Ontwerpen serverarchitectuur add-on platform FontoXML.

2015

Titel	: Ontwerpen serverarchitectuur add-on platform FontoXML.
Student	: I. J. Post
Studentnummer	: 09003614
Onderwijsinstelling	: De Haagse Hogeschool
Opleiding	: Informatica
Afstudeerperiode	: 09-02-2015 / 05-06-2015
Examinatoren	: dhr. J. J. Smeets Mevr. A.A.A.M. Jacobs
Bedrijf	: Liones
Afdeling	: FontoXML
Bedrijfsbegeleider	: dhr. B. Willems
Versie	: 1.0

Referaat

Post, I, J, Afstudeerverslag – ‘Ontwerpen serverarchitectuur add-on platform FontoXML.’, Rijswijk, Liones, februari 2015.

Dit afstudeerverslag is geschreven ter afronding van de opleiding Informatica aan De Haagse Hogeschool te Zoetermeer. Dit afstudeerverslag beschrijft de werkzaamheden die ik, gedurende de periode februari tot en met juni 2015, heb uitgevoerd met betrekking tot de opdracht ‘Ontwerpen serverarchitectuur add-on platform FontoXML’. Aan de hand van dit afstudeerverslag zullen de uitgevoerde werkzaamheden inzichtelijk gemaakt worden.

Gedurende de afstudeerperiode heb ik mij bezig gehouden met het ontwerpen van de server-architectuur voor het add-on platform voor FontoXML. FontoXML is een web-based XML editor ontwikkeld door Liones voor de auteurs van grote publicatie organisaties. De uitgevoerde werkzaamheden zijn het uitvoeren van een onderzoek naar beschikbare serveromgevingen en het ontwikkelen van een proof of concept.

Descriptoren

- Liones
- FontoXML
- Websockets
- Serveromgevingen
- Webservices

Versiebeheer

Versie	Datum	Auteur	Mutatie
0.1	13-02-2015	Ivar Post	+ Opzet initiële structuur. + §1, §2.1, §2.2, §2.3, §2.4, §3, §4.1, §4.2, §4.3, §4.4.
0.2	05-03-2015	Ivar Post	+ §4.5.3, §5.1, §5.2, §5.3.
0.3	20-03-2015	Ivar Post	+ §5.3 + §5.4 + §5.4.1 + §5.4.2 + §5.4.3 + §5.4.4.
0.4	30-03-2015	Ivar Post	+ §5.5 + §5.5.1 + §5.5.2 + §5.5.3.
0.5	08-04-2015	Ivar Post	+ §5.5.4.
0.6	20-04-2015	Ivar Post	review §1 tot en met §5.
0.7	01-05-2015	Ivar Post	Verwerken Feedback J.J.Smeets.
0.8	02-05-2015	Ivar Post	Verwerken Feedback Bedrijfsmentor.
0.9	04-05-2015	Ivar Post	Herstructureren afstudeerverslag.
0.10	05-05-2015	Ivar Post	Afronding 80% versie.
0.11	18-05-2015	Ivar Post	TTA feedback verwerkt in §4 + §5 + §6 + §7 + §8.
0.12	19-05-2015	Ivar Post	TTA feedback verwerkt in §3.3 + §8 + §5 verwijderd.
0.13	22-05-2015	Ivar Post	TTA feedback verwerkt in §9.
0.14	24-05-2015	Ivar Post	+ §12
0.15	26-05-2015	Ivar Post	TTA feedback verwerkt in §10 + §11
0.16	27-05-2015	Ivar Post	Feedback verwerkt van Tina Hanemaaijer en Bedrijfsmentor. Afronden §11.
0.17	29-05-2015	Ivar Post	Afronding feedback TTA.
0.18	01-06-2015	Ivar Post	Spellingscontrole.
1.0	03-06-2015	Ivar Post	Afronding afstudeerverslag

Voorwoord

Dit verslag beschrijft het proces dat door mij, student Informatica aan de Haagse Hogeschool te Zoetermeer, is uitgevoerd tijdens het afstudeertraject bij Liones. Het verslag gaat in op de ondernomen activiteiten voor het project “Ontwerpen serverarchitectuur add-on platform FontoXML”, dat bij Liones te Rijswijk is uitgevoerd. In dit document staat beschreven hoe het project is vormgegeven, welke werkzaamheden zijn uitgevoerd en welke keuzes zijn gemaakt.

Dit verslag is onder andere geschreven voor de examinatoren, Dhr. J.J. Smeets en Mevr. A.A.A.M. Jacobs, die aan de hand van dit verslag voldoende inzicht zullen hebben over het traject en aan de hand hiervan een beoordeling kunnen geven.

De volgende mensen wil ik graag bedanken voor de ondersteuning die ze mij hebben gegeven tijdens mijn afstudeerproject.

Allereerst wil ik graag mijn examinatoren Dhr. J.J. Smeets en Mevr. A.A.A.M. Jacobs bedanken voor de feedback die zij mij tijdens mijn afstudeertraject hebben gegeven. De feedback heeft mij geholpen om dit document zo compleet en duidelijk mogelijk te maken.

Daarnaast wil ik graag mijn bedrijfsmentor Dhr. B Willems bedanken voor het begeleiden van mij tijdens het afstudeertraject. En alle collega’s met wie ik heb mogen werken tijdens mijn afstudeertraject voor de gezellige werksfeer.

Ivar Post

Rijswijk, Juni 2015

Inhoudsopgave

Inleiding

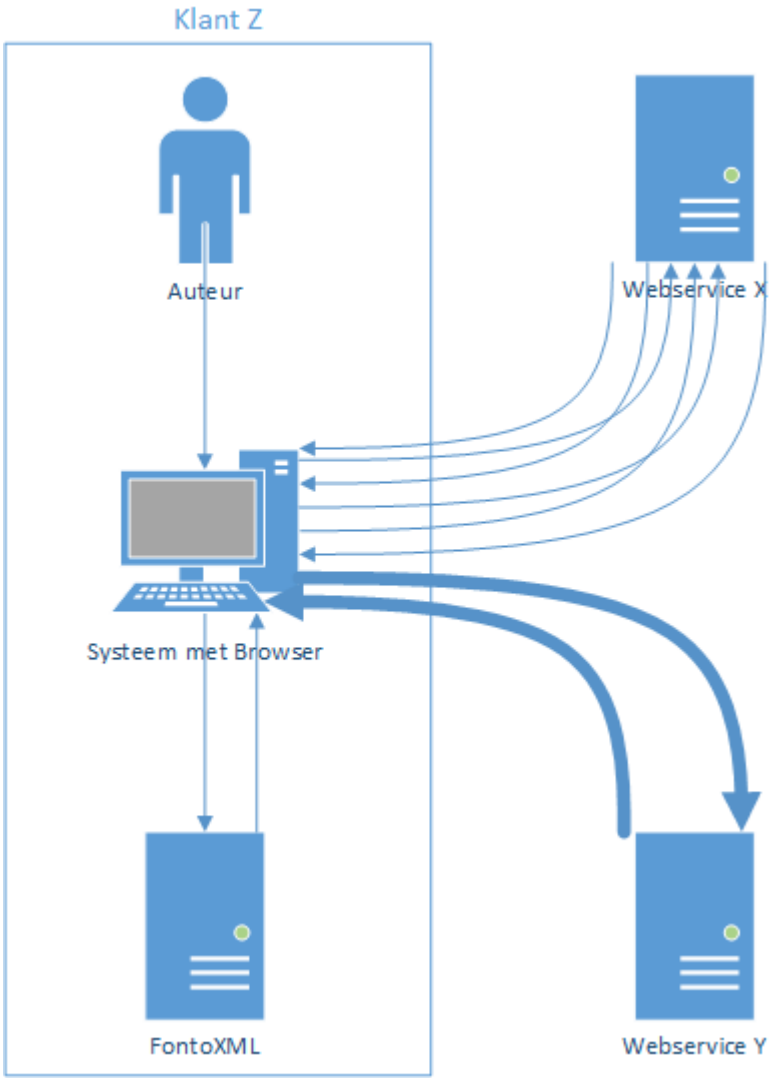
Dit document beschrijft het proces: “Ontwerpen serverarchitectuur add-on platform FontoXML.” dat is gedaan als afstudeerstage. De afstudeerstage is uitgevoerd in de periode van februari 2015 tot en met juni 2015. Tijdens dit project heb ik de serverarchitectuur ontwikkeld die FontoXML in staat moet stellen om via één centrale server met add-ons in verbinding te kunnen staan. De add-ons worden door derde partijen ontwikkeld, beheerd en aangeboden. De add-ons maken gebruik van webservices. De serverarchitectuur die ik ga ontwikkelen is een platform die in verbinding staat met deze webservices. De cliënt kan via de centrale server gebruik maken van de add-ons. De server zal de aanvragen doen om informatie te ontvangen van deze webservices. Het project focust zich op de ontwikkeling van de architectuur van de centrale server.

Om te beginnen heb ik kennis opgedaan over de huidige situatie en de problemen die hierbij optreden. Op basis van deze informatie ben ik gaan onderzoeken op welke manier een nieuw ontwikkelde centrale server de situatie kan verbeteren. Op de gekozen serveromgeving ben ik een proof of concept gaan ontwikkelen om de verbeteringen inzichtelijk te maken.

Het doel van dit document is om mijn werkzaamheden tijdens deze periode duidelijk en inzichtelijk te maken. Er wordt ingegaan op de gemaakte keuzes en deze worden onderbouwd.

1. Probleemomschrijving

Liones ontwikkelt FontoXML. FontoXML biedt extra functionaliteiten, zoals spellingscheckers of grammaticacontroles, aan door middel van het gebruik van add-ons. Deze add-ons worden door derde partijen ontsloten via webservices. De vorm waarop met de services wordt gecommuniceerd hangt af van de webservice waarmee verbinding wordt gemaakt. FontoXML haalt alle data uit een XMLDOM structuur. XMLDOM is een combinatie van XML en van DOM. XML staat voor eXtensible Mark-up Language en DOM staat voor document object model. Gecombineerd wordt het een standaard om XML-elementen te manipuleren. Alle documenten in FontoXML zijn XML documenten. FontoXML zorgt ervoor dat auteurs geen kennis nodig hebben van XML om hier gebruik van te maken. Hierdoor moet FontoXML eenvoudig data kunnen aanpassen in documenten en hiervoor wordt de DOM structuren toegepast op de XML documenten. De auteur ziet nu geen XML structuren en kan eenvoudig data aanpassen in het document. De auteur krijgt een werkervaring vergelijkbaar met standaard tekstbewerkers zoals Microsoft Word of Google Documents.



Figuur 1: Huidige situatie van communicatie tussen FontoXML en webservices.

De add-ons worden op verschillende niveaus aangeroepen. Voor de spellingscontrole wordt bijvoorbeeld per woord de add-on gebruikt en stuurt de add-on per woord een aanvraag naar de webservice. Voor de grammatica controle wordt per zin de add-on gebruikt. Deze aanvraag wordt minder aangeroepen en heeft per aanvraag meer informatie dan de spellingscontrole.

Ik ga onderzoeken of een centrale server de communicatie met de webservices kan afhandelen en verbeteren.

Tevens bekijk ik de huidige webservices die worden gebruikt en onderzoek ik of er nieuwe webservices of andere webservices gebruikt kunnen worden. De webservices maken gebruik van verschillende communicatievormen en de nieuwe omgeving zal deze vormen ondersteunen.

2. Organisatie

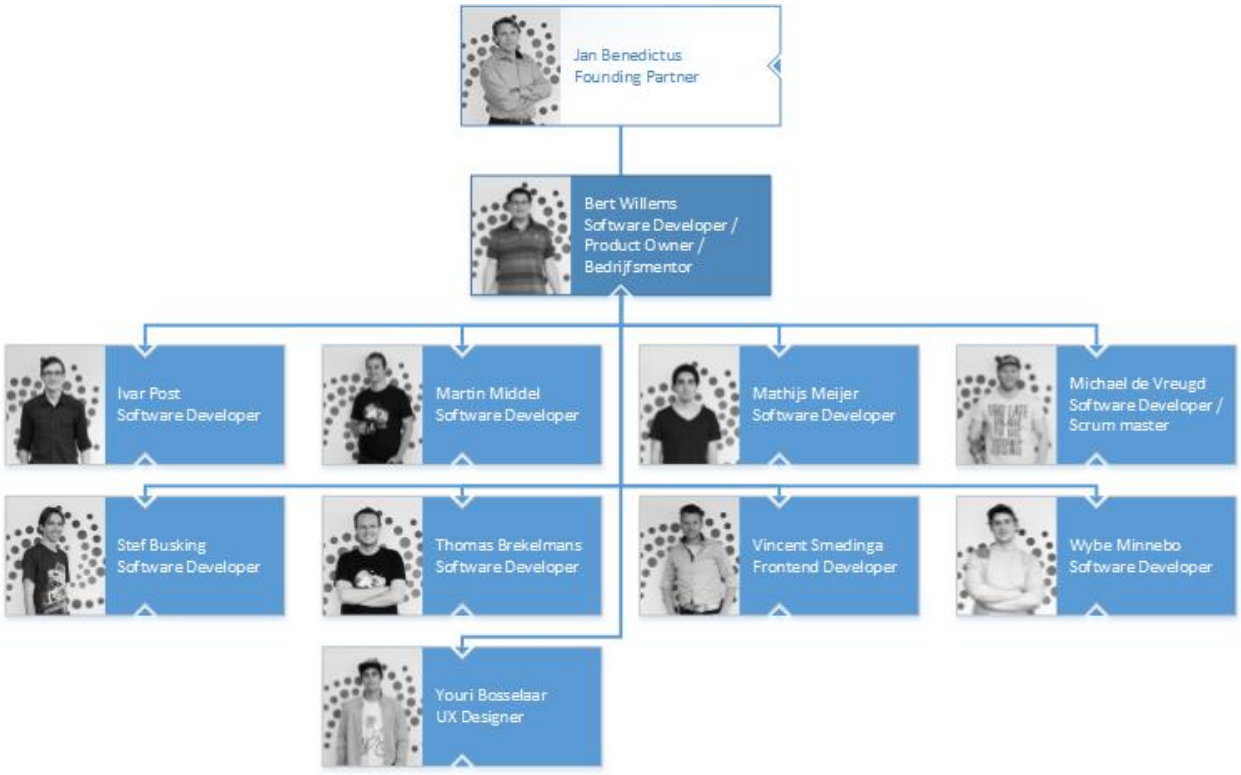
2.1 Liones

Liones is een web development bedrijf dat op verschillende gebieden voor hun klanten adviezen of applicaties uitbrengt. Liones is op verschillende gebieden actief zoals web analyse, User Interaction Design, ontwikkeling en content strategie. Liones heeft ook de applicatie FontoXML ontwikkeld. Liones beheert en ontwikkelt deze applicatie nog steeds.

2.2 FontoXML

FontoXML is een web-based XML editor en wordt gebruikt door de klanten die veel publicaties opleveren. Deze publicaties bestaan in de vorm van artikelen, vacatures, tentamens en handleidingen. Het gaat om publicaties die bestaan uit grote hoeveelheden aan tekst. FontoXML maakt gebruik van webservices om de tekst bijvoorbeeld te controleren op spelling, grammatica of terminologie.

Voor FontoXML zijn verschillende medewerkers bezig met het door ontwikkelen van de applicatie. Hiervoor heb ik een organogram opgesteld om duidelijkheid te scheppen over wie zich bezig houden met FontoXML. De belangrijkste contactpersoon voor mij is Bert Willems. Dit is de persoon waar ik het meest contact mee heb gehad. De andere medewerkers zijn altijd beschikbaar voor feedback of om vragen aan te kunnen stellen.



Figuur 2: Organogram afdeling FontoXML

2.3 Klanten

FontoXML heeft een aantal klanten die gebruik maken van de applicatie. Het bedrijf dat het intensiefst gebruik maakt van FontoXML zal uitgelicht worden om een duidelijker beeld te scheppen over de toepassing van FontoXML. De overige bedrijven zullen vergelijkbaar gebruik maken van FontoXML.

- RILM
- IDG
- Sanoma
- Kluwer

Thieme Meulenhoff

Thieme Meulenhoff is een uitgever van educatieve literatuur en is bezig om zichzelf te transformeren van een boekuitgever naar een “Learning Design Company”. De nieuwe weg die Thieme Meulenhoff wil inslaan houdt in dat ze meer gaan focussen op de vraag van individuele instellingen. Hierdoor wordt er nog meer gepubliceerd. Al deze publicaties moeten wel aan bedrijfsspecifieke standaard structuren, vorm en inhoud kunnen voldoen. FontoXML controleert de documenten op de bedrijfsspecifieke standaarden en geeft feedback aan de auteur zodat de auteur hiervoor aanpassingen kan maken. Thieme maakt op het moment van schrijven het intensiefst gebruik van FontoXML. FontoXML sluit aan bij de bedrijfswensen die Thieme Meulenhoff heeft. Door het intensieve gebruik bij Thieme Meulenhoff krijgt FontoXML veel feedback voor verbeteringen van FontoXML.

3. Aanpak

3.1 Oriënteren

Na de oriëntatiefase moet ik een duidelijk beeld hebben over de huidige situatie en welke problemen zich nu voordoen. Door middel van afspraken met medewerkers moet ik de oriëntatiefase goed kunnen doorlopen en een beschrijving van de huidige situatie kunnen opzetten. Dit document is te vinden in bijlage #4.

3.2 Het onderzoek

Het onderzoek bestaat uit een aantal deelvragen die beantwoordt moeten worden. Als alle deelvragen beantwoord zijn moet ik een keuze kunnen maken over de vorm van communicatie, de technieken en methoden die ik ga gebruiken en moet ik een beter beeld hebben over de webservices die op dit moment beschikbaar zijn.

3.2.1 Vergelijkingsonderzoek

Om meer inzicht te krijgen over de mogelijk communicatievormen die de centrale server richting de cliënt kan gebruiken, voer ik een vergelijkingsonderzoek uit. Uit dit vergelijkingsonderzoek zal duidelijk worden welke technieken beschikbaar zijn voor de communicatie tussen cliënt en de centrale server. Op basis van deze eisen en de mogelijke variaties kan gekeken worden wat het verstandigst is. Hieruit zal een duidelijk beeld gevormd worden over hoe de centrale server gaat communiceren en welke beperkingen hier eventueel voor zijn.

Als hierover een keuze is gemaakt kan ik een vervolg onderzoek starten om te kijken welke technieken en methoden hiervoor gebruikt kunnen worden.

3.2.2 Vervolgonderzoek naar ondersteuning voor uitkomsten van voorgaand onderzoek

Dit onderzoek gaat in op de uitkomsten van het voorgaande onderzoek. Hierin onderzoek ik welke mogelijkheden er zijn om de gekozen communicatievormen te ondersteunen. Aan de hand van dit onderzoek moet er een besluit komen over de programmeertaal of platform waarop de server gebouwd zal worden.

3.2.3 Onderzoek naar webservices

FontoXML maakt gebruik van diverse webservices. Eén van de wensen van Liones is dat hierop wordt uitgebreid om meer wensen van klanten te kunnen ondersteunen. Aan mij is gevraagd of ik naar huidige en nieuwe webservices wil kijken en hoe deze functioneren. Het is belangrijk dat ik een duidelijk beeld krijg van de manieren waarop deze webservices communiceren en dat ik hier rekening mee houd tijdens het ontwikkelen van het prototype. Aan het eind van het afstudeertraject zullen deze webservices zo beschreven zijn dat er beslist kan worden of de webservices in gebruik worden genomen.

3.3 Prototype ontwikkeling.

Op basis van het onderzoek zal duidelijk zijn gemaakt waar het prototype mee moet kunnen functioneren. De gekozen communicatievorm kan beperkingen hebben. Het platform, de IDE en programmeertalen kunnen hier van afhankelijk zijn. FontoXML communiceert door middel van het gebruik van JsonML. JsonML is een combinatie van JSON en XML. JSON staat voor JavaScript Object Notation en is ontwikkeld om op een lichte manier data uit te kunnen wisselen. JsonML is

ontwikkeld om lichte XML structuren uit te kunnen wisselen. Hier dient tijdens de ontwikkeling van het prototype rekening mee gehouden te worden. De ontwikkelmethode die gehanteerd wordt is SCRUM. De verantwoording voor de gekozen ontwikkelmethode wordt in het Plan van Aanpak verder besproken.

3.4 Op te leveren producten

Plan van Aanpak

Dit document wordt in de eerste week opgezet en heeft als inhoud uitgebreide beschrijvingen over de tussenproducten en het bedrijf zelf. Dit document zal duidelijkheid scheppen over de omgeving waar ik werkzaam ben en wat de werkzaamheden van mij zijn. Ieder document dat aan het eind van het project wordt opgeleverd staat hierin omschreven. Deze documentomschrijvingen geven een duidelijker beeld over de inhoud en de resultaten van de documenten. Het is nuttig om dat al in het Plan van Aanpak te beschrijven zodat er al is nagedacht over wat ik wil bereiken met mijn activiteiten. In het plan komt een overzicht waar ik mijn aanpak beschrijft.

Beschrijving huidige situatie

In deze beschrijving komt een duidelijke omschrijving met mogelijke ondersteuning van modellen voor verduidelijking van de huidige situatie. Door middel van dit document verplicht ik mij dieper te oriënteren op de huidige situatie en hoe deze op dit moment werkzaam is. Na het schrijven van het document zal ik een duidelijker beeld hebben van de applicatie en de mogelijke probleemsituaties.

Onderzoeksrapport voor de centrale server

Een belangrijk onderdeel van de centrale server is de manier van communicatie die gaat plaatsvinden tussen de centrale server en FontoXML. Uit gesprekken voor de opdrachtomschrijving is naar voren gekomen dat het van belang is dat dit asynchroon gaat functioneren. Ik onderzoek de mogelijke manieren waarop de browser kan communiceren met de centrale server. De centrale server communiceert met de webservices. Er wordt onderzocht welke manier van communiceren het effectiefst is. Het belangrijkste hierbij is dat in de nieuwe situatie de browser ontlast wordt in vergelijking tot de huidige situatie. Als er duidelijkheid is over de vormen van communicatie die worden gehanteerd tussen de browser en de centrale server en tussen de centrale server en de webservices kan er gekeken worden naar de centrale server. De centrale server zit tussen de browser en de webservices en zal met beide kunnen communiceren. Er wordt gekeken op welke manier dit gefaciliteerd kan worden op de centrale server.

Onderzoeksrapport over beschikbare webservices

De centrale server zal samen kunnen werken met webservices. Het is aan mij om te bewijzen dat nieuwe webservices eenvoudig geïntegreerd kunnen worden. Daarom wordt aan mij gevraagd om nieuwe webservices te onderzoeken en te kijken hoe deze met de centrale server kunnen communiceren. Ik krijg hierdoor inzicht over verschillende voorwaarde waaraan de centrale server moet voldoen. Webservices kunnen verschillende manieren hebben om te communiceren. Tijdens de ontwikkeling van de centrale server moet hier rekening mee worden gehouden.

API omschrijvingen

Per webservice zal duidelijk zijn wat de mogelijkheden zijn en hoe deze kunnen functioneren. Welke functies zijn interessant om te kunnen gebruiken en hoe moeten deze gebruikt worden. Kan de gehele webservice direct geïntegreerd worden of moet er nog een adapter worden gebouwd voordat de webservice met de centrale server kan communiceren.

Verantwoording gekozen methoden en ontwikkeltools

Aan het eind van de oriëntatiefase zal ik een duidelijk beeld hebben van wat er van mij wordt verwacht in de ontwikkelfase. Met dit beeld kan ik besluiten nemen over frameworks, programmeertalen, methoden en tools. Iedere keuze die ik maak zal duidelijk omschreven worden. Dit document geeft inzicht over de mijn gemaakte keuzes.

Requirements document

De nieuwe omgeving en werking van de applicatie hebben verschillende eisen vanuit de klant. In dit document worden deze eisen bijgehouden en uitgewerkt. Aan het eind van het traject wordt dit document gebruikt om te kijken of het project aan de eisen van het bedrijf voldoet. Per requirement omschrijf ik waarom het wel of niet gelukt is.

Beschrijving centrale server

Na de oriëntatiefase zal duidelijkheid zijn over het framework of platform waarop het prototype zal gaan functioneren. Hiervoor kan voor het opzetten van de centrale server een structuur ontwerp ontwikkeld worden die inzicht geeft over welke systemen en functionaliteiten worden gebruikt. In dit ontwerp kunnen mogelijk al requirements opgelost worden.

Ontwerp prototype applicatie

Beschrijving en modellering van het prototype. Door middel van het gebruik van verschillende modellen zal inzichtelijk gemaakt worden hoe het prototype functioneert. Er moet duidelijkheid zijn over de data die het prototype gebruikt en hoe deze getransformeerd wordt binnen de applicatie.

Prototype applicatie

Een werkend prototype die laat zien dat via de centrale server gecommuniceerd kan worden met verschillende webservices. Tevens zal het laten zien dat de browser bij de cliënt waar FontoXML in draait wordt ontlast.

Omschrijving testcases en testmethode

Welke testmethode ga ik hanteren en hoe diep zullen de tests gaan. Dit document moet inzicht geven over de te uit te voeren test scenario's en methode. Ik verantwoord in dit document iedere gemaakte keuze die invloed heeft op het testproces.

Testresultaten

Na het ontwikkelen en uitvoeren van de opgezette testscenario's komen er resultaten uit. Deze resultaten zullen opgeschreven worden en enige bevindingen die hierdoor gemaakt zijn duidelijk verantwoord worden.

Adviesrapport

Aan het eind van het gehele traject lever ik een rapport op met mijn bevindingen en het uiteindelijke advies. Ik beschrijf wat het bedrijf aan mijn werk heeft en hoe het bedrijf het verder op kan pakken om het bedrijf in staat te stellen nieuwe ontwikkelingen uit te voeren.

Stageverslag

Iedere week neem ik de tijd om mijn stageverslag bij te werken. In dit document wordt mijn afstudeertraject verantwoord. Op basis van dit document zullen de examinatoren een conclusie kunnen trekken.

4. Het plan van Aanpak

Het plan van aanpak is een belangrijk onderdeel bij de opstart van het afstudeertraject. Tijdens het opzetten van het plan van aanpak had ik moeite met het opstellen van een planning. Dit kwam voornamelijk omdat de opdracht onderzoekgericht is en veel nog onduidelijk is aan het begin van het traject. Deze onzekerheid is besproken met de bedrijfsmentor welke heeft kunnen bevestigen dat de focus ligt op het onderzoek. De bedrijfsmentor heeft duidelijk gemaakt dat er onzekerheden zijn bij het opzetten van de planning en het maken van de scope. Er zou meer duidelijkheid komen als het onderzoek werd uitgevoerd. De globale werkzaamheden waren wel duidelijk maar het was lastig om te specificeren wat de exacte uitkomsten waren. Tevens heb ik moeite gehad bij het opstellen van de scope en de afbakening. Het belangrijkste bij de scope en afbakening waren de vragen over hoe diep de te uit te voeren onderzoeken moesten gaan. Ik vond het lastig om hier duidelijk antwoord op te kunnen geven omdat ik geen voorkennis had over wat er over de onderwerpen te vinden zou zijn. Om de onderzoeken niet te groot te maken wou ik een afbakening hebben over de hoeveelheid aan mogelijkheden die gevonden moeten worden voor een goed onderzoek. Als laatste heb ik een onderzoeksmethode moeten kiezen. Omdat mijn traject grotendeels bestaat uit onderzoek kwam het ontwikkeltraject pas later terug. Het ontwikkeltraject is afhankelijk van het uit te voeren onderzoek. Het onderzoek bepaalt welke technieken gebruikt worden tijdens het ontwikkeltraject. Op basis van deze keuzes kan de ontwikkelmethode beïnvloedt worden.

Ondanks de onzekerheden die het onderzoek met zich mee draagt heb ik wel keuzes kunnen maken over verschillende methodes en technieken

4.1 Methoden en technieken

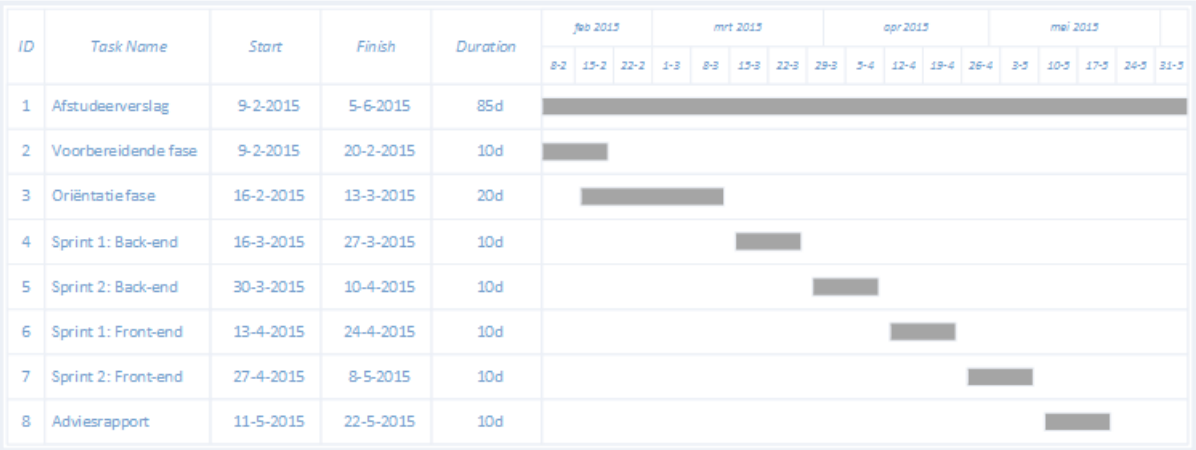
4.1.1 SCRUM

Het ontwikkelen van de centrale server is gebeurd in meerdere sprints. Per sprint heb ik nieuwe inzichten kunnen opdoen en deze kunnen verwerken in het UML model dat is gemaakt voor de centrale server. Per sprint wordt beschreven wat er is gebeurd en waarom er aanpassingen zijn gemaakt ten opzichte van het vorige model. Iedere sprint heeft een moment waar mijn bedrijfsmentor en ik samen hebben gezeten om nieuwe inzichten te bespreken en nieuwe werkzaamheden vast te stellen voor de volgende sprint. Stef Busking heeft meerdere malen aangeschoven om zijn mening te geven over de communicatievorm tussen de cliënt en de centrale server.

Om te kunnen ontwikkelen in sprints heb ik vanaf het begin al geneigd naar het gebruik van SCRUM. Dit omdat Liones deze ontwikkelmethode gebruikt tijdens hun eigen werkzaamheden.

Helaas is SCRUM niet toepasbaar op mijn project omdat ik het alleen uitvoer. Bij SCRUM worden daily stand-up meetings gehouden met het projectteam. Tijdens mijn traject zou het projectteam dan uit één project lid bestaan. Mijn voorkeur ligt bij agile methoden omdat hierbij sprints in te plannen zijn in tegenstelling tot waterval methodieken waar je vast zit aan de fase tot deze compleet is afgerond. Bij andere agile methoden, zoals Extreme Programming, wordt ook het werken in een projectteam genoemd en loop ik tegen dezelfde problemen aan als bij SCRUM. Naar aanleiding van gesprekken met collega's die bij Liones ook afgestudeerd zijn en SCRUM hebben gebruikt tijdens hun afstuderen heb ik opnieuw gekeken of SCRUM toe te passen is op mijn project. Hieruit werd

duidelijk dat de elementen die ik niet uit kon voeren in mijn eentje wel toepasbaar waren door mee te draaien met de stand-ups van het FontoXML team. Aan de hand van deze keuze heb ik mijn planning kunnen opstellen en bij het ontwikkeltraject verschillende sprints kunnen definiëren.



ID	Task Name	Start	Finish	Duration
1	Afstudeerverslag	9-2-2015	5-6-2015	85d
2	Voorbereidende fase	9-2-2015	20-2-2015	10d
3	Oriëntatie fase	16-2-2015	13-3-2015	20d
4	Sprint 1: Back-end	16-3-2015	27-3-2015	10d
5	Sprint 2: Back-end	30-3-2015	10-4-2015	10d
6	Sprint 1: Front-end	13-4-2015	24-4-2015	10d
7	Sprint 2: Front-end	27-4-2015	8-5-2015	10d
8	Adviesrapport	11-5-2015	22-5-2015	10d

Figuur 3: Gantt-Diagram van opgestelde planning.

4.1.2 Requirements

Om de server op te zetten en deze te laten voldoen aan de eisen die de opdrachtgever geeft zijn er requirements opgesteld. De requirements zijn achterhaald door middel van meetings die met de opdrachtgever zijn ingepland. Na de meeting zijn de requirements uitgewerkt en zijn ze geprioriteerd. De prioritering is gedaan aan de hand van de MoSCoW methode en in samenwerking met de opdrachtgever.

De MoSCoW methode is gekozen omdat hiermee duidelijk aan kan gegeven worden welke requirements verwerkt zullen worden en welke als eerst vervallen als er niet genoeg tijd is.

4.1.3 Overige methodes

Wegens het onderzoek dat uitgevoerd zal worden is het nog onduidelijk hoe de centrale server ontwikkeld zal worden. Na het onderzoek zal meer duidelijkheid zijn over de gebruikte technieken en methodes om de centrale server te ontwikkelen.

5. Het onderzoek

5.1 Doel van het onderzoek

Het project is begonnen met een onderzoek. Aan de hand van dit onderzoek zal ik kennis opdoen over beschikbare communicatievormen die gebruikt kunnen worden tussen FontoXML en de centrale server. Hierna zal ik een vervolg onderzoek uitvoeren om te kijken op welke manier de gekozen communicatievorm ondersteund kan worden. En een onderzoek naar beschikbare webservices en welke verschillende communicatievormen de webservices gebruiken.

Het doel van dit onderzoek is om kennis op te doen over beschikbare technieken en hierover een goed onderbouwd besluit te kunnen nemen.

5.2 Definiëren van hoofdvraag en deelvraag

Aan het begin van het onderzoek ben ik duidelijk gaan omschrijven welke hoofdvraag ik beantwoord wil hebben en hoe ik deze wil beantwoorden door middel van deelvragen. Het onderzoek moet helpen bij keuzes die gemaakt worden om de structuur van de centrale server op te kunnen zetten. Het doel van het project is om een structuur voor een server op te zetten die de add-ons van FontoXML kan faciliteren. Hierin is de communicatie tussen de verschillende systemen belangrijk. Hieruit is dan ook de volgende hoofdvraag gekomen:

"Welke communicatievormen moet de centrale server kunnen ondersteunen om de add-ons van FontoXML op een asynchrone manier te kunnen faciliteren?"

Aan de hand van de hoofdvraag ben ik deelvragen gaan opstellen. De conclusies van deze deelvragen dienen samen een antwoord te kunnen geven op de hoofdvraag. Een belangrijk onderdeel van het onderzoek is de communicatievorm tussen FontoXML en de centrale server. Hieruit is dan ook de volgende deelvraag ontstaan:

"Welke vorm van communicatie tussen de cliënt en de centrale server is het meest effectief?"

Om deze deelvraag te kunnen beantwoorden heb ik de verschillende communicatievormen vergeleken. De communicatievormen die bekeken zijn kunnen alleen vanuit een browser gestart worden. Dit komt omdat FontoXML een applicatie is die in de browser draait.

Aan de hand van deze deelvraag is nog een vervolgvraag opgesteld. Deze deelvraag zal belangrijke input leveren voor de ontwikkeling van de centrale server.

"Welke technieken kunnen de gekozen communicatievorm ondersteunen?"

Voor deze deelvragen worden verschillende technieken bekeken die de communicatievorm kunnen ondersteunen. De gekozen techniek zal gebruikt worden in de ontwikkeling van de centrale server.

