Bijlagen E t/m L

MongoDB: Een geschikte database voor het kennismanagementsysteem knowNow?

Door: Tom Keim

Bijlage E – Ontwerp documentschema
Bijlage F – Functioneel Ontwerp
Bijlage G – Technisch Ontwerp
Bijlage H – Testrapportage
Bijlage I – Adviesrapport
Bijlage J – Afstudeerplan
Bijlage K – Evaluatieformulier afstuderen
Bijlage L - Beoordeling tussentijds assessment

Titel	Bijlagen E t/m L	**Tweede Examinator**	Rianne Bechet-Tjoonk
Project	MongoDB: Een geschikte database voor het kennismanagementsysteem knowNow?	**Opdrachtgever**	Joop Snijder
Auteur	Tom Keim	**Procesbeleider**	Pascalle Hijl
Studentnummer	11031425	**Technisch begeleider**	Orhan Uyan
School	Haagse Hogeschool	**Datum**	02-10-2015
Eerste Examinator	Gerard Mijnarends	**Bedrijf**	Info Support B.V.

Bijlagen

In dit boek zijn de volgende bijlagen te vinden:

Bijlage E	**Ontwerp documentschema**
Bijlage F	**Functioneel ontwerp**
Bijlage G	**Technisch ontwerp**
Bijlage H	**Testrapportage**
Bijlage I	**Adviesrapport**
Bijlage J	**Afstudeerplan**
Bijlage K	**Evaluatieformulier afstuderen**
Bijlage L	**Beoordeling tussentijds assessment**

DE HAAGSE HOGESCHOOL

Info Support Solid Innovator

Bijlage E – Ontwerp documentschema

MongoDB: Een geschikte database voor het kennismanagementsysteem knowNow?

Ontwerp documentschema

Titel	Ontwerp documentschema
Project/Onderwerp	MongoDB: Een geschikte database voor het kennismanagementsysteem knowNow?
Versie	1.0
Status	Definitief
Datum	24-08-2015
Bestand	Ontwerp documentschema v1.0
Bedrijf	Info Support
Door	Tom Keim

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

Inhoudsopgave

1. Inleiding

In dit document worden de documentschema's gebruikt in het Proof-of-Concept van knowNow met MongoDB besproken.

In eerste instantie werd er voor het documentschema van uitgegaan dat alle waarden die getoond moeten worden op een Knowledge Item pagina, al in een document zitten en hier dus niet gejoined voor hoeft te worden (Hiërarchisch documentschema). Hiervoor was gekozen omdat het toevoegen van nieuwe, extra, velden geen groot performanceverlies oplevert[1].

Daarnaast was het opnemen van sub documenten in plaats van het leggen van relaties ook het principe van een MongoDB database. MongoDB kent namelijk geen joins[2]. Je kunt wel refereren naar een id van een ander item. Zelf raad MongoDB aan gebruik te maken van subdocumenten[3] wanneer het geen groot performanceverlies oplevert[4].

Echter is door diverse beweegredenen besloten ook te kijken naar een relationeel documentschema waar uiteindelijk ook voor gekozen is. In het relationeel documentschema wordt verwezen naar de bijhorende relaties. De beweegredenen en gemaakte keuze's zijn beschreven in het Onderzoeksrapport, Deelvraag 4: "Is het voor knowNow beter een hiërarchisch documentschema te hanteren of een relationeel documentschema?".

Beide documentschema's zijn hieronder te vinden.

[1] Onderzoek MongoDB als mogelijke vervanger van het huidige RDBMS voor het kennismanagement systeem knowNow, Paragraaf 4.4.3
[2] Onderzoek MongoDB als mogelijke vervanger van het huidige RDBMS voor het kennismanagement systeem knowNow, Paragraaf 4.2.4
[3] MongoDB, "Data Model Design" http://docs.mongodb.org/manual/core/data-model-design/
[4] Onderzoek MongoDB als mogelijke vervanger van het huidige RDBMS voor het kennismanagement systeem knowNow, Paragraaf 4.4.3

2. Documentschema Knowledge Items (Hiërarchisch documentschema)

In dit hoofdstuk worden het Hiërarchisch documentschema van de Knowledge Item collectie besproken. In dit documentschema worden alle relaties opgenomen als subdocument.

```
knowledge-document:
{
    _id: ...,
    title: "Titel van Knowledge Item",
    description: "description",
    rating: {
        currentRating: 4.5,
        numberOfRatings: 10
    },
    tags: ["tagA","tagB","tagC", "tagD", "tagE"],
    creator: people-document,
    author: people-document,
    coauthors: [
        people-document,
        ...
    ]
    followers: [
        people-document,
        ...
    ]
    communities: [
        community-document,
        ...
    ],
    comments: [
        comment-document,
        ...
    ]
    views: 100,
    creationDate: date,
    publicationDate: date,
    modificationDate: date,
    visibility: public|private,
    category: "Article"|"Concept"|"Practice"|"Presentation"|"Sample"|"Standard & Guidline"|"Template"|"Tool",
    deleted: true|false,
    source: 'http://'
}
```

Naam veld	Betekenis
_id	Een unieke code welke het Knowledge Item identificeert. Is voor elk Knowledge Item anders. De code wordt opgeslagen volgens MongoDB's ObjectID formaat. Voorbeeld: ObjectId("559447ea1bed1d408356e1e7")
title	De titel van het Knowledge Item. In string formaat.
description	De tekst van het Knowledge Item. In string formaat.
rating	De waardering van het Knowledge Item. Het subdocument bevat de huidige rating van het Knowledge Item en hoeveel mensen het item hebben beoordeeld.
tags	De tags van een Knowledge Item. In een array met strings opgeslagen. De strings in de array zijn de namen van de tags.
creator	De maker van het Knowledge Item. Bevat naar een People subdocument. Dit subdocument wordt besproken in de tabel 'Documentschema People Item'
author	De auteur van het Knowledge Item. Dit kan de creator zijn, maar er kan ook een andere auteur worden opgeslagen. Het subdocument is een People Subdocument. Deze wordt besproken in de tabel 'Documentschema People Item'.
coauthors	De coauteurs van het Knowledge Item. Dit kunnen meerdere subdocumenten van het type 'Documentschema People Item'
followers	De volgers van het Knowledge Item. Gebruikers kunnen een Knowledge Item volgen. Wordt opgeslagen als een array met daarin een of meerdere People Subdocumenten. Deze wordt besproken in de tabel 'Documentschema People Item'.
communities	Bevat de communities waaronder het Knowledge Item valt. Bevat een array met daarin één of meerdere Community Subdocumenten. Deze wordt besproken in de tabel 'Velden Subdocument Communitiy'.

comments	Bevat de reacties die op het Knowledge Item zijn geplaatst. Bevat een array met daarin één of meerdere Comment Subdocumenten. Deze wordt besproken in de tabel 'Velden Subdocument Comments'.
views	Het aantal views van het Knowledge Item. Opgeslagen als een integer.
creationDate	De datum waarop het artikel is gecreëerd. Opgeslagen in het MongoDB date formaat.
publicationDate	De datum waarop het artikel is gepubliceerd. Opgeslagen in het MongoDB date formaat.
modificationDate	De datum waarop het artikel voor het laatst is aangepast. Opgeslagen in het MongoDB date formaat.
visibility	Public of Private. Bij Public is het Knowledge Item voor alle ingelogde gebruikers zichtbaar, bij Private alleen voor de auteurs en gebruikers die lid zijn van de community waar het item onder valt.
category	De categorie waar het item onder valt. Een item kan maar onder één categorie vallen.
deleted	True of False. Bij True is het Knowledge Item verwijderd en mag deze niet meer getoond worden in de applicatie. Bij False is het artikel niet verwijderd.
source	De source van het item. Kan een link zijn naar een website waar het artikel op gevonden is.

3. Documentschema Knowledge Items (Relationeel documentschema)

In dit hoofdstuk worden het Relationeel documentschema van de Knowledge Item collectie besproken. In dit documentschema wordt gerefereerd naar de bijhorende People Items bij het creator veld, authors veld en coauthors veld. De rest van het documentschema is hetzelfde.

```
knowledge-document:
{
    _id: ...,
    title: "Titel van Knowledge Item",
    description: "description",
    rating: {
        currentRating: 4.5,
        numberOfRatings: 10
    },
    tags: ["tagA","tagB","tagC", "tagD", "tagE"],
    creator: 'peopleServiceItemId',
    author: 'peopleServiceItemId',
    coauthors: [
        'peopleServiceItemId',
        ....
    ],
    /// rest van datamodel
}
```

In de tabel hieronder zijn de velden die anders zijn in het Hiërarchisch documentschema beschreven.

Naam veld	Betekenis
creator	De maker van het Knowledge Item. Verwijst naar een People Item.
author	De auteur van het Knowledge Item. Dit kan de creator zijn, maar er kan ook een andere auteur worden opgeslagen. Het verwijst naar een is een People Item.
coauthors	De coauteurs van het Knowledge Item. Dit kunnen meerdere verwijzingen zijn naar People Items.

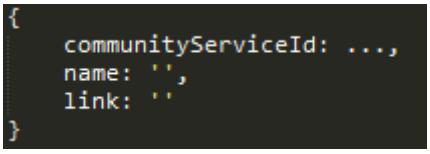
4. Documentschema People Item (Hiërarchisch & Relationeel documentschema)

In dit hoofdstuk worden de velden van het People Subdocument besproken. De velden zijn in beide documentschema's hetzelfde.

Naam veld	Betekenis
peopleServiceItemId	Een uniek id van het item in de people service. Opgeslagen volgens het MongoDB ObjectID formaat.
name	De naam van de persoon, in string formaat.
profileImage	De profielafbeelding van de persoon. Een link naar die afbeelding. In string formaat.
subText	De tekst die onder de naam van de persoon kan staan, zoals een sub titel. Kan

5. Documentschema Subdocument Communities (Hiërarchisch & Relationeel documentschema)

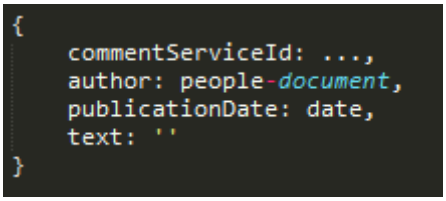
In dit hoofdstuk worden de velden van het Community Subdocument besproken. De velden zijn in beide documentschema's hetzelfde.



Naam veld	Betekenis
communityServiceItemId	Een uniek id van het item in de Community service. Opgeslagen volgens het MongoDB ObjectID formaat.
name	De naam van de community, in string formaat.
link	De link van de community, in string formaat.

6. Velden Subdocument Comments (Hiërarchisch & Relationeel documentschema)

In dit hoofdstuk worden de velden van het Comments Subdocument besproken. De velden zijn in beide documentschema's hetzelfde.



Naam veld	Betekenis
commentServiceId	Een uniek id van het item in de Comment service. Opgeslagen volgens het MongoDB ObjectID formaat.
Author	De auteur van de reactie. Verwijst naar een People Subdocument
publicationDate	De publicatie datum van de reactie. Opgeslagen in het MongoDB date formaat.
text	De tekst van de reactie. Opgeslagen in string formaat.

DE HAAGSE HOGESCHOOL

Info Support Solid Innovator

Bijlage F – Functioneel Ontwerp

MongoDB: Een geschikte database voor het kennismanagementsysteem knowNow?

Functioneel Ontwerp

Titel	Functioneel Ontwerp
Project/Onderwerp	MongoDB: Een geschikte database voor het kennismanagementsysteem knowNow?
Versie	1.0
Status	Definitief
Datum	01-10-2015
Bestand	Functioneel Ontwerp v1.0
Bedrijf	Info Support
Door	Tom Keim

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

Inhoudsopgave

1. Inleiding

Dit document beschrijft het Functioneel Ontwerp van het Proof-of-Concept van de Knowledge Service. Het Proof-of-Concept heeft als doel aan te tonen dat MongoDB voor knowNow een werkend alternatief is.

Samen met de opdrachtgever zijn requirements opgezet die moeten functioneren wil MongoDB inzetbaar zijn voor knowNow. De functionaliteiten zijn gebaseerd op de huidige functionaliteit van knowNow. Er is gekozen voor lees. schrijf, wijzig en verwijder functionaliteit omdat MongoDB mogelijk direct invloed heeft op de werking van deze functionaliteiten. Wanneer blijkt dat deze functionaliteiten niet werken met MongoDB is het niet inzetbaar voor knowNow.

In de volgende hoofdstukken worden de functionele requirements besproken. Daarna worden verschillende Use Cases besproken aan de hand van User Stories. Als laatste wordt er een domeinmodel geschetst.

2. Knowledge Items

In dit document wordt gesproken over Knowledge Items. Knowledge Items zijn de kennisitems van knowNow. In het Proof-of-Concept wordt uitgegaan van de Knowledge Items zoals ze op dit moment al in het huidige knowNow bestaan. Voor het Proof-of-Concept vinden geen aanpassingen plaats aan de attributen van een Knowledge Item om de situatie tussen het huidige knowNow en het Proof-of-Concept zo gelijk mogelijk te houden.

De attributen van een Knowledge Item zijn als volgt:

Attribuut	**Omschrijving**
Title	De titel van het knowledge item.
Description	De omschrijving van het knowledge item.
Rating	De beoordeling van het knowledge item.
Comments	Reacties op het knowledge item met daarin de tekst van de reactie, de schrijver van de reactie en de datum van de reactie.
Tags	Tags van het knowledge item. Tags zijn trefwoorden uit het Knowledge Item. Op basis van de tags kan ook worden gezocht naar artikelen die diezelfde tag bevatten.
Authors	Een Knowledge item wordt geschreven door een auteur. De gebruiker kan zelf opgeven wie de Auteurs en Medeauteurs (Co-authors) van het Knowledge Item zijn.
Followers	Het aantal mensen dat een knowledge item volgt. Wanneer een gebruiker een knowledge item volgt, krijgt deze gebruiker updates wanneer het artikel wordt gewijzigd.
Communities	Communities waar het Knowledge Item onder valt. Communities kunnen bijvoorbeeld afdelingen binnen een bedrijf zijn of interessegebieden.
Views	Hoeveel mensen het Knowledge item hebben bekeken.
Date	De publicatiedatum van het Knowledge Item.
Visibility	Public of Private. Bij Public is het item voor iedere gebruiker zichtbaar, bij private is het item alleen zichtbaar voor de gebruikers.
Categorieën	De categorieën waaronder het Knowledge Item valt, de huidige categorieën zijn als volgt: • Article • Concept • Practice • Presentation • Sample • Standard & Guidline • Template • Tool De categorieën zijn configureerbaar in de applicatie.

3. Functionele Requirements

De tabel hieronder bevat de Functionele Software Requirement voor het Proof-of-Concept. Alle requirements, waaronder de technische requirements, zijn te vinden in het Requirementsrapport (Bijlage D).

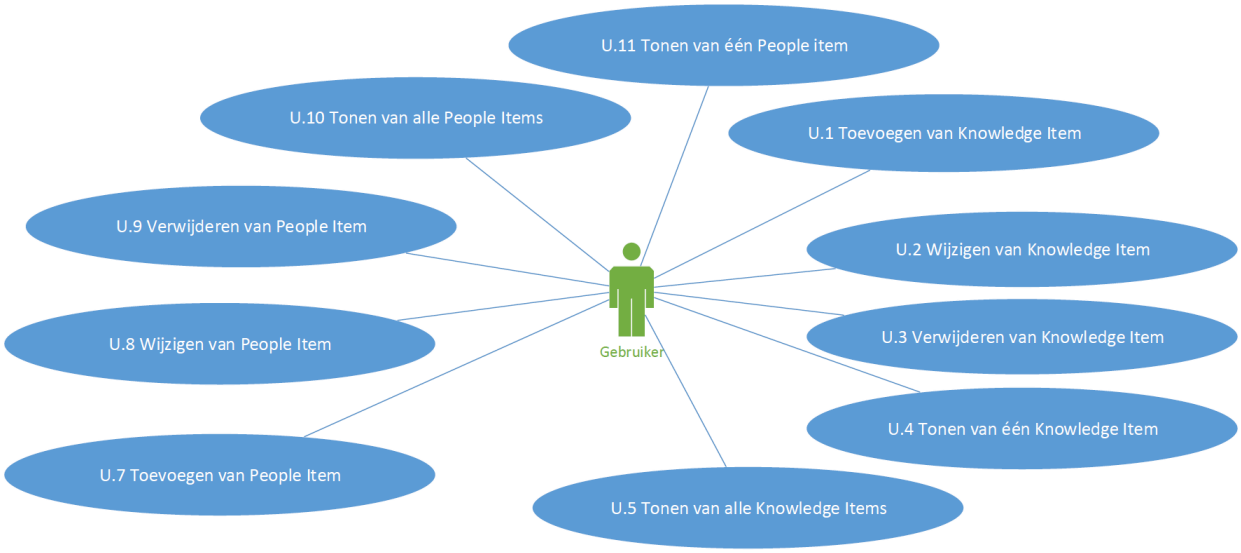
ID	Requirement
R.S4	De gebruiker moet een Knowledge Item kunnen toevoegen
R.S5	De gebruiker moet een Knowledge Item kunnen wijzigen
R.S6	De Knowledge Service moet Knowledge Items kunnen verwijderen
R.S12	De gebruiker kan een Knowledge Items opvragen aan de hand van een ID
R.S25	De gebruiker kan een lijst opvragen met alle Knowledge Items
R.S28	De gebruiker moet People Items kunnen toevoegen
R.S29	De gebruiker moet People Items kunnen wijzigen
R.S30	De gebruiker moet People items kunnen verwijderen
R.S31	De gebruiker moet alle People Items kunnen opvragen
R.S32	De gebruiker moet één People item kunnen opvragen aan de hand van een ID

4. Use Cases

4.1 Inleiding

Het Proof-of-Concept kent de volgende Use Cases. De requirements met een Won't Have-prioritering zijn weggelaten omdat deze voorlopig niet gerealiseerd worden.

ID	Use Case	Actor	Requirement
U.1	Toevoegen van Knowledge Item	Gebruiker	R.S4
U.2	Wijzigen van Knowledge Item	Gebruiker	R.S5
U.3	Verwijderen van Knowledge Item	Gebruiker	R.S6
U.4	Tonen van één Knowledge Item	Gebruiker	R.S12
U.5	Tonen van alle Knowledge Items	Gebruiker	R.S25
U.7	Toevoegen van People Item	Gebruiker	R.S28
U.8	Wijzigen van People Item	Gebruiker	R.S29
U.9	Verwijderen van People Item	Gebruiker	R.S30
U.10	Tonen van alle People Items	Gebruiker	R.S31
U.11	Tonen van één People Item	Gebruiker	R.S32



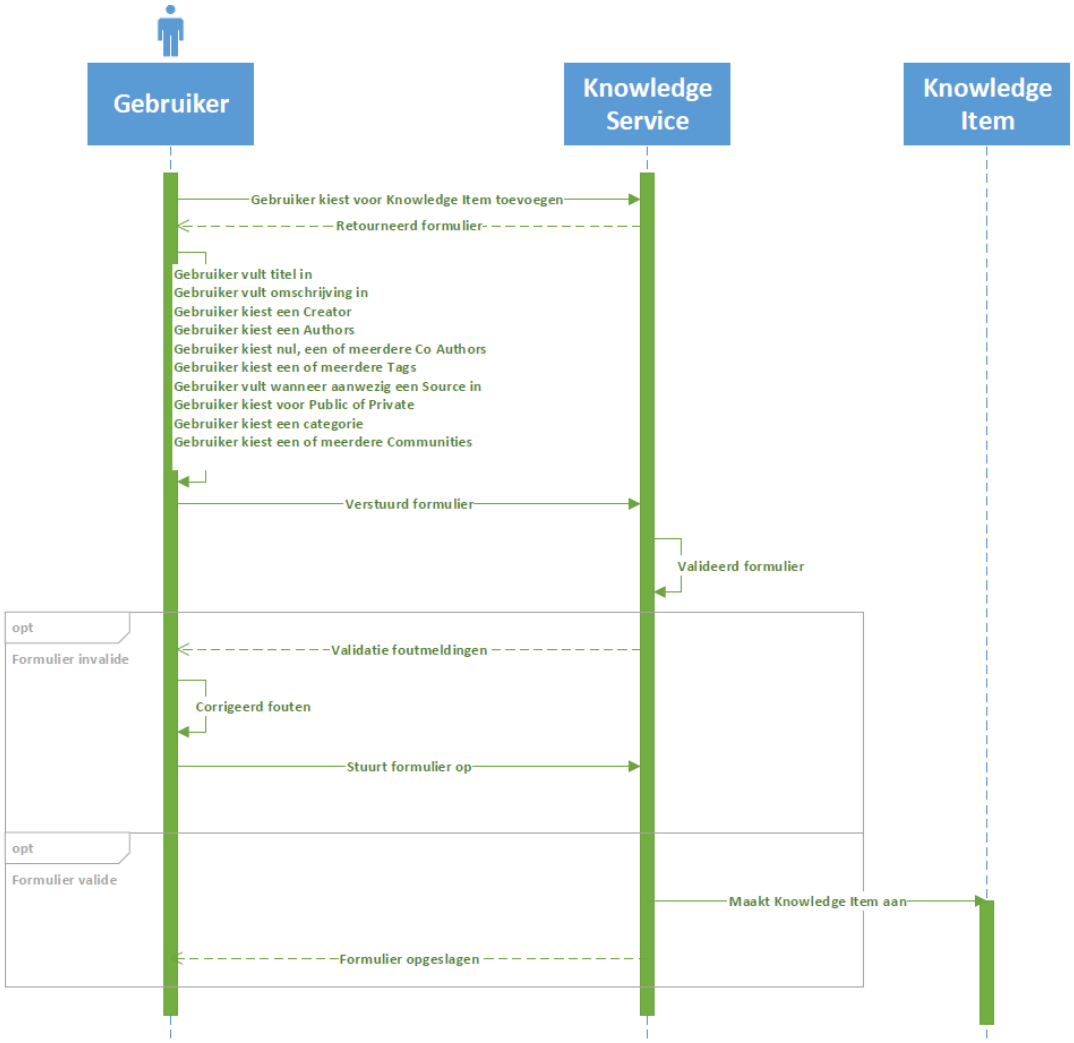
Figuur 4-1. Het Use Case Diagram

4.2 U.1 Toevoegen van Knowledge Item

4.2.1 User Story

Naam	Toevoegen van Knowledge Item
ID	U.1
Omschrijving	Een gebruiker voegt een nieuw Knowledge Item toe
Primaire actors	Gebruiker
Secondaire actors	Geen
Pre-condities	Geen
Main flow	1. Gebruiker klikt op knop om Knowledge Item toe te voegen 2. Knowledge Service retourneert formulier 3. Gebruiker vult titel in 4. Gebruiker vult omschrijving in 5. Gebruiker kiest een Creator uit een lijst met Persons uit de People Service 6. Gebruiker kiest een Author uit een lijst met Persons uit de People Service 7. Gebruiker kiest nul, een of meerdere Co Authors uit een lijst met Persons uit de People Service 8. Gebruiker kiest een of meerdere Tags 9. Gebruiker vult wanneer aanwezig een Source in 10. Gebruiker kiest voor Public of Private 11. Gebruiker kiest een categorie 12. Gebruiker kiest een of meerdere Communities 13. Gebruiker verstuurd formulier 14. De Knowledge Service controleert of alle velden zijn ingevuld [1] 15. De Knowledge Service maakt nieuw Knowledge Item aan.
Post-condities	Er is een nieuwe Knowledge Item aangemaakt.
Alternatieve flow	[1] Formulier is invalide, Knowledge Service geeft foutmelding terug. Gebruiker kan formulier wijzigen en opnieuw verzenden.

4.2.2 Sequence diagram

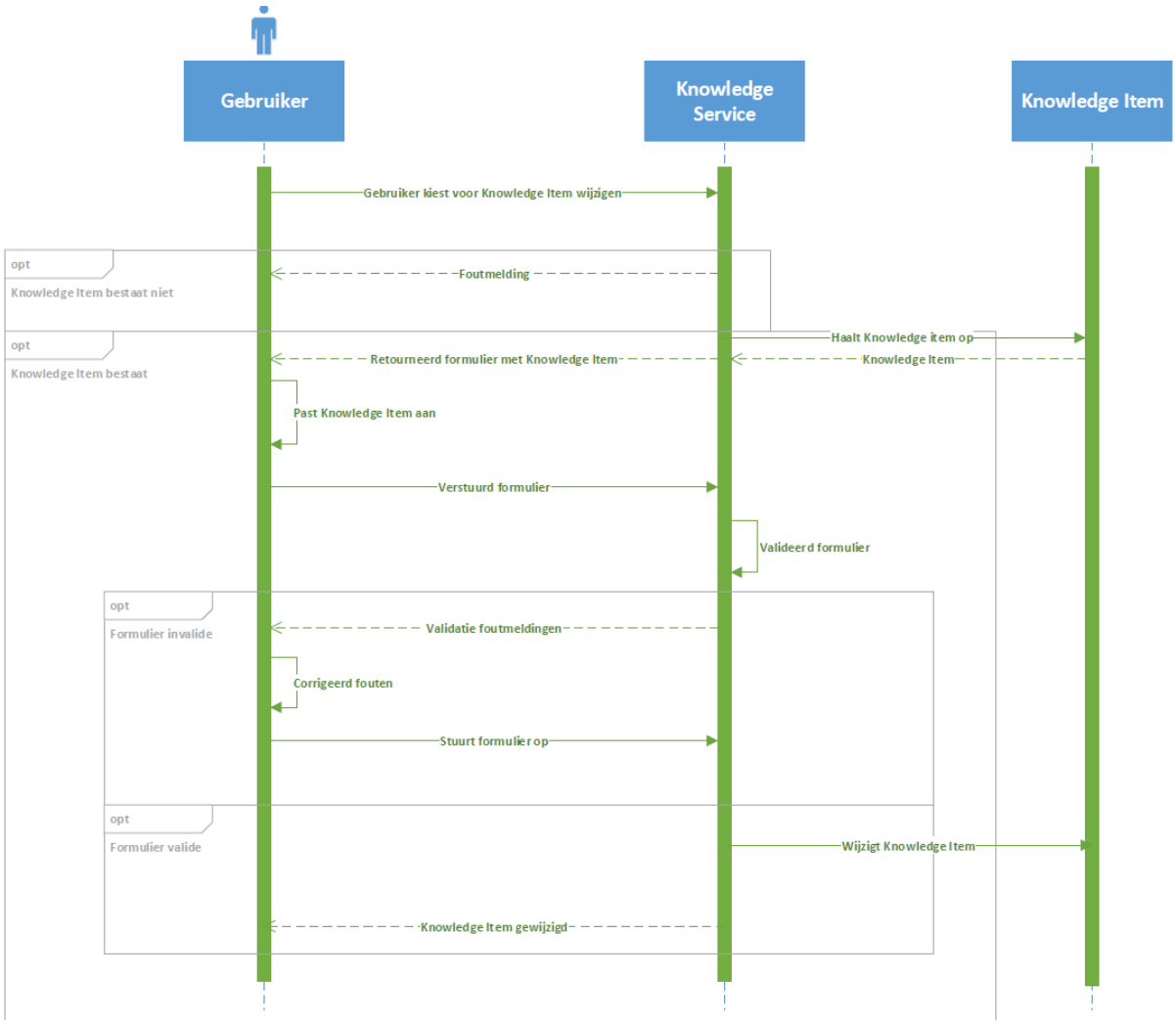


4.3 U.2 Wijzigen van Knowledge Item

4.3.1 User Story

Naam	Wijzigen van Knowledge Item
ID	U.2
Omschrijving	Een gebruiker wijzigt een Knowledge Item
Primaire actors	Gebruiker
Secondaire actors	Geen
Pre-condities	Geen
Main flow	1. Gebruiker klikt op knop om een Knowledge Item aan te passen 2. Knowledge Service controleert of Knowledge Item bestaat en niet verwijderd is[1]. 3. Knowledge Service retourneert formulier met Knowledge Item 4. Gebruiker wijzigt de velden in het formulier 5. Gebruiker stuurt gewijzigde formulier op 6. De Knowledge Service controleert of alle velden zijn ingevuld[2]. 7. Knowledge Service wijzigt het Knowledge Item.
Post-condities	Het gewijzigde Knowledge Item is gewijzigd
Alternatieve flow	[1] Het Knowledge Item bestaat niet of is verwijderd, Knowledge Service geeft foutmelding [2] Knowledge Item is onjuist, Knowledge Service geeft foutmelding terug.

4.3.2 Sequence diagram

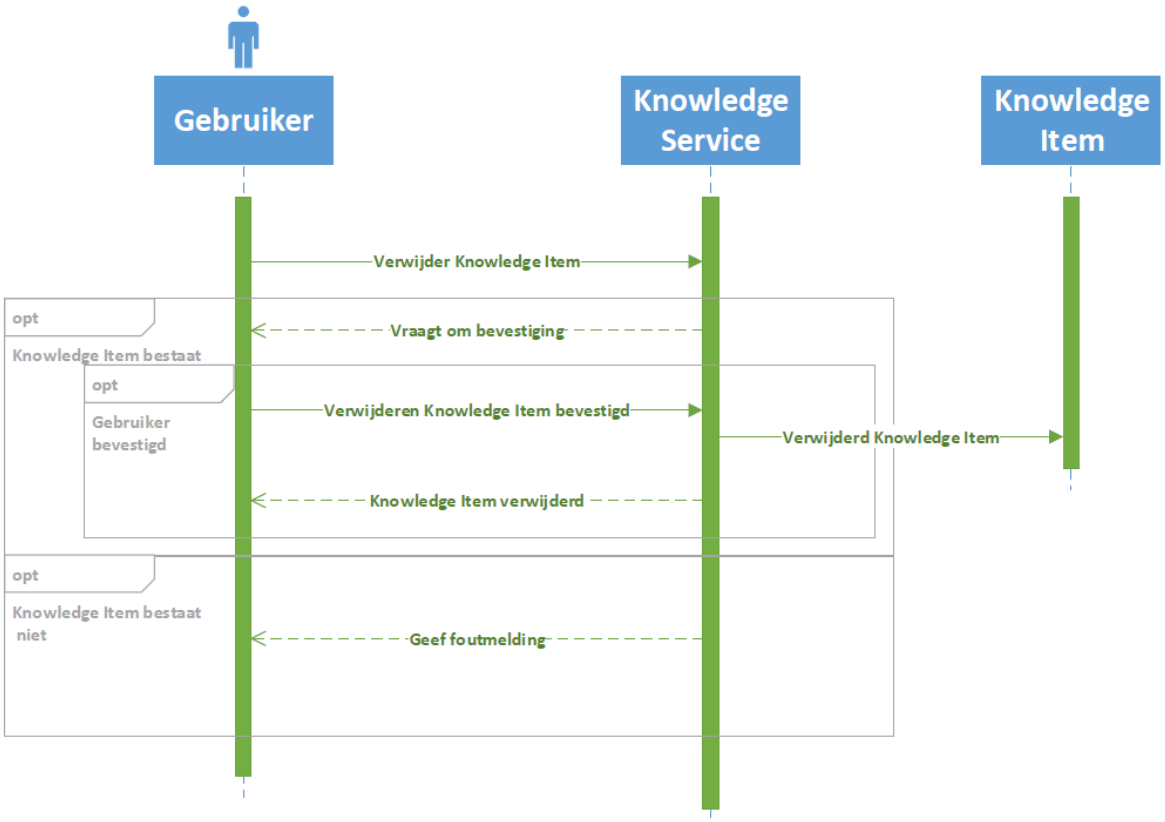


4.4 U.3 Verwijderen van Knowledge Item

4.4.1 User Story

Naam	Verwijderen van Knowledge Item
ID	U.3
Omschrijving	Een gebruiker verwijdert een Knowledge Item
Primaire actors	Gebruiker
Secondaire actors	Geen
Pre-condities	Geen
Main flow	1. Gebruiker klikt op knop voor verwijderen van Knowlege Item. 2. Knowledge Service controleert of Knowledge Item bestaat en niet al verwijderd is[1]. 3. Knowledge Service vraagt gebruiker om bevestiging. 4. Gebruiker bevestigd[2]. 5. Knowledge Service slaat het Knowledge Item op als verwijderd. 6. Gebruiker keert terug naar overzicht pagina.
Post-condities	Het Knowledge Item is als verwijderd opgeslagen
Alternatieve flow	[1] Het Knowledge Item bestaat niet of is al verwijderd, Knowledge Service geeft foutmelding [2] Gebruiker bevestigd niet. Gebruiker keert terug naar overzicht pagina en Knowledge Item is niet verwijderd.

4.4.2 Sequence diagram

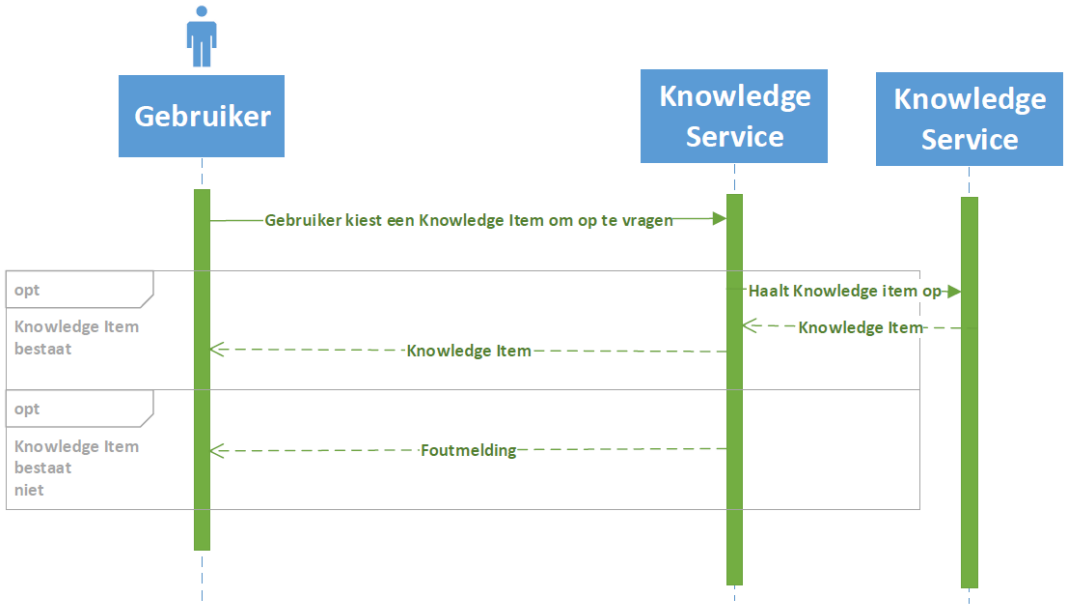


4.5 U.4 Tonen van één Knowledge Item

4.5.1 User Story

Naam	Tonen van één Knowledge Item
ID	U.4
Omschrijving	Een gebruiker vraagt een specifiek Knowledge Item op
Primaire actors	Gebruiker
Secondaire actors	Geen
Pre-condities	Geen
Main flow	1. Gebruiker klikt op een Knowledge Item om hem op te vragen. 2. De Knowledge Service controleert of het Knowledge Item bestaat en niet verwijderd is[1]. 3. De Knowledge Service retourneert het Knowledge Item.
Post-condities	De gebruiker heeft het Knowledge Item ontvangen.
Alternatieve flow	[1] Het Knowledge Item bestaat niet, Knowledge Service geeft foutmelding.

