelf contact opnemen met De Haagse Hogeschool. De opdrachtnemer zal bij 25%, 45%, 60% en drie lesweken voor het inleveren van het afstudeerverslag contact opnemen.

4.2 Informatie

4.2.1 Normstelling

Een aantal van de documenten die opgeleverd worden zijn groeidocumenten. Dit betekent dat ze in het begin nog niet veel informatie bevatten, maar na elke sprint aangevuld worden. De groeidocumenten zijn:

1. Product backlogs
2. Requirements analyse
3. Functioneel ontwerp
4. Technisch ontwerp
5. Test documenten

Document	Voor wie bestemd	Goedkeuring	In CC
Plan van aanpak	Opdrachtgever/ Technische begeleider	Technische begeleider	Procesbegeleider
Product backlogs	Opdrachtgever	Opdrachtgever	Technische begeleider
Requirements analyse	Opdrachtgever/ Technische begeleider	Technische begeleider	-
Analyse mogelijkheden Microsoft Release Management	Opdrachtgever	Opdrachtgever	Technische begeleider/ Procesbegeleider
Functioneel ontwerp	Opdrachtgever	Opdrachtgever	Technische begeleider/ Procesbegeleider
Technisch ontwerp	Technische begeleider	Technische begeleider	Opdrachtgever/ Procesbegeleider
Test document	Technische begeleider	Technische begeleider	Opdrachtgever
Voortgangsdocument	Proces begeleider	Proces begeleider	-

4.2.2 Voortgangscontrole

Alle documenten beginnen in een concept fase. Bij elk document vindt er een goedkeuring plaats. Alleen de persoon genoemd bij goedkeuring kan bepalen of het document afgerond is of niet. Bij het afronden van een document wordt de status van het document definitief. Alle datums van een status wijziging worden in het document bijgehouden.

Elk document heeft een versie nummer. Zodra een document gewijzigd is zal de versie aangepast worden.

4.3 Tijd

4.3.1 Normstelling

Om een overzicht te krijgen wanneer bepaalde taken worden uitgevoerd is er een globale planning gemaakt:

Mijlpaal	Begin	Eind	Duur
Plan van aanpak	01-02-2016	12-02-2016	2 Weken
Analyse mogelijkheden Microsoft Release Management	15-02-2016	09-03-2016	$3^{1/2}$ Weken
Sprint review/Sprint retrospect/Sprint planning	11-03-2016	11-03-2016	1 dag
Sprint 1	14-03-2016	24-03-2016	2 Weken
Sprint review/Sprint retrospect/Sprint planning	25-03-2016	25-03-2016	1 dag
Sprint 2	28-03-2016	07-04-2016	2 Weken
Sprint review/Sprint retrospect/Sprint planning	08-04-2016	08-04-2016	1 dag
Sprint 3	11-04-2016	21-04-2016	2 Weken
Sprint review/Sprint retrospect/Sprint planning	22-04-2016	22-04-2016	1 dag
Sprint 4	25-04-2016	05-05-2016	2 Weken
Sprint review/Sprint retrospect/Sprint planning	06-05-2016	06-05-2016	1 dag
Sprint 5	09-05-2016	19-05-2016	2 Weken
Sprint review/Sprint retrospect	20-05-2016	20-05-2016	1 dag
Afstudeerverslag	01-02-2016	03-06-2016	18 Weken

Elke sprint zijn er een aantal taken die opnieuw uitgevoerd worden. In de volgende lijst zijn de taken opgenomen die gedurende de sprint worden uitgevoerd. De begin datum van deze taken is 14-03-2016 en de einddatum is 19-05-2016. De totale duur is 10 weken.

- Requirements analyseren
- Functioneel ontwerp
- Technisch ontwerp
- Applicatie realiseren
- Tests documenteren
- Tests automatiseren

4.3.2 Voortgangscontrole

De voortgangscontrole zal worden gedaan door de procesbegeleider. Tweewekelijks zal een sprint review worden gehouden met de begeleider.

4.4 Geld

4.4.1 Normstelling

Tijdens het project zijn er geen bijzondere kosten die gemaakt moeten worden. Indien dit wel zo is zal dit worden toegevoegd. Indien dit wel het geval is zal er contact opgenomen worden met de opdrachtgever.

4.4.2 Voortgangscontrole

Niet van toepassing.

4.5 Middelen

4.5.1 Normstelling

Een deel van de middelen die hier zijn opgenoemd kunnen teruggevonden worden in hoofdstuk 1.8 Randvoorwaarden.

Middel	Vanaf	Frequentie	Contact persoon
Werkplek met minimaal een pc	Start project	Continu	Systeembeheer
Visual studio en SQL Server	Start eerste sprint	Continu	Systeembeheer
TFS/GIT	Start eerste sprint	Continu	Opdrachtgever
Omgeving met Microsoft Release Management en reële testdata	Start eerste sprint	Continu	Opdrachtgever
Opdrachtgever	Start project	Minimaal drie keer per sprint	-
Technisch begeleider	Start project	Minimaal twee keer per sprint	-
Trainingen kenniscenter	Indien nodig	-	Technisch begeleider

4.5.2 Voortgangscontrole

Indien een middel nodig is wordt de desbetreffende contactpersoon benaderd door de opdrachtnemer. Verzoeken voor middelen worden alleen gedaan als het nodig is om het project succesvol af te ronden.

4.6 Kwaliteit

4.6.1 Normstelling

Productkwaliteit

Om ervoor te zorgen dat het product voldoet aan de vooraf gestelde eisen worden de volgende maatregelen getroffen:

1. De requirements worden samen met de opdrachtgever concreet gemaakt
2. Opdrachtgever ontvangt functioneel ontwerp
3. Technisch begeleider beoordeelt of de ontwerpen goed zijn
4. Bij tegenstrijdige requirements moet de opdrachtgever bepalen welke requirement wel en niet gerealiseerd wordt

5. Als een requirement niet gerealiseerd kan worden, zullen de opdrachtgever en de opdrachtnemer samen beslissen of de requirements niet op een andere manier gerealiseerd kan worden
6. De applicatie gaat getest worden volgens requirements
7. Tijdens sprint review kan de opdrachtgever aangeven of hij tevreden is
8. Op elk moment kan de opdrachtgever ingrijpen, omdat er met scrum gewerkt wordt

Proceskwaliteit
Om ervoor te zorgen dat het proces goed verloopt moeten de volgende punten uitgevoerd worden:

1. Voortgangsgesprekken met procesbegeleider
2. Documenten opsturen naar de desbetreffende partijen zoals vermeldt in hoofdstuk 4.2.1
3. Voortgangsgesprekken met De Haagse Hogeschool waarbij proces en documenten worden besproken.
4. Sprint retrospect met de opdrachtgever

Gebruikerskwaliteit
Om ervoor te zorgen dat de gebruikers tevreden zijn met het uiteindelijke product wordt het volgende gedaan:

1. Sprint reviews met de opdrachtgever
2. Eventueel demonstraties aan de medewerkers wanneer een shippable product is geleverd

4.6.2 Voortgangscontrole

Voor het controleren van de voortgang van de kwaliteit wordt er per onderdeel gecontroleerd.

Onderdeel	Moment	Door wie
Product	Bij het opleveren van technische documenten en tijdens sprint reviews	Technisch begeleider en opdrachtgever
Proces	Tijdens voortgangsgesprekken, bezoek dagen van De Haagse Hogeschool en tijdens het opsturen van technische documenten	Technische begeleider, opdrachtgever en De Haagse Hogeschool
Gebruiker	Tijdens sprint review op het einde van een sprint	Opdrachtgever

4.7 Overlegvormen

Tijdens het project zijn er een aantal overlegmomenten noodzakelijk. In de volgende tabel wordt per overleg de deelnemers en frequentie aangegeven.

Overlegvorm	Deelnemers	Frequentie	Bevoegdheden
Voortgangsgesprek	Proces begeleider en Opdrachtnemer	1 keer per twee weken	-
Contactmomenten De Haagse Hogeschool	Technische begeleider	1 keer	Stopzetten afstuderen
Sprint review	Opdrachtnemer, Opdrachtgever, Technisch begeleider	1 keer per sprint	Product beoordelen en eventueel aanpassingen voor volgende sprint bedenken
Sprint retrospect	Opdrachtnemer, Opdrachtgever, Technisch begeleider	1 keer per sprint	Sprint verbeteren
Sprint planning	Opdrachtnemer, Opdrachtgever, Technisch begeleider	1 keer per sprint	Bepalen komende werkzaamheden
Gesprekken met Business unit manager	Business unit Manager	1 keer per 4 – 6 weken	-

De sprint review, sprint retrospect en sprint planning zullen plaats vinden op dezelfde dag. Het overlegmoment zal beginnen met een sprint review om de opdrachtgever te wijzen wat gerealiseerd is. Vervolgens zal de sprint retrospect plaats vinden waar de opdrachtgever en opdrachtnemer de mogelijkheid krijgen om te bespreken hoe de sprint is verlopen. Op het einde van het overlegmoment gaan de opdrachtgever en de opdrachtnemer de volgende sprint in plannen.

Voordat de sprint van start gaan worden de sprint review overlegmomenten gebruik als voortgangsgesprekken. Tijdens deze voortgangsgesprekken worden documenten zoals het plan van aanpak en de analyse besproken.

4.8 Communicatie

Tijdens het project zullen er diverse documenten uitgewisseld worden. In het volgende RACI-model is een duidelijk overzicht welke documenten gecommuniceerd worden naar de deelnemers.

Fase/proces	Ondersteunen	Uitvoeren	Beslissen (goedkeuren)	Gebruiken (informeren)
Plan van aanpak	Technisch begeleider	Weknemer	Technisch begeleider	Opdrachtgever, De Haagse Hogeschool, Technische begeleider en Proces begeleider
Analyse mogelijkheden Microsoft Release Management	Technisch begeleider	Opdrachtnemer	Technisch begeleider	Opdrachtgever, De Haagse Hogeschool, Proces begeleider

Functioneel ontwerp	Technisch begeleider	Opdrachtnemer	Technisch begeleider	Opdrachtgever, De Haagse Hogeschool, Proces begeleider
Technisch ontwerp	Technisch begeleider	Opdrachtnemer	Technisch begeleider	Opdrachtgever, De Haagse Hogeschool, Proces begeleider
Test resultaten	Technisch begeleider	Opdrachtnemer	Technisch begeleider	Opdrachtgever, De Haagse Hogeschool
Afstudeerverslag	Eerste Examinator	Opdrachtnemer	Tweede Examinator	Technisch begeleider, Eerste Examinator, Tweede Examinator

5 Oplevering

Plan van aanpak
Het plan van aanpak zal alle management aspecten van het project bevatten. In dit document kan teruggekeken worden hoe het project is opgezet en wie daar allemaal bij betrokken is.

Analyse mogelijkheden Microsoft Release Management
In de analyse zal duidelijk worden weergegeven wat de mogelijkheden zijn voor het ophalen van data uit Microsoft Release Management. In het document zal er ook een conclusie komen over welke methoden gebruikt gaat worden voor het ophalen van informatie.

Functioneel ontwerp
In het functioneel ontwerp worden de functionaliteiten van de applicatie weergegeven. Dit document bestaat uit diverse UML-diagrammen. Hierin zullen ook de requirements van het systeem in komen te staan

Technisch ontwerp
In het technisch ontwerp zal de implementatie van de functionaliteiten duidelijk worden weergegeven met UML-diagrammen.

Geteste applicatie
De volledige applicatie met bijbehorende geautomatiseerde tests.

Test documentatie
In de test documentatie zal vermeld staan waarom er voor een bepaalde test is gekozen en hoe deze is opgezet. Verder zullen de resultaten en bevindingen van de tests ook zichtbaar zijn.

Afstudeerverslag
Het afstudeerverslag zal het gehele proces van het afstuderen bevatten. Hierop zal de opdrachtnemer ook op beoordeeld worden

Bijlage II

Analyse mogelijkheden ophalen gegevens uit Microsoft Release Management

Analyserapport

Mogelijkheden verkrijgen gegevens uit Microsoft Release Management

Marc de Meza

Historie

Versie	Status	Datum	Auteur	Verandering
1.0	Concept	15-02-2016	Marc de Meza	Creatie
1.1	Concept	19-02-2016	Marc de Meza	Wijzigen van onderzoek naar analyse
1.2	Concept	24-02-2016	Marc de Meza	Feedback Felix Sedney 23-02-2016
1.3	Concept	25-02-2016	Marc de Meza	Uitvoeren analyse
1.4	Concept	02-03-2016	Marc de Meza	Feedback Felix Sedney 02-03-2016
1.5	Concept	03-03-2016	Marc de Meza	Toevoegen informatie
1.6	Concept	22-03-2016	Marc de Meza	Feedback Felix Sedney 22-03-2016
1.7	Definitief	26-03-2016	Marc de Meza	-

Distributie

Versie	Status	Datum	Aan
1.0	Concept	18-02-2016	Felix Sedney, Paul Borgeld en Laila van de Hoef
1.3	Concept	26-02-2016	Felix Sedney, Paul Borgeld en Laila van de Hoef
1.5	Concept	09-03-2016	Felix Sedney, Paul Borgeld en Laila van de Hoef
1.6	Concept	24-03-2016	Felix Sedney, Paul Borgeld en Laila van de Hoef

Referenties

Code	Bron
RF01	https://msdn.microsoft.com/library/dn217874%28v%3Dvs.120%29.aspx
RF02	https://carriere.infosupport.com/afstuderen-net-visual-deployment-pipeline
RF03	https://msdn.microsoft.com/Library/vs/alm/Release/overview
RF04	https://msdn.microsoft.com/en-us/library/vs/alm/release/overview-rm2015
RF05	https://knownow.infosupport.com/knowledge/uitlezen-gebruikers-groepen-en-rechten-van-release-management-2015-met-powershell
RF06	https://knownow.infosupport.com/knowledge/microsoft-release-management-dashboard
RF07	Principes van databases, Guy de Tré
RF08	https://www.visualstudio.com/en-us/news/release-archive-vso.aspx

Managementsamenvatting

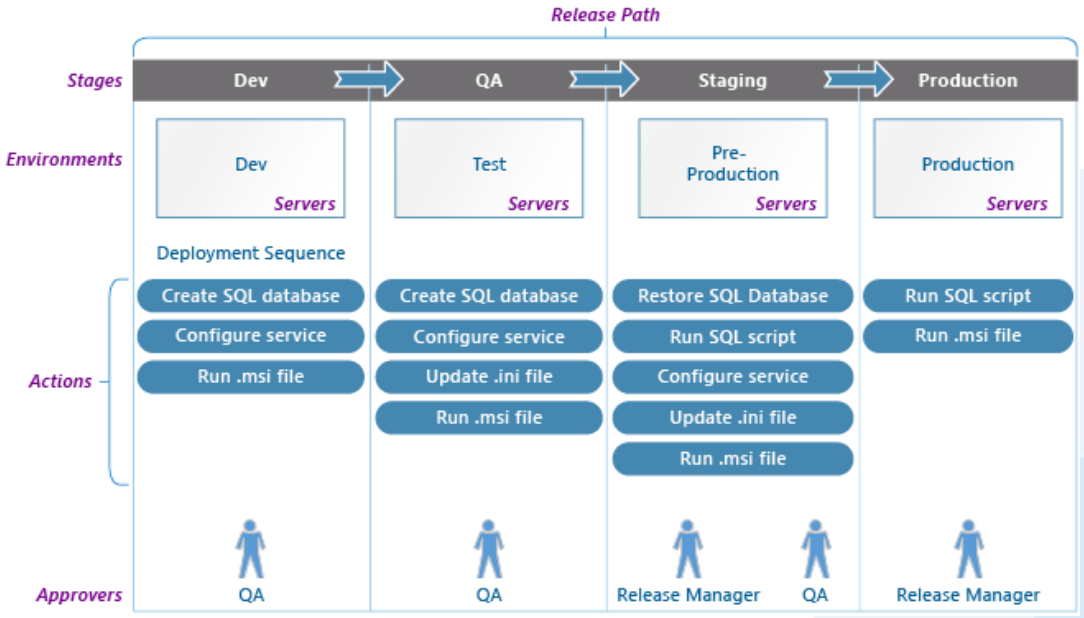
In deze analyse wordt er gekeken naar de mogelijkheden voor het verkrijgen van gegevens uit Microsoft Release Management (MRM) 2015 update 1 en update 2.

Er worden twee mogelijkheden vergeleken voor het ophalen van data. De API van MRM 2015 en het uitlezen van de database. Voor beide manieren worden de voor- en nadelen naast elkaar gelegd.

Inhoud

1. Inleiding

Sinds de komst van MRM 2013 is het mogelijk om een deploymentstraat te definiëren. Met een deploymentstraat kan men vanaf ontwikkelen tot en met productie aangeven hoe een applicatie automatisch geïnstalleerd moet worden. In figuur 1 is een deploymentstraat weergegeven. Tijdens deze installaties kan een gebruiker aangeven of er bepaalde handelingen zijn die uitgevoerd moeten worden. Deze handelingen kunnen bestaan uit bijvoorbeeld het wijzigen van een database of het configureren van een webserver.



Figuur 1. Automate deployment with Release Management RF01

Verder is het in MRM ook mogelijk om iemand in het systeem bepaalde verantwoordelijkheden te geven. Een van deze verantwoordelijkheden is het kunnen goedkeuren van een installatie op een bepaalde omgeving. Nadat er een goedkeuring is gegeven mag de applicatie verder geïnstalleerd worden op de volgende omgeving[RF01]. Nadat er een installatie heeft plaats gevonden wordt er informatie over deze installatie opgeslagen. Hierbij kan men denken aan versienummers, genomen stappen tijdens installatie en fouten tijdens installatie.

Probleemstelling
Momenteel biedt MRM 2013 geen duidelijke overzichten van de omgevingen en de software die daarop is geïnstalleerd. Het is niet mogelijk om in één oogopslag te zien hoe de installaties op die omgevingen zijn gelopen. In MRM 2015 update 1 en update 2 heeft Microsoft nieuwe overzichten gemaakt in de applicatie, maar het is nog niet duidelijk welke overzichten dan beschikbaar zullen zijn.

Om zelf een uitbreiding aan overzichten te kunnen realiseren zijn er een aantal gegevens nodig uit MRM 2015. Deze gegevens kunnen gebruikt worden om zelf een dashboard te maken waarin alle overzichten te vinden zijn. Momenteel is er kennis voor het verkrijgen van gegevens uit MRM 2013 en een deel van de gegevens uit MRM 2015 update 1. Het is nog niet duidelijk hoe alle benodigde gegevens uit MRM 2015 update 1 en update 2 te verkrijgen zijn.

Doelstelling
De doelstelling van deze analyse is het in kaart brengen van de mogelijkheden voor het verkrijgen van de gegevens uit MRM 2015 update 1 en update 2. Verder zal er uitgezocht worden welke gegevens beschikbaar zijn. Als laatste

wordt er ook gekeken van welke gegevens MRM 2015 al overzichten van heeft en of deze extern gebruikt kunnen worden.

2. Achtergrond

Om deze analyse te begrijpen moeten de volgende punten verhelderd worden.

2.1 Gegevens van Microsoft Release Management

Om een dashboard van een deploymentstraat te kunnen realiseren is het belangrijk dat alle benodigde gegevens te verkrijgen zijn. Als er een verbinding met MRM 2015 is gemaakt moet deze minimaal de volgende informatie aanbieden:

1. Omgevingen
2. Applicaties op deze omgevingen
3. Versienummers

2.2 Versie Microsoft Release Management

Momenteel wordt er gebruik gemaakt van MRM 2013, maar binnenkort wordt dit geüpdatet naar MRM 2015 update 1. Hierdoor is het niet nodig informatie uit MRM 2013 uit te lezen. Een groot verschil tussen deze twee versies is dat MRM 2015 een Application Programming Interface (API) gebruikt om te communiceren tussen server en client[RF05]. Het zal mogelijk een optie zijn om gegevens voor de dashboard uit de API te lezen.

MRM 2015 update 1 maakt gebruik van een client en een server. Tijdens het installeren van MRM 2015 update 1 moet de gebruiker eerst een server installeren. Bij het installeren van deze server kan een gebruiker verschillende instellingen meegeven zoals database locatie. Nadat de installatie heeft plaatsgevonden kan een gebruiker verbinding maken naar de server met een client.

In Team Foundation Server (TFS) 2015 update 2 heeft Microsoft besloten om de functionaliteiten van MRM te integreren in TFS[RF04]. Met deze verandering heeft Microsoft de client uit MRM verwijderd. Dit kan invloed hebben op de toekomst van de applicatie, omdat de API die momenteel wordt aangeboden voor de client mogelijk niet meer ondersteund zal worden.

2.3 Huidige kennis

Info Support maakt gebruik van twee wijzen voor het ophalen van gegevens uit MRM. Een daarvan is gerealiseerd in de 2013 versie door de database direct uit te lezen[RF05]. Info Support gebruikt deze wijze voor het genereren van kleine overzichten van het informatie in MRM. De andere wijze is gerealiseerd door gebruik te maken van de API die wordt aangeboden door MRM 2015 update 1[RF06]. Deze twee wijzen worden gebruikt tijdens het analyseren van de mogelijkheden.

3. Analyse vraagstukken

In dit hoofdstuk wordt er gekeken welke vraagstukken geanalyseerd gaan worden. Verder zal er kort uitgelegd worden waarom juist deze vraagstukken geanalyseerd zullen worden.

3.1 Api mogelijkheden

Zoals eerder vermeldt maakt MRM 2015 update 1 gebruik van een client en een server. Verder is er ook aangegeven dat deze twee door middel van een API communiceren. In dit hoofdstuk worden de mogelijkheden voor het gebruiken van deze API om gegevens op te halen uit MRM 2015 update 1 geanalyseerd.

Om ervoor te zorgen dat in MRM 2015 update 2 de gegevens nog te benaderen zijn wordt er ook geanalyseerd of de API nog zal werken in deze versie.

3.2 Database mogelijkheden

Voor het ophalen van de gegevens wordt er ook gekeken naar het direct uitlezen van de database van MRM 2015 update 1. Al de informatie die nodig is, is in de database te vinden. Het direct uitlezen van de database brengt problemen met zich mee. De mogelijkheden van het ophalen en de problemen die erbij komen worden in dit hoofdstuk geanalyseerd. Verder wordt er ook gekeken of het uitlezen van de database nog mogelijk zal zijn in MRM 2015 update 2.

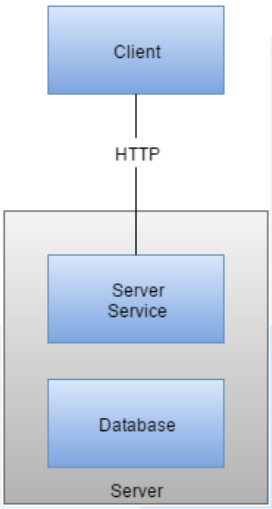
4. Application programmable interface

In dit hoofdstuk wordt er gekeken naar de bestaande mogelijkheden die MRM 2015 update 1 aanbiedt voor het verkrijgen van gegevens. Vervolgens wordt de verbinding voor het ophalen geanalyseerd. Hieruit zal duidelijk worden hoe de gegevens uit MRM gehaald kunnen worden. Verder wordt er ook gekeken of er in de toekomst wijzigingen zullen plaats vinden in het gebruik van deze mogelijkheden. Als laatste worden de voor- en nadelen beschreven.

4.1 Inleiding

MRM 2015 update 1 maakt gebruik van twee componenten die met elkaar communiceren, namelijk de client en de server. De server biedt een API via HTTP aan waarmee de client kan communiceren. Met deze API is de MRM database volledig afgeschermd van de client. Dit betekent dat de client nooit wijzigingen kan doorvoeren zonder dat MRM weet wat er gewijzigd wordt.

De API die wordt aangeboden door de server is momenteel niet gedocumenteerd. Dit brengt als probleem mee dat niet duidelijk is welke gegevens wel of niet uitgelezen kunnen worden.

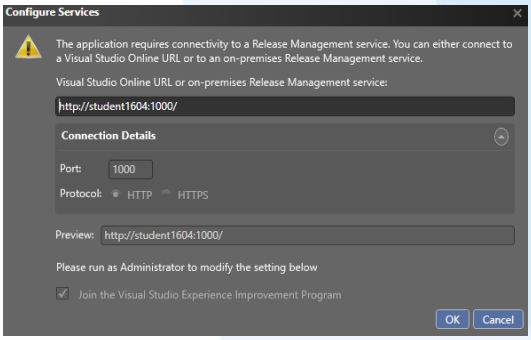


Figuur 2

4.2 Analyse verbinding

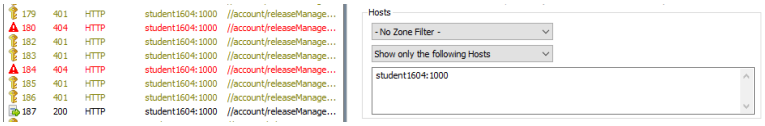
Voor het analyseren van de verbinding is er gebruik gemaakt van de tool Fiddler om de HTTP requests uit te lezen. Met Fiddler is het mogelijk om een HTTP request te onderscheppen en te analyseren. In deze analyse is het mogelijk te zien welke gegevens worden verstuurd en welke gegevens worden verkregen van de API.

Fiddler leest zonder standaard instellingen aan te passen geen verkeer uit van localhost. Daarom is de url waar MRM 2015 update 1 server op staat gewijzig van localhost naar de werkelijke url. Hierdoor was het gemakkelijker om de requests uit te lezen.



Figuur 3

Aangezien de client van MRM 2015 update 1 gebruik maakt van de API was het mogelijk om al het verkeer te zien tijdens het navigeren van de applicatie. Bij het inladen van nieuwe overzichten of detailschermen kon precies uitgelezen worden wat verstuurd moest worden en hoe het ontvangen werd door de client.



Figuur 4

4.3 Analyse Application Programming Interface

Bij het succesvol versturen van een request geeft de API een respons terug met de gegevens in XML formaat. Als een request niet succesvol is geeft de API duidelijke foutmeldingen, zoals een 401 wanneer de gebruiker niet geauthentiseerd is.

Om te weten te komen welke gegevens uit de API gehaald kunnen worden is er met Fiddler een opname gemaakt. Tijdens de opname werd er naar alle overzichten in MRM client genavigeerd. Voor alle overzichten die in de client te zien waren moest er een request gemaakt worden naar de server via de api. Nadat er naar alle overzichten genavigeerd was, was er een lijst van requests beschikbaar via door Fiddler. Deze requests konden geanalyseerd worden.

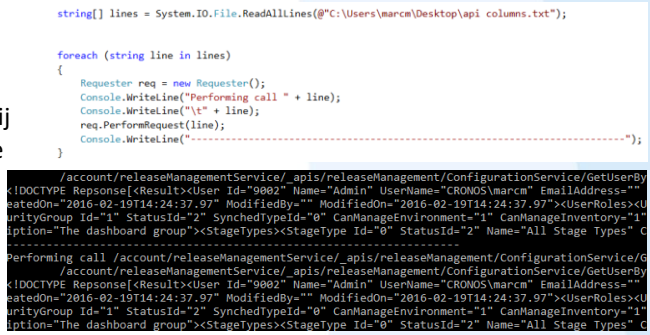
4.3.1 Authenticatie

Om informatie uit de API te krijgen is het noodzakelijk om geauthentiseerd te zijn. MRM 2015 update 1 maakt gebruik van Intergrated Windows Authentication. Dit betekent dat bij het opsturen van de request je identiteit ook gestuurd moet worden. Als dit niet het geval is, is het niet mogelijk om de API aan te spreken.

4.3.2 Opvragen gegevens proof of concept

Om zeker te zijn dat de gegevens ook daadwerkelijk te benaderen waren met de requests is er een ‘proof of concept’ (POC) gemaakt gemaakt. In deze POC zijn alle requests die door fiddler verzameld waren uitgevoerd. Bij het uitvoeren was er ook gekeken of het mogelijk was de gegevens uit te lezen.

Zoals te zien in afbeelding 5 is het gelukt om gegevens uit te lezen met behulp van de api.



Figuur 5

4.4 Toekomst

Met de komst van MRM 2015 update 2 zal MRM geïntegreerd worden in TFS. Dit betekent dat de MRM client niet meer beschikbaar zal zijn. Voor het testen of de huidige API van MRM 2015 update 1 nog werkte in de nieuwe versie is TFS 2015 update 2 geïnstalleerd. Vervolgens is zijn dezelfde API requests uitgevoerd op de aangepaste url. Met deze opzet was het niet mogelijk om de API aan te spreken. Sinds de API van MRM 2015 update 1 geen documentatie had, is er ook geen officieel bewijs van Microsoft dat deze is verwijderd uit MRM 2015 update 2.

Microsoft heeft aangegeven dat in MRM 2015 update 3 pas officieel tijd zal worden besteed aan het openbaar maken van een API voor MRM[RF08]. In MRM 2015 update 3 zal het dus nog steeds mogelijk te zijn om verbinding te maken met de API en gegevens op te halen. Welke gegevens de API vrijgeeft en of het gaat lijken op de huidige is nog niet duidelijk. Verder zal er in MRM 2015 update 3 ook Service Hooks worden vrijgegeven. Dit betekent dat een applicatie gewaarschuwd kan worden wanneer een wijziging plaats vindt in MRM.

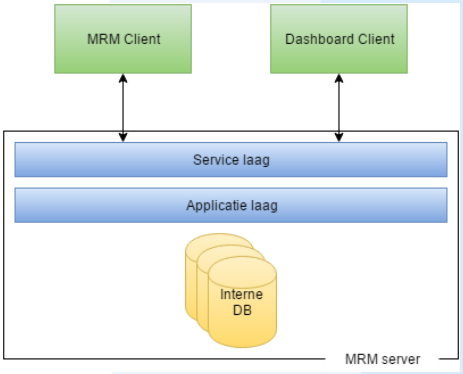
4.5 Voor- en nadadelen van het gebruiken van de api

Als er met behulp van de API een koppeling wordt gemaakt met MRM 2015 update 1 zijn er een aantal voor- en nadelen. Deze voor- en nadelen moeten overwogen worden voordat er een beslissing gemaakt kan worden.

4.5.1 Voordelen gebruik api

Abstractie van de database
Bij het gebruiken van de API is er geen kennis nodig over de structuur van de database. Het is niet nodig om te kijken hoe en waar MRM de data opslaat. Met deze abstractie laag boven de database is het mogelijk dat de interne opslag van MRM kan wijzigen zonder dat er wat verandert aan alle applicaties die verbonden zijn.

MRM kan helemaal bepalen welke gegevens worden aangeboden en hoe deze gegevens worden aangeboden.



Figuur 6

Zoals in figuur 6 te zien is zal het dashboard zich aansluiten op een bestaande API. De MRM client maakt al gebruik van deze API om precies dezelfde informatie op te halen. Door ook gebruik te maken van de API zorgen wij ervoor dat alle communicatie alleen op één manier beschikbaar is

Voorbereid op update 3
In MRM 2015 update 3 is aangegeven dat de API en webhooks beschikbaar en volledig gedocumenteerd zullen zijn[RF08]. Als er nu gebruik gemaakt wordt van de API in update 1 betekent het de bestaande data laag met kleine aanpassingen kan worden overgezet naar de nieuwe versie.

Bij het realiseren van een data laag gaan wij ervan uit dat de bron waar het informatie vandaan komt niet uitmaakt. Als wij bij de update van MRM 2015 update 3 een nieuwe api krijgen moet de data laag aangepast kunnen worden zonder dat de interne werking van het dashboard aangepast moet worden.

Veiligheid
De laatste voordeel bij het gebruiken van de API is dat er geen gevoelige informatie kan uitlekken. MRM bepaalt of informatie verzonden mag worden vanuit de API. Als MRM bepaalt dat informatie gevoelig is, wordt dit informatie niet over de api verstuurd.

4.5.2 Nadelen

Volgende update
In update 2 van MRM 2015 is het, zoals beschreven in hoofdstuk 4.4, niet mogelijk om de API zoals in update 1 te benaderen. Applicaties die zich hebben aangesloten bij de service laag van de server zullen niet meer te gebruiken zijn. Voor deze update zal er een alternatief bedacht moeten worden totdat de update 3 uit is.

Geen documentatie
Het laatste nadeel dat veel invloed kan hebben is dat de API niet gedocumenteerd is. Tijdens het ontwikkelen van een koppeling met de service laag zal het niet mogelijk zijn om documentatie te raadplegen. De verbinding tussen de client en server zal afgeluisterd moeten worden en deze gegevens moeten onderzocht worden om te kijken of ze gebruikt kunnen worden.

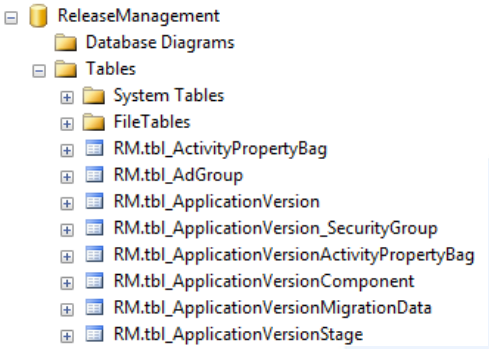
5. Database

In dit hoofdstuk wordt er gekeken naar de mogelijkheden voor het verkrijgen van gegevens uit de database van MRM 2015 update en update 2. Verder worden de voor- en nadelen besproken.

5.1 Inleiding

Tijdens het installeren van MRM 2015 update 1 server wordt er gevraagd om een SQL database op te geven. Nadat de installatie is afgerond is het mogelijk om te navigeren naar de SQL server en daarin alle MRM tabellen te benaderen. Dit kan gebruikt worden om alle gegevens die nodig zijn uit te lezen.

Figuur 7



Figuur 8

Het direct aanspreken van een database brengt enkele risico's met zich mee. Als er te veel rechten worden toegekend aan het account waarmee de tabellen worden benaderd is het mogelijk data te wijzigen in MRM zonder validatie te hebben op je invoer. Deze aspecten worden in dit hoofdstuk besproken.