Naast de communicatie tussen FontoXML en de centrale server zal er ook communicatie plaatsvinden tussen de webservices en de centrale server om de add-ons van FontoXML te kunnen faciliteren. De centrale server moet ook deze communicatievormen ondersteunen om zo flexibel mogelijk te zijn. Omdat webservices beheert worden door derde partijen kunnen deze niet aangepast worden. De centrale server moet rekening houden met verschillende beperkingen die webservices kunnen hebben. Om deze reden is de volgende deelvraag opgezet:

"Welke invloeden kunnen webservices hebben op de ontwikkeling van de centrale server?"

Om deze deelvraag te kunnen beantwoorden zullen er een aantal webservices bekeken worden om te bepalen welke beperkingen hierbij naar voren komen. En hoe deze beperkingen invloed kunnen hebben op de centrale server.

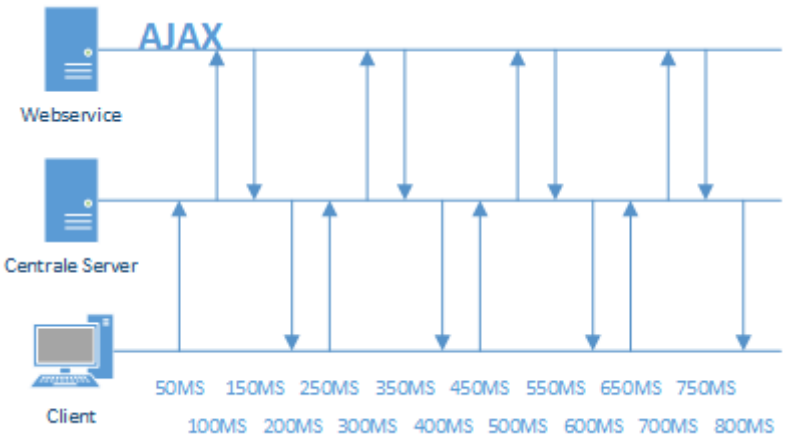
5.3 Vooronderzoek

Voor het vooronderzoek heb ik gekeken naar niveaus van verschillende communicatievormen. Ik ben begonnen met software distribution models. Software distribution models geven weer op welke manier software gedistribueerd wordt. Wordt de software compleet op het lokale systeem geïnstalleerd of wordt de software door middel van een server bereikbaar gemaakt voor de cliënten. Het software distribution model geeft een aantal vormen van mogelijke communicatievormen voor servers. Ik ben hier nauwelijks op ingegaan omdat het voor mijn gevoel niet het juiste niveau was. Na contact met mijn bedrijfsmentor bleek dat dit inderdaad niet is wat mijn bedrijfsmentor voor ogen had. Samen met de bedrijfsmentor is besloten om niet verder te kijken naar de software distribution model en op zoek te gaan naar andere vormen van communicatie. Verder zoekende naar communicatievormen ben ik verschillende protocollen tegengekomen. Deze protocollen beschrijven communicatievormen tussen een server en een browser. In overleg met de bedrijfsmentor is besloten dat hierop de focus van het onderzoek gaat liggen. Deze communicatievormen zijn gebruikt om een antwoord te geven voor de eerste deelvraag.

5.4 Het onderzoek

5.4.1 Welke vorm van communicatie tussen de cliënt en de centrale server is het meest effectief?

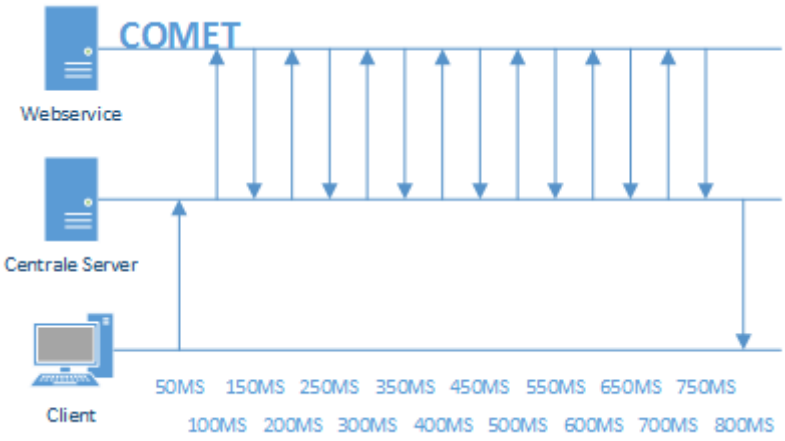
Met deze deelvraag heb ik de verschillende protocollen bekeken en vergeleken. Het standaard protocol dat gebruikt wordt door een browser die communiceert met een server is HTTP. Het HTTP protocol zelf ondersteund geen full-duplex communicatie. De nieuwe versie van HTTP, HTTP/2 gaat dit wel ondersteunen maar is op het moment nog niet volledig ontwikkeld en wordt ook nog niet door alle browsers ondersteund.



Figuur 4: Visualisatie van AJAX communicatie in de nieuwe situatie.

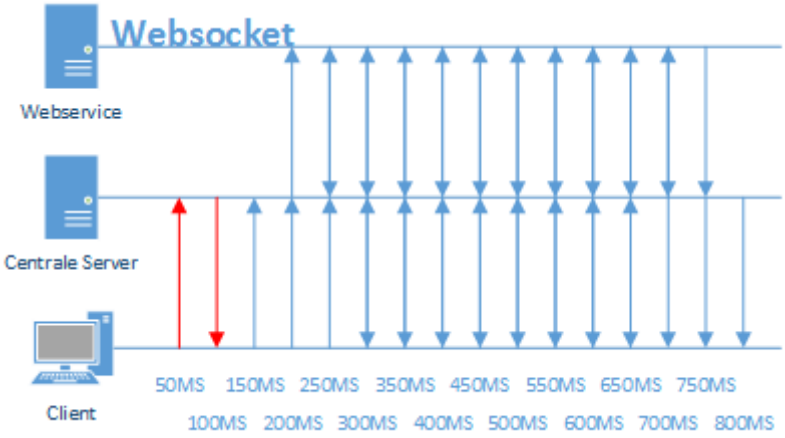
Via HTTP kan gecommuniceerd worden met een server maar kan zonder andere technieken niet asynchroon functioneren en is gebaseerd op een request / response basis. Door middel van technieken zoals AJAX kan er wel asynchroon gecommuniceerd worden vanaf de cliënt met een server maar deze vorm is niet full-duplex. Niet full-duplex kunnen communiceren, houdt in dat zodra er een request is gestuurd is er gewacht moet worden op de response en daarna wordt de connectie gesloten. In figuur 4 is te zien dat zowel tussen de cliënt en de server en tussen de server en de webservice met behulp van AJAX wordt gecommuniceerd.

De volgende techniek, COMET, simuleert een full-duplex communicatie maar is het niet. COMET is gebaseerd op dezelfde technieken als AJAX. De cliënt probeert via de centrale server verbinding te maken met de webservice. Over deze eerste verbinding wordt de data gestuurd. Als de data is gestuurd krijgt de server vaak een id-code terug om op een later moment data op te halen. Als de server de id-code heeft ontvangen wordt deze gebruikt om opnieuw te vragen of er data beschikbaar is over de data die als eerst is opgestuurd. Zodra de webservice aangeeft dat deze data beschikbaar is wordt deze teruggestuurd naar de cliënt.



Figuur 5: Visualisatie van COMET communicatie in de nieuwe situatie.

De laatste techniek is Websockets. Websockets zijn ontwikkeld om full-duplex te kunnen communiceren met servers. Een websocket stuurt in eerste instantie een HTTP request naar de server en vraagt om de verbinding te upgraden naar een websocket verbinding. Als de server dit ondersteund en ermee akkoord gaat dan kan de cliënt en de server in full duplex met elkaar communiceren. Dit zijn de rode lijnen in figuur 6. De cliënt hoeft niet te wachten tot de response van een request terug is om een nieuwe request te kunnen sturen. De server hoeft ook niet meer te wachten op een request van een cliënt en kan een response sturen zolang de verbinding in stand is. In figuur 6 is te zien dat de server en de cliënt tegelijkertijd data versturen naar elkaar. Ook lijkt de centrale server en de webservices tegelijkertijd te communiceren. Dit is niet het geval. De centrale server maakt elke keer een nieuwe AJAX verbinding met de webservice. Deze worden asynchroon aangeroepen om websockets zo goed mogelijk te kunnen ondersteunen. De websocket is een geüpgradede http protocol die full-duplex kan communiceren met een server. Omdat de websocket een geüpgradede versie van http is heeft de websocket ook een https variant voor een beveiligde verbinding.



Figuur 6: Visualisatie van websocket communicatie in de nieuwe situatie.

Websockets worden ondersteund in bijna alle programmeertalen waardoor er weinig rekening mee gehouden hoeft te worden tijdens de keuze voor de centrale server. De centrale server communiceert weer met webservices en hier moet wel rekening gehouden worden met invloeden die de webservices hebben.

Ik heb de vergelijkingen tussen AJAX, COMET en websockets voorgelegd aan mijn bedrijfsmentor en mijn eigen advies meegegeven. In mijn eigen advies heb ik de voorkeur gegeven voor websockets.

Dit omdat deze techniek toe staat om een Full-Duplex communicatie op te zetten tussen de centrale server en de cliënt. Samen met de bedrijfsmentor is besloten om mijn advies te volgen en om gebruik te maken van websockets.

5.4.2 Welke technieken kunnen de gekozen communicatievorm ondersteunen?

Na het bekend worden het protocol dat gebruikt gaat worden, ben ik gaan kijken wat voor libraries er bestaan voor Websockets. Veel van deze libraries zijn gebouwd op de low-level API die de programmeertaal zelf ondersteund. De libraries zijn high-level en om die reden veel vriendelijk in het gebruik tijdens het ontwikkelen van de centrale server. Voor FontoXML wordt Node.js gebruikt. Node.js is een platform waarmee netwerk applicaties kunnen worden gebouwd met javascript. Omdat FontoXML hier gebruik van maakt is het een reden waarom ik eerst ben gaan kijken welke libraries er beschikbaar zijn voor Node.js. Collega's bij Liones hebben zelf ook al gekeken naar verschillende websocket libraries uit interesse. Thomas Brekelmans heeft mij hierdoor gewezen op Socket.IO, een websocket library beschikbaar voor Node.js.

Figuur 7: Eenvoudige server structuur.

Socket.IO

Socket.IO noemde zichzelf de snelste websocket engine beschikbaar voor Node.js. Ik ben Socket.IO gaan bekijken. Door middel van het lezen van de beschikbare documentatie, API-beschrijving en het zelf ontwikkelen van een websocket server en cliënt heb ik de functionaliteiten bekeken. Socket.IO is eenvoudig in gebruik en heeft weinig regels code nodig om een eerste server op te kunnen zetten. Als Socket.IO wordt gebruikt zijn zowel de cliënt als de server afhankelijk van deze library. Op het moment zijn alle FontoXML versies zo geprogrammeerd dat dit geen problemen zou kunnen opleveren.

Socket.IO maakt gebruik van groepen om cliënten te kunnen groeperen. De library heeft ingebouwde functionaliteiten om te achterhalen hoeveel open connecties er op dit moment zijn en kan achterhalen wie er berichten stuurt. De cliënt verstuurt een bericht via de websocket verbinding naar de server. Eenmaal als de request is ontvangen bij de server kan de server deze terug sturen over de websocket verbinding naar de cliënt die de request heeft gestuurd, een groep van cliënten die aangemaakt is of ieder verbonden cliënt. De server kan niet van de ene cliënt naar een willekeurige cliënt berichten door sturen.

WS

Socket.IO maakt gebruik van WS. een andere websocket library voor Node.js die lager level is dan Socket.IO. WS is gebouwd met focus op prestatie en wordt nog steeds verder ontwikkeld. WS is minder vriendelijk dan Socket.IO omdat deze alleen maar data heen en weer kan sturen. Socket.IO heeft de mogelijkheid om aan een bericht een naamgeving toe te voegen. Hierdoor kan deze heel gemakkelijk aan de server of de cliënt kant worden afgevangen en verwerkt worden. WS ondersteund wel een callback functie bij een request in tegen stelling tot Socket.IO. Hierdoor ondersteund WS lang openstaande requests die dankzij de callbacks direct worden afgehandeld als deze klaar zijn. WS is tevens onafhankelijk van wat er bij de gebruiker gebruikt wordt. De cliënt kan met de WS server verbinden door middel van het standaard websocket object dat gebruikt wordt in alle moderne browsers.

SockJS

SockJS is een verzameling aan libraries die in veel verschillende programmeertalen beschikbaar is. Het stelt libraries beschikbaar voor Node.js, Erlang, Python, Cyclone, Twisted, Tornado, Java, Ver.x, Scala, JavaEE, Play Framework en Netty. Bij de server varianten hoort een cliënt side SockJS library. Voor de cliënt wordt alleen de JavaScript cliënt library aangeboden. De cliënt kan met alle bovenstaande server libraries communiceren.

SignalR

SignalR is een websocket server library die gebouwd is op ASP.NET 4.5. SignalR probeert altijd gebruik te maken van websockets en als de cliënt dit niet ondersteund zal SignalR automatische terug vallen op oudere technieken, zoals AJAX of COMET, waar de cliënt wel gebruik van kan maken. Als de server wordt gehost op het framework Azure van Microsoft kan tijdens het ontwikkelen hiermee rekening gehouden worden. Als dit goed wordt toegepast zal de server automatisch zichzelf kunnen schalen op de hoeveelheid gebruikers. Normaal gesproken leidt dit tot problemen doordat er verschillende servers komen met verschillende IP-adressen. Doordat SignalR goed geïntegreerd is met Azure worden deze problemen voorkomen zonder dat de ontwikkelaar er teveel rekening mee moet houden.

SignalR ondersteund een identificatie code voor de verbonden cliënt en kan daarom weten wie welke cliënt is en op basis daarvan berichten gaan versturen. Om deze redenen heb ik samen met mijn bedrijfsmentor het besluit genomen de centrale server met SignalR te ontwikkelen.

5.4.3 Welke invloeden kunnen webservices hebben op de ontwikkeling van de centrale server?

Door het uitvoeren van dit onderzoek heb ik inzichten willen halen over de bestaande webservices. Waar kunnen webservices voor gebruikt worden en op welke manier moet er met webservices gecommuniceerd worden? Aan het begin van het onderzoek naar webservices werd al snel duidelijk dat de hoeveelheid aan vrij beschikbare services groot is. De focus van de webservices die ik moest vinden lag op tekstuele ondersteuning. Dit omdat FontoXML hier meer functionaliteiten uit kan halen en kan aanbieden aan klanten. Tijdens het onderzoek werd duidelijk dat de webservices die nuttig zijn op verschillende niveaus gecategoriseerd kunnen worden. Deze niveaus geven weer op welke niveaus webservices data willen ontvangen of op welk niveau deze data zullen terug geven. Deze niveaus zijn fragment levels genoemd.

Fragment Levels:

- Woordniveau.
- Zinniveau.
- Tekstniveau.
- Inhoudsniveau.
- Structuurniveau.

Op woordniveau roept de applicatie de webservice iedere keer aan als een woord compleet is en voert hier een analyse over uit. Deze analyse kan bijvoorbeeld een standaard spellingscontrole zijn. Het kan ook een entiteitscontrole zijn. De webservice kijkt per woord in publieke openbare lijsten, bijvoorbeeld bij de overheid, of het een object is dat bestaat en verzorgt uit deze lijst extra

informatie over het object. Als voorbeeld kan het woord Amsterdam opgestuurd worden. Amsterdam is een object volgens de webservice en de webservice geeft de link naar de webpagina van Amsterdam terug. Aan de hand van de teruggestuurde data kan een applicatie zoals FontoXML, het woord in de tekst aanpassen en voorzien van een link of van één tekstballon als er met de muis over heen bewogen wordt.

Op zinniveau wordt de zin gecontroleerd op bijvoorbeeld grammaticale fouten en leesbaarheid. Webservices kunnen advies geven over de lengte van de zin of een suggestie terugsturen waar niet teveel onnodige woorden in staan.

Op tekstniveau wordt gekeken of de gehele tekst, of delen hiervan, voldoen aan de structuur die door het bedrijf van de klant is vast gesteld. Wordt een subkop bijvoorbeeld altijd opgevolgd door een paragraaf. Afhangend van de bedrijfsregels kan hierop een reactie door de applicatie worden gegeven.

Op inhoudsniveau wordt inhoudelijk de tekst geanalyseerd. Door deze analyse kan de gehele tekst gecontroleerd worden of het bijvoorbeeld voor de juiste leeftijd is geschreven. Ik heb webservices gevonden die ook teksten kunnen controleren op meningen, negativiteit of positiviteit.

Op structuurniveau kan de tekst worden vergeleken met de standaard structuur die het bedrijf hanteert. Door deze controle komt er consistentie in de documenten van het bedrijf.

FontoXML zal delen van het document opsturen. De centrale server zal het opgestuurde stuk kunnen aanpassen naar de fragment levels die gebruikt worden door de webservices. De webservices kunnen op meerdere niveaus tegelijk reactie geven. Deze reactie zal opnieuw omgezet worden door de centrale server zodat FontoXML hier gebruik van kan maken.

De meest voorkomende vorm van communicatie die de webservices gebruiken is REST. REST staat voor Representational State Transfer. Het is een eenvoudige vorm van communicatie en is heel licht om te kunnen versturen. REST communicatie is alleen niet beveiligd in tegenstelling tot SOAP communicatie. SOAP staat voor Simple Object Access Protocol. SOAP is een beveiligde vorm van communicatie tussen cliënt en server. Een SOAP bericht heeft standaard meer data te versturen dan een REST bericht vanwege de beveiliging.

5.5 Conclusie

Uit de deelvragen zijn verschillende conclusies gekomen. Als deze conclusies gecombineerd worden kunnen deze een antwoord geven op de hoofdvraag van het onderzoek.

De gekozen communicatievorm tussen de cliënt en de server is in overleg met de bedrijfsmentor websockets geworden. Websocket is op het moment van schrijven de enige vorm die een full-duplex communicatie ondersteund. Full-duplex communicatie is een belangrijk deel om asynchrone functionaliteiten te kunnen ondersteunen. Om deze communicatievorm te ondersteunen is gekozen voor SignalR. SignalR is ontwikkeld in C# en is ook bedoeld om gebruikt te worden in een C# serveromgeving. Hierdoor wordt de centrale server ook ontwikkeld in C#. De communicatievormen die plaatsvinden tussen de centrale server en de webservices zijn afhankelijk van de webservices. De webservices hebben verschillende vormen en kunnen zowel de AJAX techniek als de COMET techniek gebruiken. Ook hier dient rekening mee gehouden te worden tijdens de ontwikkeling van de centrale server.

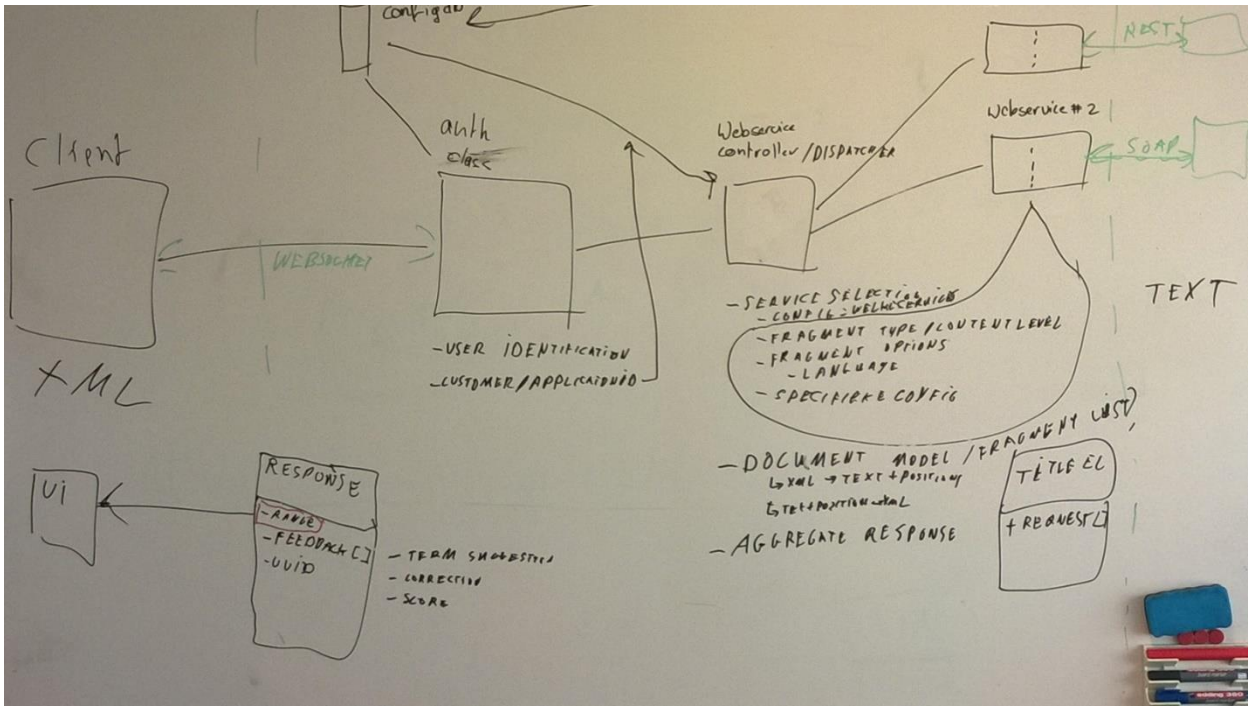
6 Sprint 0: Opstellen initieel ontwerp.

6.1 Doel

Het doel van deze sprint is om een eerste opzet te maken van de structuur van de server. Met de eerste opzet kunnen al snel verschillende problemen naar voren komen die opgelost kunnen worden. Aan het eind van de sprint is er een abstract model ontwikkeld die de centrale serveromgeving weergeeft. Tevens zijn in de eerste sprint meerdere requirements gekomen en deze zijn uitgewerkt in het requirements document.

6.2 Ontwerp

Tijdens de initiële sprint van de centrale server heb ik een model opgesteld om op abstract niveau weer te kunnen geven hoe deze eruit zou komen te zien. De meeste input voor het opstellen van dit model is uit een brainstormsessie met de bedrijfsmentor gehaald.

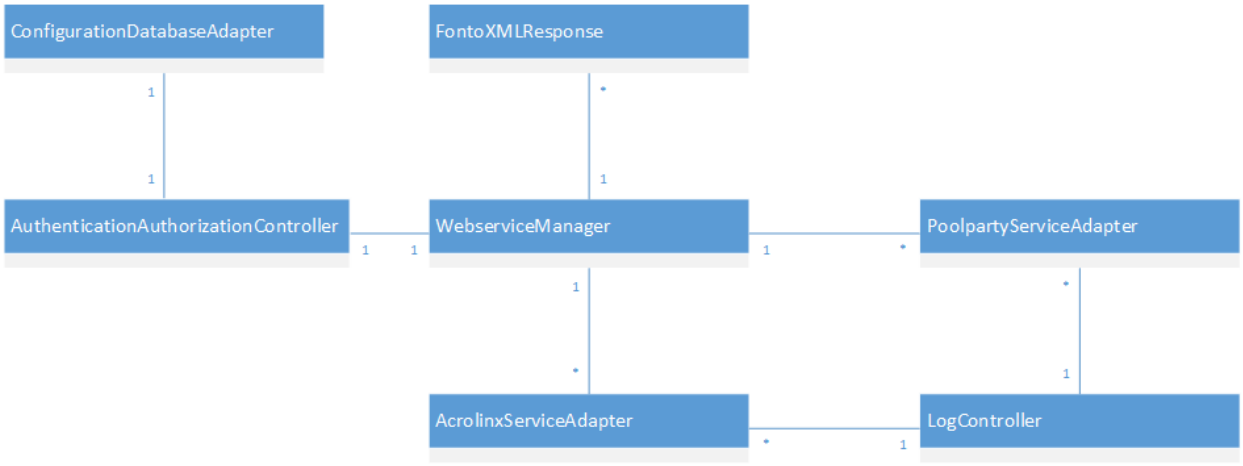


Figuur 8: Eerste brainstormsessie met bedrijfsmentor.

Uit de brainstormsessie zijn de volgende requirements gehaald:

Prioriteit	Requirement
Must	De cliënt kan vanuit de browser via de centrale server communiceren met de webservices.
Could	Het prototype bevat autorisatie functionaliteiten op binnenkomende aanvragen van cliënten.
Must	De centrale server moet de data op de juist gemapte manier terug geven aan de cliënt.
Must	Requests die naar de centrale server worden gestuurd moeten bijgehouden worden.
Should	Het prototype kan met meerdere Web Services tegelijk communiceren.
Must	De centrale server ondersteunt meerdere cliënten tegelijkertijd.
Wont	Het prototype bevat Log functionaliteiten.

Aan de hand van de requirements en de brainstormsessie is het volgende klasse diagram opgesteld. In het klasse diagram is de initiële opzet getoond.



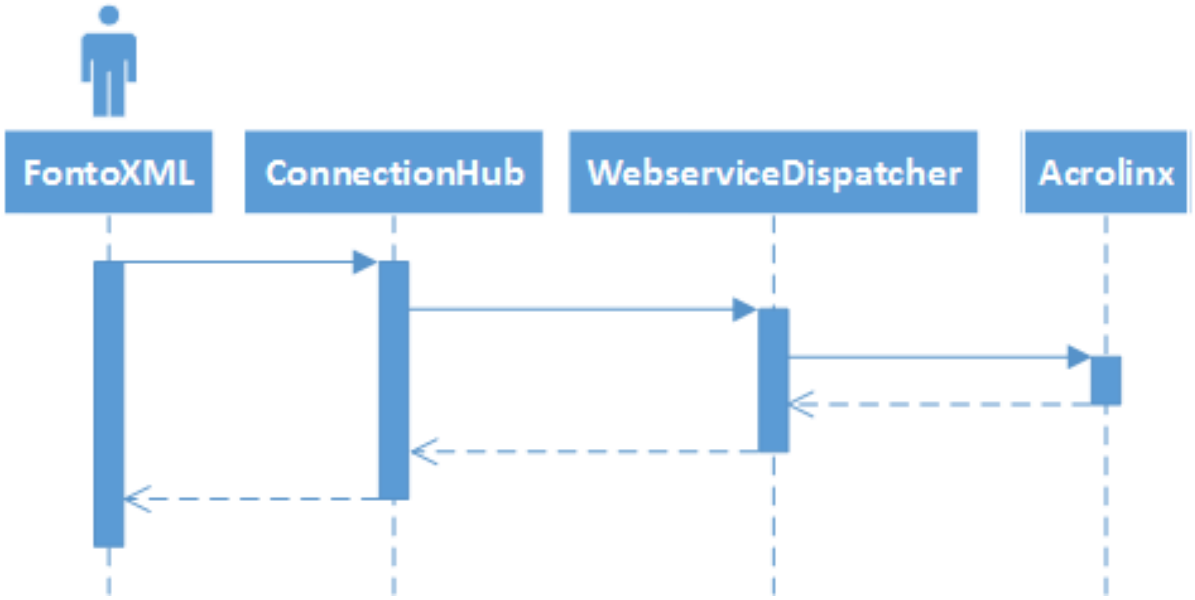
Figuur 9: klasse diagram initiële opzet.

Ieder object in het klasse diagram heeft een korte beschrijving gekregen met de verantwoordelijkheden die het object krijgt. De bedrijfsmentor heeft ook aangegeven waar de prioriteiten liggen voor de centrale server. Uit deze prioritering bleek de LogController, ConfigurationDatabaseAdapter en de AuthenticationAuthorizationController objecten lage prioriteiten te hebben. De prioriteit lag bij de structuur van de centrale server en de functionaliteit hiervan. De bedrijfsmentor geeft aan dat in de eerste sprint vooral de focus zal liggen op de eerste lijn van communicatie. Het toevoegen van meer webservices is voor latere sprints. Voor de eerste versie van het prototype is het van belang dat er contact wordt gelegd met een webservice. Hiervoor heb ik mij gefocust op de Acrolinx webservice omdat deze webservice veel verschillende functionaliteiten heeft waarmee getest kan worden. Vanwege de requirements zijn er nog steeds twee webservices gemodelleerd.

De ConfigurationDatabaseAdapter is een klasse die in verbinding staat met de database waarin data wordt opgeslagen over de gebruikers van FontoXML. Aan de hand van deze data zal een connectie geautoriseerd en geauthentiseerd worden. Dit onderdeel heeft een lage prioriteit.

De AuthenticationAuthorizationController zal alle connecties beheren en accepteren. Dit onderdeel heeft een lage prioriteit.

De WebserviceManager beheert alle webservices en zorgt ervoor dat alle binnen komende requests asynchroon naar de juiste service wordt gestuurd. De manager muteert de data voor de geselecteerde webservice zodat de webservice hier wat mee kan. Hier zal rekening gehouden worden met bijvoorbeeld HTML structuur. Het is mogelijkheid dat in de opgestuurde tekst vanaf de cliënt HTML opmaak staat zoals een bold teken of breaklines. Ik zal kijken hoe deze wordt afgehandeld door de webservice. Als de webservice hier niet mee om kan gaan zal de HTML worden afgevangen. Hierin ligt een gevaar waardoor de mapping van de resultaten terug naar de cliënt lastiger kan worden. Zodra een request terug komt van de webservice zorgt de controller dat de data wordt omgezet zodat het terug naar FontoXML kan en daar verder kan worden afgehandeld. Belangrijk is dat onafhankelijk van de service een response wordt gestuurd naar FontoXML.



Figuur 10: Sequentiediagram initiële opzet.

Het FontoXMLResponse wordt gebruikt om de data die door de webservices geleverd wordt terug te sturen naar FontoXML. Het model is beschreven zodat FontoXML altijd op dezelfde manier data terug krijgt. Dit vermindert het werk dat uitgevoerd zal worden als er een nieuwe webservice bij komt. De bedrijfsmentor geeft aan dat dit een hoge prioriteit is.

De Acrolinx en Poolparty webservice hebben zelf geen hoge prioriteit. De bedrijfsmentor geeft aan dat het niet van belang is hoeveel en welke webservices worden gebruikt. Het is wel van belang dat er met een service wordt gecommuniceerd om de werking van de centrale server te bewijzen. De adapter zelf heeft, onafhankelijk van de gekozen webservice, een hoge prioriteit gekregen. Het object weet op welke manier de service de data wil ontvangen en hoe er verbinding met deze service gemaakt kan worden. In latere sprints zullen deze klassen gebruik maken van een interface. In deze sprint ligt de focus op het maken van een verbinding tussen de cliënt en de centrale server en de centrale server en de webservices.

De LogController heeft ook een lage prioriteit gekregen omdat de focus ligt op de structuur van de centrale server en de werking hiervan. De logger houdt bij welke activiteiten uit worden gevoerd op de centrale server en is niet nodig om de centrale server te laten functioneren. Om deze reden wordt de logger pas interessant als de structuur bewezen is en de centrale server klaar wordt gemaakt voor productie.

Het is van belang dat de centrale server asynchroon kan communiceren met cliënten. Omdat de centrale server pas wat terug kan sturen zodra de webservice klaar is, moet dit niet overige verbindingen gaan blokkeren. Om deze reden is het ook van belang dat de centrale server asynchroon kan communiceren met de webservices. In deze sprint zal dit nog niet terugkomen.

Aan het eind van deze sprint zijn er verschillende startdocumenten opgezet en is er een initiële opzet ontwikkeld waarop de volgende sprint gebaseerd kan worden.

6.3 Bouwen en Testen

In de sprint is niet gebouwd en zijn er geen tests uitgevoerd. Dit omdat deze iteratie is bedoeld als houvast en beginpunt voor de hierop volgende sprints. De eerste ideeën die tijdens de oriëntatiefase zijn bedacht hebben in deze sprint vorm gekregen. En zijn gebruikt om een eerste klasse diagram op te zetten en om een requirement lijst op te kunnen zetten. Beide producten zijn gebruikt om de volgende sprint te kunnen starten.

7 Sprint 1: Opzetten structuur voor de eerste verbinding.

7.1 Doel

Het doel van deze sprint is het abstracte model verder uit te werken en er wordt begonnen met het programma dat deze functionaliteiten gaat krijgen. Dit programma moet bewijzen dat het mogelijk is om van een cliënt via de server de webservices te bereiken en de data die de webservice levert terug te sturen naar de cliënt.

7.2 Ontwerp

Voor de start van de sprint is er een brainstormsessie geweest met de opdrachtgever. In deze sessie is het abstracte model besproken dat ik heb opgezet in sprint 0. Het model is verder uitgewerkt. Opvallende verschillen met het vorige model is de toevoeging van een aantal klasse en de verwijdering van een aantal klasse. De klasse die niet meer bestaan in dit model ten opzichte van het vorige model zijn:

- ConfigurationDatabaseAdapter
- PoolpartyServiceAdapter
- LogClass

De configurationDatabaseAdapter en LogClass komen voort uit de volgende requirements:

Prioriteit	Requirement
Wont	Het prototype bevat Log functionaliteiten.
Could	Het prototype bevat authenticatie functionaliteiten op binnenkomende verbindingen van cliënten.
Could	Het prototype bevat autorisatie functionaliteiten op binnenkomende aanvragen van cliënten.

Uit deze requirements blijkt dat de klassen lage prioriteit hebben en op dit moment niet van belang zijn. Daarom is er voor gekozen de klassen uit het model te halen. De PoolpartyServiceAdapter komt voort uit de volgende requirement:

Prioriteit	Requirement
Should	Het prototype kan met meerdere Web Services tegelijk communiceren.

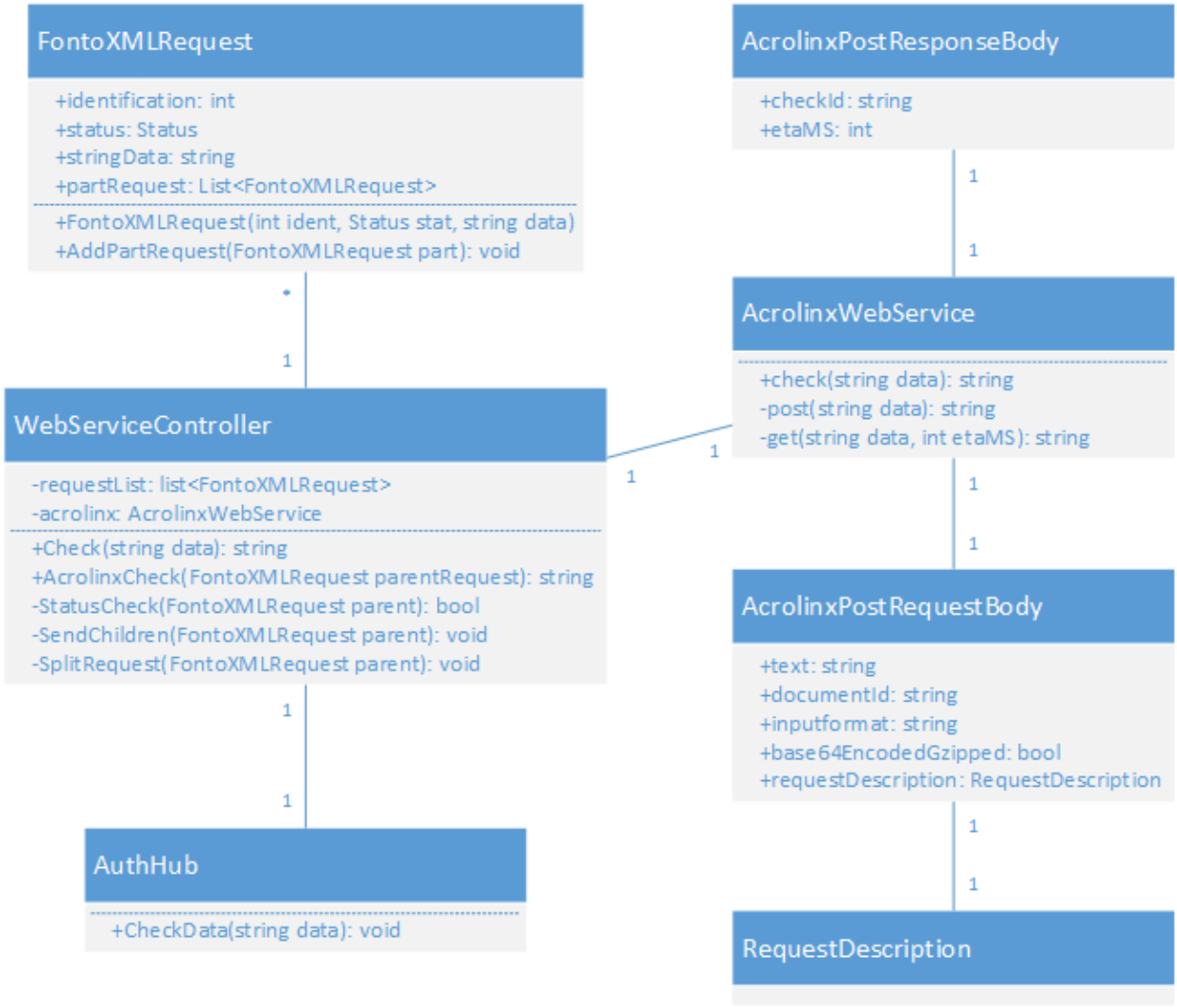
De PoolpartyServiceAdapter is zelf geen belangrijk onderdeel maar wordt gebruikt om de requirement te kunnen bewijzen. De requirement heeft niet de hoogste prioriteit gekregen en is in deze sprint nog niet belangrijk. Om deze reden is de PoolpartServiceAdapter uit het model gehaald. Omdat de requirement nog steeds een kritieke requirement is zal deze op een manier in een latere sprint worden toegevoegd.