4.5.2 Sequence Diagram

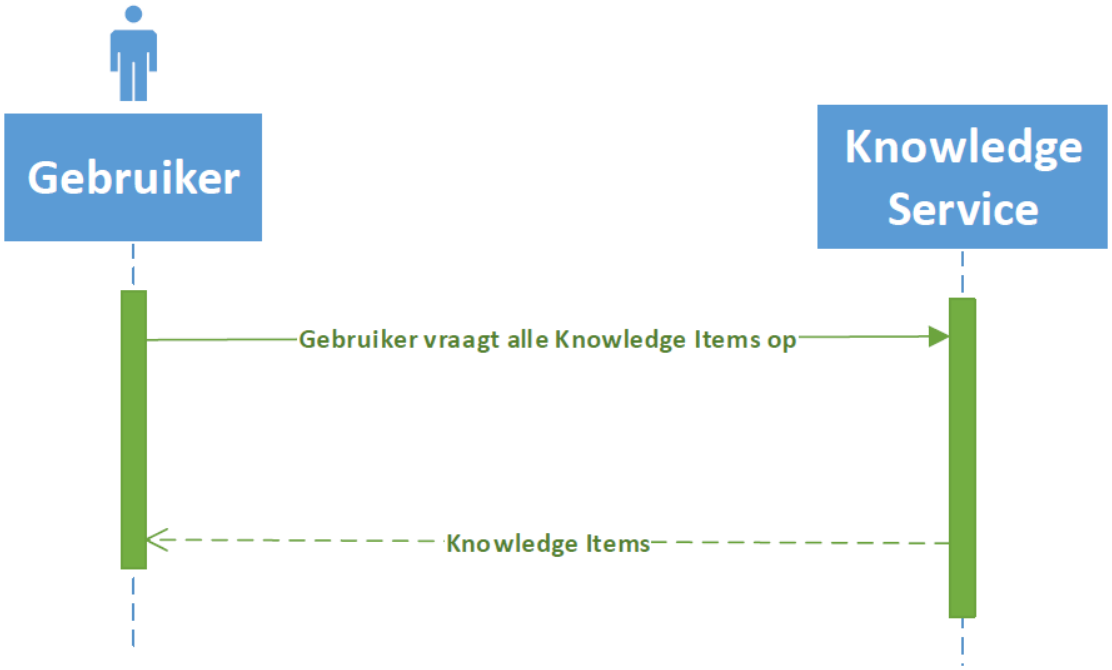


4.6 U.5 Tonen van alle Knowledge Items

4.6.1 User Story

Naam	Tonen van alle Knowledge Items
ID	U.5
Omschrijving	Toont alle Knowledge Items in een tabel.
Primaire actors	Gebruiker
Secondaire actors	Geen
Pre-condities	Geen
Main flow	1. Gebruiker vraagt alle Knowledge Items op. 2. Knowledge Service retourneert Knowledge Items.
Post-condities	De gebruiker heeft een tabel met alle Knowledge Items ontvangen.
Alternatieve flow	

4.6.2 Sequence Diagram

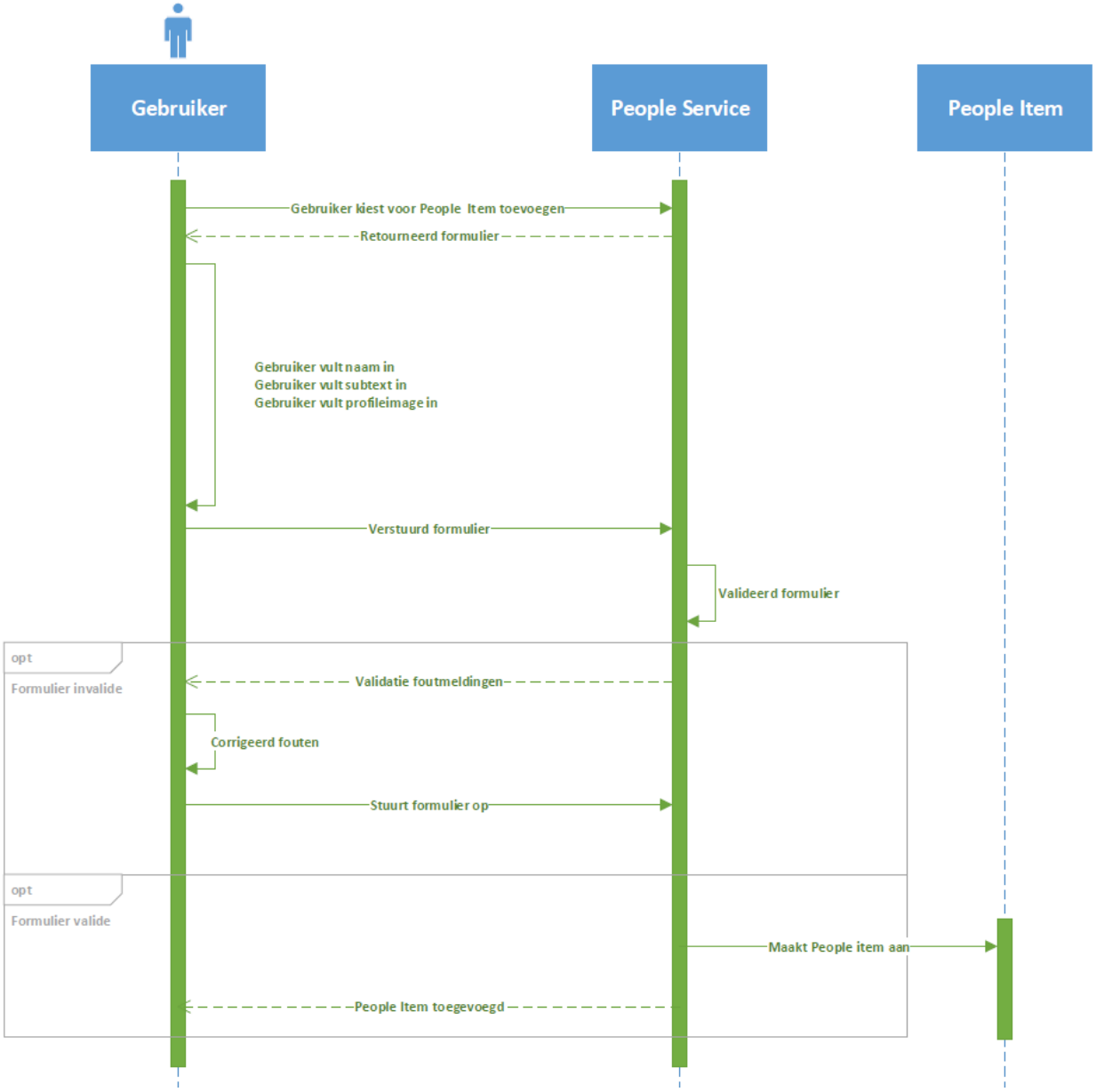


4.7 U.7 Toevoegen van een People Item

4.7.1 User Story

Naam	Toevoegen van People Item
ID	U.7
Omschrijving	Een gebruiker voegt een nieuw People Item toe
Primaire actors	Gebruiker
Secondaire actors	Geen
Pre-condities	Geen
Main flow	1. Gebruiker klikt op knop om People Item toe te voegen. 2. Gebruiker vult naam in. 3. Gebruiker vult subtext in. 4. Gebruiker vult profileimage in. 5. Gebruiker verstuurd formulier. 6. De People Service controleert of alle velden zijn ingevuld [1]. 7. De People Service maakt een People item aan.
Post-condities	Het People Item is toegevoegd.
Alternatieve flow	[1] Formulier is invalide, People Service geeft foutmelding terug. Gebruiker kan formulier wijzigen en opnieuw verwijderen

4.7.2 Sequence Diagram

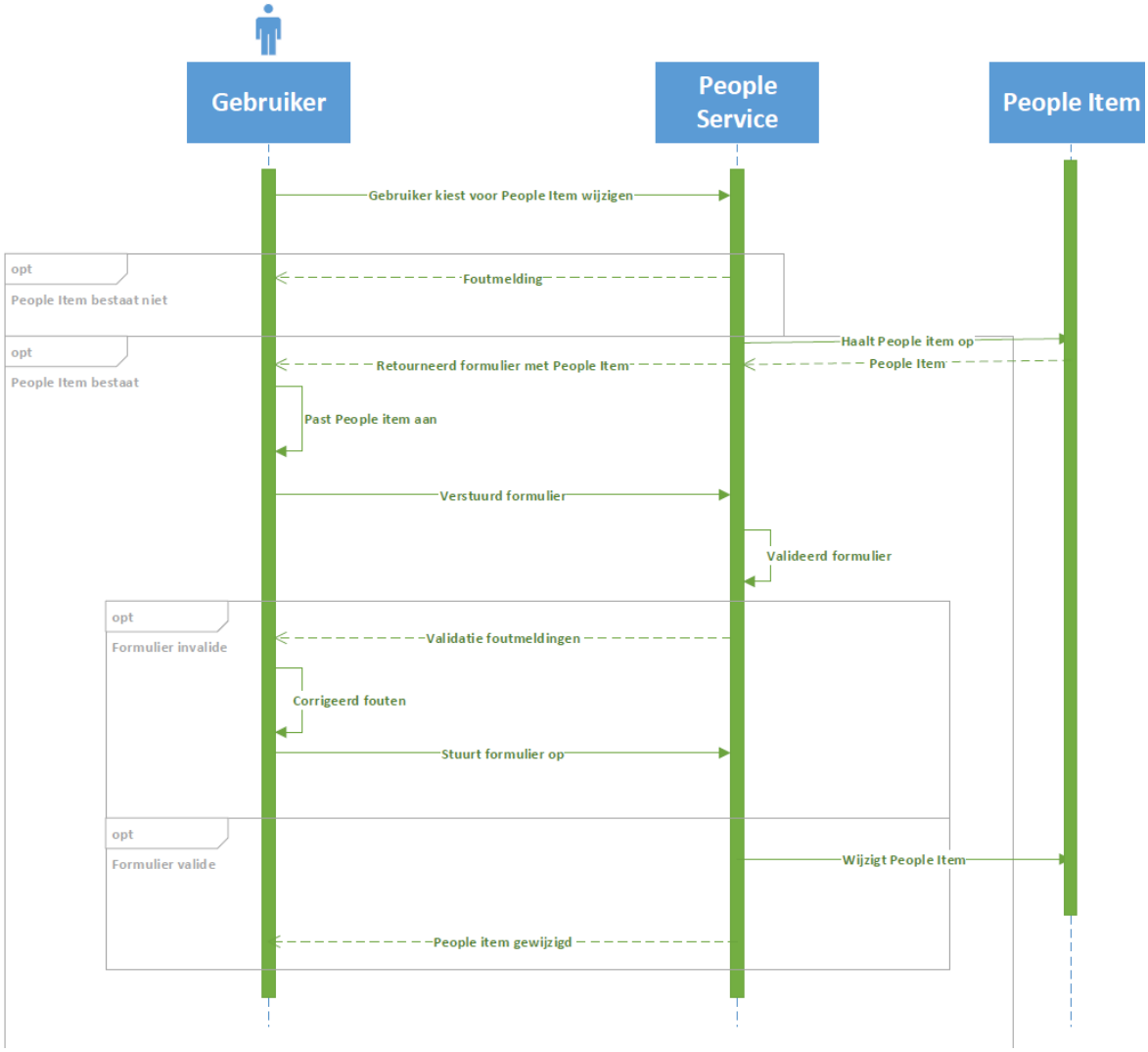


4.8 U.2 Wijzigen van People Item

4.8.1 User Story

Naam	Wijzigen van People Item
ID	U.8
Omschrijving	Een gebruiker wijzigt een People Item
Primaire actors	Gebruiker
Secondaire actors	Geen
Pre-condities	Geen
Main flow	1. Gebruiker klikt op knop om People Item aan te passen. 2. People Service controleert of People Item bestaat en niet verwijderd is[1]. 3. Gebruiker wijzigt de velden in het formulier. 4. Gebruiker stuurt gewijzigde formulier op. 5. De People Service controleert het formulier op geldigheid van data[2]. 6. De People Service wijzigt het People Item.
Post-condities	Het gewijzigde People Item is opgeslagen
Alternatieve flow	[1] Het People Item bestaat niet of is verwijderd, People Service geeft foutmelding. [2] People Item is onjuist, People Service geeft foutmelding terug.

4.8.2 Sequence diagram

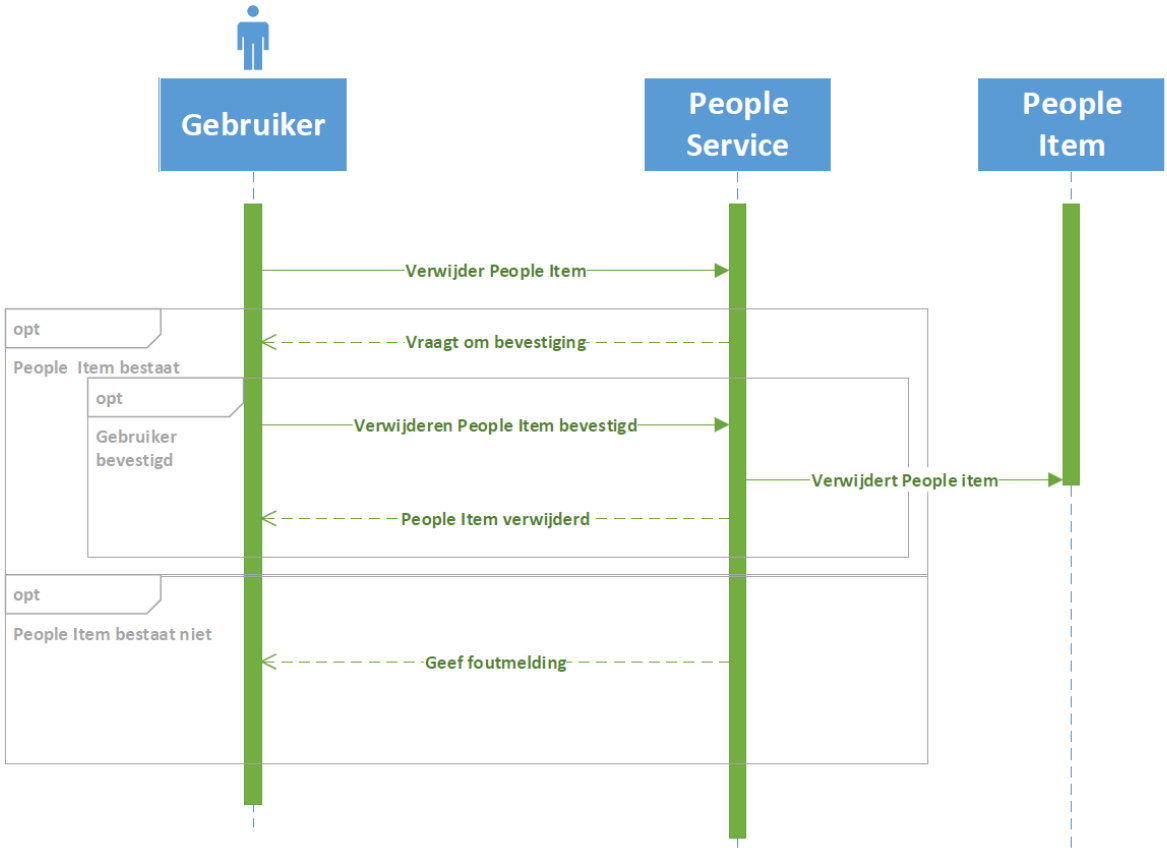


4.9 U.9 Verwijderen van People Item

4.9.1 User Story

Naam	Verwijderen van People Item
ID	U.9
Omschrijving	Een gebruiker verwijderd een People Item
Primaire actors	Gebruiker
Secondaire actors	Geen
Pre-condities	Geen
Main flow	1. Gebruiker klikt op knop om People Item te verwijderen. 2. People Service controleert of People Item bestaat en niet al verwijderd is[1]. 3. People Service vraagt gebruiker om bevestiging 4. Gebruiker bevestigd[2] 5. People Service verwijdert het People Item. 6. Gebruiker keert terug naar overzicht pagina.
Post-condities	Het People Item is verwijderd.
Alternatieve flow	[1] Het People Item bestaat niet of is al verwijderd, People Service geeft foutmelding. [2] People bevestigd niet. Gebruiker keert terug naar overzicht pagina en People Item is niet verwijderd.

4.9.2 Sequence diagram

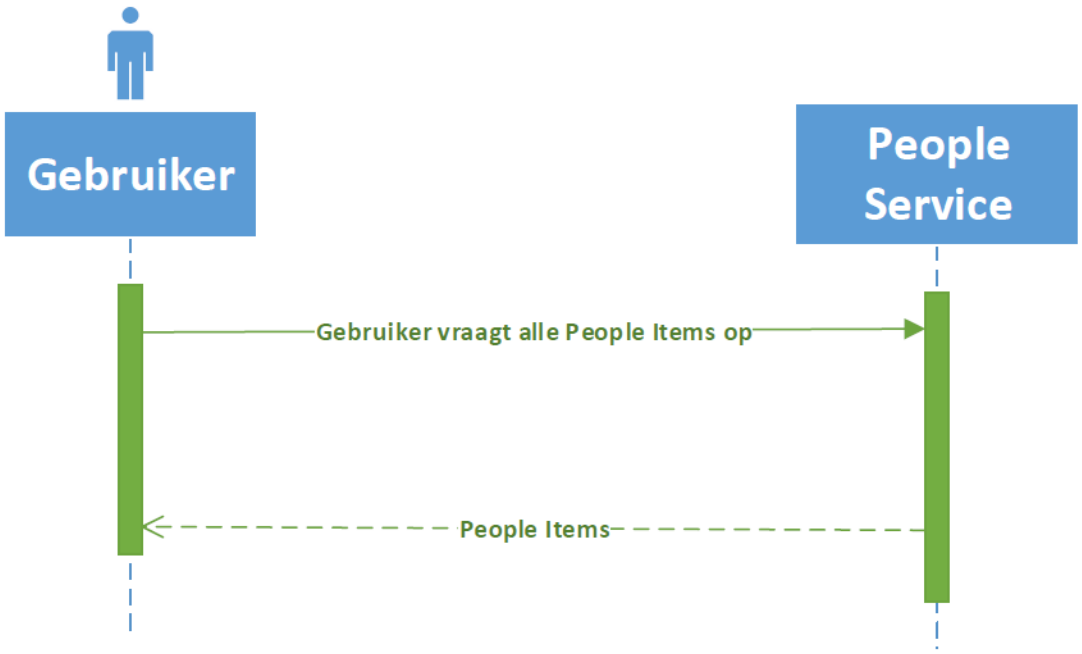


4.10 U.10 Tonen van alle People items

4.10.1 User Story

Naam	Tonen van alle People Items
ID	U.10
Omschrijving	Toont alle People Items in een tabel.
Primaire actors	Gebruiker
Secondaire actors	Geen
Pre-condities	Geen
Main flow	1. Gebruiker vraagt alle People Items op. 2. People Service retourneert People Items.
Post-condities	De gebruiker heeft een tabel met alle People Items ontvangen.
Alternatieve flow	

4.10.2 Sequence Diagram

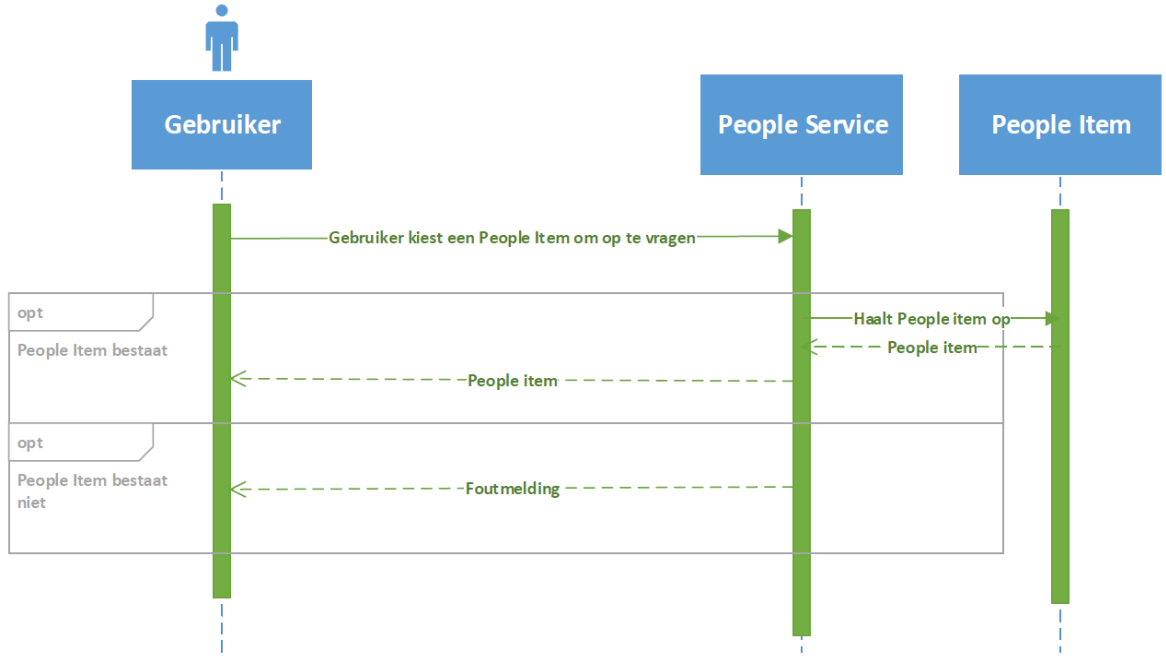


4.11 U.11 Tonen van één People Item

4.11.1 User Story

Naam	Tonen van één People Item
ID	U.11
Omschrijving	Een gebruiker vraagt een People Item op
Primaire actors	Gebruiker
Secondaire actors	Geen
Pre-condities	Geen
Main flow	1. Gebruiker klikt op knop om een People Item op te vragen. 2. De People Service controleert of het People Item bestaat en niet verwijderd is [1]. 3. De People Service retourneert het People Item.
Post-condities	De gebruiker heeft het People Item ontvangen.
Alternatieve flow	[1] Het People Item bestaat niet, People Service geeft foutmelding.

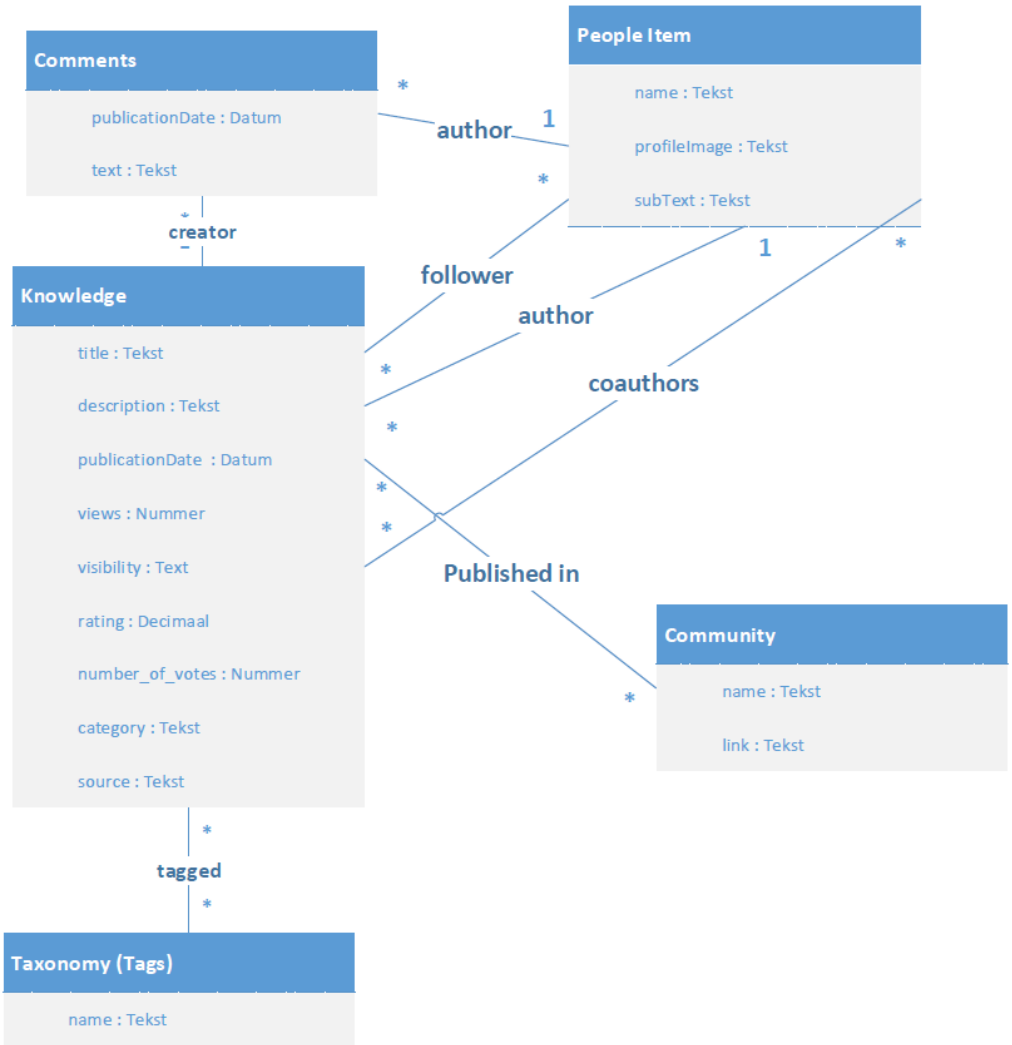
4.11.2 Sequence Diagram



5. Domeinmodel

In het figuur hieronder is het domeinmodel te vinden. Het domeinmodel is gebaseerd op de Knowledge Items zoals beschreven in Hoofdstuk 2 en de People Items. Er zijn geen nieuwe velden toegevoegd voor het Proof-of-Concept. De velden en functionele datatypen zijn afgeleid van het model van de huidige knowNow applicatie en is op sommige gebieden versimpeld. Zo bevat het People item alleen enkele velden om aan te duiden dat de hierboven beschreven Use Cases werken, maar zijn niet alle velden van een People Item meegenomen omdat deze niet benodigd zijn om het concept te kunnen bewijzen.

De comments en followers zijn wel benoemd in het domeinmodel maar zijn nog niet gebruikt in de Use Cases. Het model is wel voorbereid op de komst hiervan.



DE HAAGSE HOGESCHOOL

Info Support Solid Innovator

Bijlage G – Technisch Ontwerp

MongoDB: Een geschikte database voor het kennismanagementsysteem knowNow?

Technisch ontwerp

Titel	Technisch ontwerp
Project/Onderwerp	MongoDB: Een geschikte database voor het kennismanagementsysteem knowNow?
Versie	1.0
Status	Definitief
Datum	01-10-2015
Bestand	Technisch ontwerp v1.0
Bedrijf	Info Support
Door	Tom Keim

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

Inhoudsopgave

1. Inleiding

Dit document beschrijft het Technisch Ontwerp van het Proof-of-Concept van de Knowledge Service. Het Proof-of-Concept heeft als doel aan te tonen dat MongoDB voor knowNow een werkend alternatief is in een Microservice architectuur. De architectuur van het Proof-of-Concept is een Microservice architectuur omdat dit als beste mogelijke architectuur naar voren kwam uit het onderzoek (Onderzoeksrapport, paragraaf 3.2).

Er kon nog niet worden besloten welk documentschema het beste gehanteerd kan worden voor knowNow. Er zijn 2 mogelijke varianten. De eerste variant heeft een hiërarchisch documentschema, waarbij alle data van relaties is opgenomen als subdocument. De andere variant heeft een relationeel documentschema waarbij alleen wordt verwezen naar de relaties.

Bij het hiërarchisch documentschema is alle data weliswaar direct aanwezig, maar moet er wel extra functionaliteit worden ontwikkeld die ervoor zorgt dat data consistent wordt. Die functionaliteit zorgt mogelijk voor performanceproblemen. Bij het relationele model hoeft de data na een wijziging niet consistent gemaakt te worden, maar moet bij het ophalen van documenten wel door de applicatie data worden opgehaald van meerdere plekken. De data wordt daarna gecombineerd en dat gaat mogelijk ten koste van de snelheid van het ophalen van een item.

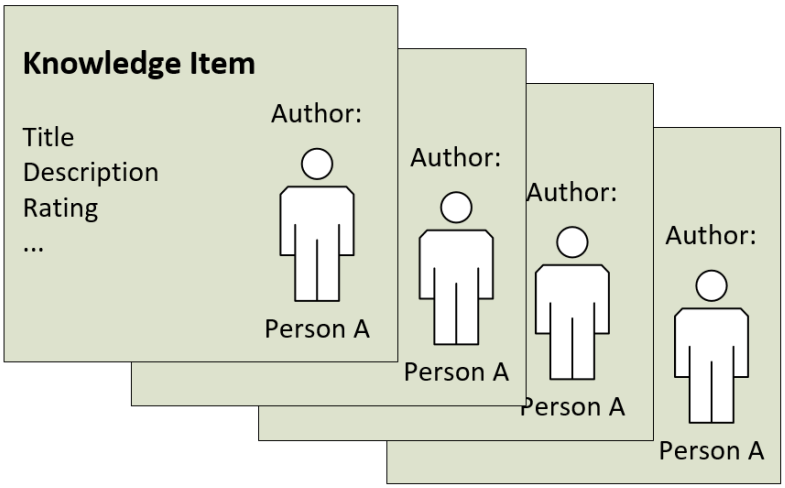
Na de ontwikkeling van het Proof-of-Concept kan door een uit te voeren performanceonderzoek tussen beide varianten geadviseerd worden welke van deze 2 verschillende documentschema's voor knowNow de betere oplossing is en kan worden aangetoond of MongoDB een alternatief is voor SQL Server.

In dit document wordt onderscheid gemaakt tussen een implementatie gebruikt voor het Proof-of-Concept met relationeel schema, en het Proof-of-Concept met het hierarschisch schema. Dit wordt dan vermeld in de naam van de paragraaf.

In de gekozen hoofdstukken wordt de gekozen architectuur, de verschillende componenten, de gebruikte technieken en het documentschema van beide varianten van het Proof-of-Concept besproken.

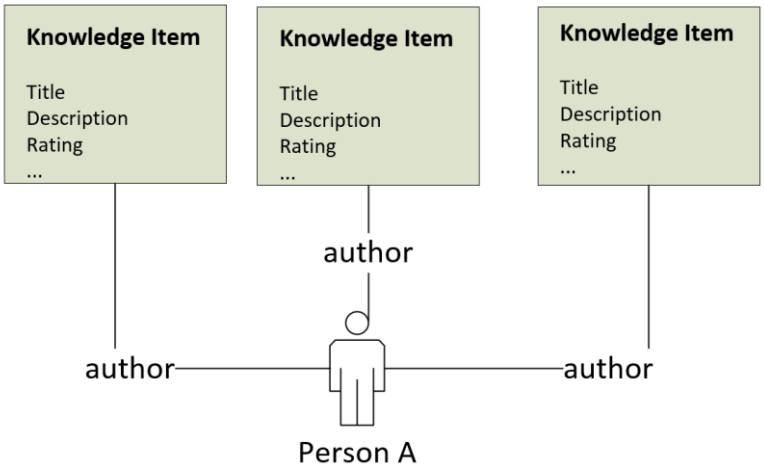
2. Varianten Proof-of-Concept

Er zijn 2 varianten ontworpen van het Proof-of-Concept. De eerste variant bevat een documentschema waarbij alle data van relaties is opgenomen. Zoals bijvoorbeeld de naam van een auteur. Bij deze variant kan een Knowledge Item pagina in één keer worden opgevraagd vanuit MongoDB en hoeft er niet gejoined te worden.



Figuur 2-1. Een documentschema waarbij data van relaties is opgenomen in een subdocument (Hiërarchisch documentschema)

Bij de tweede variant wordt er in het documentschema alleen gerefereerd naar de andere relaties. Hierdoor heeft het documentschema bijvoorbeeld alleen de unieke code van een auteur, en niet alle data.



Figuur 2-2. Een documentschema waarbij er alleen gerefereerd wordt naar de relaties (Relationeel documentschema)

Aan de hand van een performance onderzoek die beide resultaten vergelijkt zal een beslissing worden gemaakt wat de best te implementeren variant is indien de opdrachtgever ervoor kiest om gebruik te maken van MongoDB.

3. Gebruikte Technieken

Dit hoofdstuk bespreekt de gebruikte technieken voor het ontwikkelen van het Proof-of-Concept en waarom er voor deze technieken is gekozen.

3.1 MongoDB

Het Proof-of-Concept bekijkt de mogelijkheid van het gebruik van MongoDB voor knowNow. Daarom maakt het Proof-of-Concept ook gebruik van MongoDB.

MongoDB is een NoSQL document storage database[1].

3.2 C# .NET icm Web API

Er is voor C# en .NET gekozen om de aanwezige Web API in het .NET framework. Met de Web API is het mogelijk om snel een REST interface op te zetten waardoor daar minder tijd in gestoken kan worden. Daarnaast maakt het knowNow team vooral gebruik van C# wat de onderhoudbaarheid voor dat team verhoogt. Ook heeft de afstudeerder al enige ervaring met C#.

3.3 Event Bus

Het Proof-of-Concept maakt gebruikt van een Event Bus. Een Event Bus is een applicatie waar een event heen gestuurd kan worden. De Event Bus verspreid dit event vervolgens naar applicaties die gebruik maken van de Event Bus. Een event kan bijvoorbeeld na een Insert, Update of Delete transactie worden verstuurd. In een Event staat onder andere wat er veranderd is, wanneer het veranderd is en door wie de transactie is gemaakt.

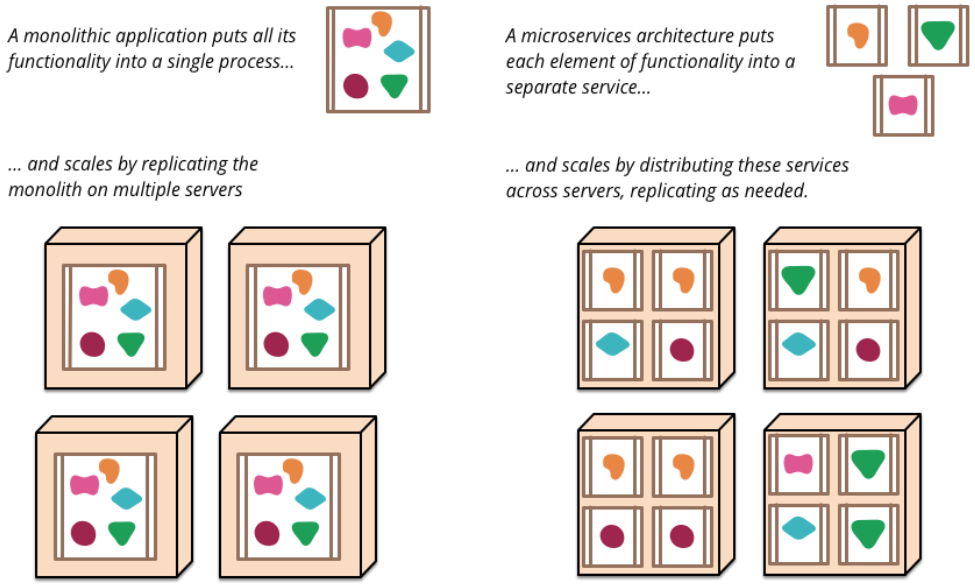
Hierdoor kan buiten de request die data worden verwerkt waardoor de gebruiker minder lang hoeft te wachten op een response. Zo wordt bijvoorbeeld via de Event Bus de Lucene.NET search engine bijgewerkt na een aanpassing in de data.

Het Proof-of-Concept maak voor de Event Bus gebruik van RabbitMQ[2] omdat dit inmiddels ook al gebruikt wordt in de huidige knowNow ervaring en hierdoor dus kennis van aanwezig is in het knowNow team.

[1] MongoDB, “Introduction”
https://www.mongodb.org/about/introduction/
[2] RabbitMQ, “Features”
https://www.rabbitmq.com/features.html

3.4 Microservices

Bij een Microservices architectuur is een applicatie opgedeeld in meerdere kleine services die elk hun eigen verantwoordelijkheid hebben[3]. Door het gebruik van microservices kan elke service apart worden vervangen en worden geschaald. Hierdoor is een microservice architectuur erg schaalbaar:



Figuur 3-1. Links de schaling van een monolithische architectuur, rechts de schaling van een Microservice architectuur

3.5 jQuery & AJAX

Het Proof-of-Concept voor het relationele documentschema maakt gebruik van AJAX[4] door middel van het jQuery framework[5]. Met AJAX is het mogelijk de browser asynchroon data op te laten halen van de applicatie. Hierdoor is het mogelijk de gebruiker ook nadat de pagina is geladen te voorzien van additionele informatie. jQuery is een Javascript library die het gebruik van AJAX eenvoudiger gemaakt. Voor jQuery is gekozen omdat de afstudeerder hier al ervaring mee had.

[3] Martin Fowler, "Microservices" http://martinfowler.com/articles/microservices.html
[4] https://developer.mozilla.org/en-US/docs/AJAX/Getting_Started
[5] https://jquery.com/

4. Architectuur

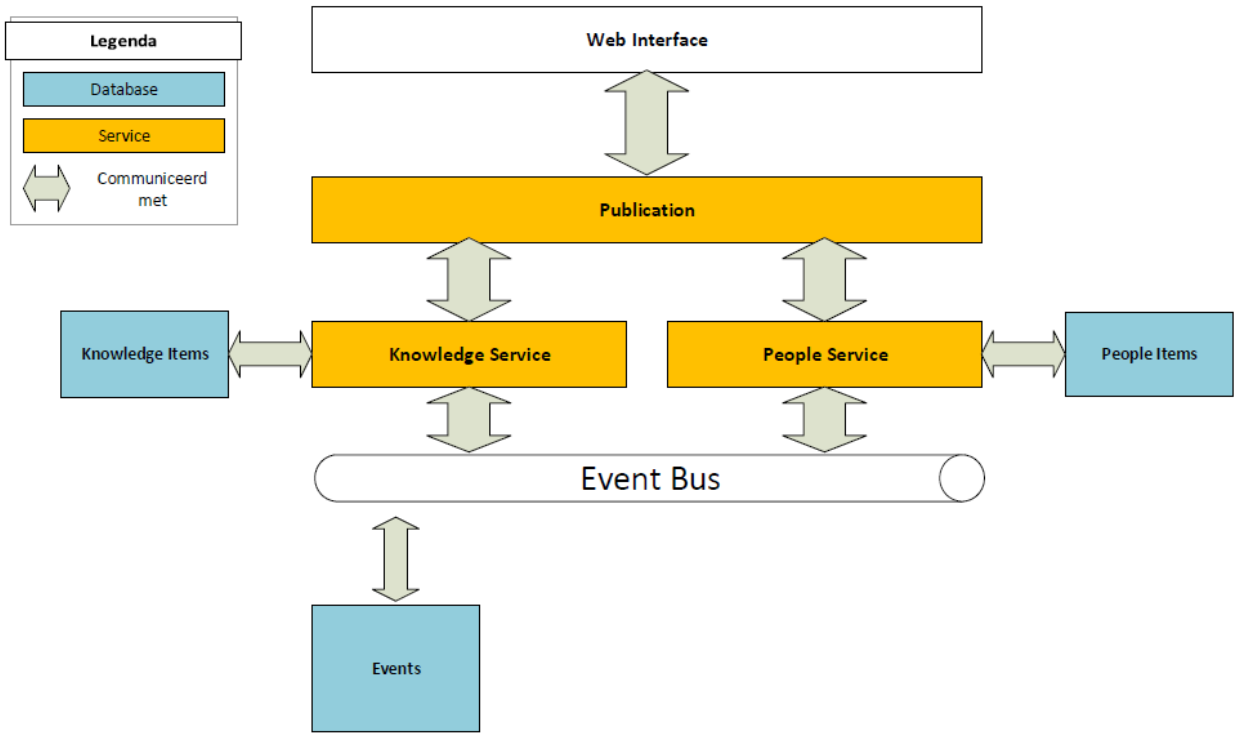
Dit hoofdstuk beschrijft de architectuur van het Proof-of-Concept. Gekozen is voor een Microservice architectuur. De achterliggende keuze voor deze architectuur is beschreven in het Onderzoek, Hoofdstuk 3.