5.2 Verbinding

5.2.1 Database server

Op een server is het mogelijk aan te geven dat een database alleen vanuit die server te benaderen is. Alle applicaties die moeten communiceren moeten dan geïnstalleerd zijn op deze server. Als dit niet het geval is kunnen de applicaties geen verbinding maken met de database.

In het geval de database alleen op zijn eigen server benaderd kan worden moet de applicatie geïnstalleerd worden op dezelfde server als de database. Hierdoor kan de applicatie direct verbinding maken naar de database op localhost.

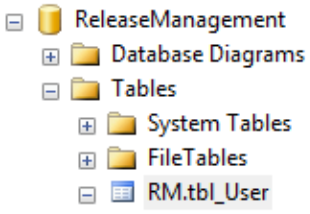
5.2.2 Database gebruiker

Binnen een SQL database is het mogelijk verschillende rollen en gebruikers te maken. Met behulp van deze rollen en gebruikers is het mogelijk om bepaalde rechten in de database toe te kennen. Deze rechten hebben invloed op het benaderen van gegevens uit de database. Als er niet wordt gewerkt met rollen en gebruikers wordt de database kwetsbaar voor ongewenste wijzigingen.

Voor het ophalen van gegevens is het niet nodig dat de database gebruiker alle rechten heeft. De database gebruiker hoeft niet te kunnen wijzigen en verwijderen. Verder is het ook niet noodzakelijk dat de database gebruiker rechten heeft voor het uitlezen van alle tabellen of velden in de database. Het moet dus alleen mogelijk zijn voor een user om de tabellen die hij nodig heeft uit te lezen.

In de installatie van de SQL server gaan wij uit van een “Discretionary Access Control” (DAC- toegangscontrole)[RF07]. Dit betekent dat een gebruiker geen rechten heeft in een database totdat expliciet toestemming is gegeven aan de gebruiker om een bepaalde actie uit te voeren. Met de DAC- toegangscontrole kunnen wij dus gemakkelijk afdwingen dat de database gebruiker die verantwoordelijk zal zijn voor het ophalen van de gegevens alleen recht heeft op het lezen van de gegevens. Een voorbeeld van het aanmaken van de gebruiker met zijn rechten is te zien in figuur 8.

```
use ReleaseManagement;

CREATE LOGIN mrm_dashbaord_ext
    WITH PASSWORD = 'mrmDashboardPa$$1';
GO

CREATE USER mrm_dashbaord_ext FOR LOGIN mrm_dashbaord_ext;
GO

CREATE ROLE release_management_external;
GO

GRANT SELECT
ON ReleaseManagement.RM.tbl_User
TO release_management_external;
GO

ALTER ROLE release_management_external
ADD MEMBER mrm_dashbaord_ext
```



Figuur 8

Hierin is te zien hoe de nieuwe gebruiker ook alleen recht heeft op de tabel die wij hebben opgegeven. Indien het nodig is dat alleen bepaalde kolommen uitgelezen kunnen worden is het mogelijk om nog een kolom aan te geven in de GRANT SELECT. Dit is handig om bijvoorbeeld geen inloggegevens uit te kunnen lezen van een user tabel.

5.2.3 Views

In plaats van het verstrekken van rechten om bepaalde tabellen direct uit te lezen is het ook mogelijk views te maken in een SQL database. Views zijn afgeleide tabellen die gegenereerd worden bij het opvragen van de view. Views worden gegenereerd aan de hand van een voorafgaande gedefinieerde expressie[RF07]. Met deze views is het mogelijk om de gegevens die wij nodig hebben uit de database eerst te definiëren. Na het definiëren van de views moet de nieuwe database gebruiker alleen maar rechten krijgen voor het lezen van deze views.

Door views te gebruiken zorgen wij ervoor dat het afgeven van rechten niet onoverzichtelijk wordt bij ingewikkelde tabellen. Een database administrator hoeft niet per tabel en per kolom te gaan kijken of de database gebruiker hier recht op heeft. De views moeten wel goedgekeurd worden voordat ze op de productieomgeving uitgevoerd mogen worden.

Het voordeel van het gebruiken van views is dat de interface waarmee wij moeten communiceren altijd vast staat. Als er een nieuwe versie van MRM uitgebracht wordt, die de database volledig wijzigt, moeten alleen de views

worden aangepast. In de views moeten de tabellen die worden bevraagd aangepast worden zonder de namen en kolommen van de views aan te passen. Hierdoor is het niet nodig om een applicatie die gebruik maakt van deze views aan te passen.

Een gevaar bij het gebruiken van views is dat een view snel complex kan worden. Tijdens het generen van de views kan het mogelijk zijn dat er overbodige tabellen worden bevraagd en hierdoor de database wat langzamer wordt.

5.3 Analyse Database tabellen

5.3.1 Verschil Release en ReleaseV2

In de database valt het op dat er twee Release tabellen bestaan. Namelijk tabellen met Release en andere tabellen met ReleaseV2 erin. Er zijn namelijk twee verschillende manieren om een applicatie uit te rollen. Deze twee manieren zijn “Agent-based Release” en “vNext Release”. Bij het uitvoeren van een “Agent-based Release” maakt MRM gebruik van de tabellen zonder V2 en voor het uitvoeren van een “vNext Release” worden de tabellen met een V2 gebruikt. Voor het ophalen van de verschillende “releases” moeten beide tabellen bevraagd worden.

5.3.2 Benodigde tabellen

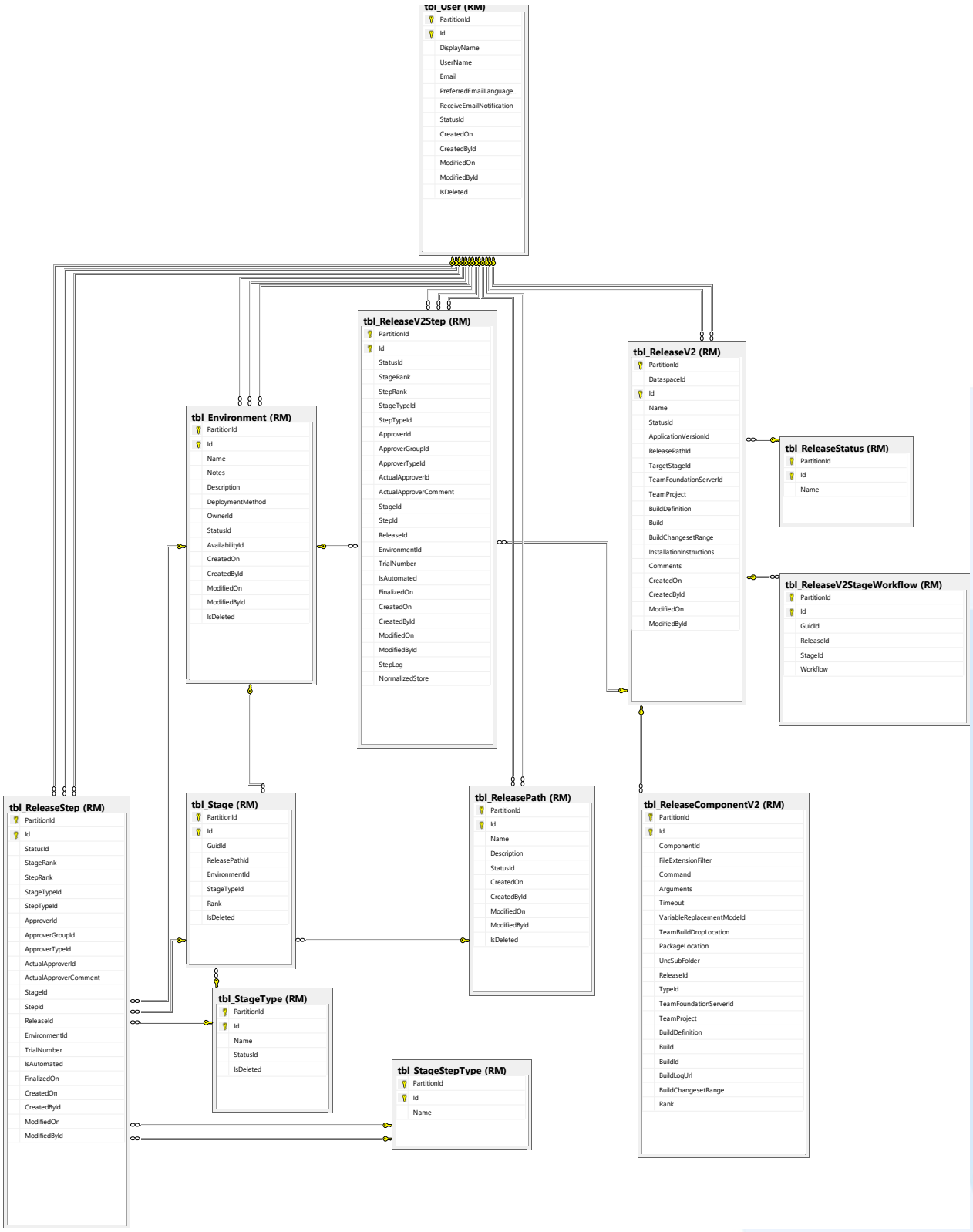
De database van MRM 2015 update 1 bestaat uit 122 tabellen. 10 van deze tabellen zijn gemarkeerd met “System” en de rest met “RM”. Alleen de tabellen met “RM” ervoor zijn van belang voor het ophalen van de gegevens. Aangezien er een groot aantal tabellen zijn wordt er alleen gekeken naar de verplichte gegevens die in hoofdstuk 2.1 zijn beschreven. Verder zal er globaal gekeken worden welke andere gegevens uit de database gehaald kunnen worden.

In figuur 1 was duidelijk te zien uit welke stappen een deploymentstraat bestaat. Uit het figuur is af te lijden dat een deploymentstraat wordt weergegeven in een Release Path. Deze Release Path is vervolgens opgedeeld in Stages waarin Environments bestaan. Verder wordt er per stage ook Actions aangegeven die uitgevoerd moeten worden. Vervolgens zien wij ook dat het mogelijk is om Approvers aan te geven. Als laatste is het ook te zien dat er Servers aanwezig zijn in elke Stage.

Met deze gegevens wordt de database geanalyseerd om te kijken of deze informatie terug te vinden is in de database structuur. Verder maken wij ook gebruik van het artikel in RF06 om een basis te krijgen welke tabellen nodig zijn. De tabellen uit dit artikel komen uit RMR 2013, maar bestaan allemaal nog in MRM 2015 update 1. Voor de “Agent-based Release” en “vNext Release” worden de volgende tabellen gebruikt:

“Agent-based”	“vNext”	Beide
Release	ReleaseV2	Environment
ReleaseComponent	ReleaseComponentV2	ReleasePath
ReleaseConfigurationVariable	ReleaseConfigurationVariableV2	ReleaseStatus
ReleaseStageWorkflow	ReleaseV2StageWorkflow	ReleaseStep
ReleaseStep	ReleaseV2Step	ReleaseStepStatus
		Stage
		StageType
		StageStepType
		SecurityGroup
		User

Om te weten welke Actions horen bij een Stage moet de tabel Component ook bevraagd worden. Hierin staan namelijk alle Actions die uitgevoerd kunnen worden. Voor de verbinding met SQL server is er een diagram gemaakt met de “vNext Release” tabellen. De tabellen voor “Agent-based Release” zien er hetzlefde uit. In figuur 9 is dit diagram weergegeven.

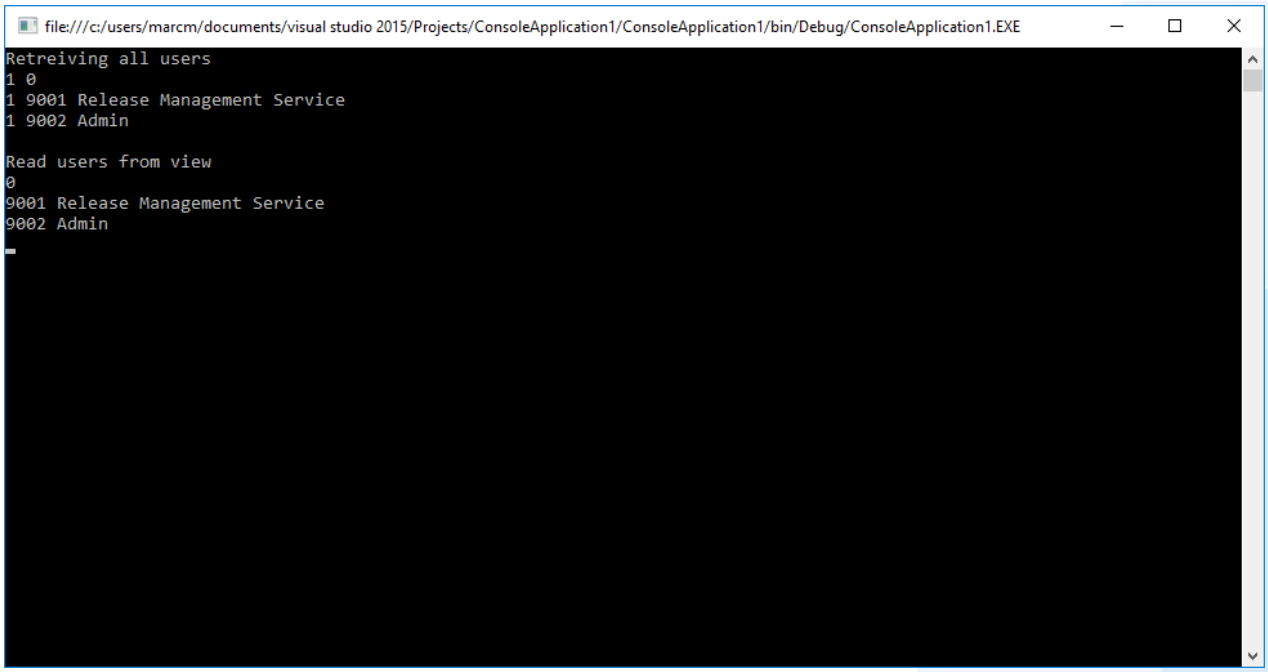

tbl_User (RM)
PartitionId
Id
DisplayName
UserName
Email
PreferredEmailLanguage...
ReceiveEmailNotification
StatusId
CreatedOn
CreatedById
ModifiedOn
ModifiedById
IsDeleted
tbl_ReleaseV2Step (RM)
PartitionId
Id
StatusId
StageRank
StepRank
StageTypeId
StepTypeId
ApproverId
ApproverGroupId
ApproverTypeId
ActualApproverId
ActualApproverComment
StageId
StepId
ReleaseId
EnvironmentId
TrialNumber
IsAutomated
FinalizedOn
CreatedOn
CreatedById
ModifiedOn
ModifiedById
StepLog
NormalizedStore
tbl_ReleaseV2 (RM)
PartitionId
DataspaceId
Id
Name
StatusId
ApplicationVersionId
ReleasePathId
TargetStageId
TeamFoundationServerId
TeamProject
BuildDefinition
Build
BuildChangesetRange
InstallationInstructions
Comments
CreatedOn
CreatedById
ModifiedOn
ModifiedById
tbl_ReleaseStatus (RM)
PartitionId
Id
Name
tbl_ReleaseV2StageWorkflow (RM)
PartitionId
Id
GuidId
ReleaseId
StageId
Workflow
tbl_Environment (RM)
PartitionId
Id
Name
Notes
Description
DeploymentMethod
OwnerId
StatusId
AvailabilityId
CreatedOn
CreatedById
ModifiedOn
ModifiedById
IsDeleted
tbl_ReleaseStep (RM)
PartitionId
Id
StatusId
StageRank
StepRank
StageTypeId
StepTypeId
ApproverId
ApproverGroupId
ApproverTypeId
ActualApproverId
ActualApproverComment
StageId
StepId
ReleaseId
EnvironmentId
TrialNumber
IsAutomated
FinalizedOn
CreatedOn
CreatedById
ModifiedOn
ModifiedById
tbl_Stage (RM)
PartitionId
Id
GuidId
ReleasePathId
EnvironmentId
StageTypeId
Rank
IsDeleted
tbl_StageType (RM)
PartitionId
Id
Name
StatusId
IsDeleted
tbl_StageStepType (RM)
PartitionId
Id
Name
tbl_ReleasePath (RM)
PartitionId
Id
Name
Description
StatusId
CreatedOn
CreatedById
ModifiedOn
ModifiedById
IsDeleted
tbl_ReleaseComponentV2 (RM)
PartitionId
Id
ComponentId
FileExtensionFilter
Command
Arguments
Timeout
VariableReplacementModeId
TeamBuildDropLocation
PackageLocation
UncSubFolder
ReleaseId
TypeId
TeamFoundationServerId
TeamProject
BuildDefinition
Build
BuildId
BuildLogUrl
BuildChangesetRange
Rank

Figuur 9

Met gegevens uit deze tabellen is de deploymentstraat te realiseren. Indien er wegens een uitbreiding meer gegevens uit MRM 2015 update 1 gehaald moeten worden is dit mogelijk, omdat alle tabellen in de database ter beschikking zijn.

5.3.3 Opvragen gegevens proof of concept

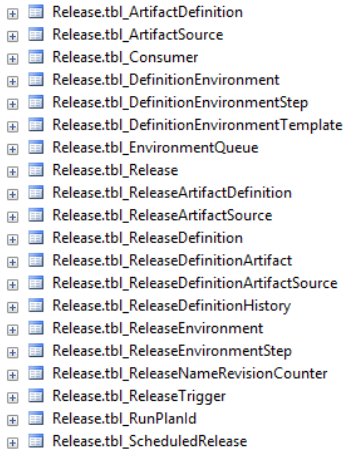
Om te bewijzen dat het mogelijk is om gegevens uit de database van MRM 2015 update 1 te lezen is er een POC gemaakt. In deze POC wordt er een verbinding met de database gelegd. Met deze verbinding worden alle gebruikers in MRM opgehaald. Het ophalen van de gebruikers wordt eerst gedaan door het uitlezen van de User tabel. Vervolgens worden alle met behulp van een view ingeladen om te bewijzen dat het ook mogelijk is om de gegevens uit een view te halen.



Figuur 10

5.4 Toekomst Release Management

In MRM 2015 update 2 hebben er veel wijzigingen plaats gevonden in de database. De database van MRM is nu geïntegreerd met die van TFS. Tijdens het installeren van TFS 2015 update 2 worden er twee databases aangemaakt, namelijk: Tfs_Configuration en Tfs_DefaultCollection. In de database DefaultCollection vinden wij tabellen van MRM terug. Deze tabellen zijn nu gegroepeerd met Release ervoor.



Figuur 11

Indien er gebruik wordt gemaakt van een verbinding met de database zal er in toekomstige versies hiermee rekening gehouden moeten worden. De database kan voortdurend aangepast worden. Waardoor de database verbinding bij elke update aangepast moet worden. Indien er views gebruikt worden, moeten de views ook aangepast worden bij een update.

De nieuwe tabellen van MRM 2015 update 2 zijn ook diep verbonden met de tabellen van TFS en haar andere diensten. Dit betekent dat het ingewikkelder gaat worden om de relaties tussen de tabellen te vinden.

Als laatste is het ook te zien dat wanneer er een nieuwe project wordt aangemaakt in TFS dat de gebruiker een optie krijgt om een nieuwe database schema te maken voor dat project. Dit betekent dat er in sommige gevallen meerdere schema's bevraagd moeten worden.

5.5 Voor- en nadelen van gebruik database

Indien er besloten wordt om een koppeling direct met de database te maken zijn er een voor- en nadelen aan verbonden.

5.5.1 Voordelen gebruik database

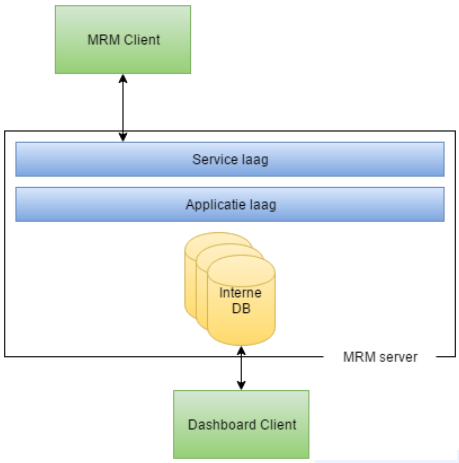
Alles ter beschikking
Een van de voordelen van het direct verbinden met de database is dat al de informatie die erin zit ter beschikking is. Bij een API zal het nodig zijn om te werken met de gegevens die je kan verkrijgen. Als de gegevens van een API niet voldoende zijn is er geen alternatief. Bij het gebruiken van de database zijn alle tabellen te benaderen en dus ook al het informatie.

Communicatie duidelijk
Het verbinden met een database is een bekende techniek en brengt ook geen moeilijkheden met zich mee. Indien er verder informatie over nodig is, is het onderwerp volledig gedocumenteerd.

Duidelijke toekomst
De toekomst van de verbinding is duidelijk. Bij elke nieuwe versie van MRM moet alleen de database geanalyseerd worden om te kijken hoeveel het is veranderd. Alle versies van MRM maken gebruik van een database en zijn daarom allemaal te benaderen.

5.5.2 Nadelen gebruik databases

Alles ter beschikking (Geen encapsulatie)
Naast dat dit een voordeel is wordt het ook gelijk genoteerd als een nadeel. Door de database te kunnen benaderen is er een grote verlies op het gebied van encapsulatie. MRM bepaalt niet meer zelf hoe er wordt omgegaan met de gegevens die hij intern opslaat. Verder kunnen er zelfs aanpassingen in de database gebeuren zonder dat MRM daar wat van af weet. In figuur 12 is te zien hoe een applicatie direct verbinding maakt en alle lagen van MRM overslaat.



Figuur 12

Overzicht
De database van MRM 2015 update 1 bestaat uit meer dan 100 tabellen. Met verloop van de tijd is het niet meer duidelijk welke tabellen bevraagd worden. Dit maakt het moeilijker om wijzigingen door te voeren in de applicatie.

Rechten beheren
De rechten van de database gebruiker die de database benaderd moeten duidelijk worden opgesteld en bijgewerkt worden. Als er teveel rechten worden weggegeven kan een applicatie data gaan wijzigen of zelfs verwijderen.

Wijzigende structuur
Na een update in MRM komen er vaak nieuwe tabellen erbij of worden ze verwijderd. Zoals te zien in hoofdstuk 5.4 is de structuur tussen twee updates extreem veranderd. Na zo een update moet een de datalaag van een applicatie volledig aangepast worden.

Wijzigende views
Indien er gekozen wordt om met views te werken moeten deze views worden aangepast na een update. Hiervoor moet ook uitgebreid getest worden of de views nog dezelfde informatie tonen zoals voorheen.

Nieuwe database verbindingen
Tussen MRM 2015 update en update 2 is de database verhuist. In update 1 was het een alleenstaande database, maar in update 2 stonden de gegevens in dezelfde database als TFS. Dit betekent bij een nieuwe update is de database mogelijk niet te benaderen, omdat er meer risico is als er iets fout gaat in de TFS database.

6. Definities

Code	Woord	Uitleg
DF01	Deploymentstraat	Een traject dat een applicatie automatisch doorloopt. In dit traject wordt de applicatie automatisch geïnstalleerd over verschillende omgevingen.

Bijlage III

Functioneel ontwerp

Functioneel ontwerp

Functioneel ontwerp van de opdracht Visual Deployment Pipeline

Marc De Meza

Inhoud

1. Inleiding

Dit document bevat de functionele ontwerpen van de opdracht Visual Deployment pipeline. In dit document is terug te vinden welke eisen en wensen aan het systeem heeft en hoe dit is geanalyseerd.

2. User stories

In dit hoofdstuk zijn de user stories van het project te vinden. Deze user stories zijn ook te vinden inTeam Foundation Server(TFS). In TFS zijn alle user stories opgeslagen.

Nummer	Story	Prioriteit	TFS id
US1	Als gebruiker wil ik een overzicht van Release Paths zodat ik een Release Path kan selecteren	1	9096
US2	Als gebruiker wil ik een detail scherm van een Release Path zien zodat ik alle informatie van die Release Path kan zien	2	9071
US3	Als gebruiker wil ik een overzicht van ReleaseTemplates die zijn gekoppeld aan een ReleasePath zodat ik kan zien welke applicaties bij welke pad horen	3	9477
US4	Als gebruiker wil ik een overzicht van alle Releases die bij één ReleasePath horen, zodat ik een overzicht kan hebben welke releases zijn uitgebracht van een bepaalde Release Path	4	9255
US5	Als gebruiker wil ik een overzicht van alle Stages die bij een Release horen zodat ik kan zien welke Stages bij de Release betrokken zijn	5	9444
US6	Als gebruiker wil ik een overzicht van Steps die horen bij een Stage, zodat ik duidelijk kan zien welke stappen zijn ondernomen in een bepaalde Stage	6	9445
US7	Als gebruiker wil ik een overzicht van Omgevingen zodat ik een keuze kan maken van welke omgeving ik een detail scherm te zien krijg	7	9446
US8	Als gebruiker wil ik een overzicht van applicaties die op een Omgeving zijn geïnstalleerd zodat ik duidelijk kan zien welke applicatie waar is geïnstalleerd	8	9447
US9	Als gebruiker wil ik een overzicht van applicaties en de versies die betrokken waren bij een omgeving van een bepaalde release zodat ik inzicht kan krijgen in welke applicaties hier bij betrokken waren	9	9831
US10	Als gebruiker wil ik dat er nieuwe data ingeladen wordt in mijn schermen als er wijzigingen hebben plaats gevonden in release management	10	10065
US11	Als gebruiker wil ik kunnen instellen welke Release Paths ik te zien krijg in mijn overzichten, zodat ik alleen informatie zie dat voor mij belangrijk is	11	10069
US12	Als gebruiker wil ik in een Release per Step meer informatie zien over welke stappen daarin zijn uitgevoerd en welke status ze hebben	12	10068
US13	Als gebruiker wil een scherm waarin de meest recente releases staan	13	10366
US14	Als gebruiker wil ik een bij elke step in een stage van een release een waarschuwing krijgen als de status van de step ‘failed’ is	14	10385

2.1 Acceptatiecriteria

Per user story heb ik samen met de opdrachtgever criteria opgesteld om te weten wanneer een user story is afgerond. Hieronder volgt per use case de acceptatiecriteria.

Nummer	Criteria
US1	**Functioneel** 1. Alle release paths in MRM moeten getoond worden 2. Van alle release paths moeten de naam en beschrijving bij staan **Niet-functioneel** 1. Als er geen release paths aanwezig zijn moet er een melding komen 2. Als er geen verbinding is met MRM moet er een waarschuwing zijn 3. Alleen MRM versie 2015 update 1 wordt ondersteund 4. Alleen Internet Explorer versie 10 en hoger worden ondersteund 5. Op tablets wordt alleen de safari browser ondersteund 6. De aantal verzoeken naar MRM moeten minimaal blijven(cache) 7. Er moeten Microsoft Technologieën gebruikt worden 8. De API is geschreven met in C# m.b.v. Webapi 9. De Front-end is geschreven met AngularJS
US2	**Functioneel** 1. Er moet een details scherm zijn van een gekozen release path a. In de details scherm moeten alle stages van een release path getoond worden i. Per stage moet de naam worden getoond ii. Pet stage moeten alle steps worden getoond 1. Per step moet de naam en security group worden getoond 2. De namen van alle security groups van de release path moeten getoond worden **Niet-functioneel** 1. Als de release path niet bestaat moet er een waarschuwing getoond worden 2. Als het informatie van de release path ouder is dan een aantal minuten (variabel) moeten er nieuwe gegevens worden ingeladen
US3	**Functioneel** 1. In de detail schermen van een release path moeten de namen van de release templates die zijn gekoppeld getoond worden **Niet-functioneel** 1. Als er geen release templates zijn moet er een waarschuwing getoond worden

Nummer	Criteria
US4	**Functioneel** 1. Er moet een overzicht komen met alle releases die zijn uitgebracht voor een bepaalde release path 2. Van deze releases moet de naam en de status worden getoond **Niet-functioneel** 1. Als het informatie van de releases ouder is dan aantal minuten (variabel) moeten er nieuwe gegevens worden ingeladen (cache) 2. Als er geen releases zijn moet er een waarschuwing worden getoond
US5	**Functioneel** 1. In de overzicht van releases moet er per release een overzicht zijn van alle stages die bij deze release horen 2. De naam van de stage moet zichtbaar zijn 3. De huidige stage moet zichtbaar zijn 4. De naam van de environment van de stage zichtbaar zijn **Niet-functioneel** 1. Voor de gebruiker moet er met icoontjes duidelijk worden gemaakt status een release heeft (X of vinkje) 2. Als het informatie van de stage ouder is dan aantal minuten (variabel) moeten er nieuwe gegevens worden ingeladen (cache)
US6	**Functioneel** 2. Alle steps binnen een stage moeten zichtbaar zijn 3. Van een step met de naam, begin tijd en status weergegeven worden **Niet-functioneel** 1. Bij de statussen moeten verschillende kleuren hebben(blauw, groen, rood) 2. Als het informatie van de step ouder is dan aantal minuten (variabel) moeten er nieuwe gegevens worden ingeladen (cache)

Nummer	Criteria
US7	**Functioneel** 1. Er moet een overzicht komen van alle omgevingen in MRM 2. Van de omgevingen moet de naam, beschikbaarheid, beschrijving, status, eigenaar en deployment method zichtbaar zijn **Niet-functioneel** 1. Als het informatie van de omgevingen ouder is dan aantal minuten (variabel) moeten er nieuwe gegevens worden ingeladen (cache) 2. Als er geen omgevingen in het systeem staan moet er een waarschuwing getoond worden 3. Als er geen verbinding is moet de gebruiker een foutmelding krijgen
US8	**Functioneel** 1. Er moet een detail scherm komen van een omgeving 2. In het detail scherm moet de naam van een omgeving getoond worden 3. Er moet een overzicht zijn van applicaties die op die omgeving zijn geïnstalleerd 4. Er moet een versie (build) nummer staan bij elke applicatie **Niet-functioneel** 1. Als er geen versie nummer is moet er staan dat deze niet aanwezig is 2. Als er geen applicaties zijn moet er een waarschuwing komen 3. Als er geen verbinding is met de server moet er een error zijn
US9	**Functioneel** 1. In de overzichten van releases moet ik release kunnen selecteren en kijken welke applicaties daarop zijn geïnstalleerd 2. Er moet een overzicht komen van alle applicaties die bij een bepaalde stage zijn betrokken 3. Per applicatie moet de naam en versie worden weergegeven 4. De applicatie van die release moet ook duidelijk worden weergegeven 5. Van de applicatie die betrokken was bij het installeren moet de versie en naam worden weergegeven **Niet-functioneel** 1. De applicatie waar het om ging moet ook duidelijk weergegeven worden (apart staan) 2. Het dashboard geeft een error als er geen verbinding is 3. Het dashboard geeft een waarschuwing als er geen applicaties aanwezig zijnss

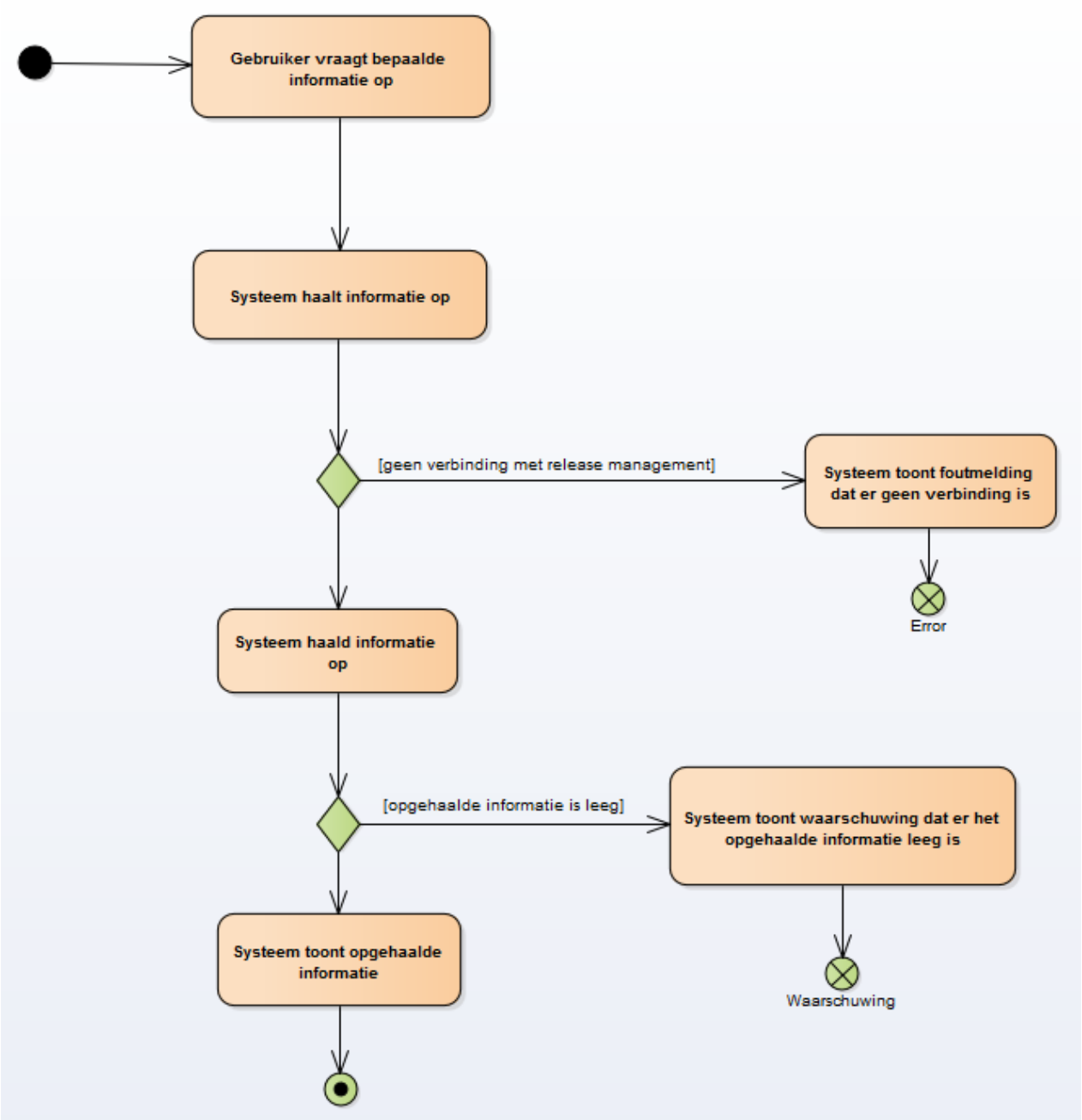
Nummer	Criteria
US10	**Functioneel** 1. Als er wijzigingen zijn in release management moeten deze ook zichtbaar worden op een scherm waar dat informatie op staat **Niet-functioneel** 1. Het wijzigen van informatie moet zodanig gebeuren dat de gebruiker geen storende wijzigingen in het scherm ziet (vloeiende wijzigingen)
US11	**Functioneel** 2. Het moet mogelijk zijn om een selectie te maken van release paths die je in alle overzichten wilt zien 3. In de overzicht van release paths worden alleen de paths gekozen die zijn geselecteerd 4. In de overzicht van applicaties op omgevingen worden alleen applicaties van de selectie release paths getoond 5. Als er geen keuze is gemaakt moeten alle paths getoond worden **Niet-functioneel** 1. De instellingen moeten voor alle gebruikers apart in te stellen zijn
US12	**Functioneel** 1. Van alle steps in een release moeten alle componenten zichtbaar zijn **2.** Van de componenten moeten de naam, status en start datum worden weergegeven **Niet-functioneel** 1. Als het informatie van de step ouder is dan aantal minuten (variabel) moeten er nieuwe gegevens worden ingeladen (cache)
US13	**Functioneel** 1. Er moet een scherm komen waarin alle releases van de geselecteerde release paths worden weergegeven 2. Er moet een selectie worden gemaakt van hoeveel releases weergegeven moeten worden (3, 5, 10, 15, 20) 3. De releases moeten in volgorde van meest recent tot oudste staan **Niet-functioneel** 1. Als het informatie van de releases ouder is dan aantal minuten (variabel) moeten er nieuwe gegevens worden ingeladen (cache) 2. Niet alle releases in MRM moeten opgehaald worden. Het moet mogelijk zijn om aan te geven tot hoever in het verleden releases worden opgehaald(instellingen in web.config)