De volgende objecten zijn toegevoegd aan het model:

- AcrolinxResponseBody
- AcrolinxPostRequestBody
- RequestDescription

Deze objecten zijn herleid uit JSON die door de webservice gebruikt worden. JSON is een vorm van notatie en is een normale string, die volgens afspraken is opgezet. Om de data van de JSON string bruikbaar te maken worden de strings omgezet in de klassen. Opvallend is dat het

RequestDescription object geen waardes heeft. Dit komt doordat de AcrolinxPostRequestBody deze klasse gebruikt en geen null-waarde hiervoor accepteert.

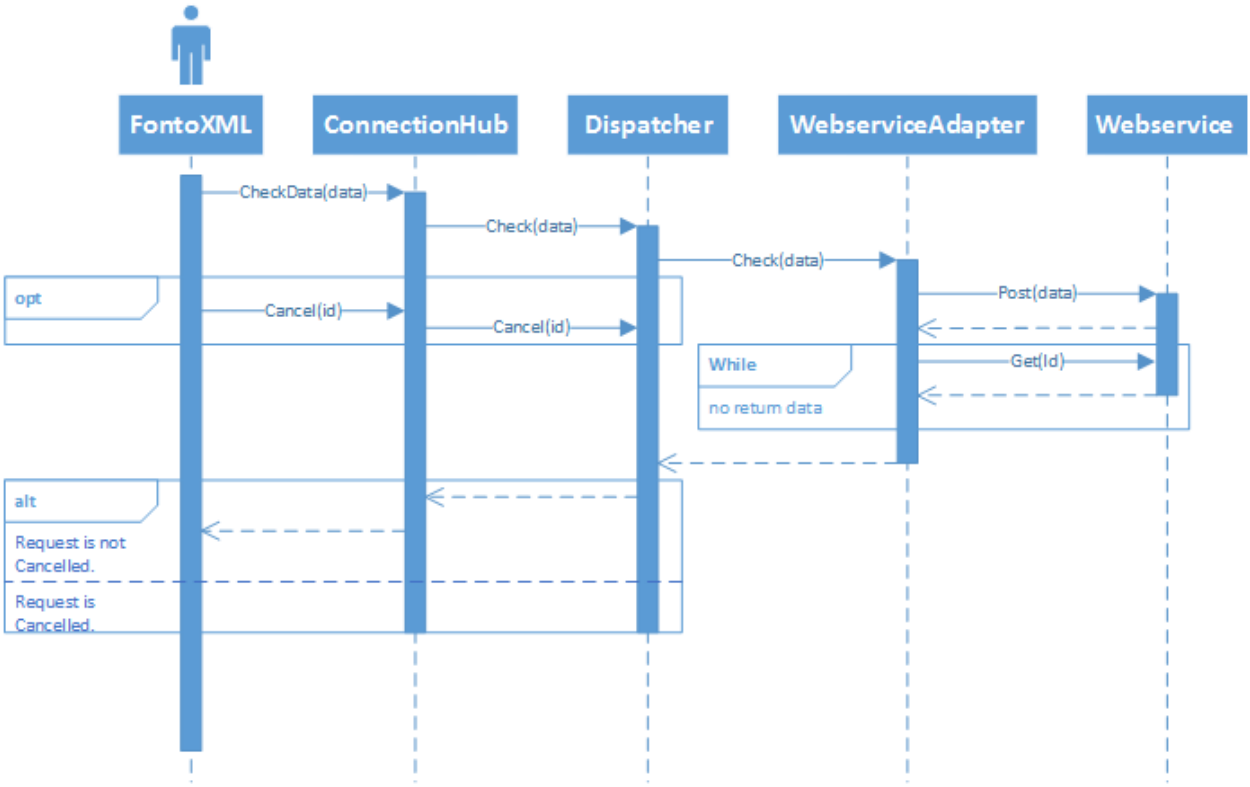


Figuur 11: Klasse diagram centrale server versie 0.2

Tevens is er in deze sprint nagedacht over hoe de volgende requirements uitgewerkt kunnen worden in de applicatie:

Prioriteit	Requirement
Must	De cliënt kan vanuit de browser via de centrale server communiceren met de webservices.
Could	Ontvangen aanvragen moeten kunnen worden gestopt als de aanvragen niet meer geldig zijn door veranderingen bij de cliënt.

Om inzicht te krijgen hoe deze requirements functioneel in de applicatie zullen komen heb ik hiervoor een sequentiediagram gemaakt. Dit sequentiediagram laat zien hoe een cancel request functioneert en welke effecten deze kan hebben op de applicatie. De sequentiediagram ziet er als volgt uit:



Figuur 12: Sequentiediagram voor het sturen en cancelen van requests.

Tijdens deze sprint is er al rekening gehouden met de requirement dat requests van de cliënt te stoppen zijn. De methodes die door de klassen gebruikt moeten worden zijn al geïmplementeerd. In deze sprint zal de applicatie nog niet asynchroon functioneren en om die reden zal die requirement nog niet te testen zijn.

7.3 Bouwen

Bij de start van het ontwikkelen van de applicatie heb ik eerst de structuur die te zien is in het klasse diagram opgezet. Na het opzetten hiervan ben ik deze gaan invullen. Hiervoor heb ik mij eerst

```
var request = Unirest.post("https://us-prod-2-integration-facade.acrolinx-
cloud.com/integration/check")
       .header("content-type", "application/json")
       .header("token", "ec845070-0840-4bda-a5b7-4fc66fc8d3f8")
       .header("authorization", "Basic realm='myRealm'")
       .body(JsonConvert.SerializeObject(AcrolinxPostRequestBody))
       .asJson<string>();
```

Figuur 13: Code voorbeeld Acrolinx Post-methode

gefocust om de verbinding vanaf de adapter naar de webservice op te kunnen zetten. De webservice kan aangeroepen worden door een REST call te doen naar het juiste adres. Deze REST call wordt in de post methode van de adapter uitgevoerd. De code hiervoor is als volgt:

In deze REST-call wordt een request naar Acrolinx gestuurd. In deze request staat het adres, drie headers en een body. Aan de hand van de drie headers kan de Acrolinx webservice de request authentiseren. In de body staat het object in welke de data staat die gecontroleerd moet worden. Met behulp van een openbaar beschikbare library wordt het AcrolinxPostRequestBody omgezet in

een string die voldoet aan JSON. Het JSON object ziet er als volgt uit:

```
{
        "documentId" : "3548fdce-29fb-44ad-8628-9fd48fd4f782",
        "inputFormat" : "HTML",
        "base64EncodedGzipped" : false,
        "requestDescription" : {},
        "text" : "inhoud"
}
```

Figuur 14: JSON Model voor de Acrolinx Post request.

De documentId is een speciale code waarmee ik als gebruiker autorisatie heb om een verbinding te maken met de Acrolinx webservice. De RequestDescription bevat nog een JSON object waarvan onbekend is wat de exacte inhoud is. Omdat het JSON object in een string is beschreven hoeft er geen object gedefinieerd te zijn en kan er door middel van curly brackets in de string aangegeven worden dat er een JSON object staat. Helaas functioneert de webservice niet als de RequestDescription niet gedefinieerd is in de request.

Als de request is verstuurd zal de Acrolinx webservice hierop een response sturen. De response is een JSON string dat omgezet wordt naar een AcrolinxPostResponseBody object. De JSON string ziet er als volgt uit:

```
{
        "checkId" : "stringHash",
        "etaMS" : 100
}
```

Figuur 15: JSON string voor de response van de Acrolinx Post request.

Beide variabele in dit object worden gebruikt tijdens de get methode die een get request stuurt naar de webservice. Aan de hand van het CheckId weet de service om welke tekst het gaat en aan de hand van etaMS wordt aangegeven hoe lang de controle van Acrolinx ongeveer zal duren. Op het moment van schrijven is deze variabele altijd 100 dit omdat Acrolinx zelf de functie nog niet goed heeft geïmplementeerd.

De get methode maakt net zoals de post methode een REST call naar de webservice. Omdat Acrolinx veel tijd nodig heeft om de controle af te ronden wordt deze REST call in een while loop uitgevoerd. Deze loop ziet er als volgt uit:

```
while (response.Body == "")
{
        response = Unirest.get("https://us-prod-2-integration-facade.acrolinx-
cloud.com/integration/checkreport/" + responseBody.checkId)
        .asJson<string>();
}
```

Figuur 16: While loop voor het ophalen van de data van Acrolinx.

De get functie blijft een REST call doen naar Acrolinx totdat er een reactie terug is gekomen. Zodra er een reactie is wordt deze direct teruggestuurd naar de cliënt die het bericht heeft gestuurd. Het object dat terug wordt gestuurd is een JSON string. Deze string wordt op dit moment niet omgezet naar een object en er worden geen mutaties over uitgevoerd. De focus van deze sprint heeft hier niet op gelegen en dit wordt gedaan in een latere sprint.

Doordat de get methode een loop bevat die de data ophaalt heeft de adapter voor Acrolinx COMET functionaliteiten gekregen. De post request is een lang openstaande request.

Het FontoXMLRequest object is gemaakt om requests van de cliënt in bij te kunnen houden en om het terug te sturen naar de cliënt zodra de webservice klaar is met het controleren van de data die gecontroleerd dient te worden. Hierdoor krijgt de cliënt altijd dezelfde vorm als response terug.

Variabele	Beschrijving
identification	Identificatie van de request
status	De huidige status van het object
stringData	De te controleren data
partRequest	Een list waarin kleinere requests bijgehouden kunnen worden.

Functies	Beschrijving
FontoXMLRequest	De constructor van de klasse.
AddPartRequest	Voor het toevoegen van de klasse.

Op dit punt zijn alle opgestelde ontwerpen van deze sprint geïmplementeerd in de centrale server. Door de sprint heen heb ik verschillende testen uitgevoerd deze zullen in het volgende hoofdstuk verder worden uitgeschreven.

7.4 Testen

In sprint 0 en in sprint 1 zijn een aantal requirements opgesteld. Omdat in sprint 0 niet geprogrammeerd is worden deze requirements in sprint 1 getest. Doordat de focus van het project op het bewijzen van de structuur ligt wordt er veel gekeken of de requirements functioneren op de centrale server. Per requirement zal gekeken worden of deze behaald is en hoe dit bewezen is.

Prioriteit	Requirement
Must	De cliënt kan vanuit de browser via de centrale server communiceren met de webservices.

Deze requirement is succesvol geïmplementeerd door het bouwen van de applicatie volgens het klasse diagram. Aan de hand van een HTML pagina die als cliënt functioneert, kan er data opgestuurd worden naar de centrale server. Na enige tijd zal de HTML pagina data terug krijgen die uit een webservice komt en deze data tonen. De data die wordt teruggestuurd wordt nog niet gecontroleerd of deze valide is. Om inzicht te krijgen over hoe deze data eruit ziet heb ik het API omschrijvingen document geschreven. Dit document is terug te vinden in bijlage #2.

Prioriteit	Requirement
Could	Ontvangen aanvragen moeten kunnen worden gestopt als de aanvragen niet meer geldig zijn door veranderingen bij de cliënt.

Deze requirement is gedeeltelijk geïmplementeerd. Omdat de applicatie nog niet asynchroon functioneert, kan de centrale server niet tegelijk een request afhandelen. Om deze reden zal een cancel request pas bij de centrale server afgehandeld worden als voorgaande requests afgehandeld zijn. De volledige werking van deze requirement zal pas getest kunnen worden als de centrale server asynchroon functioneert.

Prioriteit	Requirement
Should	Het prototype kan met meerdere Web Services tegelijk communiceren.

Deze requirement is gedeeltelijk geïmplementeerd. In deze sprint is een webservice toegevoegd. In latere sprints kunnen er meerdere webservices toegevoegd worden. Nieuwe webservices hebben op dit moment nauwelijks invloed op de rest van de centrale server. Om een webservice toe te voegen hoeft er alleen een wijziging plaats te vinden in de WebserviceController.

Prioriteit	Requirement
Could	Het prototype bevat authenticatie functionaliteiten op binnenkomende verbindingen van cliënten.
Could	Het prototype bevat autorisatie functionaliteiten op binnenkomende aanvragen van cliënten.

Deze twee requirements zijn gedeeltelijk geïmplementeerd. De centrale server heeft de AuthHub klasse gekregen. Deze klasse houdt alle verbindingen bij, ontvangt alle requests en verstuurt response naar de juiste cliënt. In deze sprint is geen controle toegevoegd over de binnenkomende verbindingen. Het controleren van de verbindingen staat buiten de scope.

7.5 Reflectie

Terug kijkend op deze sprint heb ik gedaan wat ik aan het begin van de sprint heb afgesproken. Er is niet veel fout gegaan bij het ontwerp en de ontwikkeling van de applicatie. Een minpunt die ik volgende sprint wil verbeteren is dat ik bij het kiezen van de werkzaamheden voor de sprint meer focus op het doel van de sprint. Deze sprint heb ik functionaliteiten gebouwd om requests te kunnen cancelen. Deze functionaliteiten kunnen in deze sprint niet getest worden omdat de hele centrale server nog niet asynchroon functioneert. Deze werkzaamheden had ik pas uit moeten voeren nadat ik de centrale server asynchroon heb laten werken. In deze sprint hebben deze werkzaamheden geen toegevoegde waarde gehad omdat ze niet bruikbaar waren.

8 Sprint 2: Toepassen asynchrone functionaliteiten.

8.1 Doel

Een belangrijk onderdeel van de centrale server is dat deze asynchroon requests kan ontvangen en kan afhandelen. In deze sprint ligt de focus op het ontwikkelen van deze functionaliteiten. Hiervoor zijn verschillende aanpassingen nodig in het klasse diagram.

8.2 Ontwerp

Voor de start van deze sprint is een brainstormsessie gehouden met de opdrachtgever. In deze sprint zijn de wijzigingen van de vorige sprint besproken. Aan de hand van deze brainstormsessie zijn requirements opgesteld en deze zijn verwerkt in het klasse diagram. De wijzigingen die in het nieuwe klasse diagram zitten zijn beperkt. Dit omdat de focus van deze sprint ligt op het invoeren van asynchrone functionaliteiten. Hiervoor zijn weinig aanpassingen nodig in de structuur van de server. In de brainstormsessie is ook de communicatie tussen de centrale server en de cliënt aan sprake gekomen. In de vorige sprint werd data die de webservice terugstuurde niet gemuteerd en direct teruggestuurd naar de cliënt. In deze sprint is ook hier aandacht aanbesteed om een consistente vorm van berichten te maken. De consistente vorm van berichtgeving is van belang om de cliënt zo minmogelijk extra werk uit te laten voeren als er iets is gevonden door de webservices.

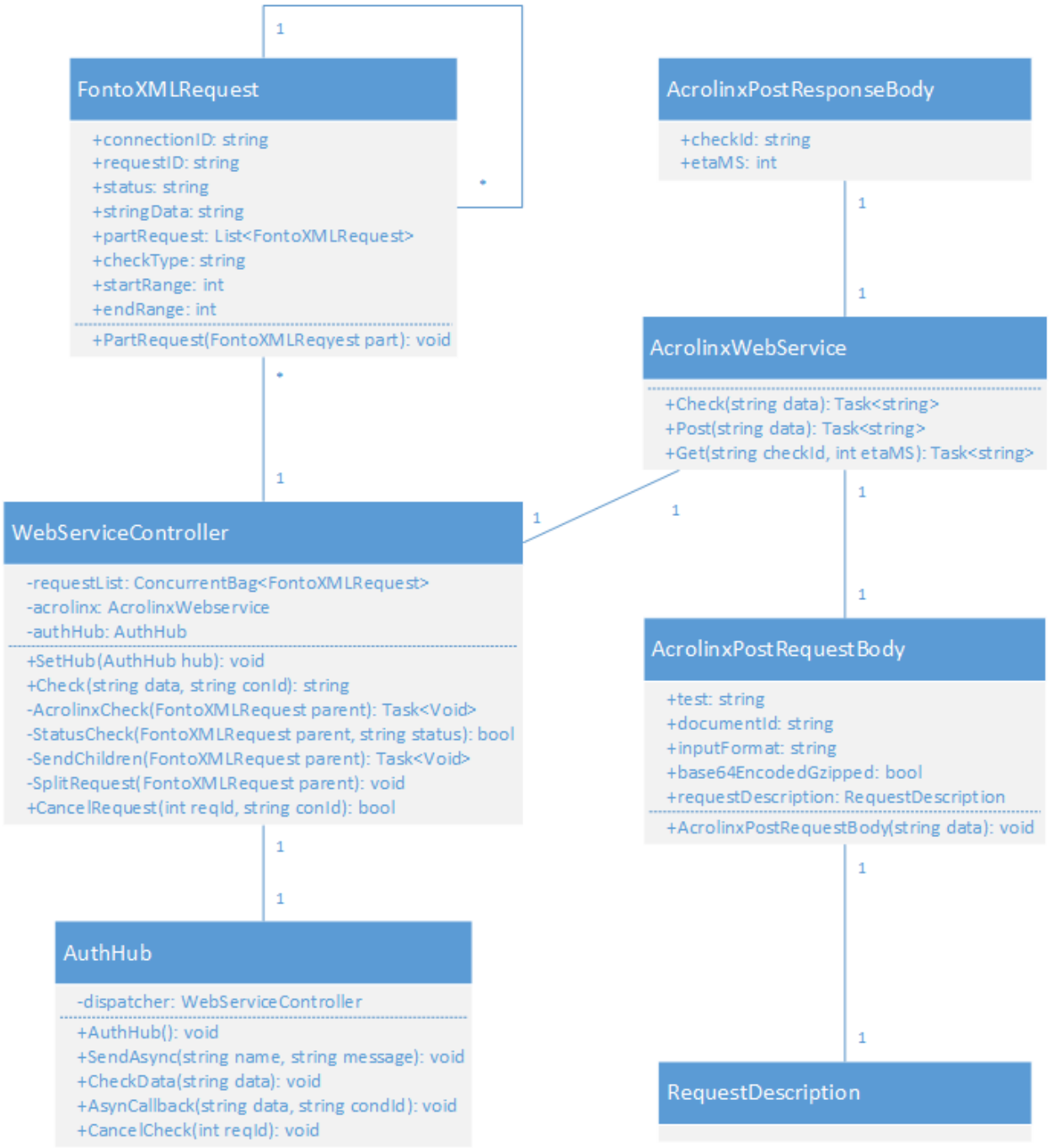
Opvallende wijzigingen in het klasse diagram zijn:

- FontoXMLRequest verwijst naar zichzelf.
- FontoXMLRequest heeft nieuwe variabele gekregen.
- Veel `void` return waardes zijn gewijzigd naar `Task<void>`.

Deze wijzigingen komen voort uit de volgende requirements.

Prioriteit	Requirement
Must	Het prototype kan asynchroon aanvragen van cliënten afhandelen.
Could	De server stuurt de data van de web service altijd op dezelfde manier terug naar de cliënt.
Must	De centrale server kan asynchroon responses terug sturen naar de cliënt.

Deze requirements hebben de focus van deze sprint gekregen. Omdat ik zelf weinig ervaring heb met asynchrone functionaliteiten in C# heb ik de requirements beperkt gehouden. Op deze manier kan ik rekening houden met het mogelijk opnieuw ontwerpen van delen van het systeem omdat op de huidige manier de functionaliteiten niet kunnen werken met asynchrone functionaliteiten.



Figuur 17: Klasse diagram centrale server versie 0.3

8.3 Bouwen

Tijdens het bouwen in deze sprint heb ik mij eerst gefocust op de asynchrone functionaliteiten. C# ondersteunt asynchrone functionaliteiten door middel van het starten van Tasks en threads. Door het toepassen van de asynchrone functionaliteiten zal de cliënt meerdere requests kunnen sturen zonder te wachten op de vorige request. Tevens zal de cliënt in staat zijn een request te kunnen cancelen als deze niet meer nodig wordt geacht. Dit kan voorkomen als data verouderd is. Omdat er verschillende requests tegelijkertijd afgehandeld zullen worden ben ik begonnen met het toepassen van asynchrone functionaliteiten in de AuthHub klasse. Dit leek mij een logisch begin omdat hier de request van de cliënt binnenkomt en de connectie tussen de centrale server en de cliënt asynchroon

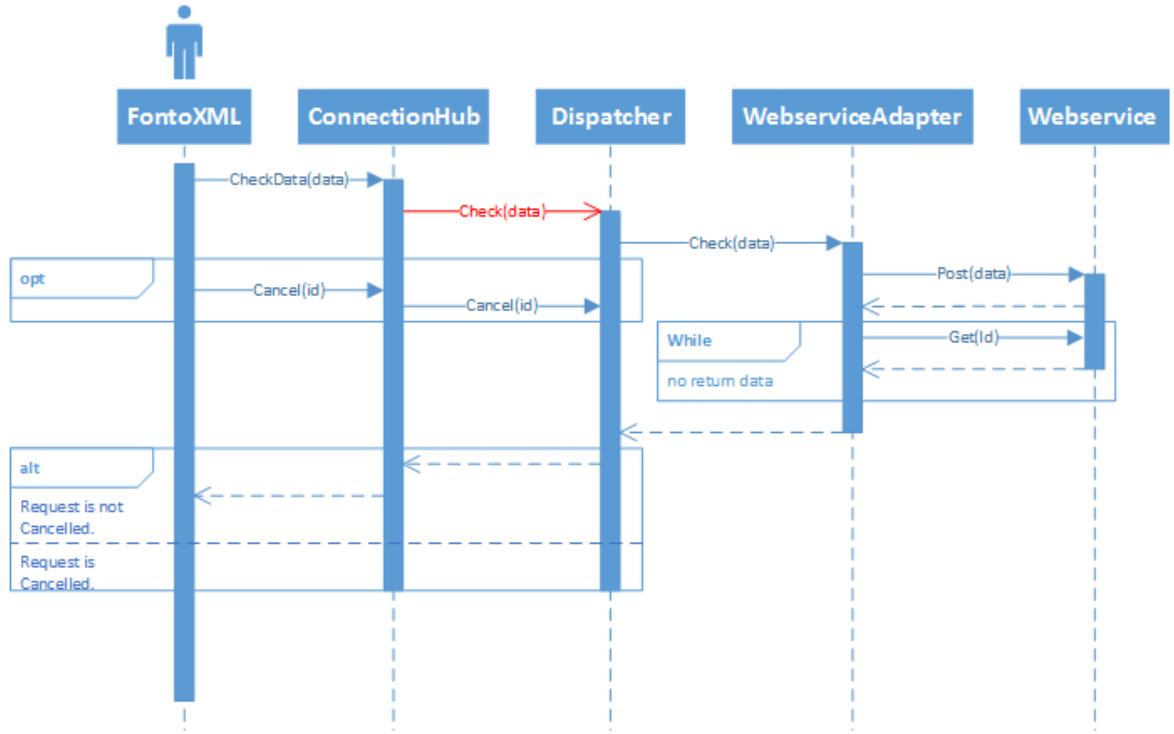
moet functioneren. Hierdoor wordt er voor iedere request dat wordt opgestuurd, een nieuwe thread aangemaakt die de request afhandelt. Zodra de thread klaar is met de request wordt de response teruggestuurd naar de cliënt. Vanwege de asynchrone functionaliteit wacht de Authhub niet tot de request is afgehandeld. Dit gedeelte is geprogrammeerd op basis van de sequentiediagram die in sprint 1 is opgezet.

Om een nieuwe thread te starten wordt via de Task.Factory een thread gebruikt om de functie in uit

```
Task.Factory.StartNew(() => dispatcher.send(message));
```

Figuur 18: Codevoorbeeld voor het starten van een thread.

te voeren. De Task.Factory is een objectpool waarin een aantal threads staan die gebruikt kunnen worden. Dit wordt gedaan omdat het aanmaken van een hele nieuwe thread veel van het systeem vraagt. Daarom worden oudere threads niet direct verwijderd en hergebruikt. Deze functionaliteiten komen uit de System.Threading.Tasks library welke een standaard library is in ASP.NET. Iedere thread die wordt gestart vraagt om een lambda expressie waarin wordt beschreven welke functionaliteit uitgevoerd moet worden in de thread. Zodra de thread is gestart is de rest van de centrale server weer vrij om nieuwe requests te ontvangen en te verwerken. Nu kan de cliënt meerdere requests sturen en de cliënt krijgt deze requests asynchroon terug. In het hoofdstuk Testen van deze sprint wordt uitgelegd hoe ik deze functionaliteit heb getest en waar ik tegen aan ben gelopen. In het volgende sequentiediagram wordt met de rode lijn aangeduid waar de asynchrone functionaliteiten van start gaan.

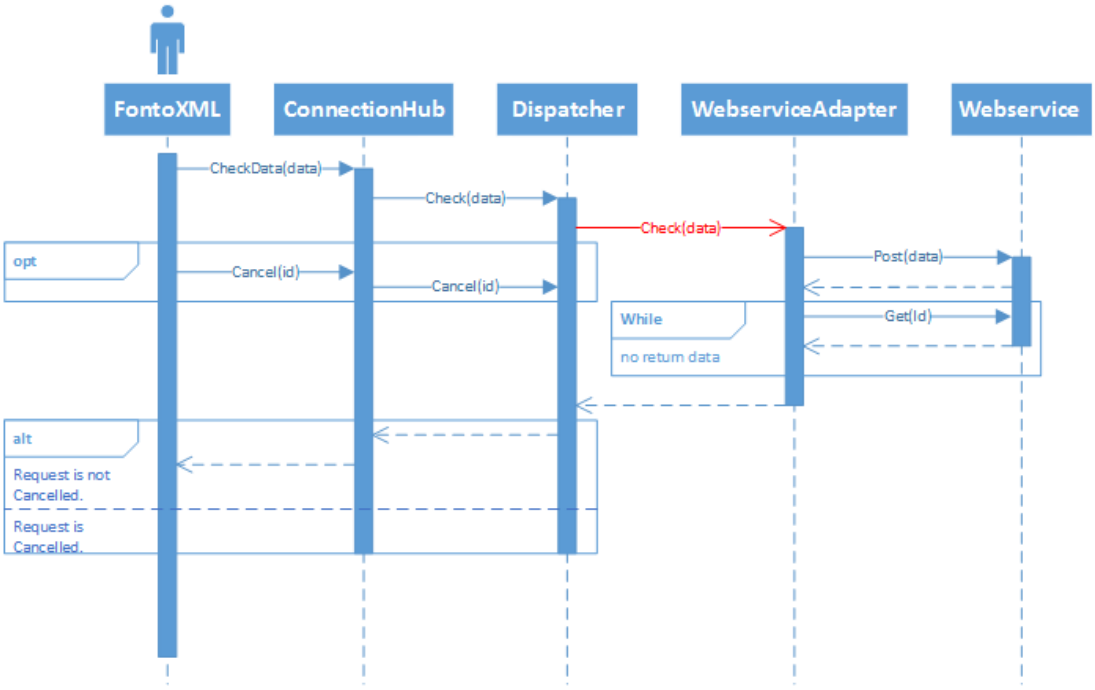


Figuur 19: Sequentiediagram voor het starten van de asynchrone functionaliteiten.

In de vorige sprint heb ik al gewerkt aan de functionaliteiten om een request te kunnen cancelen. Omdat in die sprint de server nog geen asynchrone functionaliteiten had heb ik deze functionaliteiten niet kunnen testen. Nu de asynchrone functionaliteiten zijn ingebouwd heb ik deze

uit kunnen proberen. De WebServiceController heeft een lijst waarin alle binnenkomende requests staan. Deze lijst wordt gebruikt om te controleren of een request klaar is en of deze te cancelen is. Als de request nog in de lijst staat zou de request te cancelen moeten zijn. Bij het draaien van de applicatie waarin ik eerst een request stuur gevolgd door een cancel request bleek de request nooit te cancelen. De opgestuurde request werd door de centrale server door gestuurd naar de webservice en de data van de webservice werd teruggestuurd naar de cliënt waar het getoond werd. Tijdens verder onderzoek en het volgen van de cancel request door de applicatie bleek de request list in de WebServiceController altijd leeg te zijn als er een cancel request werd gestuurd.

Ik had een vermoede dat dit probleem lag bij de asynchrone functionaliteiten. Hiervoor heb ik gekeken naar de start van de asynchrone functionaliteiten en deze verplaatst. In plaats van dat de asynchrone functionaliteiten starten in de AuthHub heb ik deze verplaats naar de WebServiceController. Hierdoor zou de lijst die wordt gemaakt in de WebServiceController onafhankelijk van de threads zijn.



Figuur 20: Sequentiediagram verandering voor asynchrone functionaliteiten.

Na de verandering van de asynchrone functionaliteiten zou de lijst niet meer leeg moeten zijn zodra er een cancel request wordt gestuurd. Dit bleek helaas ook niet de oplossing te zijn en de lijst is ook in deze situatie nog leeg. Omdat mijn kennis beperkt was over de werking van asynchrone functionaliteiten in C# zag ik zelf geen oplossingen meer. Ik heb hulp gevraagd aan een collega die meer kennis heeft van de asynchrone functionaliteiten. Hij heeft mij informatie kunnen gegeven over hoe objecten thread-safe gemaakt kunnen worden. Thread-safe houdt in dat objecten onafhankelijk van threads gemuteerd kunnen worden. In het geval van de lijst met requests houdt dit in dat iedere thread gebruik maakt van dezelfde lijst waarin dezelfde waardes staan. Als er iets in deze lijst veranderd zal deze verandering in alle threads gebeuren. In het geval van de applicatie gaat het om lijst met requests die thread-safe gemaakt moet worden. Deze lijst maakt nu gebruik van een List object waarin alles wordt opgeslagen. Als in plaats van deze List een ConcurrentBag wordt gebruikt is de list thread-safe. Een ConcurrentBag heeft dezelfde functionaliteiten als een List object.

Het verschil is dat de ConcurrentBag wel thread-safe is en de List niet. Nu de ConcurrentBag geïmplementeerd is kan de cancel request bij een gevulde lijst met openstaande requests. Dankzij de ConcurrentBag kan de cancel request bij de openstaande requests en kan deze requests stopzetten. Het stopzetten van een request gebeurt door de status variabele in de FontoXMLRequest te veranderen. De applicatie controleert wat de status is van een request en zal hierop reageren. Als dankzij de status wordt achterhaald dat een request gecanceld is zal de centrale server wachten tot alle webservices klaar zijn. Als alle webservices voor de request klaar zijn wordt de request verwijderd en zal de cliënt geen data ontvangen over de request. De centrale server heeft geen controle over de webservices en zal de webservices niet kunnen stoppen als hier data naar gestuurd is. De centrale server moet wachten tot er een response van de webservice komt. Als de request gecanceld is wordt de response van de webservice niet afgehandeld. Zodra alle webservices een response hebben gestuurd kan de request verwijderd worden uit de lijst.

Op dit moment zijn alle requirements die voor deze sprint zijn opgesteld geïmplementeerd en kunnen deze getest worden.

8.4 Testen

Aan het eind van deze sprint heb ik samen met de bedrijfsmentor een codereview uitgevoerd. Tijdens de codereview heb ik alle code aan de bedrijfsmentor laten zien. De bedrijfsmentor heeft hier feedback opgegeven. Uit de review werd duidelijk dat de manier waarop ik asynchroon data terugstuurde naar de cliënt niet optimaal was. Om de data terug te sturen is het beter om gebruik te maken van actions. Dit wordt in de volgende sprint opgepakt. Tevens bleek uit de codereview dat voor de webservices een interface ontwikkeld moet worden. Aan de hand van de interface moeten nieuwe webservices eenvoudiger toe te voegen zijn. Beide aanpassingen worden in de volgende sprint uitgewerkt.

De data die de cliënt ontvangt is wel tijdens de codereview goedgekeurd. Omdat de mapping van de data nog buiten de scope van deze sprint staat is deze goedgekeurd op basis van wat de webservice terugstuurt. Door de goedkeuring hiervan is ook bewezen dat de verbinding met de webservices werkt.

8.5 Reflectie

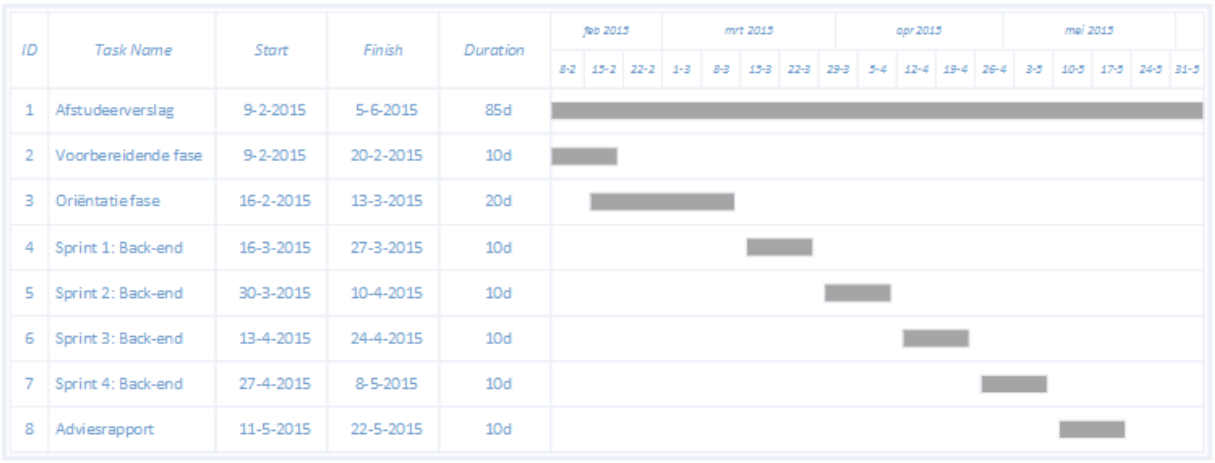
In de vorige sprint vond ik dat de focus van de sprint niet goed was. In deze sprint heb ik geprobeerd dit te verbeteren door minder requirements tegelijkertijd op te pakken. Door minder requirements op te pakken werd direct een risico van deze sprint verminderd. Dit risico was namelijk dat ik geen kennis had over asynchroon programmeren in C#. Door de ontbrekende kennis was ik bang de sprint niet in de daar voor geplande tijdsperiode haalbaar was. Uit de bovenstaande beschrijving blijkt niet hoeveel tijd ik heb gestoken in het leren van asynchroon programmeren. Ik wil hier graag nog duidelijk maken dat tijdens deze sprint veel tijd is gestoken in het opdoen van nieuwe kennis door middel van het lezen van artikelen op het microsoft developer network.