Er wordt een deel van de microservice architectuur opgezet. Elk van de delen is een op zichzelf staande applicatie. De volgende delen van de Microservice architectuur zijn ontworpen en worden door de afstudeerder ontwikkeld. De gehele applicatie maakt geen gebruik van onderdelen van het huidige knowNow.

Elke service in een Microservice architectuur heeft zijn eigen verantwoordelijkheid. Zo is de Knowledge Service verantwoordelijk voor het ophalen, wijzigen, toevoegen en verwijderen van Knowledge Items en de People Service voor het ophalen, wijzigen, toevoegen en verwijderen van People items.

Deel	Omschrijving
Knowledge Service	De Knowledge service is een service die Knowledge Items beheerd. Met de Knowledge Service kan gecommuniceerd worden met een JSON REST API. De gebruiker communiceert niet direct met de Knowledge Service maar via de Publication Layer.
People Service	De People Service bevat de gegevens van alle personen die knowNow gebruiken (People Items). Met de Knowledge Service kan gecommuniceerd worden met een JSON REST API. De gebruiker communiceert niet direct met de People Service maar via de Publication Layer.
Publication Layer	Met de Publication Layer kan de eindgebruiker communiceren door middel van een JSON REST API. De Publication Layer communiceert weer met de Knowledge Service en People Service om de eindgebruiker te kunnen beantwoorden.
Web Interface	De Web Interface communiceert met de Publication Layer met een JSON REST API. De Web Interface is een MVC project. Dit houdt in dat de modellen, views en controllers van elkaar gescheiden zijn en de verantwoordelijkheden in de applicatie duidelijk worden. Voor het grafische ontwerp wordt gebruik gemaakt van Bootstrap zodat er snel een ontwerp op te zetten is zonder dat hier veel tijd in gaat zitten.
Event Bus	De Event Bus is een middel waarmee verschillende Services met elkaar kunnen communiceren. De Event Bus wordt toegelicht in Hoofdstuk 3.3.

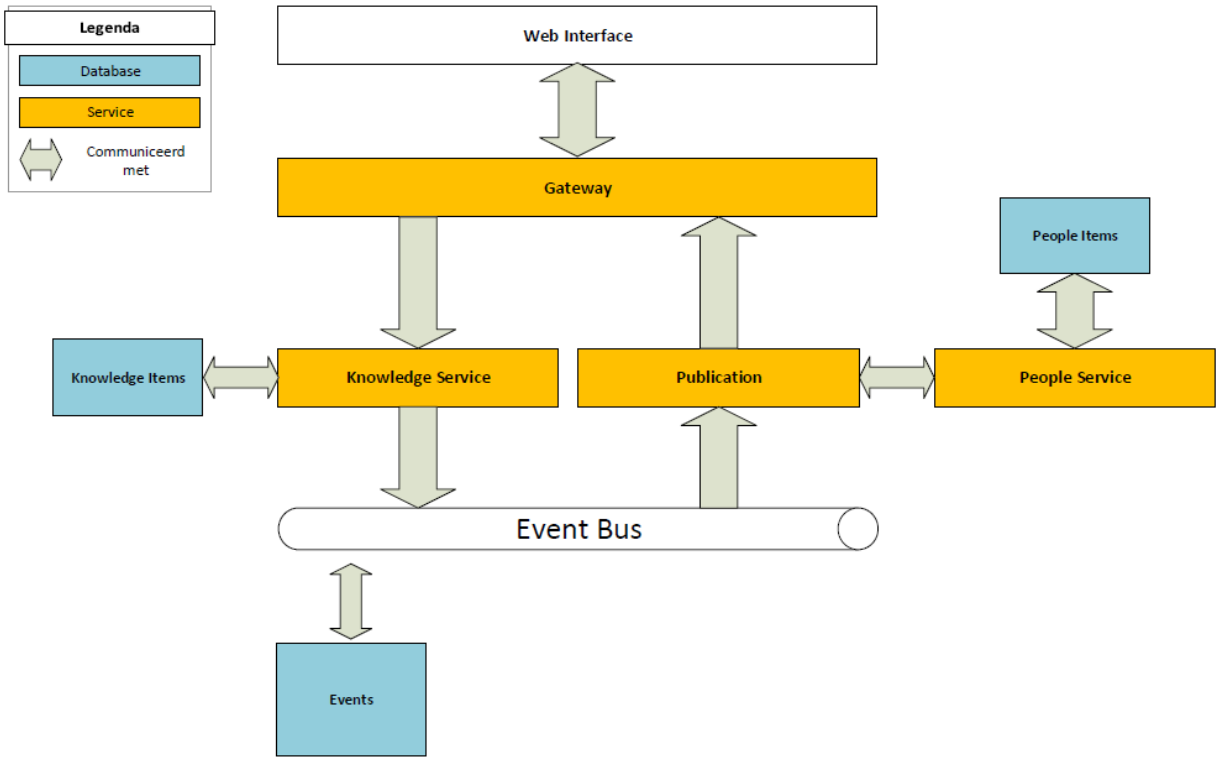
In het figuur hieronder is een overzicht van de Microservice architectuur zoals is geimplementeerd in het Proof of Concept.



Figuur 4-1. De architectuur van het Proof of Concept

In een eigenlijke Microservice architectuur hoort op de plek van de Publication Layer de Gateway te zitten. De Gateway dient er alleen voor om data door te geven. Echter in het Proof-of-Concept wordt naast het doorgeven van data ook data gecombineerd. Zo haalt deze laag bijvoorbeeld bij het toevoegen van een Knowledge Item eerst de bijhorende People Item uit de People Service op voordat deze het Knowledge Item doorzet naar de Knowledge Service. Voor deze opzet is gekozen omdat dit sneller is te implementeren dan op de correcte manier in een Microservice architectuur. Hierbij voert de gateway deze taken niet uit en moet dit door de services zelf geregeld worden.

Een microservice architectuur waar wel gebruik wordt gemaakt van een Gateway kan er als volgt uit zien:



Figuur 4-2. Een mogelijke architectuur waar er gebruik wordt gemaakt van een Gateway

In de bovenstaande architectuur kan de Gateway alleen data doorgeven aan services en ophalen van data uit services. De gateway kan de data niet muteren of combineren. In deze situatie zal bij het schrijven van een Knowledge Item naar de Knowledge Service worden gerefereerd naar de auteur. De Publication Service zal daarna het Knowledge Item met de volledig auteur opnemen door deze uit de People Service te halen.

Dit is gedaan omdat de belangrijkste reden van het ontwikkelen van het Proof of Concept is om te testen of MongoDB een optie is voor knowNow. De gekozen architectuur is een Microservice-architectuur en deze wordt deels toegepast. Echter, is dit niet het hoofdzakelijk doel van het Proof-of-Concept: er hoeft niet bewezen te worden dat de Microservice-architectuur werkt voor knowNow.

5. Componenten Proof-of-Concept

Dit hoofdstuk bevat de klassendiagrammen van het Proof-of-Concept.

Het Proof-of-Concept is opgesplitst in meerdere projecten. Hiermee worden de verantwoordelijken duidelijk gescheiden gehouden. Het Proof-of-Concept maakt geen gebruik van bestaande functionaliteiten van knowNow.

De verdeling is als volgt:

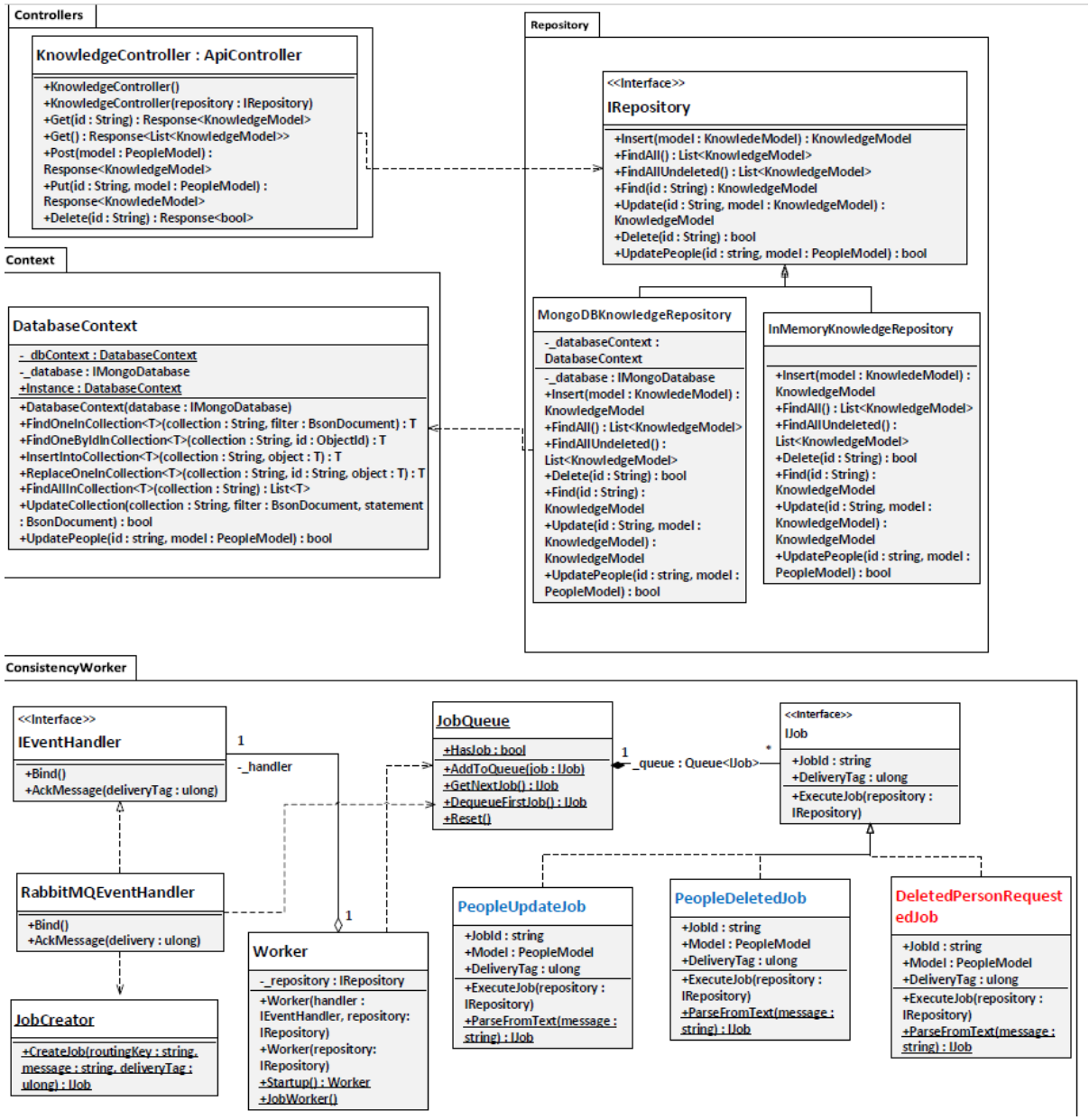
Project	Omschrijving
Knowledge Service	De Knowledge service is een C# MVC Web API project. Dit project toont en wijzigt door middel van een REST API de Knowledge Items en maakt het documentschema consistent door updates vanuit de Event Bus.
People Service	De People Service is een C# MVC Web API project. Dit project toont en wijzigt door middel van een REST API de People Items. Dit is een eenvoudige service die (een deel van de) de functionaliteit van een daadwerkelijke People Service nabootst
Publication layer	De Publication laag is een C# MVC Web API project.
Web Interface	De Web Interface is een C# MVC project die gebruik maakt van Bootstrap voor de vormgeving. De Web Interface communiceert met de Gateway via een REST API. Voor Bootstrap is gekozen omdat er dan direct een vormgeving voor de website is, waardoor hier minder tijd ingestoken hoeft te worden.
Shared Models	Het Shared Models project is een Class library. Dit project bevat de verschillende models van het Proof-of-Concept zodat deze niet in elke service apart ontwikkeld moeten worden.
APICommunicator	Het APICommunicator project is een Class library die wordt gebruikt door de Web Interface en de Gateway en bevat functionaliteit waarmee deze projecten met de andere services kunnen communiceren.

Per project wordt een klassendiagram gemaakt en wordt het doel omschreven.

5.1 Knowledge Service (Hiërarchisch & relationeel documentschema)

De Knowledge Service is verantwoordelijk voor de Knowledge Items. Via een REST API kunnen Knowledge Items worden opgehaald, gewijzigd en verwijderd. De Publication layer communiceert met de Knowledge Service. De eindgebruiker communiceert niet direct met de Knowledge Service.

In Figuur 5-1 is het klassendiagram van de Knowledge Service te vinden. In de tabel daaronder worden de verschillende klassen toegelicht. De klasse diagrammen zijn door de afstudeerder ontworpen in Microsoft Visio. De Rode klassen worden alleen gebruikt in het relationeel documentschema, de blauwe klassen alleen in het hiërarchisch documentschema.



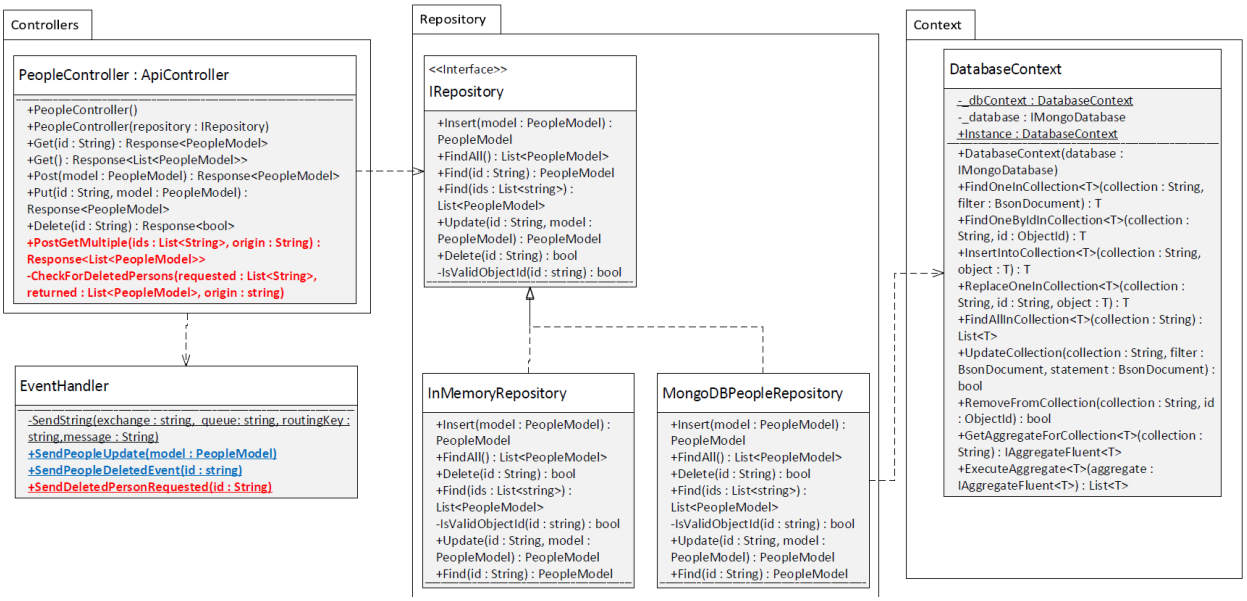
Figuur 5-1. Klassendiagram van Knowledge Service

Klassenaam	Omschrijving
KnowledgeController	De KnowledgeController zorgt voor de verwerking van alle requests via de REST API.
IRepository	Er wordt gebruikt gemaakt van een Repository Pattern[6] voor Create, Read, Update en Delete (CRUD) operaties voor de data. Hiervoor is gekozen zodat bij het testen van de applicatie een andere repository gebruikt kan worden dan bijvoorbeeld de MongoDB repository.
MongoDBKnowledgeRepository	Deze repository voert de CRUD operaties uit op een MongoDB database via de DatabaseContext.
InMemoryKnowledgeRepository	Deze repository voert de CRUD operaties uit op een lijst met Knowledge Items die (tijdelijk) wordt opgeslagen in het geheugen.
DatabaseContext	Deze klasse communiceert met MongoDB via de MongoDB C# Driver.
IEventHandler	De interface voor het afhandelen van Events vanuit een Event Bus
RabbitMQEventHandler	Deze klasse handelt event uit de RabbitMQ event bus af
JobQueue	Bevat een statische lijst met alle uit te voeren taken om de data consistent te maken
Worker	De klasse die de daadwerkelijke taken om het consistent maken van de data uit laat voeren in een thread.
IJob	Een interface voor taken die bedoeld zijn het documentschema consistent te maken
JobCreator	Een klasse die na aanleiding van de routing key (naam van event) van RabbitMQ de juiste IJob klasse aanmaakt.
PeopleUpdateJob	Een klasse die een update van een persoon afhandelt en deze doorvoert in de MongoDB database. De Job voert de update door in alle Knowledge Items met die persoon en moet dus ook alle documenten langs. Wordt alleen gebruikt in het Proof-of-concept met hiërarchisch documentschema.
PeopleDeleteJob	Een klasse die een verwijdering van een persoon afhandelt en deze doorvoert in de MongoDB database. De Job voert de verwijdering door in alle Knowledge Items met die persoon en moet dus ook alle documenten langs. Wordt alleen gebruikt in het Proof-of-concept met hiërarchisch documentschema.
DeletedPersonRequestedJob	Een klasse die een persoon verwijderd uit een Knowledge Item wanneer een relatie niet meer bestaat. Wordt verder toegelicht in de sequence diagrammen. Wordt alleen gebruikt in het Proof-of-concept met relationeel documentschema.

[6] "Microsoft Developer Network, The Repository Pattern" https://msdn.microsoft.com/en-us/library/ff649690.aspx

5.2 People Service (Hiërarchisch & relationeel documentschema)

De People Service is een C# MVC Web API project. Dit project toont en wijzigt door middel van een REST API de People Items. Dit is een eenvoudige service die (een deel van de) de functionaliteit van een daadwerkelijke People Service nabootst. De rode methode wordt alleen gebruikt in het Proof-of-Concept met het relationele documentschema. De blauwe methodes alleen in het Proof-of-Concept met het hiërarchisch documentschema.



Figuur 5-2. Klassendiagram van de People Service

Klassenaam	Omschrijving
PeopleController	De PeopleController zorgt voor de verwerking van alle requests via de REST API. De rode methode wordt alleen gebruikt in het Proof-of-Concept met het relationele documentschema.
IRepository	De Repository Interface
InMemoryKnowledgeRepository	Deze repository voert de CRUD operaties uit op een lijst met People Items die (tijdelijk) wordt opgeslagen in het geheugen.
MongoDBPeopleRepository	Deze repository voert de CRUD operaties uit op een MongoDB database via de DatabaseContext.
EventHandler	Deze geeft Events door aan RabbitMQ, de Event Bus. De mogelijke events worden omschreven in 5.2.1.

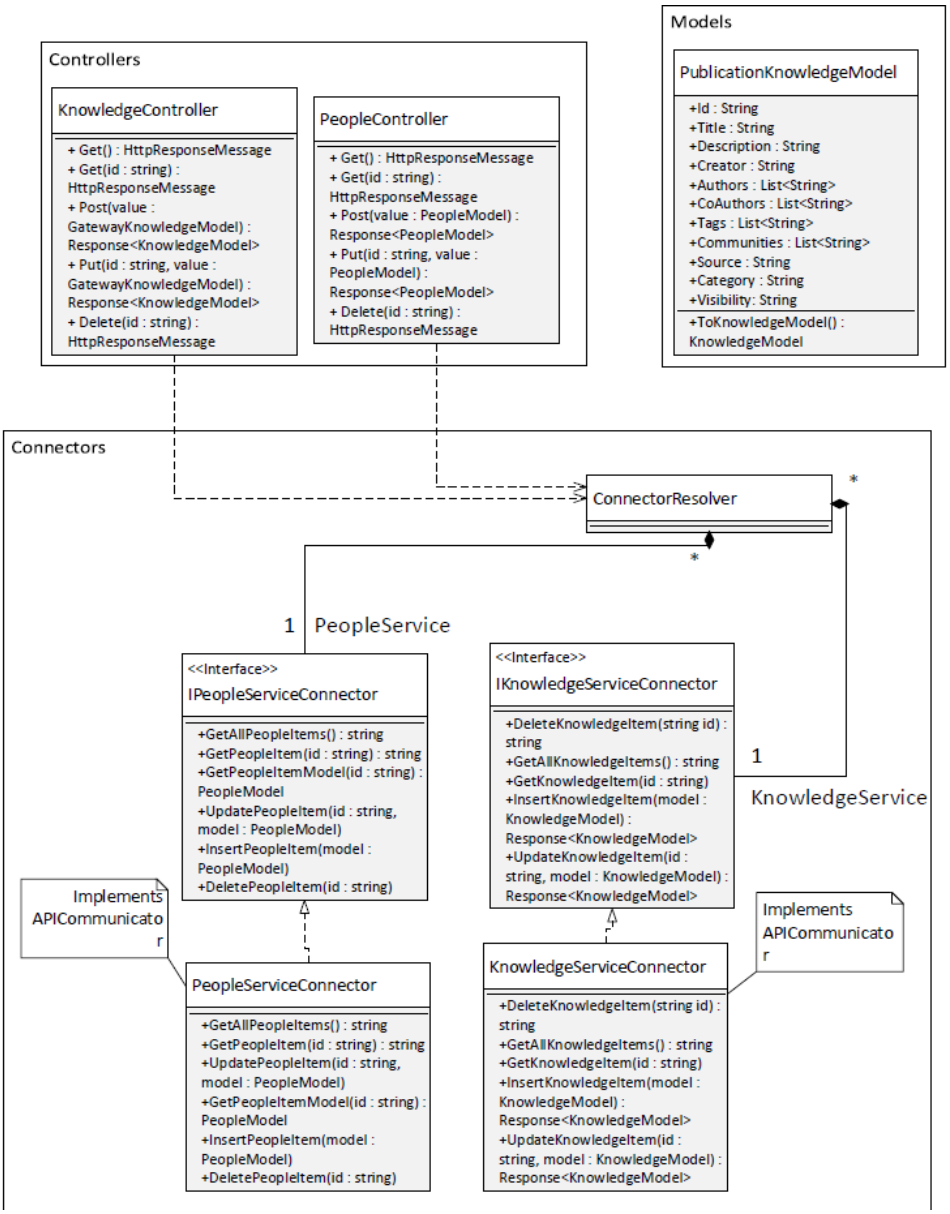
5.2.1 Events People Service

De People Service verstuurd bij enkele gebeurtenissen een Event naar de Event Bus, RabbitMQ. In de tabel hieronder worden deze events toegelicht.

Event (routing key)	Gebeurtenis
PeopleModelUpdate	Het PeopleModelUpdate event wordt naar de Event Bus verstuurd wanneer een People Item wordt gewijzigd. Het gewijzigde People Item wordt meegestuurd met het Event. Dit Event wordt alleen gebruikt in het Proof-of-Concept met hiërarchisch documentschema.
PeopleDeleted	Het PeopleDeleted event wordt naar de Event Bus verstuurd wanneer een People Item wordt verwijderd. Het id van het verwijderde People Item wordt meegestuurd met het event. Dit Event wordt alleen gebruikt in het Proof-of-Concept met hiërarchisch documentschema.
DeletedPersonRequested	Het DeletedPersonRequested event wordt naar de Event Bus verstuurd wanneer een niet gevonden People Item wordt opgevraagd. Het niet gevonden id en wie het niet gevonden People Item heeft opgevraagd wordt meegestuurd in het Event. Dit Event wordt alleen gebruikt in het Proof-of-Concept met hiërarchisch documentschema.

5.3 Publication layer (Hiërarchisch documentschema)

De Publication Layer is de applicatie waarmee de eindgebruiker via een REST API kan communiceren. Ook de Webinterface maakt gebruik van de Publication Layer. De Publication Layer zorgt dat requests worden verwerkt door de Knowledge of People Service. Bij het toevoegen van een Knowledge Item wordt daarnaast de People Service benaderd om enkele data van onder andere de auteur te verzamelen en deze mee te sturen naar de Knowledge Service.



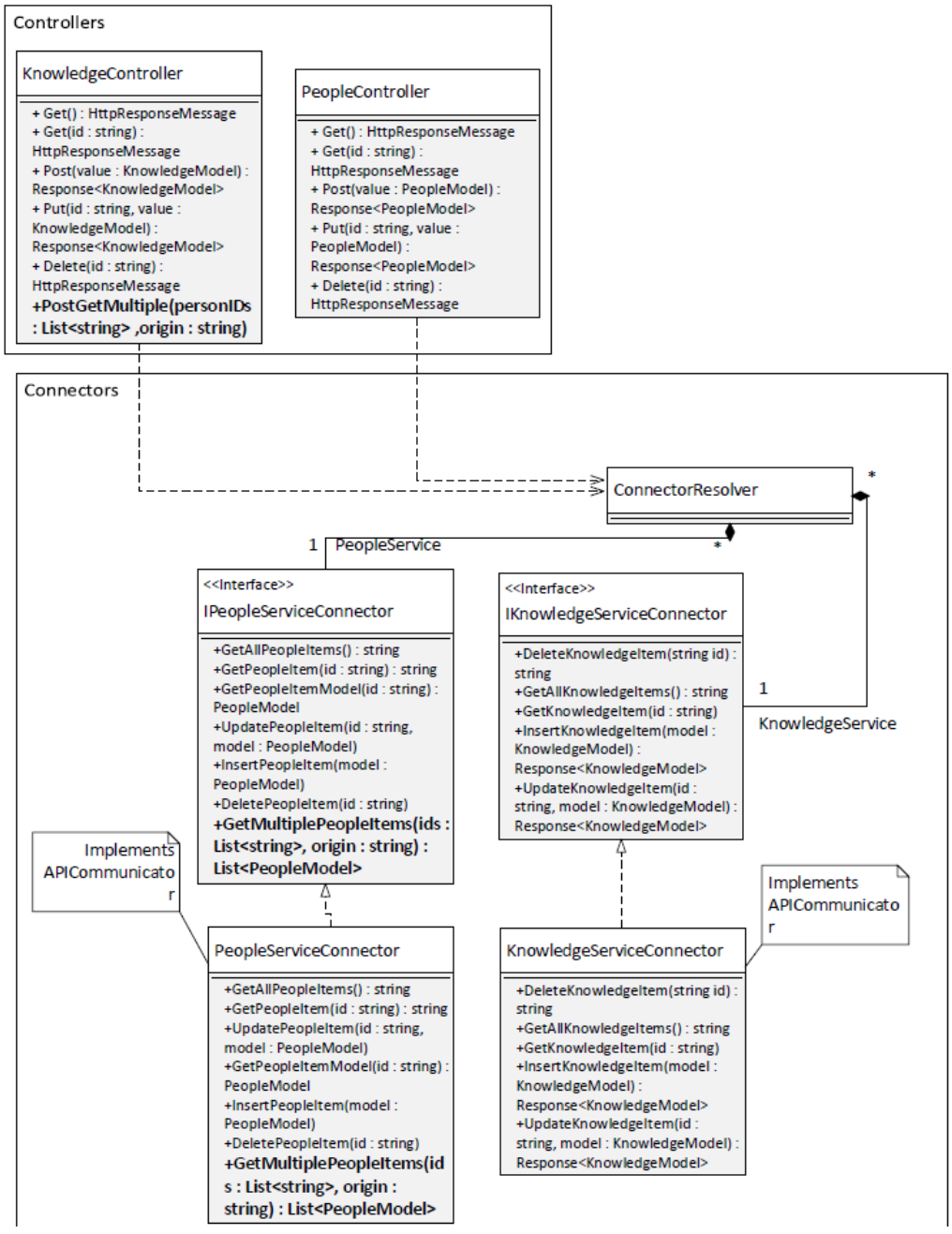
Figuur 5-3. Klassendiagram van de Publication layer (Hiërarchisch documentschema)

De verschillende klassen van de Gateway worden op de volgende pagina's besproken.

Klassenaam	Omschrijving
PeopleController	De PeopleController zorgt voor de verwerking van alle requests via de REST API.
KnowledgeController	De KnowledgeController zorgt voor de verwerking van alle reqeuest via de REST API.
GatewayKnoweldgeModel	Het GatewayKnowledgeModel is een versimpelde variant van het Knowledge Model(zie Shared Models) en wordt gebruikt voor de communicatie tussen de Webinterface en de Publication layer en de Eindgebruiker en de Publication layer. Het GatewayKnoweldgeModel bepaald de opbouw die de data in de REST API moet hebben.
IPeopleServiceConnector	Een interface voor een PeopleServiceConnector. Er is een interface gemaakt zodat er voor de testen ook gebruik gemaakt kan worden van een Mock klasse.
PeopleServiceConnector	Zorgt voor de communicatie tussen de People Service en de Gateway.
IKnowledgeServiceConnector	Een interface voor een KnowledgeServiceConnector. Er is een interface gemaakt zodat er voor de testen ook gebruik gemaakt kan worden van een Mock klasse.
KnowledgeServiceConnector	Zorgt voor de communicatie tussen de Knowledge Service en de Gateway.
ConnectorResolver	Deze klasse bevat de door de applicatie te gebruiken IPeopleServiceConnector en IKnowledgeConnector.

5.4 Publication layer (Relationeel documentschema)

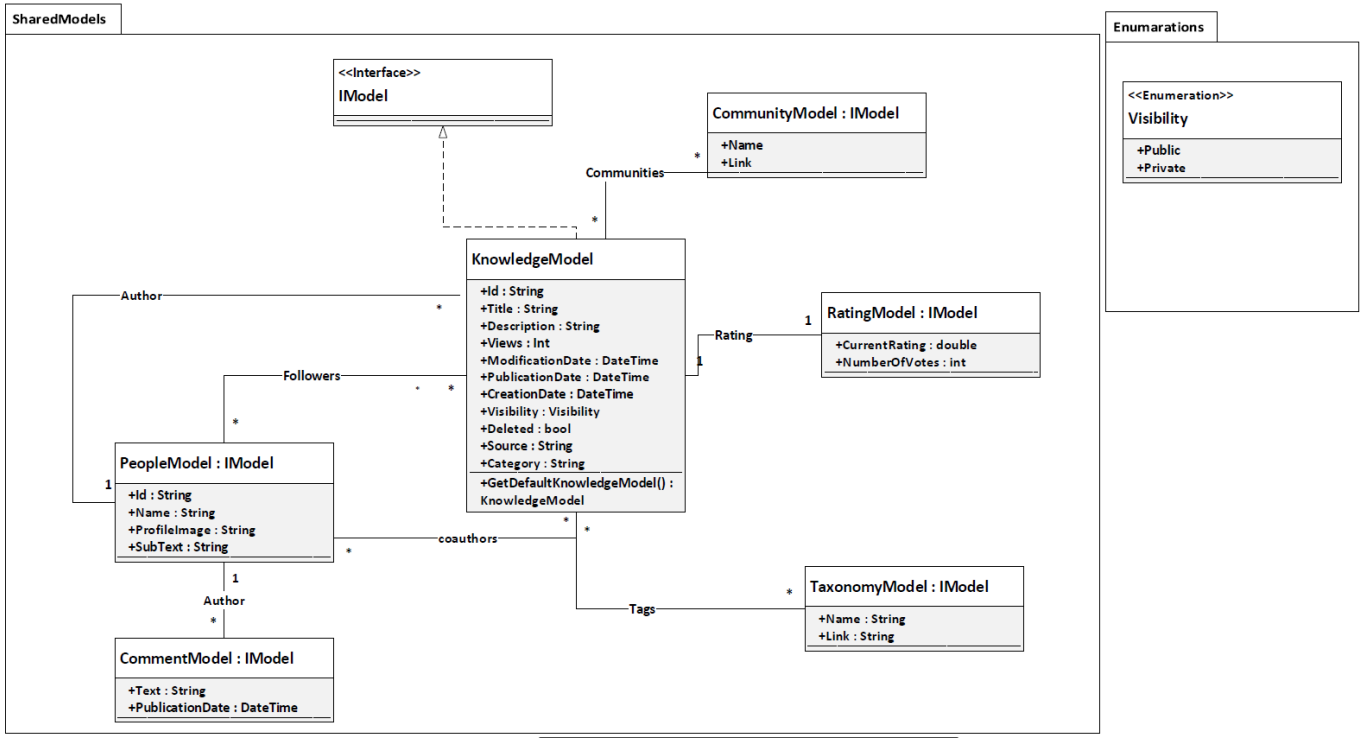
In het Proof-of-concept met relationeel documentschema is de Publication layer iets anders. Er zijn nieuwe methodes toegevoegd (dikgedrukt) die het mogelijk maken meerdere People Items in een keer op te vragen. Dit wordt gebruikt door de Web Interface om asynchroon alle bijhorende People Items op te kunnen halen.



Figuur 5-4. Klassendiagram van de Publication layer (relationeel documentschema)

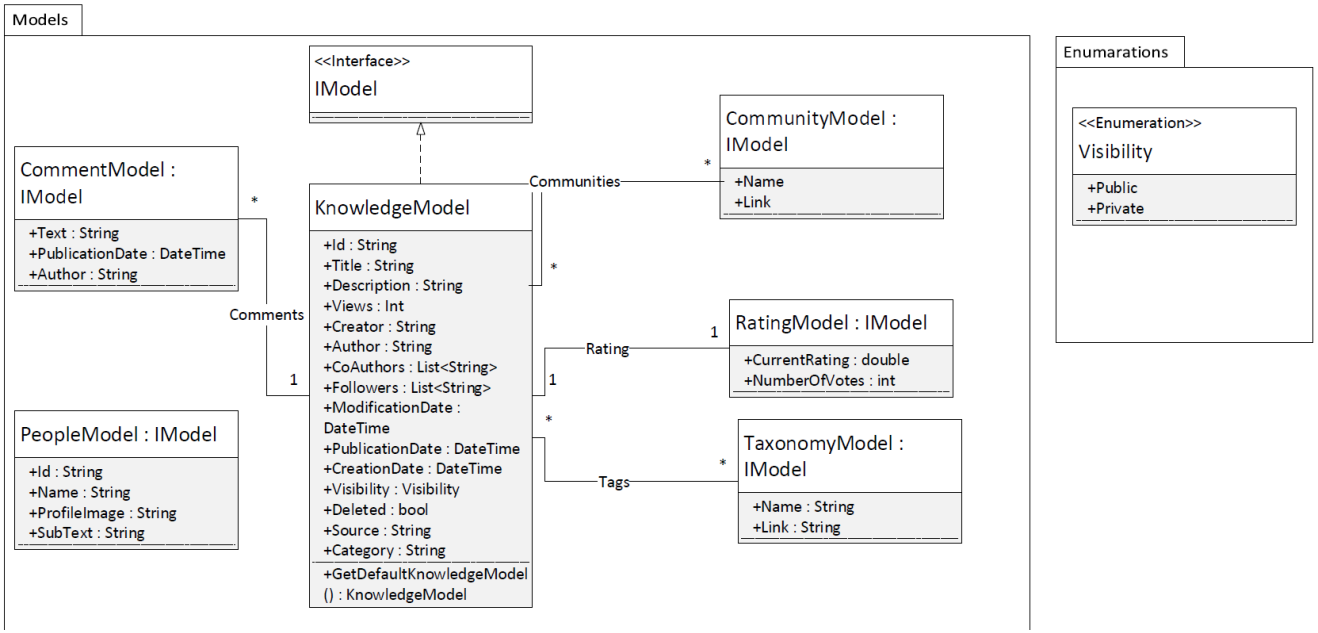
5.5 Models (Hiërarchisch documentschema)

De verschillende projecten maken veel gebruik van dezelfde modellen. Om te voorkomen dat de modellen in meerdere projecten herschreven moeten worden zijn alle modellen samengevoegd in één Class Library zodat deze hergebruikt kunnen worden door de verschillende projecten van het Proof-of-Concept.