Nummer	Criteria
US14	**Functioneel** 1. Als er een fout is opgetreden in een step moet dit worden weergegeven aan de gebruiker **Niet-functioneel** 1. De step moet een rood icoontje krijgen waarmee het duidelijk wordt dat er een fout is 2. Het icoontje van de step moet van zichtbaar naar onzichtbaar gaan (blinken)

2.2 Activity diagrammen

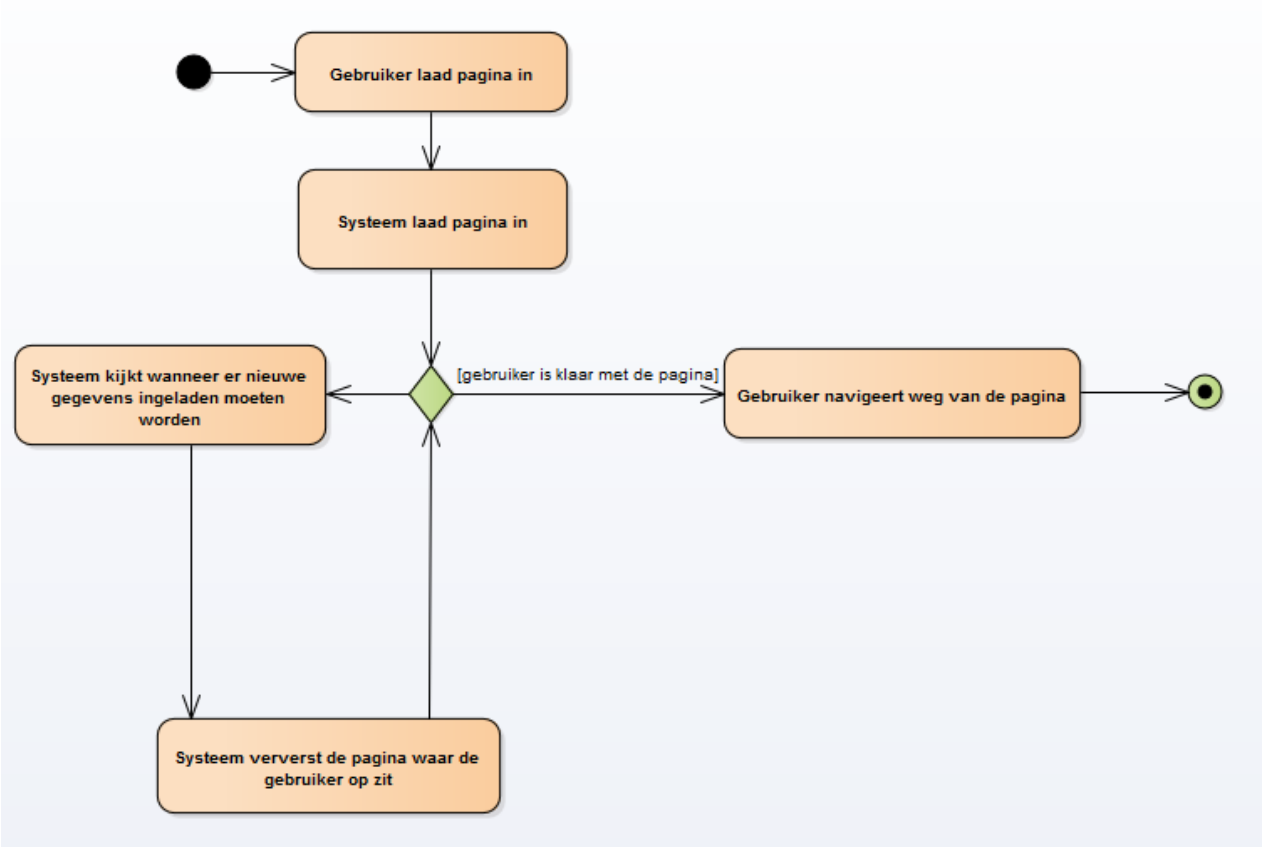
Bij sommige acties is het handig om een duidelijk beeld te krijgen hoe het systeem precies moest reageren. Bijvoorbeeld bij het tonen van waarschuwingen en foutmelding. Waar het nodig is had ik activity diagrammen gemaakt.

Tonen waarschuwing en foutmelding



Figuur 1 verloop van foutmeldingen

Herladen van gegevens
Voor het herladen van gegevens op het dashboard zijn er meerdere mogelijkheden die gebruikt kunnen worden. Deze mogelijkheden worden besproken in het Technisch ontwerp. Om een duidelijk beeld te krijgen hoe het updaten moet verlopen is er een activity diagram gemaakt hierover.

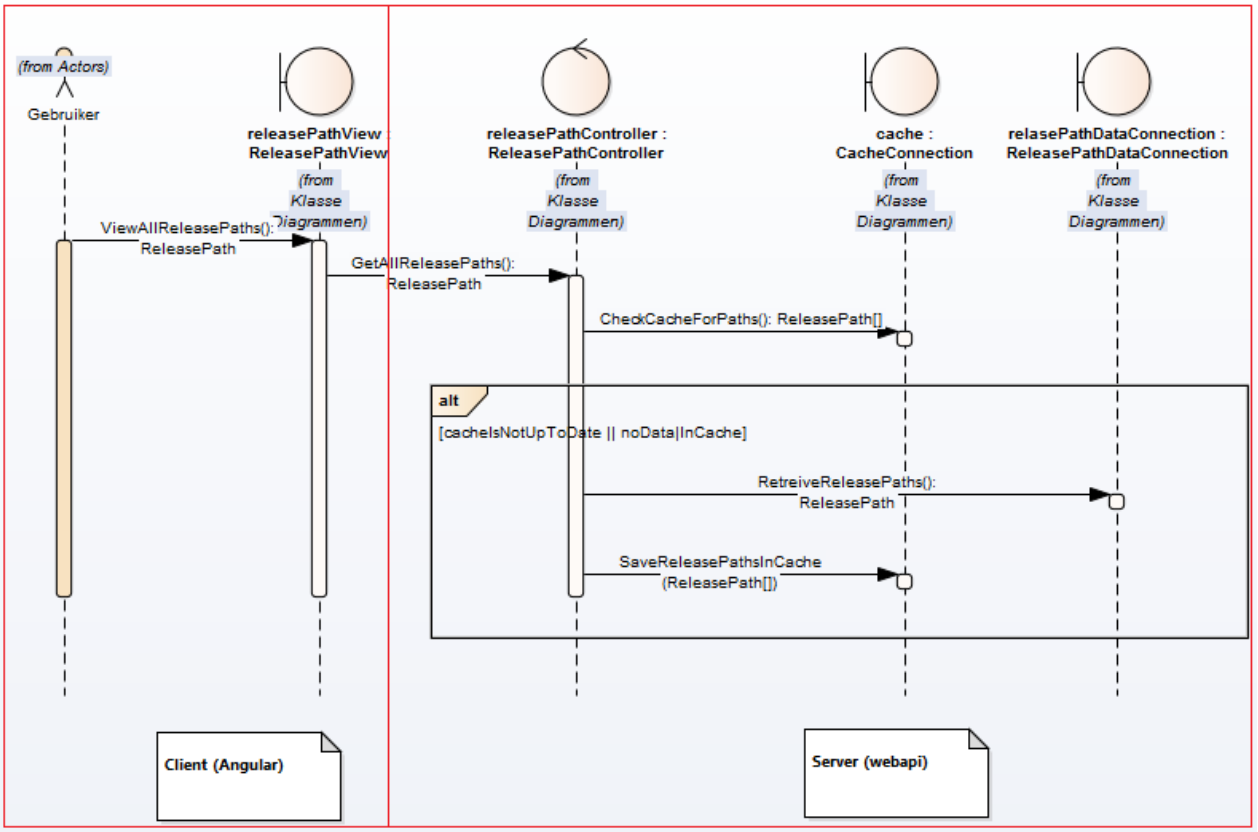


Figuur 2 herladen gegevens

2.3 Sequentie diagrammen

Om aan een analyse klasse diagram te komen moet ik begrijpen hoe bepaalde informatie door de applicatie verwerkt wordt. Ik moet weten met welke klassen een gebruiker communiceerde en hoe deze klasse gegevens ophalen uit Release Management.

Ophalen entiteiten
Voor het ophalen van alle entiteiten heb ik één sequentie diagram gemaakt. Alle ophaal acties verlopen op dezelfde manier. Indien het anders verloopt wordt dit in een andere sequentie weergegeven. Door figuur 3 heb ik een beter beeld gekregen welke klassen en welke methoden er precies per entiteiten ontstaan.



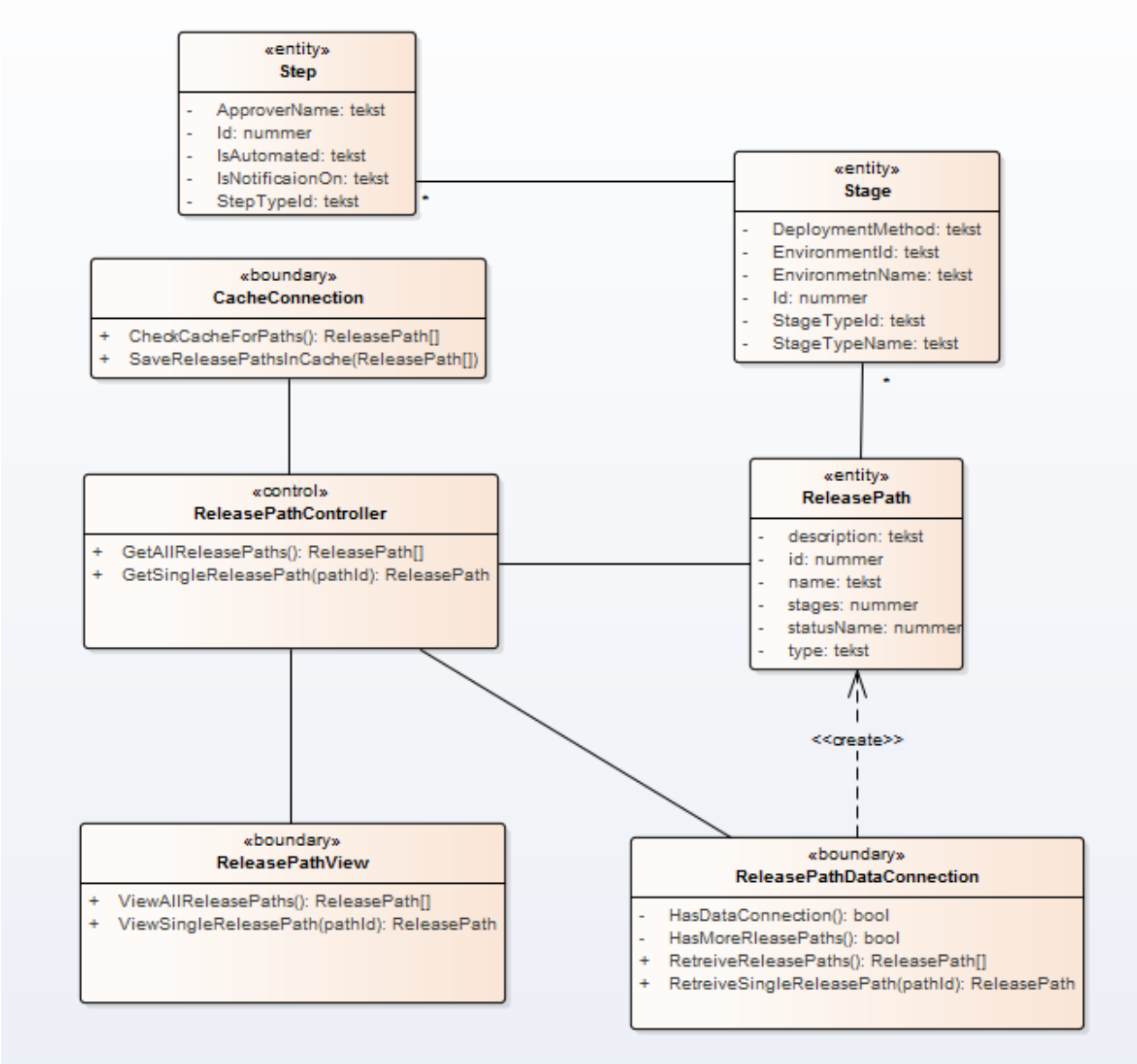
Figuur 3 ophalen gegevens

In figuur 3 is te zien hoe de gebruiker eerst met een client moet communiceren. In dit geval is dat een Angular website. Vervolgens communiceert de Angular client met een de server. De server controleert eerst of er gegevens zijn opgeslagen in de cache. Als er recente gegevens in de cache zijn worden deze opgehaald. Indien dit niet het geval is worden de nieuwe gegevens van MRM ingeladen en opgeslagen in de cache.

3. Analyse diagram

Om ervoor te zorgen dat het analyse klasse diagram overzichtelijk blijft heb ik besloten om het analyse diagram op te delen per sprint. Alleen de nuttige klassen zijn opgenomen. Deze diagrammen worden per sprint gebruikt om de opdrachtgever duidelijk te maken wat precies gerealiseerd wordt.

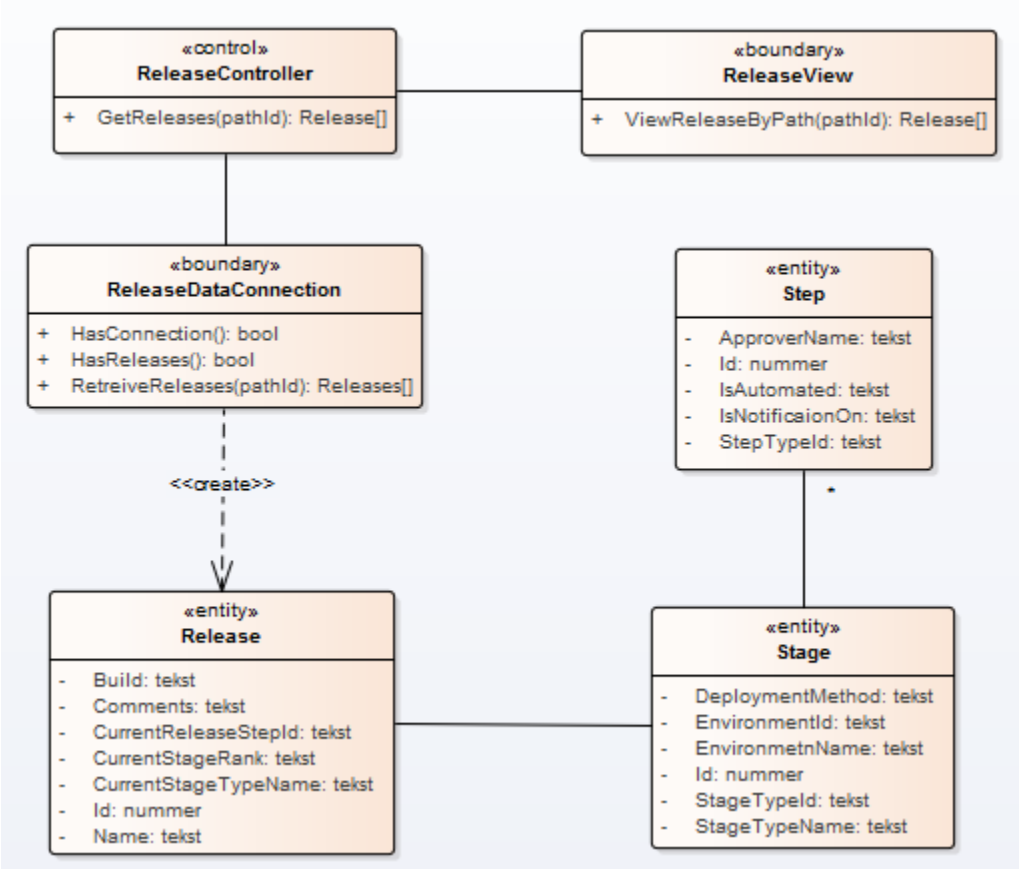
3.1 Release Paths en Releases



Figuur 4 analyse klasse diagram sprint 1

Beslissing
In de eerste sprint heb ik een analyse diagram gemaakt van de ReleasePath functionaliteit voordat ik begon met programmeren. Dit diagram gebruik ik om voor mezelf en de opdrachtgever duidelijk te maken wat er precies gerealiseerd gaat worden. In dit diagram is te zien dat ik een view wilde maken waar de gebruiker mee kan communiceren. Verder is er ook een klasse tussen de view en de verbinding voor Release Management. Dit doe ik om duidelijk te maken dat de verantwoordelijkheden van de klassen worden verdeeld.

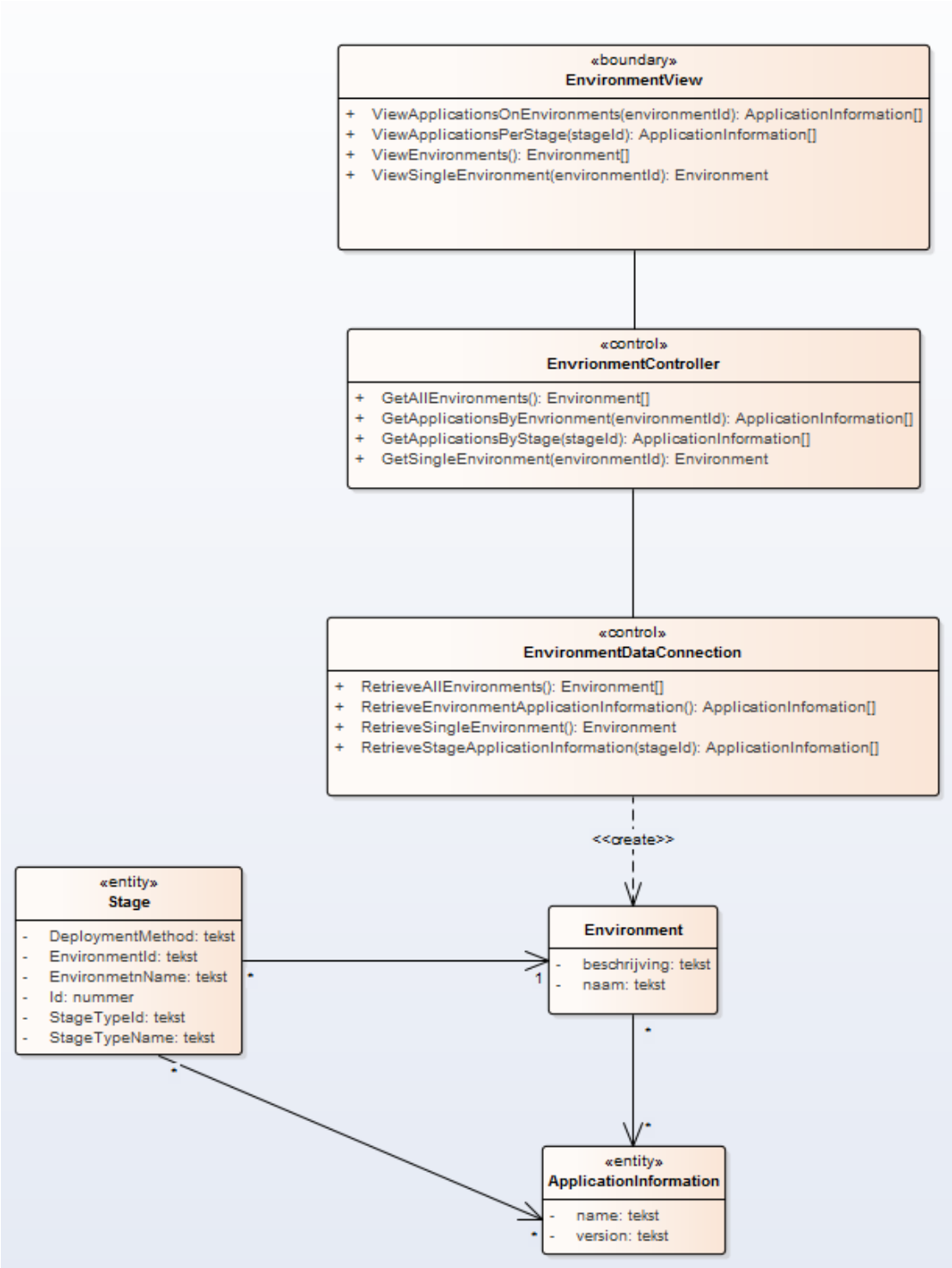
3.2 Releases



Figuur 5 analyse diagram sprint 2

Beslissing
Deze sprint heb ik een klasse diagram gemaakt van alles dat te maken heeft met de onderdelen Release. Dit doe ik om een duidelijk beeld te krijgen hoe alles met elkaar relateert. Met die diagram kan ik zien wat de relatie is tussen een Release, een Stage en een Step.

3.3 Omgevingen en versies



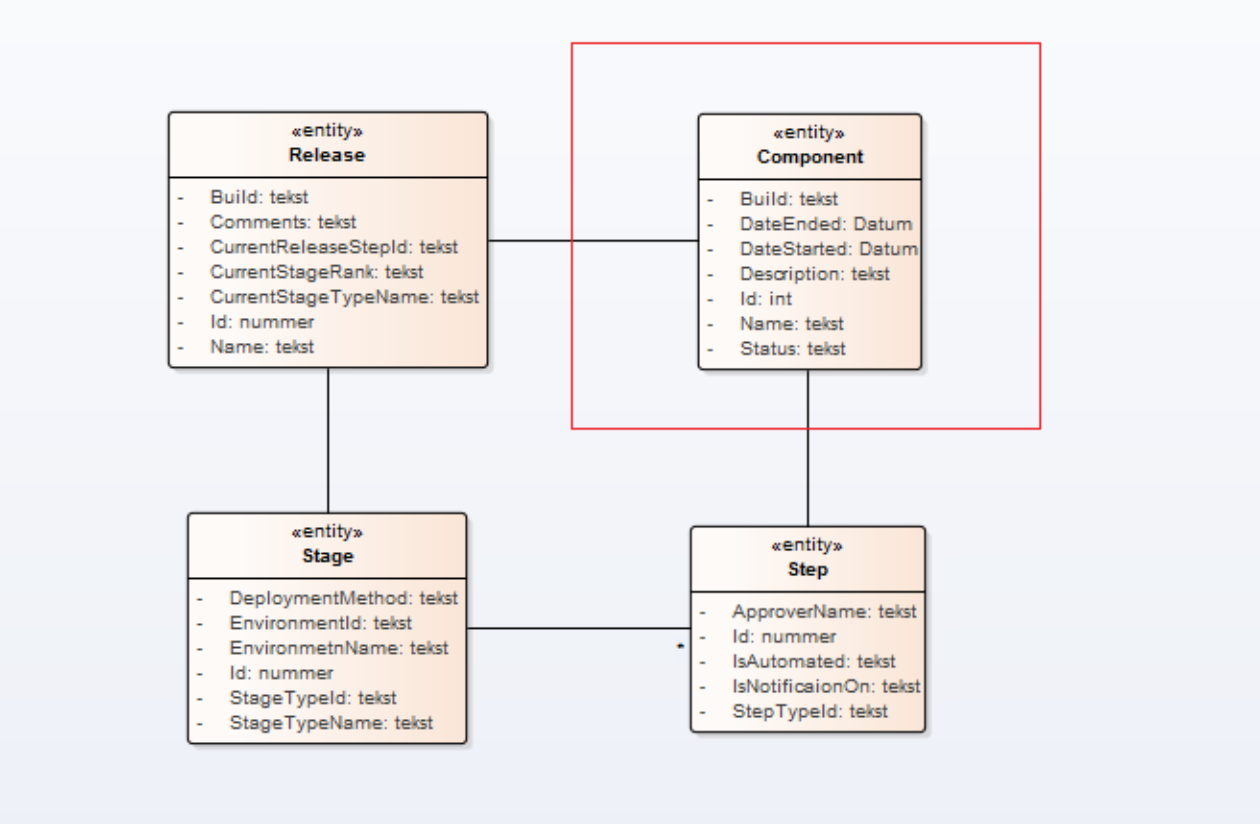
Figuur 6 analyse diagram sprint 3

Beslissing

Voor het tonen van alle versies van applicatie is het belangrijk dat de naam en versie getoond wordt. In figuur 6 is te zien dat dit een aparte klasse is. Waar het informatie precies vandaan komt is nog niet duidelijk tijdens het analyseren. Daarom is het zo weergegeven. Het is mogelijk dat dit informatie al beschikbaar is binnen de Stage of Environment. Verder is er ook te zien dat alle communicatie van de ApplicationInformation zal gaan door de Environment/view/controller.

3.4 Instellingen, live updates en component informatie

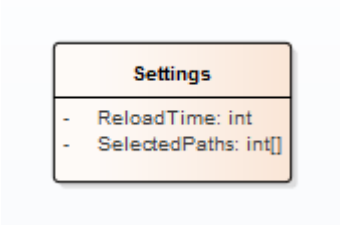
In sprint 4 werk ik aan het realiseren van live updates, instellingen en tonen van gedetailleerde informatie van een Release. In figuur 7 zijn de nieuwe klassen van deze sprint te zien.



Figuur 7 nieuwe entiteit klasse sprint 4

MRM gebruikt de component klasse om aan te geven welke stappen in een Step wordt uitgevoerd. Hierbij wordt ook de status bijgehouden. Verder gebruikt MRM dit ook om aan te geven welke componenten er verder bij een Release worden gebruikt. Deze nieuwe klasse zal door klassen in figuur 5 gebruikt worden.

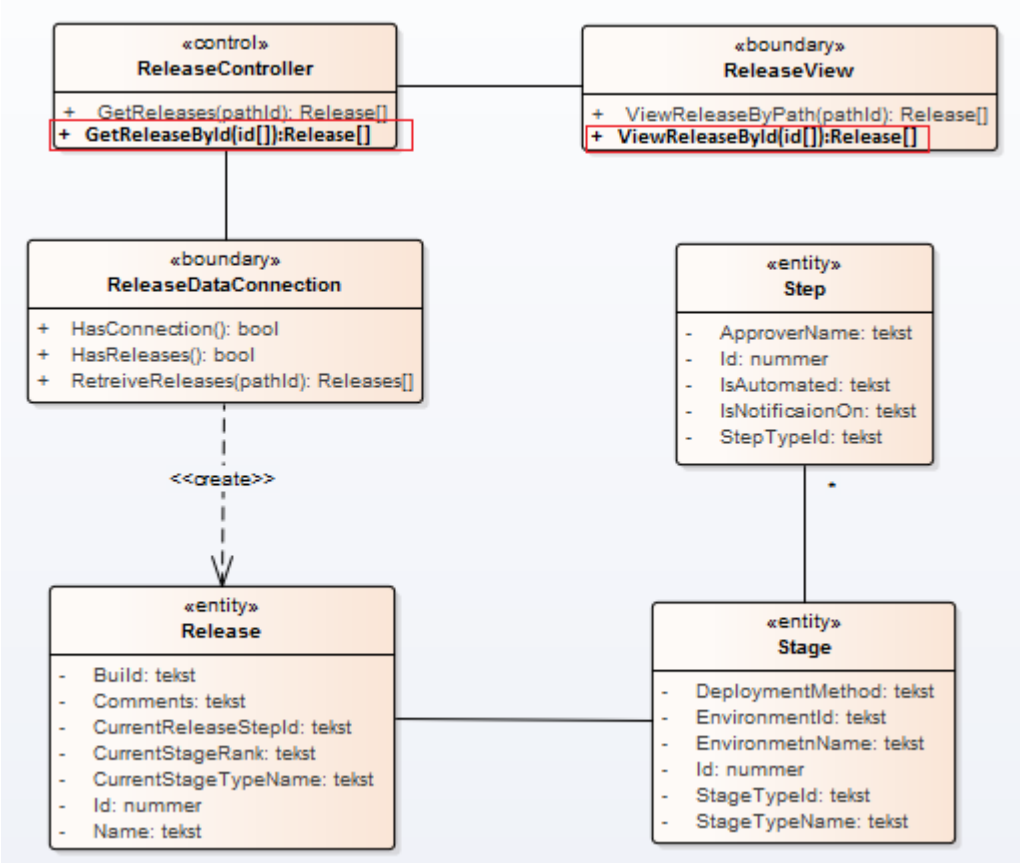
Verder moeten er ook instellingen opgeslagen worden. Deze instellingen zorgen ervoor dat een gebruiker alleen informatie ziet van ReleasePaths die hij heeft gekozen. Verder moet het ook mogelijk zijn om een interval aan te geven. Deze interval wordt gebruikt voor het ophalen van gegevens.



Figuur 8 settings gegevens

3.5 Waarschuwing fout in stage en algemene Release overzicht

In sprint 5 werk ik aan het realiseren van een algemene overzicht van Releases. Deze overzicht zal op het hoofdscherm van de applicatie te zien zijn. Hierdoor krijgt een gebruiker direct inzicht in de Releases. De setting voor het filtreren zal gelijk hierop van toepassing zijn. Het analyse klasse diagram is als volgt aangepast:

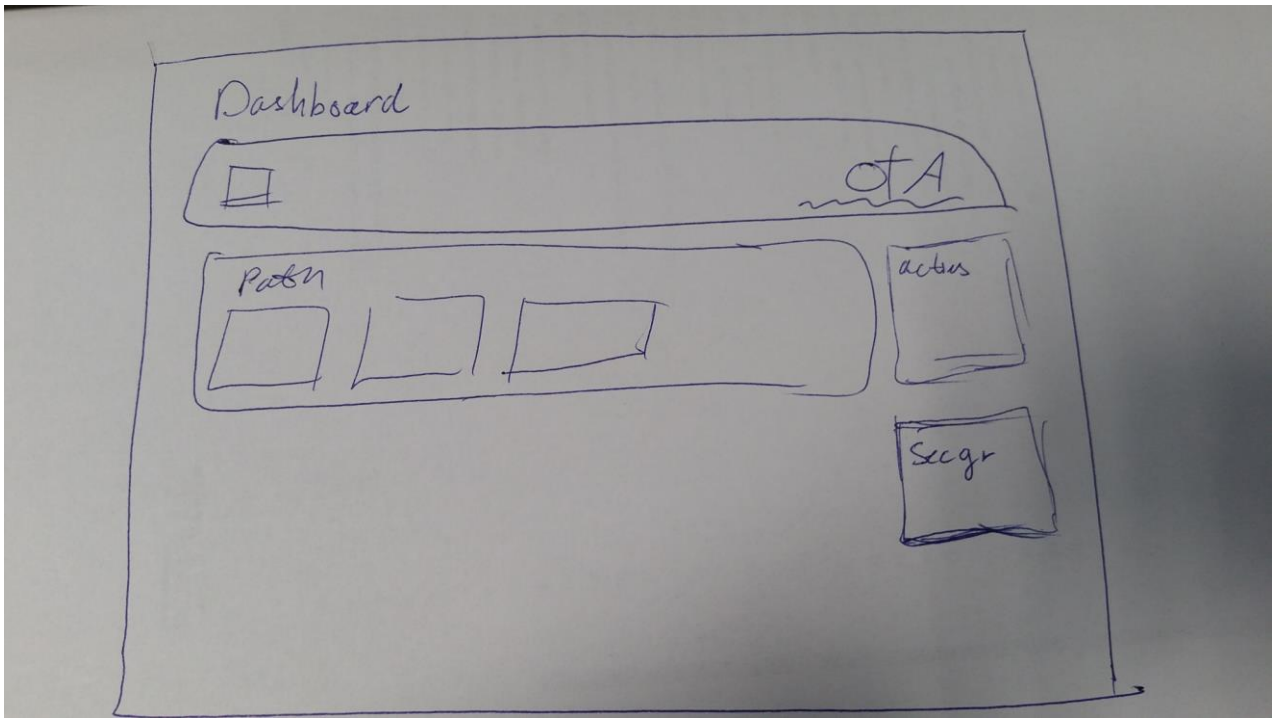


Figuur 9 wijzigingen Release controller

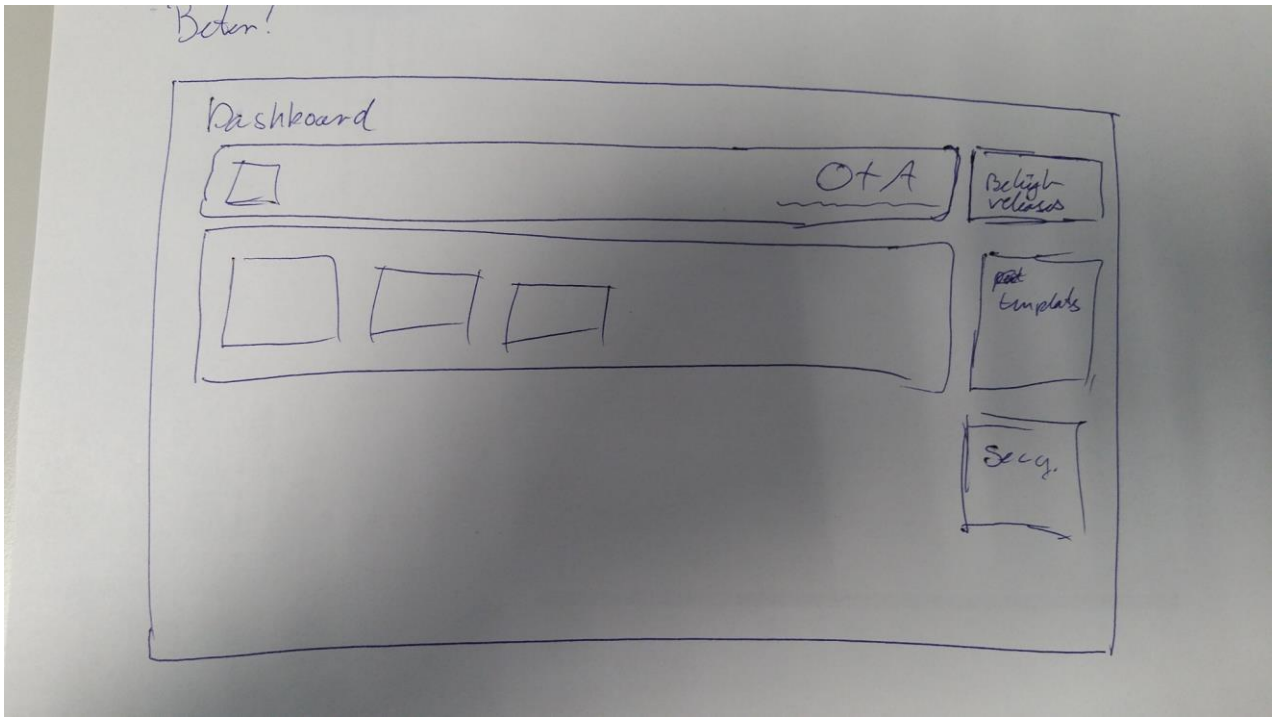
Voor de waarschuwing functionaliteit zijn er geen wijzigingen in de code uitgevoerd. Er zijn alleen grafische wijzigingen plaat gevonden.