9 Sprint 3: Reviewen server structuur.

9.1 Planning

Een derde sprint bleek nodig te zijn om het prototype verder te ontwikkelen. De derde sprint houdt in dat ik niet meer volgens de originele planning aan het werk ben. In de originele planning zou de backend nu afgerond moeten zijn en zou ik beginnen aan de ontwikkeling van de front-end. Door gesprekken met de bedrijfsmentor is duidelijk geworden dat we de front-end buiten de scope van de opdracht gaan plaatsen. Dit omdat de front-end meer gefocust is op de usability van het originele systeem en niet op de functionaliteiten van de centrale server. Tevens zijn er veel nieuwe inzichten en requirements naar voren gekomen voor de backend. Hierdoor is een extra sprint noodzakelijk. Het niet meer te hoeven ontwikkelen van het front-end gedeelte heeft geen verdere invloed op het project of mijn te behalen competenties. Door de verandering in de planning ziet deze er nu als volgt uit:

ID	Task Name	Start	Finish	Duration
1	Afstudeerverslag	9-2-2015	5-6-2015	85d
2	Voorbereidende fase	9-2-2015	20-2-2015	10d
3	Oriëntatiefase	16-2-2015	13-3-2015	20d
4	Sprint 1: Back-end	16-3-2015	27-3-2015	10d
5	Sprint 2: Back-end	30-3-2015	10-4-2015	10d
6	Sprint 3: Back-end	13-4-2015	24-4-2015	10d
7	Sprint 4: Back-end	27-4-2015	8-5-2015	10d
8	Adviesrapport	11-5-2015	22-5-2015	10d



Figuur 21: Gantt -chart vernieuwde planning

De vervolgactiviteiten die uitgevoerd zullen worden voor de backend zullen zich focussen op het verbeteren van de structuur van de centrale server en de ontwikkeling van de juiste mapping van de data.

9.2 Doel

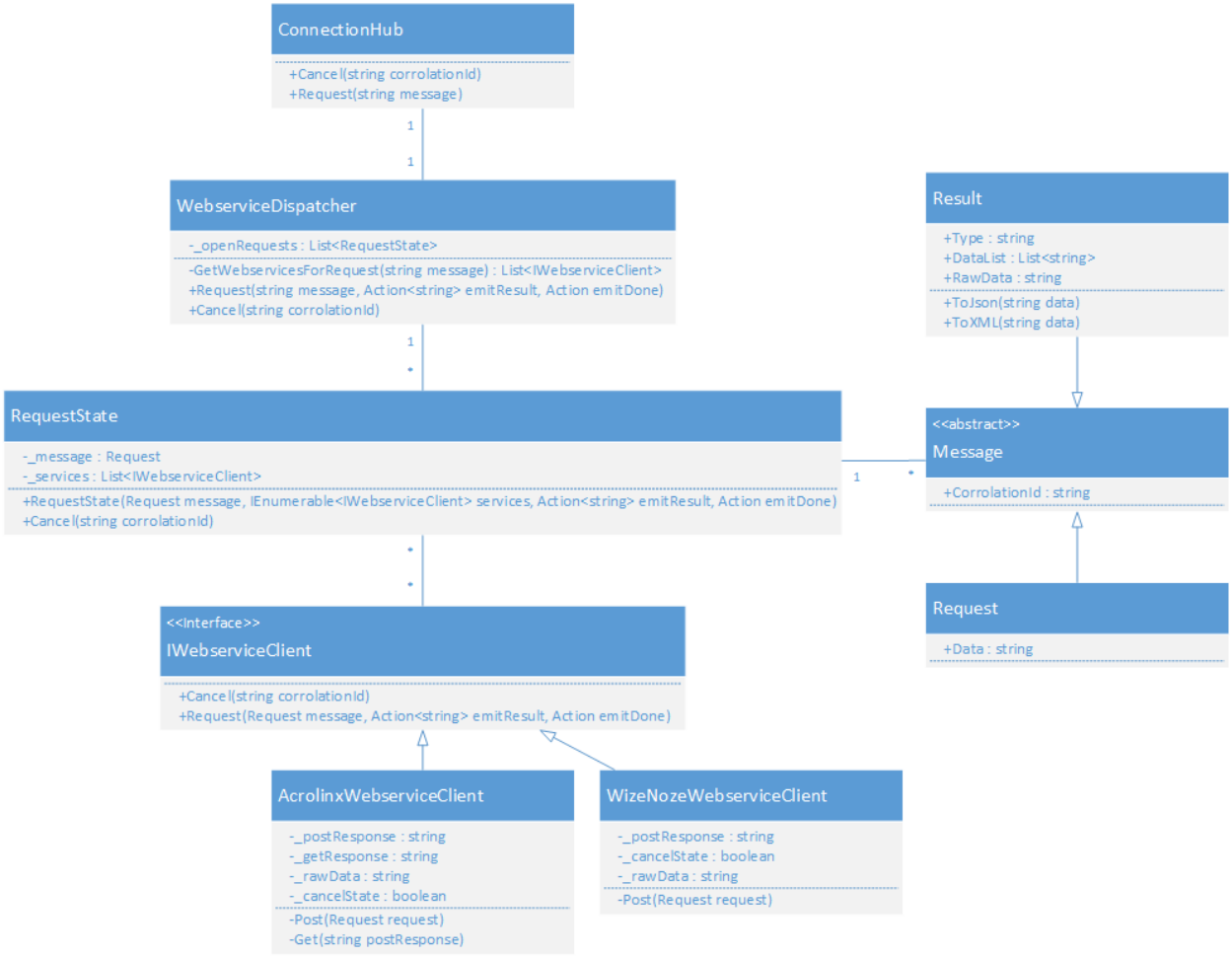
In de derde sprint is opnieuw veel aandacht besteed aan de structuur van de centrale server. Uit de codereview die vorige sprint is uitgevoerd bleek dat de structuur van de centrale server nog veel verbeterpunten heeft. In deze sprint worden zoveel mogelijk van deze punten aangepakt en verwerkt in de server.

9.3 Ontwerp

Tijdens de brainstormsessie van de derde sprint is de structuur van de centrale server veranderd door het invoeren van verschillende interfaces en abstracte klassen. Tevens zijn vele namen veranderd om te kunnen verduidelijken wat de functie is van de klasse. Het nieuwe model ziet er hierdoor anders uit dan de voorgaande modellen. In deze sprint is ook een nieuwe webservice toegevoegd. Hierdoor is de centrale server in staat om met verschillende services tegelijkertijd te kunnen communiceren. De centrale server zal net zoals bij de vorige sprint asynchroon de resultaten van de webservices terug geven aan de cliënt. Verder heeft de focus van deze sprint gelegen op het

terugsturen van de juiste data naar de cliënt. Hiervoor wordt gebruik gemaakt van het Result object. Dit object heeft de variabelen waarmee de cliënt de data op de juiste manier kan tonen.

De Result klasse is vergelijkbaar met het oude FontoXMLRequest object. Ze hebben dezelfde functionaliteiten maar zijn verschillend in variabelen. De connectionID komt niet meer terug in het nieuwe model. Door het gebruik van Actions in de centrale server hoeft deze niet opgeslagen te worden. De requestID heeft nu de naam correlationID gekregen. Status is verwijderd uit de request klasse. Voorheen werd deze gebruikt om de status van het request object te beheren. Door middel van het gebruik van Actions kan de request op een andere manier worden gecanceld. Tevens wordt er geen gebruik meer gemaakt van een ConcurrentBag om de requests in op te slaan. Om een variabele te synchroniseren zodat in alle threads dezelfde lijst wordt gebruikt kan deze lijst static gemaakt worden.



Figuur 22: Klasse Diagram centrale server versie 0.4

9.4 Bouwen

In het codevoorbeeld is te zien hoe Actions worden gebruikt om verschillende berichten terug te kunnen sturen naar de cliënt. Het codevoorbeeld staat in de ConnectionHub. De hub ontvangt een request en stuurt deze request door naar de WebserviceDispatcher. De WebserviceDispatcher krijgt naast de request zelf ook twee actions meegestuurd. De WebserviceDispatcher kan deze actions gebruiken om results terug te sturen en om aan te geven wanneer een request afgerond is. Omdat de actions worden gebruikt om data terug te kunnen sturen hebben de functies geen return value.

```
WebserviceDispatcher.Request(message, result =>
{
        // Returns any result back to the caller.
        Clients.Caller.result(result);
}, () =>
{
        // Tell the cliënt no more results will come.
        Client.Caller.done(correlationId);
});
```

Figuur 23: Codevoorbeeld hoe Actions worden gebruikt.

In deze sprint is een interface toegevoegd die door de webserviceadapters gebruikt zullen worden. Er is gekozen voor een interface omdat alle webserviceadapters altijd de volgende methodes moeten bevatten. De implementatie van deze methodes is voor iedere webservice verschillend. De interface ziet er als volgt uit:

```
public interface IWebserviceClient
{
        void Cancel();
        void Request(Request message, Action<string> emitResult, Action emitDone);
}
```

Figuur 24: Codevoorbeeld voor IWebserviceClient

Met de cancel functie wordt de request die door de webservice wordt uitgevoerd stopgezet. Als de request al is opgestuurd naar de webservice wordt door de adapter bij het terugkomen van de data gecontroleerd of de request nog niet gecanceld is. Als de request gecanceld is wordt direct de Action emitDone aangeroepen.

De vorm van berichtgeving naar de cliënt is veranderd en bestaat uit een abstracte klasse die door twee klasse worden geërfd. Het message object is de abstracte klasse en bevat een variabele. De corrolationId wordt gebruikt om een request te kunnen identificeren. Deze code zal door de cliënt en de centrale server bijgehouden worden. Hierdoor kan de cliënt een request cancelen als dit nodig is. De cliënt zal de code weer verwijderen zodra de server aangeeft dat de request compleet is afgerond.

Als er een request binnenkomt van de cliënt zal de server een Request object aanmaken. Een request object erft van het message object. Het request object heeft de variabele data. Deze variabele wordt gebruikt om de data op te slaan die door de cliënt wordt opgestuurd ter controle. Deze data wordt later door de webserviceadapter gemuteerd om opgestuurd te kunnen worden naar de webservice. Als de webservice data terugstuurt naar de webserviceadapter wordt er een Result object aangemaakt. Het result object erft ook van het message object. Het result object heeft drie variabele. Type, DataList en rawData. Aan de hand van Type wordt duidelijk gemaakt of de result over een spellingsfout gaat, over een grammaticafout gaat, of over een andere controle die

door de webservice uitgevoerd kan worden. In de DataList komt data vanuit de webservice te staan. In deze sprint heb ik geprobeerd om deze data uit te lezen en op een gestructureerde manier terug te sturen. In rawData komt de data te staan die direct vanuit de webservice komt. Hierover worden geen mutaties uitgevoerd.

Het result object heeft ook twee functies. Deze functies zijn gebouwd om te kijken op welke manieren de data het best teruggestuurd kan worden naar de cliënt. De ToJson functie kan een string omzetten naar de notatie die voldoet aan JSON. JSON is de lichtste vorm van versturen van data over het internet en daarom een goeie mogelijkheid om gebruik van te maken voor het terugsturen van de data. De ToXML functie kan een string omzetten naar XML structuur. Omdat de cliënt zelf gebruik maakt van XML structuren is dit ook een optie die ik wil overwegen. In deze sprint zijn de functionaliteiten ingebouwd om alvast inzicht op te kunnen doen voor de volgende sprint.

9.5 Testen

De veranderingen die tijdens deze sprint hebben plaatsgevonden hebben voornamelijk invloed gehad op de manier waarop data van de cliënt uiteindelijk bij de webserviceadapter komt en hoe deze data weer wordt teruggestuurd. De veranderingen hebben geen invloed gehad op de manier waarop de webserviceadapters met de webservices communiceren.

In deze sprint heb ik veel code opnieuw bekeken en opnieuw geschreven. Om deze reden moet de code uit de sprint aan alle eisen voldoen waaraan de vorige sprints ook aan hebben voldaan. De tests hebben zich gefocust op of de data op de juiste manier bij de webservices komt en of de adapter de data van de webservice weer op de juiste manier terugstuurt naar de cliënt. Hier hebben de grootste veranderingen gezeten.

Na de uitkomsten van deze sprint te vergelijken met die van de voorgaande sprint zit er alleen een verandering in de data die terugkomt bij de cliënt. Deze verandering is verwacht omdat er veranderingen zijn geweest in de berichten die naar de cliënt worden gestuurd. Voorheen werd de data vanuit de webservice direct teruggestuurd naar de cliënt. In deze sprint is dit aangepast. De cliënt zal nu meer informatie ontvangen dan alleen data uit de webservice. De centrale server zal nu het corrolationId en de type terugsturen naar de cliënt met de data van de webservice.

9.6 Reflectie

Deze sprint bestond vooral uit het opnieuw bekijken van de code die ik in vorige sprint heb geschreven. Tijdens de vorige sprints is meer kennis opgedaan over hoe asynchrone functionaliteiten geïmplementeerd kunnen worden. Hierdoor bleek een derde sprint voor de backend nodig te zijn om deze bevindingen te implementeren. De planning veranderd door deze sprint. Ik vind dat ik goed heb gekeken of dit mogelijk was door de competenties van de sprint te vergelijken. Hieruit is gebleken dat er geen veranderingen zijn gekomen in mijn competenties. In overleg met de bedrijfsmentor is besloten dat de werkzaamheden die in de originele planning van sprint drie buiten de scope worden geplaatst en dat er verder gewerkt zal worden aan wat er in de vorige sprints is ontwikkeld.

Als verbeterpunt had ik mij beter in kunnen lezen over hoe asynchrone functionaliteiten door C# worden ondersteund. Tijdens het vooronderzoek werd al duidelijk dat ik de centrale server in C# zal gaan programmeren. Op dit punt had ik meer moeten inlezen over hoe C# met asynchrone functionaliteiten omgaat en hoe deze worden ondersteund. Hierdoor had ik mogelijk al in aanraking gekomen met Actions en deze vanaf het begin van de ontwikkeling kunnen gebruiken.

10 Sprint 4: Communicatie optimalisatie.

10.1 Doel

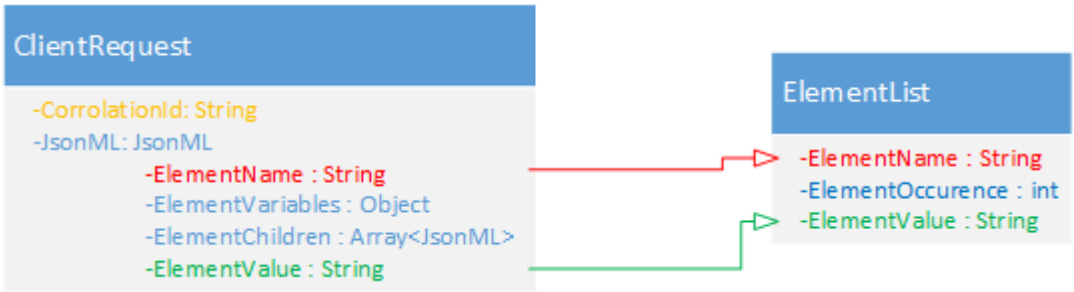
In de vorige sprints heeft de centrale server de data vanuit de webservices teruggestuurd naar de cliënt zonder de data te muteren. Voor de cliënt is deze data onbruikbaar. Deze sprint heeft de focus om de data te transformeren zodat deze wel bruikbaar is door de cliënt.

10.2 Ontwerp en Bouwen

In de vierde sprint is de focus gelegd op de communicatievorm tussen de cliënt en de centrale server. De centrale server zal de functionaliteit hebben om gecontroleerde data weer terug te geven aan de cliënt. De cliënt zal uit deze data kunnen lezen om welk stuk van de tekst het gaat en wat de data daarover wil zeggen. De originele tekst die wordt opgestuurd naar de centrale server wordt meerdere keren gemuteerd en kan daarom niet direct terug op de originele tekst worden geplaatst. De cliënt stuurt een JsonML object op naar de centrale server. JsonML is een combinatie van JSON en XML structuur. Hiermee kan de cliënt XML structuren opsturen naar de centrale server. In deze structuur staat de data opgeslagen die gecontroleerd zal worden. De centrale server zal de JsonML terug zetten naar XML. Dit omdat FontoXML gebruik maakt van de XML structuur om data te beheren. Aan de hand van de XML structuur kan de data opgestuurd worden naar de webservice en kan de data van de webservice weer terug gemapt worden op de XML structuur. Als de data gecontroleerd is en de juiste mapping is opgehaald uit de XML structuur kan alle data teruggestuurd worden naar de cliënt. Om de data terug te sturen worden verschillende result objecten gebruikt afhangend van de data die terugkomt. Omdat de webservices alleen JSON objecten kunnen ontvangen met daarin alleen tekst wordt de XML structuur vergeten. De centrale server zal er voor zorgen dat de response van de webservices weer op de juiste manier wordt gekoppeld aan het origineel. Om duidelijkheid te scheppen over alle mutaties heb ik een mapping model gemaakt. Door de grote van het model heb ik deze in stukken opgedeeld om duidelijk te kunnen beschrijven wat er met de data gebeurt.

10.2.1 Stap 1: Request naar elementList.

Ontwerp



Figuur 25: Request naar list mapping.

Zodra een request wordt opgestuurd vanaf de cliënt ziet de data eruit zoals in het ClientRequest object beschreven. Dit object wordt vanaf de ConnectionHub via de WebServiceDispatcher doorgestuurd naar de RequestState. In de RequestState zal de data omgezet worden naar de list. In de ClientRequest zijn de corrolationId, de elementName en de elementValue de belangrijkste gegevens die meegestuurd worden. ElementVariables zijn variabele van een element. Bijvoorbeeld

de width van de table element. Deze data is niet interessant voor de webservices en hoeft om deze reden ook niet verder gebruikt te worden. De elementChildren is een array met elementen die zich binnen de parent element bevinden. Door deze array wordt gelooped en wordt ook de ElementName en ElementValue van ieder element opgehaald. De corrolationId is in dit figuur niet gemapped. Deze wordt naar een ander object gemapped dat in dit figuur niet getoond is.

De data die uit de ClientRequest gehaald wordt komt in de elementList te staan. De data komt via een Tuple in de list te staan. Een regel in de list ziet er als volgt uit:

```
{(p, 14, "Noord-Holland")}
```

Figuur 26: Entry ElementList.

De combinatie van deze data geeft de volgende informatie: *"Noord-Holland is de data die gevonden kan worden op de 15de keer dat er een p element voorkomt"*. De list maakt gebruik van een 0 pointer index. Hierom zal de list beginnen met tellen bij 0 en niet bij 1. Om deze reden is de uitgeschreven tekst 1 waarde hoger dan wat er te zien is in het object.

Er zijn een paar uitzondering die niet in de list terugkomen. Voorbeelden hiervan zijn bijvoorbeeld elementen die een lege tekst bevatten. Of elementen die duidelijk maken dat er een nieuwe regel moet komen.

Bouwen

Om de JsonML uit te kunnen lezen is een functie geschreven in de RequestState klasse. Deze functie zorgt ervoor dat de juiste data uit de JsonML string wordt gehaald en in de elementList wordt geplaatst.

```
Private static void JsonMLtoMappingList(JsonReader reader,
ICollection<Tuple<string, int, string>> list)
{
       // More code goes here.
};
```

Figuur 27: Methode aanroep van de JsonML naar de list methode.

De methode is een recursieve functie. Om deze reden worden zowel de reader als de list buiten de scope van de methode aangemaakt. In de JsonReader staat de JsonML string die uitgelezen moet worden voor deze request. De list is een lijst die Tuples opneemt waarin drie waardes staan. Deze tuple is te zien in figuur 26. Zodra de functie aangeroepen wordt zal deze beginnen met het lezen van de JsonML string.

```
While (reader.TokenType != JsonToken.StartArray)
       Reader.Read();
Var elementName = reader.ReadAsString();
```

Figuur 28: Codevoorbeeld waarmee gelezen wordt tot de start van de array wordt gevonden.

Zolang er geen start gevonden kan worden van een Array zal doorgelezen worden. Zodra er wel een start van een array gevonden wordt betekent dat de naam van het element uitgelezen kan worden. De uitgelezen element name wordt later gebruikt om de tuple te vullen die in de list komt te staan. Zolang de element een tekst waarde heeft.

Nu de element name bekend is wordt er een loop gestart die de JsonML string in de reader verder probeert te lezen.

```
While (reader.Read())
{
        If (reader.TokenType == JsonToken.EndArray)
                Break;
        If (reader.TokenType == JsonToken.StartObject)
                Continue;
        If (reader.TokenType == JsonToken.StartArray)
                JsonMLtoMappingList(reader, list);
        If (reader.TokenType == JsonToken.String)
                Var elementValue = reader.Value;
}
```

Figuur 29: Codevoorbeeld waarmee children en values worden gevonden.

Na de element name zijn er vier mogelijkheden die de reader daarna kan lezen. Twee van deze waardes leveren extra informatie op. Alle vier de mogelijk waardes zal ik kort uitleggen.

Als de eerst volgende waarde het einde van een array is houdt dat in dat er geen nieuwe informatie meer volgt over het huidige element dat gelezen wordt. De loop zal dan stoppen en alle gevonden data over dat element in de list opslaan.

De volgende mogelijkheid is dat de reader de start van een object vindt. Een element kan een object hebben. In dit object worden variabele beschreven zoals de font size van de tekst in het element. Deze gegevens zijn niet interessant en daarom zal direct de volgende waarde geprobeerd gelezen te worden.

De derde mogelijkheid is een nieuwe start van een array. Als er in een element een nieuwe start van een array wordt gevonden betekent dat dit element een child element heeft. Van dit child element moeten ook alle waardes gelezen worden. Hier zal de methode zichzelf opnieuw aanroepen om de waardes van dit child element uit te lezen. Het kan voorkomen een element meerdere child elementen heeft. Deze worden ook uitgelezen en opgeslagen.

Als laatste optie kan de reader een string waarde lezen. Deze string waarde is de tekstuele waarde die gecontroleerd kan worden door de webservices. Deze waarde wordt tijdelijk opgeslagen in de string en later gebruikt om het object op te slaan.

Op dit punt zijn alle mogelijke waardes gelezen die een element kan hebben. En kan deze toegevoegd worden aan de lijst. Dit gebeurd als volgt:

```
var occurence = list.Count(element => element.Item1 == elementName);
list.Add(new Tuple<string, int, string>(elementName, occurence, elementValue));
```

Figuur 30: Toevoegen van alle gevonden data in de list.

De occurence geeft aan hoeveel keer er een element met dezelfde naam al voorkomt in de lijst. Deze waarde wordt later belangrijker voor de mapping en zal dan ook verder uitgelegd worden. De element naam en de element waarde worden in de tuple gezet en de tuple wordt aan de lijst toegevoegd.

Als de reader alles heeft gelezen is de lijst compleet. En kan de lijst omgezet worden zodat de data opgestuurd kan worden naar de webservice.

10.2.2 Stap 2: ElementList naar WebserviceRequest.

Ontwerp

Als de volledige JsonML is omgezet naar de list kan de volgende mutatie plaatsvinden. Deze mutatie zal de data klaar maken voor de webservices.

Figuur 31: List wordt omgezet naar webservice data.

Alle data in de elementList is belangrijk maar niet alle data is nuttig voor de webserviceadapter. De ElementName en ElementOccurence worden later gebruikt in een ander object. Om deze reden wordt dit hier niet gemapped.

Deze data zal in de objecten die de IWebserviceClient implementeert worden omgezet. Omdat webservices op verschillende manieren data aangeleverd willen krijgen kan de implementatie van deze functie verschillend zijn per webservice. In het geval van Acrolinx en WizeNoze willen deze beide alleen een string ontvangen. Alle elementValues in de elementList worden samen gevormd tot een string in de WebserviceRequest. Deze mutatie vindt plaats in de adapter voor de webservices. Per webservice wordt er een string gemaakt. Deze string wordt opgestuurd naar de webservice vanuit de webserviceAdapters. Ik heb geen informatie over de werking van de webservices en ik weet daarom alleen wat deze willen ontvangen en wat deze terug zullen geven.

Bouwen

In de webserviceadapter wordt nogmaals een loop gebruikt om een string te bouwen die naar de webservice opgestuurd kan worden.

```
var webserviceString= "":
for (var i = 0; i < message.ElementDataList.Count; i++)
{
        var tuple = message.ElementDataList[i];
        webserviceString += tuple.Item3 + " ";
}
```

Figuur 32: loop voor het opbouwen van de string die naar de webservice gestuurd wordt.

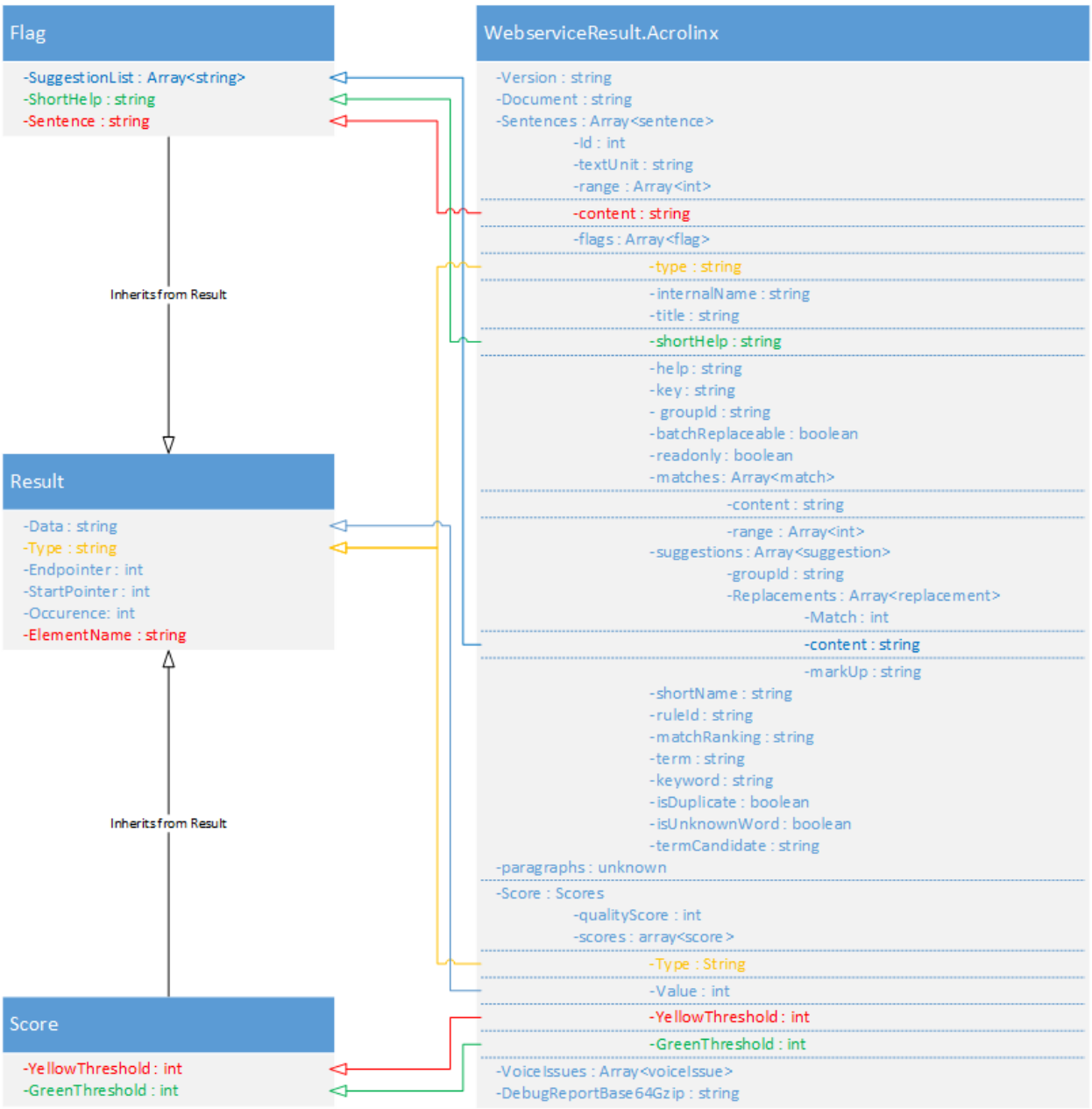
De loop gebruikt alle waardes uit de elementlist en zet deze achterelkaar en zorgt dat tussen elk stuk een spatie staat. De spatie voorkomt dat er per ongeluk woorden achter elkaar worden geplakt en voorkomt dat de webservice hierop fouten zal gaan sturen.

Als de data is opgestuurd naar de webservice zal de webserviceadapter wachten tot er een response komt van de webservice. Als er een response komt van de webservice met de data kan de volgende stap uitgevoerd worden.

10.2.3 Stap 3: Webservice result naar FontoXML result.

Ontwerp

De volgende stap is om de data die van de webservice terugkomt te mappen naar objecten die de centrale server terug zal geven aan de cliënt. Als voorbeeld heb ik hier de mapping van Acrolinx gebruikt. Het gehele schema voor de mapping is te vinden in bijlage #3.



Figuur 33: Acrolinx response object naar result objecten.

De webservice van Acrolinx geeft het volgende object terug die ik moet uitlezen. Dit object komt terug in de vorm van JSON. Het JSON object wordt geparst naar een dynamic object. Voor dynamic objecten zijn geen klasse nodig en worden alle variabele vrij geven. Er is gekozen voor een dynamic

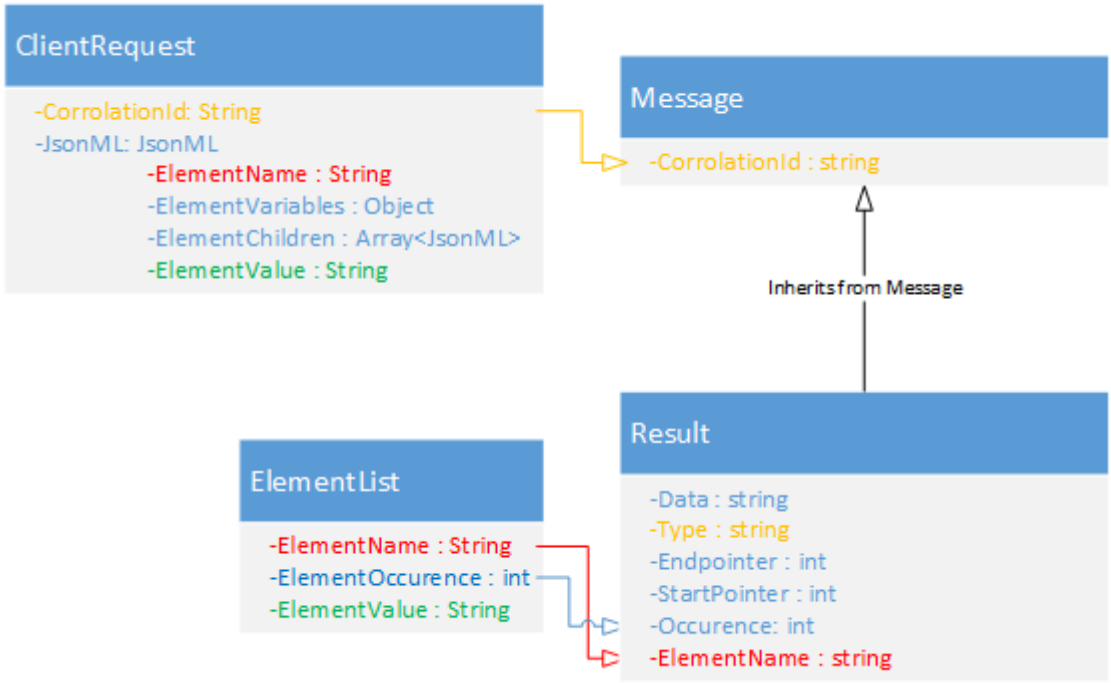
object omdat de resultaten van de webservice niet altijd exact hetzelfde eruit zien. Het kan ook voorkomen dat de webservice een JSON object terug geeft waar een error message in staat.

In figuur 33 is te zien dat uit het Acrolinx object twee nieuwe objecten kunnen worden gecreëerd. Het Score object en het Flag object. Beide objecten hebben overeenkomende waardes, namelijk: ElementName, Occurence, StartPointer, EndPointer, Type en Data. Deze zes waardes worden geërfd van de abstract klasse Result. De abstracte klasse erft op zijn beurt weer van message waar de laatste variabele in staat: CorrolationId. Beide objecten hebben een andere betekenis en daarom andere waardes. Het flag object geeft informatie over spellingsfouten, grammaticafouten, terminologiefouten, kwaliteitsfouten en stijlingsfouten. Als het flag object eenmaal gevuld is ziet deze er als volgt uit:

```
{
        CorrolationId = "0"
        ElementName = "p"
        Occurence = 14
        StartPointer = 0
        EndPointer = 13
        Type = "Spelling"
        Data = "Noord-Holland"
        Sentence = "Noord-Holland"
        ShortHelp = "Check the spelling of this word."
        SuggestionList = [ ]
}
```

Figuur 34: Acrolinx flag result object.

Vertaald zegt dit object het volgende: *"Voor request 0 heb ik een spellingsfout gevonden in de 15de keer dat het p element voorkomt. In dit element staat de spellingsfout op positie 0 tot en met 13. Als het goed is staat hier "Noord-Holland" en ik wil hier graag de volgende melding voor geven: "Check the spelling of this word." Ik heb hier helaas geen suggesties voor kunnen vinden."* De data in SuggestionList, ShortHelp, Sentence, Data en type worden uit het Acrolinx object gehaald. De data in CorrolationId, ElementName en Occcurence worden uit andere object gehaald zoals in het volgende figuur getoond.



Figuur 35: Ontbrekende mapping voor FontoXMLRequest.Flag

De ontbrekende data voor de StartPointer en de EndPointer wordt door de AcrolinxWebserviceClient erin gezet. Hoe dit gebeurt, wordt verder uitgelicht onder de kop bouwen. Nu het object compleet is ingevuld wordt deze teruggestuurd naar de cliënt. Aan de cliënt kant wordt onthouden wat de corrolationId van de request is en welk deel van de tekst is teruggestuurd. Omdat FontoXML gebruik maakt van JavaScript kan aan de hand van deze data op de volgende manier de data op de juiste locatie in de tekst weergegeven worden:

```
xmlDoc.getElementsByTagName("page")[9].getElementsByTageName("p")[14];
```

Figuur 36: Terug mappen van data bij cliënt.

"Page" en 9 komen vanuit de cliënt. De cliënt heeft onthouden welk stuk opgestuurd is naar de centrale server. In dit geval dus de 10de pagina van het document. Aan de hand van bovenstaande functie wordt van de 10de pagina het 15de paragraaf element opgehaald. In dit element kan vervolgens aanpassingen gemaakt worden om de overige data van het result weer te geven.

Bouwen

Om alle data uit te lezen die nuttig is voor de cliënt wordt er gebruik gemaakt van loops om door alle data heen te kunnen itereren. Zoals te zien in het ontwerp staat de nuttige data van de Acrolinx webservice in de sentence array en de score array. Omdat sentences als eerst voorkomt in de response wordt er eerst geïtereerd over de sentence om alle data op te halen. Iedere sentence kan een aantal flags hebben. De flags geven in de sentence fouten aan. Deze fouten kunnen grammaticafouten, spellingfouten, stijlingsfouten, terminologiefouten of kwaliteitsfouten zijn.

```
foreach (var sentence in acrolinxResponse.sentences)
{
        If (_cancelState) return;
        If (sentence.flags == null) continue;
        //Start loop to get all data from flags.
}
```

Figuur 37: Codevoorbeeld om te kijken of een sentence flags heeft.