Figuur 5-5. Models van het proof-of-concept met het hiërarchisch documentschema

5.6 Models (Relationeel documentschema)



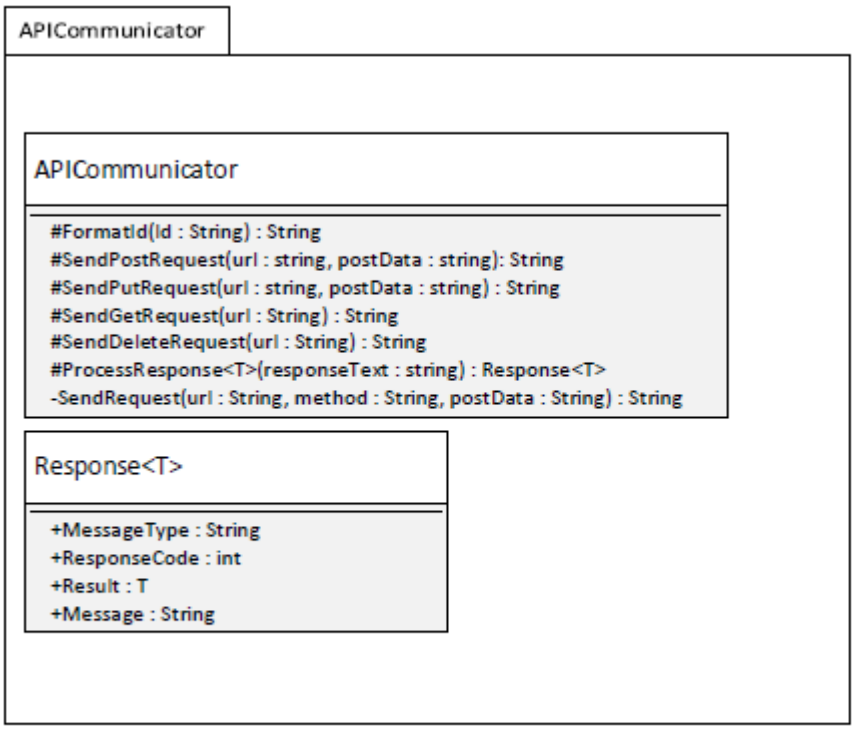
Figuur 5-6. Models van het proof-of-concept met het relationeel documentschema

Het verschil tussen de models van het Hiërarchisch en relationeel documentschema zijn de relaties tussen het KnowledgeModel en het PeopleModel. In het Hiërarchisch documentschema zijn de PeopleModel objecten opgenomen in het KnowledgeModel, alle data van een Knowledge Item, inclusief auteurs is dan al beschikbaar.

In het Relationeel model wordt er alleen vanuit het KnowledgeModel verwezen naar de PeopleModels. De client (web interface) zal aan de hand van deze verwijzing de bijhorende PeopleModels ophalen. De reden de overige models nog wel zijn opgenomen in het KnowledgeModel is omdat er voor het relationele documentschema alleen is gekeken naar de impact van het verwijzen naar People Items, het refereren naar de andere Models kan eventueel in de toekomst worden gedaan.

5.7 APICommunicator (Hiërarchisch & relationeel documentschema)

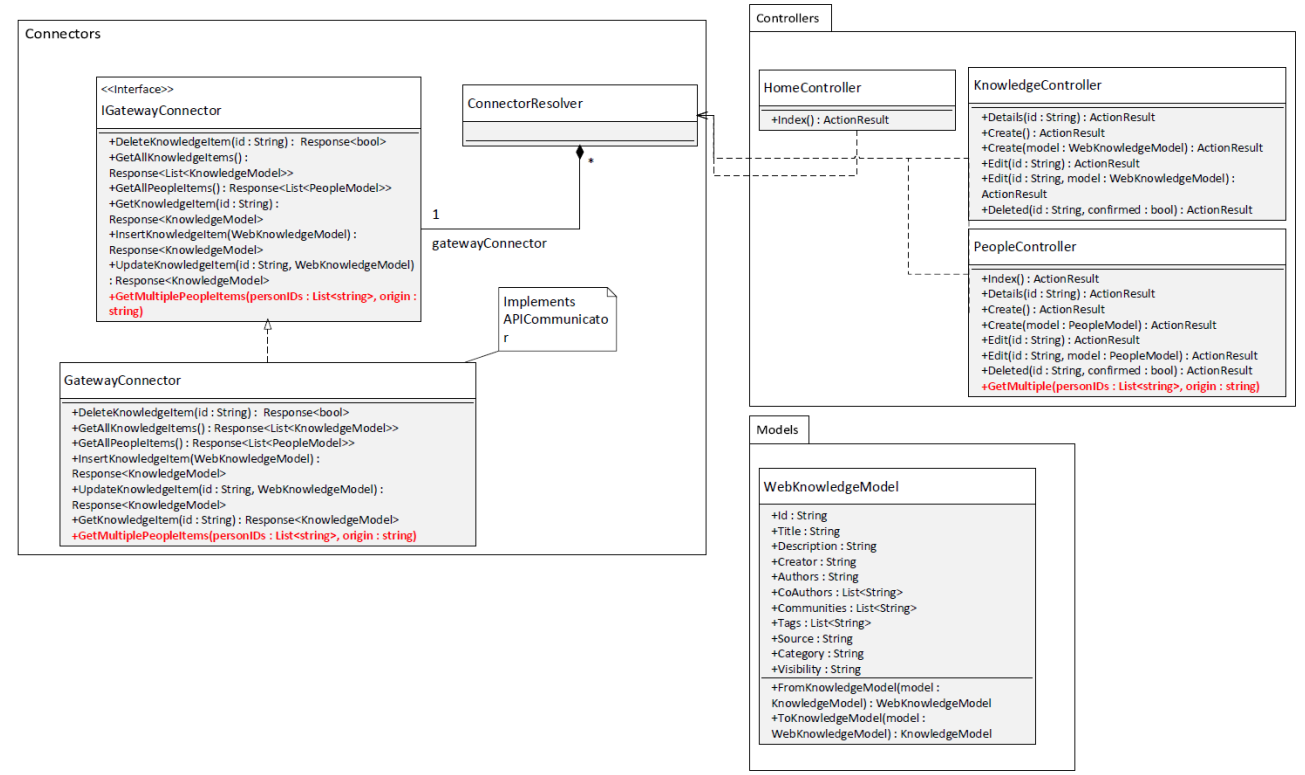
De APICommunicator wordt een Class Library die functionaliteit bevat die veel wordt gebruikt voor de communicatie tussen de verschillende services.



Figuur 5-7. Klassendiagram van de APICommunicator

Klassenaam	Omschrijving
APICommunicator	De APICommunicator bevat functionaliteit voor het communiceren over een http verbinding. De APICommunicator heeft ondersteuning voor POST-requests, PUT-requests, GET-requests en DELETE-requests.
Response	Een Response is een klasse die wordt geretourneerd door een API. In een Respose wordt onder andere een response code teruggegeven, bijvoorbeeld 200 voor geslaagd of 404 voor niet gevonden. Ook bevat een Response het resultaat. Bijvoorbeeld het opgehaalde Knowledge Item.

5.8 Web Interface (Hiërarchisch & relationeel documentschema)



Figuur 5-8. Klassendiagram van de Web Interface

In de tabel op de volgende pagina worden de verschillende klassen van de Web Interface besproken.

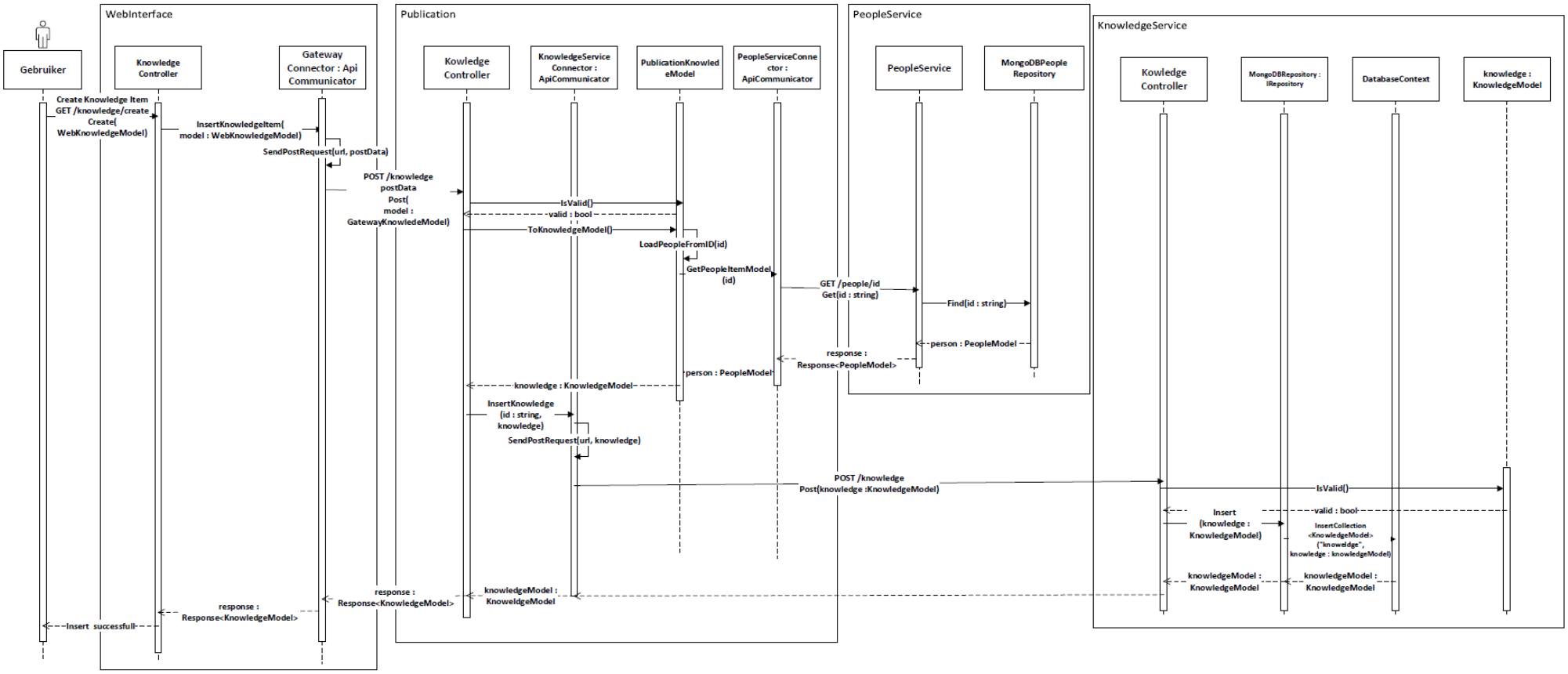
Via het Web Interface project kan een gebruiker via een webapplicatie gebruik maken van de functionaliteiten van het Proof-of-Concept. De klassen en methodes voor de Web Interface zijn in beide varianten van het Proof-of-Concept grotendeels hetzelfde. De rode methode wordt alleen gebruikt in het proof-of-concept met het relationeel documentschema. De implementatie is echter wel anders. In het Proof-of-concept met relationeel documentschema haalt de Web Interface asynchroon bijhorende People Items op, bij het hiërarchisch model zijn deze People Items al direct aanwezig.

Klassenaam	Omschrijving
HomeController	De HomeController kan alle Knowledge Items aan de gebruiker tonen
KnowledgeController	De KnowledgeController kan één Knowledge Item tonen aan de hand van een id, een Knowledge Item toevoegen, een Knowledge Item wijzigen en een Knowledge Item verwijderen.
PeopleController	De PeopleController kan één People Item tonen aan de hand van een id, een People Item toevoegen, een People Item wijzigen en een People Item verwijderen. De rode methode wordt alleen gebruikt in het Proof-of-Concept met het hiërarchisch documentschema.
WebKnoweldgeModel	Het WebKnowledgeModel is een versimpelde variant van het Knowledge Model(zie Shared Models) en wordt gebruikt voor de communicatie tussen de Webinterface en de gebruiker en is bedoeld voor de verwerking van de formulieren. Een WebKnowledgeModel kan worden gemaakt van een KnowledgeModel.
IGatewayConnector	Een interface voor een GatewayConnector Er is een interface gemaakt zodat er voor de testen ook gebruik gemaakt kan worden van een Mock klasse.
GatewayConnector	Zorgt voor de communicatie tussen de Web Interface en de Gateway.
ConnectorResolver	Deze klasse bevat de door de applicatie te gebruiken IGatewayConnector

6. Sequence diagrammen

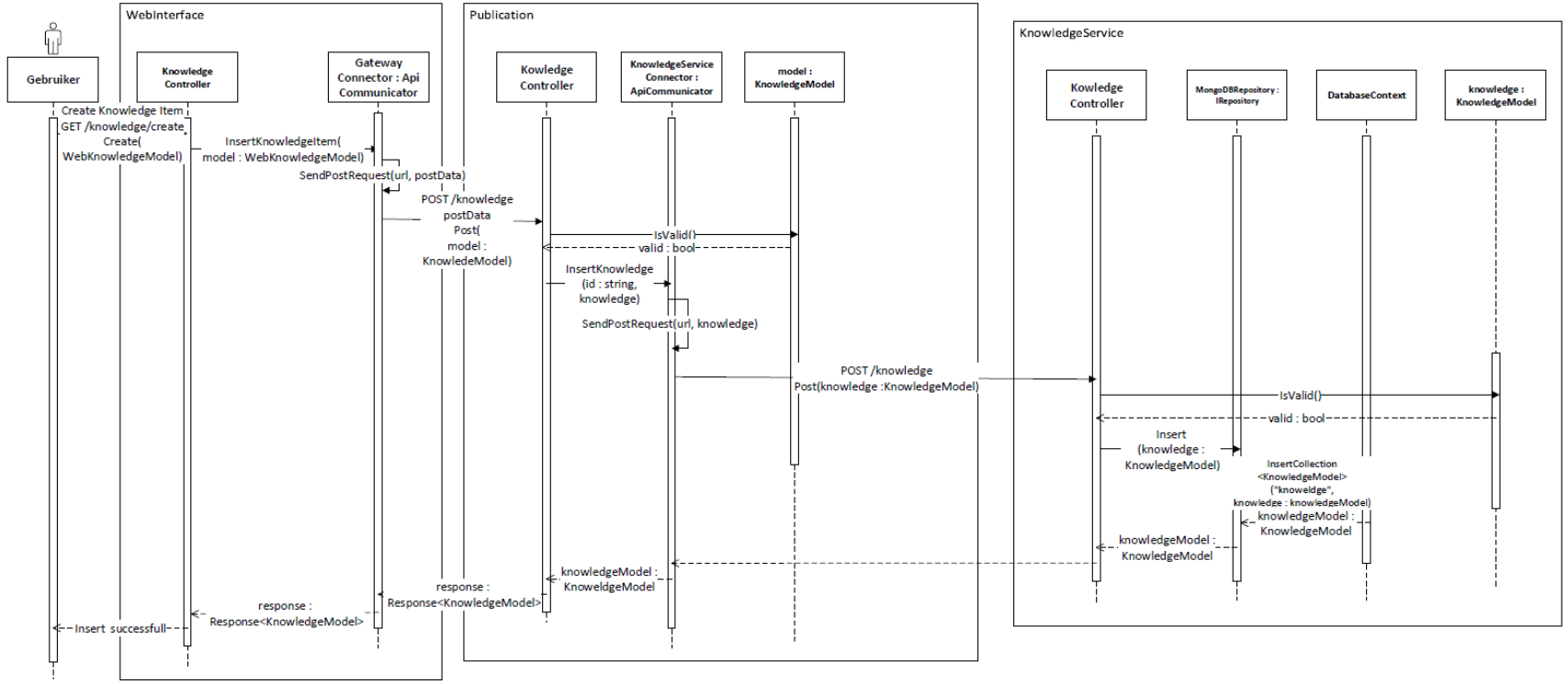
Van de Use Cases uit het Functioneel Ontwerp worden voor het Technisch Ontwerp Sequence diagrammen gemaakt.

6.1 Toevoegen van Knowledge Item (Hiërarchisch documentschema)

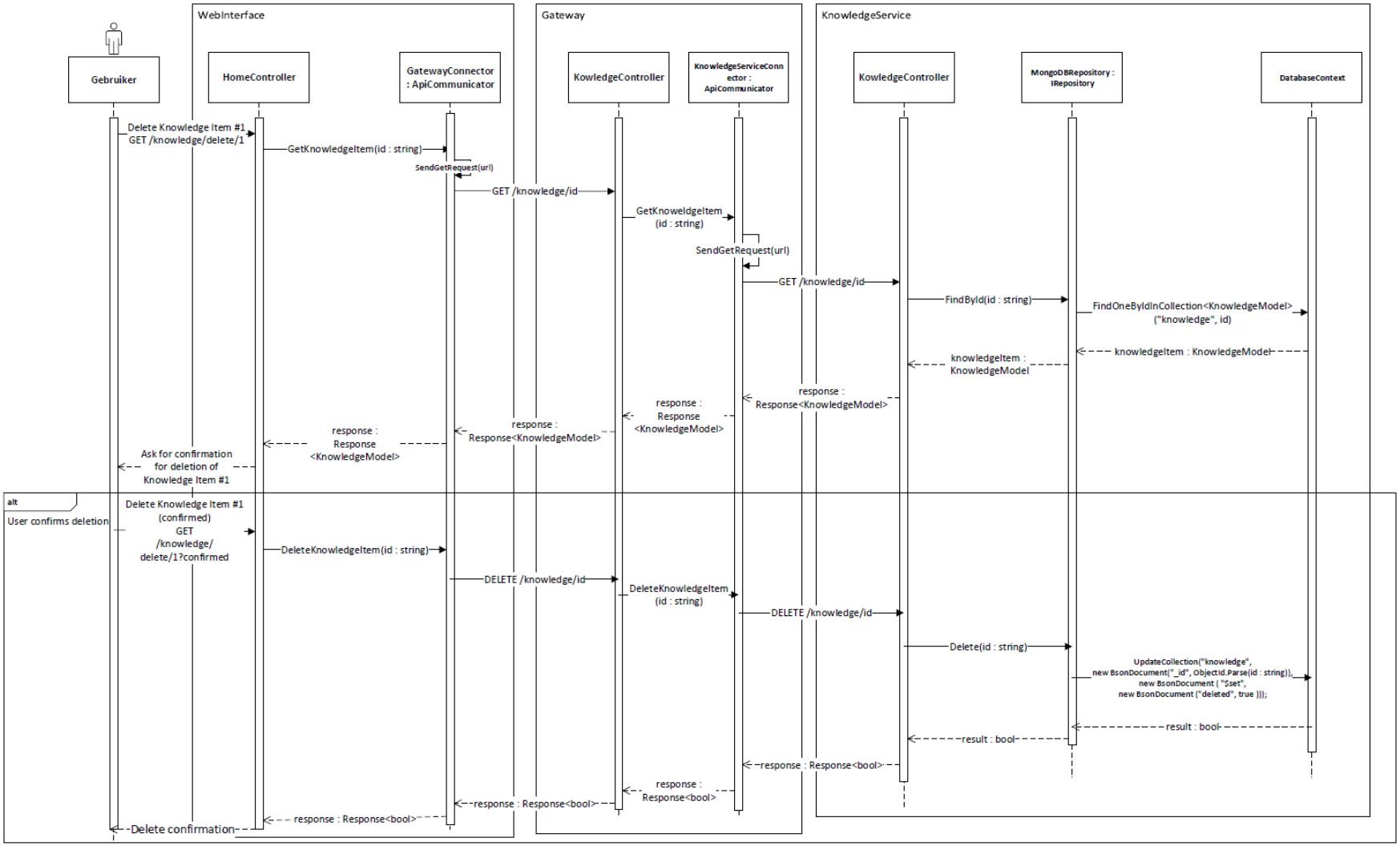


6.2 Toevoegen van een Knowledge Item (Relationeel documentschema)

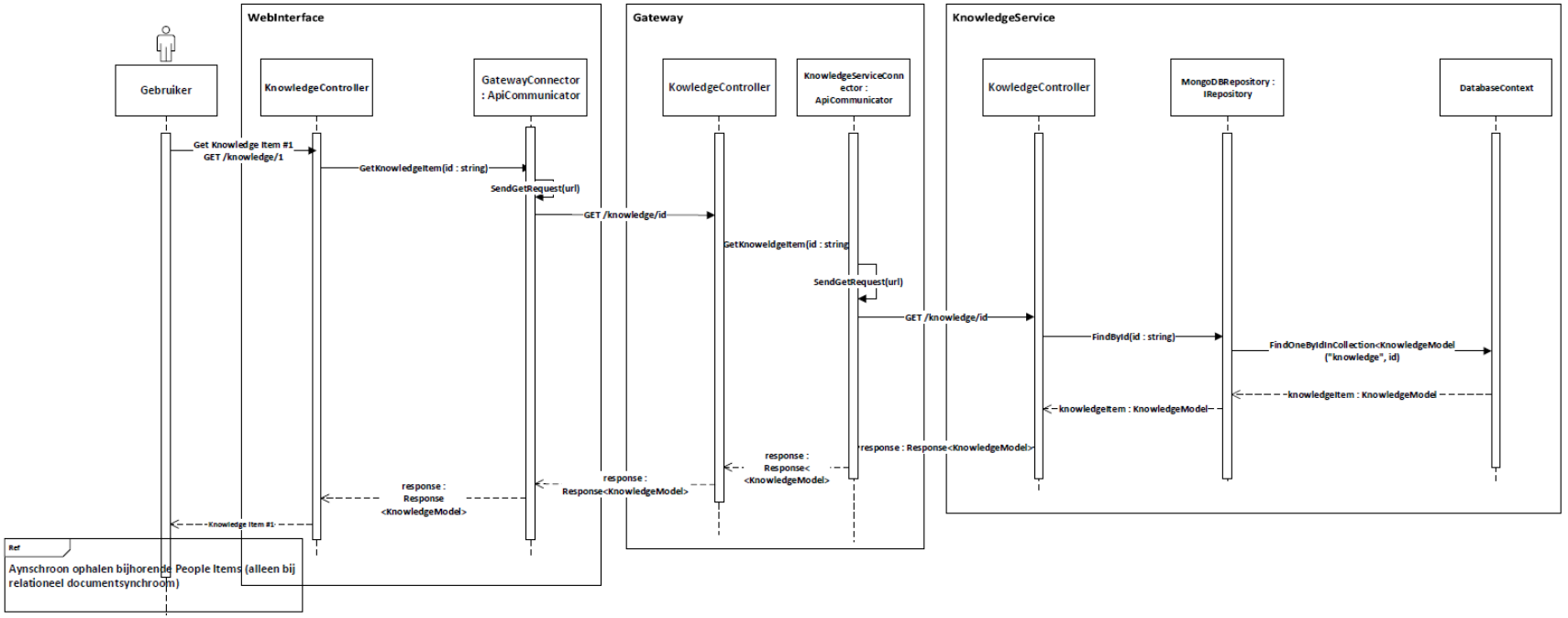
Het Proof-of-concept met het relationele documentschema hoeft bij het toevoegen van een Knowledge Item niet eerst de bijhorende People Items op te halen, er wordt alleen verwezen. Hierdoor hoeft de Publication Layer geen contact meer op te nemen met de People Service.



6.3 Verwijderen van Knowledge Item (Hiërarchisch & relationeel documentschema)



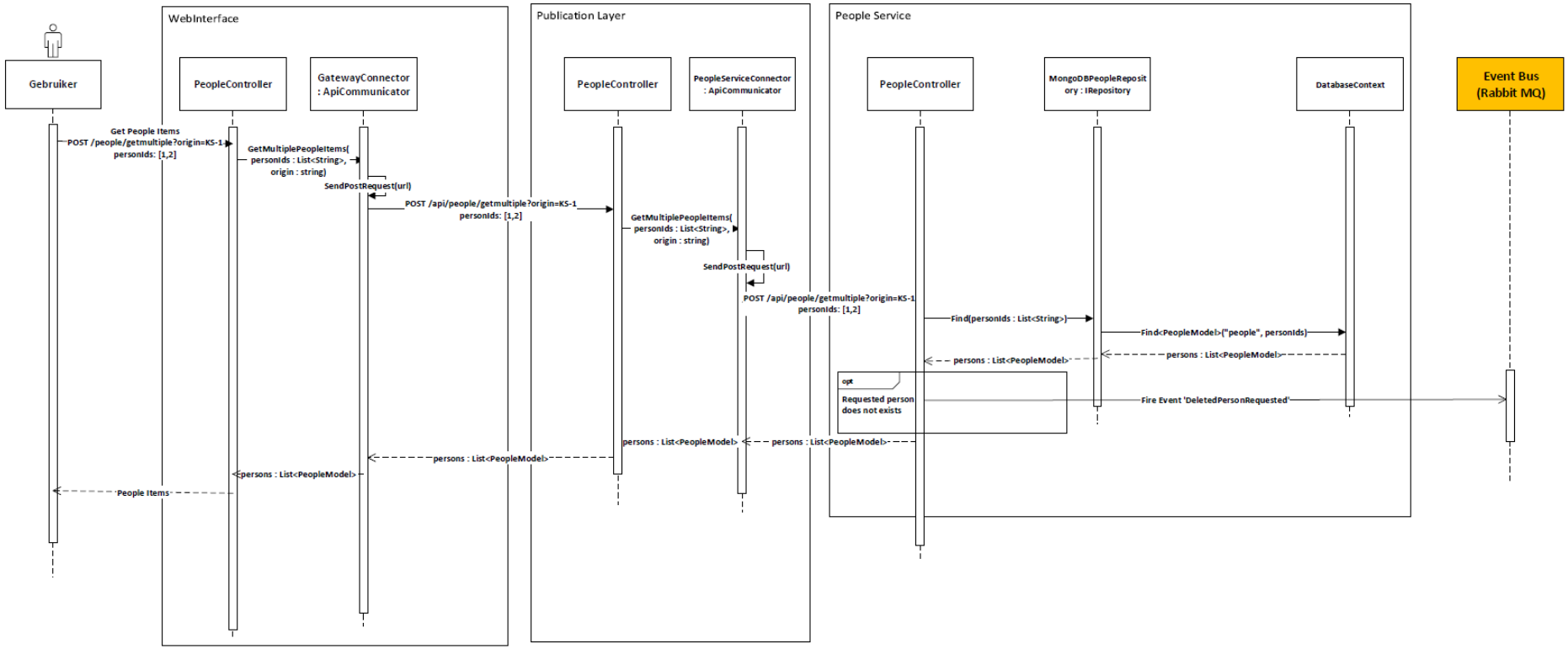
6.4 Tonen van Knowledge Items aan de hand van een ID (Hiërarchisch & relationeel documentschema)



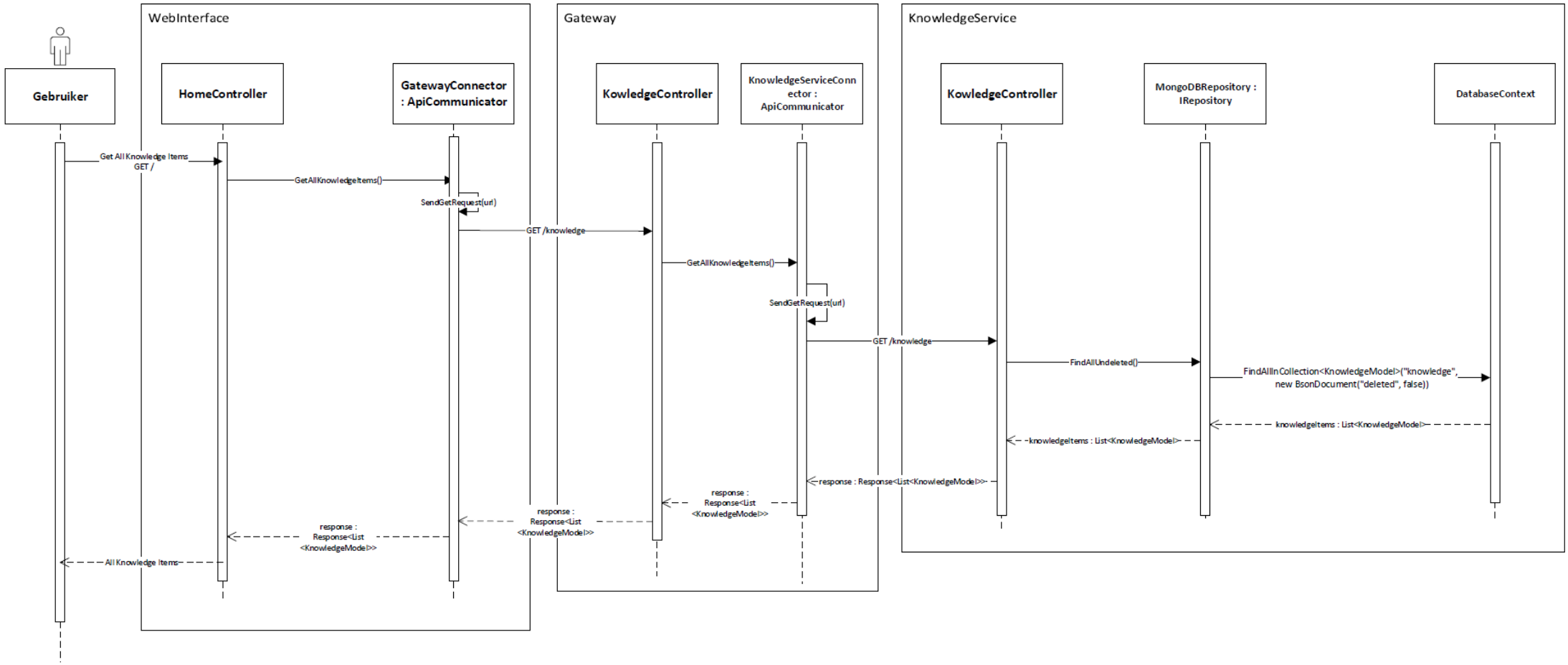
Het ‘Asynchroon ophalen bijhorende People Items’ vind alleen plaats in het Proof-of-Concept met het relationele documentschema. Dit wordt in 6.4.1 toegelicht.

6.4.1 Asynchroon ophalen bijhorende People Items (Relationeel documentschema)

Het asynchroon ophalen van bijhorende People Items is een proces wat alleen in het Proof-of-Concept met relationeel documentschema plaatsvind. Hierbij worden de bijhorende People Items opgehaald aan de hand van de referentie in het Knowledge Item. Dit gebeurt asynchroon door middel van AJAX. De gebruiker kan al beginnen met het lezen van het Knowledge Item wanneer het People Item wordt geladen. Deze sequentiediagram gaat verder op de sequentiediagram van 6.4.



6.5 Tonen van alle Knowledge Items (Hiërarchisch & relationeel documentschema)



6.6 Verwerken wijziging van People Item in Knowledge Service (Job Queue)

Dit proces vind alleen plaats in het proof-of-concept met hiërarchisch documentschema. In het relationeel documentschema is het niet nodig People Items te updaten in de Knowledge Items omdat hier alleen wordt gerefereerd naar deze People Items.

Het verwerken van een update vanuit de People Service gebeurt door middel van de in het onderzoek ontworpen Job Queue. Bij de Job Queue worden updates vanuit de People Service in een wachtrij opgeslagen. De applicatie voert de taken in de wachtrij daarna één voor één uit. Dit gaat volgens het First In, First Out principe. Een taak die als eerst in de wachtrij terecht komt, wordt ook als eerst uitgevoerd.

In het Proof-of-Concept wordt Job Queue door mij ontwikkeld door gebruik te maken van RabbitMQ en een lijst met taken (wachtrij) in de applicatie. Wanneer een persoon zijn gegevens wijzigt zal de People Service een event sturen naar de RabbitMQ Event Bus. De Knowledge Service luistert naar deze Event Bus en registreert alle update events in een lijst en werkt deze één voor één af. Wanneer een taak is uitgevoerd bevestigd de applicatie de update aan RabbitMQ. RabbitMQ zal het event hierdoor verwijderen.

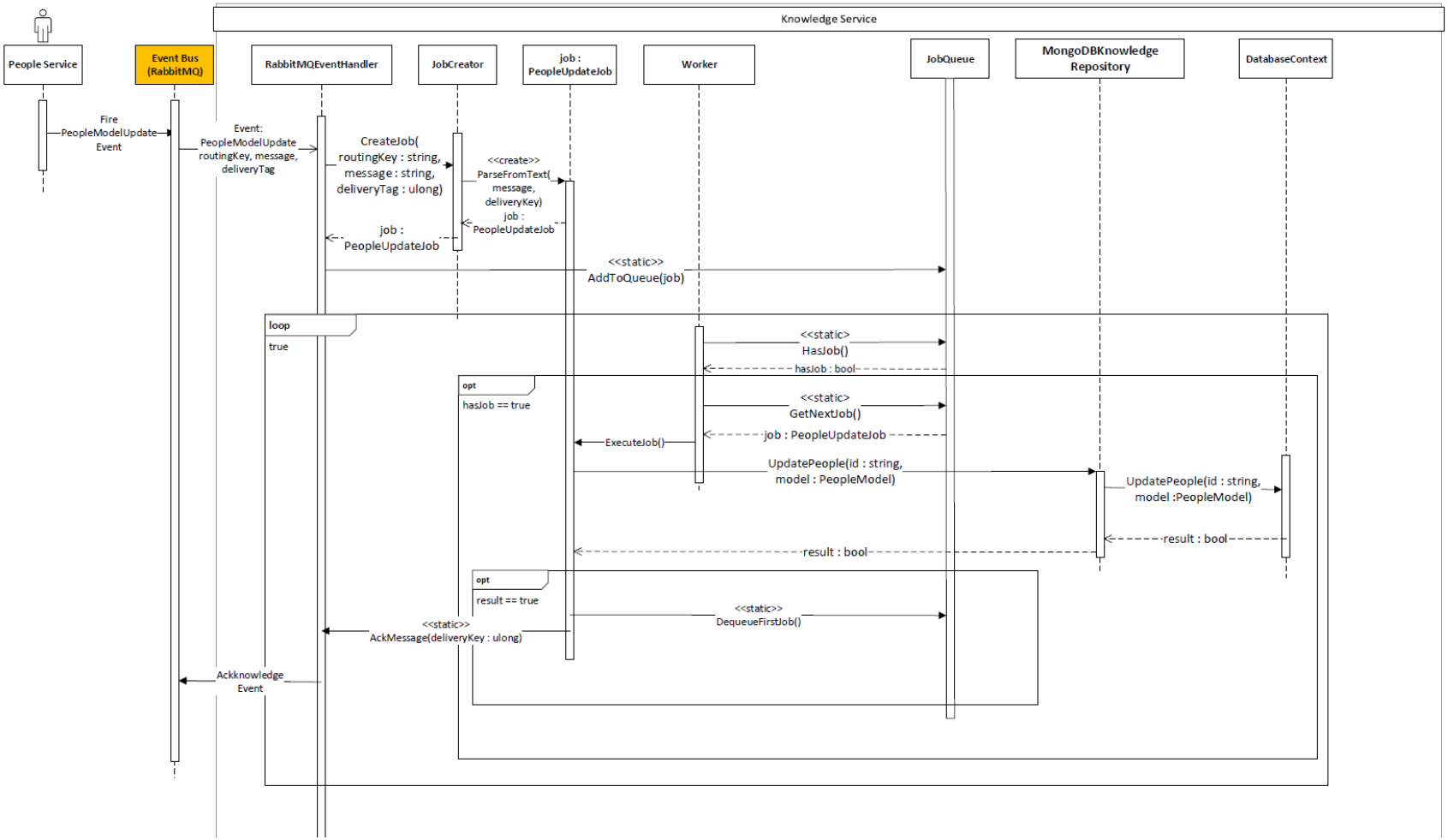
Mocht tijdens het verwerken van een update MongoDB uitvallen, dan zal de applicatie de update taak bewaren en wordt de gehele wachtrij gepauzeerd. Wanneer MongoDB weer beschikbaar komt voert deze de taak opnieuw uit en zal daarna verder gaan met het verwerken van de taken in de wachtrij.

Mocht de gehele Knowledge Service uitvallen dan zal de Knowledge Service alle onbevestigde events opnieuw ontvangen vanuit RabbitMQ. Hierdoor raken er geen onuitgevoerde events verloren. De Knowledge Service bevestigd namelijk alleen events wanneer deze volledig zijn uitgevoerd.

Door middel van de Job Queue is het mogelijk het documentschema consistent te houden met de andere services.

De Job Queue voor het verwijderen luistert naar het ‘PeopleModelChanged’ event wat direct wordt afgevuurd door de People Service na het wijzigen van een People Item.

Op de volgende pagina is de Job Queue door middel van een Sequence diagram verduidelijkt.