4. Wireframes

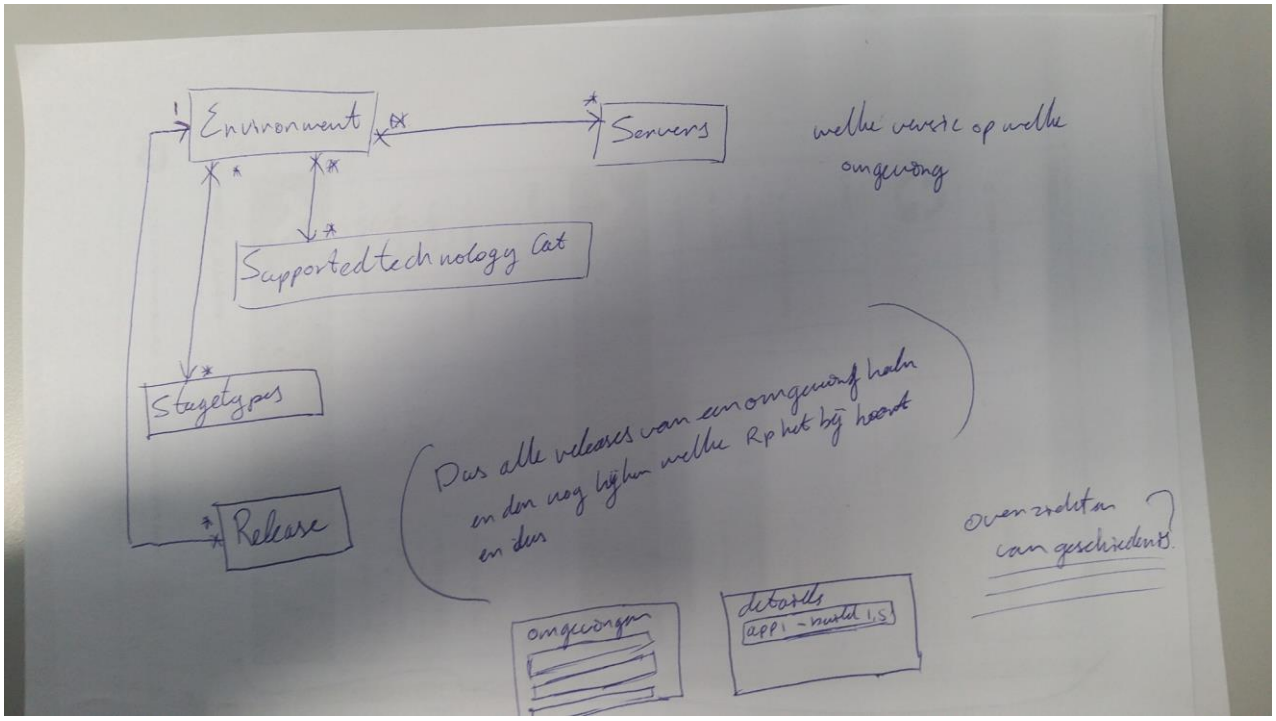
Voor sommige user stories zijn er wireframes gemaakt. Deze wireframes zijn gemaakt voor of tijdens sprint reviews.



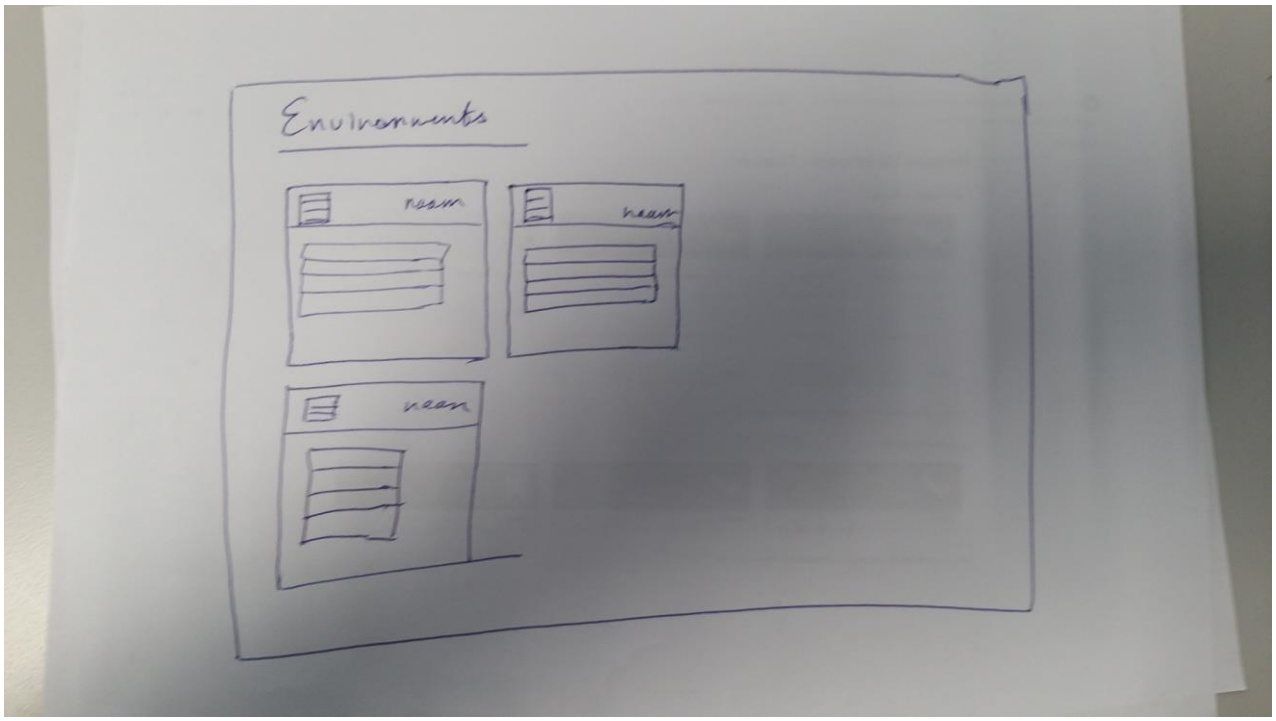
Figuur 10 initieel release path ontwerp



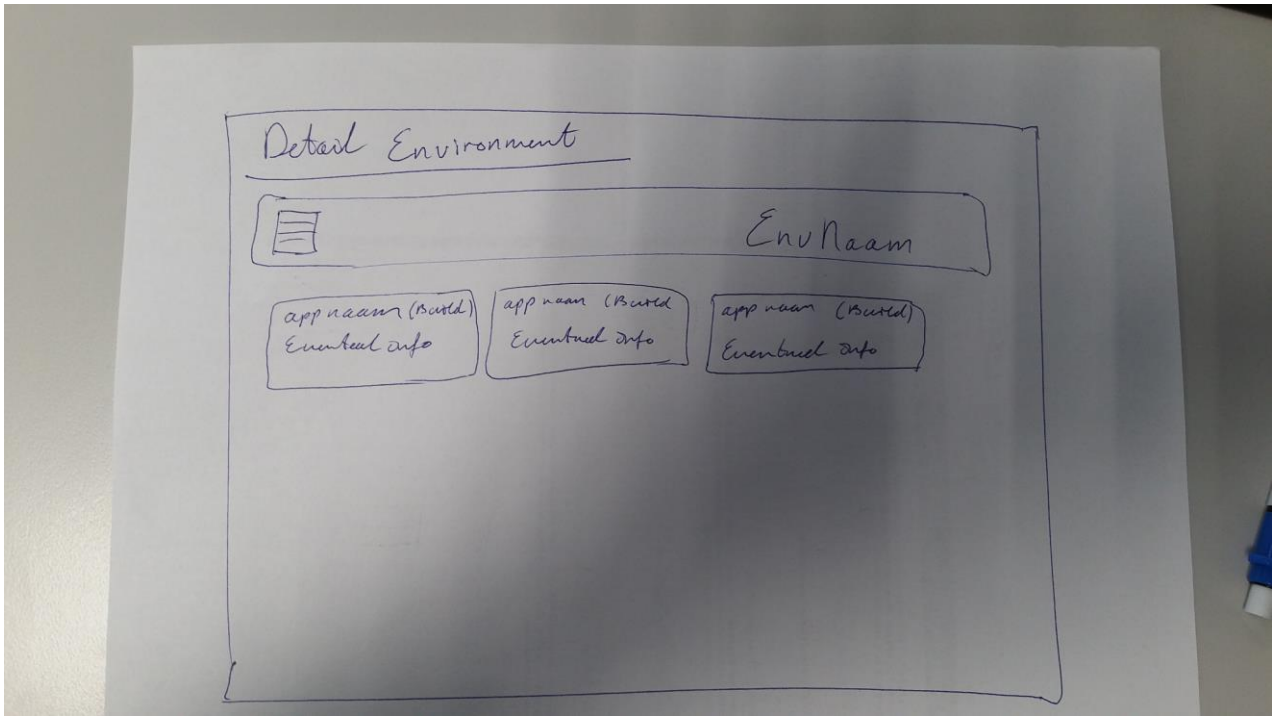
Figuur 11 wijziging in het initieel dashboard implementatie



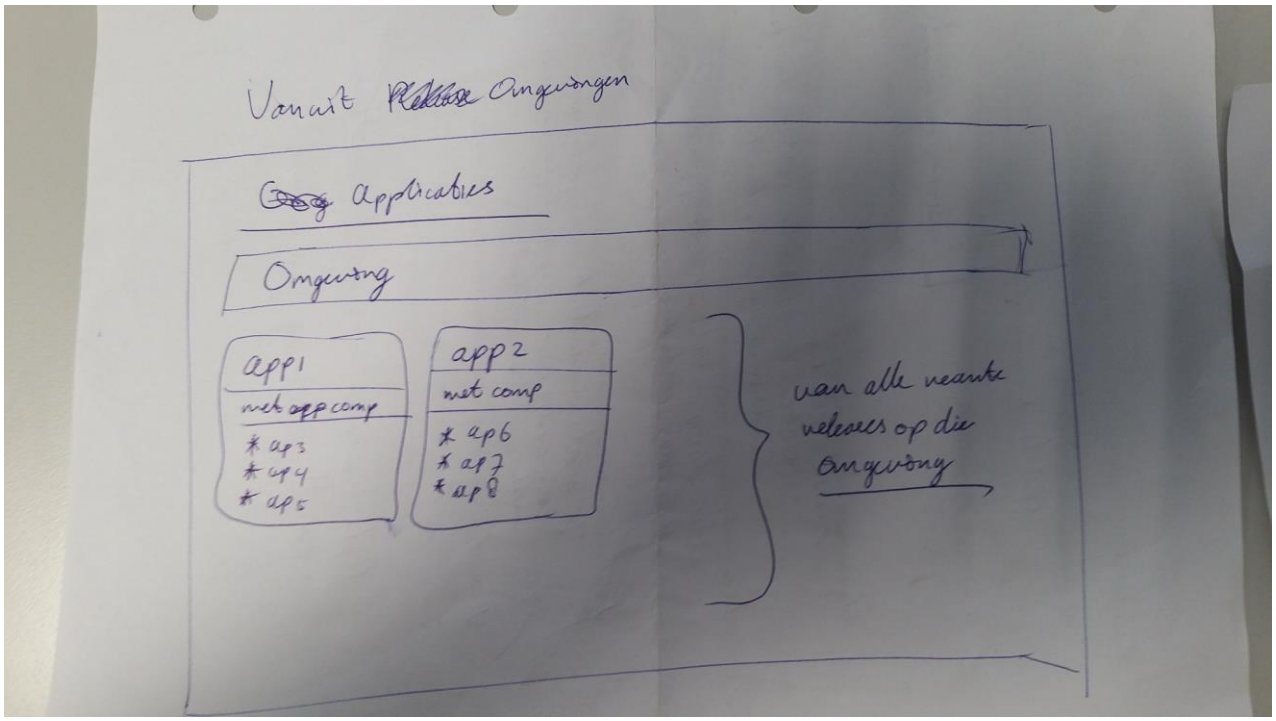
Figuur 12 brainstorm over het tonen van release informatie



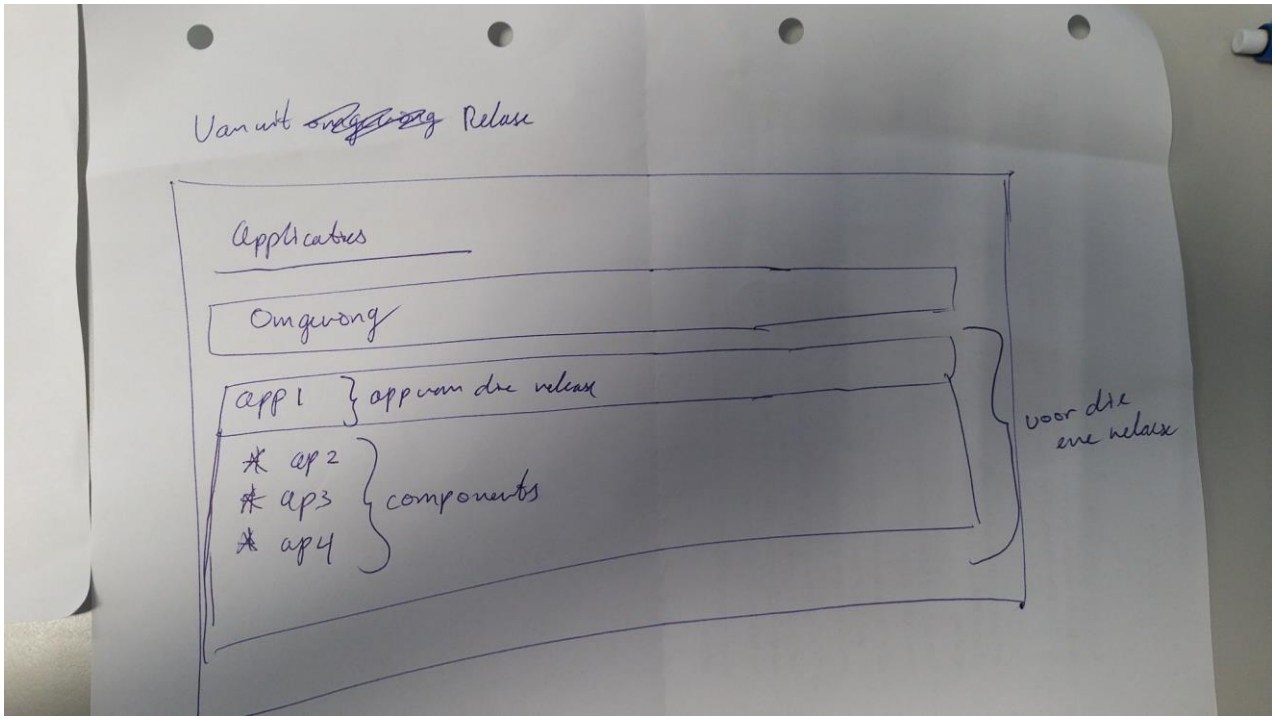
Figuur 13 omgevingen overzicht



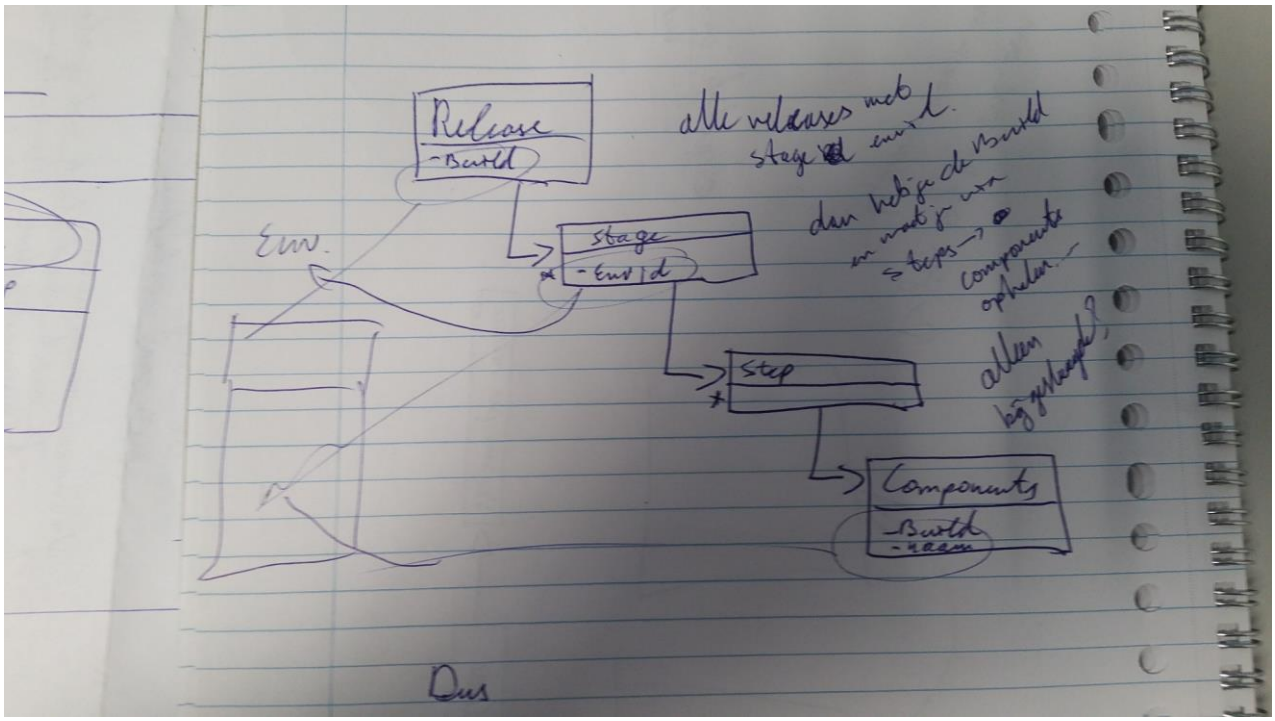
Figuur 14 omgeving detail scherm



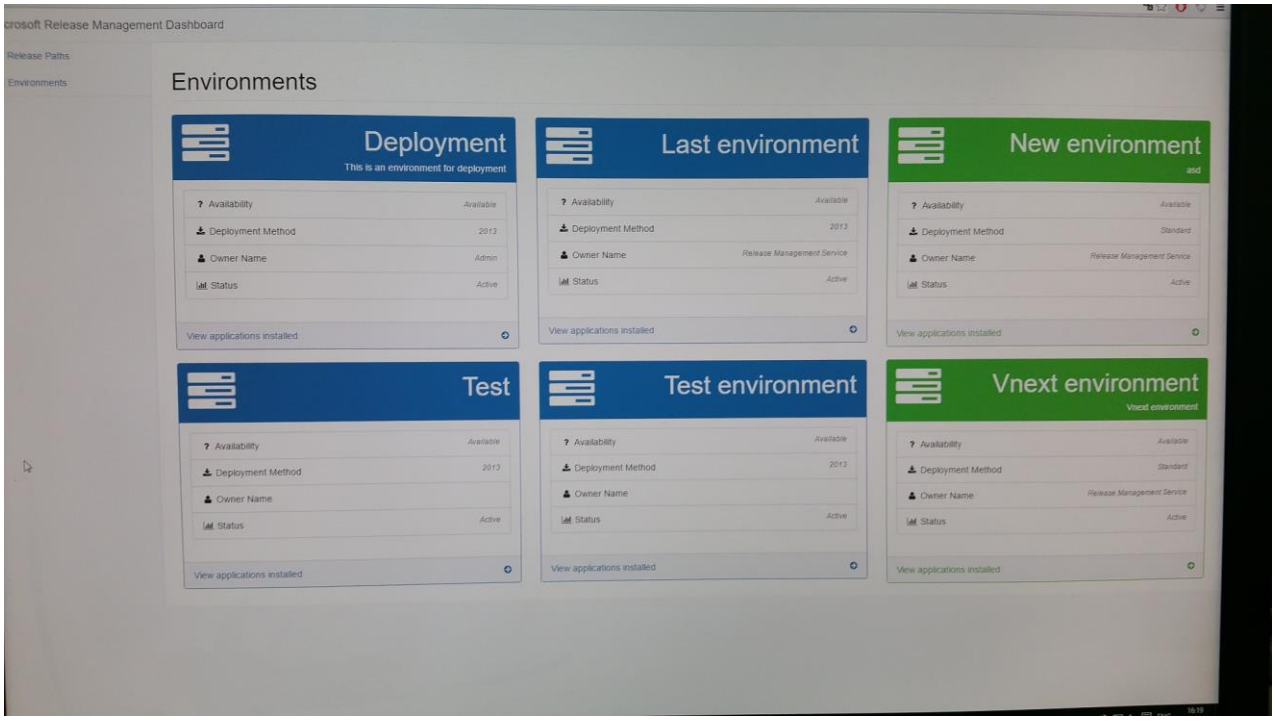
Figuur 15 aanpassing op omgeving detail scherm



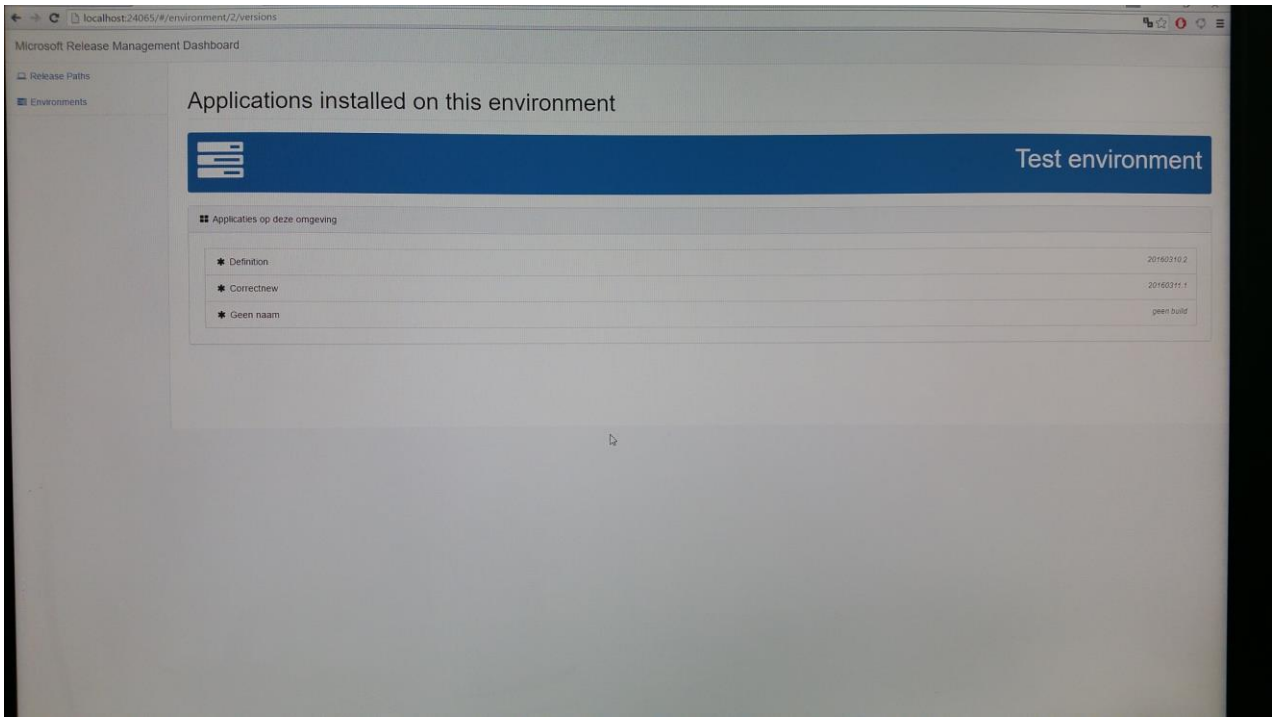
Figuur 16 stage applicatie detail scherm



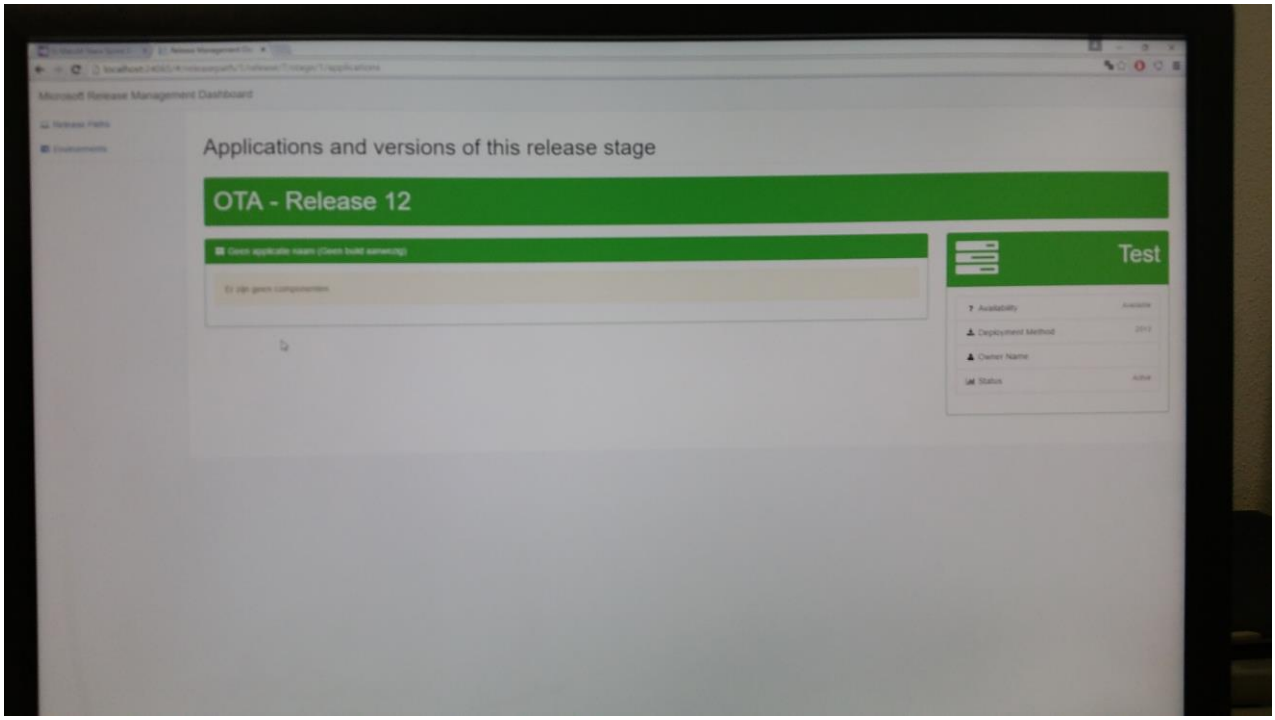
Figuur 17 brainstorm release opbouw



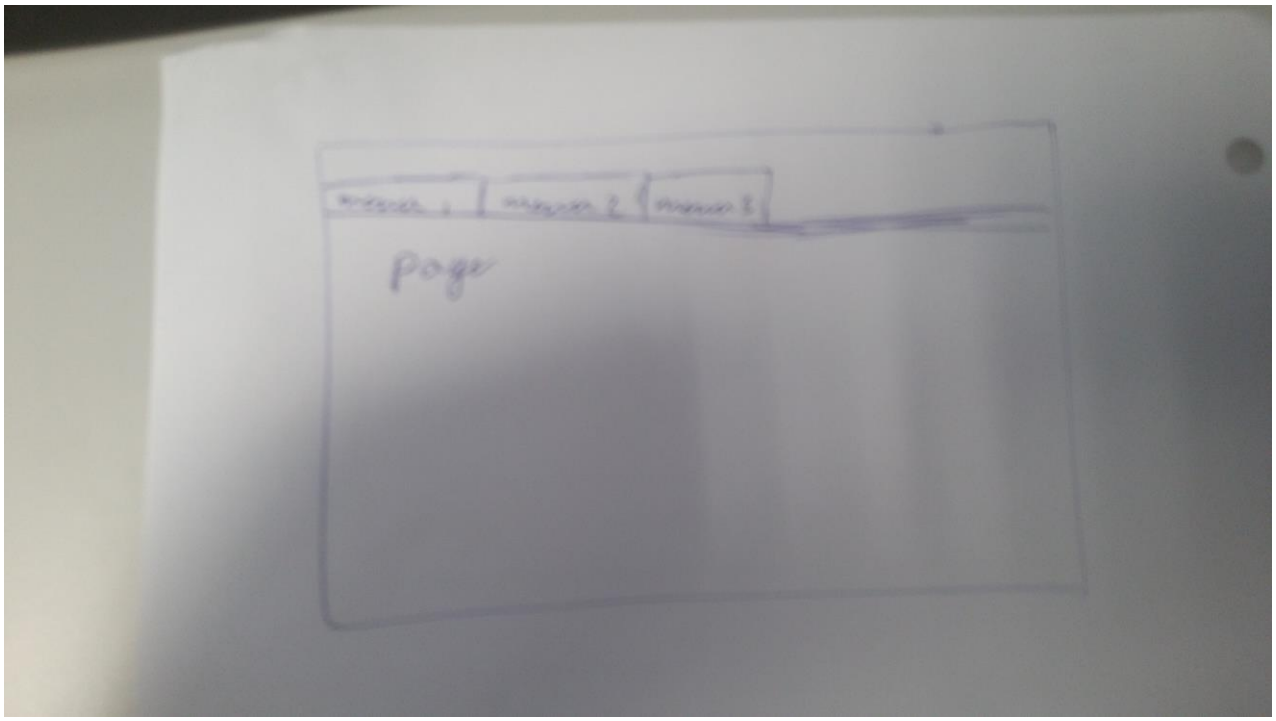
Figuur 18 implementatie ontwerp omgevingen