Tijdens de loop wordt voor iedere sentence die wordt gevonden gekeken of de request nog geldig is. Als de request nog geldig is wordt er gekeken of de sentence een array heeft met flags. Als er geen flags zijn bestaat er ook geen array. Als dit het geval is wordt naar de volgende sentence gekeken. Als er wel een array met flags is wordt er een loop gestart om iedere flag voor deze sentence op te halen.

```
foreach (var flag in sentence.flags)
{
	var flagObject = new Flag();
	flagObject.CorrolationId = fontoXMLrequest.CorrolationId;
	flagObject.StartPointer =
((string)sentence.content).IndexOf((string)flag.matches[0].content);
	flagObject.EndPointer = flagObject.StartPointer +
((string)flag.matches[0].content).Length;
	flagObject.Type = flag.type;
	flagObject.Data = flag.matches[0].content;
	flagObject.Sentence = sentence.content;
	flagObject.ShortHelp = flag.shortHelp;
	//Start loop to get all suggestions.
}
```

Figuur 38: Codevoorbeeld waar de meeste data voor een flag object uit Acrolinx wordt gehaald.

Hier worden de meeste waardes voor een flag object opgehaald. De CorrolationId wordt opgehaald uit het originele request die gestuurd is door de cliënt. StartPointer en EndPointer worden in de loop berekend. Dit wordt gedaan door te kijken naar de inhoud van `flag.matches[0].content`. In deze variabele wordt aangegeven om welk deel van de sentence het gaat. Als op de originele sentence de `.IndexOf` wordt aangeroepen met als inhoud de `flag.matches[0].content` komt hier een getal uit. Dit getal geeft weer op welk punt in de sentence het deel zal beginnen. Vervolgens zal de startpointer gebruikt worden om de endpointer te achterhalen. Dit gebeurt door bij de startpointer de lengte van het gevonden deel op te tellen. Nadat zowel de StartPointer als EndPointer op zijn gehaald wordt het type opgehaald. Het type kan vijf verschillende waardes hebben.

- GRAMMAR
- SPELLING
- TERM
- QUALITY
- STYLE

Afhangend van deze variabele moet de cliënt op verschillende manieren reageren. Dit moet gedaan worden in de cliënt zelf maar dit valt tijdens dit project buiten de scope. In Data wordt opgeslagen om welk deel het gaat van de zin als tekst. In sentence wordt opgeslagen wat de volledige zin is waar wat over wordt gezegd. In de shorthelp wordt een korte verklarende tekst van Acrolinx opgeslagen die wat verteld over de melding. Op dit moment is bijna alle data voor het flag object opgeslagen. Wat ontbreekt, zijn de suggesties die worden gegeven door Acrolinx. Deze worden opgehaald door nogmaals een loop te starten.

```
foreach (var suggestion in flag.suggestions)
{
	foreach (var replacement in suggestion.replacements)
	{
		flagObject.SuggestionList.Add((string)replacement.content);
	}
}
//Get the last variables to complete flag object.
```

Figuur 39: Codevoorbeeld voor het ophalen van suggesties.

Een flag kan meerdere suggesties hebben om de fout op een manier te kunnen verbeteren. Het is voor mij onduidelijk waarom hier twee niveaus zijn gebruikt. Maar per suggestie zijn er meerdere replacements. De informatie die in replacements staan is interessant voor de cliënt. Deze worden opgehaald en toegevoegd aan de suggestielijst van het Flag Object. Op dit punt is er voor deze flag geen extra data meer te halen uit de AcrolinxResponse Object. Het enige wat nog ontbreek in het flag object is de originele locatie in de tekst van de cliënt. Deze wordt als volgt toegevoegd aan het flag object.

```
Foreach (var element in message.ElementDataList.Where(element => ((string)
sentence.content).Contains(element.Item3) || element.Item3.Contains((string)
sentence.content)))
{
	flagObject.ElementName = element.Item1;
	flagObject.Occurence = element.Item2;
	break;
}
```

Figuur 40: Codevoorbeeld voor het ophalen van de laatste mapping.

In de ElementDataList staan alle elementen met de daar bijbehorende locatie in de originele tekst van de cliënt. Uit deze lijst moet het juiste element gehaald worden om de mapping op te halen die daarbij is opgeslagen. Zodra in de loop het juiste element is gevonden kan hier de elementName en de occurence opgehaald worden en in het flagObject gezet worden. Nu zijn alle variabele voor het FlagObject gevuld en kan deze teruggestuurd worden naar de cliënt.

```
result(JsonConvert.SerializeObject(flagObject));
```

Figuur 41: Codevoorbeeld voor het terug sturen van een flagObject.

In deze regel code wordt het flagObject omgezet naar een string zoals beschreven in figuur 34. Eenmaal omgezet wordt deze met behulp van de action result teruggestuurd naar de cliënt. Nu het flag object is teruggestuurd kan er gekeken worden of een sentence nog een flag heeft. Als dit het geval is zal er nog een flag worden teruggestuurd totdat de sentence geen flags meer heeft. Als de sentence geen nieuwe flag meer heeft kan er gekeken worden naar de volgende sentence. Dit zal zichzelf blijven herhalen tot de laatste sentence geen nieuwe flags meer heeft om terug te sturen.

Een vergelijkende loop is ook opgezet voor het score object. Voor een score object wordt er alleen andere informatie gevonden. Als ook alle score objecten zijn teruggestuurd is er geen data meer voor de cliënt vanaf de Acrolinx webservice. De AcrolinxWebservinceClient zal dan gebruik maken van de done Action om aan de webservice dispatcher aan te geven dat hij klaar is.

10.3 Testen

Ook voor de laatste sprint is een codereview uitgevoerd. Deze keer samen met ontwikkelaar Martin Middel. Ik ben samen met hem opnieuw door alle code gelopen. Hij zag de code voor de eerste keer. Hierdoor had ik een frisse blik die met mij mee kon kijken. Ik heb hem alle code uitgelegd en hij heeft daar waar nodig opmerkingen gemaakt. Veel van deze opmerkingen zijn gegaan over waarom ik verschillende keuzes heb gemaakt. Na uitleg te geven kon hij ook inzien waarom ik de keuze had gemaakt. Een van deze opmerkingen ging over de manier van mapping en waarom ik voor deze vorm had gekozen en niet voor een andere manier. Overige opmerkingen die hij kon maken over de code hebben niet geleid tot grote veranderingen in de structuur. Er zijn wel aanpassingen geweest in de code. Een van deze aanpassingen verminderde de code. In de code wordt gecontroleerd of een request nog geldig is. Deze controle had ik, volgens Martin, teveel geïmplementeerd. Na de codereview heb ik zelf gekeken of dit het geval was. Ook ik kwam er achter dat de controle beter kon functioneren als ik op het juiste moment controleerde. Voor de rest had Martin geen opmerkingen die voor code verandering hebben geleid.

10.4 Reflectie

Deze sprint vond ik de lastigste sprint. Omdat ik hierzelf de data van de webservice moest interpreteren en omvormen zodat deze nuttig zou zijn voor de cliënt. Aan de hand van de documentatie die over de webservice beschikbaar was werd mij duidelijk wat de kern functionaliteiten waren. Aan de hand hiervan heb ik de nuttige data uit de webservice kunnen halen. Het volgende probleem lag bij de mapping van de data. Omdat de webservice niet de structuur van het originele document onthield moest ik deze zelf weer zien terug te zetten. Op dit moment is het functioneel maar ik zie daar nog veel ruimte voor verbetering. Zeker als er nieuwe webservices worden toegevoegd zullen er nieuwe inzichten plaats vinden voor de mapping.

11 Adviesrapport

11.1 Veranderingen door de server

De volgende veranderingen vind ik de belangrijkste veranderingen die door het implementeren van de centrale server zullen plaatsvinden. Vanwege het belang heb ik deze hier verder uitgelicht.

11.1.1 Requests duren langer

Requests die FontoXML stuurt zullen langer duren dan in de huidige situatie. In de huidige situatie stuurt FontoXML de requests direct naar de webservices. In de nieuwe situatie zal dit via de centrale server gaan lopen. Hierdoor wordt het aantal keer dat er data over een verbinding wordt gestuurd verdubbeld. De kans dat opgestuurde requests hierdoor niet meer nodig zijn neemt hierdoor toe. In de tijd dat de request is opgestuurd en weer terugkomt, kan de gebruiker de tekst al aangepast hebben. Ook hebben aanpassingen een beperkte invloed. In de gekozen mapping zijn veranderingen alleen geldig als deze plaats hebben gevonden in de tekst value van een element of als er een element die meerdere keren voorkomt is toegevoegd.

```
Situatie 1:
<p>Deze tekst dient als voorbeeld.</P>

Situatie 2:
<p>Deze tekst wordt als voorbeeld gebruikt.</p>
```

Figuur 42: Aanpassen van een element.

```
Situatie 1:
<p>Dit is het eerste voorbeeld element.</P>
<p>Dit is het tweede voorbeeld element.</p>

Situatie 2:
<p>Dit is het eerste voorbeeld element.</P>
<p>Dit is een nieuw element.</P>
<p>Dit is het tweede voorbeeld element.</p>
```

Figuur 43:Toevoegen van een element.

11.1.2 Sturen van request is lichter

In de huidige situatie moet voor iedere request die gestuurd wordt een nieuwe connectie met de webservice gemaakt worden. De browser houdt deze connecties bij. In de toekomst wilt FontoXML meer webservices gaan implementeren waardoor het aantal verbindingen in de huidige situatie zal toenemen. Hierin ligt het gevaar dat een maximaal aantal verbindingen per browser bereikt zal worden. Met de centrale server zal er nog steeds connecties opgezet worden per webservice. Het verschil is dat de centrale server geen limiet heeft aan maximale connecties. Het voordeel dat de server deze nu afhandelt, in plaats van de browser waar FontoXML in draait, is dat het niet meer bijgehouden hoeft te worden in de browser. Door het gebruik van websockets zal er altijd maar 1 verbinding openstaan naar 1 host. Hierdoor zal de grens van maximaal openstaande connecties door de browser nooit bereikt worden.

Compare ▼ name	score	PerfTiming	Connections per Hostname ↑	Max Connections
Chrome 34 →	12/16	yes	6	10
Firefox 27 →	11/16	yes	6	17
Safari 7.0.1 →	11/16	no	6	17
IE 11 →	12/16	yes	13	17

Figuur 44: Vergelijking maximaal aantal connecties via http://www.browserscope.org/

Voor iedere request moet er opnieuw headerdata worden gestuurd. In de huidige situatie wordt dit gedaan door de browser. Hierdoor wordt de browser opnieuw belast. In de nieuwe situatie wordt dit afgehandeld door de centrale server. De browser moet nu eenmalig een header sturen naar de centrale server om de verbinding open te zetten. Hierna kan de browser data sturen zonder zich zorgen te hoeven maken over nieuwe headers.

11.2 Toekomstige technieken

Http1.1 is de belangrijkste reden waarom een browser maar een beperkt aantal connecties open kan hebben. Volgens rfc2616 hoofdstuk 8.1.4[1] wordt aangeraden om maar tot twee verbindingen te ondersteunen. Http1.1 is een verouderde technologie en is gebaseerd op een aanvraag response model. Tijdens de ontwikkeling van Http werd gedacht dat dit de oplossing zou zijn voor de communicatie over internet. In de huidige tijd zijn de wensen van ontwikkelaars veranderd en willen meer uit http kunnen halen. Ontwikkelaars willen bijvoorbeeld in real-time data kunnen aanpassen op de webpagina. Met http moet de gehele webpagina opnieuw geladen worden en dat willen ontwikkelaars voorkomen. Hiervoor zijn verschillende technieken ontwikkeld die dit kunnen, waaronder JavaScript. JavaScript draait alleen in de browser en wordt dan ook met de Http request opgehaald. Daarna zal er geen informatie meer opgehaald worden van het Http adres. Het kan wel voorkomen dat in de JavaScript verbinding wordt gelegd met bijvoorbeeld een database die extra informatie levert. Ook deze verbindingen functioneren als een request response verbinding. In tegenstelling tot Http wordt hier niet alle beschikbare data opgestuurd en alleen de data waarin de cliënt geïnteresseerd is. Hierdoor zijn ontwikkelaars in staat geweest om webpagina's dynamischer te laden. Hoewel de webpagina dynamischer is geworden is het nog steeds niet wat ontwikkelaars willen zien. Na verdere ontwikkeling is uiteindelijk websockets ontwikkeld. Websockets zet ontwikkelaars instaat om asynchroon en dynamische webpagina's te kunnen ontwikkelen. Ze

[1] (Fielding, et al., 1999)

hoeven niet iedere keer opnieuw requests te sturen en te wachten op antwoord. Ze kunnen nu een verbinding openen met een server en over deze verbinding full-duplex communiceren.

Als reactie op de ontwikkeling is Google gaan kijken naar een nieuwe versie voor http om de oude techniek te vervangen door een nieuwe techniek die de nieuwe technologieën ondersteund. Deze nieuwe techniek is SPDY genoemd. Hieraan heeft google gewerkt en heeft later besloten het als een open standaard te beschouwen en is samen met het W3C aan de slag gegaan om het verder te ontwikkelen. In deze samenwerking is SPDY omgedoopt tot HTTP2.0. HTTP2.0 moet ontwikkelaars in staat stellen de nieuwe technieken toe te passen over een http verbinding zonder dat daar speciale technieken voor gebruikt moeten worden. HTTP2.0 is nog niet compleet en in de laatste specificatie staat beschreven dat het ondersteuning biedt voor een connectie die vergelijkbaar is met websockets.[2]

Als de centrale server doorontwikkeld wordt en in productie wordt gebracht is het handig deze techniek in de gaten te houden. HTTP 2.0 kan verbeteringen mee brengen voor de communicatie tussen de cliënt en de centrale server.

11.3 Door ontwikkeling

11.3.1 Authenticatie / Autorisatie

Om de centrale server te beveiligen van ongewenste gebruikers moeten nieuwe connecties geauthentiseerd worden. Als de verbinding geauthentiseerd is moet gekeken worden of de verbinding ook geautoriseerd is om te doen wat deze wil doen. Door middel van de autorisatie kunnen de functionaliteiten beperkt worden. Het is gewenst om functionaliteiten te kunnen beperken zodat deze tegen betaling aangeboden kunnen worden aan klanten.

11.3.2 Configuratie database

Op dit moment zijn er delen die hardcoded in de applicatie staan. Hieronder valt voornamelijk de configuratie die nodig is voor de webservices. Mijn advies is om deze hardcoded delen uit de applicatie te halen en in een database te plaatsen waar ze makkelijker aan te passen zijn als dat nodig is.

De database is niet alleen voor de webservice maar kan ook gebruikt worden om klantgegevens in op te slaan voor de authenticatie en autorisatie.

11.3.3 Nieuwe webservices

Nieuwe functionaliteiten kunnen altijd gevraagd worden vanuit de gebruiker. Om hier snel op in te kunnen spelen is het nuttig om al vooruit te blikken naar mogelijk nieuwe bruikbare webservices die in de toekomst aangevraagd kunnen worden door klanten. Tevens kunnen de huidige webservices misschien door andere webservices vervangen worden voor betere functionaliteiten. Om als voorbeeld te nemen kunnen we kijken naar Acrolinx. Acrolinx geeft op het moment extreem veel informatie over de data die opgestuurd wordt. Minpunt aan Acrolinx is dat al deze data in een klap zal worden gegeven en niet in kleine delen kan worden opgevraagd. Hierdoor komt er vaak veel ongebruikte data terug die niet interessant is voor het verstuurde request. Al deze data moet verzameld worden in Acrolinx en zal de tijd die nodig is om de controle uit te voeren vergroten. Als

[2] (Belshe, et al., 2015)

er hier voor in de plaats bijvoorbeeld meerdere webservices komen die gecombineerd hetzelfde kunnen als Acrolinx kan er veel preciezer data worden opgevraagd.

Tevens kan het zijn dat er meerdere webservices zijn voor dezelfde functionaliteit alleen dat bijvoorbeeld, de taal afwijkend is. Dit is op het moment een van de grotere beperkingen. Omdat hierdoor sommige webservices helemaal niet beschikbaar kunnen worden gesteld voor een klant.

11.3.4 Log functionaliteiten.

De data mutatie voor de webservice is gevoelig voor fouten. Hierdoor is het noodzakelijk om duidelijk terug te kunnen halen welke fout waar is ontstaan. Komt de foutmelding doordat er wijzigingen zijn gemaakt in de webservice? Of is FontoXML voor een nieuwe klant ontwikkeld die nieuwe structuren gebruiken in de documenten. Het is van belang deze traceerbaar te maken zodat deze fouten afgevangen en opgelost kunnen worden.

Naast het oplossen van fouten in het programmeerwerk is het een nuttige tool om te bekijken welke services veel worden gebruikt en welke nauwelijks. Uit deze informatie kunnen trends gevonden worden en hier kan op ingespeeld worden.

12 Evaluatie

12.1 Procesevaluatie

Over het algemeen ben ik tevreden over hoe ik het proces heb aangepakt ondanks dat deze aan het begin van het traject lastig in te delen was. Aan het begin van het traject was nog onduidelijk hoe de opdracht voltooid kon worden. Het traject heeft afgehangen van het onderzoek dat ik aan het begin heb uitgevoerd. Ondanks de onzekerheden heb ik aan het begin een redelijke planning op kunnen stellen en mij hieraan kunnen houden. Er heeft wel een wijziging in de planning plaatsgevonden tijdens het traject. De wijziging kwam doordat ik zelf een ontwikkeldeel heb onderschat. De centrale server zelf heeft veel meer tijd ingenomen dan ik aan het begin had verwacht. Hierdoor is de front-end buiten de scope geplaatst om tijd vrij te maken voor het verder ontwikkelen van de centrale server zelf. Dit is in overleg geweest met de bedrijfsmentor en ik vind dat ik deze verandering op de juiste manier heb opgelost.

Ik heb tijdens dit project veel meer kennis opgedaan over hoe SCRUM functioneert in een organisatie. Ik heb de methode als fijn kunnen ervaren en zal deze ook in vervolgprojecten willen gebruiken als dit wordt toegelaten. Dankzij de ingeplande sprints heb ik mijzelf structuur kunnen geven aan mijn werkzaamheden.

Het traject heeft tot gewenste resultaten geleid. Desondanks zou ik het project een volgende keer anders aanpakken. Door de onzekerheden die aan het begin waren van het traject heb ik tijd besteed aan activiteiten die later niet bruikbaar waren. Als ik aan het begin van het traject meer tijd had besteed om deze onzekerheden weg te nemen had ik hierdoor beter kunnen focussen op de activiteiten die gedaan moesten worden.

12.2 Productevaluatie

In dit hoofdstuk zal ik mijn opgeleverde producten evalueren. Ieder document die beschreven staat in mijn opdrachtomschrijving wordt kort toegelicht en beschreven hoe deze tijdens het traject zijn terug gekomen. De grote producten die zijn ontwikkeld tijdens het traject zullen een diepere evaluatie krijgen.

12.2.1 Onderzoeksrapport

Het onderzoeksrapport is een van de twee grote producten die zijn opgeleverd tijdens het traject. Het onderzoeksrapport geeft antwoord op de hoofdvraag*: "Welke communicatievormen moet de centrale server kunnen ondersteunen om de add-ons van FontoXML op een asynchrone manier te kunnen faciliteren?"*. Om deze hoofdvraag te beantwoorden heb ik een drietal deelvragen opgesteld.

- Welke vorm van communicatie tussen de cliënt en de centrale server is het meest effectief?
- Welke technieken kunnen de gekozen communicatievorm ondersteunen?
- Welke invloeden kunnen webservices hebben op de ontwikkeling van de centrale server?

Als ik iedere deelvraag heb onderzocht kan ik antwoord geven op de deelvragen zelf en samen kan ik een antwoord geven op de hoofdvraag. Ik vind dat de kwaliteit van het onderzoeksrapport varieert per beantwoorde vraag. De eerste deelvraag: *"Welke vorm van communicatie tussen de cliënt en de centrale server is het meest effectief?"* vind ik kwalitatief goed en het best uitgewerkt in vergelijking met de andere deelvragen. Voor het antwoorden van deze deelvraag wordt duidelijk uiteengezet welke mogelijkheden er zijn en op welke manier deze kunnen gaan werken met een centrale server.

Voor iedere vorm wordt duidelijk omschreven wat de voordelen en nadelen zijn. Hieruit wordt vervolgens een duidelijke conclusie getrokken. Ter afronding van deze deelvraag heb ik mijn advies voorgelegd aan de bedrijfsmentor. De manier waarop ik deze deelvraag heb opgepakt, hebt beantwoord en heb afgerond vind ik goed. Ik ben dan ook tevreden over de kwaliteit van deze deelvraag.

De volgende deelvraag: *"Welke technieken kunnen de gekozen communicatievorm ondersteunen?"* is een onderzoek die volgt op de conclusies die ik heb getrokken uit de eerste deelvraag. Uit onderzoeksperspectief vind ik deze deelvraag van mindere kwaliteit. De hoeveelheid aan technieken die beschikbaar zijn maakt het voor mij onmogelijk om alles te bekijken. Om deze reden heb ik mijn selectie aan technieken moeten beperken. De focus heeft gelegen op technieken die later ondersteund kunnen worden door developers die werken bij Liones. De grootste kennis die de medewerkers hebben zijn van JavaScript en van C#. Ik heb hiervoor een aantal technieken geselecteerd in deze programmeertalen en deze met elkaar vergeleken. Door de beperking heb ik niet naar technieken gekeken waarvoor nieuwe kennis opgedaan moet worden. Het had kunnen zijn dat er technieken bestaan die zoveel meerwaarde bieden dat de kennis hiervoor in huis gehaald wordt. Door de beperkingen vind ik deze deelvraag minder goed uitgewerkt en ben hier minder tevreden over.

De derde deelvraag: *"Welke invloeden kunnen webservices hebben op de ontwikkeling van de centrale server?"* is een onderzoek geweest om te kijken op welke manier webservices de centrale server beïnvloeden. Dit onderzoek vind ik ook van mindere kwaliteit. Er zijn ruim over 13.000 verschillende webservices beschikbaar. Het is onmogelijk voor mij om al deze webservices te bekijken. Hierdoor heb ik maar een beperkt aantal webservices bekeken. Door deze beperking ben ik bang dat ik invloeden van webservices heb gemist. De informatie die dit onderzoek heeft opgeleverd is nuttig geweest voor de ontwikkeling van de centrale server. Ondanks dat de kans bestaat dat ik een belangrijke invloed heb gemist heb ik wel het gevoel dat hier voldoende rekening mee is gehouden tijdens de ontwikkeling. De uitvoer van deze deelvraag vind ik niet goed maar de opgeleverde informatie is zeer nuttig geweest en daar ben ik tevreden mee.

Als naar het gehele onderzoeksrapport wordt gekeken ben ik tevreden over de structuur van het document. Er wordt duidelijk omschreven waar ik een antwoord op wil hebben en op welke manier ik hierop een antwoord ga geven. Als er wordt gekeken naar de uitvoer is dit verschillend per deelvraag. Bij de eerste deelvraag ben ik tevreden over het verloop hiervan. Bij de tweede en derde deelvraag ben ik hier minder tevreden over. Desondanks is de informatie die de deelvragen opgeleverd hebben nuttig geweest voor het project. Alles bij elkaar genomen ben ik tevreden met het onderzoek ondanks dat er verbeterpunten zijn.

12.2.2 Proof of Concept

Het proof of concept heb ik ontwikkeld aan de hand van het onderzoek. Voor het proof of concept heb ik een aantal modellen opgesteld om de structuur van de centrale server in kaart te brengen. Aan de hand van deze modellen heb ik het proof of concept geprogrammeerd. Het geprogrammeerde dient te bewijzen dat de opgestelde modellen kunnen functioneren. Aan het eind van het traject is dit ook het geval geweest. Het proof of concept heeft bewezen dat de centrale server data van de cliënt kan ontvangen, deze kan omvormen en kan doorsturen naar de webservice. De data van de webservice wordt weer omgezet door de centrale server naar een

manier waarop de cliënt deze kan lezen en kan gebruiken. Het belangrijkste is hiervoor dat de centrale server kan aangeven waar deze data terug moet komen in de cliënt. Tevens heeft het proof of concept bewezen dat het eenvoudig is om nieuwe webservices toe te voegen. Iedere webservice heeft zijn eigen klasse en maakt gebruik van een interface om het programmeerwerk te vereenvoudigen. De berichten die door de centrale server worden teruggestuurd verschillen per controle die wordt uitgevoerd. Ook hiermee is voldoende rekening gehouden tijdens het ontwerp. Het proof of concept heeft hiermee kunnen bewijzen wat het moest bewijzen. Tevens kan het proof of concept doorontwikkeld worden om deze productie klaar te maken. Ik ben zelf tevreden over het proof of concept.

12.2.3 Overige producten

In de opdrachtomschrijving staan een tal van producten genoemd die aan het eind van het traject opgeleverd worden. Tijdens het traject zijn er inzichten opgedaan over hoe deze documenten terug komen in het verslag. Hier zal ik beschrijven hoe deze documenten terug zijn gekomen in het verslag.

Het Plan van Aanpak heeft een hoofdstuk gekregen in het verslag dat de inhoud bespreekt.

De beschrijving van de huidige situatie is tijdens de oriëntatiefase van het afstudeertraject opgezet. Aan de hand van dit document heb ik mijzelf meer inzicht gegeven over hoe de huidige applicatie functioneert. Dit document is in bijlage #4 te vinden.

In de opdrachtomschrijving staat dat er twee onderzoeksrapporten komen. Een onderzoeksrapport voor de centrale server en een onderzoeksrapport over de beschikbare webservices. Tijdens het traject bleek er nog een onderzoek nodig te zijn. Deze drie onderzoeken zijn samen in een onderzoeksrapport gekomen. Dit onderzoeksrapport heeft zijn eigen hoofdstuk gekregen in het verslag.

De API omschrijvingen beschrijven hoe de Acrolinx en WizeNoze webservices functioneren. Welke data deze teruggeven, hoe deze data eruit ziet en hoe deze data terug wordt gestuurd naar de cliënt. Dit document is te vinden in bijlage #2.

De verantwoording voor de gekozen methoden en ontwikkeltools komen vooral voort uit de uitgevoerde onderzoeken. Tijdens de onderzoeken zijn veel methodes en technieken bekeken en hieruit goed beargumenteerde conclusies getrokken.

Het requirements document is door alle sprints bijgewerkt. Tijdens de sprints werden nieuwe inzichten gedaan en kwamen er nieuwe requirements naar boven. Deze requirements zijn bijgehouden in het requirements document welke te vinden is in bijlage #5.

Tijdens het afstudeertraject bleken zowel de beschrijving voor de centrale server en het ontwerp voor de prototype server dicht bij elkaar te liggen. Om deze reden heb ik besloten deze documenten samen te voegen in een document. In dit document is de beschrijving van de centrale server te vinden. Deze is terug te vinden in bijlage #7.

Het prototype applicatie is de geschreven code. In het verslag worden verschillende functionaliteiten met behulp van codevoorbeelden uitgelegd.

De testmethode is omschreven in Bijlage #6. Door de gekozen vorm van testen zijn er geen testcases opgesteld.

Door de gekozen testmethode zijn er bij verschillende sprints codereviews uitgevoerd. De uitkomsten van deze codereviews zijn bij de sprints opgenomen.

Na de sprints is er een adviesrapport opgesteld om alle bevinden op te schrijven. Het adviesrapport heeft een hoofdstuk gekregen in het verslag.

12.3 Competentie evaluatie

12.3.1 Selecteren methoden, technieken en tools.

Selecteren en adviseren over methoden en technieken om een systeem te ontwikkelen, beheren en te testen.

Ik heb deze competentie behaald tijdens mijn afstudeertraject. Dit heb ik gedaan door middel van het onderzoek naar de meest effectiefste vorm van communicatie tussen de cliënt en centrale server. Uit dit onderzoek heb ik onder andere de selectie gemaakt om gebruik te maken van websockets.

12.3.2 Selecteren van standaardsoftware.

Adviseren over en/of selecteren van software, zoals een applicatie (commerciële software of open source), framework (al dan niet gebaseerd op open standaarden), databasemanagementsysteem of besturingssysteem.

Tijdens het onderzoek heb ik duidelijke inzichten gekregen over hoe het proof of concept zou moeten gaan functioneren. Hiervoor heb ik duidelijke keuzes gemaakt en deze met argumentatie kunnen onderbouwen. Aan de hand van deze argumenten heb ik ook mijn bedrijfsmentor kunnen overtuigen om mijn adviezen te volgen.

12.3.3 Ontwerpen Systeemdeel

Beschrijven van systeemdelen (subsystemen, componenten, modules), hun onderliggende structuur en het gedrag in detail, zodanig dat bouwen van het systeemdeel mogelijk is.

Aan de hand van het opstellen van verschillende UML modellen door het project heen heb ik bewezen deze competentie voldoende onder de knie te hebben. Met deze UML modellen heb ik voor mijzelf en voor collega's duidelijk kunnen maken hoe de centrale server functioneert. Ik heb in deze UML modellen complexe functionaliteiten kunnen beschrijven en deze kunnen uitwerken.

12.3.4 Bouwen applicatie

Verfijnen en/of transformeren van het (detail)ontwerp van de systeemdelen (subsystemen, componenten, modules) van een applicatie.

Door het gebruik van C# en JavaScript heb ik verschillende systeemdelen kunnen ontwikkelen om de centrale server uit te kunnen werken. Aan de hand van de opgestelde UML documenten is het mij gelukt om de centrale server te ontwikkelen en te laten functioneren. Hierdoor heb ik bewezen deze competentie voldoende te kunnen.

13 Termenlijst

- JSON
  - JavaScript Object Notation. JSON is een licht gewicht data uitwisseling notatie. JSON is zowel voor mensen als machines eenvoudig te lezen.
- XML
  - EXtensible Mark-up Language. XML wordt gebruikt om tekst in data te kunnen markeren en een structuur te geven. Hierdoor is deze voor zowel mensen als machines eenvoudig te lezen is.
- JsonML
  - Combinatie van JSON en XML. Hierdoor kan de structuur die door XML wordt beschreven eenvoudig worden uitgewisseld met andere systemen.
- DOM
  - Document Object Model. DOM wordt gebruikt om te beschrijven op welke manier de interactie met objecten werkt en op welke manier deze objecten worden gepresenteerd.
- SOAP
  - Simple Object Access Protocol. Is ontwikkeld om gestructureerde data op veilige manier te kunnen versturen over het internet.
- REST
  - REpresentational State Transfer. REST is een minder veilige maar meer gebruikte manier van communicatie over het internet. Het doet hetzelfde als SOAP maar geeft minder prioriteit aan veiligheid en meer prioriteit aan lichte berichten.
- XMLDOM
  - Combinatie van XML en DOM. Uit de combinatie komt een duidelijke beschrijving van objecten en hoe de interactie, presentatie en structuur eruit ziet.
- AJAX
  - Asynchronous Javascript And Xml. AJAX is ontwikkeld om een asynchroon werkende web applicatie te kunnen ontwikkelen. Ondanks de naam hoeft XML niet gebruikt te worden voor het versturen van data. AJAX kan een request versturen en een response ontvangen.
- COMET
  - COMET is een variatie op AJAX. COMET maakt gebruik van de AJAX techniek om request die langer duren later op te halen.
- Websocket
  - De websocket is dat wat COMET probeerde te bereiken. Door middel van websockets is het mogelijk om een connectie te openen en over de connectie van zowel de cliënt als de server data te sturen wanneer deze beschikbaar is.
- HTTP
  - Hypertext Transfer Protocol. HTTP is het protocol dat wordt gebruikt om applicaties te laten communiceren over het internet.
- TLS
  - Transport Layer Security. TLS is een laag over HTTP om te zorgen voor extra security over de verbindingen die worden opgezet.
- Full-Duplex communicatie

    - Bij Full-Duplex communicatie zijn de gebruikers instaat gelijktijdig met elkaar te kunnen communiceren. Het is vergelijkbaar met een telefoongesprek tussen twee personen waar beide personen op hetzelfde moment kunnen praten en luisteren.
- Acrolinx
    - Acrolinx is een bedrijf met een webservice die gebruikt kan worden om meer informatie te verkrijgen over opgestuurde teksten. Bij de extra informatie hoort bijvoorbeeld spellingscontrole, terminologie, grammatica en scores over de gehele tekst.
- WizeNoze
    - WizeNoze is ontwikkeld om auteurs te helpen met het testen van hun tekst of het op het juiste niveau voor de doelgroep is.
- PoolParty
    - PoolParty wordt gebruikt om de terminologie binnen bedrijf consistent te houden.

Bijlage #1: Opdrachtomschrijving

Afstudeerplan

Informatie afstudeerder en gastbedrijf (structuur niet wijzigen**)**

Afstudeerblok: 2015-1.1 (start uiterlijk 9 februari 2015)
Startdatum uitvoering afstudeeropdracht: 09-02-2015
Inleverdatum afstudeerdossier volgens jaarrooster: 5 juni 2015

Studentnummer:	09003614
Achternaam:	dhr Post
Voorletters:	I.J.
Roepnaam:	Ivar
Adres:	Rijswijkseweg 348
Postcode:	2516HN
Woonplaats:	Den Haag
Telefoonnummer:	
Mobiel nummer:	06-30712433
Privé emailadres:	Ivar.post@live.com

Opleiding:	Informatica
Locatie:	Zoetermeer
Variant:	voltijd

Naam studieloopbaanbegeleider:	A.A.A.M Jacobs
Naam begeleidend examinator:	John Smeets
Naam tweede examinator:	Nanny Jacobs

Naam bedrijf:	Liones
Afdeling bedrijf:	
Bezoekadres bedrijf:	Polakweg 7
Postcode bezoekadres:	2288GG
Postbusnummer:	2846
Postcode postbusnummer:	
Plaats:	Rijswijk
Telefoon bedrijf:	070-31 91 923
Telefax bedrijf:	
Internetsite bedrijf:	www.liones.com

Achternaam opdrachtgever:	Dhr. Willems
Voorletters opdrachtgever:	B.
Titulatuur opdrachtgever:	
Functie opdrachtgever:	Product Architect
Doorkiesnummer opdrachtgever:	

Email opdrachtgever: willems@fontoxml.com

Achternaam bedrijfsmentor: Dhr. Willems
Voorletters bedrijfsmentor: B.
Titulatuur bedrijfsmentor:
Functie bedrijfsmentor: Product Architect
Doorkiesnummer bedrijfsmentor:
Email bedrijfsmentor: willems@fontoxml.com

Doorkiesnummer afstudeerder:
Functie afstudeerder (deeltijd/duaal):

Titel afstudeeropdracht:

Onderzoek doen naar de verbetering van de communicatie tussen FontoXML en Web services door middel van het gebruik van een serveromgeving.

Opdrachtomschrijving

1. **Bedrijf**

   Liones is een internetbureau voor bedrijven die grote hoeveelheden content geproduceerd, beheerd en gecontroleerd moeten hebben. Hiervoor wordt FontoXML gebruikt. FontoXML is een web-based XML editor waar de auteur zijn tekst kan converteren naar het XML formaat en de standaarden van het bedrijf waar de auteur werkt. Voor de gebruiker is er zo min mogelijk XML zichtbaar. Liones ontwikkelt voor deze web applicatie nieuwe integraties van web services die nuttig zijn voor de klant. Bijvoorbeeld een add-on die controleert of de inhoud uniek genoeg is en daarmee de Search Engine Optimization (SEO) score verbeterd. Door het verbeteren van de SEO score komt de website hoger te staan in de resultaten lijst van search engines zoals Google.

2. **Probleemstelling**

   FontoXML communiceert met meerdere web services tegelijkertijd. Voor de ene web service worden weinig aanvragen gedaan maar per aanvraag veel informatie verstuurd. Ook is er de mogelijkheid dat een web service heel veel aanvragen stuurt met weinig informatie. Op dit moment worden deze aanvragen asynchroon aan de cliënt side in de browser bijgehouden. Het probleem hierbij is dat de browser vol loopt met het bijhouden van alle verschillende aanvragen. De student gaat onderzoeken of een server de communicatie tussen de web services kan afhandelen en verbeteren.