Knowledge Service
People Service
Event Bus (RabbitMQ)
RabbitMQEventHandler
JobCreator
job : PeopleUpdateJob
Worker
JobQueue
MongoDBKnowledge Repository
DatabaseContext
Fire PeopleModelUpdate Event
Event: PeopleModelUpdate routingKey, message, deliveryTag
CreateJob(routingKey : string, message : string, deliveryTag : ulong)
<<create>> ParseFromText(message, deliveryKey)
job : PeopleUpdateJob
job : PeopleUpdateJob
<<static>> AddToQueue(job)
loop
true
<<static> HasJob()
hasJob : bool
opt
hasJob == true
<<static> GetNextJob()
job : PeopleUpdateJob
ExecuteJob()
UpdatePeople(id : string, model : PeopleModel)
UpdatePeople(id : string, model :PeopleModel)
result : bool
result : bool
opt
result == true
<<static>> DequeueFirstJob()
<<static>> AckMessage(deliveryKey : ulong)
Ackknowledge Event

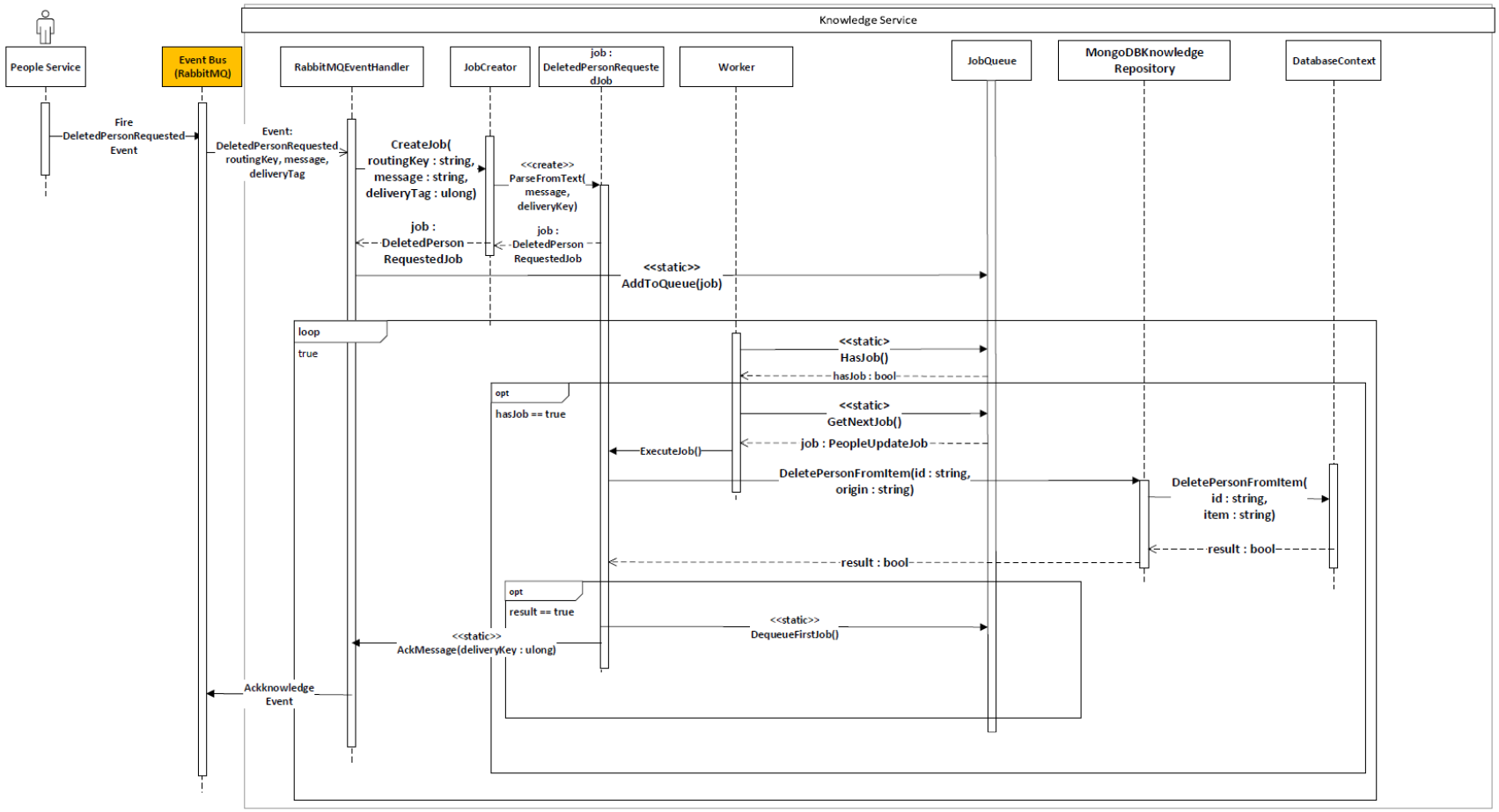
6.7 Verwerken verwijdering van People Item in Knowledge Service

6.7.1 Hiërarchisch documentschema

Het verwijderen van een People Item uit Knowledge Items bij het Proof-of-Concept met hiërarchisch documentschema gebeurt op eenzelfde manier als het wijzigen. Ook dit gebeurt via de Job Queue waarbij alle Knowledge Items worden langsgelopen om alle verwijderde People Items uit de Knowledge items te halen. De Job Queue voor het verwijderen luistert naar het 'PeopleDeleted' event wat direct wordt afgevuurd door de People Service na het verwijderen van een People Item.

6.7.2 Relationeel documentschema

Bij het verwijderen van een People Item in het relationeel documentschema wordt in eerste instantie geen event afgevuurd door de People Service. Hierdoor worden referenties van People Items niet direct verwijderd. Het verwijderen van een referentie gebeurt pas op het moment dat een People Item wordt opgevraagd wat niet bestaat. De People Service vuurt dan het event 'DeletedPersonRequested' af waarna de Knowledge Service de verwijzing naar het People Item voor dat Knowledge Item verwijderd. Het verwijderen van een item in het relationeel documentschema is in de sequencediagram in de volgende pagina weergeven.


Knowledge Service
People Service
Event Bus (RabbitMQ)
RabbitMQEventHandler
JobCreator
job : DeletedPersonRequestedJob
Worker
JobQueue
MongoDBKnowledge Repository
DatabaseContext
Fire DeletedPersonRequested Event
Event: DeletedPersonRequested routingKey, message, deliveryTag
CreateJob(routingKey : string, message : string, deliveryTag : ulong)
<<create>> ParseFromText(message, deliveryKey)
job : DeletedPerson RequestedJob
job : DeletedPerson RequestedJob
<<static>> AddToQueue(job)
loop
true
<<static> HasJob()
hasJob : bool
opt
hasJob == true
<<static> GetNextJob()
job : PeopleUpdateJob
ExecuteJob()
DeletePersonFromItem(id : string, origin : string)
DeletePersonFromItem(id : string, item : string)
result : bool
result : bool
opt
result == true
<<static>> DequeueFirstJob()
<<static>> AckMessage(deliveryKey : ulong)
Ackknowledge Event

6.8 Overige Use Cases

Use Case	Uitleg
Toevoegen van People Item	Het toevoegen van een People Item lijkt erg veel op het toevoegen van een Knowledge Item, echter verloopt dit via de People Service in plaats van de Knowledge Service. Het is daarom niet meegenomen als sequence diagram.
Wijzigen van People Item	Het wijzigen van een People Item lijkt erg veel op het wijzigen van een Knowledge Item, echter verloopt dit via de People Service in plaats van de Knowledge Service. Het is daarom niet meegenomen als sequence diagram. Na het wijzigen van een People Item wordt door de People Service het event 'PeopleModelUpdate' verstuurd naar de Event Bus, RabbitMQ.
Verwijderen van People Item	Het verwijderen van een People Item lijkt erg veel op het toevoegen van een Knowledge Item, echter verloopt dit via de People Service in plaats van de Knowledge Service. Het is daarom niet meegenomen als sequence diagram. Na het verwijderen van een People Item wordt door de People Service het event 'PeopleDeleted' verstuurd naar de Event Bus, RabbitMQ.
Tonen van alle People Items	Het tonen van alle People Items werkt hetzelfde als het tonen van alle Knowledge Items en is daarom niet meegenomen als sequence diagram.
Tonen van één People Item	Het tonen van een People Item werkt hetzelfde als het tonen van een Knowledge Item en is daarom niet meegenomen als sequence diagram. Wanneer een niet-bestaand People Item wordt opgehaald wordt een 'DeletedPersonRequested' event verstuurd naar de Event Bus.
Wijzigen van een Knowledge item	Het wijzigen van een Knowledge Item lijkt erg veel op het toevoegen van een Knowledge Item en is hierom niet meegenomen.

7. Documentschema Knowledge Service

Voor het Proof-of-Concept wordt er gebruik gemaakt van 2 verschillende documentschema's. Het hiërarchische documentschema en het relationele documentschema. De documentschema's worden toegelicht in het document ' Ontwerp documentschema'. Deze docu

DE HAAGSE HOGESCHOOL

Info Support Solid Innovator

Bijlage H – Testrapportage

MongoDB: Een geschikte database voor het kennismanagementsysteem knowNow?

Testrapportage

Titel	Testrapportage
Project/Onderwerp	MongoDB: Een geschikte database voor het kennismanagementsysteem knowNow?
Versie	1.0
Status	Definitief
Datum	01-10-2015
Bestand	Testrapportage v1.0
Bedrijf	Info Support
Door	Tom Keim

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

Inhoudsopgave

1. Inleiding

Dit rapport beschrijft de testen die zijn uitgevoerd om beide varianten van het Proof-of-Concept te testen. Het Proof-of-Concept is getest met de volgende testsoorten:

- Proces-cyclus-test (alleen Proof-of-Concept met Hiërarchisch documentschema)
- Use Case testen (beide varianten)

Als testbasis wordt het onderzoeksrapport, het Functioneel Ontwerp, het Technisch Ontwerp en beide varianten van het Proof-of-Concept gebruikt. Ik raad de lezer van dit document aan deze documenten eerst te lezen en het Proof-of-Concept eerst door te nemen voordat dit document wordt gelezen. Hierdoor zijn de diverse termen die in dit document worden gebruikt voor de lezer duidelijker.

2. Testen van de Job Queue

Deze test is alleen uitgevoerd op het Proof-of-Concept met hiërarchisch documentschema. Het Proof-of-Concept met relationeel documentschema maakt geen gebruik van de Job Queue om People Items te wijzigen in de Knowledge Items.

De Job Queue, zoals omschreven in het Onderzoeksrapport en het Technisch Ontwerp, is een bedachte oplossing om MongoDB consistent te kunnen houden met andere services. MongoDB kent namelijk geen transacties op collectie niveau, enkel op document niveau. Hierdoor zal tijdens het uitvoeren van een update van bijvoorbeeld een auteur die in alle documenten moet worden bijgewerkt de database inconsistent kunnen raken wanneer de database tijdens deze update uitvalt.

De Job Queue is geïmplementeerd in het Proof-of-Concept wat voor de afstudeeropdracht is gerealiseerd. Om te testen of de bedachte oplossing ook daadwerkelijk functioneert is besloten een test uit te voeren.

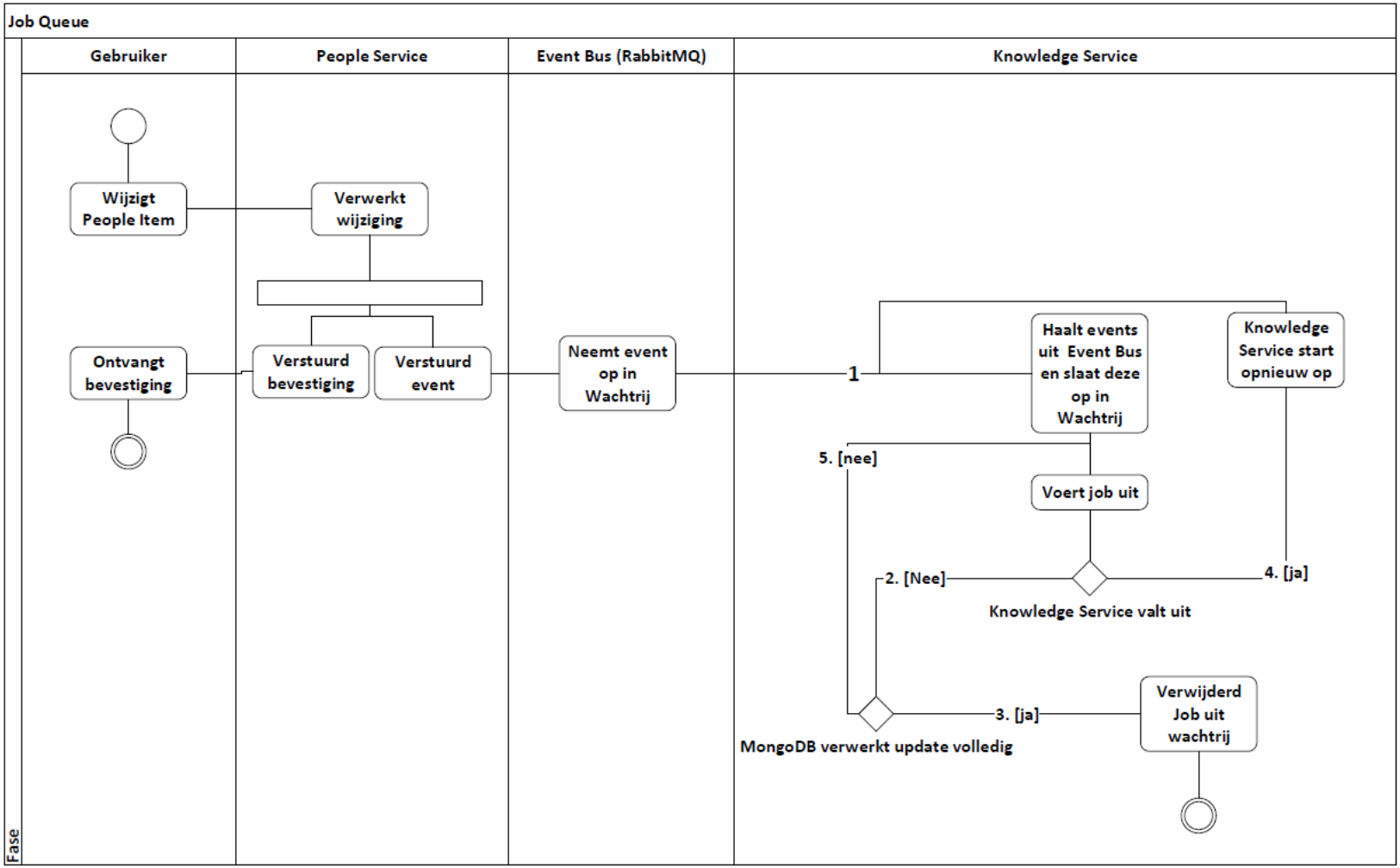
Voor het testen van de Job Queue is gekeken naar TMap Next. In een document van TMap, 'Overzicht toegepaste testvormen'[1] is te vinden welke kwaliteitsattributen gelden voor welk testvormen. Er is gekozen voor een Procestest. Dit wordt gedaan door middel van een Proces Cyclus Test[2]. Voor de Proces Cyclus Test is gekozen omdat deze test duidelijk de verschillende paden test die de Job Queue kan belopen. Hierdoor worden alle mogelijke situaties van de Job Queue getest. De Proces Cyclus Test test de suitability van de implementatie. De suitability test de mate waarin de gewenste functies aanwezig zijn en geschikt om gespecificeerde taken uit te voeren[3].

Er wordt eerst een Activity Diagram gemaakt. Hierin zijn alle paden van de Job Queue schematisch weergeven. Daarna wordt de testmaat en de mogelijke paden die bij die testmaat horen bepaald. Daarna worden logische en fysieke testcases opgezet en wordt er daadwerkelijk getest.

Alleen de Job Queue implementatie in de Knowledge Service wordt getest. Hiermee wordt alleen getest of de bedachte Job Queue werkt. Het proces voordat een taak bij de Job Queue aankomt is buiten beschouwing gelaten.

[1] TMap, "Overzicht Toegepaste testvormen"
http://www.mentorum.nl/docs/Traindocs/Overzicht_Toegepaste_testvormen.pdf
[2] TMap, Procescyclustest
http://www.tmap.net/wiki/process-cycle-test-pct
[3] SmartTest, ISO9126
http://www.smartest.nl/verdieping/kwaliteitsmodellen/iso9126

2.1 Activity Diagram



2.2 Testmaat & paden

Er zijn drie verschillende testmaten. Testmaat 1, Testmaat 2 en Testmaat 3. Bij testmaat 1 wordt elk pad één keer belopen. Omdat soms paden meerdere keren doorlopen kunnen worden is er gekozen voor Testmaat 2. Testmaat 2 wordt behaald door eerst per beslispunt alle mogelijke paden die er in gaan te bepalen en alle mogelijke paden die eruit gaan. In het geval van de bovenstaande diagram resulteert dat in de volgende tabel:

Beslispunt	Pad in	Pad uit
Knowledge Service valt uit (A)	1,5,4	2,4
MongoDB verwerkt update volledig (B)	2	5,3

Daarna worden alle mogelijke combinaties van de in en uitgang bepaald:

Beslispunt	Paden
A	1-2;5-2;4-2;1-4;5-4;4-4
B	2-5;2-3

Hierna moet er bij testmaat 2 voor gezorgd worden dat elke combinatie minimaal een keer wordt doorlopen. Dit leid tot de volgende combinaties van paden:

- **Testcase 1:** 1-2-3
- **Testcase 2:** 1-4-2-3
- **Testcase 3:** 1-2-5-2-3
- **Testcase 4:** 1-4-4-2-3
- **Testcase 5:** 1-2-5-4-2-3

Op basis van deze paden worden in 2.3 Logische testcases opgesteld zodat deze paden voor de lezer te begrijpen zijn.

Bij testmaat 3 worden in plaats van combinaties met 2 paden combinaties met 3 paden gebruikt. Er is voor testsoort 2 gekozen omdat dit voor deze test een voldoende basis geeft. Alle mogelijke paden zijn dan tenminste een keer doorlopen.

2.3 Logische testcases

2.3.1 Testcase 1: Geslaagde Job

In deze situatie is een update Job geslaagd. De stappen in de testcase zijn als volgt:

1. De gebruiker wijzigt een People Item
2. De People Service verwerkt dit People Item
3. De People Service stuurt een Event naar de RabbitMQ Event Bus
4. De Knowledge Service ontvangt de Job van de RabbitMQ Event Bus en plaatst deze in de wachtrij
5. De Knowledge Service start de uitvoering van de update
6. MongoDB verwerkt de update
7. De Knowledge Service verwijdert de Job uit de wachtrij.

2.3.2 Testcase 2: Uitvallen van de Knowledge Service

In deze situatie valt de Knowledge Service uit tijdens het uitvoeren van een Job en wordt deze daarna weer opgestart om de Job opnieuw uit te voeren.

1. De gebruiker wijzigt een People Item
2. De People Service verwerkt dit People Item
3. De People Service stuurt een Event naar de RabbitMQ Event Bus
4. De Knowledge Service ontvangt de Job van de RabbitMQ Event Bus en plaatst deze in de wachtrij
5. De Knowledge Service start de uitvoering van de update
6. De Knowledge Service wordt uitgezet
7. De Knowledge Service wordt opgestart
8. De Knowledge Service ontvangt de Job van de RabbitMQ Event Bus en plaatst deze in de wachtrij
9. De Knowledge Service start de uitvoering van de update
10. MongoDB verwerkt de update
11. De Knowledge Service verwijdert de Job uit de wachtrij.

2.3.3 Testcase 3: MongoDB valt uit

In deze situatie valt MongoDB uit tijdens de verwerking van een Job. Hierdoor wordt de Job net zo lang opnieuw geprobeerd totdat MongoDB weer beschikbaar is. Als dat het geval is wordt de Job opnieuw uitgevoerd.

1. De gebruiker wijzigt een People Item
2. De People Service verwerkt dit People Item
3. De People Service stuurt een Event naar de RabbitMQ Event Bus
4. De Knowledge Service ontvangt de Job van de RabbitMQ Event Bus en plaatst deze in de wachtrij
5. De Knowledge Service start de uitvoering van de update
6. MongoDB valt uit bij het verwerken van de update, de database is nu mogelijk inconsistent.
7. MongoDB start weer op
8. De Knowledge Service start de uitvoering van de update opnieuw
9. MongoDB verwerkt de update
10. De Knowledge Service verwijdert de Job uit de wachtrij.

2.3.4 Testcase 4: De Knowledge Service valt twee keer uit bij het uitvoeren van een Job.

In deze situatie valt de Knowledge Service uit tijdens de uitvoering van een Job. Hierna wordt de Knowledge Service opnieuw opgestart maar valt opnieuw uit. Hierna start de Knowledge Service wel correct opnieuw op en wordt de Job opnieuw uitgevoerd.

1. De gebruiker wijzigt een People Item
2. De People Service verwerkt dit People Item
3. De People Service stuurt een Event naar de RabbitMQ Event Bus
4. De Knowledge Service ontvangt de Job van de RabbitMQ Event Bus en plaatst deze in de wachtrij
5. De Knowledge Service start de uitvoering van de update
6. De Knowledge Service wordt uitgezet
7. De Knowledge Service wordt opgestart
8. De Knowledge Service ontvangt de Job van de RabbitMQ Event Bus en plaatst deze in de wachtrij
9. De Knowledge Service wordt uitgezet
10. De Knowledge Service wordt opgestart
11. De Knowledge Service ontvangt de Job van de RabbitMQ Event Bus en plaatst deze in de wachtrij
12. De Knowledge Service start de uitvoering van de update
13. MongoDB verwerkt de update
14. De Knowledge Service verwijdert de Job uit de wachtrij.

2.3.5 Testcase 5: MongoDB valt uit en daarna valt de Knowledge Service uit

In deze situatie valt MongoDB uit tijdens het uitvoeren van een update. Wanneer MongoDB weer beschikbaar is valt de Knowledge Service uit. Deze wordt hierna opnieuw opgestart waardoor de Job uiteindelijk slaagt.

1. De gebruiker wijzigt een People Item
2. De People Service verwerkt dit People Item
3. De People Service stuurt een Event naar de RabbitMQ Event Bus
4. De Knowledge Service ontvangt de Job van de RabbitMQ Event Bus en plaatst deze in de wachtrij
5. De Knowledge Service start de uitvoering van de update
6. MongoDB valt uit bij het verwerken van de update, de database is nu mogelijk inconsistent.
7. MongoDB start weer op
8. De Knowledge Service start de uitvoering van de update opnieuw
9. De Knowledge Service wordt uitgezet, de database is nu mogelijk inconsistent
10. De Knowledge Service wordt opgestart
11. De Knowledge Service ontvangt de Job van de RabbitMQ Event Bus en plaatst deze in de wachtrij
12. De Knowledge Service start de uitvoering van de update
13. MongoDB verwerkt de update
14. De Knowledge Service verwijdert de Job uit de wachtrij.

2.4 Fysieke testcases & Uitvoering

2.4.1 Testcase 1: Geslaagde Job

In deze situatie is een update Job geslaagd. De stappen in deze fysieke testcase zijn beschreven in de onderstaande tabel. Bij elke stap is opgeschreven of het verwacht resultaat overeenkomt met het geconstateerde resultaat tijdens het uitvoeren van de test.

Stap	Verwacht resultaat	Klopt?
De gebruiker wijzigt de naam van het People Item 1 van 'Dai Potts' naar 'Dai Pott'.	-	Ja
De People Service wijzigt in zijn database People Item 1 van 'Dai Potts' naar 'Dia Pott'	In de People Service is de naam aangepast van 'Dia Potts' naar 'Dia Pott'	Ja
De People Service maakt een Event aan waarin staat dat People Item 1 is aangepast van 'Dia Potts' naar 'Dia Pott' en stuurt deze aan de RabbitMQ Event Bus	In de RabbitMQ Event bus staat een event waarin staat dat People Item 1 is aangepast van 'Dia Potts' naar 'Dia Pott'	Ja
De Knowledge Service ontvangt de Job die People Item 1 aanpast van 'Dia Potts' naar 'Dia Pott' van de RabbitMQ event Bus en slaat deze op in de wachtrij.	De Knowledge Service heeft het event/job opgenomen die People Item 1 aanpast van 'Dia Potts' naar 'Dia Pott' in zijn wachtrij	Ja
De Knowledge Service start de uitvoering van de update om People Item 1 van 'Dia Potts' naar 'Dia Pott' aan te passen.	-	Ja
MongoDB past in alle documenten de auteur met People item 1 de naam aan van 'Dia Potts' naar 'Dia Pott'.	Alle auteurs in de documenten in de Knowledge Service MongoDB database zijn aangepast van 'Dia Potts' naar 'Dia Pott'	Ja
De Knowledge Service verwijdert de Job die People item 1 aanpast uit de wachtrij.	De job is uit de wachtrij verwijderd. Alle documenten in de MongoDB database die een auteur met People item 1 hebben hebben de nieuwe naam 'Dia Pott'.	Ja

Bij het uitvoeren van de test zijn geen bijzonderheden geconstateerd. Alle resultaten kwamen overeen met het verwachte resultaat.

2.4.2 Testcase 2: Uitvallen van de Knowledge Service

In deze situatie valt de Knowledge Service uit tijdens het uitvoeren van een Job en wordt deze daarna weer opgestart om de Job opnieuw uit te voeren. De stappen in deze fysieke testcase zijn beschreven in de onderstaande tabel. Bij elke stap is opgeschreven of het verwacht resultaat overeenkomt met het geconstateerde resultaat tijdens het uitvoeren van de test.

Stap	Verwacht resultaat	Klopt?
De gebruiker wijzigt de naam van het People Item 1 van 'Dai Potts' naar 'Dai Pott'.	-	Ja
De People Service wijzigt in zijn database People Item 1 van 'Dai Potts' naar 'Dia Pott'	In de People Service is de naam aangepast van 'Dia Potts' naar 'Dia Pott'	Ja
De People Service maakt een Event aan waarin staat dat People Item 1 is aangepast van 'Dia Potts' naar 'Dia Pott' en stuurt deze aan de RabbitMQ Event Bus	In de RabbitMQ Event bus staat een event waarin staat dat People Item 1 is aangepast van 'Dia Potts' naar 'Dia Pott'	Ja
De Knowledge Service ontvangt de Job die People Item 1 aanpast van 'Dia Potts' naar 'Dia Pott' van de RabbitMQ event Bus en slaat deze op in de wachtrij.	De Knowledge Service heeft het event/job opgenomen die People Item 1 aanpast van 'Dia Potts' naar 'Dia Pott' in zijn wachtrij	Ja
De Knowledge Service start de uitvoering van de update om People Item 1 van 'Dia Potts' naar 'Dia Pott' aan te passen.	-	Ja
De Knowledge Service wordt uitgeschakeld.	De Knowledge Service wordt uitgeschakeld. Mogelijk is wanneer de Job al is verstuurd naar MongoDB de database inconsistent geraakt	Ja
De Knowledge Service wordt weer opgestart.	De Knowledge Service is weer beschikbaar	Ja
De Knowledge Service ontvangt de Job die People Item 1 aanpast van 'Dia Potts' naar 'Dia Pott' van de RabbitMQ event Bus en slaat deze op in de wachtrij.	De Knowledge Service ontvangt opnieuw hetzelfde event als hiervoor omdat deze nog niet uit de wachtrij is verwijderd.	Ja
De Knowledge Service start de uitvoering van de update om People Item 1 van 'Dia Potts' naar 'Dia Pott' aan te passen.	-	Ja
MongoDB past in alle documenten de auteur met People item 1 de naam aan van 'Dia Potts' naar 'Dia Pott'.	Alle auteurs in de documenten in de Knowledge Service MongoDB database zijn aangepast van 'Dia Potts' naar 'Dia Pott'	Ja
De Knowledge Service verwijdert de Job die People item 1 aanpast uit de wachtrij.	De job is uit de wachtrij verwijderd. Alle documenten in de MongoDB database die een auteur met People item 1 hebben hebben de nieuwe naam 'Dia Pott'.	Ja

Bij het uitvoeren van de test zijn geen bijzonderheden geconstateerd. Alle resultaten kwamen overeen met het verwachte resultaat.

2.4.3 Testcase 3: MongoDB valt uit

In deze situatie valt MongoDB uit tijdens de verwerking van een Job. Bij elke stap is opgeschreven of het verwacht resultaat overeenkomt met het geconstateerde resultaat tijdens het uitvoeren van de test. De stappen in deze fysieke testcase zijn beschreven in de onderstaande tabel. Bij elke stap is opgeschreven of het verwacht resultaat overeenkomt met het geconstateerde resultaat tijdens het uitvoeren van de test.

Stap	Verwacht resultaat	Klopt?
De gebruiker wijzigt de naam van het People Item 1 van 'Dai Potts' naar 'Dai Pott'.	-	Ja
De People Service wijzigt in zijn database People Item 1 van 'Dai Potts' naar 'Dia Pott'	In de People Service is de naam aangepast van 'Dia Potts' naar 'Dia Pott'	Ja
De People Service maakt een Event aan waarin staat dat People Item 1 is aangepast van 'Dia Potts' naar 'Dia Pott' en stuurt deze naar de RabbitMQ Event Bus	In de RabbitMQ Event bus staat een event waarin staat dat People Item 1 is aangepast van 'Dia Potts' naar 'Dia Pott'	Ja
De Knowledge Service ontvangt de Job die People Item 1 aanpast van 'Dia Potts' naar 'Dia Pott' van de RabbitMQ event Bus en slaat deze op in de wachtrij.	De Knowledge Service heeft het event/job opgenomen die People Item 1 aanpast van 'Dia Potts' naar 'Dia Pott' in zijn wachtrij	Ja
De Knowledge Service start de uitvoering van de update om People Item 1 van 'Dia Potts' naar 'Dia Pott' aan te passen.	-	Ja
MongoDB past in alle documenten de auteur met People item 1 de naam aan van 'Dia Potts' naar 'Dia Pott'.	Alle auteurs in de documenten in de Knowledge Service MongoDB database zijn aangepast van 'Dia Potts' naar 'Dia Pott'	Ja
MongoDB wordt uitgeschakeld tijdens het uitvoeren van de job.	De Job faalt en de Knowledge Service blijft het proberen. MongoDB is mogelijk in een inconsistente staat waarbij een deel van de documenten wel is aangepast en een deel van de documenten nog niet.	Ja
MongoDB wordt weer ingeschakeld.	MongoDB zal opstarten.	Ja
De Knowledge Service start de uitvoering van de update om People Item 1 van 'Dia Potts' naar 'Dia Pott' aan te passen opnieuw.	-	Ja
MongoDB past in alle documenten de auteur met People item 1 de naam aan van 'Dia Potts' naar 'Dia Pott'.	Alle auteurs in de documenten in de Knowledge Service MongoDB database zijn aangepast van 'Dia Potts' naar 'Dia Pott'	Ja
De Knowledge Service verwijdert de Job die People item 1 aanpast uit de wachtrij.	De job is uit de wachtrij verwijderd. Alle documenten in de MongoDB database die een auteur met People item 1 hebben hebben de nieuwe naam 'Dia Pott'.	Ja

Bij het uitvoeren van de test zijn geen bijzonderheden geconstateerd. Alle resultaten kwamen overeen met het verwachte resultaat.

Nadat MongoDB uitviel is gekeken of de database inconsistent is geraakt. Dit bleek het geval. Van de 2000 documenten die geüpdate hadden moeten zijn, bleken maar ongeveer 900 documenten geüpdate. Dit is mogelijk omdat MongoDB is uitgevallen tijdens het uitvoeren van de update. Toen MongoDB opnieuw werd opgestart werd de Job opnieuw uitgevoerd en was MongoDB weer in een consistente staat.

2.4.4 Testcase 4: De Knowledge Service valt twee keer uit bij het uitvoeren van een Job.

In deze situatie valt de Knowledge Service uit tijdens de uitvoering van een Job. Hierna wordt de Knowledge Service opnieuw opgestart maar valt opnieuw uit. Hierna start de Knowledge Service wel correct opnieuw op en wordt de Job opnieuw uitgevoerd. De stappen in deze fysieke testcase zijn beschreven in de onderstaande tabel. Bij elke stap is opgeschreven of het verwacht resultaat overeenkomt met het geconstateerde resultaat tijdens het uitvoeren van de test.

Stap	Verwacht resultaat	Klopt?
De gebruiker wijzigt de naam van het People Item 1 van 'Dai Potts' naar 'Dai Pott'.	-	Ja
De People Service wijzigt in zijn database People Item 1 van 'Dai Potts' naar 'Dia Pott'	In de People Service is de naam aangepast van 'Dia Potts' naar 'Dia Pott'	Ja
De People Service maakt een Event aan waarin staat dat People Item 1 is aangepast van 'Dia Potts' naar 'Dia Pott' en stuurt deze aan de RabbitMQ Event Bus	In de RabbitMQ Event bus staat een event waarin staat dat People Item 1 is aangepast van 'Dia Potts' naar 'Dia Pott'	Ja
De Knowledge Service ontvangt de Job die People Item 1 aanpast van 'Dia Potts' naar 'Dia Pott' van de RabbitMQ event Bus en slaat deze op in de wachtrij.	De Knowledge Service heeft het event/job opgenomen die People Item 1 aanpast van 'Dia Potts' naar 'Dia Pott' in zijn wachtrij	Ja
De Knowledge Service start de uitvoering van de update om People Item 1 van 'Dia Potts' naar 'Dia Pott' aan te passen.	-	Ja
De Knowledge Service wordt uitgeschakeld.	De Knowledge Service wordt uitgeschakeld. Mogelijk is wanneer de Job al is verstuurd naar MongoDB de database inconsistent geraakt	Ja
De Knowledge Service wordt weer opgestart.	De Knowledge Service is weer beschikbaar	Ja
De Knowledge Service wordt uitgeschakeld.	De Knowledge Service wordt uitgeschakeld. Mogelijk is wanneer de Job al is verstuurd naar MongoDB de database inconsistent geraakt	Ja
De Knowledge Service wordt weer opgestart.	De Knowledge Service is weer beschikbaar	Ja
De Knowledge Service ontvangt de Job die People Item 1 aanpast van 'Dia Potts' naar 'Dia Pott' van de RabbitMQ event Bus en slaat deze op in de wachtrij.	De Knowledge Service ontvangt opnieuw hetzelfde event als hiervoor omdat deze nog niet uit de wachtrij is verwijderd.	Ja
De Knowledge Service start de uitvoering van de update om People Item 1 van 'Dia Potts' naar 'Dia Pott' aan te passen.	-	Ja
MongoDB past in alle documenten de auteur met People item 1 de naam aan van 'Dia Potts' naar 'Dia Pott'.	Alle auteurs in de documenten in de Knowledge Service MongoDB database zijn aangepast van 'Dia Potts' naar 'Dia Pott'	Ja
De Knowledge Service verwijdert de Job die People item 1 aanpast uit de wachtrij.	De job is uit de wachtrij verwijderd.	Ja

Alle resultaten kwamen overeen met het verwachte resultaat.

2.4.5 Testcase 5: MongoDB valt uit en daarna valt de Knowledge Service uit

In deze situatie valt MongoDB uit tijdens het uitvoeren van een update. Wanneer MongoDB weer beschikbaar is valt de Knowledge Service uit. Deze wordt hierna opnieuw opgestart waardoor de Job uiteindelijk slaagt. De stappen in deze fysieke testcase zijn beschreven in de onderstaande tabel. Bij elke stap is opgeschreven of het verwacht resultaat overeenkomt met het geconstateerde resultaat tijdens het uitvoeren van de test.

Stap	Verwacht resultaat	Klopt?
De gebruiker wijzigt de naam van het People Item 1 van 'Dai Potts' naar 'Dai Pott'.	-	Ja
De People Service wijzigt in zijn database People Item 1 van 'Dai Potts' naar 'Dia Pott'	In de People Service is de naam aangepast van 'Dia Potts' naar 'Dia Pott'	Ja
De People Service maakt een Event aan waarin staat dat People Item 1 is aangepast van 'Dia Potts' naar 'Dia Pott' en stuurt deze aan de RabbitMQ Event Bus	In de RabbitMQ Event bus staat een event waarin staat dat People Item 1 is aangepast van 'Dia Potts' naar 'Dia Pott'	Ja
De Knowledge Service ontvangt de Job die People Item 1 aanpast van 'Dia Potts' naar 'Dia Pott' van de RabbitMQ event Bus en slaat deze op in de wachtrij.	De Knowledge Service heeft het event/job opgenomen die People Item 1 aanpast van 'Dia Potts' naar 'Dia Pott' in zijn wachtrij	Ja
De Knowledge Service start de uitvoering van de update om People Item 1 van 'Dia Potts' naar 'Dia Pott' aan te passen.	-	Ja
MongoDB past in alle documenten de auteur met People item 1 de naam aan van 'Dia Potts' naar 'Dia Pott'.	Alle auteurs in de documenten in de Knowledge Service MongoDB database zijn aangepast van 'Dia Potts' naar 'Dia Pott'	Ja
MongoDB wordt uitgeschakeld tijdens het uitvoeren van de job.	De Job faalt en de Knowledge Service blijft het proberen. MongoDB is mogelijk in een inconsistente staat waarbij een deel van de documenten wel is aangepast en een deel van de documenten nog niet.	Ja
MongoDB wordt weer ingeschakeld.	MongoDB zal opstarten.	Ja
De Knowledge Service start de uitvoering van de update om People Item 1 van 'Dia Potts' naar 'Dia Pott' aan te passen opnieuw.	-	Ja
MongoDB past in alle documenten de auteur met People item 1 de naam aan van 'Dia Potts' naar 'Dia Pott'.	Alle auteurs in de documenten in de Knowledge Service MongoDB database zijn aangepast van 'Dia Potts' naar 'Dia Pott'	Ja
De Knowledge Service verwijdert de Job die People item 1 aanpast uit de wachtrij.	De job is uit de wachtrij verwijderd. Alle documenten in de MongoDB database die een auteur met People item 1 hebben hebben de nieuwe naam 'Dia Pott'.	Ja
De Knowledge Service wordt uitgeschakeld.	De Knowledge Service wordt uitgeschakeld. Mogelijk is wanneer de Job al is verstuurd naar MongoDB de database inconsistent geraakt	Ja
De Knowledge Service wordt weer opgestart.	De Knowledge Service is weer beschikbaar	Ja

Stap	Verwacht resultaat	Klopt?
De Knowledge Service ontvangt de Job die People Item 1 aanpast van 'Dia Potts' naar 'Dia Pott' van de RabbitMQ event Bus en slaat deze op in de wachtrij.	De Knowledge Service ontvangt opnieuw hetzelfde event als hiervoor omdat deze nog niet uit de wachtrij is verwijderd.	Ja
De Knowledge Service start de uitvoering van de update om People Item 1 van 'Dia Potts' naar 'Dia Pott' aan te passen.	-	Ja
MongoDB past in alle documenten de auteur met People item 1 de naam aan van 'Dia Potts' naar 'Dia Pott'.	Alle auteurs in de documenten in de Knowledge Service MongoDB database zijn aangepast van 'Dia Potts' naar 'Dia Pott'	Ja
De Knowledge Service verwijdert de Job die People item 1 aanpast uit de wachtrij.	De job is uit de wachtrij verwijderd. Alle documenten in de MongoDB database die een auteur met People item 1 hebben hebben de nieuwe naam 'Dia Pott'.	Ja

Bij het uitvoeren van de test zijn geen bijzonderheden geconstateerd. Alle resultaten kwamen overeen met het verwachte resultaat.