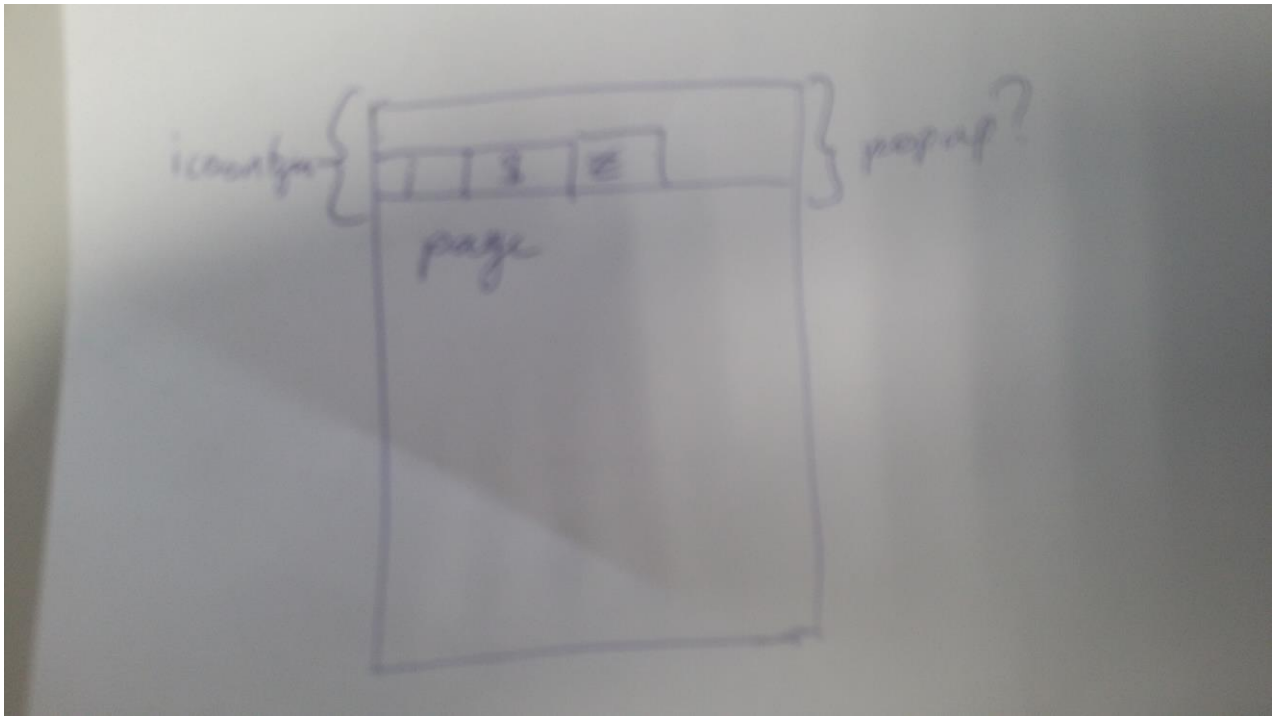
Figuur 19 implementatie omgeving applicaties



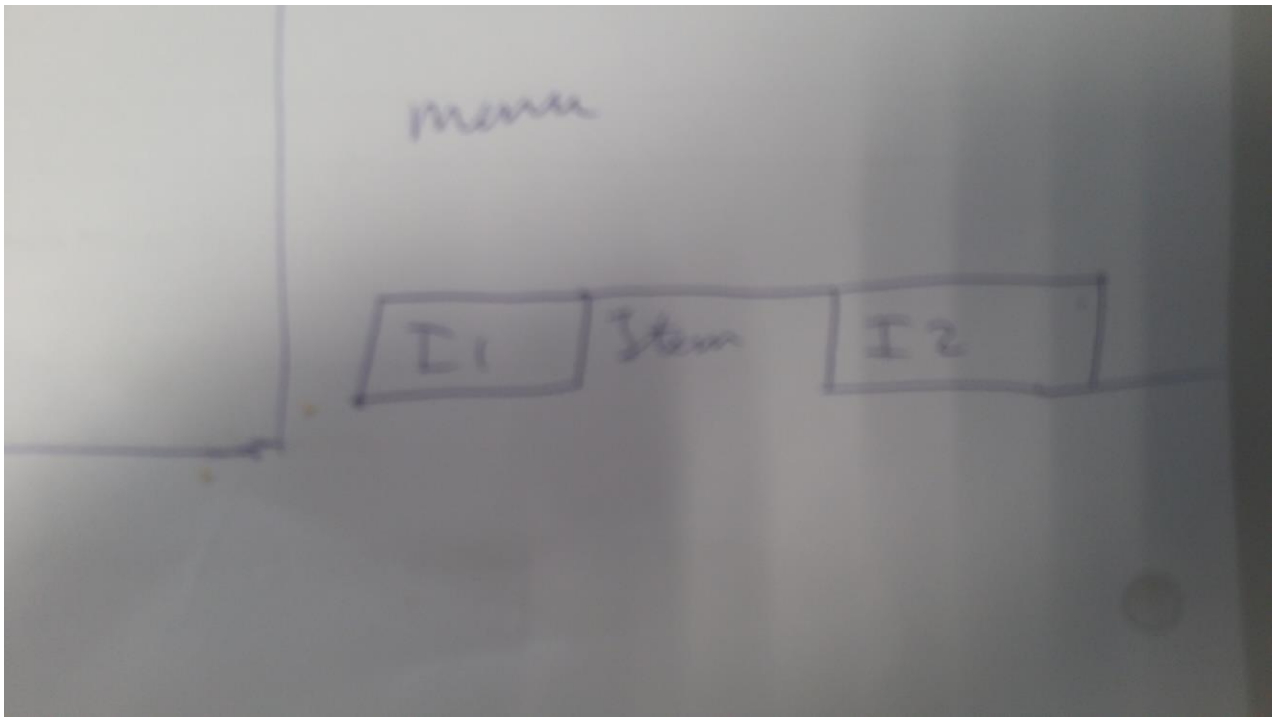
Figuur 20 implementatie informatie over stage



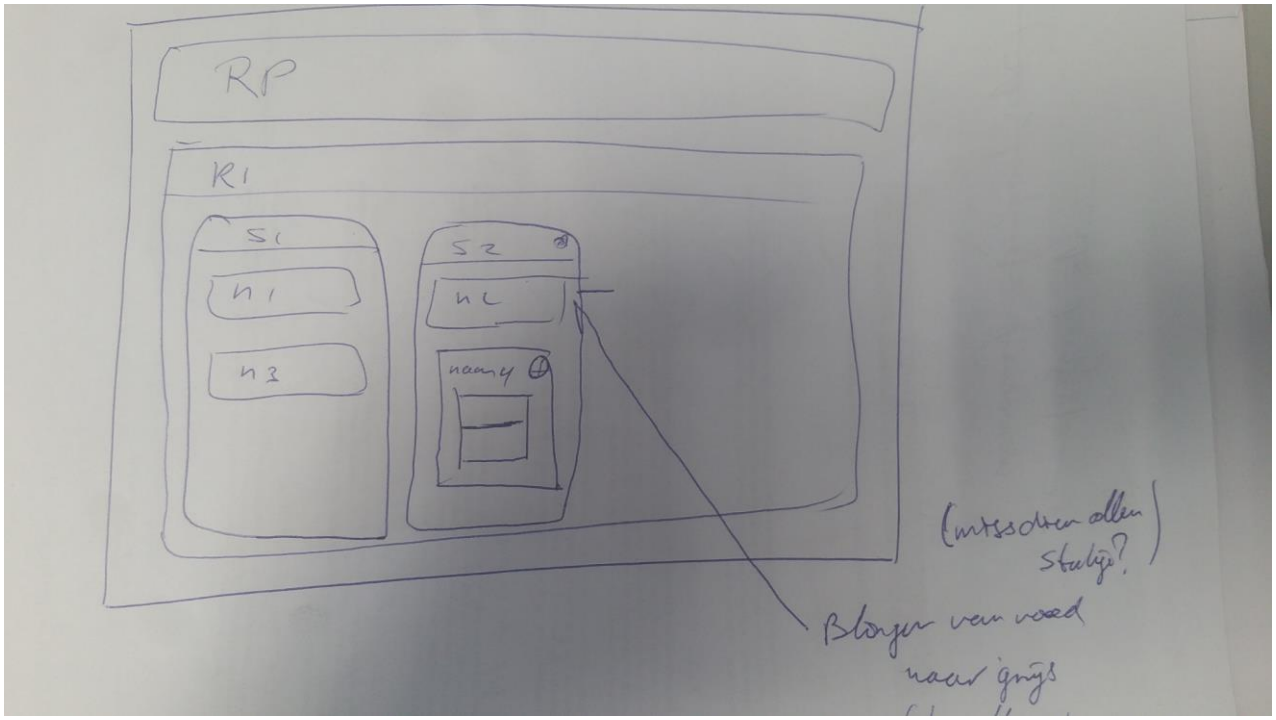
Figuur 21 wijzigen van het menu



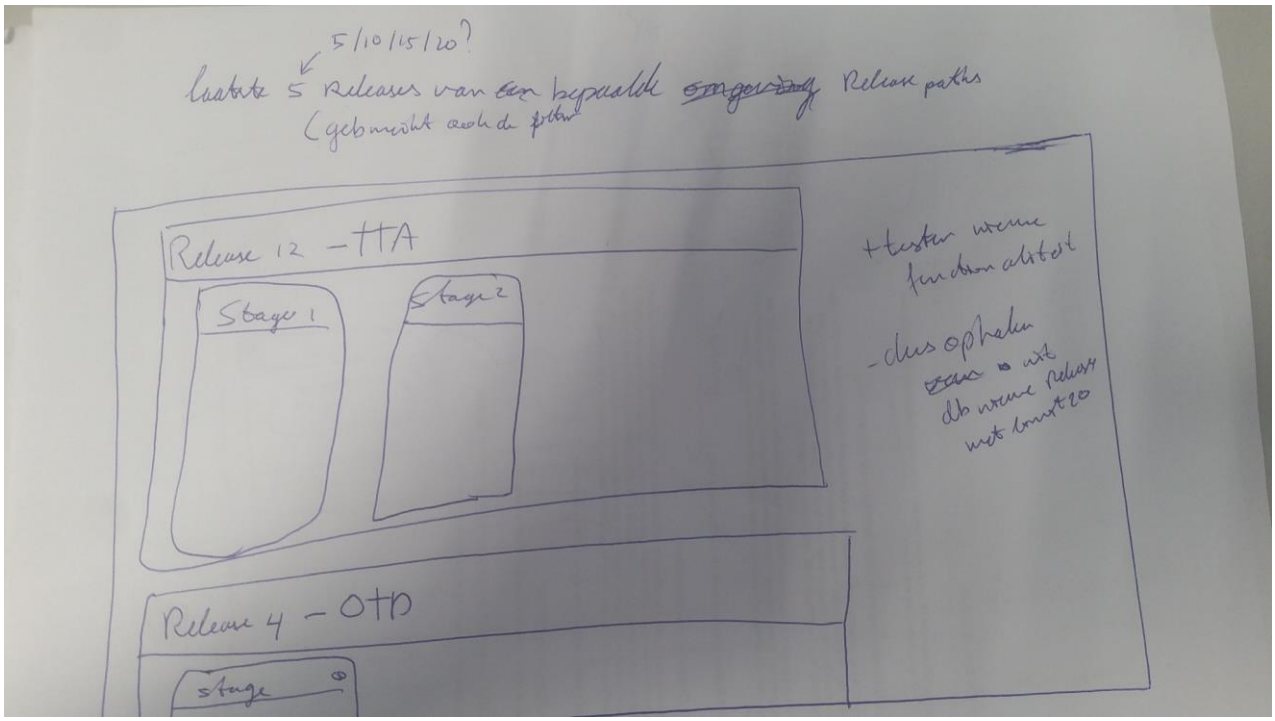
Figuur 22 mobile first ontwerp van het menu



Figuur 23 menu als tabblad ontwerp



Figuur 24 fout icoontje op een stage en een step



Figuur 25 verschillende releases van verschillende release paths

Bijlage IV

Technisch ontwerp

Technisch ontwerp

Technisch ontwerp van de opdracht Visual Deployment Pipeline

Marc De Meza

Inhoud

1. Inleiding

Dit document bevat de technisch ontwerp van de opdracht Visual Deployment Pipeline. Hierin zijn alle gemaakte keuzes tijdens het ontwerpen van het systeem terug te vinden. Verder worden deze keuzes ondersteund door diagrammen. Als laatste is er in de diagrammen ook een overzicht van de klassen in het systeem en hoe deze interactie hebben met elkaar.

Aangezien het systeem uit twee delen bestaat zijn de diagrammen ook zo verdeeld. De delen waaruit dit systeem bestaat zijn AngularJS als front-end en .NET Webapi als backend.

2. Webapi

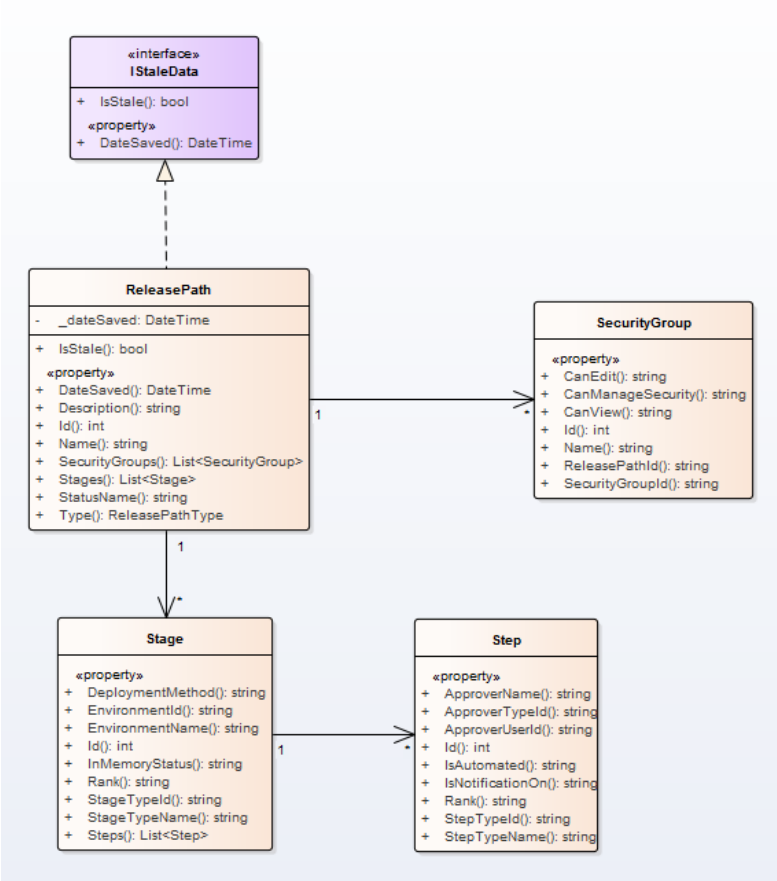
In dit hoofdstuk zijn alle technische ontwerp documenten te vinden over de webapi. Hierin is het mogelijk om te zien welke structuur de applicatie heeft en welke argumentatie de beslissingen hebben. Om te voorkomen dat er teveel informatie in één diagram komt te staan worden de diagrammen verdeeld. Deze verdeling wordt gedaan aan de hand van de sprints.

2.1 Release Paths

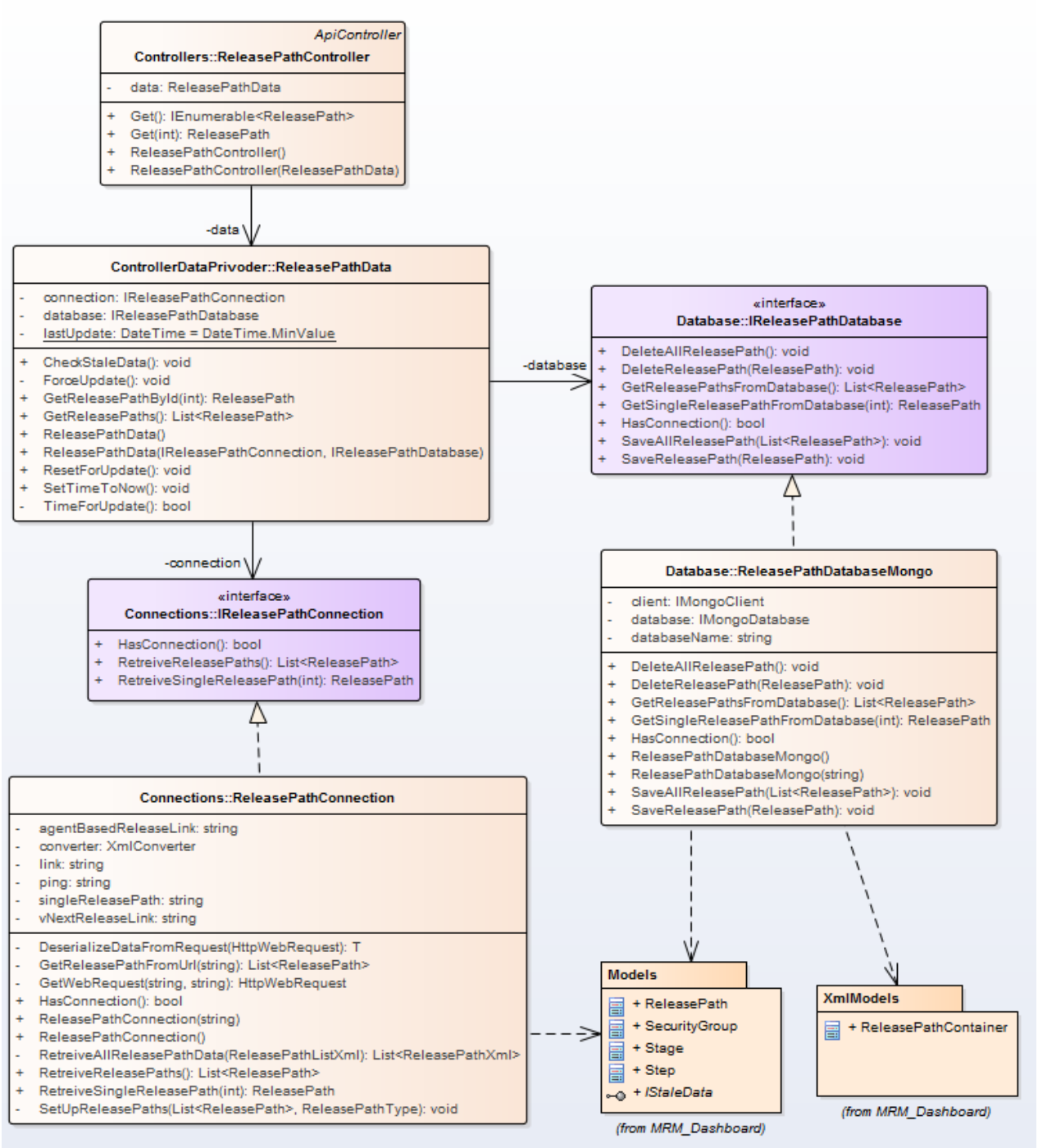
In deze paragraaf worden de diagrammen van de eerste sprint verwerkt. Gedurende de eerste sprint wordt er gewerkt aan het onderdeel ReleasePaths. Voor de eisen en wensen van deze onderdelen kan er verwezen worden naar het functioneel ontwerp.

2.1.1 Klasse diagram webapi

In figuur 1 is de klasse diagram van de entiteiten van de webapi te zien. In figuur 2 is de klasse diagram te zien van de klassen die verbinding maken met de webapi, database en Release Management



Figuur 1 klasse diagram entiteiten



Figuur 2 klasse diagram webapi

Beslissing Interface database en MRM

Tussen de klasse ReleasePathData en de database is er een interface gezet. Met deze interface wordt er een abstractie laag boven de database geplaatst. Nu is het mogelijk de database op elk moment te vervangen met een andere implementatie van een database. Verder zorgt deze er ook voor dat het testen van de klasse ReleasePathData gemakkelijker gaat. Dit is gemakkelijker, omdat het mogelijk is de database volledig te mocken zonder dat er kennis is van de implementatie van de database.

Deze interface is ook gebruikt voor de verbinding met MRM. Het moet in de toekomst ook mogelijk zijn om andere versies van MRM te ondersteunen. Dit betekend dat de databron vervangen zal worden. Met een abstractie boven MRM is het mogelijk dit gemakkelijk te vervangen. De nieuwe databron moet alleen maar de interface implementeren.

Beslissing verantwoordelijkheden
De klasse ReleasePathController moet zo min mogelijk verantwoordelijkheden hebben, aangezien deze klasse alleen gebruikt wordt als eerste communicatiepunt tussen clients en server. De logica van de applicatie bevindt zich in de klasse ReleasePathData.

2.1.2 Klasse diagram xml

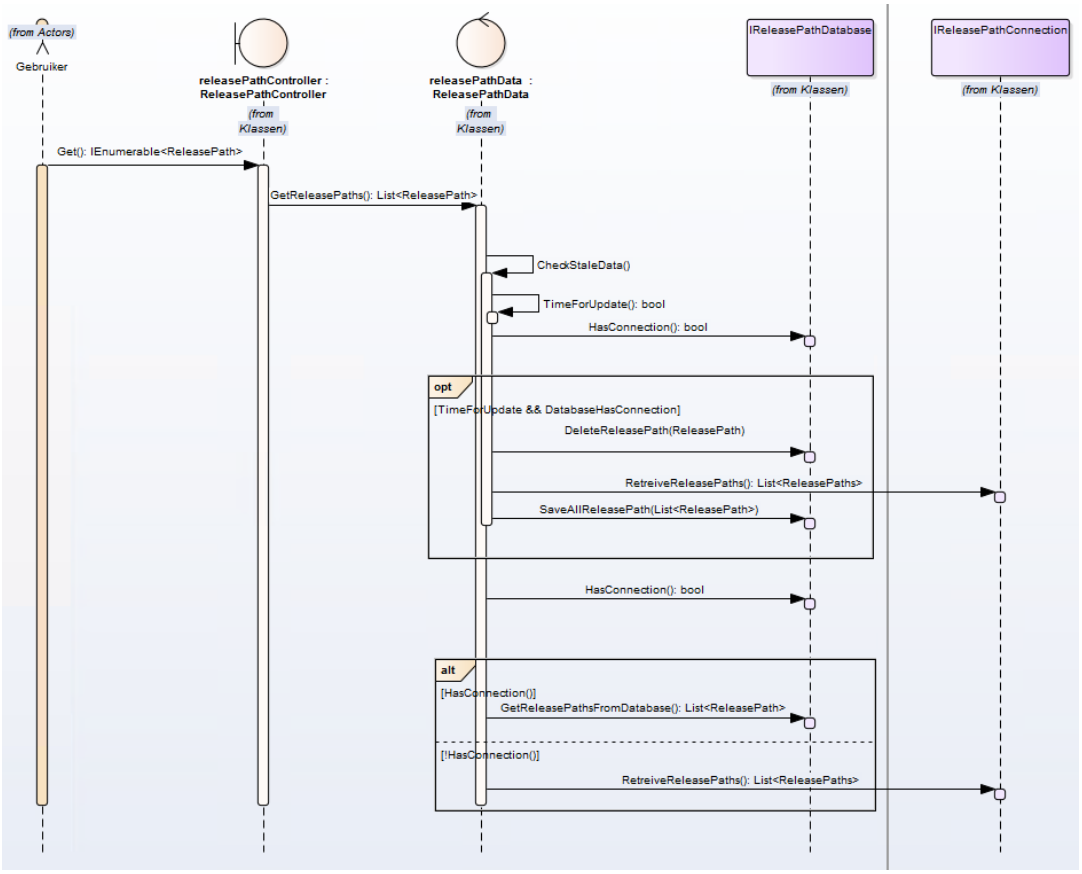
Alle requests die worden uitgelezen van MRM geven een XML response terug. Hierdoor is het nodig om alle gegevens te deserialiseren. Bij sommige klasse is het noodzakelijk om een tussen klasse te gebruiken voor het deserialiseren. In figuur 3 is deze klasse te zien.

Figuur 3 tussen klasse voor ophalen lijst

2.1.3 Sequentie diagrammen

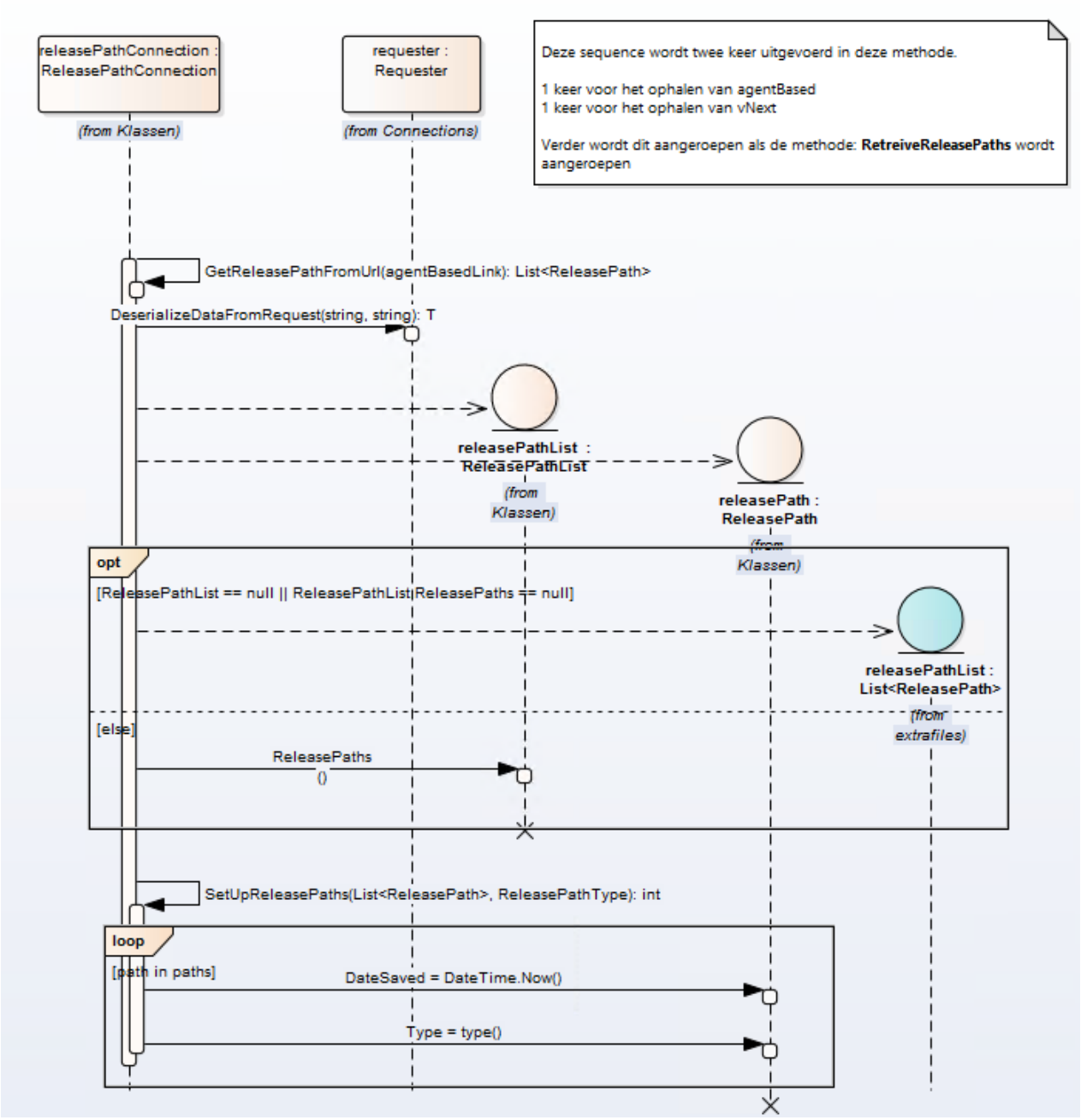
Voor het ophalen van ReleasePaths zijn er een aantal sequentie diagrammen gemaakt. Deze diagrammen worden verdeeld in het ophalen van alle ReleasePaths en van één ReleasePath. Verder communiceren de sequentie diagrammen ook met interfaces. De implementatie van de interfaces worden in andere sequentie diagrammen weergegeven.

ReleasePath GetAll



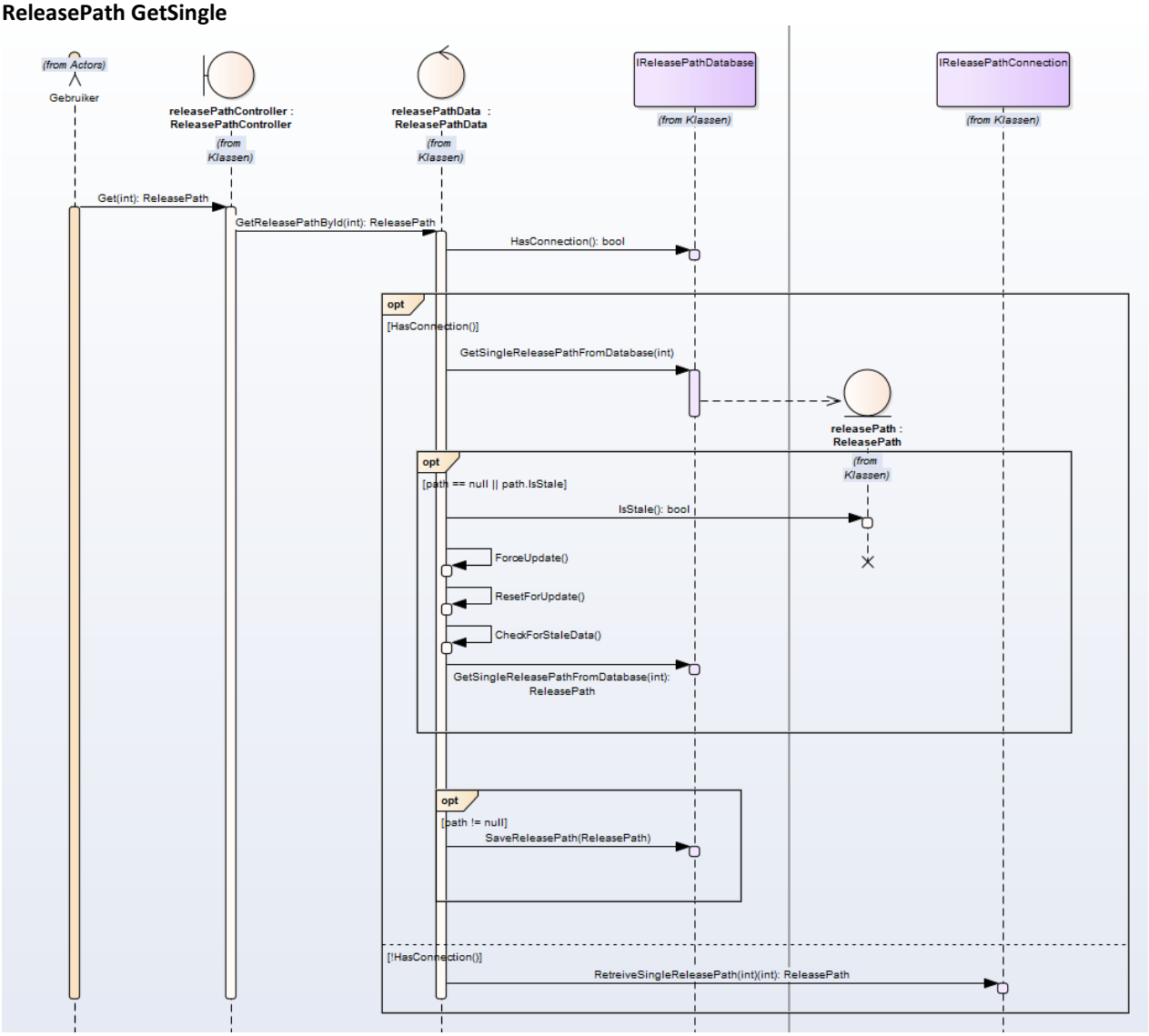
Figuur 4 sequentie voor het ophalen van alle ReleasePaths

ReleasePath GetAll Release Management implementatie



Figuur 5 sequentie voor de implementatie van GetAll

ReleasePath GetSingle



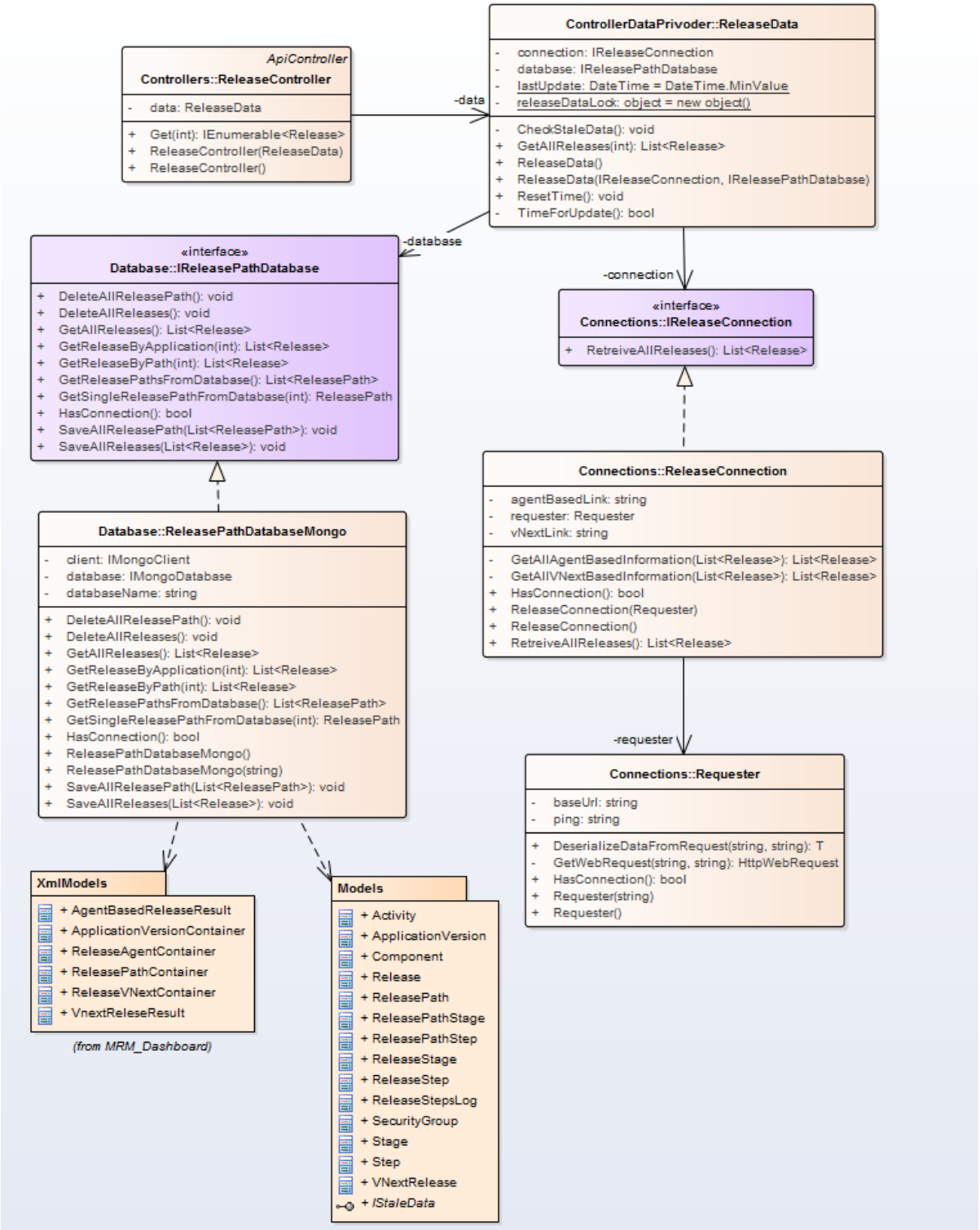
Figuur 6 sequentie diagram voor het ophalen van één ReleasePath

2.2 Releases

In deze paragraaf worden alle diagrammen van de tweede sprint verwerkt. Gedurende deze sprint staan de releases centraal.