   Tevens bekijkt de student de huidige web services die worden gebruikt en onderzoekt of er nieuwe web services of andere web services gebruikt kunnen worden.

3. **Doelstelling van de afstudeeropdracht**

   Het doel van de afstudeerstage is om binnen een periode van 17 weken te bewijzen dat door middel van het gebruik van een server, FontoXML effectiever kan communiceren met de

web services. In plaats van dat aan de cliënt side meerdere connecties met de web services open worden gehouden wordt er nog maar 1 connectie met de server onderhouden. De server handelt alle aanvragen af met de web services. De student gaat deze serveromgeving ontwikkelen. De student onderzoekt welke software distributie model hiervoor wordt gebruikt en waar deze server komt te draaien.

4. **Resultaat**

Het resultaat van het project is een beschrijving voor de opdrachtgever waaruit blijkt wat de meerwaarde van de server is en waarom dit verbeteringen in de communicatie tussen FontoXML en de web services oplevert. Als proof of concept levert de student een prototype op waar uit blijkt dat FontoXML effectiever functioneert als de web services via een server worden afgehandeld. Voor het prototype maakt de student een testrapport waarin de resultaten van de gelopen tests staan.

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

Voorbereidende fase:

- Plan van aanpak opstellen.

Oriëntatie fase:

- Beschrijven van de huidige situatie.
- Opstellen onderzoeksrapport over beschikbare Cloud omgevingen.
- Opstellen onderzoeksrapport over beschikbare Web Services.
- Opstellen API omschrijvingen per gevonden Web Service.
- Beschrijven gekozen methodes en ontwikkeltools op basis van de onderzoeksrapporten.
- Opstellen requirements voor de prototype applicatie.

Ontwikkelfase:

- Ontwerpen server (Uit de oriëntatie fase moet duidelijk worden of deze Cloud based gaat zijn of niet).
- Ontwerpen prototype applicatie.
- Ontwikkelen prototype applicatie.
- Ontwikkelen van de beschrijving van de functionaliteiten van de applicatie.
- Opstellen test scenario’s

Evaluatiefase:

- Uitvoeren van de test scenario’s.
- Adviesrapport over vervolg activiteiten ontwikkelen.

Fase / Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Voorbereidende fase																	
Oriëntatie fase																	
Ontwikkel fase																	
Evaluatie fase																	

6. **Op te leveren (tussen)producten**

- Plan van aanpak
- Beschrijving van huidige situatie.
- Onderzoeksrapport over beschikbare Cloud omgevingen.
- Onderzoeksrapport over beschikbare web services.
- API omschrijvingen per gevonden Web Services.
- Verantwoording gekozen methodes en ontwikkeltools.
- Requirements document.
- Beschrijving van server structuur.
- API prototype applicatie.
- Ontwerp prototype applicatie.
- Prototype applicatie.
- Omschrijving testcases en testmethode.
- Testrapport.
- Adviesrapport.
- Stageverslag.

7. **Te demonstreren competenties en wijze waarop**

1.1 Selecteren methoden, technieken en tools.

- De student zal na het oriënteren op de Cloud omgevingen en Web services beschrijven welke methodes en ontwikkeltools worden gebruikt en waarom hij voor deze heeft gekozen.

1.3 Selecteren van standaardsoftware.

- Op basis van de onderzoeksrapporten uit de oriëntatie fase zal de student de juiste standaardsoftware kiezen. De keuze is afhankelijk van de ondersteuning die de mogelijke Cloud omgeving heeft en wat de web services zelf kunnen ondersteunen.

3.2 Ontwerpen systeemdeel

- De beschrijving van het de prototype applicatie komt terug in de te ontwikkelen API omschrijving. In dit document worden de functionaliteiten en het gedrag duidelijk beschreven.

3.3 Bouwen applicatie

- Op de server zal de student een prototype applicatie ontwikkelen die gebruik kan maken van gevonden web services.

Bijlage #2: API omschrijvingen

1. FontoXML

1.1 Request Object

FontoXML stuurt JsonML objecten naar de server. De server muteert deze data verder voor gebruik voor de webservices. JsonML is ontwikkeld om via JSON, XML structuren op te kunnen sturen.

1.2 Response Objecten

1.2.1 WizeNoze Metric Response Object

Het WizeNoze Metric Response Object geeft metric resultaten van WizeNoze weer. Er zijn 3 types die teruggestuurd worden door WizeNoze.

- GradeLevelMetric
- CoarseGradeLevelMetric
- VocabularySizeMetric

Het object dat FontoXML opgestuurd zal krijgen staat in figuur 44.

```
{
        CorrolationId = "0",
        ElementName = "p",
        Occurence = 14,
        StartPointer = 0,
        EndPointer = 13,
        Type = "Grade Level Metric",
        Data = "Groep 8",
        TextSuitability = "good",
        High = "Voortgezet Onderwijs",
        Low = "Groep 7"
}
```

Figuur 45: WizeNoze Metric Response Object.

1.2.2 WizeNoze ComplexTerm Response Object

Het WizeNoze ComplexTerm Response Object geeft moeilijke termen terug en per moeilijke term een list met mogelijke vervangingen voor dit woord. Het is een mogelijkheid dat WizeNoze geen suggesties heeft voor het woord en een lege lijst terug zal sturen.

Het object dat FontoXML opgestuurd zal krijgen staat in figuur 45.

```
{
        CorrolationId = "0",
        ElementName = "p",
        Occurence = 14,
        StartPointer = 0,
        EndPointer = 13,
        Type ="Difficult Term",
        Data = "Zwolle",
        SuggestionList =
        [
                "overdreven",
                "schreeuwerig",
                "hoogdravend",
                "overdadig",
                "bombastisch",
                "retorisch",
                "pathetisch"
        ]
}
```

Figuur 46: WizeNoze ComplexTerm Response Object

1.2.3 Acrolinx Flag Response Object

Het Acrolinx Flag Response Object geeft extra informatie over vijf mogelijke onderdelen. Quality, Spelling, Grammar, Style en Term. Het object moet geïnterpreteerd worden aan de hand van het type dat wordt meegegeven.

Het object dat FontoXML opgestuurd zal krijgen ziet er als volgt uit:

```
{
	CorrolationId = "0",
	ElementName = "p",
	Occurence = 14,
	StartPointer = 0,
	EndPointer = 28,
	Type ="GRAMMAR",
	Data = "Twenty-five gigabit Ethernet",
	Sentence = "Twenty-five gigabit Ethernet is growing in popularity.",
	ShortHelp = "Use singular nouns with singular articles and numbers,
such as a or this. Use plural nouns with plural articles and numbers, such
as several or these.",
	SuggestionList =
	[
	]
}
```

Figuur 47: Acrolinx Flag Response Object

1.2.4 Acrolinx Score Response Object

Het Acrolinx Score Response Object geeft scores over 5 mogelijke onderdelen. Quality, Spelling, Grammar, Style en Term. Het object moet geïnterpreteerd worden aan de hand van het type dat wordt meegegeven. De score geldt altijd voor het gehele opgestuurde request.

Het object dat FontoXML opgestuurd zal krijgen staat in figuur 47.

```
{
	CorrolationId = "0",
	ElementName = "document",
	StartPointer = 0,
	EndPointer = 85012,
	Type = "GRAMMAR",
	Data = 77,
	YellowThreshold = 30,
	GreenThreshold = 70
}
```

Figuur 48: Acrolinx Score Response Object.

2 WizeNoze

2.1 PostRequest Object

Om objecten op te sturen naar de WizeNoze webservice, moet er een JSON object in de header van de request meegestuurd worden. Dit object ziet er als volgt uit. De request zelf is een REST request en wacht daarom op een response.

```
{
       Language = "nl",
       Content = [ "Example <coupleAssociate> Maak koppels van de Nederlanse
provincies en hun hoofdsteden.   Groningen     Groningen     Friesland
Leeuwarden     Drenthe     Assen     Overijssel     Zwolle     Flevoland
Lelystad     Gelderland     Arnhem     Utrecht     Utrecht     Noord-Holland
Haarlem     Zuid-Holland     Den Haag     Zeeland     Middelburg     Noord-Brabant
's-Hertogenbosch     Limburg     Maastricht   Eindhoven Amsterdam Rotterdam Goes
Heerenveen Venlo Nijmegen In één keer goed! Goed gedaan! Dit zijn de juiste
koppels. Goed, maar misschien in het verstandig nogmaals de theorie te bestuderen.
Helaas, probeer het nogmaals.  Dit is niet het juiste antwoord. " ]
}
```

Figuur 49: PostRequest Object

2.2 PostResponse Object

Het response object bevat alle informatie die WizeNoze heeft gevonden over de opgestuurde data. De informatie die teruggestuurd wordt naar de cliënt zijn terug te vinden in de metrics array, Paragraph.metrics array en de annotatedHtml variabele.


```
{
    "document": {
        "status": "success",
        "readingLevelBracketId": 5,
        "readingLevelBracketProbabilities": [
            {
                "readingLevelBracketId": 1,
                "confidence": 0.14716171422686342
            },
            {
                "readingLevelBracketId": 2,
                "confidence": 0.15153766597022167
            },
            {
                "readingLevelBracketId": 3,
                "confidence": 0.15470994533958324
            },
            {
                "readingLevelBracketId": 4,
                "confidence": 0.1717024488140749
            },
            {
                "readingLevelBracketId": 5,
                "confidence": 0.18751879058805807
            },
            {
                "readingLevelBracketId": 6,
                "confidence": 0.18736943506119882
            }
        ],
```

```
    "metrics": [
        {
            "name": "AVI 2006",
            "value": "Plus",
            "low": "Plus",
            "high": "Plus",
            "textSuitability": "unknown"
        },
        {
            "name": "Grade Level Metric",
            "value": {
                "label": "groep8",
                "name": "Groep 8"
            },
            "low": {
                "label": "groep7",
                "name": "Groep 7"
            },
            "high": {
                "label": "groep9",
                "name": "Voortgezet Onderwijs"
            },
            "textSuitability": "good"
        },
        {
            "name": "Coarse Grade Level Metric",
            "value": {
                "label": "groep90",
                "name": "Voortgezet Onderwijs"
            },
            "low": {
                "label": "groep90",
                "name": "Voortgezet Onderwijs"
            },
            "high": {
                "label": "groep90",
                "name": "Voortgezet Onderwijs"
            },
            "textSuitability": "hard"
        },
        {
            "name": "Vocabulary Size Metric",
            "value": 12338,
            "low": 9739,
            "high": 17002,
            "textSuitability": "easy"
        }
    ],
    "textSuitability": null,
    "annotatedText": null,
    "annotatedHtml": null
},
"paragraphs": [
    {
        "status": "success",
        "readingLevelBracketId": 5,
        "readingLevelBracketProbabilities": [
            {
                "readingLevelBracketId": 1,
```

```
            "confidence": 0.14716171422686342
        },
        {
            "readingLevelBracketId": 2,
            "confidence": 0.15153766597022167
        },
        {
            "readingLevelBracketId": 3,
            "confidence": 0.15470994533958324
        },
        {
            "readingLevelBracketId": 4,
            "confidence": 0.1717024488140749
        },
        {
            "readingLevelBracketId": 5,
            "confidence": 0.18751879058805807
        },
        {
            "readingLevelBracketId": 6,
            "confidence": 0.18736943506119882
        }
    ],
    "metrics": [
        {
            "name": "AVI 2006",
            "value": "Plus",
            "low": "Plus",
            "high": "Plus",
            "textSuitability": "unknown"
        },
        {
            "name": "Grade Level Metric",
            "value": {
                "label": "groep8",
                "name": "Groep 8"
            },
            "low": {
                "label": "groep7",
                "name": "Groep 7"
            },
            "high": {
                "label": "groep9",
                "name": "Voortgezet Onderwijs"
            },
            "textSuitability": "good"
        },
        {
            "name": "Coarse Grade Level Metric",
            "value": {
                "label": "groep90",
                "name": "Voortgezet Onderwijs"
            },
            "low": {
                "label": "groep90",
                "name": "Voortgezet Onderwijs"
            },
            "high": {
                "label": "groep90",
```

```
                        "name": "Voortgezet Onderwijs"
                    },
                    "textSuitability": "hard"
                },
                {
                    "name": "Vocabulary Size Metric",
                    "value": 12338,
                    "low": 9739,
                    "high": 17002,
                    "textSuitability": "easy"
                }
            ],
            "textSuitability": "good",
            "annotatedText": "[Example](#\"@#@Uncommon/difficult term@#@\")
<[coupleAssociate](#\"@#@Uncommon/difficult term@#@\")>[ [<i class='fa fa-
asterisk'></i>]](# \"@#@Uncommon/difficult term@#@: >\") Maak koppels van de
[Nederlanse](#\"@#@Uncommon/difficult term@#@\") provincies en hun
[hoofdsteden](#\"@#@Uncommon/difficult term@#@\").   Groningen     Groningen
Friesland     [Leeuwarden](#\"@#@Uncommon/difficult term@#@\")
[Drenthe](#\"@#@Uncommon/difficult term@#@\")     Assen
[Overijssel](#\"@#@Uncommon/difficult term@#@\")
[Zwolle](#\"@#@Uncommon/difficult term@#@ @#@Suggestions@#@: [overdreven,
schreeuwerig]\")     [Flevoland](#\"@#@Uncommon/difficult term@#@\")
[Lelystad](#\"@#@Uncommon/difficult term@#@\")
[Gelderland](#\"@#@Uncommon/difficult term@#@\")     Arnhem     Utrecht     Utrecht
Noord-Holland     [Haarlem](#\"@#@Uncommon/difficult term@#@\")     Zuid-Holland
Den Haag     [Zeeland](#\"@#@Uncommon/difficult term@#@\")
[Middelburg](#\"@#@Uncommon/difficult term@#@\")     Noord-
[Brabant](#\"@#@Uncommon/difficult term@#@\")     '[s](#\"@#@Uncommon/difficult
term@#@\")-[Hertogenbosch](#\"@#@Uncommon/difficult term@#@\")     Limburg
[Maastricht](#\"@#@Uncommon/difficult term@#@\")
[Eindhoven](#\"@#@Uncommon/difficult term@#@\") Amsterdam Rotterdam
[Goes](#\"@#@Uncommon/difficult term@#@\") [Heerenveen](#\"@#@Uncommon/difficult
term@#@\") [Venlo](#\"@#@Uncommon/difficult term@#@\") Nijmegen In één keer goed![
[<i class='fa fa-asterisk'></i>]](# \"@#@Long sentence@#@: Groningen     Groningen
Friesland     Leeuwarden     Drenthe     Assen     Overijssel     Zwolle
Flevoland     Lelystad     Gelderland     Arnhem     Utrecht     Utrecht     Noord-
Holland     Haarlem     Zuid-Holland     Den Haag     Zeeland     Middelburg
Noord-Brabant     's-Hertogenbosch     Limburg     Maastricht   Eindhoven Amsterdam
Rotterdam Goes Heerenveen Venlo Nijmegen In één keer goed!\") Goed gedaan! Dit zijn
de juiste koppels. Goed, maar misschien in het verstandig nogmaals de theorie te
bestuderen. Helaas, probeer het nogmaals.  Dit is niet het juiste antwoord. ",
            "annotatedHtml": "<span data-issue-description='Uncommon/difficult
term'>Example</span> <<span data-issue-description='Uncommon/difficult
term'>coupleAssociate<span data-issue-description='Uncommon/difficult
term'></span>></span> Maak koppels van de <span data-issue-
description='Uncommon/difficult term'>Nederlanse</span> provincies en hun <span
data-issue-description='Uncommon/difficult term'>hoofdsteden</span>.   <span data-
issue-description='Long sentence'>Groningen     Groningen     Friesland     <span
data-issue-description='Uncommon/difficult term'>Leeuwarden</span>     <span data-
issue-description='Uncommon/difficult term'>Drenthe</span>     Assen     <span
data-issue-description='Uncommon/difficult term'>Overijssel</span>     <span data-
issue-description='Uncommon/difficult term' data-lemma='zwol' data-suitable-
synonyms='overdreven,schreeuwerig' data-additional-
synonyms='hoogdravend,overdadig,bombastisch,retorisch,pathetisch'>Zwolle</span>
<span data-issue-description='Uncommon/difficult term'>Flevoland</span>     <span
data-issue-description='Uncommon/difficult term'>Lelystad</span>     <span data-
issue-description='Uncommon/difficult term'>Gelderland</span>     Arnhem
Utrecht     Utrecht     Noord-Holland     <span data-issue-
```

```
description='Uncommon/difficult term'>Haarlem</span>     Zuid-Holland     Den Haag
<span data-issue-description='Uncommon/difficult term'>Zeeland</span>     <span
data-issue-description='Uncommon/difficult term'>Middelburg</span>     Noord-<span
data-issue-description='Uncommon/difficult term'>Brabant</span>     '<span data-
issue-description='Uncommon/difficult term'>s</span>-<span data-issue-
description='Uncommon/difficult term'>Hertogenbosch</span>     Limburg     <span
data-issue-description='Uncommon/difficult term'>Maastricht</span>   <span data-
issue-description='Uncommon/difficult term'>Eindhoven</span> Amsterdam Rotterdam
<span data-issue-description='Uncommon/difficult term'>Goes</span> <span data-
issue-description='Uncommon/difficult term'>Heerenveen</span> <span data-issue-
description='Uncommon/difficult term'>Venlo</span> Nijmegen In één keer
goed!</span> Goed gedaan! Dit zijn de juiste koppels. Goed, maar misschien in het
verstandig nogmaals de theorie te bestuderen. Helaas, probeer het nogmaals.  Dit is
niet het juiste antwoord. <div data-dictionary><div data-dictionary-
entry='zwol'><div data-synonyms-suitable><div data-synonyms-
type='placeholder'><span data-synonym>overdreven</span><span data-
synonym>schreeuwerig</span></div></div><div data-synonyms-additional><div data-
synonyms-type='placeholder'><span data-synonym>hoogdravend</span><span data-
synonym>overdadig</span><span data-synonym>bombastisch</span><span data-
synonym>retorisch</span><span data-
synonym>pathetisch</span></div></div></div></div>"
        }
    ]
}
```

3 Acrolinx

3.1 PostRequest Object

Acrolinx maakt gebruik van long polling. De eerste request is een PostRequest waarmee alle data naar de webservice wordt gestuurd. De request heeft in de header een JSON object die er als volgt uitziet.


```
{
	"documentId":"3548fdce-29fb-44ad-8628-9fd48fd4f782",
	"inputFormat":"HTML",
	"base64EncodedGzipped":false,
	"requestDescription":{},
	"text":"Example <coupleAssociate> Maak koppels van de Nederlanse provincies
en hun hoofdsteden.  Groningen  Groningen  Friesland  Leeuwarden
Drenthe  Assen  Overijssel  Zwolle  Flevoland  Lelystad
Gelderland  Arnhem  Utrecht  Utrecht  Noord-Holland  Haarlem
Zuid-Holland  Den Haag  Zeeland  Middelburg  Noord-Brabant  's-
Hertogenbosch  Limburg  Maastricht  Eindhoven Amsterdam Rotterdam Goes
Heerenveen Venlo Nijmegen In één keer goed! Goed gedaan! Dit zijn de juiste
koppels. Goed, maar misschien in het verstandig nogmaals de theorie te bestuderen.
Helaas, probeer het nogmaals.  Dit is niet het juiste antwoord. "
}
```


Figuur 50: PostRequest Object

3.2 PostResponse Object

Als response object krijgt de server het object terug dat te zien is in figuur 50. Aan de hand van het checkId kan de data met een get request opgehaald worden. de etaMS moet aan duiden hoelang de controle ongeveer gaat duren. Deze functionaliteit werkt op dit moment niet.


```
{
	"checkId" : "stringHash",
	"etaMs" : 100
}
```


Figuur 51: PostResponse Object

3.3 GetRequest Object

Voor de get request hoeft geen JSON object gebruikt te worden. De opgestuurde data kan via een URL opgehaald worden van de Acrolinx services.

3.4 GetResponse Object

Het response object bevat alle informatie die Acrolinx heeft gevonden over de opgestuurde data. De informatie die teruggestuurd wordt naar de cliënt is terug te vinden in de flags array en de scores array.


```
{
    "version": null,
    "document": null,
    "sentences": [
        {
            "id": 0,
            "textUnit": null,
            "range": [
                0,
                54
            ],
            "content": "Twenty-five gigabit Ethernet is growing in popularity.",
```

```
            "flags": [
                {
                    "type": "GRAMMAR",
                    "internalName": "np_number_agreement",
                    "title": "number agreement",
                    "shortHelp": "Use singular nouns with singular articles and
numbers, such as a or this. Use plural nouns with plural articles and numbers, such
as several or these.",
                    "help": null,
                    "key": "np_number_agreement",
                    "groupId": null,
                    "batchReplaceable": null,
                    "readonly": null,
                    "matches": [
                        {
                            "content": "Twenty-five",
                            "range": [
                                0,
                                11
                            ]
                        },
                        {
                            "content": "gigabit Ethernet",
                            "range": [
                                12,
                                28
                            ]
                        }
                    ],
                    "suggestions": [],
                    "shortName": "number agreement",
                    "ruleId": "EN20130780858SL",
                    "matchRanking": null,
                    "term": null,
                    "keyword": null,
                    "isDuplicate": null,
                    "isUnknownWord": null,
                    "termCandidate": null
                }
            ]
        },
        {
            "id": 1,
            "textUnit": null,
            "range": [
                55,
                91
            ],
            "content": "Do you think it will replace 10 GbE?",
            "flags": null
        },
        {
            "id": 2,
            "textUnit": null,
            "range": [
                92,
                156
            ],
```

```
            "content": "Are there security implications of 25 GbE we should be
aware of?",
            "flags": null
        },
        {
            "id": 3,
            "textUnit": null,
            "range": [
                158,
                228
            ],
            "content": "When it comes to networking, the faster the better -- at
least for me.",
            "flags": null
        },
        {
            "id": 4,
            "textUnit": null,
            "range": [
                229,
                295
            ],
            "content": "That way I can get my security assessments performed more
quickly.",
            "flags": null
        },
        {
            "id": 5,
            "textUnit": null,
            "range": [
                296,
                398
            ],
            "content": "This new 25 gigabit version of Ethernet will most certainly
help -- if it ever makes it to mainstream.",
            "flags": null
        },
        {
            "id": 6,
            "textUnit": null,
            "range": [
                399,
                627
            ],
            "content": "I'm just now seeing many enterprises complete their
rollouts of Gigabit Ethernet, so I suspect we're still years away from seeing 25
GbE (or the formal standard, 40 GbE and 100 GbE) outside of data centers and in the
mainstream.",
            "flags": [
                {
                    "type": "STYLE",
                    "internalName": "sentence_too_long",
                    "title": "sentence too long: 40 words",
                    "shortHelp": "This sentence has more than 25 words. Long
sentences are difficult to read and to translate. Use short sentences instead.",
                    "help": null,
                    "key": "sentence_too_long",
                    "groupId": null,
                    "batchReplaceable": null,
```

```
                    "readonly": null,
                    "matches": [
                        {
                            "content": "I",
                            "range": [
                                399,
                                400
                            ]
                        },
                        {
                            "content": "mainstream",
                            "range": [
                                616,
                                626
                            ]
                        }
                    ],
                    "suggestions": [],
                    "shortName": "sentence too long: 40 words",
                    "ruleId": null,
                    "matchRanking": null,
                    "term": null,
                    "keyword": null,
                    "isDuplicate": null,
                    "isUnknownWord": null,
                    "termCandidate": null
                },
                {
                    "type": "GRAMMAR",
                    "internalName": "where_were_confusion",
                    "title": "where/were confusion",
                    "shortHelp": "Use where to refer to a location.Use were as the
past tense plural form of the verb to be.Use we're as a contraction of we are.",
                    "help": null,
                    "key": "where_were_confusion",
                    "groupId": null,
                    "batchReplaceable": null,
                    "readonly": null,
                    "matches": [
                        {
                            "content": "we",
                            "range": [
                                494,
                                496
                            ]
                        },
                        {
                            "content": "'re",
                            "range": [
                                496,
                                499
                            ]
                        }
                    ],
                    "suggestions": [],
                    "shortName": "where/were confusion",
                    "ruleId": "EN20063241425MS",
                    "matchRanking": null,
                    "term": null,
```

```
                    "keyword": null,
                    "isDuplicate": null,
                    "isUnknownWord": null,
                    "termCandidate": null
                }
            ]
        },
        {
            "id": 7,
            "textUnit": null,
            "range": [
                629,
                728
            ],
            "content": "The practical side of me sees 25 GbE as having the same
security issues as all of its predecessors:",
            "flags": null
        },
        {
            "id": 8,
            "textUnit": null,
            "range": [
                730,
                926
            ],
            "content": "Default configurations -- such as passwords and open Web
interfaces -- can be manipulated and subsequently have to be locked down via
security standards and policies that have been around forever.",
            "flags": [
                {
                    "type": "STYLE",
                    "internalName": "sentence_too_long",
                    "title": "sentence too long: 29 words",
                    "shortHelp": "This sentence has more than 25 words. Long
sentences are difficult to read and to translate. Use short sentences instead.",
                    "help": null,
                    "key": "sentence_too_long",
                    "groupId": null,
                    "batchReplaceable": null,
                    "readonly": null,
                    "matches": [
                        {
                            "content": "Default",
                            "range": [
                                730,
                                737
                            ]
                        },
                        {
                            "content": "forever",
                            "range": [
                                918,
                                925
                            ]
                        }
                    ],
                    "suggestions": [],
                    "shortName": "sentence too long: 29 words",
                    "ruleId": null,
```

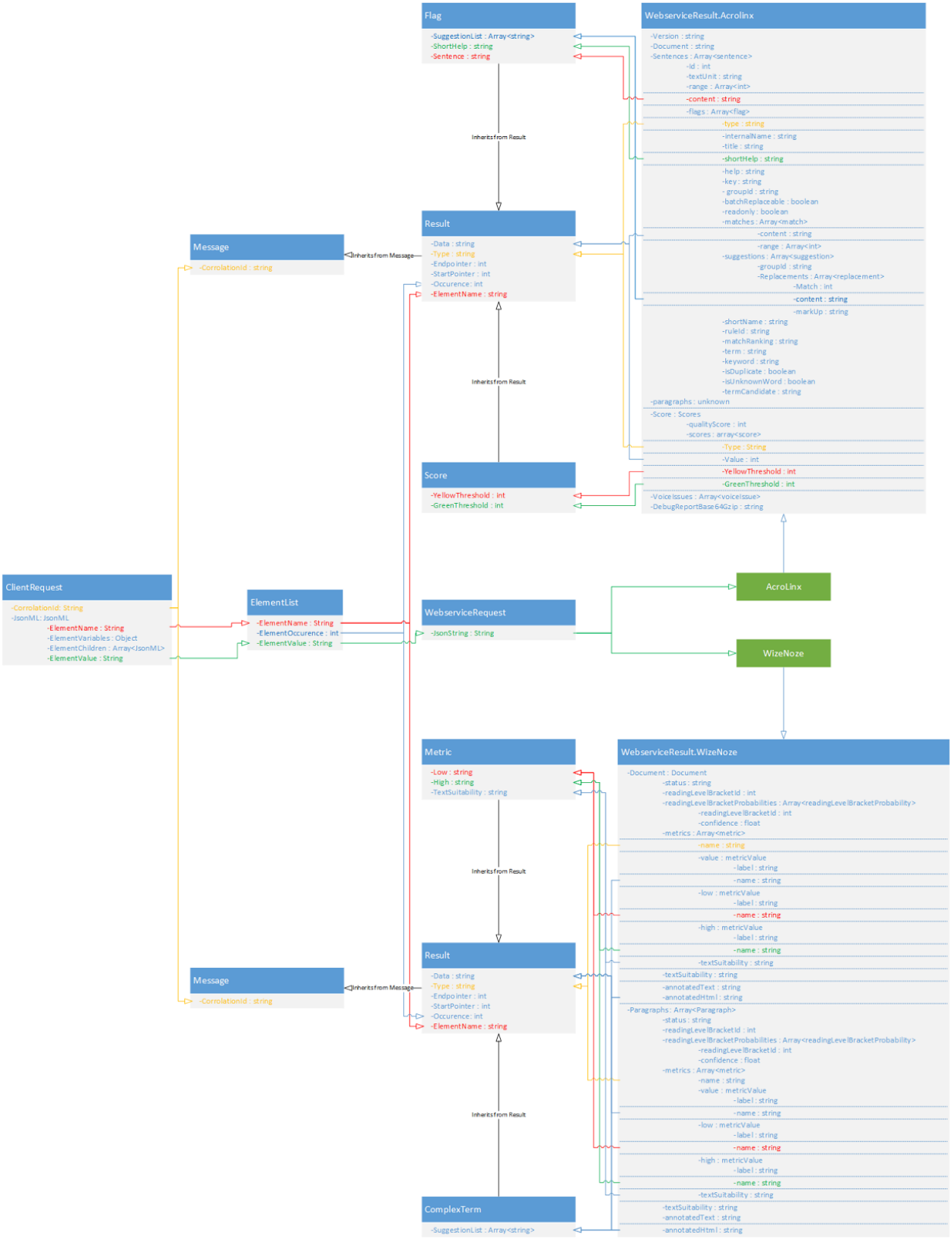
```
                    "matchRanking": null,
                    "term": null,
                    "keyword": null,
                    "isDuplicate": null,
                    "isUnknownWord": null,
                    "termCandidate": null
                }
            ]
        },
        {
            "id": 9,
            "textUnit": null,
            "range": [
                927,
                1071
            ],
            "content": "Unencrypted traffic flowing across the wire that can be
easily intercepted by anyone on your network with some free tools and bit of ill
intent.",
            "flags": null
        },
        {
            "id": 10,
            "textUnit": null,
            "range": [
                1072,
                1254
            ],
            "content": "Third-party security controls and network management
systems such as security information and event management, intrusion prevention and
network analyzers that may not be compatible.",
            "flags": null
        },
        {
            "id": 11,
            "textUnit": null,
            "range": [
                1255,
                1374
            ],
            "content": "The more things change with information technology, the
more we're seeing how important the security basics really are.",
            "flags": null
        },
        {
            "id": 12,
            "textUnit": null,
            "range": [
                1375,
                1563
            ],
            "content": "While the blazing speeds will offer benefits, such as being
able to run network assessments at a quicker pace, the same security challenges are
going to occur that much faster with 25 GbE.",
            "flags": [
                {
                    "type": "STYLE",
                    "internalName": "sentence_too_long",
                    "title": "sentence too long: 33 words",
```

```
                    "shortHelp": "This sentence has more than 25 words. Long
sentences are difficult to read and to translate. Use short sentences instead.",
                    "help": null,
                    "key": "sentence_too_long",
                    "groupId": null,
                    "batchReplaceable": null,
                    "readonly": null,
                    "matches": [
                        {
                            "content": "While",
                            "range": [
                                1375,
                                1380
                            ]
                        },
                        {
                            "content": "GbE",
                            "range": [
                                1559,
                                1562
                            ]
                        }
                    ],
                    "suggestions": [],
                    "shortName": "sentence too long: 33 words",
                    "ruleId": null,
                    "matchRanking": null,
                    "term": null,
                    "keyword": null,
                    "isDuplicate": null,
                    "isUnknownWord": null,
                    "termCandidate": null
                },
                {
                    "type": "GRAMMAR",
                    "internalName": "missing_word",
                    "title": "missing word",
                    "shortHelp": "There seems to be a missing word here. For
example, in the phrase according the requirements, the word to is missing.",
                    "help": null,
                    "key": "missing_word",
                    "groupId": null,
                    "batchReplaceable": null,
                    "readonly": null,
                    "matches": [
                        {
                            "content": "that",
                            "range": [
                                1534,
                                1538
                            ]
                        },
                        {
                            "content": "much",
                            "range": [
                                1539,
                                1543
                            ]
                        },
```

```
                        {
                            "content": "faster",
                            "range": [
                                1544,
                                1550
                            ]
                        },
                        {
                            "content": "with",
                            "range": [
                                1551,
                                1555
                            ]
                        },
                        {
                            "content": "25",
                            "range": [
                                1556,
                                1558
                            ]
                        },
                        {
                            "content": "GbE",
                            "range": [
                                1559,
                                1562
                            ]
                        },
                        {
                            "content": ".",
                            "range": [
                                1562,
                                1563
                            ]
                        }
                    ],
                    "suggestions": [],
                    "shortName": "missing word",
                    "ruleId": "EN20112381633SL",
                    "matchRanking": null,
                    "term": null,
                    "keyword": null,
                    "isDuplicate": null,
                    "isUnknownWord": null,
                    "termCandidate": null
                }
            ]
        },
        {
            "id": 13,
            "textUnit": null,
            "range": [
                1564,
                1630
            ],
            "content": "Are you -- and your systems -- ready to drink from that
fire hose?",
            "flags": null
        }
```

```
    ],
    "paragraphs": null,
    "scores": {
        "qualityScore": 77,
        "scores": [
            {
                "type": "QUALITY",
                "value": 77,
                "yellowThreshold": 30,
                "greenThreshold": 70
            },
            {
                "type": "SPELLING",
                "value": 100,
                "yellowThreshold": 30,
                "greenThreshold": 70
            },
            {
                "type": "GRAMMAR",
                "value": 54,
                "yellowThreshold": 30,
                "greenThreshold": 70
            },
            {
                "type": "STYLE",
                "value": 54,
                "yellowThreshold": 30,
                "greenThreshold": 70
            },
            {
                "type": "TERM",
                "value": 100,
                "yellowThreshold": 30,
                "greenThreshold": 70
            }
        ]
    },
    "voiceIssues": [],
    "debugReportBase64Gzip": null
}
```

Bijlage #3: Mapping Schema.