2.5 Resultaat

Uit de fysieke testgevallen die hierboven zijn besproken is gebleken dat de Job Queue naar behoren functioneert. In alle testcases is de data van de auteurs in de Knowledge Service consistent geworden met de data uit de People Service.

3. Use Case Testing

3.1 Aanpak

In dit hoofdstuk wordt de functionaliteit van het Proof-of-Concept getest. Er is gezocht naar een geschikte testvorm van TMap Next. In het document 'Overzicht toegepaste testvormen'[4] is te vinden welke kwaliteitsattributen gelden voor welk testvormen. Er is gekozen voor een Use Case test. Bij een Use Case test worden de kwaliteitsattributen functionaliteit en inpasbaarheid getest.

Als Testbasis worden de Use Cases uit het Functioneel ontwerp[5] en beide varianten van het Proof-of-Concept gebruikt.

Door de functionaliteit van het Proof-of-Concept te testen kan worden bewezen of de Use Cases die belangrijk zijn voor de werking van knowNow ook werken in combinatie met MongoDB.

De bedoeling van de Use Case testen is niet om te testen of het Proof-of-Concept volledig functioneert maar of MongoDB kan werken voor knowNow. Niet alle Use Cases worden getest met een testmethode. Alleen de Use Cases van de Knowledge Service worden getest (U.1 t/m U.5). De Use Cases van de People Service (U.7 t/m U.11) worden niet getest met een testmethode. Dit is gedaan omdat de functionaliteiten erg veel op elkaar lijken. Wanneer alle Use Cases van de Knowledge Service met MongoDB werken, zullen ook alle Use Cases van de People Service met MongoDB kunnen werken. Het testen of een wijziging van een People Item wordt doorgevoerd in de Knowledge Items is al gedaan met de Proces Cyclus Test. Wanneer de niet-geteste Use Cases toch niet blijken te werken ligt dat niet aan MongoDB maar aan fouten in het Proof-of-Concept zelf. Door niet alle Use Cases te testen wordt er tijd bespaard maar kan toch worden bewezen of MongoDB een alternatief is voor SQL server voor knowNow.

Van elke Use Case worden eerst een of meerdere logische testen opgezet en daarna fysieke testen. Deze fysieke testen worden uitgevoerd op beide varianten van het Proof-of-Concept, het Proof-of-Concept met hiërarchisch documentschema en het Proof-of-Concept met relationeel documentschema.

Na het uitvoeren van de fysieke testen kunnen eventueel gevonden fouten waar mogelijk worden hersteld en kan worden gesteld of de Use Cases volledig aan de verwachtingen voldoen. Wanneer dat het geval is kan worden gesteld dat MongoDB een mogelijke database is voor knowNow.

[4] TMap, "Overzicht Toegepaste testvormen"
http://www.mentorum.nl/docs/Traindocs/Overzicht_Toegepaste_testvormen.pdf
[5] Functioneel Ontwerp, Bijlage F

3.2 U.1: Toevoegen van Knowledge Item

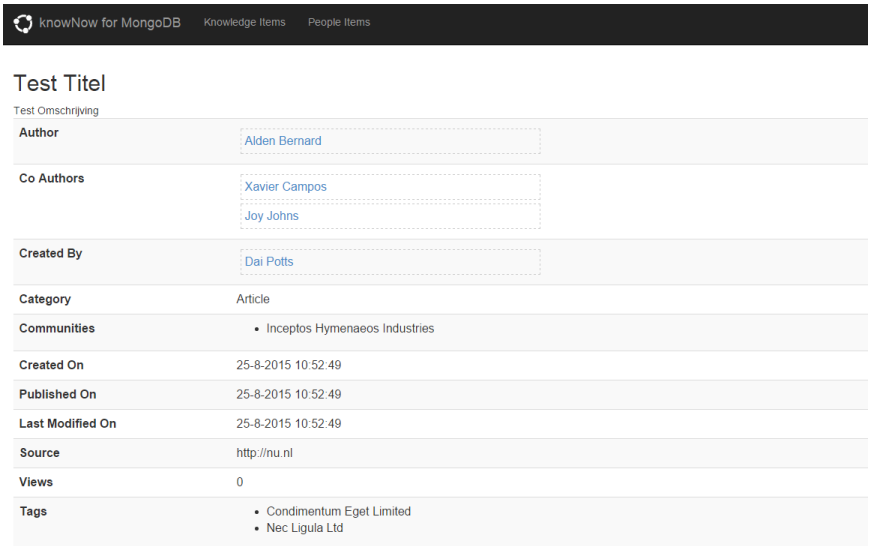
3.2.1 Logische testen

Logische testcase U.1.1 **Main flow**	Uitzondering	Gewenst resultaat	Werkelijk resultaat	Opmerkingen
1. Gebruiker klikt op knop om Knowledge Item toe te voegen		Knowledge Service retourneert formulier	Knowledge Service retourneert formulier	
3-12. Gebruiker vult alle velden in				
13. **Gebruiker verstuurd formulier**		De Knowledge Service controleert of alle velden zijn ingevuld	De Knowledge Service controleert of alle velden zijn ingevuld	
		De Knowledge Service maakt nieuw Knowledge Item aan.	De Knowledge Service maakt nieuw Knowledge Item aan.	

Logische testcase U.1.2 **Main flow**	Uitzondering	Gewenst resultaat	Werkelijk resultaat	Opmerkingen
1. Gebruiker klikt op knop om Knowledge Item toe te voegen		Knowledge Service retourneert formulier	Knowledge Service retourneert formulier	
3-12. Gebruiker vult niet alle velden in				
13. **Gebruiker verstuurd formulier**		De Knowledge Service controleert of alle velden zijn ingevuld	De Knowledge Service controleert of alle velden zijn ingevuld	
	[1]	Formulier is invalide, Knowledge Service geeft foutmelding terug. Gebruiker kan formulier wijzigen en opnieuw verzenden.	Formulier is invalide, Knowledge Service geeft foutmelding terug. Gebruiker kan formulier wijzigen en opnieuw verzenden.	

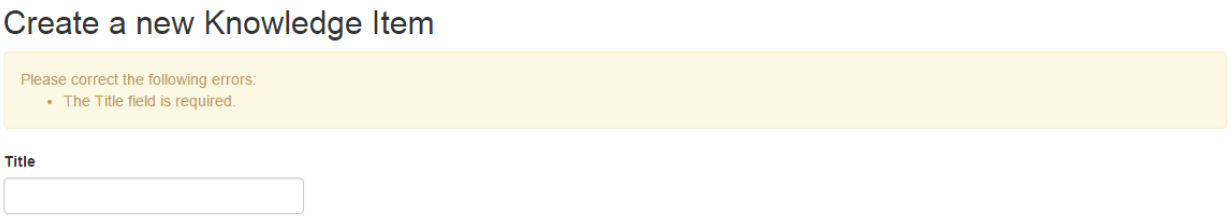
3.2.2 Fysieke testen

Fysieke testcase U.1.1 **Main flow**	**Uitgevoerd door:** Tom Keim **Gewenst resultaat**	**Test geslaagd:** PoC hiërarchisch documentschema: JA PoC relationeel documentschema: JA **Werkelijk resultaat**
1. Gebruiker klikt op knop om Knowledge Item toe te voegen	Knowledge Service retourneert formulier	Knowledge Service retourneert formulier
3-12. Gebruiker vult velden in: **Titel:** Test Titel **Omschrijving:** Test Omschrijving **Creator**: Dia Potts **Author**: Alden Bernhard **Co Authors**: Xavier Campos, Joy Johns **Tags:** Condimentum Eget Limited, Nec Ligula Ltd **Source:** http://nu.nl **Visibility**: Public **Category**: Article **Communities:** Inceptos Hymenaeos Industries		
13. **Gebruiker verstuurd formulier**	De Knowledge Service controleert of alle velden zijn ingevuld	De Knowledge Service controleert of alle velden zijn ingevuld
	De Knowledge Service maakt nieuw Knowledge Item aan met de ingevulde gegevens	De Knowledge Service maakt nieuw Knowledge Item aan met de ingevulde gegevens. (zie figuur 3-1)



Figuur 3-1. Het toegevoegde Knowledge Item

Fysieke testcase U.1.2 Main flow	**Uitgevoerd door:** Tom Keim **Gewenst resultaat**	**Test geslaagd:** PoC hiërarchisch documentschema: JA PoC relationeel documentschema: JA **Werkelijk resultaat**
1. Gebruiker klikt op knop om Knowledge Item toe te voegen	Knowledge Service retourneert formulier	Knowledge Service retourneert formulier
3-12. Gebruiker vult velden in: **Titel:** **Omschrijving:** Test Omschrijving **Creator**: Dia Potts **Author**: Alden Bernhard **Co Authors**: Xavier Campos, Joy Johns **Tags:** Condimentum Eget Limited, Nec Ligula Ltd **Source:** http://nu.nl **Visibility**: Public **Category**: Article **Communities:** Inceptos Hymenaeos Industries		
13. **Gebruiker verstuurd formulier**	De Knowledge Service controleert of alle velden zijn ingevuld	De Knowledge Service controleert of alle velden zijn ingevuld
	Formulier is invalide, Knowledge Service geeft foutmelding terug. Gebruiker kan formulier wijzigen en opnieuw verzenden.	Formulier is invalide, Knowledge Service geeft foutmelding terug. Gebruiker kan formulier wijzigen en opnieuw verzenden. (zie figuur 3-2)



Figuur 3-2. De titel was leeggelaten. Systeem geeft foutmelding

3.3 U.2: Wijzigen van Knowledge Item

3.3.1 Logische testen

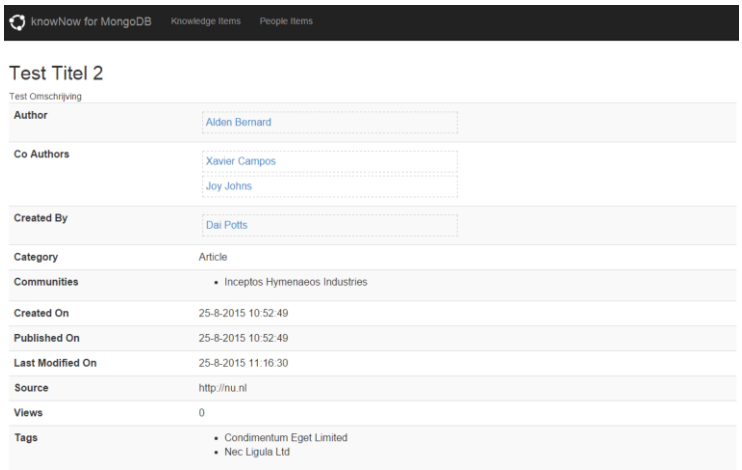
Logische testcase U.2.1 Main flow	Uitzondering	Gewenst resultaat	Werkelijk resultaat	Opmerkingen
1. Gebruiker klikt op knop om een Knowledge Item aan te passen		Knowledge Service controleert of Knowledge Item bestaat en niet verwijderd is	Knowledge Service controleert of Knowledge Item bestaat en niet verwijderd is	
		Knowledge Service retourneert formulier met Knowledge Item	Knowledge Service retourneert formulier met Knowledge Item	
4. Gebruiker wijzigt de velden in het formulier				
5. Gebruiker stuurt gewijzigde formulier op		De Knowledge Service controleert of alle velden zijn ingevuld	De Knowledge Service controleert of alle velden zijn ingevuld	
		Knowledge Service wijzigt het Knowledge Item	Knowledge Service wijzigt het Knowledge Item	

Logische testcase U.2.2 Main flow	Uitzondering	Gewenst resultaat	Werkelijk resultaat
1. Gebruiker klikt op knop om een Knowledge Item aan te passen		Knowledge Service controleert of Knowledge Item bestaat en niet verwijderd is	Knowledge Service controleert of Knowledge Item bestaat en niet verwijderd is
	[1]	Het Knowledge Item bestaat niet of is verwijderd, Knowledge Service geeft foutmelding	Het Knowledge Item bestaat niet of is verwijderd, Knowledge Service geeft foutmelding

Logische testcase U.2.3 Main flow	Uitzondering	Gewenst resultaat	Werkelijk resultaat	Opmerkingen
1. Gebruiker klikt op knop om een Knowledge Item aan te passen		Knowledge Service controleert of Knowledge Item bestaat en niet verwijderd is	Knowledge Service controleert of Knowledge Item bestaat en niet verwijderd is	
		Knowledge Service retourneert formulier met Knowledge Item	Knowledge Service retourneert formulier met Knowledge Item	
4. Gebruiker wijzigt de velden in het formulier				De gebruiker wijzigt het formulier naar een invalide staat.
5. Gebruiker stuurt gewijzigde formulier op		De Knowledge Service controleert of alle velden zijn ingevuld	De Knowledge Service controleert of alle velden zijn ingevuld	
	[2]	Knowledge Item is onjuist, Knowledge Service geeft foutmelding terug.	Knowledge Item is onjuist, Knowledge Service geeft foutmelding terug.	

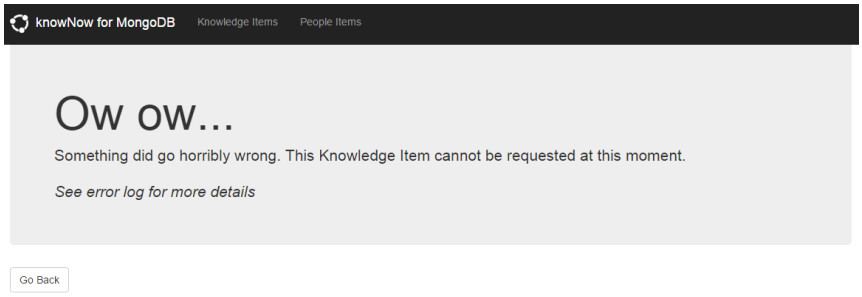
3.3.2 Fysieke testen

Fysieke testcase U.2.1 **Main flow**	**Uitgevoerd door:** Tom Keim **Gewenst resultaat**	**Test geslaagd:** PoC hiërarchisch documentschema: JA PoC relationeel documentschema: JA **Werkelijk resultaat**
1. Gebruiker klikt op knop om een Knowledge Item aan te passen bij Knowledge Item met titel 'Test Titel'	Knowledge Service controleert of Knowledge Item bestaat en niet verwijderd is	Knowledge Service controleert of Knowledge Item bestaat en niet verwijderd is
	Knowledge Service retourneert formulier met Knowledge Item	Knowledge Service retourneert formulier met Knowledge Item
4. Gebruiker past titel aan **Titel:** Test Titel 2		
5. **Gebruiker stuurt gewijzigde formulier op**	De Knowledge Service controleert of alle velden zijn ingevuld	De Knowledge Service controleert of alle velden zijn ingevuld
	Knowledge Service wijzigt het Knowledge Item	Knowledge Service wijzigt het Knowledge Item. (zie figuur 3-3)



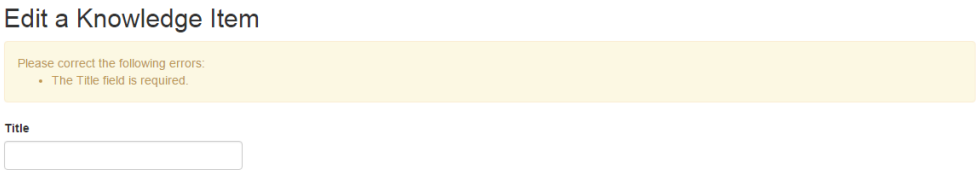
Figuur 3-3. De titel van het Knowledge Item is gewijzigd

Fysieke testcase U.2.2 Main flow	**Uitgevoerd door:** Tom Keim **Gewenst resultaat**	**Test geslaagd:** PoC hiërarchisch documentschema: JA PoC relationeel documentschema: JA **Werkelijk resultaat**
1. Gebruiker klikt op knop om een Knowledge Item aan te passen maar het Knowledge Item is inmiddels al verwijderd.	Knowledge Service controleert of Knowledge Item bestaat en niet verwijderd is	Knowledge Service controleert of Knowledge Item bestaat en niet verwijderd is
	Het Knowledge Item bestaat niet of is verwijderd, Knowledge Service geeft foutmelding	Het Knowledge Item bestaat niet of is verwijderd, Knowledge Service geeft foutmelding (zie figuur 3-4)



Figuur 3-4. Foutmelding na opvragen niet bestaand Knowledge Item

Fysieke testcase U.2.3 Main flow	**Uitgevoerd door:** Tom Keim **Gewenst resultaat**	**Test geslaagd:** PoC hiërarchisch documentschema: JA PoC relationeel documentschema: JA **Werkelijk resultaat**
1. Gebruiker klikt op knop om een Knowledge Item aan te passen bij Knowledge Item met titel 'Test Titel'	Knowledge Service controleert of Knowledge Item bestaat en niet verwijderd is	Knowledge Service controleert of Knowledge Item bestaat en niet verwijderd is
	Knowledge Service retourneert formulier met Knowledge Item	Knowledge Service retourneert formulier met Knowledge Item
4. Gebruiker past titel aan **Titel:**		
5. **Gebruiker stuurt gewijzigde formulier op**	De Knowledge Service controleert of alle velden zijn ingevuld	De Knowledge Service controleert of alle velden zijn ingevuld
	Knowledge Item is onjuist, Knowledge Service geeft foutmelding terug.	Knowledge Item is onjuist, Knowledge Service geeft foutmelding terug. (zie figuur 3-5)



Figuur 3-5. Tijdens het wijzigen is de titel weggehaald. Systeem geeft foutmelding

3.4 U.3. Verwijderen van Knowledge Item

3.4.1 Logische testen

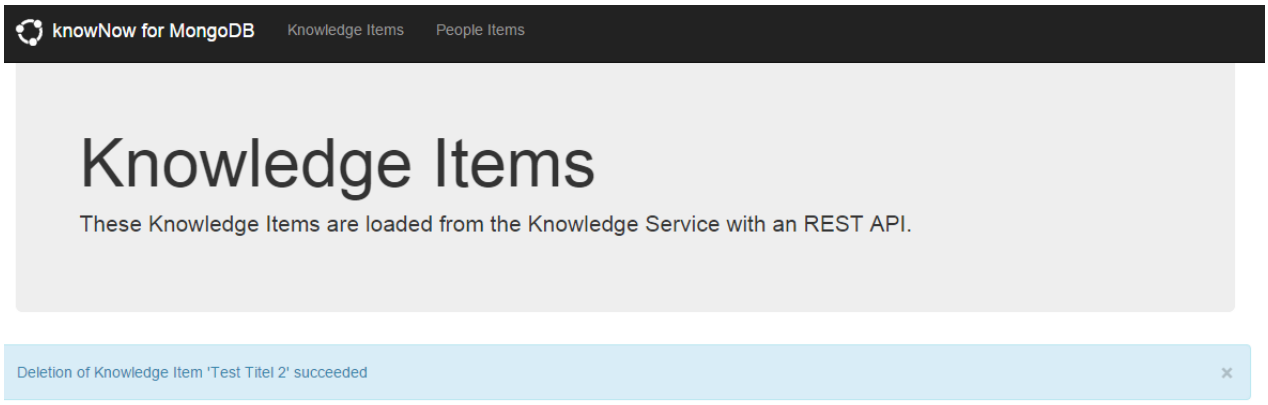
Logische testcase U.3.1 Main flow	Uitzondering	Gewenst resultaat	Werkelijk resultaat	Opmerkingen
1. Gebruiker klikt op knop voor verwijderen van Knowledge Item.		Knowledge Service controleert of Knowledge Item bestaat en niet verwijderd is	Knowledge Service controleert of Knowledge Item bestaat en niet verwijderd is	
		Knowledge Service vraagt gebruiker om bevestiging.	Knowledge Service vraagt gebruiker om bevestiging.	
4. Gebruiker bevestigd		Knowledge Service slaat het Knowledge Item op als verwijderd	Knowledge Service slaat het Knowledge Item op als verwijderd	

Logische testcase U.3.2 Main flow	Uitzondering	Gewenst resultaat	Werkelijk resultaat	Opmerkingen
1. Gebruiker klikt op knop voor verwijderen van Knowledge Item.		Knowledge Service controleert of Knowledge Item bestaat en niet verwijderd is	Knowledge Service controleert of Knowledge Item bestaat en niet verwijderd is	
	[1]	Het Knowledge Item bestaat niet of is al verwijderd, Knowledge Service geeft foutmelding	Het Knowledge Item bestaat niet of is al verwijderd, Knowledge Service geeft foutmelding	

Logische testcase U.3.3 Main flow	Uitzondering	Gewenst resultaat	Werkelijk resultaat	Opmerkingen
1. Gebruiker klikt op knop voor verwijderen van Knowledge Item.		Knowledge Service controleert of Knowledge Item bestaat en niet verwijderd is	Knowledge Service controleert of Knowledge Item bestaat en niet verwijderd is	
		Knowledge Service vraagt gebruiker om bevestiging.	Knowledge Service vraagt gebruiker om bevestiging.	
4. Gebruiker bevestigd niet.	**[2]**	Knowledge Item is niet verwijderd	Knowledge Item is niet verwijderd	

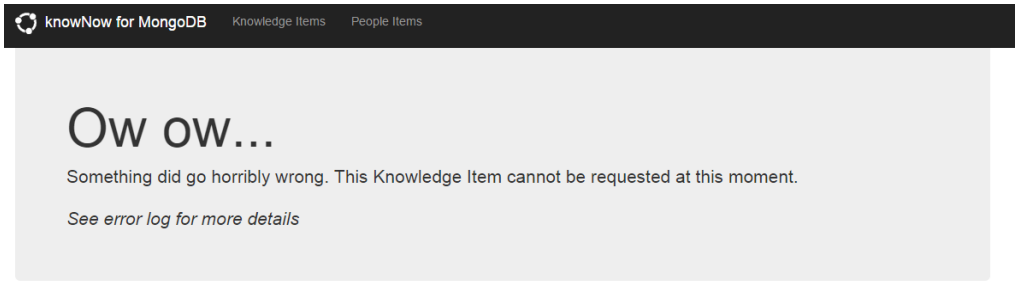
3.4.2 Fysieke testen

Fysieke testcase U.3.1 Main flow	**Uitgevoerd door:** Tom Keim **Gewenst resultaat**	**Test geslaagd:** PoC hiërarchisch documentschema: JA PoC relationeel documentschema: JA **Werkelijk resultaat**
1. . Gebruiker klikt op knop voor verwijderen van Knowledge Item met titel 'Test Titel 2'	Knowledge Service controleert of Knowledge Item bestaat en niet verwijderd is	Knowledge Service controleert of Knowledge Item bestaat en niet verwijderd is
	Knowledge Service vraagt gebruiker om bevestiging	Knowledge Service vraagt gebruiker om bevestiging
4. Gebruiker bevestigd verwijdering van 'Test Titel 2'	Knowledge Service slaat het Knowledge Item op als verwijderd	Knowledge Service slaat het Knowledge Item op als verwijderd (Figuur 3-6)



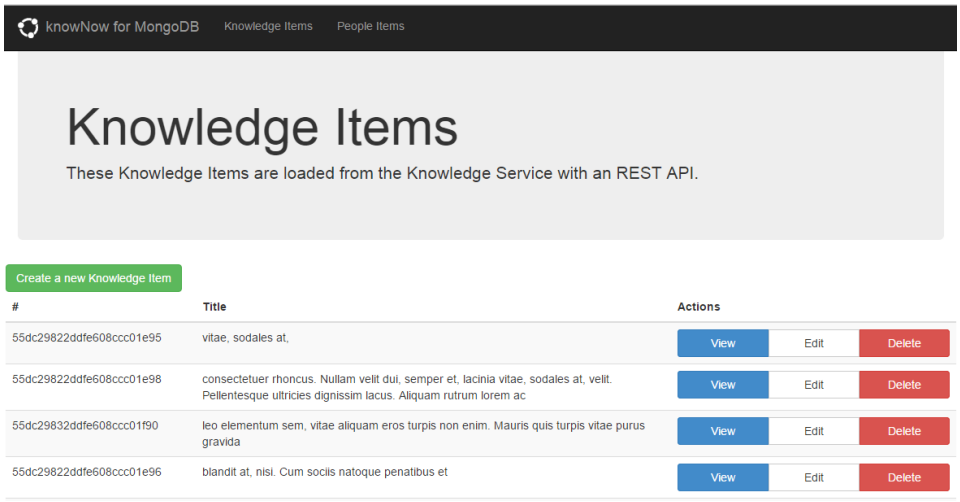
Figuur 3-6. Knowledge Item is opgeslagen als verwijderd

Fysieke testcase U.3.2 Main flow	**Uitgevoerd door:** Tom Keim **Gewenst resultaat**	**Test geslaagd:** PoC hiërarchisch documentschema: JA PoC relationeel documentschema: JA **Werkelijk resultaat**
1. . Gebruiker klikt op knop voor verwijderen van Knowledge Item met titel 'Test Titel 2' terwijl Knowledge Item al is verwijderd	Knowledge Service controleert of Knowledge Item bestaat en niet verwijderd is	Knowledge Service controleert of Knowledge Item bestaat en niet verwijderd is
	Het Knowledge Item bestaat niet of is al verwijderd, Knowledge Service geeft foutmelding	Het Knowledge Item bestaat niet of is al verwijderd, Knowledge Service geeft foutmelding (Figuur 3-7)



Figuur 3-7. Knowledge Service geeft foutmelding. Knowledge Item niet gevonden

Fysieke testcase U.3.3 Main flow	**Uitgevoerd door:** Tom Keim **Gewenst resultaat**	**Test geslaagd:** PoC hiërarchisch documentschema: JA PoC relationeel documentschema: JA **Werkelijk resultaat**
1. . Gebruiker klikt op knop voor verwijderen van Knowledge Item met titel 'Test Titel 2'	Knowledge Service controleert of Knowledge Item bestaat en niet verwijderd is	Knowledge Service controleert of Knowledge Item bestaat en niet verwijderd is
	Knowledge Service vraagt gebruiker om bevestiging	Knowledge Service vraagt gebruiker om bevestiging
4. Gebruiker annuleert verwijdering van 'Test Titel 2'	Knowledge Item is niet verwijderd	Knowledge Item is niet verwijderd (Figuur 3-8)



Figuur 3-8. Het Knowledge Item is niet verwijderd. De gebruiker keer terug naar de overzicht pagina

3.5 U.4 Tonen van één Knowledge Item

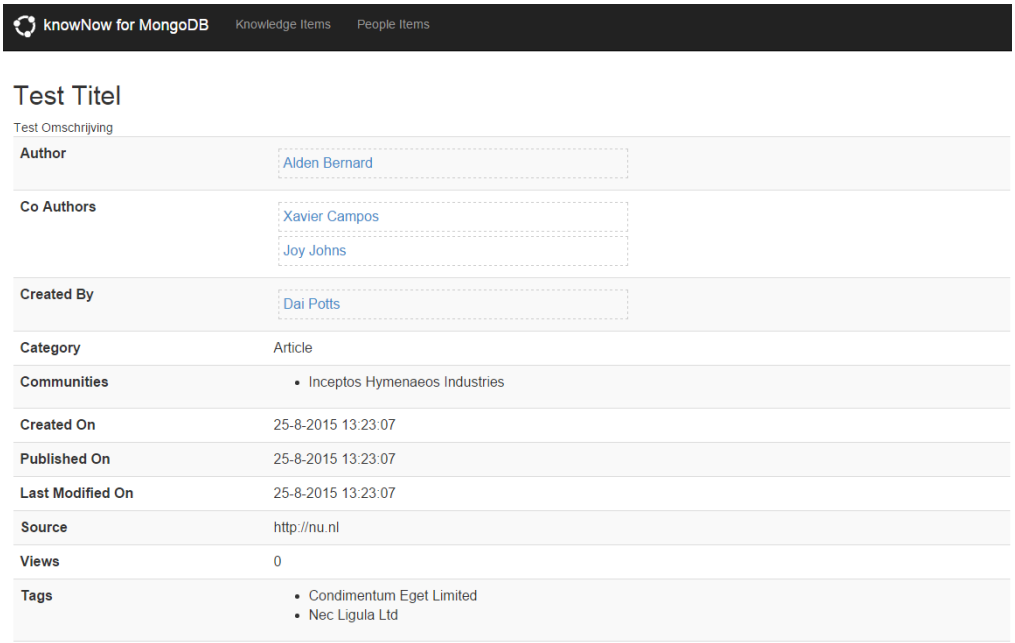
3.5.1 Logische testen

Logische testcase U.4.1 Main flow	Uitzondering	Gewenst resultaat	Werkelijk resultaat	Opmerkingen
1. Gebruiker klikt op een Knowledge Item om hem op te vragen.		Knowledge Service controleert of Knowledge Item bestaat en niet verwijderd is	Knowledge Service controleert of Knowledge Item bestaat en niet verwijderd is	
		De Knowledge Service retourneert het Knowledge Item.	De Knowledge Service retourneert het Knowledge Item.	

Logische testcase U.4.2 Main flow	Uitzondering	Gewenst resultaat	Werkelijk resultaat	Opmerkingen
1. Gebruiker klikt op een Knowledge Item om hem op te vragen.		Knowledge Service controleert of Knowledge Item bestaat en niet verwijderd is	Knowledge Service controleert of Knowledge Item bestaat en niet verwijderd is	
	[1]	Het Knowledge Item bestaat niet, Knowledge Service geeft foutmelding.	Het Knowledge Item bestaat niet, Knowledge Service geeft foutmelding.	

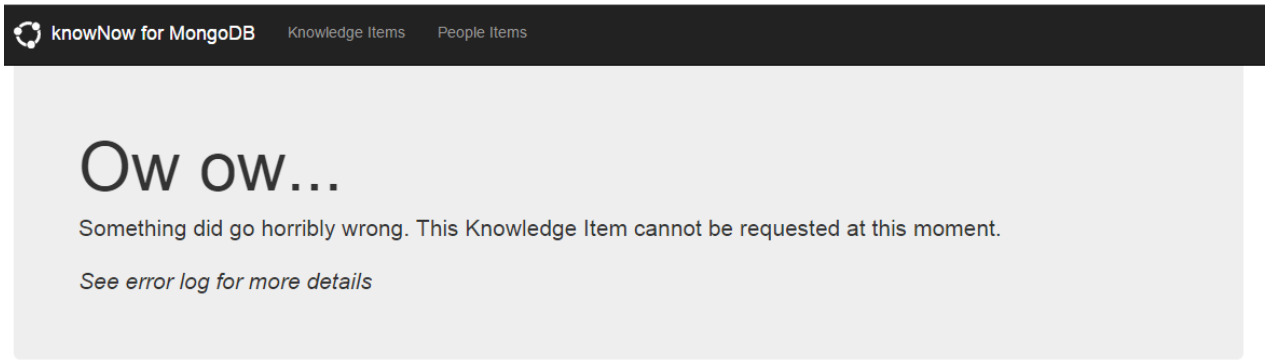
3.5.2 Fysieke Testen

Fysieke testcase U.4.1 Main flow	**Uitgevoerd door:** Tom Keim **Gewenst resultaat**	**Test geslaagd:** PoC hiërarchisch documentschema: JA PoC relationeel documentschema: JA **Werkelijk resultaat**
1. Gebruiker klikt op Knowledge Item 'Test Titel' om hem op te vragen	Knowledge Service controleert of Knowledge Item bestaat en niet verwijderd is	Knowledge Service controleert of Knowledge Item bestaat en niet verwijderd is
	De Knowledge Service retourneert het Knowledge Item 'Test Titel'	De Knowledge Service retourneert het Knowledge Item 'Test Titel' (Figuur 3-9)



Figuur 3-9. De Knowledge Service retourneert het item 'Test Titel'

Fysieke testcase U.4.2 Main flow	**Uitgevoerd door:** Tom Keim	**Test geslaagd:** PoC hiërarchisch documentschema: JA PoC relationeel documentschema: JA
	Gewenst resultaat	**Werkelijk resultaat**
1. Gebruiker klikt op Knowledge Item 'Test Titel' om hem op te vragen maar is inmiddels al verwijderd	Knowledge Service controleert of Knowledge Item bestaat en niet verwijderd is	Knowledge Service controleert of Knowledge Item bestaat en niet verwijderd is
	Het Knowledge Item bestaat niet, Knowledge Service geeft foutmelding.	Het Knowledge Item bestaat niet, Knowledge Service geeft foutmelding. (Figuur 3-10)



Figuur 3-10. Het Knowledge Item bestaat niet

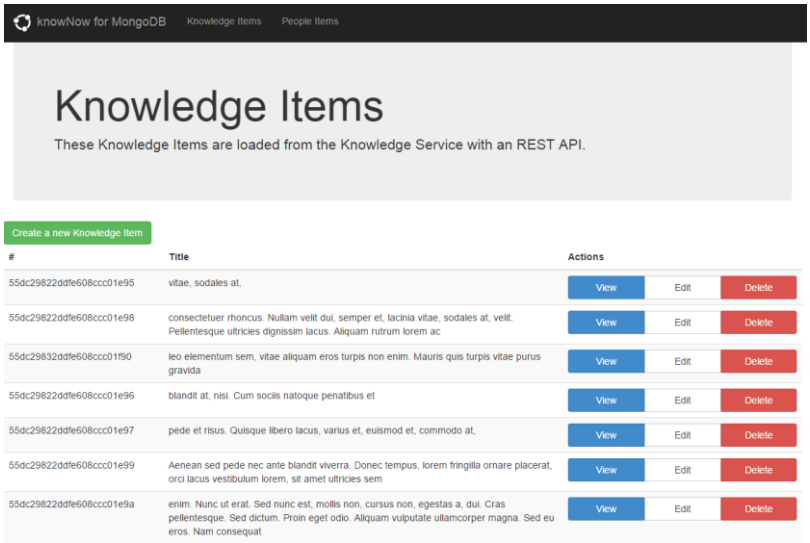
3.6 U.5. Tonen van alle Knowledge Items

3.6.1 Logische testen

Logische testcase U.5.1 Main flow	Uitzondering	Gewenst resultaat	Werkelijk resultaat	Opmerkingen
Gebruiker vraagt alle Knowledge Items op.		Knowledge Service retourneert Knowledge Items.	Knowledge Service retourneert Knowledge Items.	

3.6.2 Fysieke testen

Fysieke testcase U.5.1 Main flow	**Uitgevoerd door:** Tom Keim	**Test geslaagd:** PoC hiërarchisch documentschema: JA PoC relationeel documentschema: JA
	Gewenst resultaat	**Werkelijk resultaat**
Gebruiker vraagt alle Knowledge Items op.	Knowledge Service retourneert Knowledge Items.	Knowledge Service retourneert Knowledge Items. (Figuur 3-11)



Figuur 3-11. De Knowledge Service retourneert alle Knowledge Items

3.7 Resultaten

Alle uitgevoerde testen zijn geslaagd op beide varianten van het Proof-of-Concept. Hiermee kan worden gesteld dat MongoDB voor de belangrijkste Use Cases werkt en het een werkend alternatief is voor de SQL Server database van knowNow.