2.2.1 Klasse diagram webapi

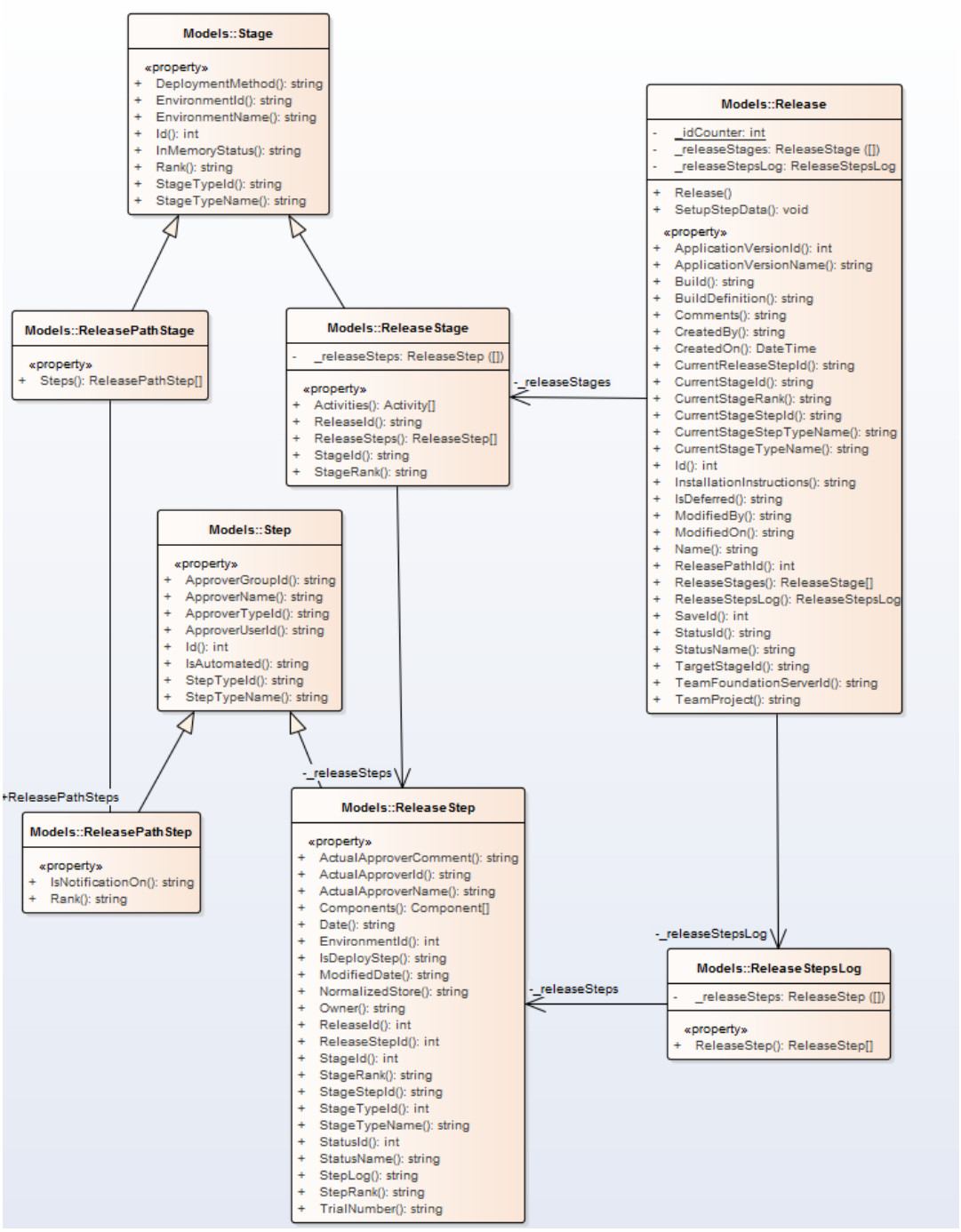
In dit diagram is te zien hoe de klassen zijn opgebouwd voor de webapi functionaliteit. In dit diagram is er ook te zien hoe er weer interfaces zitten tussen de verbinding met release management en de database. Dit is om te voorkomen dat de applicatie te sterk afhankelijk wordt van Release Management. Als laatste is er te zien hoe het deserialiseren van de requests ook uit de connectie klasse is gehaald. Het extraheren van deze klasse is om ervoor te zorgen dat er geen dubbele code in de applicatie komt te staan en dat de verantwoordelijkheden van elke klasse ook verbeterd worden.



Figuur 7 klasse diagram webapi sprint 2

2.2.2 Klasse diagram entiteiten

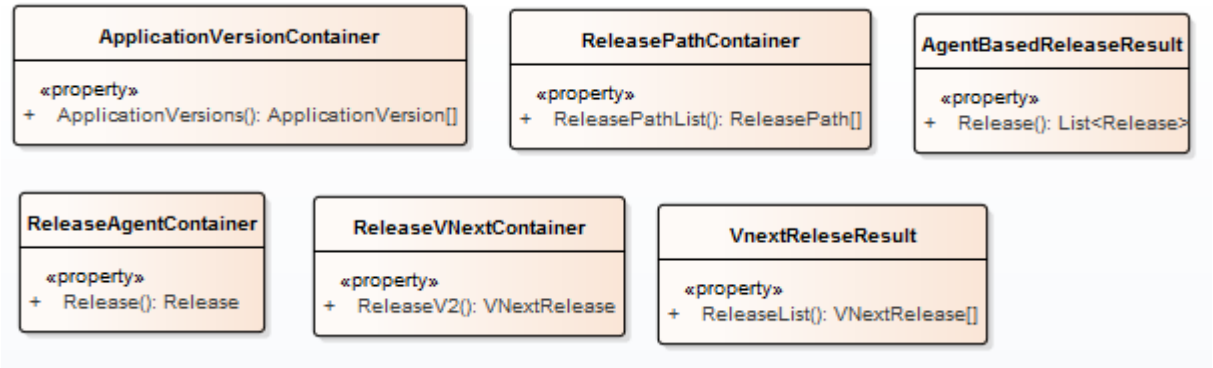
In dit diagram is te zien hoe alle nieuwe entiteiten met elkaar verbonden zijn. Er zijn ook verschillende Stages die uit Release Management afkomstig zijn. Deze Stages hebben een aantal attributen algemeen. Dit is hetzelfde voor de Step klasse. Er is besloten om een superklasse te maken voor Stages en Steps. Hierdoor is er geen dubbele code in de klasse. Het wijzigen van deze structuur heeft geen invloed op het deserialiseren van de requests.



Figuur 8 entiteiten van sprint 2

2.2.3 Klasse diagram xml models

Tijdens het deserialiseren zijn er een aantal klasse die niet direct gedeserialiseerd worden wegens het formaat waarin het XML werd gestuurd. Daarom is het noodzakelijk gebruik te maken van bepaalde tussen klasse. Nadat deze klassen zijn gedeserialiseerd is het mogelijk de lijst eruit te halen en gebruiken in de applicatie.

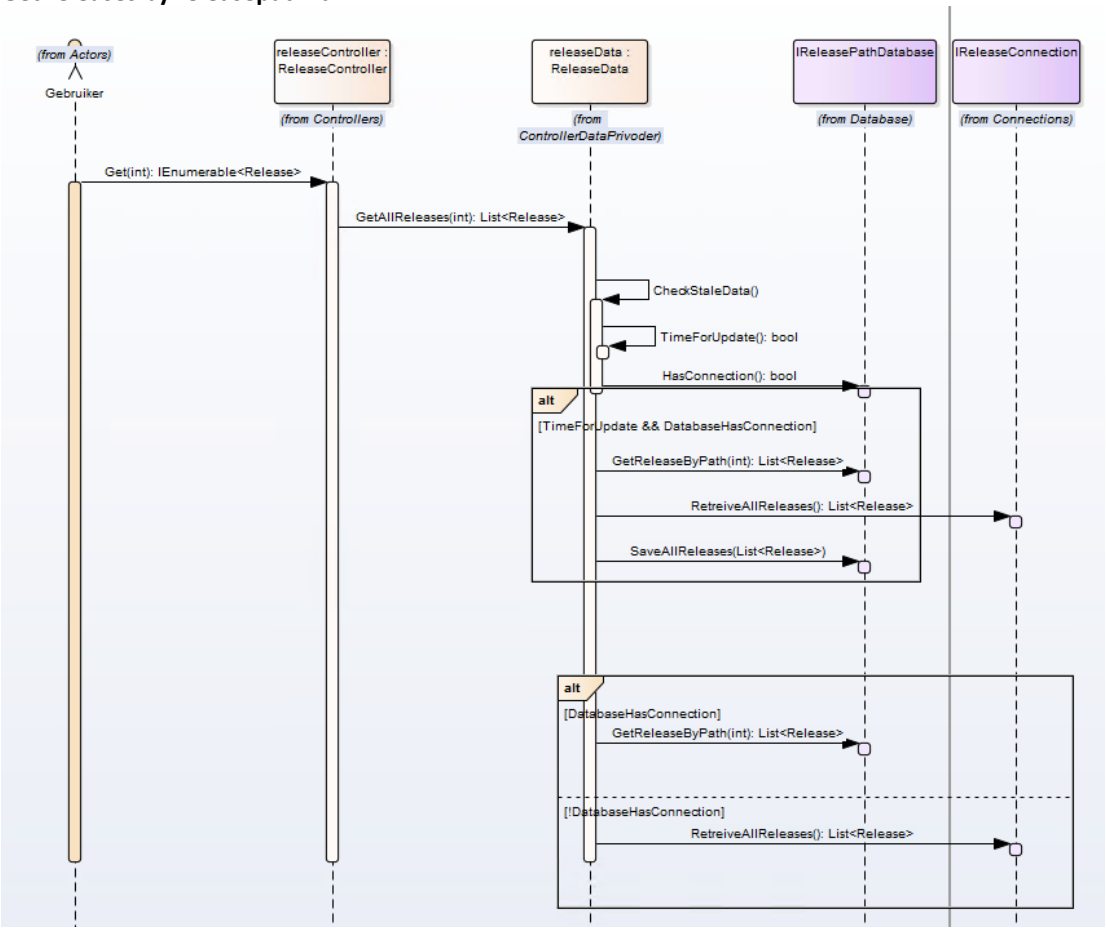


Figuur 9 tussen klassen

2.2.4 Sequentie diagrammen

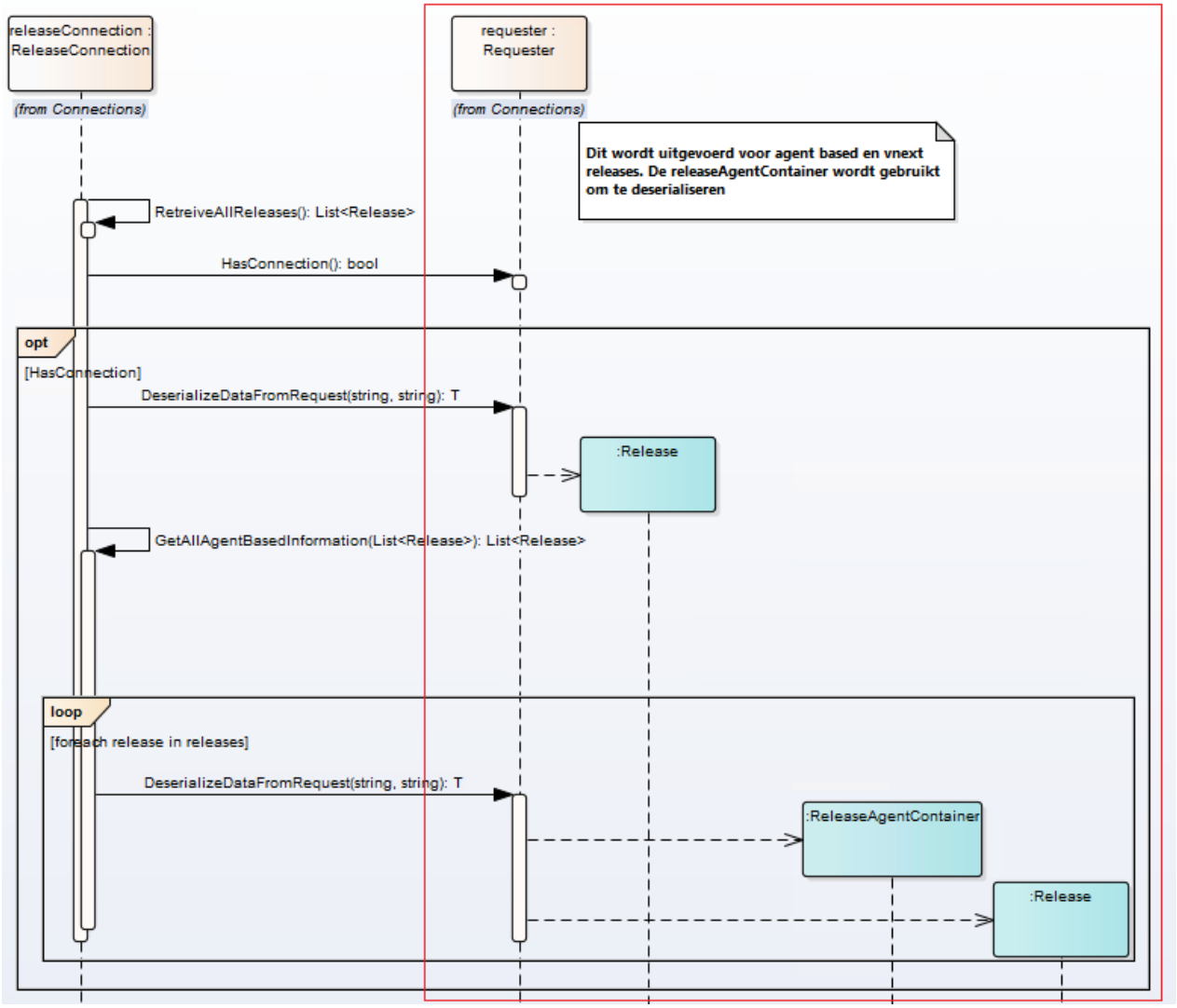
Verder zijn er ook sequentie diagrammen gemaakt. In deze sequentie diagrammen is duidelijk te zien hoe de interactie verloopt tussen de objecten.

Get releases by releasepath id



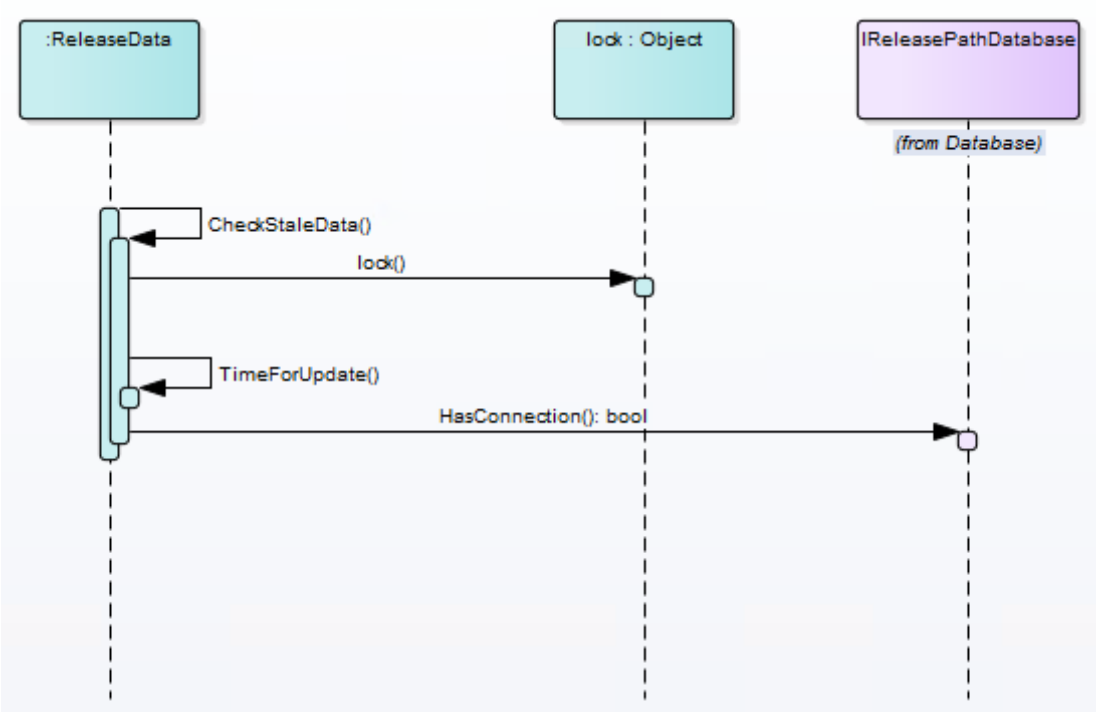
Figuur 10 sequentie voor het ophalen van Releases

Implementatie van IReleaseConnection



Figuur 11 implementatie van IReleaseConnection

Locken gegevens naar de database



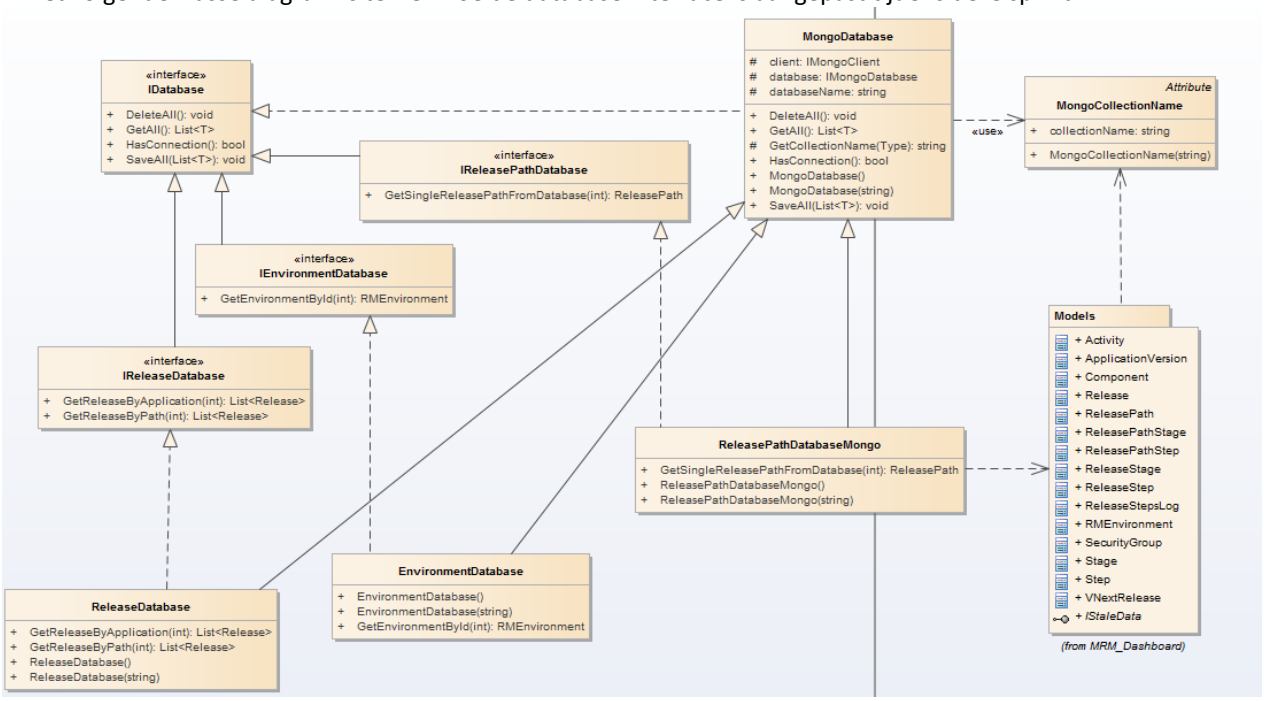
Figuur 12 sequentie diagram van het locken van threads

In de sequentie in figuur 12 is er een klein stukje van een sequentie weergegeven. In dit stukje sequentie wordt weergegeven dat de methode CheckStaleData wordt gelocked voordat er wordt gekeken of er gegevens opgehaald en opgeslagen moeten worden. Als deze lock niet aanwezig is, treden er concurrency problemen op.

2.3 Omgevingen

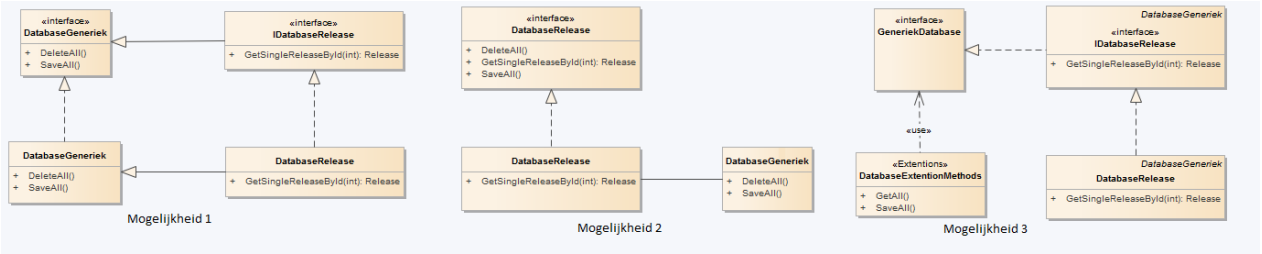
2.3.1 Klasse diagram database structuur

In het volgende klasse diagram is te zien hoe de database interface is aangepast tijdens deze sprint.



Figuur 13 refactorslag database verbinding

De keuze om voor dit ontwerp te gaan is naar aanleiding van drie mogelijkheden die zijn bedacht. Deze mogelijkheden zijn te zien in figuur 14 In mogelijkheid 1 zijn twee interfaces beschreven. Eén interface voor alle generieke methoden en een andere interface voor specifieke methoden. De interface met specifieke methoden erft alles van de interface met generieke methoden over. Hierdoor is het af te dwingen dat een klasse die specifieke methoden uitvoert ook generieke methoden moet uitvoeren. Verder zijn er ook twee klassen in het diagram die elk een interface implementeren. Eén van deze klasse implementeert alle generieke methoden en de andere alle specifieke methoden. Om ervoor te zorgen dat de specifieke klasse ook voldoet aan de overerving van interfaces moet deze klasse alleen nog alles van de database klasse overerven. Dit ontwerp is ook gemakkelijk te mocken na het implementeren, omdat de interfaces met mocks uitgewisseld worden.



Figuur 14 mogelijkheden voor database refactor

Opdelen logica in partials

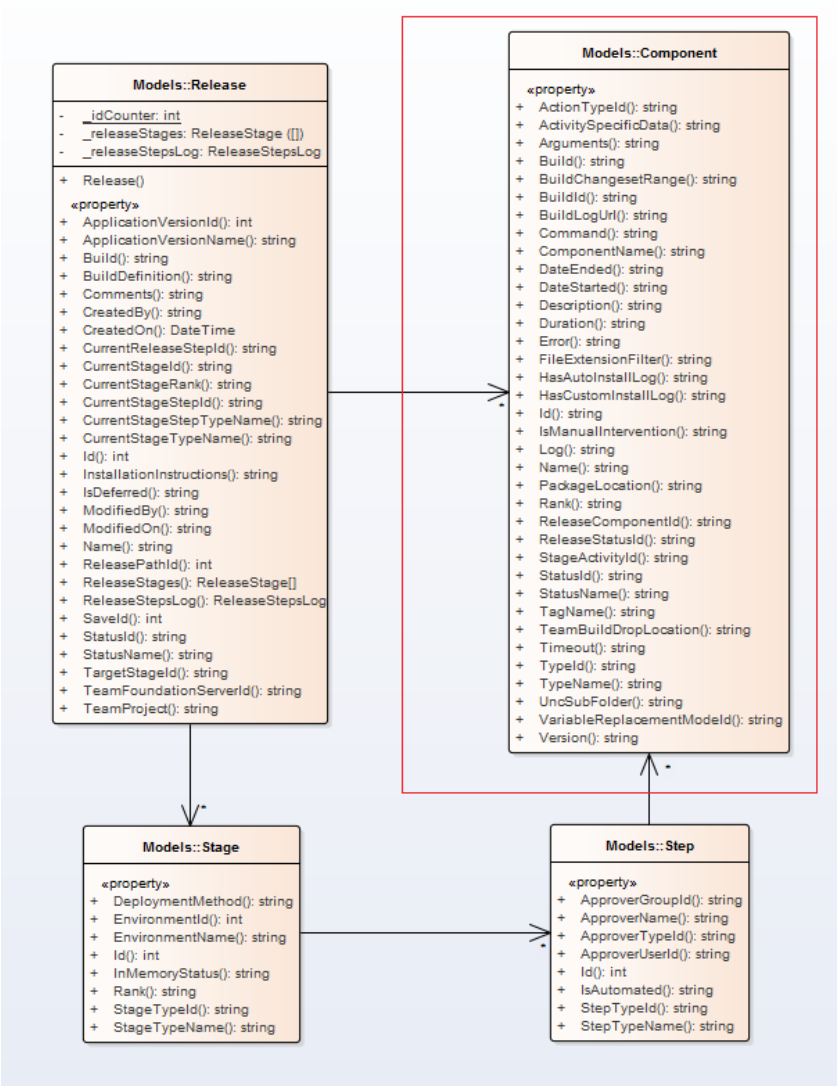
Als laatste is er ook veel logica bijgekomen in de klasse Release. Deze klasse heeft al veel attributen met XML attributen erboven. Hierdoor is er besloten een partial te maken van deze klasse. In één helft van de partial zijn alle methoden geschreven en in de andere helft zijn de attributen. Hierdoor blijft de klasse overzichtelijk.

2.4 Instellingen, componenten en live updates

In dit paragraaf zijn alle beslissingen van de vierde sprint te vinden. Tijdens deze sprint werk ik aan het realiseren van het weergeven van componenten, het uitvoeren van live updates en het wijzigen van instellingen. De instellingen hebben betrekking tot het bepalen wanneer er precies een update plaats vindt of welke ReleasePaths getoond worden.

2.4.1 Nieuwe klasse

Om de componenten bij de Release en ReleaseStep op te halen moet er een kleine aanpassing worden gemaakt aan de models. Deze aanpassing is het toevoegen van één klasse en het goed zetten van de relaties naar deze klasse. Verder wordt het deserialiseren niet aangepast. De AngularJS code hoeft ook niet aangepast te worden. Alleen het weergeven van de details moet worden aangepast. In figuur 15 is te zien welke klassen erbij is gekomen.

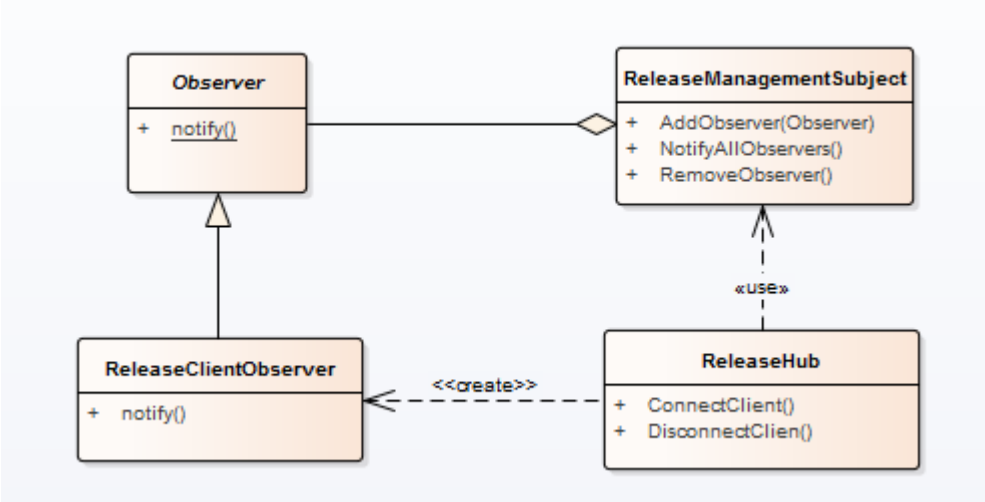


Figuur 15 nieuwe klasse component

2.4.2 Live updates

Als het dashboard alle Releases aan het tonen is en er een nieuwe Release wordt uitgevoerd met MRM moet een gebruiker eerst het dashboard verversen voordat alle nieuwe gegevens zichtbaar zijn. Dit verlaagt de gebruikersvriendelijkheid, omdat iemand voortdurend moet controleren of het scherm is ververst. Het is mogelijk om het scherm zelf verantwoordelijk te maken voor het verversen van zijn gegevens. Voor dit probleem zijn er twee oplossingen bedacht. De eerste oplossing was het gebruik maken van websocket en de tweede oplossing was het gebruiken van long polling.

Oplossing 1 websocket
Websockets is een protocol dat gebruikt kan worden voor communicatie in twee richtingen over een TCP verbinding. Moderne web browsers implementeren deze protocol en maken het dus mogelijk voor servers en clients om direct met elkaar te communiceren. Het .Net framework heeft een package dat gebruikt kan worden voor het snel realiseren van een verbinding met websockets. Deze package heet SignalR. Met SignalR is het mogelijk om een centrale punt te maken waar alle clients naar toe kunnen verbinden. Deze clients kunnen met elkaar en de server communiceren zonder dat er requests op de server worden uitgevoerd. Dit kan gebruikt worden om live updates te realiseren door de server naar de clients een bericht te laten sturen zodra er nieuwe wijzigingen zijn in release management. In figuur .. is een implementatie te zien van websockets met SignalR.



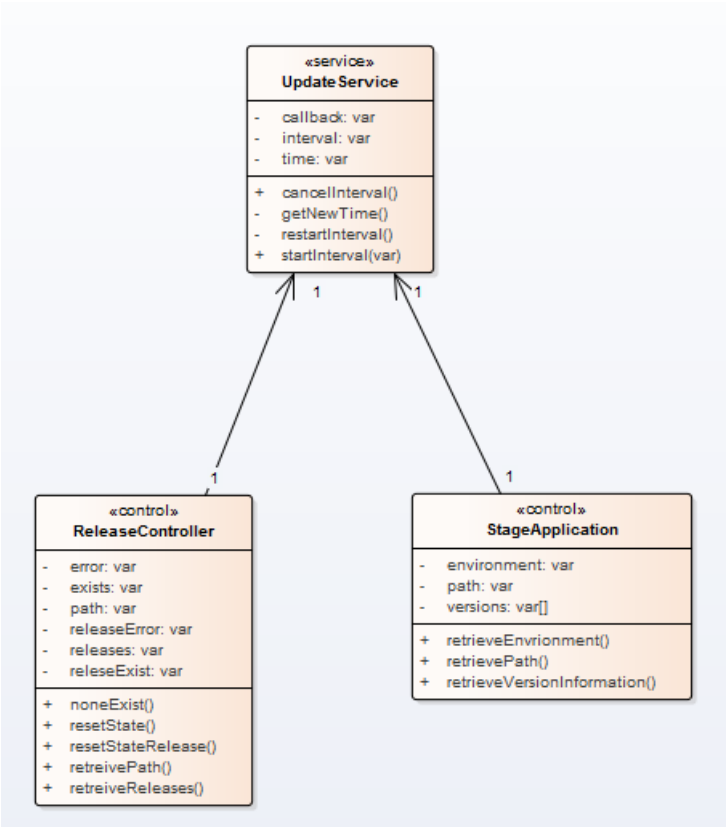
Figuur 16 klasse diagram van SignalR implementatie

In dit ontwerp is de ReleaseHub verantwoordelijk voor het afhandelen van alle connections en disconnections. Dit betekend dat de ReleaseHub ook de observers moet aanmaken. Nadat een Observer is aangemaakt kan deze door de ReleaseHub worden toegevoegd of verwijderd uit de ReleaseManagementSubject. De ReleaseManagementSubject is verantwoordelijk voor het afluisteren naar MRM. Bij wijzigingen in MRM wordt dit doorgegeven aan alle Observers. De nadeel van deze oplossing is dat het de complexiteit van de applicatie sterk verhoogt.

Oplossing 2 long polling
Long polling is een term dat gebruikt wordt voor het beschrijven van een techniek waar een website met gebruik van Javascript requests doet naar een server. Met deze requests ververst de website de data die wordt getoond aan de gebruiker. Officieel gezien is dit geen vorm van een push notificatie, maar het effect lijkt er wel op. Met long

polling is het belangrijk om te weten dat alle requests asynchroon worden uitgevoerd. De gebruiker merkt dus niet aan de website dat het bezig is met nieuwe gegevens inladen. Dit is goed voor het dashboard, omdat een gebruiker nog verder informatie wilt bekijken terwijl het dashboard zijn data ververst.

Long polling kan op verschillende wijze geïmplementeerd worden in het dashboard. Niet alle schermen moeten een live update krijgen. Het is noodzakelijk dat de schermen waar alle Releases op staan en de schermen waar alle versies van een bepaalde Stage live updates krijgen. Dit betekend dat de methode die gebruikt wordt om deze gegevens in te laten met een bepaalde interval aangeroepen moet worden. In figuur 17 is te zien hoe dit geïmplementeerd kan worden.