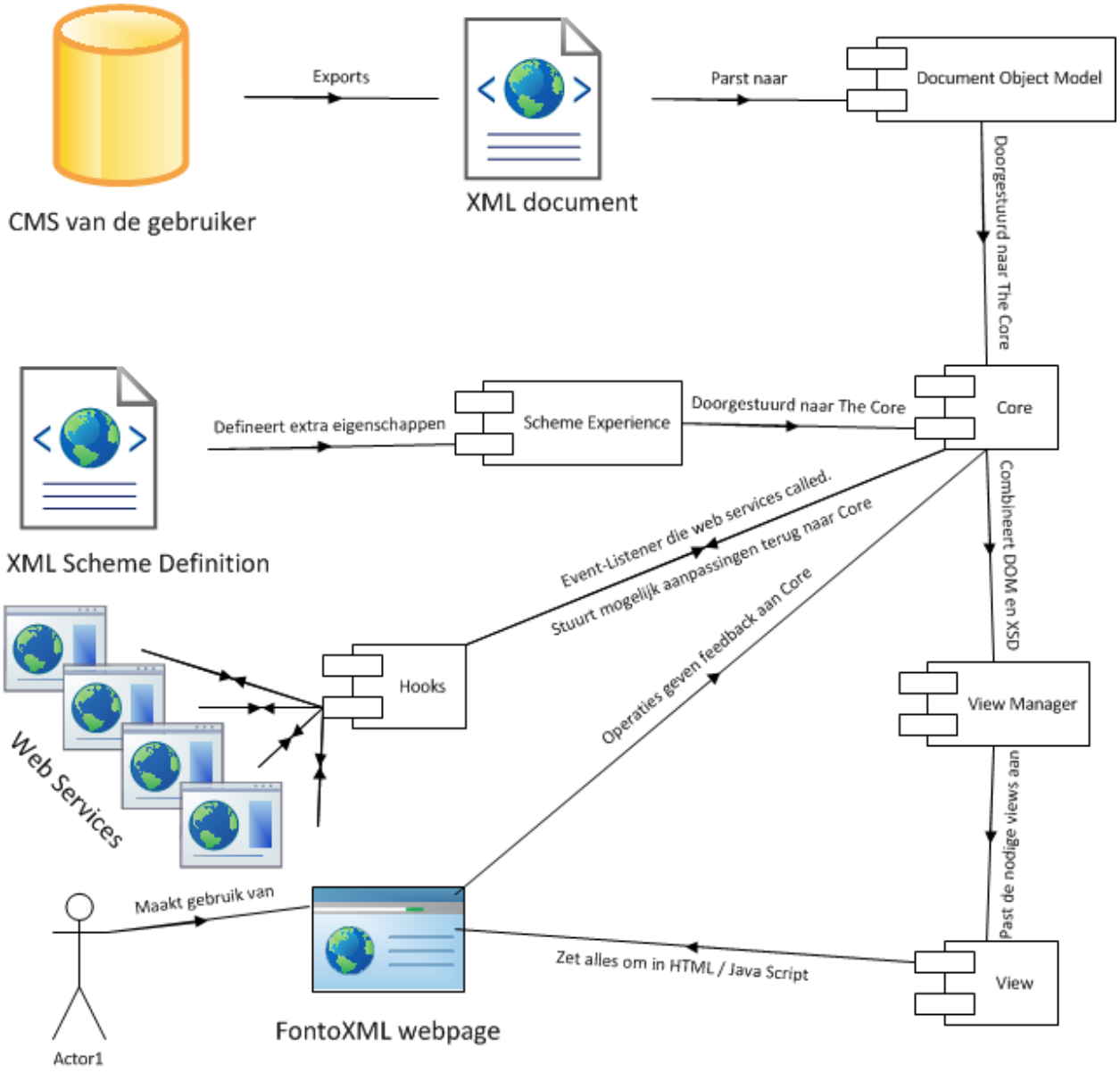
Figuur 52: Mapping schema

Bijlage #4: Beschrijving Huidige Situatie

Inleiding

Dit document is geschreven om inzicht te geven in de huidige situatie van het systeem. Door middel van modellen moet de communicatie, in het systeem en met andere systemen duidelijk weergegeven worden. Er wordt beschreven met welke web services het systeem communiceert en wat de functionaliteit van de web services zijn.

1. Model huidige situatie



Figuur 53: Model huidige situatie.

2. Beschrijving elementen

2.1 CMS van de gebruiker.

Dit is het content management system van de klant. Dit kunnen verschillende systemen zijn zoals, maar niet beperkt tot, Google Drive, Sharepoint of Onedrive. Deze systemen kunnen van documenten XML varianten beschikbaar stellen waar FontoXML verder mee gaat werken.

2.2 XML document

Dit is het XML document dat uit het CMS van de klant wordt geëxporteerd met de standaard variabele gedefinieerd.

2.3 Document Object Model

Het Document Object Model (DOM) is een gestructureerde representatie van het XML document. Door middel van de DOM kan de core informatie uit de XML document halen en combineren met de XSD.

2.4 XML Scheme Definition

De XML Scheme Definition (XSD) definieert in een schema de inhoud voor XML documenten.

2.5 Scheme Experience

De scheme experience beschrijft alle extra attributen die gebruikt worden door FontoXML maar niet terug komen in de XSD.

2.6 Core

De core weet welk DOM bij welk XSD hoort. De core combineert deze en stuurt deze door naar de View Manager die het verder afhandelt.

2.7 View Manager

Iedere keer als er aanpassingen worden gemaakt moeten de HTML views mogelijk weer aangepast worden. De manager weet welke veranderingen op welke view effect hebben en zorgt ervoor dat deze views geüpdate worden.

2.8 View

De HTML views die real-time aangepast moeten kunnen worden.

2.9 FontoXML webpage

De web applicatie die beschikbaar is voor de klant. Hierop werkt de gebruiker en via deze pagina krijgt de gebruiker eventueel feedback die de Hooks geven.

2.10 Actor1

Dit zijn de medewerkers van de klant die gebruik maken van FontoXML. Zij zijn degene die aanpassingen maken in de documenten en operaties uitvoeren op het systeem.

2.11 Hooks

Hooks zijn eventListeners. Deze luistert naar evenementen waar de klant wilt dat gecontroleerd wordt. Deze listeners worden bijna bij iedere keypress geactiveerd. Dit omdat FontoXML real-time feedback moet kunnen geven aan de gebruiker. Iedere listeners heeft de mogelijkheid om een webservice te activeren die vervolgens aan de slag gaat en feedback gaat geven.

2.12 Webservices

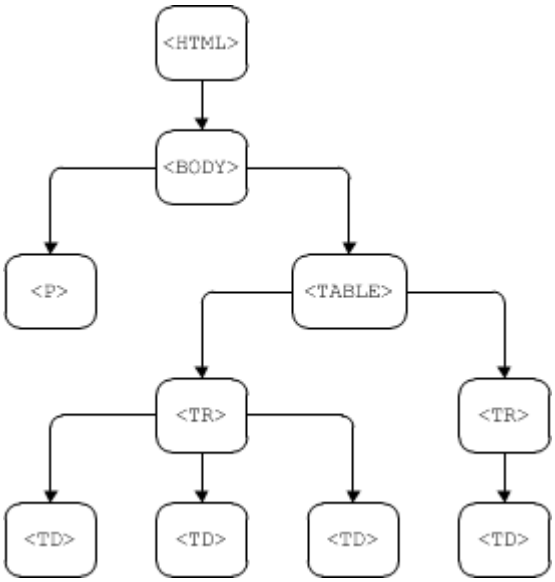
De webservices voeren de controles uit en geven feedback terug aan de hooks. De hooks reageren hier verder op en nemen de nodige acties die in real time door de gebruiker als feedback worden gezien.

3. Webservices

In de huidige situatie kunnen er al verschillende webservices gebruikt worden in FontoXML door klanten. In dit onderdeel worden de beschikbare webservices beschreven. Een klant hoeft niet alle webservices beschikbaar te hebben. Het kan zijn dat een klant er voor heeft gekozen om maar een webservice te laten implementeren of zelfs geen enkele.

3.1 Spell checker webservice

De spell checker webservice kan, per document, op aanvraag geactiveerd worden. Nadat de functie is geactiveerd wordt de huidige inhoud van het document naar de service ter controle gestuurd. En vervolgens wordt de functie om de seconde uitgevoerd. Bij eerste aanroep stuurt de functie alle informatie door die in de hoogste parent node te vinden is. In het geval van het figuur is dat `<HTML>` Daarna wordt er per aanpassing de parent node van de data doorgestuurd. De node wordt vertaald naar JsonML en doorgestuurd naar de webservice. De webservice controleert ieder woord op spelling en wordt daarna teruggestuurd. De webservice stuurt een array list terug waaruit te herleiden is in welke node de spelfout staat en waar de spelfout staat in de node. Aan de hand hiervan kan er feedback gegeven worden aan de gebruiker door er een rode lijn onder te tekenen. Een van de problemen die deze service heeft is dat hij asynchroon werkt (zoals gewenst door Liones). Dankzij de asynchrone werking kunnen er wijzigingen plaatsvinden in de node tussen het versturen en ontvangen van de request, waardoor de locatie van de spelfout veranderd. Hierdoor kan er op een verkeerde plek een fout aangegeven worden.



Figuur 54: Voorbeeld van een node tree.

3.2 Entity Recognition

Deze service zorgt er voor dat er entiteiten zoals, namen, wetgevingen, artikelnummers en specifieke termen worden herkent. Zodra de entiteiten zijn herkend worden er links voorgesteld naar lijsten of overzichten. Hierdoor kan de auteur content maken met relevante links en meta-data zonder er zelf veel tijd aan te moeten besteden.

3.3 Search Engine Optimization (SEO) check

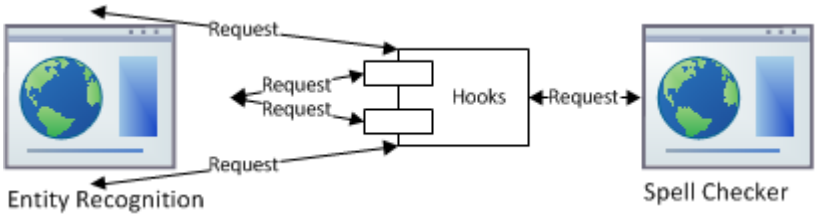
De SEO Check helpt de auteur zijn tekst te optimaliseren voor zoek machines zoals google. Het controleert de tekst op het moment dat het wordt geschreven. De auteur kiest zelf termen waarop de SEO score gebaseerd moet worden. Terwijl de tekst geschreven wordt krijgt de auteur feedback over hoe hij de tekst kan verbeteren om een hogere SEO score te krijgen.

4. Service Model

Het huidige service model van FontoXML is Software as a Service (SaaS). De gebruiker kan via de webbrowser bij de applicatie en kan vanuit de browser direct communiceren met de webservices. Dit is waar Liones nu problemen ziet. Omdat de applicatie in de webbrowser draait worden ook de resources gebruikt van de browser. De webservices maken intensief gebruik van deze resources. Omdat het verdienmodel van FontoXML gedeeltelijk op deze webservices zijn gebaseerd zullen deze uitgebreid worden met meer webservices. Hoe meer webservices hoe meer resources van de browser wordt gevraagd. Dit houdt in dat deze steeds voller gaat lopen. De te gebruiken resources in een browser zijn al beperkt en daarom wilt Liones kijken naar een andere manier waarop met webservices gecommuniceerd kan worden.

5. Communicatie tussen Web Services en FontoXML

Webservices worden in de applicatie aangeroepen via zogenoemde “Hooks”. Hooks zijn event listeners waarop gereageerd moet worden. Iedere hook krijgt een prioriteit en wordt op basis hiervan uitgevoerd. De hook blijft bestaan tot er vanuit de webservice reactie is. De hook zal op basis van deze reactie, acties uitvoeren.



Figuur 55: Communicatie tussen hooks en webservices.

Hooks worden op verschillende events aangemaakt. In de huidige situatie bestaan er twee extreme web services. De spellingchecker en de Entity recognition.

De spelling checker wordt alleen maar aangeroepen als de gebruiker hiervoor vraagt. De gebruiker activeert deze door middel van het drukken op de spelling controle knop. Dit is een extreme situatie omdat er maar één request wordt verstuurd met veel te controleren data. Door de grote hoeveelheid data blijft deze request vaker langer openstaan dan andere requests.

De Entity recognition is de andere extreme web service. Deze webservice controleert woorden en kijkt of hiervoor websitelinks beschikbaar zijn. Bijvoorbeeld als de gebruiker het woord Amsterdam intikt verandert de hook het woord en wordt er een link toegevoegd naar http://www.amsterdam.nl/. De Entity recognition reageert op iedere toets aanslag. Hierdoor komen extreem veel requests, met weinig data, naar de webservice. Het is duidelijk dat er veel requests worden gestuurd waarvan we weten dat het geen entiteit is. Als deze requests niet snel worden afgehandeld zullen deze opstapelen in de browser en steeds meer resources in beslag nemen. Ten nadeel van andere webservices.

Bijlage #5: Requirements

1. Inleiding

Om een duidelijk beeld te krijgen van wat de opdrachtgever verwacht van het nieuwe systeem zijn er een aantal requirements opgesteld. In dit document zijn alle opgestelde requirements te vinden en iedere requirement is verder uitgewerkt in SMART-tabellen. Aan het eind van het project zal dit document per requirement beschrijvingen krijgen waarin staat beschrijven of de requirement terug te vinden is in het systeem.

2. Aanpak van opstellen requirements

Iedere requirement krijgt de volgende meta data: ID, Requirement titel, eigenaar en prioriteit. Iedere requirement wordt verder uitgeschreven in een SMART-tabel. SMART is acroniem samengesteld uit de woorden: Specifiek, Meetbaar, Acceptabel, Relevant en Tijd Gebonden[3]. Ieder woord heeft zijn eigen mening en als deze op de juiste manier wordt toegepast op de gestelde eisen moet er een helder beeld komen van deze eis. De volgende vragen kunnen gesteld worden om het zo duidelijk mogelijk op te schrijven.

SMART-Tabel	
(S)pecifiek	- Wat wil ik behalen? - Waarom wil ik het behalen? - Wie is betrokken? - Waar speelt het probleem zich af?
(M)eetbaar	- Hoe vaak komt het voor? - Hoe weet ik dat het compleet is? - Zijn de indicatoren telbaar?
(A)cceptabel	- Hoe kan ik dit behalen? - Hoe realistisch is dit om te halen?
(R)elevant	- Moet ik hier tijd aan besteden? - Is dit het juiste moment om op te lossen? - Past dit bij onze doelen?
(T)ijd gebonden	- Wat is er te doen? - Wanneer moet het af zijn?

3. Requirements lijst

ID	Requirement Titel	Eigenaar	Prioriteit
1	De cliënt kan vanuit de browser via de centrale server communiceren met de webservices.	Bert Willems	Must Have
2	De server moet eenvoudig uit te breiden zijn voor nieuwe web services.	Bert Willems	Should Have

[3] http://en.wikipedia.org/wiki/SMART_criteria

3	Het prototype bevat Log functionaliteiten.	Bert Willems	Won't Have
4	Het prototype bevat autorisatie functionaliteiten op binnenkomende aanvragen van cliënten.	Bert Willems	Could Have
5	Het prototype bevat authenticatie functionaliteiten op binnenkomende verbindingen van cliënten.	Bert Willems	Could Have
6	Het prototype kan met meerdere Web Services tegelijk communiceren.	Bert Willems	Should Have
7	Het prototype kan asynchroon aanvragen van cliënten afhandelen.	Bert Willems	Must Have
8	De server omgeving kan binnenkomende aanvragen muteren en doorsturen naar web services.	Bert Willems	Must Have
9	De server stuurt de data van de web service altijd op dezelfde manier terug naar de cliënt.	Bert Willems	Could Have
10	Ontvangen aanvragen moeten kunnen worden gestopt als de aanvragen niet meer geldig zijn door veranderingen bij de cliënt.	Bert Willems	Could Have
11	De centrale server moet de data op de juist gemapte manier terug geven aan de cliënt.	Bert Willems	Must Have
12	Requests die naar de centrale server worden gestuurd moeten bijgehouden worden.	Bert Willems	Must Have
13	De centrale server ondersteunt meerdere cliënten tegelijkertijd.	Bert Willems	Must Have
14	De centrale server kan asynchroon responses terug sturen naar de cliënt.	Bert Willems	Must Have

4. SMART-tabellen

1 - De cliënt kan vanuit de browser via de centrale server communiceren met de webservices.	

(S)pecifiek	- Wat wil ik behalen?
	  - Add-ons in FontoXML moeten via de centrale server webservices kunnen aanroepen.
	- Waarom wil ik het behalen?
	  - Om de browser, waar FontoXML in draait, te ontlasten.
	- Wie is betrokken?
	  - Bert Willems
	- Waar speelt het probleem zich af?
	  - Tussen FontoXML en de centrale server.

(M)eetbaar	- Hoe vaak komt het voor?
	  - Iedere keer dat FontoXML gebruikt wilt maken van een Add-on zal er informatie naar de centrale server worden gestuurd.
	- Hoe weet ik dat het compleet is?
	  - De centrale server moet relevante informatie, uit de webservice, teruggeven voor de gebruikte add-on.
	- Zijn de indicatoren telbaar?
	  - Nee.

(A)cceptabel	- Hoe kan ik dit behalen?
	  - Ontwikkeling van de centrale server die aanvragen van FontoXML kan ontvangen en deze kan gebruiken om data naar de webservice te sturen.
	- Hoe realistisch is dit om te halen?
	  - Zeer realistisch. Dit is de belangrijkste requirement.

(R)elevant	- Moet ik hier tijd aan besteden?
	  - Ja, belangrijkste requirement.
	- Is dit het juiste moment om op te lossen?
	  - Ja, hiervoor is het project gestart.
	- Past dit bij onze doelen?
	  - Ja, deze requirement en het doel van het project liggen dicht bij elkaar.

(T)ijd gebonden	- Wat is er te doen?
	  - Communicatie opzetten tussen FontoXML en de centrale server en tussen de centrale server en de web service.
	- Wanneer moet het af zijn?
	  - Eerste iteratie.

2 - De server moet eenvoudig uit te breiden zijn voor nieuwe web services.	
(S)pecifiek	- Wat wil ik behalen? - Nieuwe webservices moeten eenvoudig toe te voegen zijn aan de centrale server.

	- Waarom wil ik het behalen?
	  - Om toekomstige ontwikkelingen eenvoudig toepasbaar te laten zijn.
	- Wie is betrokken?
	  - Bert Willems.
	- Waar speelt het probleem zich af?
	  - De centrale server.

(M)eetbaar	- Hoe vaak komt het voor?
	  - Als klanten vragen om add-ons die nog niet ondersteund kunnen worden door de centrale server.
	- Hoe weet ik dat het compleet is?
	  - Het toevoegen van een nieuwe webservice beïnvloedt geen of weinig andere delen van het systeem en kost relatief weinig tijd om te bouwen.
	- Zijn de indicatoren telbaar?
	  - Nee.

(A)cceptabel	- Hoe kan ik dit behalen?
	  - Door middel van het ontwikkelen van een interface.
	- Hoe realistisch is dit om te halen?
	  - Zeer realistisch.

(R)elevant	- Moet ik hier tijd aan besteden?
	  - Ja. De eenvoud van het toevoegen van webservices is belangrijke requirement die in de toekomst een grote rol gaat spelen.
	- Is dit het juiste moment om op te lossen?
	  - Ja. Vroeg rekening mee houdt bespaart later een hoop tijd om het om te zetten.
	- Past dit bij onze doelen?
	  - Ja. De webservices zijn een belangrijk onderdeel van de centrale server.

(T)ijd gebonden	- Wat is er te doen?
	  - Ontwikkelen van een interface voor de webservices en low coupling toepassen op de webservices.
	- Wanneer moet het af zijn?
	  - derde iteratie.

3 - Het prototype bevat Log functionaliteiten.	

(S)pecifiek	- Wat wil ik behalen?
	  - Log functionaliteiten zodat gebeurtenissen op de server terug te zien zijn.
	- Waarom wil ik het behalen?
	  - Om mogelijke foutmeldingen te kunnen traceren.
	- Wie is betrokken?
	  - Bert Willems.
	- Waar speelt het probleem zich af?
	  - Op de centrale server.

(M)eetbaar	- Hoe vaak komt het voor?
	  - Eenmaal.
	- Hoe weet ik dat het compleet is?
	  - Als de centrale server de juiste data opslaat op de log locatie.
	- Zijn de indicatoren telbaar?
	  - Nee.

(A)cceptabel	- Hoe kan ik dit behalen?
	  - Door middel van het ontwikkelen van een object die log functionaliteiten heeft.
	- Hoe realistisch is dit om te halen?
	  - Zeer realistisch.

(R)elevant	- Moet ik hier tijd aan besteden?
	  - Nee. Het is een optioneel onderdeel dat pas belangrijk wordt bij het productie klaar maken van de centrale server.
	- Is dit het juiste moment om op te lossen?
	  - Nee. De prioriteit is laag bij de concept versie.
	- Past dit bij onze doelen?
	  - Nee.

(T)ijd gebonden	- Wat is er te doen?
	  - Ontwikkelen van een object met log functionaliteiten.
	- Wanneer moet het af zijn?
	  - Tijdens productie klaar maken van de centrale server.

4 - Het prototype bevat autorisatie functionaliteiten op binnenkomende aanvragen van cliënten.	

(S)pecifiek	- Wat wil ik behalen?
	  - Binnenkomende verbindingen controleren of deze bevoegd zijn de functionaliteiten te gebruiken.
	- Waarom wil ik het behalen?
	  - Gebruikers kunnen beperken in de functionaliteiten die beschikbaar zijn.
	- Wie is betrokken?
	  - Bert Willems.
	- Waar speelt het probleem zich af?
	  - Op de centrale server.

(M)eetbaar	- Hoe vaak komt het voor?
	  - Iedere keer dat een cliënt probeert gebruik te maken van add-ons.
	- Hoe weet ik dat het compleet is?
	  - Een cliënt kan toegelaten of geweigerd worden om de functionaliteit uit te voeren.
	- Zijn de indicatoren telbaar?
	  - Nee.

(A)cceptabel	- Hoe kan ik dit behalen? - Door middel van het toevoegen controles op binnen komende aanvragen.

	- Hoe realistisch is dit om te halen? - Zeer realistisch.

(R)elevant	- Moet ik hier tijd aan besteden?
	  - Nee. De autorisatie valt buiten de scope.
	- Is dit het juiste moment om op te lossen?
	  - Nee.
	- Past dit bij onze doelen?
	  - Nee.

(T)ijd gebonden	- Wat is er te doen?
	  - Controle toevoegen op aanvragen van cliënten.
	- Wanneer moet het af zijn?
	  - Tijdens vervolg onderzoek.

5 - Het prototype bevat authenticatie functionaliteiten op binnenkomende verbindingen van cliënten.	

(S)pecifiek	- Wat wil ik behalen?
	  - De centrale server heeft een controle om te kijken of binnenkomende verbinding toestemming hebben om gebruik te maken van de centrale server.
	- Waarom wil ik het behalen?
	  - Voorkomen van kwaadwillende verbindingen.
	- Wie is betrokken?
	  - Bert Willems
	- Waar speelt het probleem zich af?
	  - Op de centrale server.

(M)eetbaar	- Hoe vaak komt het voor?
	  - Iedere keer als er een nieuwe verbinding opgezet wordt tussen de cliënt en de centrale server.
	- Hoe weet ik dat het compleet is?
	  - Verbindingen die niet toegestaan zijn worden geweigerd.
	- Zijn de indicatoren telbaar?
	  - Nee.

(A)cceptabel	- Hoe kan ik dit behalen?
	  - Door een cliënt te identificeren en op basis van deze identificatie de cliënt afwijzen of toelaten.
	- Hoe realistisch is dit om te halen?
	  - Zeer realistisch.

(R)elevant	- Moet ik hier tijd aan besteden?
	  - Nee. Authenticatie valt buiten de scope van dit project.
	- Is dit het juiste moment om op te lossen?
	  - Nee.
	- Past dit bij onze doelen?
	  - Nee.

(T)ijd gebonden	- Wat is er te doen?

	- Binnenkomende verbindingen identificeren en authentiseren. - Wanneer moet het af zijn? - Niet, valt buiten de scope.

6 - Het prototype kan met meerdere Web Services tegelijk communiceren.	

(S)pecifiek	- Wat wil ik behalen?
	  - De centrale server kan data sturen en ontvangen van een webservice die beheert wordt door een derde partij.
	- Waarom wil ik het behalen?
	  - Is kern van de functionaliteiten van de centrale server.
	- Wie is betrokken?
	  - Bert Willems.
	- Waar speelt het probleem zich af?
	  - De centrale server

(M)eetbaar	- Hoe vaak komt het voor?
	  - Iedere keer als vanaf de cliënt een add-on wordt aangeroepen.
	- Hoe weet ik dat het compleet is?
	  - De centrale server kan data opsturen naar webservice en ontvangt hier relevante data voor terug.
	- Zijn de indicatoren telbaar?
	  - Nee.

(A)cceptabel	- Hoe kan ik dit behalen?
	  - Door een succesvolle verbinding op te zetten tussen de centrale server en een webservice en vervolgens hier over data proberen te sturen.
	- Hoe realistisch is dit om te halen?
	  - Zeer realistisch. Deze functionaliteit wordt op het moment gebruikt door FontoXML zelf.

(R)elevant	- Moet ik hier tijd aan besteden?
	  - Ja. Is een kern functionaliteit van de centrale server.
	- Is dit het juiste moment om op te lossen?
	  - Ja.
	- Past dit bij onze doelen?
	  - Ja.

(T)ijd gebonden	- Wat is er te doen?
	  - Een webservice selecteren waarmee verbinding gemaakt zal worden. Het opzetten van de verbinding en het analyseren van de terug gekomen data.
	- Wanneer moet het af zijn?
	  - Eerste sprint.

7 - Het prototype kan asynchroon aanvragen van cliënten afhandelen.	

(S)pecifiek	- Wat wil ik behalen?
	  - De centrale server kan asynchroon meerdere aanvragen van cliënten afhandelen.
	- Waarom wil ik het behalen?
	  - kern functionaliteit van de centrale server.
	- Wie is betrokken?
	  - Bert Willems.
	- Waar speelt het probleem zich af?
	  - Centrale Server

(M)eetbaar	- Hoe vaak komt het voor?
	  - voor iedere cliënt die een aanvraag verstuurt.
	- Hoe weet ik dat het compleet is?
	  - De server kan aanvragen afhandelen en in willekeurige volgorde weer terug sturen naar de juiste cliënt.
	- Zijn de indicatoren telbaar?
	  - Nee.

(A)cceptabel	- Hoe kan ik dit behalen?
	  - Door asynchrone functionaliteiten te bouwen in de server.
	- Hoe realistisch is dit om te halen?
	  - Zeer realistisch.

(R)elevant	- Moet ik hier tijd aan besteden?
	  - Ja.
	- Is dit het juiste moment om op te lossen?
	  - Ja.
	- Past dit bij onze doelen?
	  - Ja.

(T)ijd gebonden	- Wat is er te doen?
	  - De server dient asynchroon webservices aan te kunnen roepen om data asynchroon terug te kunnen geven aan de cliënt.
	- Wanneer moet het af zijn?
	  - Tweede iteratie.

8 - De server omgeving kan binnenkomende aanvragen muteren en doorsturen naar web services.	

(S)pecifiek	- Wat wil ik behalen?
	  - De binnenkomende data moet gemuteerd worden zodat deze bruikbaar is door de webservice.
	- Waarom wil ik het behalen?
	  - Webservices kunnen niet omgaan met de data die door FontoXML verstuurd zal worden.
	- Wie is betrokken?
	  - Bert Willems.
	- Waar speelt het probleem zich af?

	- De centrale server.

(M)eetbaar	- Hoe vaak komt het voor?
	  - Iedere keer als er data naar een webservices wordt gestuurd.
	- Hoe weet ik dat het compleet is?
	  - De webservice kan de data lezen en geeft hier relevante informatie over.
	- Zijn de indicatoren telbaar?
	  - Nee.

(A)cceptabel	- Hoe kan ik dit behalen?
	  - De binnen gekomen data moet gelezen worden en de nuttige data moet worden opgestuurd.
	- Hoe realistisch is dit om te halen?
	  - Zeer Realistisch.

(R)elevant	- Moet ik hier tijd aan besteden?
	  - Ja.
	- Is dit het juiste moment om op te lossen?
	  - Ja.
	- Past dit bij onze doelen?
	  - Ja.

(T)ijd gebonden	- Wat is er te doen?
	  - De opgestuurd data moet uitgelezen worden. De nuttige data moet opgestuurd worden. Data waar de webservice niet mee om kan gaan moet eruit gehaald worden.
	- Wanneer moet het af zijn?
	  - Eerste iteratie.

9 - De server stuurt de data van de web service altijd op dezelfde manier terug naar de cliënt.	

(S)pecifiek	- Wat wil ik behalen?
	  - Een vorm van berichtgeving waarin de data staat die de centrale server wilt terugsturen.
	- Waarom wil ik het behalen?
	  - Eenvoud van implementatie bij de cliënt.
	- Wie is betrokken?
	  - Bert Willems
	- Waar speelt het probleem zich af?
	  - Centrale Server

(M)eetbaar	- Hoe vaak komt het voor?
	  - Iedere keer als de centrale server een object klaar heeft waar informatie van de webservice in staat.
	- Hoe weet ik dat het compleet is?
	  - De data die teruggestuurd wordt naar de cliënt heeft altijd dezelfde vorm.
	- Zijn de indicatoren telbaar?
	  - Nee.

(A)cceptabel	- Hoe kan ik dit behalen?
	  - Goed indexeren welke data de webservice proberen terug te sturen.
	- Hoe realistisch is dit om te halen?
	  - Gemiddelde. Door de verschillende webservices kan er altijd specifieke data zijn die niet aan de structuur voldoet.

(R)elevant	- Moet ik hier tijd aan besteden?
	  - Ja.
	- Is dit het juiste moment om op te lossen?
	  - Ja.
	- Past dit bij onze doelen?
	  - Ja.

(T)ijd gebonden	- Wat is er te doen?
	  - Kijken of het mogelijk is data van webservices in een consistente structuur terug te sturen naar de cliënt. Deze structuur ontwerpen en implementeren.
	- Wanneer moet het af zijn?
	  - Tweede iteratie.

10 - Ontvangen aanvragen moeten kunnen worden gestopt als de aanvragen niet meer geldig zijn door veranderingen bij de cliënt.	

(S)pecifiek	- Wat wil ik behalen?
	  - Requests die nog op de server bezig zijn moeten gestopt kunnen worden.
	- Waarom wil ik het behalen?
	  - Requests kunnen ongeldig worden door aanpassingen aan de cliënt kant.
	- Wie is betrokken?
	  - Bert Willems.
	- Waar speelt het probleem zich af?
	  - De cliënt kan doorwerken terwijl de server de aanvraag aan het afhandelen is. Hierdoor kan de data die terugkomt verouderd zijn en niet meer gelding.

(M)eetbaar	- Hoe vaak komt het voor?
	  - Iedere keer als er wijzigingen worden aangebracht in data die opgestuurd is naar de webservice.
	- Hoe weet ik dat het compleet is?
	  - Een request die niet meer geldig is kan stopgezet worden en de server zal dan geen data meer terug sturen naar de cliënt.
	- Zijn de indicatoren telbaar?
	  - Nee.

(A)cceptabel	- Hoe kan ik dit behalen? - Door een request te sturen naar de cliënt die een andere request stop kan zetten. - Hoe realistisch is dit om te halen?

	- Zweer realistisch.

(R)elevant	- Moet ik hier tijd aan besteden?
	  - Ja
	- Is dit het juiste moment om op te lossen?
	  - Ja
	- Past dit bij onze doelen?
	  - Ja.

(T)ijd gebonden	- Wat is er te doen?
	  - Een request stop baar maken.
	- Wanneer moet het af zijn?
	  - Tweede iteratie.

11 - De centrale server moet de data op de juist gemapte manier terug geven aan de cliënt.	

(S)pecifiek	- Wat wil ik behalen?
	  - De data die de webservice geeft moet begrijpelijk zijn voor FontoXML.
	- Waarom wil ik het behalen?
	  - Hieronder kan de data op de juiste plek terug worden gezet in de originele tekst.
	- Wie is betrokken?
	  - Bert Willems.
	- Waar speelt het probleem zich af?
	  - Centrale Server.

(M)eetbaar	- Hoe vaak komt het voor?
	  - Iedere keer al een webservice data terug geeft.
	- Hoe weet ik dat het compleet is?
	  - De server geeft de data van de webservice op een begrijpbare manier terug.
	- Zijn de indicatoren telbaar?
	  - Nee.

(A)cceptabel	- Hoe kan ik dit behalen?
	  - Door het ontwikkelen van een map functie.
	- Hoe realistisch is dit om te halen?
	  - Zeer realistisch.

(R)elevant	- Moet ik hier tijd aan besteden?
	  - Ja.
	- Is dit het juiste moment om op te lossen?
	  - Ja.
	- Past dit bij onze doelen?
	  - Ja.

(T)ijd gebonden	- Wat is er te doen? - Ontwikkelen van mapping functionaliteiten. - Wanneer moet het af zijn?

	- Tweede iteratie..

12 - Requests die naar de centrale server worden gestuurd moeten bijgehouden worden.	

(S)pecifiek	- Wat wil ik behalen?
	  - Binnenkomende requests moeten bijgehouden worden.
	- Waarom wil ik het behalen?
	  - Aan de hand van de lijst kunnen requests gecanceld worden en gecontroleerd worden of ze al klaar zijn.
	- Wie is betrokken?
	  - Bert Willems.
	- Waar speelt het probleem zich af?
	  - De centrale server.

(M)eetbaar	- Hoe vaak komt het voor?
	  - Iedere keer als er een request naar de centrale server wordt gestuurd..
	- Hoe weet ik dat het compleet is?
	  - Onafgeronde requests zijn te vinden in een lijst op de centrale server.
	- Zijn de indicatoren telbaar?
	  - Ja.

(A)cceptabel	- Hoe kan ik dit behalen?
	  - Ieder binnenkomende request wordt opgeslagen in een lijst.
	- Hoe realistisch is dit om te halen?
	  - Zeer realistisch.

(R)elevant	- Moet ik hier tijd aan besteden?
	  - Ja.
	- Is dit het juiste moment om op te lossen?
	  - Ja.
	- Past dit bij onze doelen?
	  - Ja.

(T)ijd gebonden	- Wat is er te doen?
	  - Ontwerpen van een lijst waarin de requests kunnen komen te staan.
	- Wanneer moet het af zijn?
	  - Tweede iteratie.

13 - De centrale server ondersteunt meerdere cliënten tegelijkertijd.	
(S)pecifiek	- Wat wil ik behalen? - Er kunnen meerdere cliënten tegelijker tijd communiceren met de centrale server. - Waarom wil ik het behalen?

	- Huidige situatie maakt gebruik van meerdere cliënten.
	- Wie is betrokken?
	  - Bert Willems.
	- Waar speelt het probleem zich af?
	  - Centrale Server.

(M)eetbaar	- Hoe vaak komt het voor?
	  - Eenmaal.
	- Hoe weet ik dat het compleet is?
	  - De centrale server ondersteund meerdere cliënten die verbinding maken.
	- Zijn de indicatoren telbaar?
	  - Ja.

(A)cceptabel	- Hoe kan ik dit behalen?
	  - Door de centrale server nieuwe cliënten asynchroon af te laten handelen.
	- Hoe realistisch is dit om te halen?
	  - Realistisch.

(R)elevant	- Moet ik hier tijd aan besteden?
	  - Ja.
	- Is dit het juiste moment om op te lossen?
	  - Ja.
	- Past dit bij onze doelen?
	  - Ja.

(T)ijd gebonden	- Wat is er te doen?
	  - De server moet asynchroon nieuwe cliënten af kunnen handelen.
	- Wanneer moet het af zijn?
	  - Tweede iteratie.

14 - De centrale server kan asynchroon responses terug sturen naar de cliënt.	

(S)pecifiek	- Wat wil ik behalen?
	  - Responses worden teruggestuurd naar de cliënt wanneer deze beschikbaar zijn.
	- Waarom wil ik het behalen?
	  - Hierdoor zullen responses eerder al terug komen in plaats van dat deze moet wachten totdat alles is afgerond.
	- Wie is betrokken?
	  - Bert Willems.
	- Waar speelt het probleem zich af?
	  - Op de centrale server.

(M)eetbaar	- Hoe vaak komt het voor?
	  - Iedere keer als er een result klaar is om teruggestuurd te worden.
	- Hoe weet ik dat het compleet is?
	  - Er komen results terug voordat de gehele request is afgehandeld.

	- Zijn de indicatoren telbaar? - Ja.

(A)cceptabel	- Hoe kan ik dit behalen?
	  - Door het ophalen van de data en het verwerken hiervan in een aparte thread te doen.
	- Hoe realistisch is dit om te halen?
	  - Realistisch.

(R)elevant	- Moet ik hier tijd aan besteden?
	  - Ja.
	- Is dit het juiste moment om op te lossen?
	  - Ja.
	- Past dit bij onze doelen?
	  - Ja.

(T)ijd gebonden	- Wat is er te doen?
	  - Een thread ontwikkelen die de data kan muteren en terug kan geven.
	- Wanneer moet het af zijn?
	  - Tweede iteratie.

5. Afronding requirements

In dit hoofdstuk worden alle requirements beschreven of deze gehaald zijn. Als ze gehaald zijn wordt er beschreven hoe deze zijn behaald, als deze niet zijn behaald wordt beschreven waarom ze niet zijn behaald.

1 – De cliënt kan vanuit de browser via de centrale server communiceren met de webservices.