4. Unit Tests

Voor beide varianten van het Proof-of-Concept zijn enkele Unit Tests opgezet die het Proof-of-Concept testen. De Unit Tests zijn gebruikt door de afstudeerder om nieuwe, onbekende dingen te kunnen testen voordat deze in het Proof-of-Concept worden gebruikt. Geteste functionaliteiten zijn onder andere:

- De vertaling van MongoDB documenten naar C# objecten;
- De validatie van C# models;

De Unit Tests testen niet het volledige Proof-of-Concept. De belangrijke onderdelen van het Proof-of-Concept worden getest door de Use-Case Test. De Unit tests zijn alleen gebruikt zodat de afstudeerder weet dat onbekende gebruikte functionaliteiten ook daadwerkelijk functioneren als bedoeld.

```
[TestMethod]
public void FromBsonCreationDateTest()
{
    string bson = @"{""creationDate"" : ISODate(""2015-07-09T10:59:43.049Z"")}";
    var knowledge = BsonSerializer.Deserialize<KnowledgeModel>(bson);
    Assert.AreEqual(Convert.ToDateTime("9-7-2015 10:59:43.049Z"), knowledge.CreationDate);
}
```

Figuur 4-1. Een voorbeeld van een Unit Test die test of een datum goed wordt omgezet vanuit MongoDB BSON formaat

DE HAAGSE HOGESCHOOL

Info Support Solid Innovator

Bijlage I - Adviesrapport

MongoDB: Een geschikte database voor het kennismanagementsysteem knowNow?

Adviesrapport

Titel	Adviesrapport
Project/Onderwerp	MongoDB: Een geschikte database voor het kennismanagementsysteem knowNow?
Versie	1.0
Status	Definitief
Datum	01-10-2015
Bestand	Adviesrapport v1.0
Bedrijf	Info Support
Door	Tom Keim

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

Inhoudsopgave

1. Inleiding

In dit document doe ik enkele aanbevelingen aan het knowNow team met op basis van ervaringen die ik tijdens het uitvoeren van het onderzoek en bouwen van het Proof-of-Concept met MongoDB heb opgedaan. Daarnaast wordt er ook geadviseerd hoe er in MongoDB voor knowNow gebruik gemaakt kan worden van backups en wat de beste sharding key voor knowNow is.

Uiteindelijk doe ik een aanbeveling of ik MongoDB een beter alternatief vind dan SQL Server voor knowNow, en waar bij het gebruik van MongoDB op gelet moet worden door het knowNow team.

Als basis van dit document wordt de onderzoeksrapport, het Functioneel Ontwerp, het Technisch Ontwerp en beide varianten van het Proof-of-Concept gebruikt. Ik raad de lezer van dit document aan deze documenten eerst te lezen en het Proof-of-Concept eerst door te nemen voordat dit document wordt gelezen. Hierdoor zijn de diverse termen die in dit document worden gebruikt voor de lezer duidelijker.

2. Backups

MongoDB biedt door middel van Replica Sets[1] de mogelijkheid om data redundant op te slaan, wanneer er één server uitvalt kunnen andere servers deze overnemen. Replica Sets bevatten kopieën van data. Door het gebruik van Replica Sets is het mogelijk een redundante omgeving te creëren. Echter is het ook van belang om data terug zetten wanneer er wat mis gaat. Bijvoorbeeld wanneer er verkeerde wijzingen gedaan zijn aan de data of de gehele MongoDB cluster corrupt raakt. Om data terug te zetten dienen er backups gemaakt te worden.

In dit hoofdstuk wordt een advies gegeven voor de beste mogelijkheid om de data in een MongoDB database te backuppen. MongoDB kent meerdere mogelijkheden. In dit hoofdstuk worden de volgende mogelijkheden toegelicht:

- Filesystem backups
- MongoDB backuptools
- MMS backups
- Ops Manager backups

Aan de opties worden de volgende eisen gesteld:

- Voor knowNow zijn incrementele backups belangrijk. Er moeten meermaals per dag backups gemaakt kunnen worden. Wanneer elke keer de gehele database geback-upt moet worden kan dit bij de verwachte groei van knowNow in de toekomst voor problemen zorgen.
- De opdrachtgever heeft de eis gesteld dat data niet onversleuteld opgeslagen mag worden bij een derde partij.

[1] Replica sets is een manier waarmee ervoor gezorgd kan worden dat de data redundant blijft. Zie: Onderzoek “MongoDB als mogelijke vervanger van het huidige RDBMS voor het kennismanagement systeem knowNow”, 3.6.3. Replica Sets voor meer informatie.

2.1 Filesystem backups

MongoDB slaat alle data op het systeem waar het opdraait. Het is mogelijk deze bestanden te backuppen. Dit kan door de bestanden te kopiëren, of opties die besturingssysteem aanbiedt te gebruiken. Zoals Linux LVM[2] waarmee het mogelijk is gehele schijven te backuppen.

Bij deze mogelijkheid mogen er geen wijzigingen aan de data worden gemaakt wanneer er een backup wordt gemaakt[2]. Dit houdt in dat de gehele MongoDB database tijdelijk read-only wordt. Daarnaast moet de database in een valide staat zijn. Alle transacties moeten zijn afgerond en alle wijzigingen moeten opgeslagen zijn op de harde schrijf[2]. Anders ontstaat inconsistente data en gaan wijzigingen mogelijk verloren.

Bij een Sharded Cluster moet de balancer tijdelijk worden uitgeschakeld[2]. De balancer zorgt ervoor dat de data wordt verdeeld over de verschillende shards. Wanneer de balancer aan staat tijdens het maken van de backup bestaat de mogelijkheid dat deze data gaat verplaatsen en dat er duplicatie ontstaat. Mogelijk wordt data daardoor niet meegenomen in de backup.

Wanneer de Shards niet op dezelfde fysieke machine staan, dient van elke fysieke machine apart een backup gemaakt te worden.

Met deze mogelijkheid is het niet mogelijk incrementele backups te maken. Alle bestanden dienen bij elke backup opnieuw gekopieerd te worden.

Doordat alle data op de schijf wordt gebackuped, worden ook de indexen op de data meegenomen.

Voordelen	Nadelen
• Alles wordt gebackuped en is in een keer terug te zetten • Beheer ligt in eigen hand • Ook indexes worden meegenomen en hoeven na het terugzetten dus niet opnieuw gemaakt te worden.	• Tijdens het maken van de backup kan er niet naar de database worden geschreven. Hierdoor kunnen gebruikers geen nieuwe artikelen toevoegen aan knowNow. • Naast aanpassingen aan MongoDB dienen er ook aanpassingen gedaan te worden aan het besturingssysteem. • Elke fysieke server dient apart gebackuped te worden. • Geen incrementele backups. Alle data moet bij elke backup worden meegenomen. Bij een grote database kan een backup hierdoor lang duren. Dit kan voor problemen zorgen indien knowNow groeit.

[2] Linux's LVM, Met LVM (Local Volume Management) is het onder andere mogelijk snapshots te maken van schijven en deze te kopieeren naar andere (backup) media
http://docs.mongodb.org/manual/tutorial/backup-with-filesystem-snapshots/

2.2 MongoDB backup tools

MongoDB bevat commando's die het mogelijk maken om alle documenten uit de database te halen en op te slaan in BSON formaat. Deze commando's zijn *mongodump* en *mongorestore*[3]. Bij het gebruik van een Sharded Cluster hoeft in tegenstelling tot de Filesystem backup niet elke Shard apart te worden gebackuped. Het mongodump-commando hoeft per backup maar één keer te worden uitgevoerd.

Net zoals bij de Filesystem backups kunnen er met het mongodump-commando geen incrementele backups worden gemaakt. Alle data wordt bij elke backup opnieuw meegenomen, gewijzigd of niet.

Voordelen	Nadelen
• Alle documenten wordt gebackuped. • Geen backup nodig op alle shards. Het mongodump-commando hoeft maar een keer uitgevoerd te worden om alle data te backuppen. • Tijdens het maken van een backup kan er volledig gebruik worden gemaakt van de database. knowNow blijft volledig functioneel voor de gebruiker.	• Geen incrementele backups. Alle data moet bij elke backup worden meegenomen. Bij een grote database kan een backup hierdoor lang duren. Dit kan voor problemen zorgen indien knowNow groeit.

2.3 MMS Backups

MongoDB Management Service (MMS) is een cloudoplossing van Mongo, Inc.[3] waarin het mogelijk is één of meerdere MongoDB instanties te monitoren en te beheren via een cloudapplicatie.

MMS biedt de optie om data van een MongoDB server te backuppen naar de servers van Mongo, Inc. en is samen met Ops Manager de enige mogelijkheid om incrementele backups te maken van MongoDB[4].

Met MMS Backups is het mogelijk om een database terug te zetten naar elk moment in de laatste 24 uur. De granulariteit hiervan is ongeveer 15 minuten[4].

De retentie[4] van de backups is standaard als volgt, maar kan later worden aangepast door de gebruiker[4]:

- Backups om de 6 uur worden voor 2 dagen bewaard;
- Dagelijkse backups worden 1 week bewaard;
- Wekelijkse backups worden 1 maand bewaard;
- Maandelijkse backups worden 1 jaar bewaard.

De backups worden over de verbinding versleuteld verzonden naar Mongo, Inc. Echter, worden de backups onversleuteld opgeslagen op de servers van Mongo, Inc.[5].

Aan het gebruik van MMS backups zijn de volgende kosten verbonden:

Aantal servers	$50 per server [5]per maand
Backup (optioneel)	Eerste 1GB replica set gratis Daarna $2.50 per GB per maand

De precieze kosten voor knowNow zijn onbekend. Hier kan eventueel later onderzoek naar gedaan worden.

Voordelen	Nadelen
• Incrementele backups per 15 minuten. Hierdoor kan de data van knowNow vaak geback-upt worden waardoor bij problemen nooit veel data verloren gaat. • Eenvoudig in te stellen, vereist weinig configuratie • Tijdens het maken van een backup kan er volledig gebruik worden gemaakt van de database. knowNow blijft volledig functioneel voor de gebruiker.	• Er zijn kosten aan verbonden. • Data wordt onversleuteld bij Mongo, Inc. opgeslagen en kunnen mogelijk de data gebruiken. knowNow moet voldoen aan wetgeving en het onversleuteld opslaan van data bij een derde partij mag niet. Deze optie valt daarmee direct af.

[3] Mongo, Inc. is het commerciële bedrijf achter MongoDB.
[4] De tijd dat een backup bewaard wordt door MongoDB, INc.
[5] Voor MMS rekent MongoDB, Inc. config servers, arbiters en routers niet als servers voor de kosten van MMS.

2.4 Ops Manager backups

Ops Manager backups biedt hetzelfde als MMS backups. Echter, Ops Manager is on-premises[6]. Dit houdt in dat de backups worden opgeslagen op servers die in eigen beheer zijn.

Voor Ops Manager is MongoDB Enterprise Advanced nodig. Dit is de commerciële variant van MongoDB. De prijzen van MongoDB Enterprise Advanced zijn onbekend. Hiervoor dient contact opgenomen te worden met Mongo, Inc[7].

Voordelen	Nadelen
• Incrementele backups per 15 minuten. Hierdoor kan de data van knowNow vaak geback-upt worden waardoor bij problemen nooit veel data verloren gaat. • Servers in eigen beheer • Tijdens het maken van een backup kan er volledig gebruik worden gemaakt van de database. knowNow blijft volledig functioneel voor de gebruiker.	• Onbekende kosten

2.5 Conclusie

Voor knowNow zijn incrementele backups belangrijk. Er moeten meermaals per dag backups gemaakt kunnen worden. Wanneer elke keer de gehele database geback-upt moet worden kan dit bij de verwachte groei van knowNow in de toekomst voor problemen zorgen.

Alleen Ops manager en MMS bieden incrementele backups. Alleen bij die 2 mogelijkheden hoeft niet bij elke backup alle data gekopieerd te worden. Dit betekent dat backups elkaar ook sneller kunnen opvolgen omdat het backuppen van een deel van de data(incrementeel) sneller gaat dan het backuppen van alle data. Hoe groter de dataset, hoe langer de backup duurt.

De filesystem backup mogelijkheid valt af. Bij deze mogelijkheid is het niet mogelijk te schrijven naar de database wanneer de backup wordt gemaakt. Hierdoor is het voor de eindgebruiker niet mogelijk nieuwe Knowledge Items toe te voegen wanneer er een backup wordt gemaakt. knowNow zal dan bij elke backup niet meer de volledige functionaliteit kunnen aanbieden.

De MongoDB backup tools mogelijkheid is een goede optie wanneer de opdrachtgever geen gebruik wenst te maken van MMS Backups vanwege de onversleutelde data of van de Ops Manager vanwege de licentiekosten. De Backup tool biedt geen mogelijkheid voor incrementele backups en moet dus elke keer alle data opnieuw backuppen. De backupfrequentie is dan ook afhankelijk van de duur van het maken van een backup. Deze duur is op dit moment nog onbekend.

De beste opties zijn de MMS en Ops Manager backups. Deze bieden incrementele backups waardoor het mogelijk is hoog frequentie backups te maken, terwijl knowNow volledig bruikbaar blijft. Het voordeel van MMS backups is dat er geen on-premise backup architectuur opgezet moet worden. Het nadeel is echter de kosten per GB aan opslag en de opslag van onversleutelde data bij een externe partij (Mongo, Inc.). Omdat door de opdrachtgever de eis is gesteld dat data niet onversleuteld opgeslagen mag worden bij een derde partij valt deze optie af.

Daardoor raad ik het gebruik van Ops Manager aan. Hierbij wordt de backup architectuur in eigen beheer genomen. Hiervoor is wel ervaring nodig en meer configuratie. Ook dient er rekening gehouden te worden met de kosten voor de aanschaf van de commerciële MongoDB licentie. Deze kosten zijn onbekend en indien de opdrachtgever dit wilt weten dient er contact opgenomen te worden met Mongo, Inc. die een aanbod zullen doen.

3. Schaling

Dit hoofdstuk beschrijft wat nodig is om MongoDB te schalen voor knowNow. MongoDB heeft op zijn eigen website enkele aanbevelingen staan wanneer er geschaald moet worden[13]. Zij raden om vooral niet te vroeg op te schalen en alleen te schalen door nieuwe servers toe te voegen wanneer dit ook daadwerkelijk nodig. Bijvoorbeeld wanneer de opslagruimte bijna vol is of het ramgeheugen bijna volledig wordt gebruikt.

Er is geen onderzoek gedaan naar wat knowNow nodig heeft aan servers. Door nieuwe inzichten tijdens het uitvoeren van het afstudeerproject is hier geen tijd meer voor gevonden. Echter, is er wel een onderzoek gedaan naar de beste keuze van een Sharding Key voor knowNow. Een Sharding Key is een sleutel waarmee wordt bepaald welke data op welke server terecht moet komen wanneer er gebruik gemaakt wordt van een geschaalde omgeving.

Dit onderzoek wordt gedaan omdat de keuze van een Sharding Key dusdanig belangrijk is, het is niet later aan te passen en omdat er zo direct begonnen kan worden met schalen wanneer dat nodig is. De Sharding Key is dan al gekozen en hoeft niet eerst te worden onderzocht door het knowNow team.

3.1 Sharding Key

Een Sharding Key is een sleutel waarmee de data wordt verdeeld over de verschillende shards van MongoDB. In paragraaf 3.5.2 van het onderzoek (Bijlage B) wordt de Sharding Key en de Shards uitgelegd. Een Sharding Key zorgt ervoor dat de documenten in een collectie worden verdeeld over meerdere MongoDB instanties.

De keuze van een Sharding Key is erg belangrijk. De Sharding Key is later namelijk niet zomaar aan te passen[12].

3.1.1 Eisen

MongoDB heeft enkele eisen opgesteld waar een Sharding Key aan moet voldoen. Wanneer een veld in een document niet aan deze eisen voldoet kan dit geen Sharding Key worden.

1. De data van een Sharding Key kan later niet veranderen, het veld moet hetzelfde blijven als een document eenmaal is toegevoegd[8].
2. Een Sharding Key kan geen array zijn (Multi-key index). Wanneer een veld in een document bestaat uit meerdere waarden kan deze niet worden gebruikt[10].

3.1.2 Kandidaten

Elk veld in de opgezette collectie voor de Knowledge Items is een potentiële kandidaat voor de sharding key. Een kandidaat moet voldoen aan de eisen zoals opgesteld in 3.1.1. De velden van de Knowledge Items zijn opgezet in het Technisch Ontwerp. Hieronder wordt elk van de velden besproken en wordt gekeken of het een kandidaat is voor een Sharding Key. Sommige velden zijn ‘Deels kandidaat’. In dit geval voldoet het veld niet aan de eerste eis omdat het veld aanpasbaar moet zijn. Echter is het mogelijk de originele waarde van deze velden de bewaren in een nieuw veld en deze originele waarde te gebruiken als Sharding Key.

Veld	Beargumentatie	Kandidaat?
_id	Het _id is de unieke code van een document in de collectie. Het _id is niet aan te passen[11] en voldoet hiermee aan de eerste eis. Ook is het _id geen array.	**Ja**
title	De title is de titel van een Knowledge Item. De titel kan later worden aangepast door de gebruiker. Hierdoor voldoet de title niet aan de eerste eis. Echter, kan de oorspronkelijke titel mogelijk wel bewaard blijven in een ander veld om de (oorspronkelijke) titel toch als Sharding Key te gebruiken. Daarom is dit een potentiële kandidaat	**Deels**
description	De description is de tekst van een Knowledge Item. Dit veld is aan te passen door de gebruiker en voldoet dus niet aan de eerste eis. Echter, kan de oorspronkelijke description mogelijk wel bewaard blijven in een ander veld om de (oorspronkelijke) description toch als Sharding Key te gebruiken. Daarom is dit een potentiële kandidaat	**Deels**
rating	De rating is de beoordeling van een item. De beoordeling kan altijd veranderen wanneer iemand een item beoordeeld en voldoet dus niet aan de eerste eis. De oorspronkelijke rating bewaren is niet nuttig, omdat deze altijd 0 is bij het aanmaken van een nieuw artikel.	**Nee**
tags	Er kunnen meerdere tags worden toegevoegd aan een item. Hierdoor betreft het een array en voldoet dit veld niet aan de eisen.	**Nee**
creator	De creator is de persoon die het item heeft toegevoegd. Het item zal altijd voor het eerst door die persoon zijn toegevoegd en dit veld zal dus niet veranderen. Er kan maar een creator zijn per item. Dit veld voldoet aan alle eisen.	**Ja**
author	De auteur is de persoon die het item heeft gemaakt. De auteur kan later worden aangepast en voldoet niet aan de eerste eis. De oorspronkelijke auteur kan net zoals bij title en description bewaard worden, echter zal dit vaak hetzelfde zijn als een creator waardoor dit niet nuttig is.	**Nee**
coauthor	De co-author is een persoon of zijn meerdere personen die mee hebben geholpen aan het item en kan aangepast worden. Hiermee voldoet het niet aan de eerste eis en tweede eis.	**Nee**
followers	Followers zijn personen die het item volgen. Er kunnen meerdere followers zijn waardoor het veld niet voldoet aan de tweede eis. Ook kunnen followers later veranderen als een nieuwe persoon het item volgt.	**Nee**
communities	Communities kunnen later worden toegekend en er kunnen meerdere communities per item zijn. Hierdoor valt dit veld af.	**Nee**
comments	Comments kunnen later worden toegevoegd en er kunnen meerdere comments per item zijn. Hierdoor valt dit veld af.	**Nee**
views	Het aantal weergaven van een item wijzigt wanneer iemand het item leest. Hierdoor voldoet dit veld niet aan alle eisen. Het is niet nuttig het oorspronkelijke aantal views bij te houden omdat bij het aanmaken van een nieuw item altijd 0 is.	**Nee**
creationDate	De creationDate is de datum waarop het item is geplaatst. Dit verandert niet. Dit veld voldoet aan alle eisen.	**Ja**
modificationDate	De modificationDate verandert wanneer het item wordt aangepast. Hierdoor voldoet dit veld niet aan de eerste eis. Het is niet nuttig om de oorspronkelijke modificationDate te gebruiken voor sharding key, omdat dit hetzelfde is als de creationDate.	**Nee**
publicationDate	De publicationDate is de datum waarop het item is gepubliceerd. Theoretisch kan dit veld worden aangepast als er functionaliteit is waarbij het item gedepubliceerd en geherpubliceerd kan worden. Hiermee voldoet het niet aan alle feiten. Het is niet nuttig om de oorspronkelijke publicationDate te gebruiken voor sharding key, omdat dit hetzelfde is als de creationDate.	**Nee**

visibility	De visibility is de zichtbaarheid van het item. Het kan Public en Private zijn. Dit veld is door de gebruiker aan te passen en voldoet hiermee niet aan de eerste eis. Echter, kan de oorspronkelijke visibility mogelijk wel bewaard blijven in een ander veld om de (oorspronkelijke) visibility toch als Sharding Key te gebruiken. Daarom is dit een potentiële kandidaat	**Deels**
deleted	Het deleted veld duidt aan of het item is verwijderd of niet. Omdat een item later kan veranderen is dit veld aanpasbaar. Hierdoor voldoet het niet aan de eerste eis. Het is niet nuttig de oorspronkelijke deleted waarde als Sharding Key te gebruiken. Deze is bij het toevoegen van een nieuw item namelijk altijd 'niet-verwijderd'.	**Nee**
category	De category bevat de categorie van het item. Er zijn op het moment van schrijven 8 verschillende categorieën. Dit veld kan later worden aangepast en voldoet hiermee niet aan de eerste eis. Echter, kan de oorspronkelijke categorie mogelijk wel bewaard blijven in een ander veld om de categorie toch als Sharding Key te gebruiken. Daarom is dit een potentiële kandidaat.	**Deels**

3.1.3 Sharding Key

In paragraaf 3.1.2 is gekeken welke velden van een Knowledge Item mogelijk gebruikt kunnen worden als Sharding Key. In deze paragraaf worden deze Sharding Keys naast elkaar gelegd en met elkaar vergeleken. Dit wordt gedaan door middel van enkele aanbevelingen die door MongoDB worden gedaan:

Kies een Sharding Key die veel op te splitsen is MongoDB plaatst alle documenten met dezelfde Sharding Key waarde in dezelfde chunk. Een chunk is niet deelbaar over meerdere Shards. Wanneer er maar weinig mogelijke Sharding Key waarden zijn, bijvoorbeeld in het geval van geslacht (man of vrouw), kan de data ook maar opgesplitst worden in 2 chunks en dus maximaal 2 shards. Er bestaat dan de kans dat de shard vol of overbelast raakt.
Kies een Sharding Key dat een hoge willekeurigheid (Write Scaling) Het wordt aangeraden een Sharding Key te kiezen met een grote willekeurigheid. Hierdoor worden lees- en schrijf opdrachten verdeeld over de verschillende shards en wordt voorkomen dat maar een deel van de shards alle opdrachten verwerkt en een deel niks doet. Een Sharding Key met hoge willekeurigheid is bijvoorbeeld de unieke code van een document, voor elk document is deze anders.
Kies een Sharding Key die één Shard raakt (Query Isolation) Hierbij wordt tijdens de keuze van een Sharding Key rekening gehouden met veelgebruikte queries op de database. Wanneer de Sharding Key in een leesquery zit wordt er maar één Shard benaderd. Als de Sharding Key niet in de query zit dienen alle Shards benaderd te worden.
Maak gebruik van een samengestelde Sharding Key als er geen optimale key is Er kan niet altijd een optimale Sharding Key worden gevonden in een document[9], daarom kan er soms een combinatie worden gemaakt tussen verschillende velden die speciaal dient als Sharding Key.

Van elke kandidaat is gekeken naar 2 mogelijkheden: de normale waarde en een gehashte waarde[6]. MongoDB biedt ondersteuning voor beide typen[8].

Om te kijken of een veld te maken krijgt met een goede of minder goede Query Isolation wordt gekeken naar de Use Cases van de Knowledge Service. Deze Use Cases zijn als volgt:

Use Case	Actor
Toevoegen van Knowledge Item	Gebruiker
Wijzigen van Knowledge Item	Gebruiker
Verwijderen van Knowledge Item	Gebruiker
Tonen van één Knowledge Item	Gebruiker
Tonen van alle Knowledge Items	Gebruiker
Verwerken update persoon in Knowledge Items	People Service

De sharding key wordt bepaald aan de hand van de aanbevelingen en de hierboven beschreven Use Cases.

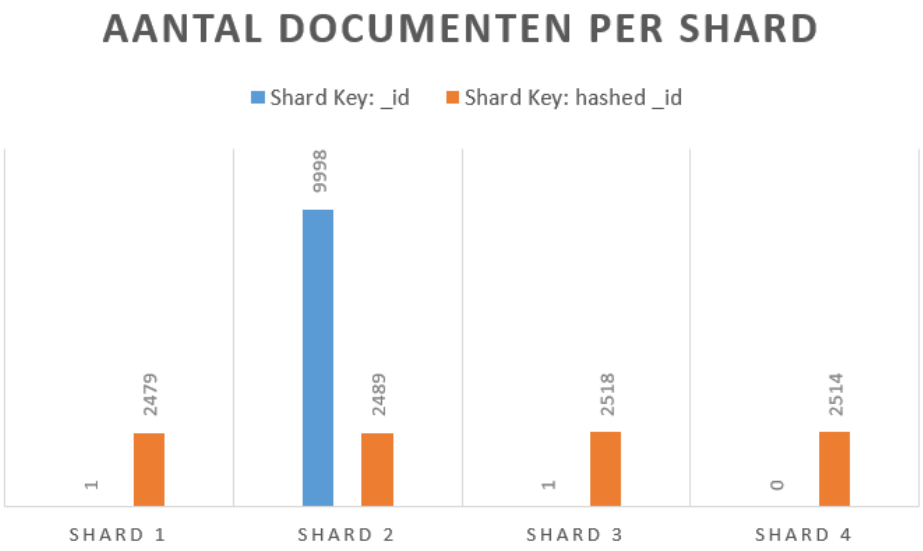
[6] Een gehashte waarde is een wiskundige berekening die een waarde omzet. De omgezette waarde kan niet worden omgezet naar de oorspronkelijke waarde[15].

3.1.3.1 _id

Het _id veld zelf is geen geschikte sharding key. Het _id veld is namelijk oplopend. Elk nieuw item krijgt een _id hoger dan het item daarvoor. Hierdoor worden alle nieuwe items op dezelfde shard toegevoegd. Waardoor één shard alle schrijftransacties moet uitvoeren.

Een gehasht _id heeft dit probleem niet. MongoDB raadt het gebruik van een gehasht _id zelf aan indien voor het _id wordt gekozen[8]. Documenten met een gehasht _id komen niet allemaal terecht op dezelfde shard waardoor er wel Write Scaling is.

Het verschil tussen een niet-gehasht en gehasht veld is goed duidelijk in het onderstaande figuur.



In dit figuur is te zien dat bij het gebruik van het veld _id voor een sharding key bijna alle data op één shard terecht komt. Bij het gebruik van een gehasht _id is de data gelijk verdeeld over de 4 shards. De resultaten in dit figuur komen uit een opgezette test waarbij 10.000 documenten een gesharde omgeving worden toegevoegd. Een test met _id als Sharding Key, een test met het gehashte _id als Sharding Key.

Het _id veld is voor elk document anders, het is immers een unieke code. Daarom zijn er veel splitsingsmogelijkheden.

Het gehashte _id veld zal net als het normale _id veld zorgen voor een hoge Query Isolation voor de Use Case 'Tonen van één Knowledge Item'. Het Knowledge Item wordt dan namelijk opgehaald op basis van het _id. MongoDB zal ook bij het gebruik gehasht _id een hoge Query Isolation bieden, MongoDB vertaald namelijk bij het ophalen automatisch het niet gehashte _id naar een gehasht _id om zo te kunnen achterhalen in welke Shard het item zit. [8].

Veld	Veel splitsings-mogelijkheden	Write Scaling	Query Isolation	Score
_id	+	-	+	+
_id (gehasht)	+	+	+	+ + +

3.1.3.2 title (oorspronkelijke)

Zowel de oorspronkelijke titel als een gehashte oorspronkelijke titel hebben veel splitsingsmogelijkheden. De meeste items hebben verschillende titels. Ook is de Write Scaling van beide velden hoog, niet alle titels beginnen met dezelfde letter.

Query Isolation is niet van toepassing voor dit veld. Er zijn geen Use Cases in knowNow waarbij items opgehaald moeten worden aan de hand van de titel.

Veld	Veel splitsings-mogelijkheden	Write Scaling	Query Isolation	Score
title (oorspronkelijke)	+	+	-	+
title (oorspronkelijke, gehasht)	+	+	-	+

3.1.3.3 description (oorspronkelijke)

Zowel de oorspronkelijke description als een gehashte oorspronkelijke description hebben veel splitsingsmogelijkheden. Items zullen grotendeels andere descriptions hebben. Ook is de Write Scaling van beide velden hoog, niet alle titels beginnen met dezelfde letter.

Query Isolation is niet van toepassing voor dit veld. Er zijn geen Use Cases in knowNow waarbij items opgehaald moeten worden aan de hand van de description.

Veld	Veel splitsings-mogelijkheden	Write Scaling	Query Isolation	Score
description (oorspronkelijke)	+	+	-	+
description (oorspronkelijke, gehasht)	+	+	-	+

3.1.3.4 creator

Het creator veld heeft veel splitsingsmogelijkheden, echter zullen er mensen zijn die veel Knowledge Items publiceren en mensen zijn die weinig Knowledge Items publiceren. Daarom is er geen Write Scaling. Ook zijn er geen Use Cases voor die creators opvragen. Hierdoor is er ook geen sprake van Query Isolation.

Veld	Veel splitsings-mogelijkheden	Write Scaling	Query Isolation	Score
creator	+	-	-	-
creator (gehasht)	+	-	-	-

3.1.3.5 creationDate

Het creationDate veld is net zoals het _id een oplopend veld, elk nieuwe item bevat eenzelfde of hogere creationDate als het item daarvoor. Hierdoor zal alle nieuwe data net zoals bij _id op één shard terechtkomen.

Het is ook mogelijk te kiezen voor een gehashte creationDate. Deze heeft net als bij _id het voordeel dat de data eerlijk wordt verdeeld over de verschillende shards.

Er zijn geen Use Cases voor het opvragen van Knowledge Items aan de hand van een creationDate. Daarom is er geen voordeel op het gebied van Query Isolation.

Veld	Veel splitsings-mogelijkheden	Write Scaling	Query Isolation	Score
creationDate	+	-	-	-
creationDate (gehasht)	+	+	-	+

3.1.3.6 visibility

Het visibility veld heeft maar 2 mogelijke waarden: Public en Private. Hierdoor zijn er maar 2 splitsingsmogelijkheden waardoor er maximaal 2 shards te gebruiken zijn. Daarnaast zijn er geen Use Cases die items opvragen aan de hand van de visibility. Vanwege het geringe aantal mogelijke waarden is er ook geen write scaling, alles zal op 2 shards moeten worden geschreven. Indien er in de toekomst meer shards nodig zijn voor knowNow is dat dan met deze Sharding Key niet mogelijk.

Veld	Veel splitsings-mogelijkheden	Write Scaling	Query Isolation	Score
visibility	-	-	-	- - -
visibility (gehasht)	-	-	-	- - -

3.1.3.7 category

Het veld category heeft op het moment van schrijven maar 8 verschillende waarden, waardoor het aantal splitsingsmogelijkheden laag is. Daarnaast zit het grootste gedeelte van de items in één categorie, Artikelen. Hierdoor is er geen Write Scaling.

Veld	Veel splitsings-mogelijkheden	Write Scaling	Query Isolation	Score
category	+/-	-	-	--
category (gehasht)	+/-	-	-	--

3.1.3.8 Combinaties

Er is niet gekeken naar een combinatie van meerdere velden. Hiertoe is besloten omdat de opgestelde Use Cases de bovenstaande velden niet nodig hebben voor het ophalen van items vanuit MongoDB, naast het _id. Hierdoor zal een combinatie van meerdere velden ook niet zinvol zijn.

3.1.4 Conclusie

Er wordt aangeraden om een gehasht _id veld te gebruiken als Sharding Key.

In de tabel hieronder zijn alle bevindingen van de Sharding Key kandidaten samengevat.

Veld	Veel splitsings-mogelijkheden	Write Scaling	Query Isolation	Score
_id (gehasht)	+	+	+	+++
creationDate (gehasht)	+	+	-	+
title (oorspronkelijke)	+	+	-	+
title (oorspronkelijke, gehasht)	+	+	-	+
description (oorspronkelijke)	+	+	-	+
description (oorspronkelijke, gehasht)	+	+	-	+
_id	+	-	+	+
creator	+	-	-	-
creator (gehasht)	+	-	-	-
creationDate	+	-	-	-
category	+/-	-	-	--
category (gehasht)	+/-	-	-	--
visibility	-	-	-	- - -
visibility (gehasht)	-	-	-	- - -

Voor de Use Case ‘Verwerken update persoon in Knowledge Items’ is het hebben van een hoge Write Scaling van belang. Bij een hoge Write Scaling kan een update parallel uitgevoerd worden op alle shards, bij een lage Write Scaling zal de update maar op een klein deel van de shards uitgevoerd worden en duurt de update mogelijk langer.

Bij de Use Case ‘Tonen van één Knowledge Item’ is een hoge Query Isolation voor het _id veld belangrijk, bij deze Use Case wordt een item opgehaald op basis van het _id.

Voor de velden die oorspronkelijke waarden van een ander veld bevat zoals title (oorspronkelijke) en description (oorspronkelijke) is duidelijk geworden dat dit geen winst geeft op splitsingsmogelijkheden, write scaling en query isolation vergeleken met bijvoorbeeld het niet-gehashte _id. Hierdoor heeft het geen voordeel om voor deze velden een nieuw veld te maken om ze te kunnen gebruiken als Sharing Key. Deze opties vallen daarom ook af.

Het _id veld is uniek en heeft hierdoor de meeste splitsingsmogelijkheden. Ook heeft een gehasht _id een betere Write Scaling dan het normale _id veld zoals te zien in 5.3.1. Een hoge Write Scaling is van belang voor het toevoegen van items en het updaten vanuit andere services. Bij een te lage Write Scaling zullen alleen de laatste shards daar taken voor moeten uitvoeren, terwijl je dat zoveel mogelijk wilt verdelen.

3.2 Aanbeveling schaling

Doordat er geen onderzoek is gedaan wat er specifiek voor knowNow nodig is aan servers kan er ook geen advies worden gegeven in hoeverre MongoDB voor knowNow moet worden geschaald. Echter, er is wel een advies voor de Sharding Key gegeven.

Ik raad het knowNow team aan om het advies van MongoDB op te volgen en vooral niet te vroeg op te schalen. Begin met één server en kijk goed naar wat deze server aankan. Wanneer bevonden wordt dat de server het niet meer trekt kan er begonnen worden met het opschalen. Hierbij dient begonnen te worden met 2 shards waarna er onbeperkt nieuwe shards aan toegevoegd kunnen worden[14].

De door mij geadviseerde Sharding Key kan hiervoor goed worden gebruikt. Deze Sharding Key zorgt voor een gelijke verdeling van lees- en schrijfperformance over de meerdere shards.

4. Documentschema

In Bijlage B, Onderzoeksrapport, Deelvraag 4 is onderzoek gedaan naar het beste documentschema voor knowNow wanneer er gebruik wordt gemaakt van MongoDB. Dit is gedaan aan een aantal punten die door de opdrachtgever belangrijk zijn bevonden:

Snelheid (performance)	De snelheid is van belang voor de gebruiker zodat deze niet te lang hoeft te wachten voordat de actie is uitgevoerd. De snelheid wordt onderzocht door middel van een performancetest op de Use Cases in de tabel van de vorige pagina. Voor de snelheid is alleen de initiële query van belang omdat deze maatgevend is voor het lezen, de gebruiker kan op dat moment starten met het lezen van het item terwijl additionele informatie, zoals bijhorende auteurs, nog wordt bijgeladen.
Schaalbaarheid	Een oplossing met een hoge schaalbaarheid is nadrukkelijk gewenst. Een oplossing met hogere schaalbaarheid kan beter meegroeien wanneer het aantal gebruikers van de applicatie toeneemt. knowNow moet om kunnen gaan met in ieder geval 200.000 en mogelijk zelfs 3 miljoen gebruikers waardoor een hoge schaalbaarheid belangrijk is.
Consistentie	Het hebben van direct consistente data heeft de voorkeur tegenover uiteindelijke consistentie.
Onderhoudbaarheid / eenvoud van de oplossing	Een beter onderhoudbare, eenvoudige oplossing is beter omdat dit het voor andere ontwikkelaars makkelijker maakt aanpassingen te doen aan de applicatie en eventuele problemen zo sneller opgelost kunnen worden.

Het resultaat van de toetsing, waarbij al deze punten zijn getest op Use Cases van beide varianten van het documentschema is te vinden in de volgende tabel. Duidelijk is dat het relationeel schema op veel punten beter scoort dan het hiërarchisch documentschema.