Figuur 17 implementatie van long polling AngularJS

In figuur 15 is te zien hoe een ReleaseController en een StageApplication klasse gebruik kunnen maken van de UpdateService. Bij het inladen van de controller wordt de methode startInterval aangeroepen met als parameter de callback methode die aangeroepen moet worden. De UpdateService gaat vervolgens zelf de interval ophalen. Hierna wordt de callback bij elke interval aangeroepen. Voordat er wordt genavigeerd naar een andere controller wordt de methode cancelInterval aangeroepen. De code van het starten van de timer ziet er als volgt uit:

```
var startInterval = function (c) {
	getNewTime();
	callback = c;
	restartInterval();
}

var restartInterval = function () {
	interval = $interval(function () {
		callback();
	}, time);
}
```

Het aanroepen van de cancel wordt als volgt gedaan:

```
$scope.$on('$destroy', function () {
   updateservice.cancel();
});
```

Resultaat

Deze beslissing is voorgelegd aan de opdrachtgever. De opdrachtgever heeft besloten om te kiezen voor de tweede oplossing. Deze beslissing is genomen omdat het niet nodig is om de applicatie complexer te maken dan het al is. Verder is het niet erg dat er vaak requests worden gedaan naar de server.

2.4.3 Instellingen opslaan

Wanneer iemand het dashboard gebruikt wilt hij niet alle gegevens te zien krijgen die zijn buurman ook ziet. Hij wilt alleen informatie zien waar hij bij betrokken is. Daarom is het nodig dat de instellingen op de client worden opgeslagen. Om gegevens op een client op te slaan zijn er twee mogelijkheden. Deze mogelijkheden bestaan uit het opslaan in cookies of in localstorage. In het volgende tabel is er informatie te zien over cookies en localstorage.

Overweging	Localstorage	Cookie
Doel	Gegevens opslaan op de client en door de client te laten gebruiken	Gegevens opslaan op de client en uitwisselen tussen client en server
Elke request verstuurd	Nee	Ja
Vervaltijd	Geen	Aangeven
Opslag	5MB	4069B

De laatste twee punten, namelijk vervaltijd en opslag zijn niet doorslaggevend voor het kiezen van een opslag methode. Het aangeven van een vervaltijd op een cookie is niet moeilijk en indien het nodig is om meer dan 4069B op te slaan kunnen er meerdere cookies worden opgeslagen.

Bij deze beslissing gaat het voornamelijk om de functionaliteit dat gerealiseerd wordt. Ten eerste willen we een interval kunnen kiezen die de client kan gebruiken om te bepalen wanneer het een verzoek naar de server moet doen. Dit betekend dat deze gegevens nooit naar de server verstuurd moeten worden. De overige instelling is het bepalen welke ReleasePath je te zien krijgt in het ReleasePath overzicht. In AngularJS is het mogelijk om alle gegevens van ReleasePaths in te laden en deze vervolgens te filtreren aan de hand van gegevens in de instellingen. Een AngularJS filter is een klasse dat je data kan filtreren aan de hand van logica die je zelf schrijft. Hierdoor is het ook niet nodig om deze instellingen te versturen naar de server.

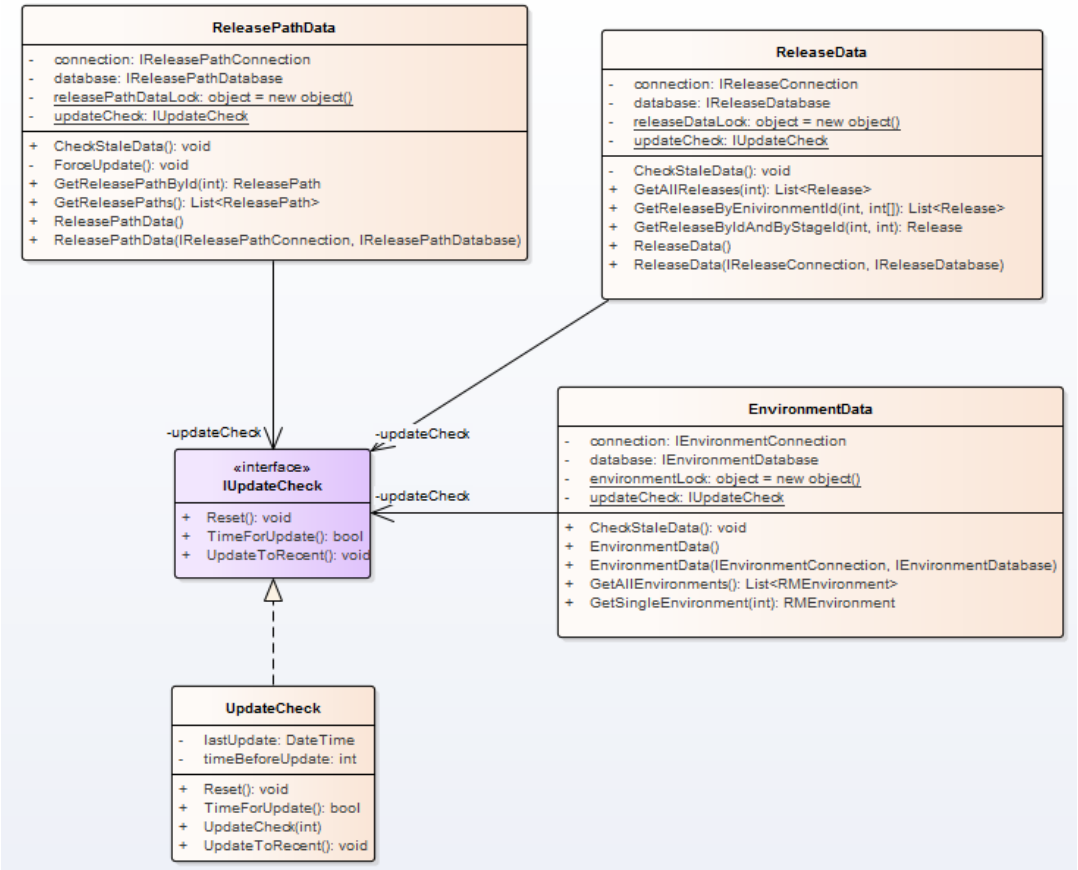
De enige functionaliteit waar een instelling naar de server moet gaan is het tonen van alle applicaties en versies op een omgeving. Om deze overzichten te tonen is het nodig dat er op de server alle ReleasePaths gesorteerd worden voordat er bepaald wordt welke applicaties zijn geïnstalleerd. Aangezien dit de enige keer is dat er een instelling naar de server moet gaan is er gekozen om gebruik te maken van localstorage. Cookies zullen alleen maar onnodig informatie tussen de client en server versturen.

2.4.4 Refactor stale data

In de voorgaande sprints was er geen aandacht besteedt aan het verversen van gegevens in de cache. In deze sprint staat dat onderwerp centraal. Daarom wordt dit stukje in de code gerefactored. Momenteel zijn er drie klasse in de applicatie die de cache met een interval verversen. Om ervoor te zorgen dat er geen dubbele code ontstaat is er een interface gemaakt met de methoden die nodig zijn voor het updaten. Vervolgens is er een implementatie gemaakt die met een interval aangeeft wanneer er een update uitgevoerd moet worden.

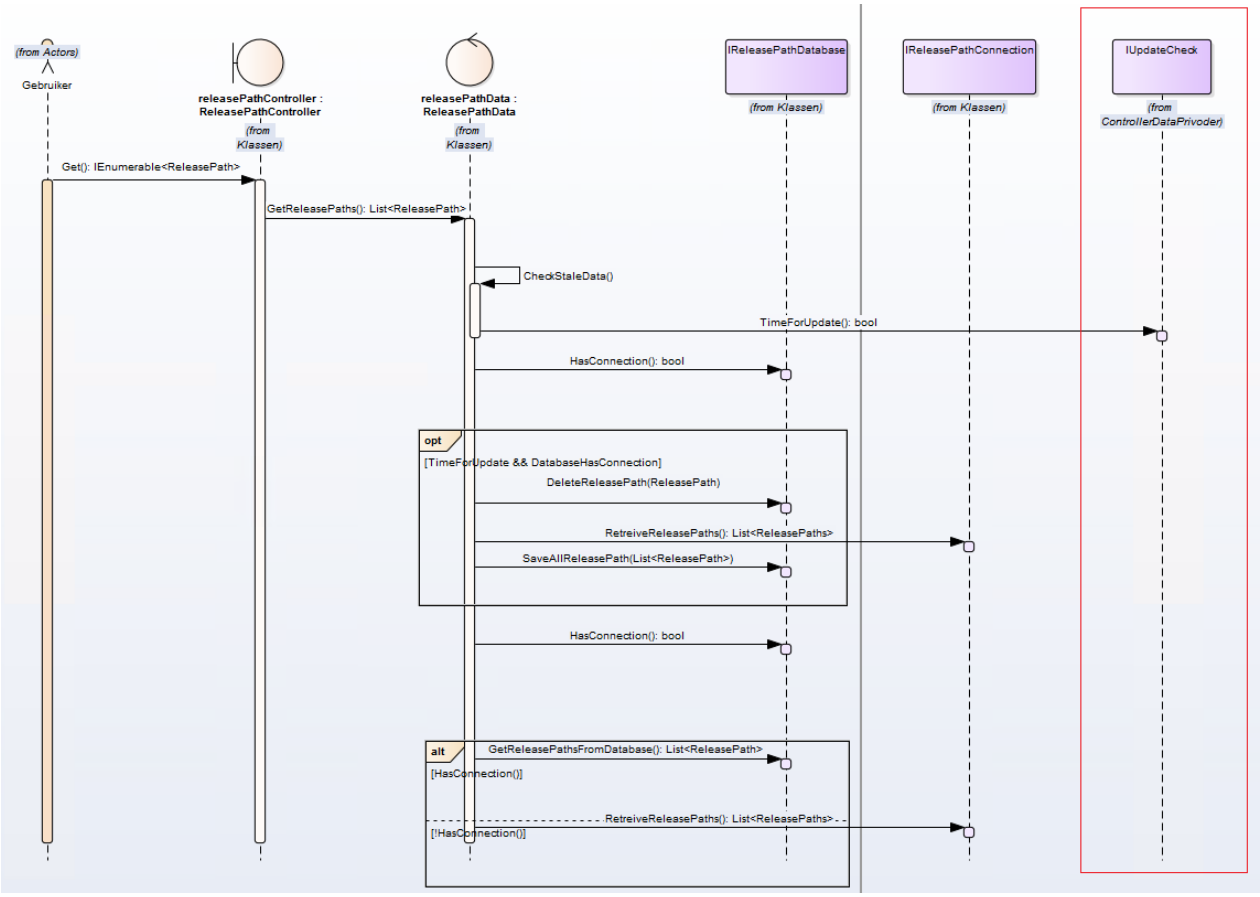
Er is besloten om een interface te maken, zodat er op elk moment een andere manier van updaten gerealiseerd kan worden. Als MRM in de toekomst events kan versturen bij een update is het mogelijk om hier een andere implementatie van de interface voor te maken.

In figuur 18 is te zien hoe dit is geïmplementeerd. In de klasse die de interface gebruiken is te zien dat deze statisch wordt bijgehouden. Bij elke request naar de server maakt webapi een nieuwe instantie aan van de controller. Dit betekend dat er ook nieuwe instanties van de data klassen gemaakt wordt. Als er nieuwe instanties van de IUpdateCheck gemaakt worden betekend het dat de cache elke keer opnieuw ververst wordt. Met een statische referentie betekend het dat dit object altijd hetzelfde zal zijn voor alle instanties van die klasse.



Figuur 18 nieuwe interface voor het updaten

In figuur 19 is te zien hoe dit de sequentie diagram beïnvloed. De nieuwe interface is aangegeven in het rood. In plaats van alle logica intern aan te roepen kan de klasse ReleasePathData deze interface aanroepen.



Figuur 19 aanpassing aan de sequentie van updaten

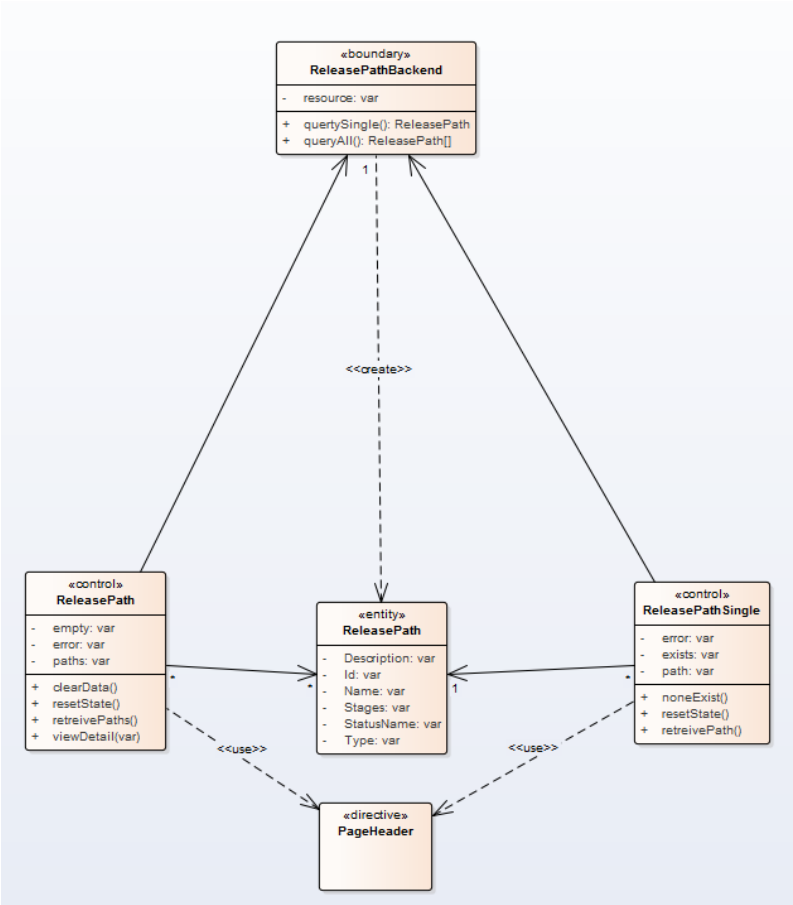
3. AngularJS

In dit hoofdstuk worden de technische ontwerpen van de AngularJS applicatie weergegeven. Per sprint zijn er diagrammen van het ontwerp gemaakt.

3.1 Release Path

In deze paragraaf worden alle diagrammen behandeld van het onderdeel ReleasePaths.

3.1.1 Klasse diagram



Figuur 20 AngularJS diagram van release paths

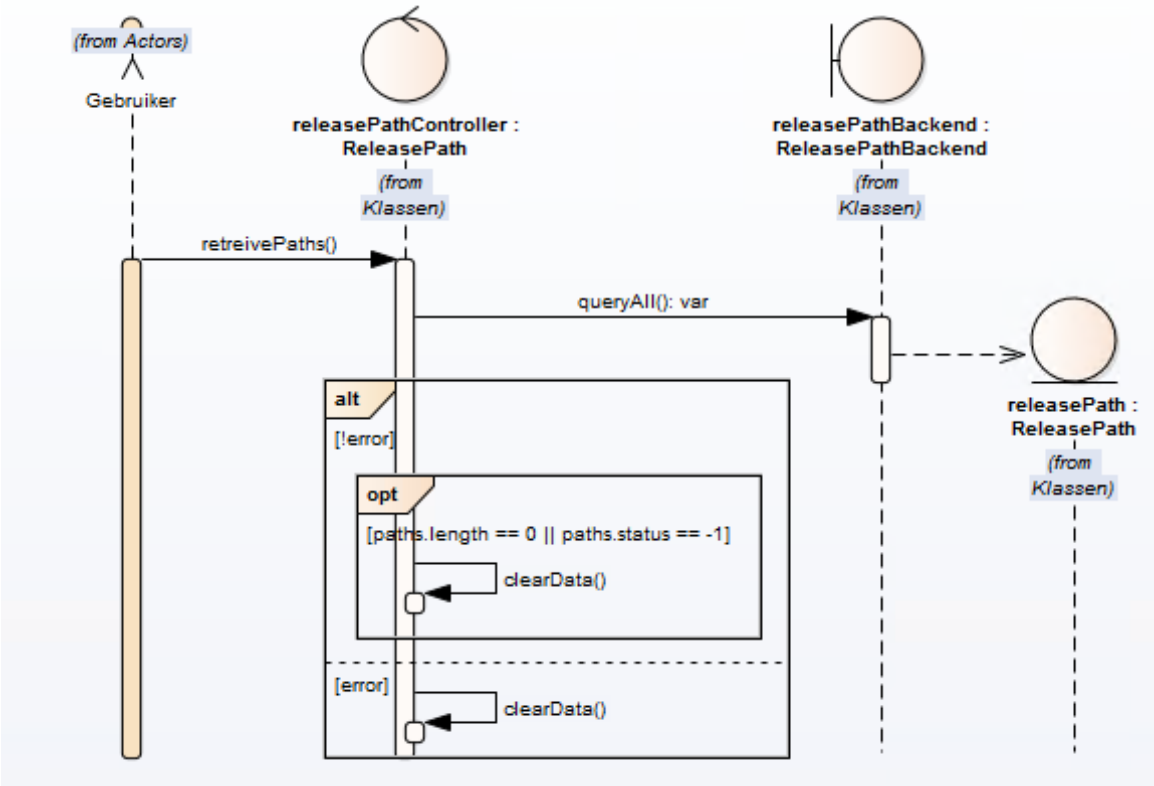
Beslissing ReleasePath Backend
In figuur 20 is te zien dat er een klasse is ReleasePathBackend. Deze klasse is een service in AngularJS. AngularJS maakt van al haar services een Singleton bij het inladen van de service. Hierdoor is het mogelijk deze service overal

in de applicatie aan te spreken. Er is besloten de verbinding met de database in een service te zetten zodat deze vanuit meerdere controllers gebruikt kan worden. Hierdoor vermijden we het hergebruik van code voor het ophalen van de ReleasePaths.

3.1.2 Sequentie diagrammen

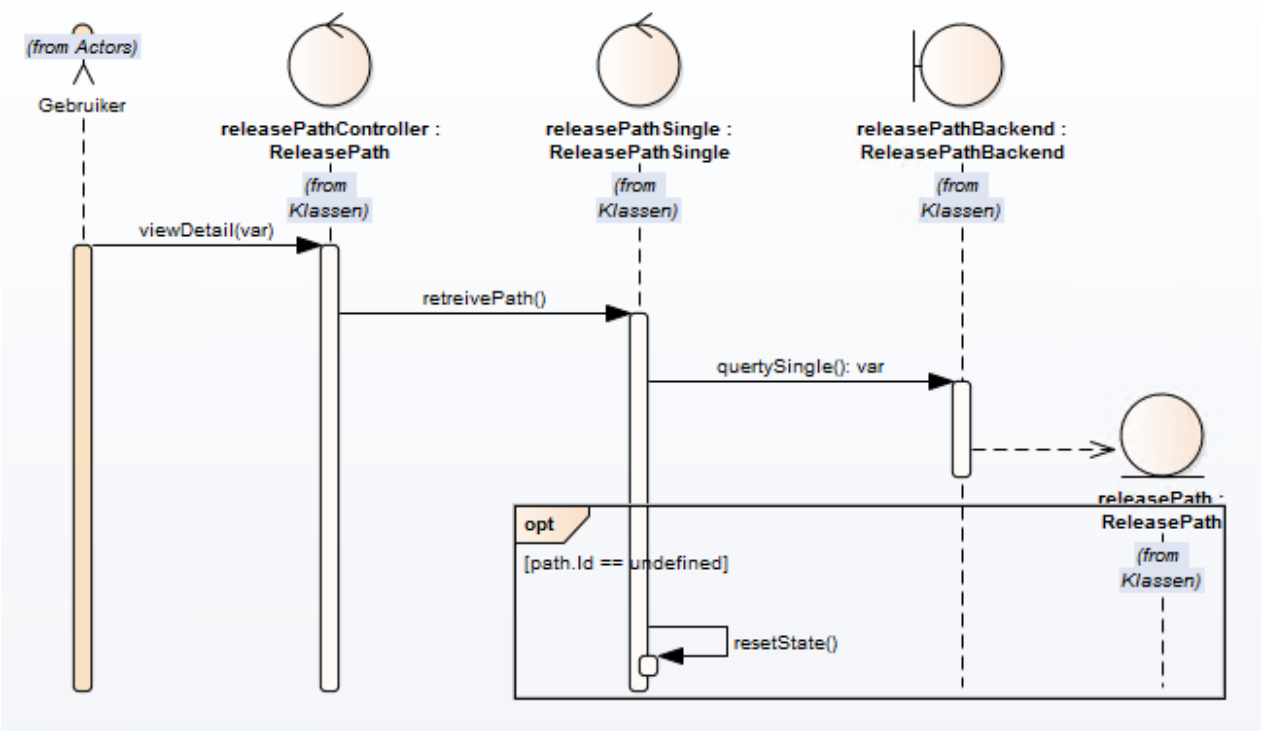
Voor het weergeven van een ReleasePath en meerdere ReleasePaths zijn er sequentie diagrammen gemaakt. In deze sequentie diagrammen is duidelijk te zien met welke controllers er wordt gesproken voor het ophalen van alle gegevens die nodig zijn aan de AngularJS kant.

ReleasePaths



Figuur 21 sequentie diagram voor het ophalen van release paths

ReleasePath



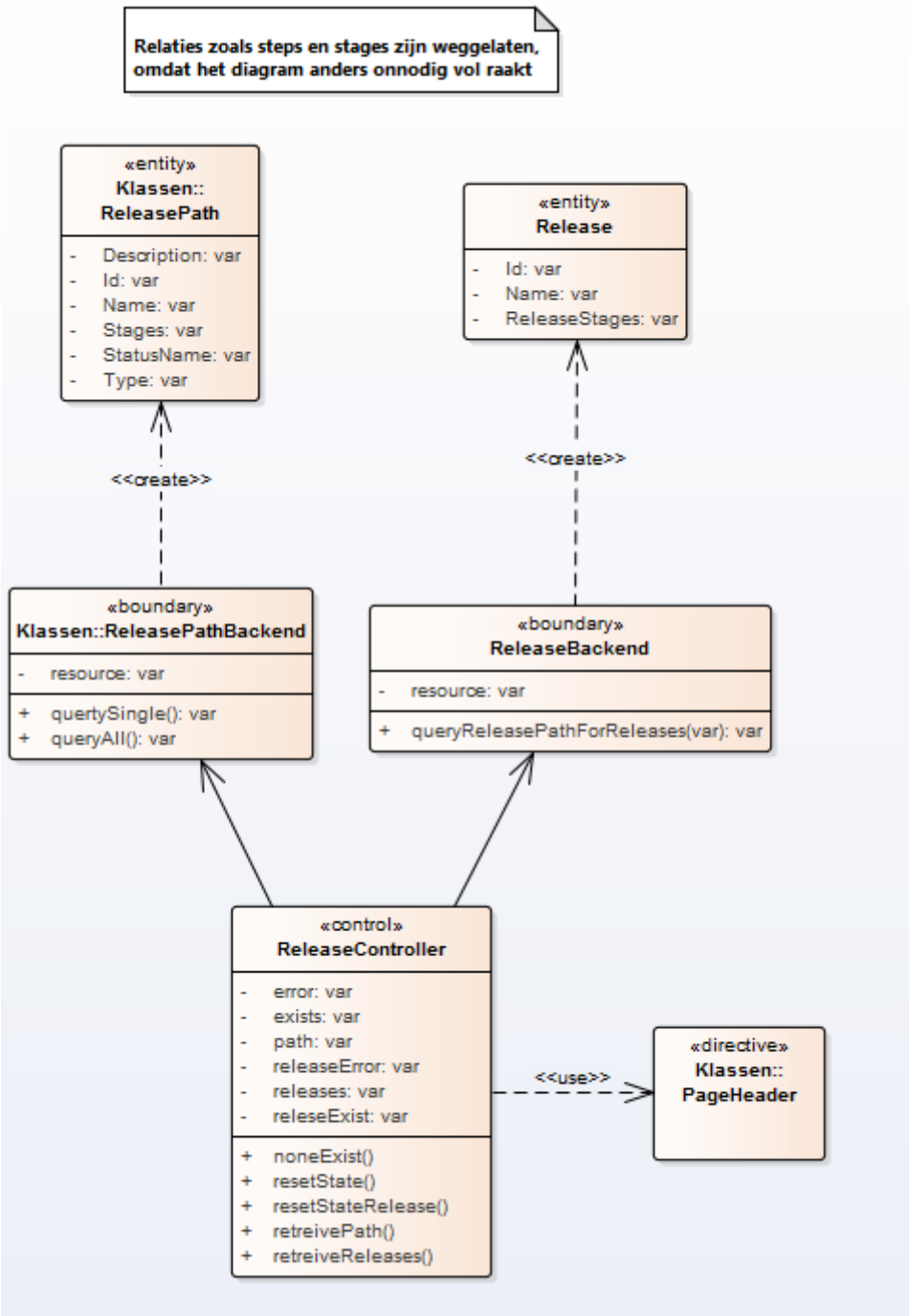
Figuur 22 sequentie voor het ophalen van één release path

3.2 Releases

In deze paragraaf worden de diagrammen van het AngularJS deel beschreven. Deze diagrammen horen bij de tweede sprint.

3.2.1 Klasse diagram

In het volgende diagram is te zien hoe de klasse van het AngularJS deel opgebouwd zijn. Er wordt weer gebruik gemaakt van de service van releasepaths, maar voor het ophalen van de releases is er een nieuwe service gemaakt.

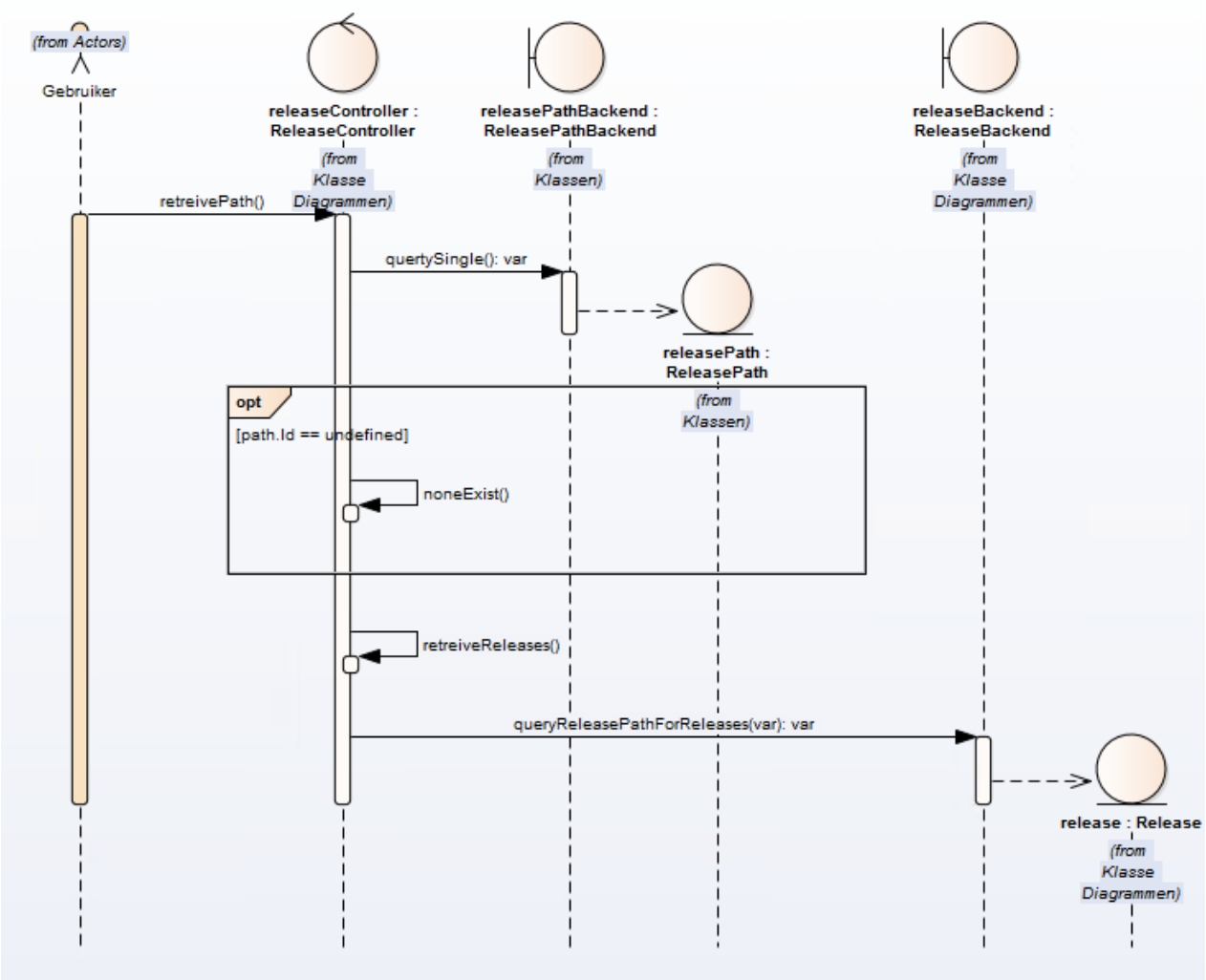


Figuur 23 klasse diagram AngularJS release paths

3.2.2 Sequentie diagrammen

In het volgende diagram is de sequentie te vinden van het AngularJS deel.

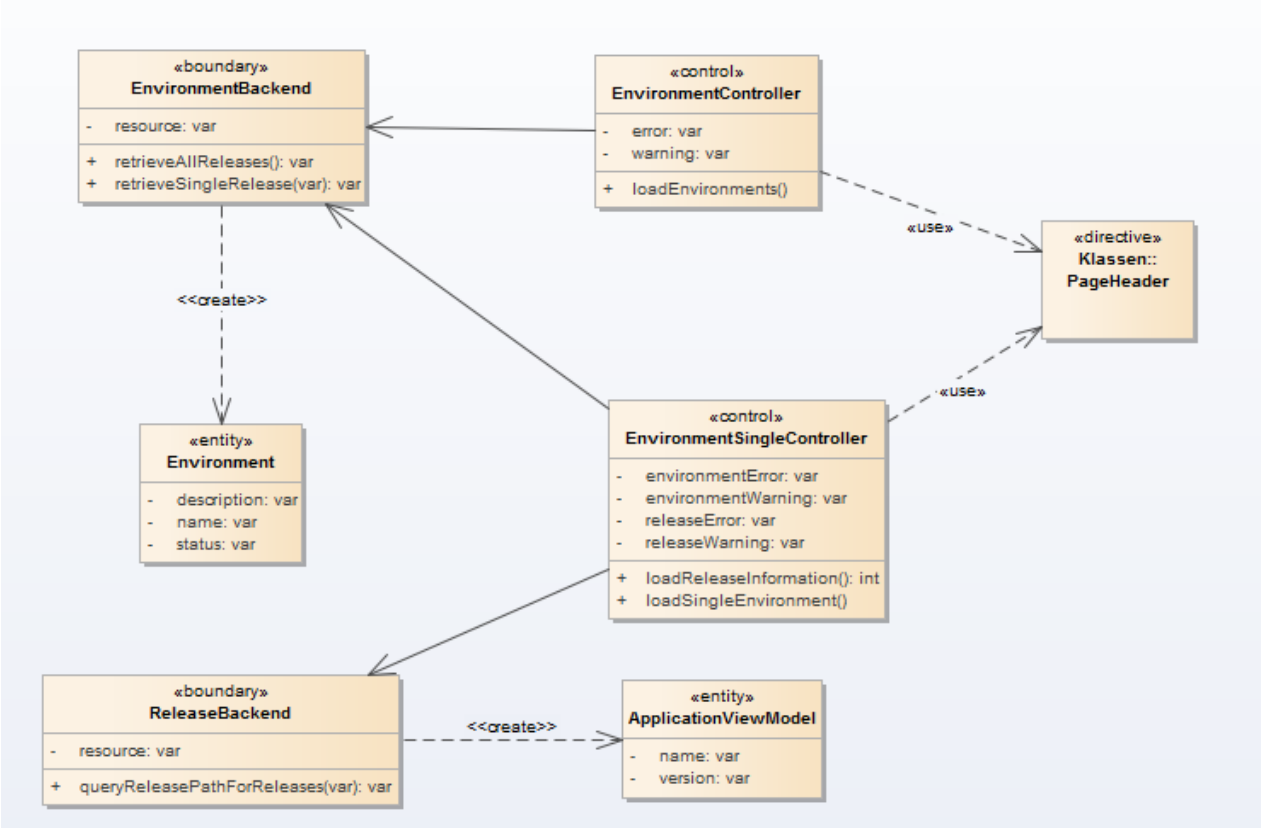
Ophalen releases die bij een release path horen



Figuur 24 sequentie ophalen Releases AngularJS

3.3 Omgevingen

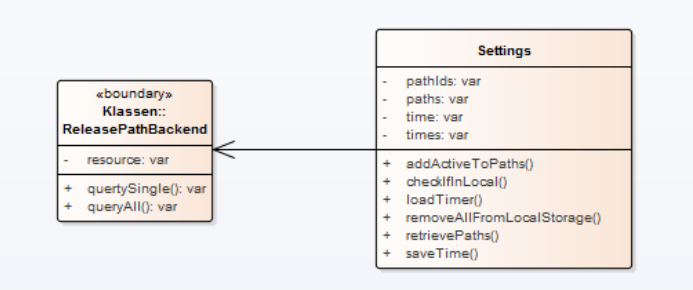
In de AngularJS code van deze sprint is te zien waarom het belangrijk is om gebruik te maken van verschillende services. De EnvironmentSingleController is verantwoordelijk voor het weergeven van een Environment en Release informatie. Het ophalen van Release informatie wordt al gedaan in de service die daar verantwoordelijk voor is. Hierdoor is er geen dubbele code ontstaan.



Figuur 25 klasse diagram AngularJS Releases

3.4 Instellingen

In het volgende diagram is te zien welke klasse erbij zijn gekomen in het AngularJS deel. Er is duidelijk te zien dat de instelling klasse de ReleasePathBackend hergebruikt.