Deze requirement is behaald en is geïmplementeerd in de huidige server structuur. Vanwege de hoge prioriteit is deze functionaliteit sinds de eerste versie van de centrale server geïmplementeerd. De requirement is bewezen door na de bouw vanaf de cliënt data te sturen naar de server. De server geeft over deze data informatie die vanuit een webservice ontvangen wordt. Of de ontvangen data bruikbaar is voor de cliënt is voor deze requirement niet belangrijk.

2 – De server moet eenvoudig uit te breiden zijn voor nieuwe webservices.

Deze requirement heeft betrekking op nieuwe webservices die geïmplementeerd moeten worden om de functionaliteiten van de webservice uit te breiden. Toevoegingen voor webservices hebben in de huidige implementatie hoge koppeling met een interface klasse. De interface klasse verplicht nieuwe webservices twee functies te implementeren. Voor iedere webservice moet er mapping functionaliteiten toegevoegd worden. Afhangend van de complexiteit van de webservice zal deze de meeste tijd opnemen om een nieuwe webservice toe te voegen. Elke webservice klasse heeft een list met daarin alle opgestuurd elementen beschikbaar. Op deze list kan de data van de webservice worden gemapt.

Per webservice moeten er nog objecten aangemaakt worden die erven van het result object. In deze objecten komen alle vormen van interessante data die de webservice geeft.

Het werk dat verricht moet worden om een nieuwe service toe te voegen is niet eenvoudig door de verplichting van het zelf ontwikkelen van de mapping van de webservice data. Naast de mapping is de overige implementatie eenvoudig en hoeft er nauwelijks rekening gehouden te worden met andere klassen.

Bij de front-end dient wel rekening gehouden te worden met de nieuwe result objecten. Bij de front-end moet iedere nieuwe vorm van result een visuele reactie hebben. Zoals bij spellingsfouten altijd rode lijntjes onder het verkeerd gespelde woord worden getoond.

3 – Het prototype bevat Log functionaliteiten.

Deze requirement is niet uitgevoerd. Aan het begin van het project is duidelijk gemaakt wat het nut is van deze functionaliteiten. Voor het proof of concept zijn deze functionaliteiten niet van belang. Om deze reden is deze requirement in overleg met de bedrijfsmentor buiten de scope geplaatst.

4 – Het prototype bevat autorisatie functionaliteiten op binnen komende aanvragen van cliënten.

Deze requirement heeft invloed gehad op de structuur van de centrale server. Het kunnen controleren van verbindingen is een belangrijk onderdeel voor een productie klare server. Wegens de invloed op de structuur die deze requirement heeft is er in de centrale server een connection hub toegevoegd. Deze klasse heeft als verantwoordelijkheid om de binnenkomende verbindingen te

controleren. Wegens de lage prioritering zijn deze functionaliteiten niet gebouwd. In de huidige situatie functioneert deze klasse niet anders dan een doorgeef luik van requests.

5 – Het prototype bevat authenticatie functionaliteiten op binnenkomende verbindingen van cliënten.

Zie requirement 4.

6 – Het prototype kan met meerdere webservices tegelijk communiceren.

Om alle add-ons te kunnen ondersteunen waarvan FontoXML gebruik maakt moet de centrale server ook met meerdere webservices kunnen communiceren. Deze communicatie moet gelijktijdig gebeuren om te voorkomen dat webservices niet elkaar gaan blokkeren. Deze requirement is in de centrale server gebouwd door middel van het gebruik van threads. Voor iedere webservice die wordt gebruikt wordt een nieuwe thread gestart. Door het gebruik van threads kan iedere webservice aangeroepen worden en kan de data teruggestuurd worden wanneer deze beschikbaar is. Zonder dat een andere webservice hier invloed op kan hebben. De requirement is bewezen doordat de cliënt nu data terug krijgt op willekeurige orde.

7 – Het prototype kan asynchroon aanvragen van cliënten afhandelen.

Een cliënt kan meerdere requests sturen en wilt per request antwoord ontvangen als deze klaar is. De ene request is groter dan de ander. Het is niet van belang in welke volgorde de requests terugkomen naar de cliënt. Om te voorkomen dat de grote requests blokkerend werken voor latere requests wordt ieder request afgehandeld in een eigen thread. Deze requirement heeft invloed gehad op requirement 12. Welke invloed dat is geweest zal bij requirement 12 te lezen zijn. Deze requirement is bewezen doordat de cliënt nu data kan terug krijgen van een request die als tweede is gestuurd voordat de eerste request is afgerond.

8 – De server omgeving kan binnenkomende aanvragen muteren en doorsturen naar web services.

De manier waarop de cliënt data stuurt naar de server kan niet direct gebruikt worden om de data naar de webservices te sturen. Iedere webservice heeft hiervoor zijn eigen requirements. De server moet deze data omzetten zodat de webservices deze kunnen gebruiken. De binnenkomende data wordt gelezen en in een list gezet. Deze list wordt gebruikt om requirement twee te ondersteunen. Iedere webservice adapter krijgt een list met dezelfde structuur. Deze list wordt niet alleen gebruikt om de data van de webservice te kunnen mappen maar ook om de gestuurde data in een structuur te zetten die gebruikt kan worden door de webservice. Deze requirement is bewezen omdat, er een list komt waarin alle elementen komen te staan met de bijbehorende tekst inhoud. Deze list wordt door iedere adapter gebruikt om de data in de juiste structuur te zetten en naar de webservice te sturen. De webservice geeft na deze transformatie geen fout meldingen en gebruikt de data om de controle uit te voeren.

9 – De server stuurt de data van de webservice altijd op dezelfde manier terug naar de cliënt.

Deze requirement is niet volledig geïmplementeerd. Aan het begin van het project is aangegeven om een vorm van communicatie op te zetten tussen de cliënt en de server. Dit om implementatie aan de front-end kant te vereenvoudigen. Tijdens de ontwikkeling van de webservices bleek de data van

deze webservice grote verschillen te hebben. Hierdoor bleek een vorm van communicatie onmogelijk. Om alsnog zo dicht mogelijk bij het voltooien van de requirement te komen bestaat elk result uit een erving van een klasse. Door deze overerving zal ieder result altijd over dezelfde vorm van mapping beschikken. De overige data die vanuit de webservice wordt geleverd zal per result afwijkend kunnen zijn.

10 – Ontvangen aanvragen moeten kunnen worden gestopt als de aanvragen niet meer geldig zijn door veranderingen bij de cliënt.

Een van de grotere onderdelen van FontoXML is dat alles real-time gebeurd. Hierdoor kan de gebruiker nog steeds verder werken ook al zijn er controles die worden uitgevoerd door de server. Hierdoor kan het voorkomen dat data die is opgestuurd veranderd voordat de controle is afgerond. Om te voorkomen dat er data terugkomt die niet meer bestaat moet een request te stoppen zijn. Deze requirement is geïmplementeerd en is beïnvloed door requirement 7. Doordat ieder request in een eigen thread draait kan het cancel-request niet direct de andere requests beïnvloeden. Om een request dus te kunnen cancelen moet er een mogelijkheid zijn om threads elkaar te laten beïnvloeden. Hiervoor is de oplossing van requirement 12 gebruikt. De oplossing voor requirement 12 creëert een lijst die alle requests bijhoudt. Deze lijst is thread-safe gemaakt en kan aangepast worden door alle threads. Nu kan een cancel request bij de standaard request en kan deze stopgezet worden. Om een request stop te kunnen zetten wordt door de cancel request in de te cancelen request de status van de request op cancel gezet. Omdat een request niet 100% van de tijd in het beheer is van de centrale server (de webservice beheert het tijdens het uitvoeren van de controle) kan een request niet verwijderd worden. Zodra de webservice data terug geeft wordt gecontroleerd of deze request nog geldig is. Als hij dat niet is wordt er niks meer uitgevoerd en wordt de request uit de lijst verwijderd. Zodra dit is gebeurd wordt de cliënt ingelicht dat de request verwijderd is en dat de cliënt hier ook niks meer voor hoeft te onthouden. Er is getest of na het sturen van een request deze gecanceld kan worden en of dit ook mogelijk is als er resultaten van de request terug zijn gekomen. In beide gevallen stopt de server met het terug sturen van results van de request en krijgt de cliënt bericht dat de request is gecanceld.

11 – De centrale server moet de data op de juiste gemapte manier terug geven aan de cliënt.

De data die wordt gegeven door de webservices is niet direct toepasbaar door de cliënt. Hiervoor moet de data omgezet worden naar iets wat de cliënt kan begrijpen. Zoals in requirement negen is beschreven is geprobeerd om hier een vorm van te maken. Dit is tijdens de ontwikkeling duidelijk geworden dat dit geen mogelijkheid is. Iedere webservice geeft verschillende data terug. Ondanks de verschillen is de mapping wel hetzelfde voor iedere webservice. De server krijgt geen informatie over de structuur van het gehele document. Alleen over de structuur van het gedeelte dat is opgestuurd. De cliënt dient bij te houden welk onderdeel is opgestuurd zodat dit later gebruikt kan worden de informatie op de juiste positie te zetten. Samen met de data die van uit de webservice wordt teruggestuurd kan de data op de juiste positie terug gezet worden.

12 – Requests die naar de centrale server worden gestuurd moeten bijgehouden worden.

Het is van belang om aan de cliënt te laten weten wanneer een request is afgerond zodat deze de request aan de cliënt kant kan opruimen. Iedere request wordt naar meerdere webservices gestuurd. Iedere webservice krijgt zijn eigen thread waarin de webservice wordt afgehandeld. Als

alle webservices klaar zijn moet er een bericht naar de cliënt gestuurd worden. Omdat iedere webservice een eigen thread heeft weten deze niet van elkaar wanneer de ander klaar is. Om deze reden wordt de request bijgehouden. Iedere request weet wel wanneer een webservice klaar is. De request weet ook wanneer alle webservices afgerond zijn. Om requirement tien te kunnen ondersteunen worden alle requests bijgehouden in een static dictionary. Een static dictionary is thread-safe en kan door meerdere threads gelezen en aangepast worden. Via deze list moeten alle requests beschikbaar worden zodat deze stop gezet kunnen worden vanuit een andere thread. In de huidige centrale server kan via een cancel request een van de vorige requests van de cliënt stopgezet worden. Als de request al bij de webservices ligt zal er gewacht worden tot deze data teruggeeft. De teruggeven data wordt niet afgehandeld en de service wordt verwijderd uit de request. Als de request geen services meer heeft wordt ook deze verwijderd en dit doorgegeven aan de cliënt.

13 – de centrale server ondersteunt meerdere cliënten tegelijkertijd.

Een belangrijk onderdeel van de server is dat deze meerdere cliënten tegelijkertijd kan afhandelen. De server doet dit met behulp van de gekozen websocket library. De library kan voor iedere cliënt binnen komende request asynchroon afhandelen. Dit is getest door meerdere cliënten te openen en via deze cliënten requests te sturen. Iedere cliënt kreeg iets terug wanneer de request was afgerond. Onafhankelijk van welke cliënt als eerste een request had gestuurd.

14 – De centrale server kan asynchroon responses terug sturen naar de cliënt.

Deze requirement is al gedeeltelijk beschreven door requirement 13. Het belang van deze requirement is dat de server data terugstuurt wanneer deze afgerond is. Hiervoor hoeft een webservice of request nog niet volledig klaar te zijn. De cliënt ontvangt data wanneer er iets beschikbaar is om terug te sturen door de server.

Bijlage #6: Test omschrijving

1. Inleiding

In dit document staat de uitgevoerde vorm van testen beschreven. Aan het eind van dit document moet duidelijk zijn welke test is uitgevoerd en waarom voor deze vorm van testen is gekozen.

1.1 Codereview

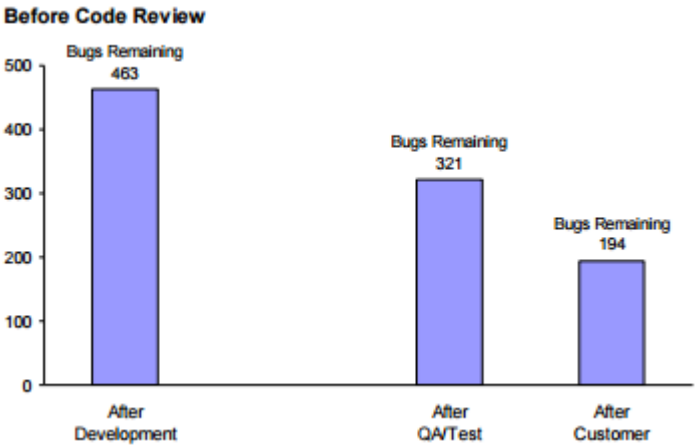
Er is gekozen om een codereview uit te voeren over de geschreven code. In het volgende hoofdstuk wordt beschreven waarom er voor deze vorm is gekozen en hoe de codereview exact vorm heeft gekregen.

1.2 Gekozen methode

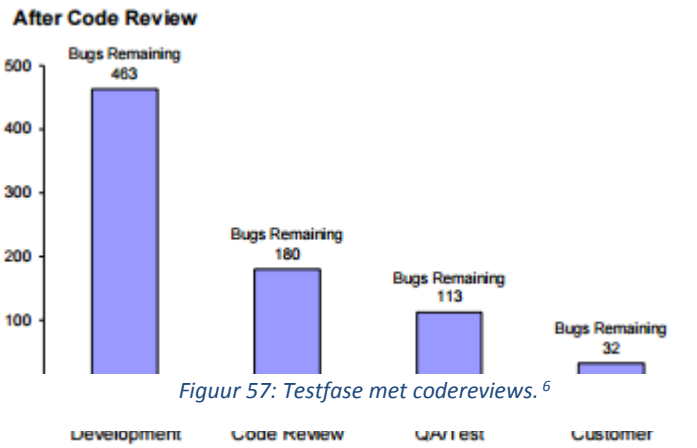
Codereview is een eenvoudige en effectieve manier om code te kunnen testen. Tijdens codereviews worden fouten of wijzigingen gevonden in de code. Deze fout of wijziging wordt een defect genoemd. Een defect is als volgt gedefinieerd:

> *"When a reviewer or consensus of reviewers determines that code must be changed before it is acceptable, it is a "defect"."*[4]

Bij het testen is het belangrijk dat er zo minmogelijk defects door klanten worden gevonden. In figuur 52 is te zien dat zonder codereview er 40% van de defects bij klanten terecht komen. In figuur 53, waar codereviews wel worden toegepast, is te zien dat er maar 6% van de defects bij klanten terecht komen. Door de codereviews worden 283 defects gevonden. Van deze 283 defects zijn er 208 defects die ook zonder codereview gevonden hadden worden gevonden. Dankzij codereviews worden er 75 defects gevonden die niet worden afgevangen in de overige testcases.



Figuur 56: Testfase zonder codereviews.[5]



Figuur 57: Testfase met codereviews.[6]

Uit deze twee figuren is duidelijk gemaakt dat codereview een groot effect heeft op het aantal defects dat in software blijft hangen als deze naar de klant wordt gestuurd.

Codereviews kan dus een grote impact hebben op de hoeveelheid defects. Om deze reden heb ik gekozen om deze vorm van testen toe te passen op mijn geschreven code. Na dit besluit ben ik dieper ingegaan op codereviews om hier vorm aan te geven.

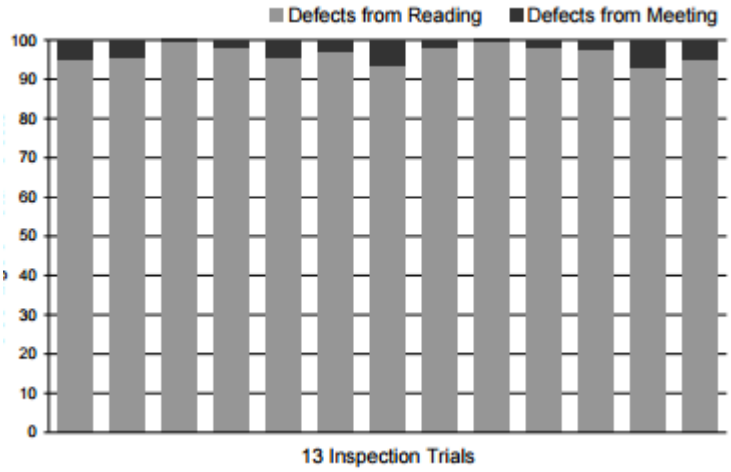
[4],[5],[6] (Cohen, Teleki, & Brown, 2013)

1.3 Light-weight codereview tegenover formal codereview

Codereviews kunnen op twee niveaus uitgevoerd worden. Light-weight codereview en formal codereview. In de light-weight vorm van codereview zijn de eigenaar van de code en een tot twee ontwikkelaars aanwezig die de code doorlopen om fouten te vinden. Bij de light-weight vorm wordt vaak op korte termijn een afspraak gemaakt met de betrokken ontwikkelaars. De ontwikkelaars nemen dan de code, die gecontroleerd moet worden, door en geven feedback aan de ontwikkelaar die de code heeft geschreven. Alle communicatie is informeel en wijzigingen die worden doorgevoerd worden vaak niet officieel bijgehouden. Het voordeel aan light-weight codereview is dat deze snel in te plannen is en dat er maar een beperkt aantal mensen bij betrokken zijn. Wijzigingen die worden aangegeven worden vaak tijdens de codereview aangepast tenzij de wijziging van groter formaat is. Als alle aangegeven fouten uit de applicatie zijn gehaald kan doorgegaan worden met de applicatie.

De formal vorm van codereview heeft meer planning nodig dan de light-weight vorm. Bij een formal codereview worden vaak drie tot zes ontwikkelaars gevraagd om de codereview uit te voeren. Hiervoor wordt een meeting ingepland en in de meeting wordt de codereview uitgevoerd. Tijdens de meeting zijn vaak stukken code op papier beschikbaar. In de meeting wordt ieder defect officieel beschreven en wordt pas na de meeting opgepakt om aan te passen.

Vanwege de beperkte tijd die mijn collega's bij FontoXML hebben ben ik geforceerd om voor de light-weight codereview te kiezen. In hoeverre beïnvloed dit het aantal defects dat zal gevonden worden? Uit het volgende onderzoek waar light-weight en formal codereviews met elkaar vergeleken worden blijkt het volgende:



Figuur 58: Defects van lezen tegenover defects door meetings.

Uit deze tabel is te zien welke defects gevonden worden tijdens de light-weight codereview en welke tijden de formal codereview. Dit verschil is ~4%. Hieruit kan men zich afvragen of de formal vorm van codereview de toegevoegde waarde heeft die de moeite voor het opzetten doet vermoeden. Ondanks dat ik al geforceerd was om light-weight codereviews uit te voeren is dit geen probleem voor het aantal defects dat hieruit zal komen. De light-weight codereview zal vergelijkbare resultaten opleveren als de formal codereview. Nu ik hier voor mezelf duidelijkheid over heb gecreëerd ben ik gaan kijken hoe ik de light-weight codereview kan uitvoeren.

1.3.1 Over-the-shoulder codereview.

Er zijn een viertal vormen waarop de light-weight codereview uitgevoerd kan worden.

- Over-the-shoulder codereview.
- E-mail pass-around.
- Pair-programming.
- Tool-assisted reviews.

Voor over-the-shoulder codereview zal de programmeur die de code heeft ontwikkeld samen met een andere programmeur door de code gaan. De programmeur legt de code uit aan de reviewer. De reviewer heeft hier de kans om defects te melden. Als een defect is gemeld kan deze direct aangepast worden in de code en gecontroleerd worden door de reviewer. Deze vorm is handig om te gebruiken bij nieuw ontwikkelde code omdat er direct grote delen van code gecontroleerd kan worden.

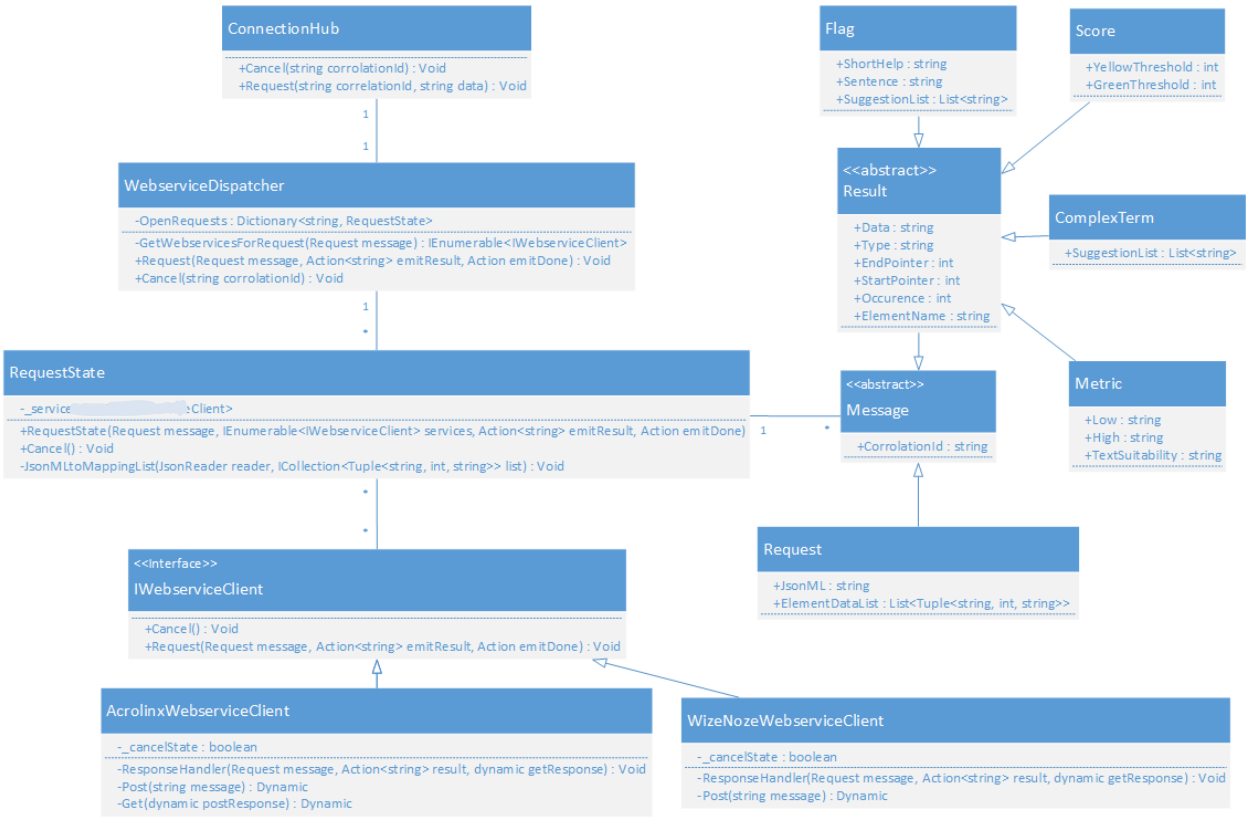
Voor E-mail pass-around wordt de code die gereviewed moet worden rond gestuurd naar alle betrokkenen binnen het team. Het team kan dan op momenten waar het uitkomt de code reviewen en terug sturen naar de ontwikkelaar. Deze vorm is handig om te gebruiken als verbeteringen in de code zijn doorgevoerd. Deze verbeteringen zijn vaak kleine stukken code en zijn daarom snel te begrijpen en te controleren.

Bij pair-programming wordt een werkstation gebruikt door twee ontwikkelaars om code te ontwikkelen. Ik heb mij zelf niet verdiept in de vorm omdat alle medewerkers hun eigen taken hebben en niet de tijd om 15 weken te besteden aan pair-programming.

De tool-assisted reviews maakt gebruik van tools die de controle uitvoeren. Voor iedere commit worden standaard controles uitgevoerd om de code te controleren. Zo worden veel standaard programmeerfouten ontdekt tijdens het commiten van de code. Als de code foutloos is kan de commit pas doorgezet worden. Hiervoor dient wel het systeem opgezet te worden voor aan het werk kan gaan en dit kost veel tijd. Deze tijd die er in gestoken wordt kan op de lange termijn terug gewonnen worden maar niet in een periode van 15weken.

Na het bekijken van de mogelijkheden zijn er twee over gebleven die ik kan gebruiken. Het zou of over-the-shoulder codereview worden of e-mail pass around. ik heb gekozen voor de lichte code review over the-shoulder-review. E-mail pass-around is vooral nuttig voor kleine verbeteringen die snel te snappen zijn. Ik ben bezig met een hele nieuwe applicatie en daardoor zal er vaak veel code worden toegevoegd.

Bijlage #7: Beschrijving centrale Server



Figuur 59: Klasse Diagram centrale server final versie.

ConnectionHub

De ConnectionHub is een Hub voor alle Connecties. De ConnectionHub erft van de Hub klasse die vanuit de Microsoft.AspNet.SignalR library wordt geladen. Deze library maakt het mogelijk om de websocket connecties op te zetten. Door de erving worden gemaakte connecties bijgehouden en kan de ConnectionHub responses sturen naar connecties.

+Request(string correlationId, string data) : Void

Met de request functie worden requests van cliënten afgehandeld door de server. Voor ieder binnenkomend bericht maakt de functie een nieuwe Request aan met de correlationId en de data. De correlationId wordt door zowel de server als de cliënt bijgehouden. Dit is nodig om de cliënt de mogelijkheid te geven om uitstaande requests te cancelen als deze door veranderingen in de tekst niet meer valide zijn. De data bevat een JsonML object waarin de XML structuur en de data in staat. Er is gekozen voor JsonML omdat dit de lichtste manier is van het versturen van XML structuren. Om deze reden is de data variabele een string geworden. Er is gekozen voor het versturen van XML structuren om zo uitgebreide functionaliteiten uit te kunnen voeren. Functionaliteiten zoals, bijvoorbeeld, het automatisch kunnen samen vatten van teksten door web services. De functie maakt van de binnenkomende gegevens een Request object die samen met twee Actions naar de WebserviceDispatcher wordt gestuurd. De eerste Action is bedoeld om Results van de request terug te kunnen sturen naar de Cliënt en de tweede Action is bedoeld om aan de

cliënt te laten weten dat de request helemaal is afgehandeld en dus niet meer onthouden hoeft te worden aan de cliënt kant. Deze variabele worden verder uitgelegd bij de WebserviceDispatcher.

+Cancel(string CorrolationId): Void

Met de cancel functie worden cancel requests van cliënten afgehandeld door de server. Voor ieder binnenkomend cancel request wordt aan de WebserviceDispatcher de correlationId doorgegeven zodat deze de request kan cancelen als deze nog open staat.

WebserviceDispatcher

De webserviceDispatcher ontvangt van de connectionHub een Request object en twee Actions. Aan de hand van het ontvangen request object moet de WebserviceDispatcher de juiste webservices kunnen selecteren. De dispatcher start de webservices en houdt bij welke services nog bezig zijn voor welke request. Hiervoor wordt een Dictionary gebruikt. In de Dictionary staat per request de CorrolationId opgeslagen samen met een RequestState object. De inhoud van een requestState object wordt later uitgelegd.

+Request(Request message, Action<string> emitResult, Action emitDone) : Void

Voor ieder binnenkomende request bij de WebserviceDispatcher worden de bijbehorende webservices geladen. Met bijbehorend wordt bedoeld dat de WebserviceDispatcher de request bekijkt en de juiste webservices laadt. Voor een request met Engelse inhoud wordt bijvoorbeeld geen Nederlandse spellingchecker geladen. Als de webservices geladen zijn wordt een requestState object aangemaakt met de services en de emitResult action. Voor de emitDone wordt een nieuw Action aangemaakt bij de requestState. Deze action haalt bij aanroepen de requestState uit de Dictionary en roept vervolgens de emitDone action aan. Het requestState object wordt, samen met de CorrolationId, opgeslagen in de Dictionary. Aan de hand van het CorrolationId kan de request uit de Dictionary worden gehaald en gecanceld worden.

+Cancel(string corrolationId) : Void

De cancel functie controleert of de request horende bij de corrolationId nog in de Dictionary beschikbaar is. Als deze in de dictionary beschikbaar is wordt van de bijbehorende RequestState de Cancel functie aangeroepen en wordt de RequestState verwijderd uit de Dictionary.

-GetWebservicesForRequest(Request message) : IEnumerable<IWebserviceClient>

De GetWebservicesForRequest functie laadt alle webservices die bruikbaar zijn voor het verstuurde request. De functie controleert de inhoud van de request op bijvoorbeeld Taal om te voorkomen dat webservices zonder ondersteuning van de taal worden geladen. Alle geladen IWebserviceClient objecten worden in een List opgeslagen en teruggegeven aan de bovenliggende functie.

RequestState(Request message, IEnumerable<IWebserviceClient> services, Action<string> emitResult, Action emitDone)

Het RequestState object stelt een request voor die alle geladen webservices bevat. Aan de hand van deze geladen webservices kan de RequestState nieuwe threads starten om iedere webservice asynchroon af te handelen. Het is een requirement om de server results terug te geven wanneer deze beschikbaar zijn en het is een requirement om dit asynchroon te laten doen. Dit is als een requirement opgesteld omdat webservices langere tijd kunnen doen over een request en hierdoor de overige functionaliteiten van de server kunnen blokkeren. De server moet open blijven om nieuwe requests te kunnen ontvangen en afhandelen ook al is de vorige request nog niet afgehandeld. Deze requirements worden in de requestState door de asynchrone functionaliteiten afgevangen. Bij het aanroepen van de RequestState wordt de data van de request omgevormd. De data is op dit moment nog steeds in de vorm van JsonML. JsonML kan niet gebruikt worden door de webservices en moet daarom gemuteerd worden. De mutatie wordt verder uitgelegd in de JsonMLtoMappingList functie.

+Cancel() : Void

Als de cancel functie van de StateRequest wordt aangeroepen wordt voor ieder geladen webservice de cancel functie van die webservice aangeroepen.

-JsonMLtoMappingList(JsonReader reader, ICollection<Tuple<string, int, string>> list) : Void

Deze functie muteert de JsonML van de request naar een list. In de list wordt een Tuple opgeslagen met twee strings en een integer. De eerste string bevat de elementnaam van het XML object in de JsonML. De integer geeft aan om de hoeveelste voorkoming van dit element het gaat in de JsonML. De tweede string en de laatste variabele heeft de daadwerkelijke data die gecontroleerd moet worden door de webservice. Er is gekozen voor deze structuur omdat de data die door de webservices teruggeven wordt niet direct toepasbaar is op de data die de cliënt heeft staan. De data van de webservice moet gekoppeld worden aan de data die de cliënt heeft staan. Omdat de cliëntkant gebruik maakt van XML kan de data uitgelezen worden door een elementname en het toevoegen van de zoveelste occurence van dat element. Omdat de opgestuurde data vaak maar een deel is van de het gehele object moet aan de cliëntkant bijgehouden worden om welk deel het gaat. Deze list wordt later gebruikt om de data vanuit de webservice te kunnen koppelen.

AcrolinxWebserviceClient: IWebserviceClient

Deze klasse wordt gebruikt om contact te leggen met de Acrolinx Webservice. Het heeft om deze reden adapter functionaliteiten.

+Cancel(): Void

De cancel functie in de AcrolinxWebserviceClient zet de _cancelState van het object op false. Als de variabele op false is gezet worden er geen verdere functionaliteiten uitgevoerd.

+Request(Request message, Action<string> result, Action done): Void

De request functie krijgt een Request object binnen waarin de list met de gemuteerde data staat. Aan de hand van deze data zal de functie een string maken die te gebruiken is door de Acrolinx Webservice. Als de data begrijpelijk is gemaakt voor de webservice wordt deze opgestuurd via de Post functie. Deze functie wordt later uitgelegd. Als de data is opgestuurd wordt deze opgehaald door de Get functie. Deze functie wordt later uitgelegd. Als alle data is opgehaald wordt deze door de ResponseHandler gemuteerd en als results teruggestuurd naar de cliënt. Als deze functie afgerond is wordt de done action aangeroepen en is de request compleet afgerond voor deze webservice.

-Post(string message) : Dynamic

In de post functie staat de logica om de data te kunnen versturen naar de webservice van Acrolinx. Dit wordt gedaan door een REST call met de juiste headers en de body waarin de te controleren data staat. De headers zijn op dit moment hardcoded toegevoegd. Bij doorontwikkeling moet er een database ontwikkeld worden om deze hardcoded data te kunnen verwijderen. De response van de REST call is een JSON object waarin een checkId en een geschatte tijd van afronding in staat. De geschatte tijd wordt niet gebruikt omdat deze aan de kant van Acrolinx nog niet functioneert. De checkId wordt gebruikt om de data met de get functie weer op te halen.

-Get(dynamic responseBody) : Dynamic

In de responseBody staat een JSON object die door de post functie is ontvangen van de webservice. Zoals bij de post functie omschreven staat in het object een geschatte tijd waar, op dit moment, geen gebruik van gemaakt kan worden. Om deze reden wordt door de server long-polling toegepast om de data op te kunnen halen. Aan de hand van een rest call en de checkId uit de responseBody wordt een REST call gemaakt naar de webservice. Als de webservice nog niet de volledige controle heeft uitgevoerd zal de response leeg zijn. Zolang deze leeg is zal de functie iedere seconde opnieuw controleren of er een response beschikbaar is. Als er wel een response beschikbaar wordt deze opgehaald en naar de ResponseHandler functie gestuurd.

-ResponseHandler(Request message, Action<string> result, dynamic getResponse) : Void

De responseHandler maakt gebruik van de originele request en de getResponse om de data te kunnen koppelen zodat de Cliënt deze op juiste plek kan afhandelen. Als er een result object klaar is wordt deze via de Action terug gestuurd naar de cliënt. Voor de complete manier waarop de mapping functie wordt uitgevoerd wordt verwezen naar het document over de mapping. Als er een element is gemapt door de server wordt er hiervoor een object aangemaakt dat teruggestuurd zal worden naar de Cliënt. Voor Acrolinx is dit of een Flag object of een Score object.

WizeNozeWebserviceClient : IWebserviceClient

Deze klasse wordt gebruikt om contact te leggen met de WizeNoze Webservice. Het heeft om deze reden adapter functionaliteiten.

+Cancel() : Void
De cancel functie in de WizeNozeWebserviceClient zet de _cancelState van het object op false. Als de variabele op false is gezet worden er geen verdere functionaliteiten uitgevoerd.

+Request(Request message, Action<string> emitResult, Action emitDone) : Void
De request functie krijgt een Request object binnen waarin de list met de gemuteerde data staat. Aan de hand van deze data zal de functie een string maken die te gebruiken is door de WizeNoze Webservice. Als de data begrijpelijk is gemaakt voor de webservice wordt deze opgestuurd via de Post functie. Deze functie wordt later uitgelegd. De post functie geeft alle data door aan de ResponseHandler functie. De responseHandler muteert alle data en als er results beschikbaar zijn worden deze terug gestuurd naar de cliënt. Als deze functie afgerond is wordt de done action aangeroepen en is de request compleet afgerond voor deze webservice.

-Post(string message) : dynamic

In de post functie staat de logica om de data te kunnen versturen naar de webservice van WizeNoze. Dit wordt gedaan door een REST call met de juiste headers en de body waarin de te controleren data staat. De headers zijn op dit moment hardcoded toegevoegd. Bij door ontwikkeling moet er een database ontwikkeld worden om deze hardcoded data te verwijderen. De response van de REST call is een JSON object waarin de data staat over de uitgevoerde check van WizeNoze. Deze data wordt door gestuurd naar de ResponseHandler functie.

-ResponseHandler(Request message, Action<string> result, dynamic response) : Void

De responseHandler maakt gebruik van de originele request en de postResponse om de data te kunnen koppelen zodat de Cliënt deze op juiste plek kan afhandelen. Als er een result object klaar is wordt deze via de Action result teruggestuurd naar de cliënt. Voor de complete manier waarop de mapping functie wordt uitgevoerd wordt verwezen naar het document over de mapping. Als er een element is gemapt door de server wordt er hiervoor een object aangemaakt dat teruggestuurd zal worden naar de Cliënt. Voor WizeNoze is dit of een metric object of een complexTerm object.