Use Case/Toetsingspunt	Score hiërarchisch documentschema	Score relationeel documentschema
Ophalen Knowledge Item		
Snelheid	+/-	+/-
Schaalbaarheid	+/-	-
Consistentie	-	+
Onderhoudbaarheid	+/-	+/-
Toevoegen Knowledge Item		
Snelheid	-	+
Schaalbaarheid	+/-	+/-
Consistentie	+/-	+/-
Onderhoudbaarheid	-	+
Wijzigen Knowledge Item		
Snelheid	-	+
Schaalbaarheid	+/-	+/-
Consistentie	+/-	+/-
Onderhoudbaarheid	-	+
Wijzigen People Item		
Snelheid	+/-	+/-
Schaalbaarheid	-	+
Consistentie	-	+
Onderhoudbaarheid	-	+
Verwijderen People Item		
Snelheid	+/-	+/-
Schaalbaarheid	-	+
Consistentie	+/-	-
Onderhoudbaarheid	-	-
Totaal	**-10**	**+6**
Schaalbaarheid	-2	+1
Snelheid	-2	+2
Consistentie	-2	+1
Onderhoudbaarheid	-4	+2

Daarom raad ik knowNow ook aan om gebruik de maken van een relationeel documentschema waarbij de gebruiker een item alvast kan lezen terwijl additionele informatie, zoals de auteur, asynchroon bijgeladen wordt. De opdrachtgever gaf aan dat het bijladen van de additionele informatie geen probleem is, maar mocht er toch twijfel ontstaan dan raad ik aan om een gebruikersacceptatietest te doen om er zo zeker van te zijn dat het bijladen door de gebruiker niet als hinderlijk wordt ervaren.

Wel is het belangrijk om te kijken of er in de toekomst mogelijke nieuwe Use Cases ontstaan welke de uitkomst van het onderzoek kunnen beïnvloeden. Ook is het aan te raden te kijken hoe vaak een wijziging en verwijdering van een People Item nou daadwerkelijk plaatsvind. Wanneer dat aantal nihil is kan altijd worden besloten alsnog te kijken naar het hiërarchisch documentschema.

4.1 Aanbevelingen documentschema

Alhoewel MongoDB geen vast documentschema hanteert en het daarmee mogelijk is verschillende schema's te hanteren binnen een collectie raad ik wel aan om ervoor te zorgen dat het documentschema binnen een collectie hetzelfde is. Dit zorgt ervoor dat een applicatie die communiceert met MongoDB weet welk schema hij moet verwachten en niet eerst moet kijken volgens welk schema het document is opgebouwd. Hierdoor blijft de applicatie overzichtelijker en ook beter onderhoudbaar.

4.2 Migraties

In het onderzoek is bekeken of stapelbare migraties mogelijk zijn met MongoDB. Uit de ontwikkelde demo is gebleken dat het mogelijk is om migraties te stappelen voor collecties in MongoDB. Zowel het aanpassen van het schema, het aanpassen van data en het controleren van schema's is in de demo mogelijk. Daarom is te stellen dat gestapelde migraties, zoals nu al in het huidige knowNow gebeurt ook mogelijk zijn voor MongoDB.

Later, na het ontwikkelen van de demo, bleek dat het bijhouden van versienummer per collectie misschien niet de handigste oplossing is omdat MongoDB geen schema per collectie kent, maar per document. Daarom raad ik aan om in plaats van het bijhouden van de versie per collectie de versie per document bij te houden. Hierdoor kan, mocht MongoDB tijdens een migratie uitvallen, bepaald worden welke documenten nog geüpdatet moeten worden.

5. Job Queue

Om bij het hiërarchisch documentschema als subdocument opgenomen relaties consistent te maken is door mij de Job Queue ontwikkeld. De Job Queue zorgt ervoor deze subdocumenten te allen tijde uiteindelijk consistent raken, ook al valt de Knowledge Service of MongoDB uit. Alhoewel bij het geadviseerde relationeel documentschema de Job Queue niet nodig is voor wijzigingen, en de Job Queue daardoor niet ingezet wordt, zijn er toch nog enkele adviezen die ik wil geven mocht er in de toekomst toch van de Job Queue gebruik gemaakt worden.

5.1 Threading

Op dit moment is de Job Queue een single threaded[7] methode. Hiervoor is gekozen omdat dit ervoor zorgt dat de Job Queue geheel idempotent[8] is. Echter hoeft de Job Queue niet geheel idempotent te zijn, de Job Queue hoeft alleen idempotent te zijn voor de wijziging van hetzelfde item.

Hierdoor is het toch mogelijk de Job Queue in meerdere threads op te delen, zolang er maar voor wordt gezorgd dat de aanpassingen aan een item in dezelfde thread terecht komen. Het proces is dan nog steeds idempotent. Wanneer de stappen opnieuw worden uitgevoerd is het resultaat hetzelfde. Het voordeel van het in meerdere threads opdelen is dat de taken parallel uitgevoerd worden. Of door het gebruik van meerdere threads ook performancewinst behaald wordt moet worden getest.

5.2 Uitval

Op dit moment voert de Job Queue een taak net zolang uit totdat deze slaagt. Dit kan er tot leiden dat de gehele Job Queue vastloopt wanneer een taak onuitvoerbaar is. In dat geval is het wenselijk om een soort ‘uitvalbak’ te creëren waardoor de Job Queue niet geheel vastloopt en ook andere taken uitgevoerd kunnen blijven worden.

In deze uitvalbak komen dan taken terecht die meermaals opnieuw zijn geprobeerd maar nooit geslaagd zijn. De beheerders van de applicatie krijgen melding wanneer een taak in deze uitvalbak terecht komt en kunnen deze dan handmatig doorvoeren. De beheerders moeten er dan wel op letten of de taak nog actueel is. Dit houdt in dat er bekeken moet worden of er na het uitvallen van de taak geen nieuwe taak is geweest die de uitgevallen taak zal overschrijven.

[7] Single threaded houdt in dat de Job Queue draait in één Thread(proces) en hierdoor maar een taak tegelijkertijd kan uitvoeren.
[8] Idempotent is een eigenschap van een object wat zegt dat het object niet meer veranderd als een operatie nogmaals wordt uitgevoerd[16].

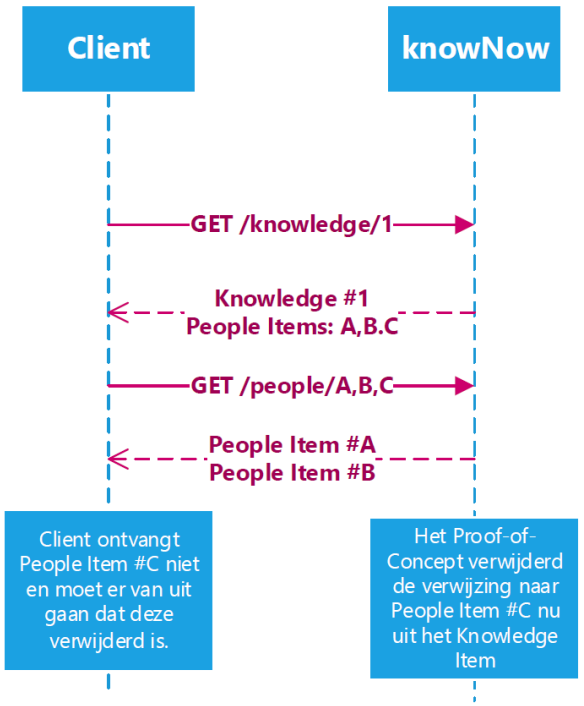
6. Consistentie

Bij het hiërarchisch documentschema zorgt de Job Queue voor het consistent maken van de documenten. Wordt een People Item verwijderd, dan haalt de Job Queue deze uit de Knowledge Items, wordt een People Item gewijzigd, dan voert de Job Queue deze wijziging door in de Knowledge Item.

Bij het relationeel documentschema is het consistent maken van de Knowledge Items niet nodig bij het wijzigen van een People Item.

Het verwijderen van een People Item in het relationeel documentschema is anders. Doordat MongoDB geen referentiele integriteit kent zoals SQL Server, worden verwijzingen niet automatisch verwijderd door MongoDB.

Een oplossing die door mij is geïmplementeerd in het Proof-of-Concept verwijderd een relatie niet op het moment dat een People Item wordt verwijderd. De applicatie verwijderd het pas op het moment dat blijkt dat tijdens het ophalen van een Knowledge Item een verwijzing naar een People Item niet meer bestaat. Hierdoor blijft een Knowledge Item met een verwijzing naar een verwijderd People Item inconsistent totdat deze wordt opgevraagd. Dit betekend ook dat een client[9] mogelijk niet bestaande verwijzingen naar People Items ontvangt en dat de client hier mee om moet kunnen gaan. Dit is duidelijk te zien in figuur 4-1.



Figuur 6-1. Het ophalen van een Knowledge Item met verwijzing naar verwijderd People item

De volgende keer dat de client hetzelfde item opvraagt heeft de applicatie de verwijzing al verwijderd. Hierdoor raken alle documenten wel uiteindelijk consistent.

Het voordeel van deze oplossing is dat wanneer er een relatie verwijderd wordt, het niet meteen doorgevoerd hoeft te worden in alle documenten, maar één relatie pas wordt verwijderd op het moment dat één Knowledge Item wordt opgevraagd. Hierdoor wordt de verwijdertransactie opsplitst in veel kleinere transacties die pas uitgevoerd worden wanneer nodig. Dit maakt de oplossing dus veel schaalbaarder.

Het grote nadeel is dat clients soms inconsistente items ontvangen, een relatie wordt pas uit een item verwijderd wanneer blijkt dat deze niet meer bestaat. Ook kan het soms langere tijd duren voordat de database volledig consistent is, sommige items worden maar heel weinig en misschien wel nooit meer opgevraagd.

In het Proof-of-Concept is deze oplossing succesvol geïmplementeerd. Wanneer de consistentie door knowNow belangrijker wordt geacht dan de schaalbaarheid raad ik aan om ook in het relationeel documentschema gebruik te maken van de Job Queue die alle relaties direct verwijderd. Wanneer de schaalbaarheid belangrijker wordt gevonden kan er gebruik worden gemaakt van de hier uitgelegde oplossing.

[9] Een client is een applicatie, zoals de webinterface, die gebruik maakt van het Proof-of-Concept via de API.

7. Performance

Tijdens het uitvoeren van het afstudeerproject zijn diverse testen gedaan naar de performance. Echter, zijn al deze performancetesten uitgevoerd en gebruikt om 2 mogelijke implementaties te vergelijken. Doordat deze testen niet zijn uitgevoerd op een met productie vergelijkbare omgeving kunnen deze resultaten niet gebruikt worden om te kijken hoe knowNow met MongoDB performt in een productieomgeving in vergelijking tot SQL Server.

Het is, wegens tijdgebrek, niet mogelijk geweest om dit te onderzoeken. Echter, raad ik knowNow wel aan om dit eerst te doen voordat besloten wordt om MongoDB ook daadwerkelijk voor knowNow te gaan gebruiken. Aan de hand van een performancetest uitgevoerd op een met productie vergelijkbare omgeving kan worden bepaald of MongoDB op het gebied van snelheid beter is dan het huidige SQL server.

8. Conclusie & Aanbeveling

Door de uitkomst van het onderzoek naar het documentschema raad ik het knowNow team aan gebruik te maken van een relationeel documentschema. Het relationele documentschema scoort op de geteste Use Cases en toetsingspunten het beste. Ik raad wel aan om eerst door middel van een gebruikersacceptatie test te kijken of het asynchroon ophalen van additionele informatie niet als hinderlijk wordt ervaringen.

Alhoewel MongoDB geen vast documentschema hanteert en het daarmee dus mogelijk is verschillende schema's te hanteren binnen een collectie, raad ik aan om ervoor te zorgen dat het documentschema binnen een collectie hetzelfde is. Dit zorgt ervoor dat een applicatie die communiceert met MongoDB weet welk schema hij moet verwachten en niet eerst moet kijken volgens welk schema het document is opgebouwd. Hierdoor blijft de applicatie overzichtelijker en ook beter onderhoudbaar.

In het relationele documentschema wordt de referentiele integriteit, wat MongoDB niet kent, bij het verwijderen van een relatie opgelost door een relatie te verwijderen op het moment dat wordt gemerkt dat deze niet meer bestaat. In het Proof-of-Concept is bewezen dat deze methode werkt. Echter betekent dit wel dat de documenten in een inconsistente staat blijven tot ze allemaal één keer zijn opgevraagd. De gebruiker merkt dit zelf niet en het is hierdoor voor de gebruiker van de applicatie een goede oplossing. Echter kan ik mij voorstellen dat in sommige gevallen, bijvoorbeeld wanneer er ook rapporten gemaakt moeten worden, inconsistente data wel voor problemen kunnen zorgen. Wanneer dat het geval is, raad ik aan te kijken of de Job Queue in het relationele documentschema inzetbaar is voor het direct verwijderen van alle relaties net zoals bij het hiërarchisch documentschema. Hierdoor wordt de data wel zo snel mogelijk consistent gemaakt, maar vervalt het voordeel op schaalbaarheid bij het relationeel documentschema. Bij het verwijderen van een People Item worden dan ook alle relaties in één keer verwijderd in plaats van alleen op het moment dat wordt gemerkt dat de relatie niet meer bestaat.

Op basis van het Proof-of-Concept en het onderzoek kan geconcludeerd worden dat MongoDB een werkbare database is voor de Knowledge Items van knowNow. Er moet echter wel heel veel gedaan worden om dit te bereiken. Zo moeten er diverse oplossingen worden ontwikkeld die bij een RDBMS helemaal niet nodig zijn. Daarom ontstaat de vraag of MongoDB wel daadwerkelijk een beter alternatief is. Er zijn door mij diverse zaken uitgezocht en ontwikkeld die functionaliteit nabootsen die standaard al aanwezig zijn in een RDBMS, zoals de Job Queue die ervoor zorgt dat er toch gegarandeerd kan worden dat een wijziging volledig wordt doorgevoerd. In een RDBMS is dit standaard mogelijk door middel van atomaire transacties waarbij de transactie of volledig slaagt, of helemaal niet.

Ook het missen van referentiële integriteit in MongoDB is een nadeel. De applicatie die van de database gebruik maakt zal zelf oplossingen in moeten bouwen om de data uiteindelijk consistent te maken. SQL Server heeft wel referentiële integriteit waardoor verwijzingen door de database zelf verwijderd worden.

Het knowNow team moet zich afvragen of de voordelen van MongoDB opwegen tegen de diverse concessies die gedaan moeten worden op het gebied van atomaire transacties en referentiële integriteit. De voordelen die MongoDB kan bieden op het gebied van schaalbaarheid en performance zijn binnen deze opdracht niet uitputtend onderzocht omdat er vanuit de Sprints nieuwe onderzoeksvragen zijn ontstaan waardoor hier geen tijd meer voor is gevonden. De schaalbaarheid van MongoDB lijkt erg hoog, onder andere door het toepassen van sharding maar wat exact nodig is voor knowNow om aan de verwachte groei te voldoen is onbekend.

Mocht de opdrachtgever MongoDB toch willen inzetten, ondanks deze zwaarwegende concessies, raad ik aan eerst een performance onderzoek te doen waarbij wordt gekeken naar wat benodigd is om de gebruikers een high performant applicatie te kunnen aanbieden. Wanneer de groep gebruikers in die test steeds wordt vergroot en de MongoDB database indien nodig bij wordt geschaald, kan worden gekeken wat er benodigd is aan servers.

De resultaten van dit onderzoek kunnen dan vergeleken worden met een vergelijkbaar Proof-of-Concept (ook met Microservice-architectuur) wat gebruik maakt van SQL Server of een alternatief RDBMS als database. Maak geen gebruik van de huidige datastructuur van het op Orchard ontwikkelde knowNow, maar een voor een Microservice-architectuur geoptimaliseerde datastructuur waardoor de resultaten beter te vergelijken zijn.

Wanneer blijkt dat de performance van MongoDB zo veel beter is dan die van een RDBMS waarbij dit voordeel opweegt tegen de concessies van MongoDB, kan besloten worden om MongoDB daadwerkelijk in te zetten. Iedereen die aan het project werkt moet dan wel zeker weten waar ze aan beginnen en hoe de diverse processen, zoals een Job Queue, werken om ongewenste problemen in de toekomst te voorkomen. Voor het team moet duidelijk zijn wat wel met MongoDB kan en vooral wat MongoDB niet kan. Met name de missende atomaire transacties en het niet bieden van referentiële integriteit zijn sterk complicerende factoren voor het slagen van MongoDB als vervanger van de huidige database.

9. Referenties

Code	Bron
[1]	MongoDB, Backup Strategies Compared http://blog.cloud.mongodb.com/post/86915115595/mongodb-backup-strategies-compared
[2]	Backup a Sharded Cluster with Filesystem Snapshots http://docs.mongodb.org/manual/tutorial/backup-sharded-cluster-with-filesystem-snapshots/
[3]	MongoDB, “Backup with Mongodump” http://docs.mongodb.org/manual/core/backups/#backup-with-mongodump
[4]	MongoDB, “Cloud Backups” https://www.mongodb.com/cloud/backup
[5]	MMS FAQ, “Is My Data Safe?” https://docs.mms.mongodb.com/faq/
[6]	MongoDB, “Ops Manager Backup Software” http://docs.mongodb.org/manual/core/backups/#backup-with-mms-onprem
[7]	MongoDB, “MongoDB Enterprise Advanced” https://www.mongodb.com/products/mongodb-enterprise-advanced
[8]	MongoDB, “Sharding Key” http://docs.mongodb.org/manual/core/sharding-shard-key/
[9]	MongoDB, “Choose a Sharding Key” http://docs.mongodb.org/manual/tutorial/choose-a-shard-key/
[10]	MongoDB, “Sharding Sharding Key Indexes” http://docs.mongodb.org/manual/core/sharding-shard-key-indexes/
[11]	MongoDB, “Modify Documents” http://docs.mongodb.org/manual/tutorial/modify-documents/
[12]	MongoDB, “Sharing FAQ” http://docs.mongodb.org/manual/faq/sharding/
[13]	MongoDB, “Sharded Cluster Requirements” http://docs.mongodb.org/manual/core/sharded-cluster-requirements/
[14]	MongoDB, “MongoDB Limits and Thresholds” http://docs.mongodb.org/manual/reference/limits/
[15]	Techtarget, “Hashing” http://searchsqlserver.techtarget.com/definition/hashing
[16]	Wikipedia, “Idempotentie” https://nl.wikipedia.org/wiki/Idempotentie

DE HAAGSE HOGESCHOOL

Info Support Solid Innovator

Bijlage J - Afstudeerplan

MongoDB: Een geschikte database voor het kennismanagementsysteem knowNow?

Afstudeerplan

Informatie afstudeerder en gastbedrijf (*structuur niet wijzigen*)

Afstudeerblok: 2015-1.2 (start uiterlijk 11 mei 2015)
Startdatum uitvoering afstudeeropdracht: 28 april 2015
Inleverdatum afstudeerdossier volgens jaarrooster: 5 oktober 2015

Studentnummer: 11031425
Achternaam: dhr Keim (*) *weghalen niet van toepassing*
Voorletters: T.M.
Roepnaam: Tom
Adres: Beppie Nooijstraat 7
Postcode: 2642BR
Woonplaats: Pijnacker
Telefoonnummer:
Mobiel nummer: 0612580406
Privé emailadres: tomkeim94@gmail.com

Opleiding: Informatica
Locatie: Den Haag (*) *weghalen niet van toepassing*
Variant: voltijd

Naam studieloopbaanbegeleider: Gerda in ‘t Veld
Naam begeleidend examinator: G.A. Mijnarends
Naam tweede examinator: H.G.J. Bechet

Naam bedrijf: Info Support
Afdeling bedrijf:
Bezoekadres bedrijf: Kruisboog 42
Postcode bezoekadres: 3905 TG
Postbusnummer:
Postcode postbusnummer:
Plaats: Veenendaal
Telefoon bedrijf: 0318 552020
Telefax bedrijf: 0318 552355
Internetsite bedrijf: http://www.infosupport.com/

Achternaam opdrachtgever: dhr Snijder (*) *weghalen niet van toepassing*
Voorletters opdrachtgever: J.
Titulatuur opdrachtgever: Ing.
Functie opdrachtgever: Product Owner knowNow
Doorkiesnummer opdrachtgever: 06 53643473
Email opdrachtgever: joop.snijder@infosupport.com

Achternaam bedrijfsmentor: mw Hijl (*) *weghalen niet van toepassing*
Voorletters bedrijfsmentor: P.D.
Titulatuur bedrijfsmentor:
Functie bedrijfsmentor: Afstudeercoördinator
Doorkiesnummer bedrijfsmentor: 06 45378507
Email bedrijfsmentor: pascalle.hijl@infosupport.com
NB: bedrijfsmentor mag dezelfde zijn als de opdrachtgever

Doorkiesnummer afstudeerder:
Functie afstudeerder (deeltijd/duaal):

Titel afstudeeropdracht:

Onderzoek doen naar MongoDB als mogelijke vervanger van het huidige RDBMS voor het kennismanagement systeem knowNow

Opdrachtomschrijving

1. Bedrijf

Info Support is een succesvol en winstgevend IT bedrijf met meer dan 400 medewerkers in dienst waarvan het hoofdkantoor gevestigd is in Veenendaal. Het ontwikkelt, beheert en host software op maat voor een groot aantal klanten. Het bedrijf, opgericht in 1986, kiest voor de achterliggende technologie vooral voor Microsoft (.NET, C#, SQL Server, Azure), Oracle en Java. Info Support heeft een eigen kenniscentrum waar meerdere docenten de medewerkers van het bedrijf trainingen geven over diverse branche- en IT gerelateerde onderwerpen. Hierdoor blijft het bedrijf innoveren en past het, waar mogelijk, de nieuwste technologieën toe.

2. Probleemstelling

Een van de producten van Info Support is knowNow. knowNow is een zelf lerend kennismanagementsysteem dat het kennisaanbod binnen een organisatie afstemt op basis van het zoekgedrag van de gebruiker. De organisaties kunnen daardoor hun productiviteit en kwaliteit van de informatie in het kennismanagementsysteem verhogen.

Op dit moment maakt knowNow gebruik van Lucene.NET en Microsoft SQL Server. Lucene.NET is een in C# geschreven, op .NET gebruikers gerichte, port van de *Lucene search engine library*. Lucene is een tekst gebaseerde API waarmee informatie geïndexeerd wordt zodat de teksten op een efficiënte manier kunnen worden doorzocht. Microsoft SQL Server is een veelgebruikt Relationeel Database Management Systeem (RDBMS) van Microsoft.

Info Support wil graag de performance en schaalbaarheid van knowNow verbeteren. Daarom is door Info Support besloten onderzoek te doen welke Database Management Systeem (DBMS) het beste door knowNow gebruikt kan worden. Uit dit onderzoek is gebleken dat MongoDB, een document-georiënteerde NoSQL DBMS, de beste keuze is.

NoSQL databases vragen een andere manier van data modellering. NoSQL databases kunnen goed om gaan met data-opslag en met verschillen in de data. Maar welke consequenties het gebruik van een NoSQL database zal hebben op het technisch ontwerp en op het ophalen van de data in knowNow is niet bekend.

Ook is de uitbreidbaarheid van knowNow indien er gebruik wordt gemaakt van MongoDB onbekend. Zo is niet bekend of de datastructuur aangepast dient te worden indien er een nieuw interface bijkomt.

3. Doelstelling van de afstudeeropdracht

De afstudeeropdracht heeft de volgende doelstellingen:

- Onderzoek doen naar de invloed van MongoDB op het technisch ontwerp van knowNow (in het technisch ontwerp wordt onder andere de architectuur, performance, security en data access besproken);
- Onderzoek doen naar welke invloed de gebruikte interface (REST endpoints) heeft op de wijze van data opslag in MongoDB. Is het bijvoorbeeld mogelijk om iets op één manier op te slaan wanneer er 3 verschillende endpoints zijn?;
- Onderzoek doen naar de uitbreidbaarheid bij het gebruik van MongoDB wanneer er later nieuwe (REST endpoints) toegevoegd dienen te worden;
- Het maken van een proof-of-concept met daarin:
    - Een technisch ontwerp van (een deel van) knowNow waarbij de relationele database vervangen is door een MongoDB database;
    - Het (her)ontwikkelen van (een deel van) knowNow waarbij de relationele database vervangen is door een MongoDB database, op basis van het technisch ontwerp, waarbij niet wordt ingeleverd op het gebied van functionaliteit van het (her)ontwikkelde deel;
- Het evalueren van de resultaten uit de gedane onderzoeken aan de hand van het gemaakte proof-of-concept.
- Indien tijd over: onderzoek doen naar schaalbaarheid binnen knowNow systeem. Heeft een MongoDB database invloed op aantal concurrent gebruikers, of schaalbaarheid in groei van data?

4. Resultaat

Na het succesvol afronden van de hierboven geformuleerde doestellingen heeft Info Support een beter beeld wat de consequenties zijn voor het technisch ontwerp en de REST interfaces van knowNow indien Microsoft SQL Server vervangen zal worden door MongoDB.

Op basis van het onderzoek en proof-of-concept zou Info Support mogelijk kunnen bepalen om Microsoft SQL Server ook daadwerkelijk te vervangen door MongoDB.

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

Tijdens het afstuderen zal er gebruik worden gemaakt van de softwareontwikkelmethode Scrum. De afstudeerperiode wordt opgedeeld in 8 sprints van elk 2 weken.

De eerste sprint zal vooral in het teken staan van het kennis te maken met de bedrijfsprocessen van Info Support, het bekijken van de architectuur van het bestaande knowNow en het maken van het Plan van Aanpak. Door het bekijken van de architectuur van het huidige knowNow kan beter in kaart worden gebracht wat het proof-of-concept moet bevatten waardoor de meetbaarheid van het eindresultaat van het proof-of-concept bepaald kan worden.

De andere activiteiten zijn hieronder ingedeeld. Per activiteit is de verwachte duur in werkdagen aangegeven en in welke sprints waarschijnlijk aan deze activiteiten wordt gewerkt.

Omschrijving activiteit	**Verwachte duur in werkdagen**	**Sprints waarin wordt verwacht dat er aan de activiteit wordt gewerkt**
Kennis maken met bedrijfsprocessen Info Support en bekijken van architectuur van knowNow	3	1
Plan van Aanpak opstellen	3	1
Onderzoek doen naar de invloed van MongoDB op het technisch ontwerp van knowNow (in het technisch ontwerp wordt onder andere de architectuur, performance, security en data access besproken)	8	1, 2
Onderzoek doen naar welke invloed de gebruikte interface (REST endpoints) heeft op de wijze van data opslag in MongoDB.	6	2
Onderzoek doen naar de uitbreidbaarheid bij het gebruik van MongoDB wanneer er later nieuwe (REST endpoints) toegevoegd dienen te worden.	6	3
Een technisch ontwerp van (een deel van) knowNow waarbij de relationele database vervangen is door een MongoDB database voor het proof-of-concept.	10	3, 4
Het ontwikkelen van (een deel van) knowNow op basis van het technisch ontwerp voor het proof-of-concept.	20	3 - 7
Het testen van het gemaakte proof-of-concept.	8	3 - 7
Het evalueren van de resultaten uit de gedane onderzoeken aan de hand van het gemaakte proof-of-concept.	6	7, 8
Opbouwen van afstudeerdossier	15	-

6. Op te leveren (tussen)producten

De aan Info Support op te leveren producten zijn als volgt:

- Plan van Aanpak;
- Onderzoekrapportages van de onderzoeken en conclusies van:
    - de invloed van MongoDB op het technisch ontwerp van knowNow;
    - welke invloed de gebruikte interface (REST endpoints) heeft op de wijze van data opslag in MongoDB;
    - de uitbreidbaarheid bij het gebruik van MongoDB wanneer er later nieuwe (REST endpoints) toegevoegd dienen te worden;
- Een proof-of-concept met daarin een technisch ontwerp van (een deel van) knowNow en een (her)ontwikkeld (deel van) knowNow waarbij de traditionele database vervangen is door een MongoDB database;
- De uitkomsten van het evalueren van de resultaten uit de gedane onderzoeken aan de hand van het gemaakte proof-of-concept.

7. Te demonstreren competenties en wijze waarop

Competentie	Complexiteit	Wijze waarop
2.2 Ontwerpen, bouwen en bevragen van een database	Niveau 4	Deze competentie zal worden gedaan op niveau 4. Er wordt een MongoDB database ontworpen en gebouwd. Niveau 4 wordt behaald omdat de beschikbaarheid van data gewaarborgd dient te blijven en performance een kritische factor is.
3.3 Bouwen applicatie	Niveau 4	De complexiteit van deze competentie wordt behaald doordat voor het proof-of-concept een bestaand systeem aangepast dient te worden; er moet in plaats van een DBMS gebruik gemaakt worden van de no-sql oplossing MongoDB.
3.5 Uitvoeren van en rapporten over het testproces	Niveau 3	De complexiteit wordt behaald door het maken van testscripts voor het proof-of-concept. Er wordt gebruik gemaakt van een testframework. De testrapportage omvat conclusies over het in productie nemen van het systeem.
3.1. Ontwerpen systeemarchitectuur	Niveau 3	Deze complexiteit wordt behaald door het ontwerpen van de architectuur van een applicatie, het proof-of-concept van knowNow;

DE HAAGSE HOGESCHOOL

Info Support Solid Innovator

Bijlage K - Evaluatieformulier afstuderen

MongoDB: Een geschikte database voor het kennismanagementsysteem knowNow?

DE HAAGSE HOGESCHOOL

Faculteit IT & Design

Delft
Den Haag
Zoetermeer

Evaluatieformulier afstuderen

In te vullen door opdrachtgever c.q. bedrijfsmentor(en)

Student: Tom Keim

Periode:28 april t/m 5 oktober 2015

Bedrijf c.q. instelling: Info Support

Bedrijfsmentor: Joop Snijder (opdrachtgever)

Plaats: Veenendaal

Datum: 23-09-2015

1. **Heeft de student zich zelf snel en goed ingewerkt in het bedrijf en de uit te voeren afstudeeropdracht?**

Tom heeft zich snel ingewerkt. Goed was ook dat hij snel zijn PvA op orde had en goed voor ogen had, hoe hij de zaken wilde aanpakken.

2. **Hoe beoordeelt u de communicatieve vaardigheden van de student (in de samenwerking met collega's, in contacten met de opdrachtgever, bij mondelinge presentaties, schriftelijke rapportages)?**

De communicatieve vaardigheden van Tom zijn voldoende, maar voor verbetering vatbaar. Tom heeft duidelijke progressie geboekt tijdens zijn afstuderen en daar hard aan gewerkt. Mondelinge toelichtingen en presentaties mogen rustiger.

3. Hoe heeft de student tijdens het uitvoeren van de opdracht gefunctioneerd?

- **Qua verantwoordelijkheid** **goed** / voldoende / matig / onvoldoende
- **Qua zelfstandigheid** goed / voldoende / matig / onvoldoende
- **Qua planmatig werken** goed / voldoende / matig / onvoldoende
- **Qua creativiteit** goed / voldoende / matig / onvoldoende
- **Qua productiviteit** goed / voldoende / matig / onvoldoende
- **Qua samenwerken met collega's** goed / voldoende / matig / onvoldoende
- **Qua draagvlakontwikkeling** goed / voldoende / matig / onvoldoende
- **Qua inspelen op bedrijfscultuur** goed / voldoende / matig / onvoldoende
- **Qua rekening houden met de specifieke context van het bedrijf** goed / voldoende / matig / onvoldoende
- **Qua het op gang brengen van de nodige veranderingen** goed / voldoende / matig / onvoldoende

4. Hoe beoordeelt u de kennis en kunde van de student in verhouding tot wat u verwacht van een bijna afgestudeerde?

De kennis en kunde van Tom is goed. Voor zijn afstudeeropdracht heeft hij specifieke trainingen gevolgd, maar pakte alles snel op.

5. Hoe beoordeelt u de kwaliteit van de opgeleverde (tussen)producten?

Goed. Tom levert zijn spullen netjes op en de kwaliteit is goed**.**

6. Bent u tevreden over het opgeleverde (eind)product?

- **In hoeverre heeft u gekregen wat is afgesproken?**

Ik heb gekregen wat is afgesproken. De afspraken zijn steeds in goed overleg bijgesteld en Tom heeft kunnen inspelen op de gewijzigde wensen.

- **In hoeverre voldoet het (eind)product aan uw verwachtingen?**

Met de uitkomst van het onderzoek van Tom hebben we een besluit genomen om MongoDB niet in te gaan zetten. Hij heeft ons dus van goede informatie voorzien, zodat wij een strategisch besluit konden nemen. Het voldoet dus aan de verwachtingen.

- **Wat is de bruikbaarheid en onderhoudbaarheid hiervan?**

Zie boven.

- **Wat gebeurt er met het opgeleverde (eind)product?**

Op basis van het eindproduct is besloten om MongoDB niet in te zetten. De impact zou enorm zijn op zowel de ontwikkeling als het onderhoud. Door het onderzoek van Tom besparen we ontzettend veel tijd, geld en energie.

- **Kunt u direct met het opgeleverde product aan de slag?**

Ja

7. Zijn er nog aspecten voor u van belang die nog niet aan de orde zijn geweest?

Tom is een zeer enthousiaste en gedreven IT-er. Het was een plezier om Tom te begeleiden.

8. **Bent u bereid een volgende keer weer uw medewerking te verlenen aan het beschikbaar stellen van een afstudeerplaats** (graag met toelichting)**?**

Ja. Info Support stelt ieder jaar veel plaatsen ter beschikking.

DE HAAGSE HOGESCHOOL

Info Support Solid Innovator

Bijlage L - Beoordeling tussentijds assessment

MongoDB: Een geschikte database voor het kennismanagementsysteem knowNow?

Bespreking concept	**Tussentijds assessment**	Eerste beoordeling

Formulier tussentijds assessment

Student:
T.Keim

Studentnummer:
11031425

Datum:
07-09-
2015

eerste TTA: 27-8-2015

Tijdens het tussentijds assessment is het volgende geconstateerd:		**ja**	**nee**
a	*Het voortgangsverslag is ontvangen*		x
b	*Het afstudeerdossier is digitaal beschikbaar*		x
c	*Het afstudeerdossier is opgebouwd conform de richtlijnen*		x
d	*Het goedgekeurde afstudeerplan is aanwezig*		x
e	*Het plan van aanpak is aanwezig*		x
f	*Reeds geleverd commentaar is aanwezig*		x
g	*Het afstudeerdossier geeft voldoende inzicht in de stand van zaken*	x	
h	*De afstudeeropdracht is tot nu toe naar behoren uitgevoerd*	x	

Aanpak	**O**	**T**	**V**	**G**
Passend			x	
Theoretisch verantwoord			x	
Samenhang uitvoering beroepstaken			x	

Beroepstaken op afgesproken niveau uitgevoerd?		**O**	**T**	**V**	**G**
1	2.2 Ontwerpen, bouwen en bevragen van een database		x	x	
2	3.1: ontwerpen systeemarchitectuur			x	
3	3.3: Bouwen applicatie		x	x	
4	3.5: Uitvoeren van en rapporteren over het testproces		x	x	

Producten	**O**	**T**	**V**	**G**
Tussenproducten			x	
Eindproducten			x	

Effectief communiceren	**O**	**T**	**V**	**G**
Binnen afstudeerbedrijf daar is geen zicht op				
Afstudeerdossier		x	x	

Reflectie	**O**	**T**	**V**	**G**
Inzicht in eigen functioneren			x	
Inzicht in eigen leerproces			x	

<u>Toelichting per beoordelingscriterium</u>

Aanpak
Er is al behoorlijk wat werk uitgevoerd. De gekozen aanpak past bij het op te lossen probleem.

Beroepstaken op afgesproken niveau uitgevoerd?
Beroepstaak 2.2: onderzoek Mongo DB en ontwerp van 2 documentschema's met in achtneming van performance en beschikbaarheid van data Beroepstaak 3.1: er ligt een architectuur ontwerp voor het te bouwen prototype van knowHow (microservices architectuur) Beroepstaak 3.3: voor elke service uit de microservices architectuur is een applicatie ontwikkeld, deels gebruikmakend van nieuwe technieken en toepassing van design patterns Beroepstaak 3.5: Job queue (onderdeel van de POC) is getest met behulp van de procescyclustest

Producten
Wat er ligt is van redelijke kwaliteit. Opgeleverde producten heten anders dan in de opdrachtsomschrijving aangegeven. Het is

wenselijk om een toelichting te geven waarom producten anders zijn geworden dan afgesproken. In het eindverslag mag meer aandacht besteed worden aan ontwerp en bouw prototype. Onderzoekgedeelte in het eindverslag is nu relatief groot.

Effectief communiceren
Binnen bedrijf is nu niet te beoordelen omdat er nog geen evaluatie is ontvangen. Onderzoek krijgt veel aandacht, prototype veel minder. Hierover is de bijlage wel meer te vinden.

Reflectie
Is in orde

Advies

x	Inleveren **(bindend advies)**
	Verlengen **(vrijblijvend advies)**
	Stoppen **(vrijblijvend advies)**

Besluit student

Aankruisen welke beslissing de student heeft genomen (alleen na vrijblijvend advies)

x	**Afstudeerdossier wordt op afgesproken datum ingeleverd**	Inleverdatum: 5-10-2015
	Afstudeerperiode wordt verlengd	Inleverdatum:
	Student stopt met afstudeeropdracht	

Naam begeleidend examinator: G.A.Mijnarends

Naam tweede examinator: H.G.J.Bechet-Tjoonk

Datum: 7 sept 2015

Dit formulier wordt door de tweede examinator digitaal ingevuld, waarna de begeleidend examinator het per email verstuurt naar de student met een cc naar de coördinator van ICT & Media @ Work (**A.M.Schipper@hhs.nl**). Het formulier dient door de student te worden opgenomen in het afstudeerdossier.