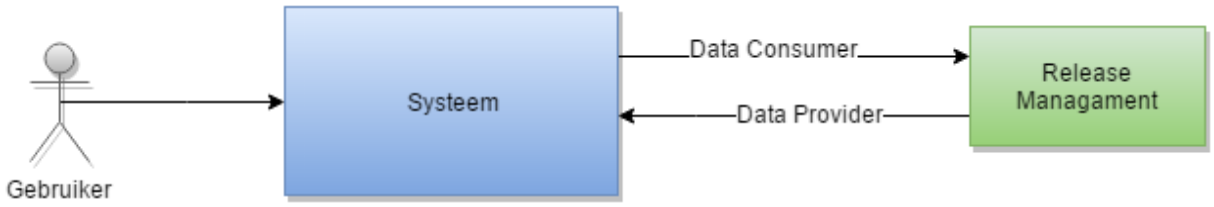
Figuur 26 klasse diagram instellingen

4. Architectuur

Om een duidelijk beeld te krijgen hoe de twee delen samen met elkaar en Release Management communiceren wordt er ook aandacht besteed aan de architectuur van het systeem. In dit hoofdstuk zijn alle beslissingen te vinden met betrekking tot het architectuur.

4.1 Context

In figuur 27 is te zien met welke systemen het dashboard mee gaat communiceren.



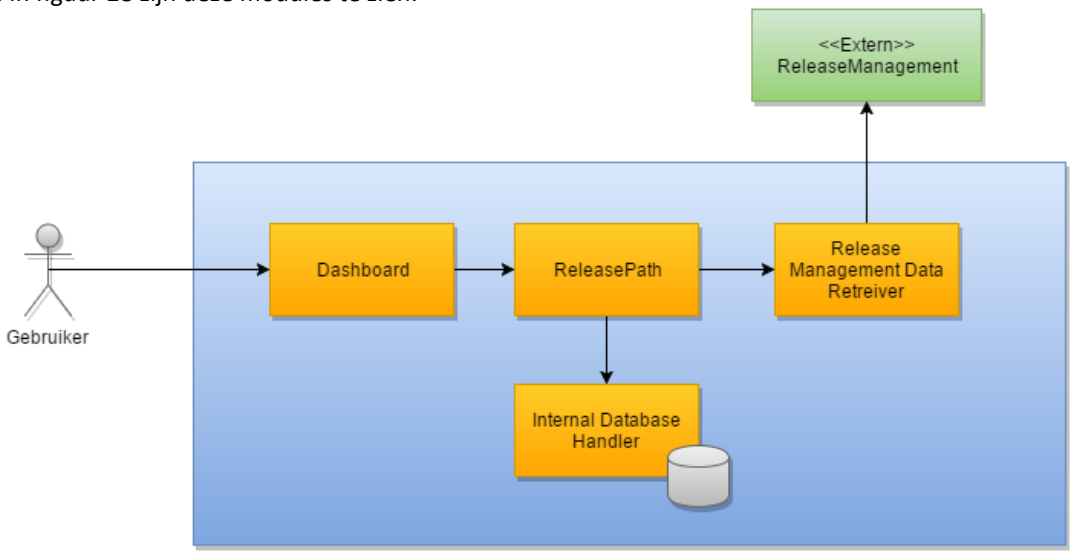
Figuur 27 context te dashboard

Actor
Het dashboard kent maar één actor, namelijk de gebruiker. De gebruiker zal het systeem gebruiken om duidelijke overzichten van Release Management in één oogopslag te zien. Verder zal het voor hem ook mogelijk zijn om specifieke informatie van Release Management op te zoeken.

Extern systeem
Release Management zal gebruikt worden voor het ophalen van informatie dat weergegeven moet worden. In onze situatie is Release Management een data provider en is het systeem een Data consumer. Het is belangrijk dat de verbinding tussen deze twee systemen zodanig wordt gebouwd dat het gemakkelijk te vervangen is met een ander verbinding. Dit moet gedaan worden, omdat er bij nieuwe versies van Release Management veel kan veranderen aan de structuur van de verbinding.

4.2 Functioneel

Het systeem bevat een aantal modules die ervoor zorgen dat de gebruiker alle acties kan uitvoeren met het systeem. In figuur 28 zijn deze modules te zien.



Figuur 28 opbouw modules van één onderdeel dashboard

Dashboard
Het Dashboard module is verantwoordelijk voor het presenteren van alle overzichten die de gebruiker nodig heeft. Verder kan de gebruiker ook instellingen wijzigen van het systeem met deze module.

Release Management Data Retriever
Deze module is verantwoordelijk voor het aanbieden van een abstractie laag tussen het systeem en Release Management. Deze abstractie laag zorg ervoor dat er met andere versies van Release Management gecommuniceerd kan worden zonder dat het systeem daar rekening mee hoeft te houden.

Internal Database Handler
De Internal Database Handler zorgt ervoor dat alle informatie die uit Release Management komt kan worden opgeslagen in een database. Het systeem hoeft geen kennis te hebben van welke database gebruikt wordt. De Internal Database Handler zorgt namelijk voor een abstractie laag op de database. Zo kan een database gemakkelijk vervangen worden zonder dat het systeem uit komt te staan.

ReleasePath
De ReleasePath module verbindt de Release Management Data Retriever met de Internal Database Handler. Deze module zorgt ervoor dat informatie die afkomstig is van Release Management wordt gecached in de database. Verder is deze module ook verantwoordelijk voor het legen van de cache zodra de informatie niet meer recent is. Als laatste biedt de ReleasePath module ook de mogelijkheid voor het dashboard te verkrijgen.

4.3 Informatie

4.3.1 Communicatie format

In deze paragraaf wordt er tijd besteed aan het analyseren van gegevens waarmee het systeem te maken zal hebben. Dit gaat voornamelijk om de externe bronnen, maar ook de gegevens die het systeem zal aanbieden aan haar eigen dashboard.

Release Management 2015 update 1 – XML
De data dat uit Release Management 2015 update 1 komt bestaat uit XML. De Release Management Data Retriever module zal deze data moeten omzetten modellen die gelezen kunnen worden door het systeem zelf.

ReleasePath – JSON/XML
De ReleasePath module zal een REST API aanbieden waarmee het dashboard kan communiceren. Een REST API biedt meestal de mogelijkheid om JSON en XML uit te lezen. Welk formaat gewenst is wordt meestal aangegeven bij het opvragen van de gegevens.

4.3.2 Opslag Format

Bij het ophalen van alle Releases in MRM is er een duidelijke vertraging in het deserialiseren. Dit komt omdat er voor elke Release gedetailleerde informatie opgehaald moet worden. Om ervoor te zorgen dat een gebruiker niet te lang moet wachten en dat het dashboard niet voortdurend verzoeken verstuurd naar MRM is er besloten gebruik te maken van een cache. Een cache heeft als nadeel dat er concurrency problemen kunnen ontstaan. Het kan namelijk voorkomen dat twee requests tegelijk de cache gaan proberen te verversen. Hierdoor moet er duidelijk gemaakt worden welke klasse en methoden gelocked moeten worden en wat de gevolgen hiervan zijn.

Voor het opslaan van de gegevens in de cache wordt er gebruik gemaakt van de database MongoDB. Er is voor MongoDB gekozen, omdat de data die momenteel uit Release Management 2015 update 1 komt aangepast gaat worden in toekomstige versies van Release Management. Dat kan worden afgeschermd met de abstractie laag die gebouwd gaat worden, maar dat betekent niet de bepaalde gegevens niet meer beschikbaar zullen zijn. sMongoDB gaat goed om met data dat geen vaste structuur heeft. Alle gegevens die worden opgeslagen worden in een collectie opgeslagen. In deze collectie kunnen alle elementen er anders uit zien.

Indien het noodzakelijk is om een andere opslag te kiezen kan dit zonder teveel moeilijkheden. Zolang de toekomstige implementatie van een opslag methode voldoet aan de abstractie laag kan dit vervangen worden.

4.4 Ontwikkelen

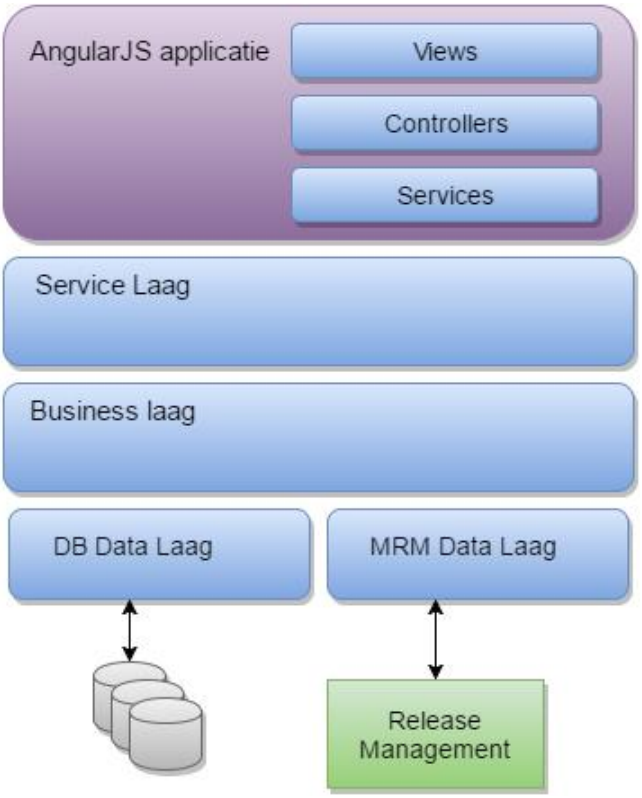
4.4.1 Lagenmodel

Het systeem zal verdeelt worden in lagen. Deze lagen zorgen voor een scheiding tussen bepaalde functionele elementen. Het is namelijk niet gewenst dat het dashboard element ook direct de database kan aanspreken zonder dat de business laag hier van af weet. Als dit mocht dan was er geen manier om ervoor te zorgen dat de informatie die opgeslagen was correct was.

In figuur 29 is het lagen model te zien van het systeem. Er is gekozen voor deze lagen model, omdat het hiermee mogelijk is om af te dwingen dat een functionele element geen stap mag overslaan in het systeem. Het zal niet mogelijk zijn voor het dashboard om direct te communiceren met Release Management. Het dashboard zal moeten spreken met de service laag.

Met dit lagen model encapsuleren we bepaalde gegevens en bieden we een interface aan om deze gegevens op te halen. Bij het wijzigen van een van de lagen is het ook niet noodzakelijk om het systeem helemaal aan te passen. Zolang de nieuwe laag dezelfde interface aanbiedt zal het systeem blijven werken zoals verwacht.

Het zal ook mogelijk zijn om nieuwe systemen toe te voegen aan het systeem. Als deze kunnen communiceren met de service laag is het mogelijk nieuwe clients toe te voegen aan het systeem zonder dat er wijzigingen aangebracht moeten worden.



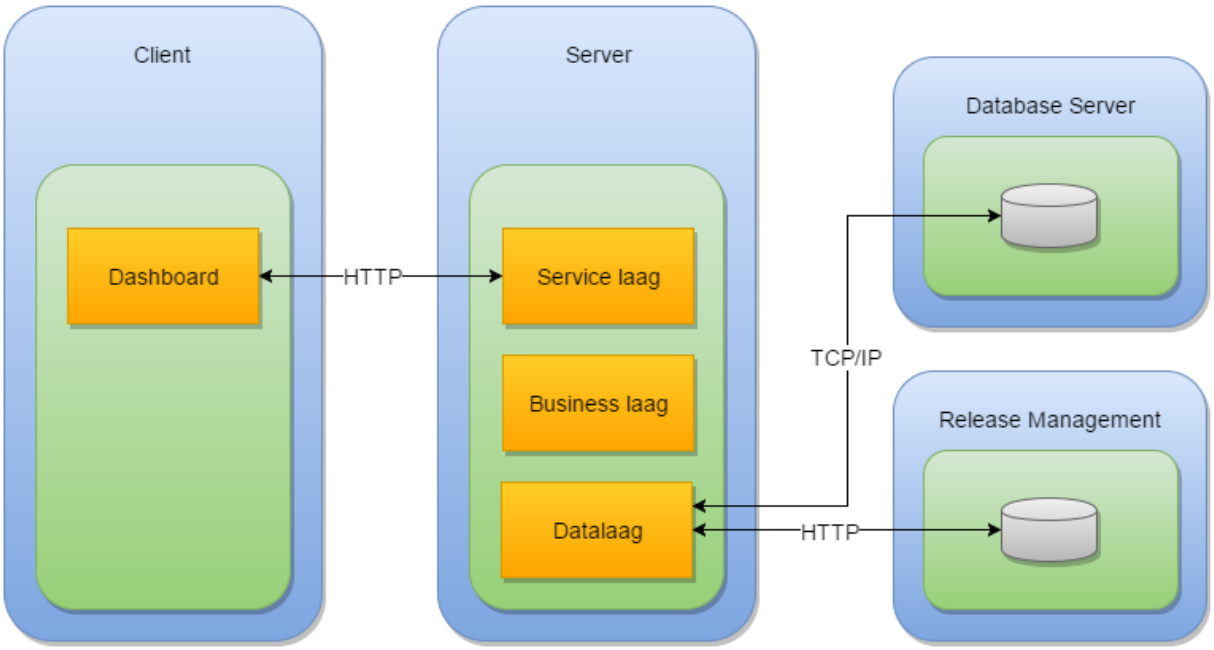
Figuur 29 lagenmodel applicatie

Service laag
De service laag zal worden gerealiseerd door gebruik te maken van WebApi. Dit is een framework van Microsoft dat het mogelijk maakt om API's mee te schrijven

Dashboard en eventuele andere clients
Het dashboard moet op verschillende clients bekeken kunnen worden. Deze clients zijn de verschillende browsers. Naar aanleiding van een gesprek met de opdrachtgever blijkt het dat de meest gebruikte clients Internet Explorer en Safari zijn.

4.4.2 Scheiding Architectuur

Het systeem wordt verspreid op verschillende locaties. Alles dat te maken heeft met presentatie wordt uitgevoerd op de client. De logica van de applicatie is te vinden op de server. Een voordeel van deze scheiding is dat de gebruiker niet merkt wanneer er een update wordt uitgevoerd op de applicatie. Als de applicatie op de server wordt geüpdatet wordt het dashboard bij de eerst volgende verbinding opnieuw ingeladen en communiceert het weer alleen met de service laag. De client heeft geen geïnstalleerde bestanden die aangepast moeten worden. Alleen een werkende browser is nodig.



Figuur 30

Alle clients die willen communiceren met de service laag moeten dit doen door HTTP requests. Deze protocol wordt gebruikt voor het versturen van een verzoek om data op te halen. Indien dit verzoek correct is verstuurd zal de service laag de gewenste gegevens terug sturen. Een API communiceert altijd met HTTP. Bij het uitvoeren van een HTTP request moet er altijd een HTTP method worden meegegeven. Deze HTTP methods bestaan uit GET, POST, PUT en DELETE. De get wordt gebruikt bij het ophalen van gegevens. De POST, PUT en DELETE worden gebuikt om wijzigingen uit te voeren op de gegevens. Aangezien het systeem alleen gegevens toont wordt alleen de GET HTTP method ondersteund. De data laag maakt verbinding op twee manieren. De verbinding met Release Management wordt ook gedaan door middel van het HTTP protocol. Elk request die naar Release Management gaat wordt beantwoord met XML gegevens.

Bijlage V

Handleiding

Handleiding

Visual Deployment Pipeline

Marc De Meza

Inhoud

1. Inleiding

In deze handleiding staat beschreven welke stappen uitgevoerd moeten worden om de Visual Deployment Pipeline (VDP) online te krijgen. Hierbij wordt er gekeken naar de instellingen, Microsoft Release Management (MRM) account, deployment en testomgeving.

2. Instellingen

Voordat de applicatie gebruikt kan worden zijn er een aantal instellingen die in de web.config ingesteld moeten worden. Deze instellingen zijn als volgt:

```
<appSettings>
  <add key="ReleaseManagementApiLink" value="https://rm.rhea.infosupport.net/" />
  <add key="ApiVersion" value ="3.0"/>
  <add key="Username" value ="CRONOS\marcm"/>
  <add key="MongoDbLink" value="ReleaseManagementMongoDatabase" />
  <add key="LimitReleases" value="20" />
  <add key="ReleaseCache" value="5" />
  <add key="EnvironmentCache" value="20" />
  <add key="ReleasePathCache" value="20" />
  <add key="ReleaseDate" value="6" />
</appSettings>
```

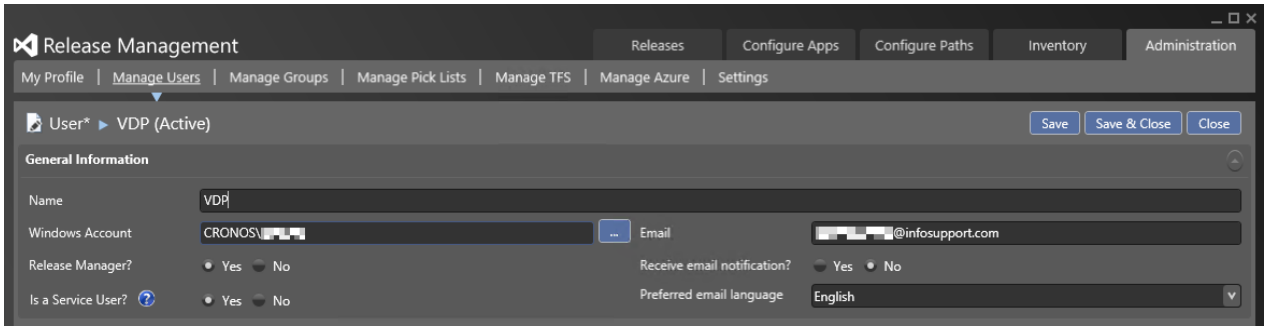
Figuur 1 instellingen web.config

Instelling	Betekenis
ReleaseManagementApiLink	Dit is de link naar de MRM API. Deze link moet inclusief port zijn.
ApiVersion	Deze instelling moet aangepast worden aan de hand van de versie van MRM. 2015 Update 1 kan 6.0 of 7.0 gebruiken. 2013 update 4 gebruikt 3.0
Username	Naast het instellen van de username als AppPool identity moet de applicatie ook weten namens welke account de requests worden uitgevoerd. Deze username moet dezelfde zijn als de AppPool identity(4.1).
MongoDbLink	Deze link is nodig voor de MongoDb database.
LimitReleases	Deze waarde bepaald hoeveel Releases naar de gebruiker worden verstuurd in de Home overzicht.
ReleaseCache	Deze waarde bepaald na hoeveel minuten de cache van Release wordt ververst. Deze waarde wil je zo laag mogelijk om recente informatie te zien.
EnvironmentCache	Deze waarde bepaald na hoeveel minuten de cache van Omgevingen wordt ververst.
ReleasePathCache	Deze waarde bepaald na hoeveel minuten de cache van Release Paths wordt ververst.

Instelling	Betekenis
ReleaseDate	In MRM is het mogelijk om alle Release Paths met een bepaalde datum range op te halen. In de MRM client wordt dit aangegeven met een nummer. 5 betekent de afgelopen 28 dagen en 0 betekent alles dat in geschiedenis staat. 0 Moet niet worden gebruikt als er veel Releases in het systeem staan.

3. Release management account

In MRM is nog het nog nodig om een account te maken voor de VDP waarmee er gegevens uit MRM gehaald kunnen worden. Het Windows Account dat hier gebruikt wordt moet later ook op de IIS server ingesteld worden.



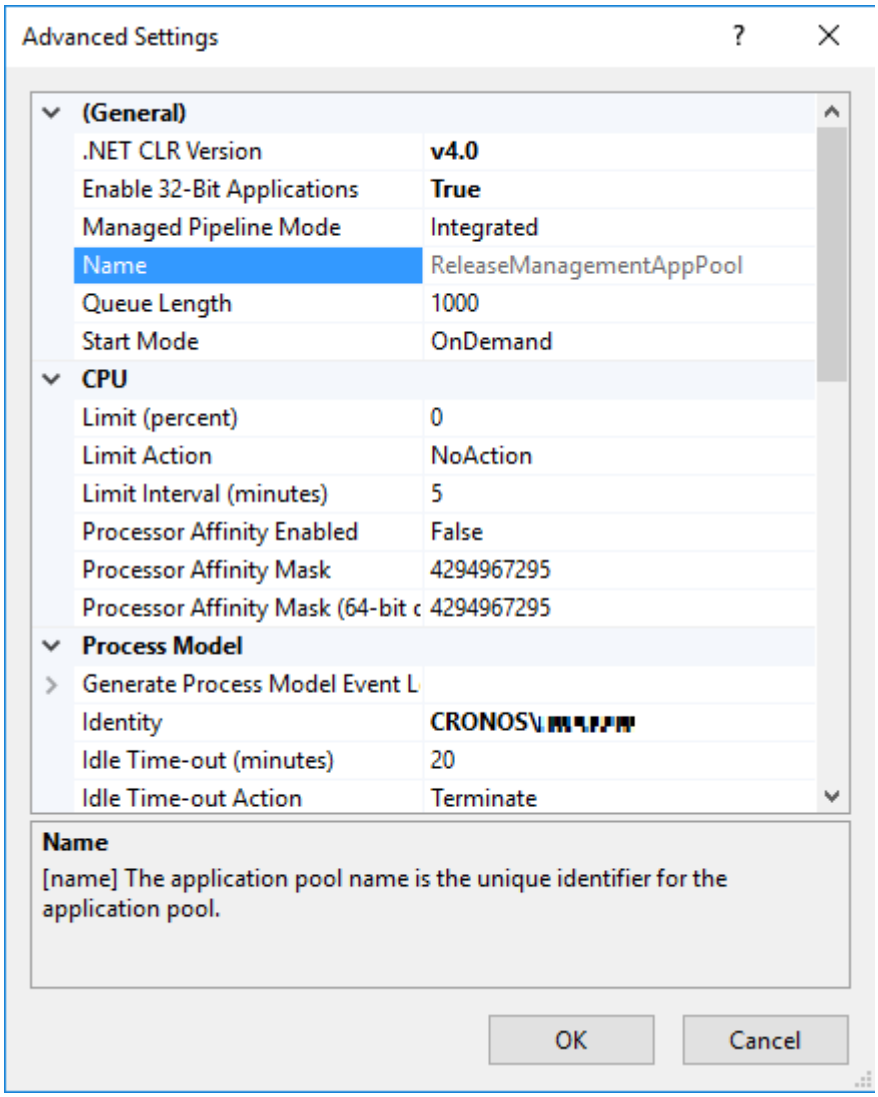
3.1 Rechten

De rechten van het account waarmee de VDP informatie ophaalt kunnen ook beperkt worden. Het account wordt alleen gebruikt om gegevens op te halen. Er worden geen instellingen gewijzigd of Releases uitgevoerd met dit account.

4. Deployment

4.1 IIS identity

Wanneer de VDP verbindt met MRM wordt de account van de VDP geverifieerd. Normaal zal de identity van de App pool gebruikt worden. Dit levert niet altijd de gewenste resultaten. Daarom is het nodig om de identity van de App pool te wijzigen naar de account die in MRM is ingesteld.



Figuur 2 Identity van de app pool wijzigen

4.2 MongoDB

De VDP gebruikt MongoDB om alle gegevens te cachen. De versie die momenteel ondersteunt wordt is 3.2 en de installatie handleiding is te vinden op https://docs.mongodb.com/manual/tutorial/install-mongodb-on-windows/. Nadat de installer is gerund en MondoDB is geïnstalleerd moet deze nog geconfigureerd worden om als een service te draaien. Het configureren kan als volgt:

Eerst maken we een locatie waar alles opgeslagen moet worden

```
mkdir c:\data

mkdir c:\data\db

mkdir c:\data\log
```

Vervolgens navigeren we naar de locatie waar MongoDB is geïnstalleerd en maken we daarin een bestand aan die "mongod.cfg" heet. In dit bestand moet het volgende staan:

```
systemLog:

    destination: file

    path: c:\data\log\mongod.log

storage:

    dbPath: c:\data\db
```

Als laatste moeten we nog de service installeren door de volgende stappen uit te voeren:

```
"C:\mongodb\bin\mongod.exe" --config "C:\mongodb\mongod.cfg" --install

net start MongoDB
```

5. Testomgeving

Indien de applicatie automatisch getest gaat worden is het van belang om te weten dat er twee verschillende test soorten zijn de afhankelijk zijn van MRM en de data die erin staat. Deze tests zijn de UI- en de integratietests. Bij het uitvoeren van deze tests is het belangrijk dat er verbonden kan worden met MRM. Verder is het ook belangrijk dat er bepaalde data in staat. Welke data dit precies is, staat genoteerd per test. Indien deze omgevingen niet ingericht kunnen worden, moeten deze tests niet automatisch uitgevoerd worden.

5.1 Backup MRM database

Indien nodig is er ook een backup gemaakt van de data dat gebruikt is gedurende het ontwikkelen van de VDP. Deze is wordt ook verstuurd met deze handleiding.

Bijlage VI

Afstudeerplan

Afstudeerplan

Informatie afstudeerder en gastbedrijf

Afstudeerblok: 2016-1.1 (start uiterlijk 8 februari 2016)
Startdatum uitvoering afstudeeropdracht: 1 februari 2016
Inleverdatum afstudeerdossier volgens jaarrooster: 3 juni 2016

Studentnummer: 10043659
Achternaam: dhr de Meza
Voorletters: M.R.
Roepnaam: Marc
Adres: Herengracht 78B
Postcode: 2312LE
Woonplaats: Leiden
Telefoonnummer: 0651123848
Mobiel nummer: 0651123848
Privé emailadres: marcdemeza@gmail.com

Opleiding: Informatica
Locatie: Den Haag
Variant: voltijd

Naam studieloopbaanbegeleider: A. Wieman
Naam begeleidend examinator: G.A. Mijnarends
Naam tweede examinator: J.J. van der Hoek

Naam bedrijf: Info Support
Afdeling bedrijf:
Bezoekadres bedrijf: Kruisboog 42
Postcode bezoekadres: 3905TG
Postbusnummer:
Postcode postbusnummer:
Plaats: Veenendaal
Telefoon bedrijf: +31 318 552020
Telefax bedrijf: +31 318 552355
Internetsite bedrijf: www.infosupport.com

Achternaam opdrachtgever: dhr Borgeld
Voorletters opdrachtgever: P
Titulatuur opdrachtgever:
Functie opdrachtgever:
Doorkiesnummer opdrachtgever:
Email opdrachtgever: paul.borgeld@infosupport.com

Achternaam bedrijfsmentor: mw Van de Hoef
Voorletters bedrijfsmentor: L
Titulatuur bedrijfsmentor:
Functie bedrijfsmentor:
Doorkiesnummer bedrijfsmentor:
Email bedrijfsmentor: laila.vandehoef@infosupport.com

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

Titel afstudeeropdracht:
Ontwikkelen van een dashboard applicatie die een overzicht geeft van welke software is geïnstalleerd op welke omgeving met informatie uit Microsoft Release Management van Team Foundation Server.

Opdrachtomschrijving

1. Bedrijf (verder uitwerken met onderstaande punten)

Info Support is een bedrijf dat zich richt op het ontwikkelen van software op maat, grote administratieve bedrijfsapplicaties, maar ook integratie- en business intelligence projecten.

Naast het ontwikkelen van grote IT projecten is Info Support ook in staat om deze projecten te beheren en te onderhouden. Vanaf het begin van een project tot aan het onderhouden hiervan maakt Info Support gebruik van verschillende omgevingen voor Ontwikkelen, Testen, Acceptatie en Productie.

2. Probleemstelling

Info Support maakt gebruik van Team Foundation Server 2013. Team Foundation Server 2013 heeft een nieuwe functionaliteit toegevoegd, namelijk Microsoft Release Management. Met Microsoft Release Management kan een deployment straat gedefinieerd worden. Dit houdt in dat er kan beschreven worden hoe software geïnstalleerd moet worden op welke omgeving (Ontwikkel, Test, Acceptatie en Productie). Tijdens de installatie is er ook een overzicht van hoe de installatie verlopen is en welke stappen zijn uitgevoerd.

Helaas is er in Microsoft Release Management nog geen inzicht in welke versies van software op welke omgeving is geïnstalleerd of hoe de volledige deployment pipeline van een applicatie eruit ziet. Momenteel is er een eenvoudige view gemaakt door Info Support waarin een deel van deze informatie wordt weergegeven. Deze view geeft nog lang niet alle gewenste informatie weer en als er meer informatie nodig is moeten de release managers langs de omgevingen om deze informatie op te halen.

Voor release managers is het van belang om direct grafisch inzicht te kunnen krijgen in welke versies van software op welke omgevingen zijn geïnstalleerd.

3. Doelstelling van de afstudeeropdracht

De afstudeer opdracht zal, door middel van informatie uit Microsoft Release Management, op een centrale locatie grafisch inzicht geven aan de release managers van welke versies van software op welke omgevingen zijn geïnstalleerd.

4. Resultaat

Als de afstudeeropdracht af is zullen release managers een dashboard hebben waarin ze de volledige deployment pipeline van verschillende applicaties kunnen zien. Verder zal het dashboard ervoor zorgen dat er tijd bespaard wordt tijdens het bekijken van de volledige deployment pipeline. Alle informatie zal voor de release managers op een centrale punt te vinden zijn.

Het dashboard zal per applicatie de volledige geschiedenis van de installaties kunnen weergeven. Dit houdt in dat alle informatie over de installatie die uit Microsoft Release Management komt, weergegeven zal worden in het dashboard. Verder zal er ook per omgeving gezien kunnen worden welke versies van welke software is geïnstalleerd. Het dashboard zal via web en mobile benaderd kunnen worden.

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

- Plan van aanpak schrijven
- Analyseren Release Management TFS2013 Data ophalen
- Requirements gesprekken
- Ontwerpen database dashboard
- Ontwerpen Architectuur dashboard
- Bouwen koppeling Release Management
- Bouwen dashboard web
- Bouwen dashboard mobile
- Testen dashboard (Unit/Service/IU)

6. Op te leveren (tussen)producten

- Plan van aanpak
- Requirements document
- Resultaten analyse: mogelijkheden gegevens uit TFS2013 halen
- Ontwerp documentatie
- Ontwerp database
- Implementatie database
- Testresultaten
- Applicatie

7. Te demonstreren competenties en wijze waarop

1.4 Uitvoeren analyse door definitie van requirements
Niveau 3 – Lastig / Zelfstandig

De opdracht betreft één applicatie met een groot aantal requirements. Deze opdracht zal één opdrachtgever hebben, maar meerdere stakeholders. Deze stakeholders zijn de release managers die het uiteindelijk gaan gebruiken.

Een deel van de requirements zijn al bekend bij de opdrachtgever, maar het analyseren van de nieuwe requirements en het vastleggen van de bestaande requirements moeten nog gebeuren. De opdracht zal ook uitgevoerd worden met een agile ontwikkelmethodiek. Dit betekent dat er nog verschillende requirements kunnen wijzigen of erbij komen.

Tijdens het analyseren van de requirements zal er een nieuwe techniek gebruikt worden die door Info Support gehanteerd wordt, namelijk Requirements as Scenario's. Deze scenario's zullen later ook gebruikt worden voor het opstellen en uitvoeren van tests.

3.2 Ontwerpen systeemdeel
Niveau 3 - Lastig / Zelfstandig

Applicatie
De applicatie die gebouwd moet worden zal objectgeoriënteerd zijn. Tijdens het bouwen zal er ook gemodelleerd moeten worden en er moet rekening gehouden worden met uitbreidbaarheid, testbaarheid en hergebruik. Voor het modelleren zal er gebruik worden gemaakt van UML.

Database
Het database die bij deze opdracht hoort zal gebruik maken van meerdere tabellen. Verder moet tijdens het bevragen van de database ook meerdere tabellen geraadpleegd worden. Als laatste moet de integriteit van de database ook gewaarborgd worden.

Architectuur
De dashboard zal gebruik maken van een externe informatie bron, namelijk Microsoft Release Management. Om een goed beeld te krijgen van hoe deze systemen met elkaar communiceren zal er ook tijd worden besteed aan het maken van Architecture Viewpoints.

3.3 Bouwen applicatie
Niveau 4 - Complex / Zelfstandig

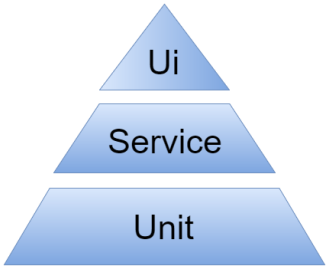
De applicatie gaat een koppeling moeten maken met een bestaande software. Dit betekend dat er data van een bestaand systeem gehaald zal worden en deze zal vervolgens verstuurd worden naar het dashboard. Verder wordt het systeem ook gebouwd in Visual Studio en wordt er gebruik gemaakt van C#, ASP.NET en SQL server. Er wordt ook gebruik gemaakt van verschillende omgevingen, zoals ontwikkel, test, acceptatie en productie.

Voor het maken van de koppeling tussen het dashboard en Microsoft Release Management bestaat er al kennis bij Info Support. In het verleden is het hun gelukt om data te verwerken door middel van het uitlezen van de database. Tijdens de opdracht zal er een analyse worden uitgevoerd om te kijken of er andere manieren bestaan voor het uitlezen van de gegevens uit Microsoft Release Management.

Het accent en de complexiteit van de opdracht ligt in het maken van het dashboard zelf. Dit dashboard zal alle informatie die uit Microsoft Release Management over het deployment pipeline komt moeten verwerken en weergeven zoals afgesproken met de release managers. Verder zal er ook een Mobiele versie van dit dashboard moeten komen.

3.5 Uitvoeren van en rapporteren over het testproces
Niveau 3 - Lastig / Zelfstandig

Info Support verwacht dat de test standaard geautomatiseerd zijn. Info Support gebruikt de testpiramide voor het testen. Dit betekent dat Unit tests breed uitgevoerd zullen worden. Vervolgens worden de services ook getest en als laatste wordt de user interface getest met E2E tests. Verder maakt Info Support ook gebruik van Test Driven Development. Al deze punten zullen tijdens de opdracht uitgevoerd worden.



Voor het testen van het dashboard zal de verbinding met Microsoft Release Management gemocked worden.